

HOW ARE PRINCIPAL COMPONENT ANALYSIS AND SINGULAR VALUE DECOMPOSITION RELATED?

BY ANDRE PERUNICIC ([HTTPS://MAIL.GOOGLE.COM/MAIL/?VIEW=CM&FS=1&TF=1&TO=ANDRE@INTOLI.COM](https://mail.google.com/mail/?view=cm&fs=1&tf=1&to=andre@intoli.com)) | AUGUST 23, 2017

Follow @prncc (<https://github.com/prncc>)

57

(<https://github.com/prncc/followers>)

Star (<https://github.com/intoli/intoli-article-materials>)

235

(<https://github.com/intoli/intoli-article-materials>)

Introduction

Principal Component Analysis, or PCA, is a well-known and widely used technique applicable to a wide variety of applications such as dimensionality reduction, data compression, feature extraction, and visualization. The basic idea is to project a dataset from many correlated coordinates onto fewer uncorrelated coordinates called *principal components* while still retaining most of the variability present in the data.

Singular Value Decomposition, or SVD, is a computational method often employed to calculate principal components for a dataset. Using SVD to perform PCA is efficient and numerically robust. Moreover, the intimate relationship between them can guide our intuition about what PCA actually does and help us gain additional insights into this technique.

In this post, I will explicitly describe the mathematical relationship between SVD and PCA and highlight some benefits of doing so. If you have used these techniques in the past but aren't sure how they work internally this article is for you. By the end you should have an understanding of the motivation for PCA and SVD, and hopefully a better intuition about how to effectively employ them.

Encoding Data as a Matrix

Before we discuss how to actually perform PCA it will be useful to standardize the way in which data is encoded in matrix form. Since PCA and SVD are linear techniques, this will allow us to manipulate the data using linear transformations more easily.

It's simplest to start with an example, so suppose that you are given the width w , height h , and length l measurements of $n = 1000$ rectangular boxes. We can encode the measurements of box i as a (w_i, h_i, l_i) tuple called a *sample*. Each sample is a vector in $d = 3$ dimensions, since there are 3 numbers describing it. Because vectors are typically written horizontally, we transpose the vectors to write them vertically:

$$x_i = \begin{pmatrix} w_i \\ h_i \\ l_i \end{pmatrix} \quad \text{and} \quad x_i^T = (w_i, h_i, l_i) .$$

To pack the data into a single object we simply stack the samples as rows of a data matrix,

$$X = \begin{pmatrix} w_1 & h_1 & l_1 \\ w_2 & h_2 & l_2 \\ \vdots & \vdots & \vdots \\ w_{1000} & h_{1000} & l_{1000} \end{pmatrix} .$$

The columns of this matrix correspond to a single coordinate, i.e., all the measurements of a particular type.

In the general case, we are working with a d -dimensional dataset comprised of n samples. Instead of using letters like h , w , and l to denote the different coordinates, we simply enumerate the entries of each data-vector so that $x_i^T = (x_{i1}, \dots, x_{id})$. Before placing these vectors into a data matrix, we will actually subtract the mean of the data

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i = \left(\frac{1}{n} \sum_{i=1}^n x_{i1}, \dots, \frac{1}{n} \sum_{i=1}^n x_{id} \right)^T$$

from each sample for later convenience. We then use these *zero-centered* vectors as rows of the matrix

$$X = \begin{pmatrix} x_1^T - \mu^T \\ x_2^T - \mu^T \\ \vdots \\ x_n^T - \mu^T \end{pmatrix} .$$

See Figure 1 for a visual illustration of the case $n = 1000, d = 2$. Placing data into a matrix is particularly convenient because it allows us to write the sample covariance around the mean using matrices as

$$S = \frac{1}{n-1} \sum_{i=1}^n (x_i - \mu)(x_i - \mu)^T = \frac{1}{n-1} X^T X .$$

Dividing by $n - 1$ is a typical way to correct for the bias (https://en.wikipedia.org/wiki/Bessel%27s_correction) introduced by using the sample mean instead of the true population mean. The covariance matrix will take center stage as we work through understanding principal components and singular value decomposition.

What Is the Covariance Matrix?

