PCA: THEORETICAL AND COMPUTATIONAL CONSIDERATIONS

LUCAS TUCKER

Abstract. This paper traces PCA from its theoretical underpinnings to a couple algorithms used today.

Contents

1. Theory and Motivation 1
2. Practical Algorithms 4
References 5

1. Theory and Motivation

We first examine a commonly used linear method to reduce the dimension of a data matrix. This method, Principal Component Analysis (PCA), is most helpful for high-dimensional data with linearly related, highly correlated features. This discussion will lead to a discussion of current implementations of the algorithm in software libraries. Note that by "large scale PCA" we refer to datasets with a large number of samples relative to the number of features.

Definition 1.1. We define a "data matrix" to be $\mathbf{X} \in \mathbb{R}^{p\times n}$, with elements $\mathbf{X} = (X_{ij})$ for $i \in [p]$, $j \in [n]$. Intuitively, we may think of a row of $\mathbf{X}$ as a feature vector across each of n observations.

Remark 1.2. Suppose we are interested in creating $k \leq p$ new features for our data out of linear combinations of features from our original observations. If we let $\mathbf{x}_i$ denote the i-th row of our data matrix $\mathbf{X}$, for $i \in [p]$, then such a new feature $\mathbf{y} \in \mathbb{R}^n$ would take the form

$$\mathbf{y} = \sum_{i=1}^{p} a_i \mathbf{x}_i = \mathbf{a}'\mathbf{X}$$

for $\mathbf{a} = (a_1, ..., a_p)' \in \mathbb{R}^p$.

Remark 1.3. If we want our new features $\mathbf{y}$ to capture the variance of our data well, we may wish to maximize $\text{Var}(\mathbf{a}'\mathbf{X})$, where in this case $\mathbf{X}$ is regarded as

Date: September 15, 2023.

the underlying p-variate random variable for the data matrix. However, since this quantity depends on the magnitude $||\mathbf{a}||$, we perform the maximization

$$\max_{||\mathbf{a}||_2=1} \text{Var}(\mathbf{a}'\mathbf{X}) = \max_{||\mathbf{a}||_2=1} \sum_{i=1}^{p}\sum_{j=1}^{p} a_i a_j \text{Cov}(\mathbf{x}_i, \mathbf{x}_j) = \max_{||\mathbf{a}||_2=1} \mathbf{a}'\mathbf{\Sigma}\mathbf{a}$$

where $\mathbf{\Sigma}$ denotes the covariance matrix for $\mathbf{X}$ with $\Sigma_{ij} = \text{Cov}(i,j)$, i.e. the covariance between features i and j.

Definition 1.4. We use the "sample covariance matrix" as an unbiased estimator for $\mathbf{\Sigma}$ given by $\widehat{\mathbf{\Sigma}} = \frac{1}{n-1}\sum_{j=1}^{n}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})'$. In practice, the correlation matrix is often used to standardize measurements.

Remark 1.5. If we treat the p original features of our data matrix $\mathbf{X}$ as "directions" in a p-dimensional space, we may want to find $k \leq p$ orthogonal directions (linear combinations of the original p) that maximize variance, in which case we are interested in $\mathbf{a}_i \in \mathbb{R}^p$ with

$$\mathbf{a}_i = \max_{||\mathbf{a}||_2=1} (\mathbf{a}'\widehat{\mathbf{\Sigma}}\mathbf{a}) \text{ such that } \langle \mathbf{a}, \mathbf{a}_j \rangle = 0 \text{ for } j < i$$

for $i \in [k]$.

Lemma 1.6. *The $\mathbf{a}_i$ as defined in Remark 1.5 are eigenvectors of the covariance matrix $\widehat{\mathbf{\Sigma}}$, i.e. for $i \in [k]$ we have $\widehat{\mathbf{\Sigma}}\mathbf{a}_i = \lambda_i \mathbf{a}_i$ for $\lambda_i \in \mathbb{R}$.*

Proof. We make use of Lagrange multipliers. In the case that $i = 1$, we wish to maximize

$$\mathcal{L}_1 := \mathbf{a}'\widehat{\mathbf{\Sigma}}\mathbf{a} - \lambda(\mathbf{a}'\mathbf{a} - 1)$$

so that taking the derivative and setting equal to $\mathbf{0}_p$ yields

$$\nabla_{\mathbf{a}}\mathcal{L}_1 = 2\widehat{\mathbf{\Sigma}}\mathbf{a} - 2\lambda\mathbf{a} := \mathbf{0}_p$$

$$\Rightarrow \widehat{\mathbf{\Sigma}}\mathbf{a} = \lambda\mathbf{a}$$

hence $\mathbf{a}$ is an eigenvector of $\widehat{\mathbf{\Sigma}}\mathbf{a}$ with eigenvalue $\lambda_1 := \lambda$. Now suppose for the sake of induction that $\widehat{\mathbf{\Sigma}}\mathbf{a}_j = \lambda_j \mathbf{a}_j$ for $j < i$. Then, since we are maximizing $\mathbf{a}_i'\widehat{\mathbf{\Sigma}}\mathbf{a}_i$ subject to the constraints $||\mathbf{a}||_2^2 = 1$ as well as $\langle \mathbf{a}, \mathbf{a}_j \rangle = 0$ for $j < i$, our Lagrangian is