The j -th column of X is nothing but the j -th coordinate that our zero-centered dataset is encoded in. The jk -th entry of the product $\frac{1}{n-1} X^T X$ is therefore given as the (scaled) dot product of the j -th column of X , denoted $x_{\bullet,j}$, and the k -th column of X , denoted $x_{\bullet,k}$. That is,

$$\frac{1}{n-1} x_{\bullet,j} \cdot x_{\bullet,k} = \frac{1}{n-1} x_{\bullet,j}^T x_{\bullet,k} = \frac{1}{n-1} \sum_{i=1}^n x_{ij} x_{ik}.$$

When $k = j$ this gives us the variance of the data along the k -th coordinate axis, and otherwise we obtain a measure of how much the two coordinates vary together.

To see that $\sum_{i=1}^n (x_i - \mu)(x_i - \mu)^T = X^T X$ is a short exercise in matrix multiplication. Note that the term on the left is a sum of products of vectors represented as $d \times 1$ and $1 \times d$ matrices, producing a result of size $d \times d$.

Principal Component Analysis

One of the first papers to introduce PCA as its known today was published in 1933 by Hotelling (<https://babel.hathitrust.org/cgi/pt?id=wu.89097139406;view=1up;seq=5>). The author's motivation was to transform a set of possibly correlated variables into "some more fundamental set of independent variables ... which determine the values [the original variables] will take." Coming from psychology, his first choice for naming them was "factors," but given that this term already had a meaning in mathematics he called the reduced set of variables "components" and the technique to find them "the method of principal components." These components are chosen sequentially in a way that lets their "contributions to the variances of [the original variables] have as great a total as possible."

Mathematically, the goal of Principal Component Analysis, or PCA, is to find a collection of $k \leq d$ unit vectors $v_i \in \mathbb{R}^d$ (for $i \in 1, \dots, k$) called *Principal Components*, or PCs, such that

1. the variance of the dataset projected onto the direction determined by v_i is maximized and
2. v_i is chosen to be orthogonal to v_1, \dots, v_{i-1} .

Now, the projection of a vector $x \in \mathbb{R}^d$ onto the line determined by any v_i is simply given as the dot product $v_i^T x$. This means that the variance of the dataset projected onto the first Principal Component v_1 can be written as

$$\frac{1}{n-1} \sum_{i=1}^n (v_1^T x_i - v_1^T \mu)^2 = v_1^T S v_1.$$

To actually find v_1 we have to maximize this quantity, subject to the additional constraint that $\|v_1\| = 1$. Solving this optimization problem using the method of Lagrange multipliers (https://en.wikipedia.org/wiki/Lagrange_multiplier) turns out to imply that

$$S v_1 = \lambda_1 v_1,$$

which just means that v_1 is an eigenvector of the covariance matrix S . In fact, since $\|v_1\| = v_1^T v_1 = 1$ we also conclude that the corresponding eigenvalue is exactly equal to the variance of the dataset along v_1 , i.e.,

$$v_1^T S v_1 = \lambda_1.$$

You can continue this process by projecting the data onto a new direction v_2 while enforcing the additional constraint that $v_1 \perp v_2$, then onto v_3 while enforcing $v_3 \perp v_1, v_2$, and so on.

The end result is that the first k principal components of X correspond exactly to the eigenvectors of the covariance matrix S ordered by their eigenvalues. Moreover, the eigenvalues are exactly equal to the variance of the dataset along the corresponding eigenvectors.

Intoli Smart Proxies

If you're doing serious web scraping, then using proxies is a must. Our Intoli Smart Proxies service (/beta/) makes it easy to stop getting blocked by bot mitigation services. Your requests are intelligently routed through clean residential IPs where they are likely to succeed, failed requests are automatically retried, and you can even make requests through remote browsers preloaded with customizations that make it hard to detect your scraper. Enter your email address below to get access to the same tooling that we use for all of our own web scraping!

Your email	GET STARTED
------------	--------------------

You may have noticed that this result suggests that there exists a full set of orthonormal eigenvectors for S over \mathbb{R} . Indeed, since S is a real symmetric matrix, meaning that $S = S^T$, the Real Spectral Theorem (<http://www.math.lsa.umich.edu/~speyer/417/SpectralTheorem.pdf>) implies exactly that. This is a non-trivial result which we will make use of later in the article, so let's expand on it a little bit. Consider the case of $k = d < n$. Taking $k = d$ principal components may seem like a strange choice if our purpose is to understand X through a lower dimensional subspace, but doing so allows us to construct a $d \times d$ matrix V whose columns are the eigenvectors of S and which therefore diagonalizes S , i.e.,

$$S = V\Lambda V^T = \sum_{i=1}^r \lambda_i v_i v_i^T,$$

where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_d)$ and $r = \text{rank}(X)$. In other words, the principal components are the columns of a rotation matrix and form the axes of a new basis which can be thought of as "aligning" with the dataset X . Of course, this image works best when X is blobby or looks approximately normal.

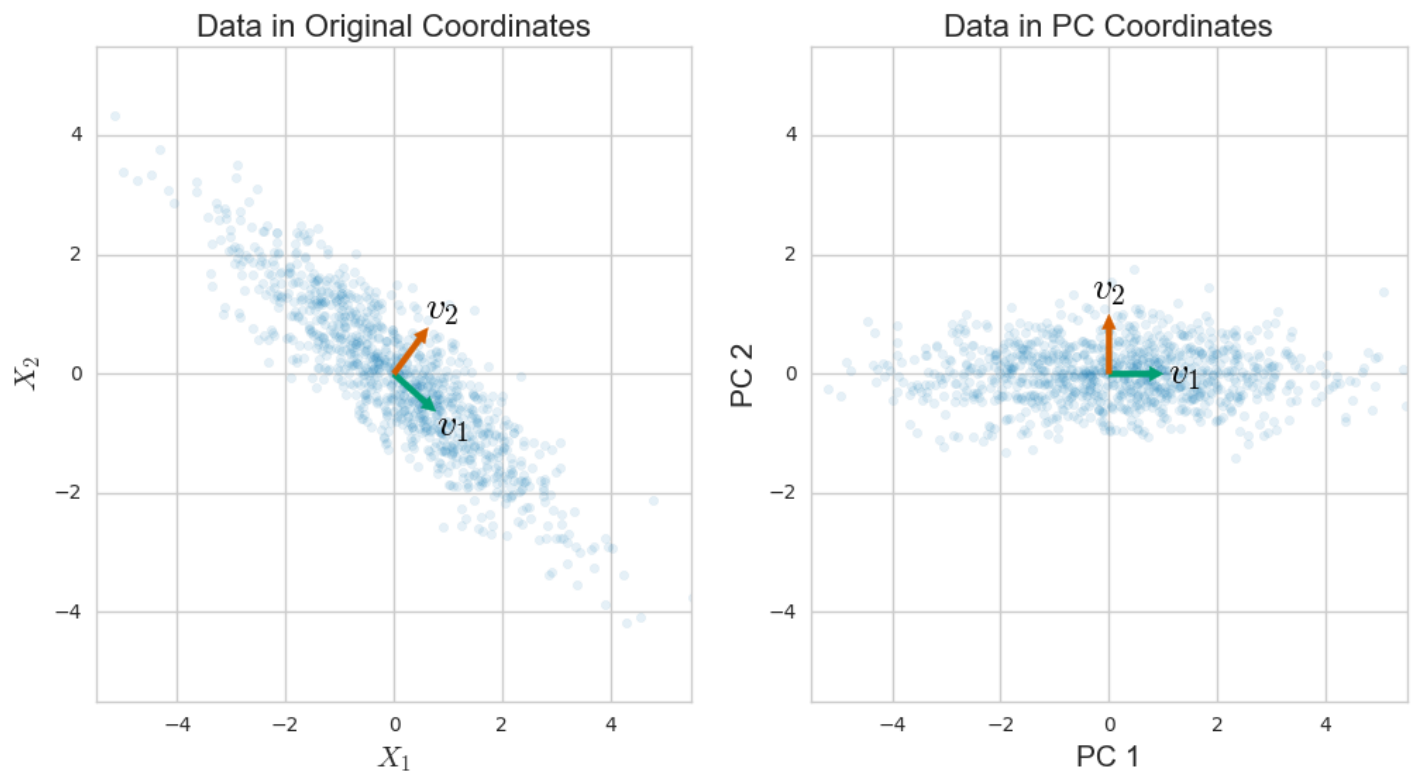


Figure 1: The original and uncorrelated view of 1000 samples drawn from a multivariate Gaussian.

Looking for a job at an exciting startup?

We have some close personal friends building an awesome tutoring platform, and they're looking to hire. If this sounds like something you might be interested in, then shoot us an email (<https://mail.google.com/mail/?view=cm&fs=1&tf=1&to=contact@intoli.com>) and we'll get you in touch!