$$\mathcal{L}_i := \mathbf{a}'\widehat{\mathbf{\Sigma}}\mathbf{a} - \lambda(\mathbf{a}'\mathbf{a} - 1) - \sum_{j<i} \delta_j(\mathbf{a}'\mathbf{a}_j - 0)$$

where the δ_j and λ are the Lagrange multipliers for each constraint. We then have

$$\nabla_{\mathbf{a}}\mathcal{L}_i = 2\widehat{\mathbf{\Sigma}}\mathbf{a} - 2\lambda\mathbf{a} - \sum_{j<i}\delta_j\mathbf{a}_j =: \mathbf{0}_p$$

$$\Rightarrow \mathbf{0}_p = \langle \mathbf{a}_\ell, 2\widehat{\mathbf{\Sigma}}\mathbf{a} \rangle - \langle \mathbf{a}_\ell, 2\lambda\mathbf{a} \rangle - \langle \mathbf{a}_\ell, \sum_{j<i}\delta_j\mathbf{a}_j \rangle = 2\mathbf{a}'\widehat{\mathbf{\Sigma}}\mathbf{a}_\ell - \delta_\ell$$

$$= 2\lambda_\ell \mathbf{a}'\mathbf{a}_\ell - \delta_\ell \Rightarrow \delta_\ell = \mathbf{0}_p$$

for $\ell \in [i-1]$. Therefore,

$$2\widehat{\mathbf{\Sigma}}\mathbf{a} - 2\lambda\mathbf{a} - \sum_{j<i}\delta_j\mathbf{a}_j = 2\widehat{\mathbf{\Sigma}}\mathbf{a} - 2\lambda\mathbf{a} = \mathbf{0}_p$$

so that

$$\widehat{\mathbf{\Sigma}}\mathbf{a} = \lambda_i\mathbf{a}$$

for $\lambda_i := \lambda$, so that by induction we may conclude the result. □

Corollary 1.7. *Our k new features $\mathbf{y}_i := \mathbf{a}_i'\mathbf{X}$ due to PCA satisfy*

$$\frac{\lambda_i}{\sum_{j=1}^k \lambda_j} = \frac{\mathrm{Var}(\mathbf{y}_i)}{\sum_{j=1}^k \mathrm{Var}(\mathbf{y}_j)}$$

and for $k = p$ we have

$$\sum_{j=1}^k \mathrm{Var}(\mathbf{y}_j) = \sum_{i=1}^p \mathrm{Var}(\mathbf{x}_i)$$

Proof. To satisfy the first part of Corollary 1.7, it suffices to observe that

$$\mathrm{Var}(\mathbf{y}_j) = \mathbf{a}_j'\widehat{\mathbf{\Sigma}}\mathbf{a}_j = \lambda_j \mathbf{a}_j'\mathbf{a}_j = \lambda_j$$

For the second part, we have that $\widehat{\mathbf{\Sigma}} \in \mathbb{R}^{p\times p}$ has trace equal to its sum of eigenvalues, hence

$$\mathrm{Tr}(\widehat{\mathbf{\Sigma}}) = \sum_{i=1}^p \mathrm{Var}(\mathbf{x}_i) = \sum_{j=1}^p \lambda_j = \sum_{j=1}^p \mathrm{Var}(\mathbf{y}_j)$$

□

Remark 1.8. Corollary 1.7 suggests that our new features $\mathbf{y}_j$ define a proportion of the total variance, which is specified by their respective eigenvalues λ_j. This also helps gives us an answer to the number k of distilled features we want. In particular, we fix a threshold percentage and choose the smallest k such that the top k PCA-derived features explain at least that percentage of the total variance.

Theorem 1.9. *(Singular Value Decomposition) Any real matrix $A \in \mathbb{R}^{m\times n}$ may be expressed as $A = U\Sigma V'$, where U and V are orthogonal and Σ is diagonal with non-negative entries. The values of Σ are called the "singular values".*