Singular Value Decomposition

Singular Value Decomposition is a matrix factorization method utilized in many numerical applications of linear algebra such as PCA. This technique enhances our understanding of what principal components are and provides a robust computational framework that lets us compute them accurately for more datasets.

Let's start with a review of SVD for an arbitrary $n \times d$ matrix A . SVD is motivated by the fact that when viewed as a linear transformation, A maps the unit sphere $\mathbb{S}^d \subset \mathbb{R}^d$ to a (hyper)ellipse in \mathbb{R}^n . Let's consider an example with $n = d = 2$ in order to more easily visualize this fact.

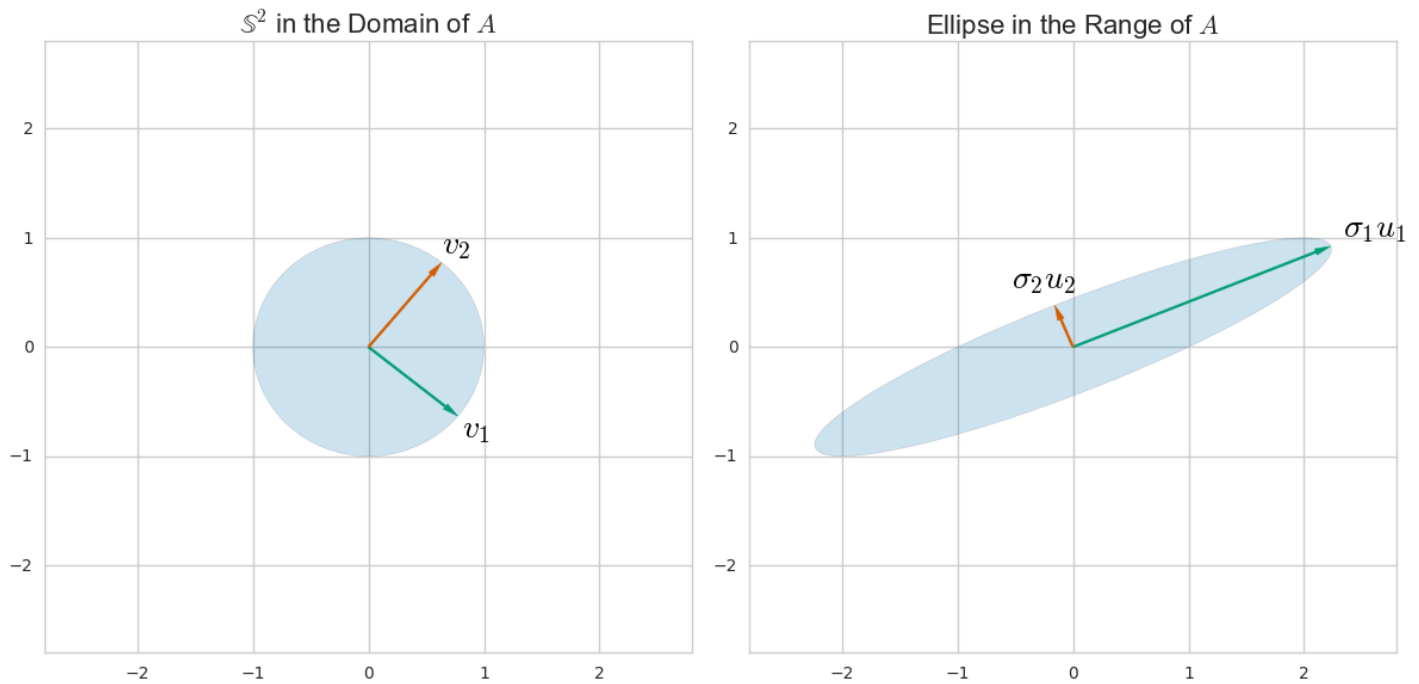


Figure 2: Action of $A = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$ on the unit sphere in \mathbb{R}^2 .

From this figure we can extract a few useful definitions which hold for arbitrary dimensions with a few extra caveats:

- The lengths σ_i of the semi-axes of the ellipse $AS^d \in \mathbb{R}^n$ are the *singular values* of A .
- The unit vectors u_i along the semi-axes of the ellipse are called the "*left*" *singular vectors* of A .
- The unit vectors v_i such that $Av_i = \sigma_i u_i$ are called the "*right*" *singular vectors* of A .