Proof. By the Spectral Theorem we may decompose the symmetric matrix $A'A$ as

$$A'A = V\Lambda V' =: \sum_{i=1}^n \lambda_i \mathbf{v}_i \mathbf{v}_i'$$

(the $\mathbf{v}_i$ are columns of V) so that we may define

$$\mathbf{u}_i := \frac{A\mathbf{v}_i}{\sqrt{\lambda_i}}$$

Then,

$$U := (\mathbf{u}_1, ..., \mathbf{u}_n) \in \mathbb{R}^{m\times n}$$

satisfies

$$U = AV\,\mathrm{diag}\left(\frac{1}{\sqrt{\lambda_1}}, ..., \frac{1}{\sqrt{\lambda_n}}\right) := AV\Sigma^{-1}$$

hence

$$A = U(V\Sigma^{-1})^{-1} = U\Sigma V^{-1} = U\Sigma V'$$

where V is orthogonal (due to the Spectral Theorem), Σ is diagonal with $\Sigma_{ii} = \sqrt{\lambda_i} > 0$ ($\lambda_i > 0$ since $A'A$ is symmetric), and

$$\langle \mathbf{u}_i, \mathbf{u}_j \rangle = \frac{1}{\sqrt{\lambda_i\lambda_j}} \mathbf{v}_j' A'A\mathbf{v}_i = \frac{\lambda_i}{\sqrt{\lambda_i\lambda_j}} \langle \mathbf{v}_i, \mathbf{v}_j \rangle = 0$$

so that U is also orthogonal. □

Remark 1.10. Note that, by applying SVD to A,

$$A'AV = V\Sigma' U' U \Sigma V' V = V\Sigma'\Sigma = V\Sigma$$

hence the columns $\mathbf{v}_i$ of V are eigenvectors of our sample data matrix $A'A$, where $\mathbf{v}_i' A$ is the i-th principal component of A. This means that an accurate Singular Value Decomposition yields an accurate PCA solution.

2. Practical Algorithms

Remark 2.1. We now present a popular randomized Singular Value Decomposition due to Halko et al. (2010). The intuition is to find a low-dimensional subspace (spanned by H) that captures most of the action of A using a random matrix G, so that $A \approx QQ'A$. The `orth` subroutine orthonormalizes H using Gram-Schmidt (as in QR decomposition) and the `svd` command invokes the LAPACK singular value decomposition for the smaller matrix $A'Q$, using the `gesvd` subroutine.

Algorithm 1 Randomized SVD

Require: $A \in \mathbb{R}^{m\times n}$, $k \ll \min(m,n)$, over-sampling parameter d
  Assign $l \leftarrow k+2$
  Sample $G \in \mathbb{R}^{n\times l}$ with $g_{ij} \sim \mathcal{N}(0,1)$
  Compute $H := (AG \,|\, AA'AG \,|\, \dots \,|\, (AA')^{i-1}AG \,|\, (AA')^d AG) \in \mathbb{R}^{m\times l(d+1)}$
  Compute $Q = \mathrm{orth}(H)$
  Compute $T := A'Q$
  Compute $\mathrm{svd}(T) = \tilde{V}\tilde{\Sigma}W'$
  Compute $\tilde{U} := QW$
  return $\tilde{U}(:, 1:k)$, $\tilde{V}(:,1:k)$, $\tilde{\Sigma}(1:k, 1:k)$

Remark 2.2. Observe that, if $\mathbf{v}_i$ is an eigenvector of AA' with eigenvalue σ_{ii}, i.e. a singular vector of A, then $\mathbf{v}$ is also an eigenvector of

$$(AA')^q AA'(AA')^q = ((AA')^q A)((AA')^q A)'$$

so that $\mathbf{v}$ is a singular vector of $(AA')^q A$ with singular value σ_{ii}^{2q+1}. Thus, this process biases the action of the resulting matrix toward more dominant singular vectors (those with higher absolute value), which is advantageous for low-rank construction. The python library `scikit-learn`'s implementation of `randomized_svd` does by default the above algorithm but computes Q by orthonormalizing after repeated multiplication by A and A'.

Remark 2.3. What happens if we cannot even fit A into memory? One way to handle this is by iteratively computing the singular value decomposition in batches as follows. Note that the "correction" subroutine stores a running mean correction vector in M.

Algorithm 2 `scikit-learn`'s SVD calculation in `IncrementalPCA` (simplified)

Require: $A \in \mathbb{R}^{m\times n}$, $k \ll \min(m,n)$ estimated components, batch size b
  Initialize empty array P
  for batch $B \subset A$ with $B \in \mathbb{R}^{b\times n}$ **do**
    Assign $M \leftarrow \text{correction}(B, M)$
    Concatenate $B \leftarrow [P, B, M] \in \mathbb{R}^{(b+k+1)\times n}$
    Compute $\text{svd}(B) = U\Sigma V'$
    Assign $P \leftarrow (\Sigma V')[:k]$
  end for
  return U, Σ, V'

Remark 2.4. The above approach effectively incorporates the singular vectors divined from parts of the matrix A seen thus far.

References

[1] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M.,& Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830. https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[2] Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence Main Track. Pages 3695-3704. https://doi.org/10.24963/ijcai.2023/411

[3] André Altmann, Laura Toloşi, Oliver Sander, Thomas Lengauer, Permutation importance: a corrected feature importance measure, Bioinformatics, Volume 26, Issue 10, May 2010, Pages 1340–1347, https://doi.org/10.1093/bioinformatics/btq134

[4] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM Review, 53(2):217–288, 2011.