Proving the existence and uniqueness of the singular values and singular vectors is a bit beyond the scope of this article, but the proof is relatively straightforward and can be found in any book on numerical linear algebra (see below for a reference). The two-dimensional image in Figure 2 hopefully provides enough guidance for the result to be intuitive, however.

Get Help from Our Data Experts

Do you need to make sense of your data? Whether it's variable importance identification, dimensionality reduction, or other data analysis tasks our experts can guide you in the right direction.

[✉ GET STARTED NOW \(/CONTACT\)](#)

What may not be immediately apparent from the picture is that depending on n , d and $r = \text{rank}(A)$, some of the left singular vectors may "collapse" to zero. This happens when the matrix A does not have full rank for instance, since then its range must be a subspace of \mathbb{R}^n with dimension $r < d$. In general, there are exactly $r = \text{rank}(A)$

singular values and the same number of left singular vectors.

By stacking the vectors v_i and u_i into columns of matrices \hat{V} and \hat{U} , respectively, we can write the relations $Av_i = \sigma_i u_i$ as

$$A\hat{V} = \hat{U}\hat{\Sigma},$$

where $\hat{\Sigma} = \text{diag}(\sigma_1, \dots, \sigma_r)$. By padding $\hat{\Sigma}$ with zeros and adding arbitrary orthonormal columns to \hat{U} and \hat{V} we obtain a more convenient factorization

$$AV = U\Sigma.$$

Since V is unitary – that is, it has unit length, orthonormal columns – it follows that $V^{-1} = V^T$, so multiplying by V^T gives us the *singular value decomposition* of A :

$$A = U\Sigma V^T.$$

Things got a bit busy here, so here's a visual summary of this result.

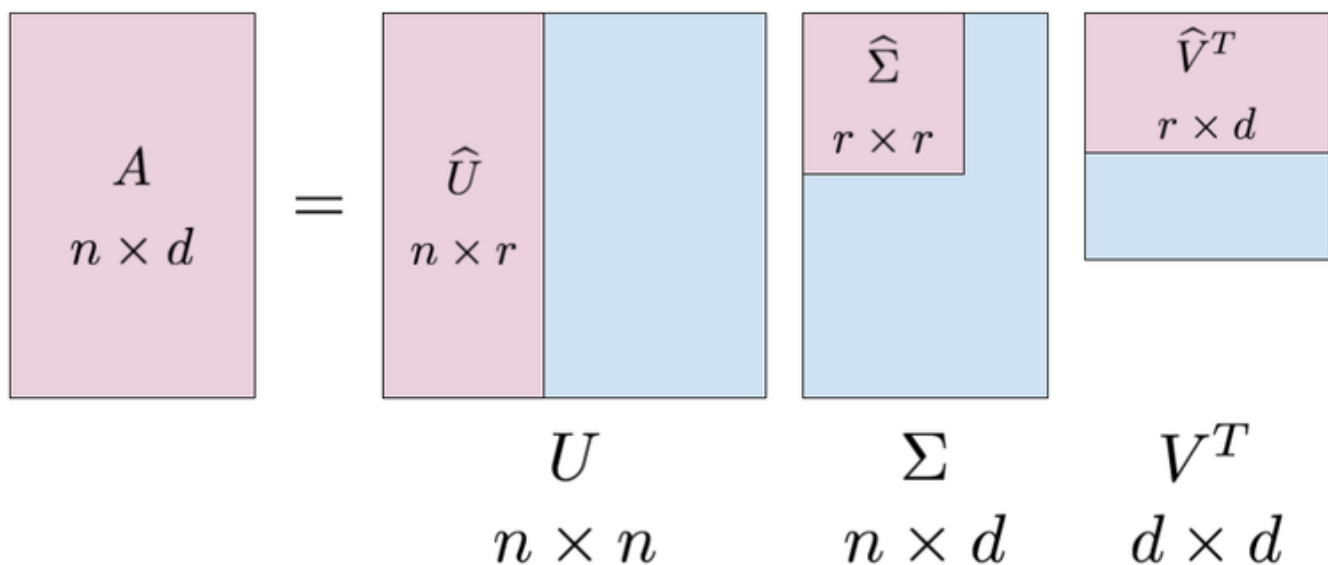


Figure 3: Components of the singular value decomposition of a matrix A . The internal areas correspond to the stacked vectors and singular values motivated by Figure 2.

Note that this result is in the end consistent with the ellipse picture above since the effect of the three transformations is to switch to a new basis using V^T , stretch axes to get a (hyper)ellipse using Σ , and rotate and reflect using U (which doesn't affect the shape of the ellipse).

Relation Between SVD and PCA

Since any matrix has a singular value decomposition, let's take $A = X$ and write

$$X = U\Sigma V^T.$$

We have so far thought of A as a linear transformation, but there's nothing preventing us from using SVD on a data matrix. In fact, note that from the decomposition we have

$$X^T X = (U \Sigma V^T)^T (U \Sigma V^T) = V \Sigma^T U^T U \Sigma V^T = V (\Sigma^T \Sigma) V^T,$$

which means that $X^T X$ and $\Sigma^T \Sigma$ are similar. Similar matrices have the same eigenvalues, so the eigenvalues λ_i of the covariance matrix $S = \frac{1}{n-1} X^T X$ are related to the singular values σ_i of the matrix X via

$$\sigma_i^2 = (n-1) \lambda_i,$$

for $i \in \{1, \dots, r\}$, where as usual $r = \text{rank}(A)$.

To fully relate SVD and PCA we also have to describe the correspondence between principal components and singular vectors. For the right singular vectors we take

$$\widehat{V}^T = \begin{pmatrix} \frac{v_1^T}{\sigma_1} \\ \vdots \\ \frac{v_r^T}{\sigma_r} \end{pmatrix}$$

where v_i are the principal components of X . For the left singular vectors we take

$$u_i = \frac{1}{\sqrt{(n-1)\lambda_i}} X v_i.$$

Before proving that these choices are correct, let's verify that they at least make intuitive sense. From Figure 2 we can see that the right singular vectors v_i are an orthonormal set of d -dimensional vectors that span the row space of the data. Since X is zero centered we can think of them as capturing the spread of the data around the mean in a sense reminiscent of PCA. We also see that the column space of X is generated by the left singular vectors u_i . The column space of a data matrix is a summary of the different coordinates of the data, so it makes sense that we've chosen u_i to be a (scaled) projection of X into the direction of the i -th principal component. These observations are encouraging but imprecise, so let's actually prove that our choices are correct algebraically. The result will follow from this general fact.

Fact. Let $A = U \Sigma V^T$ be the singular decomposition of the real matrix A . Then,

$$A = \sum_{i=1}^r \sigma_i u_i v_i^T.$$

Proof. Let's prove this statement in a small example to simplify the notation. The general case will follow analogously. So, let $v_i = (v_{1i}, v_{2i})^T$ for $i = 1, 2$ and $u_i = (u_{1i}, u_{2i}, u_{3i})^T$ for $i = 1, 2, 3$. That is,

$$\begin{aligned}
 U\Sigma V^T &= \begin{pmatrix} u_1 & u_2 \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \end{pmatrix} \\
 &= \begin{pmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} v_{11} & v_{21} \\ v_{12} & v_{22} \end{pmatrix}
 \end{aligned}$$

Next, split up the matrix Σ so that

$$U\Sigma V^T = U \begin{pmatrix} \sigma_1 & 0 \\ 0 & 0 \end{pmatrix} V^T + U \begin{pmatrix} 0 & 0 \\ 0 & \sigma_2 \end{pmatrix} V^T$$

and tackle the terms individually. One way to rewrite the first term is

$$\begin{aligned}
 U \begin{pmatrix} \sigma_1 & 0 \\ 0 & 0 \end{pmatrix} V^T &= \begin{pmatrix} u_{11}\sigma_1 & 0 \\ u_{21}\sigma_1 & 0 \\ u_{31}\sigma_1 & 0 \end{pmatrix} \begin{pmatrix} v_{11} & v_{21} \\ v_{12} & v_{22} \end{pmatrix} \\
 &= \sigma_1 u_1 v_1^T,
 \end{aligned}$$

where the last step follows because the entries highlighted in blue (the second column of the first matrix and the second row of V^T) do not affect the result of the matrix multiplication. A similar calculation shows that the second term is $\sigma_2 u_2 v_2^T$, which proves the claim for the given example. Note that in general some of the singular values could be 0, which makes the sum go only up to $r = \text{rank}(A)$. \square

Let's apply this fact to X . We have that

$$\begin{aligned}
 U\Sigma V^T &= \sum_{i=1}^r \sigma_i u_i v_i^T \\
 &= \sum_{i=1}^r \sqrt{(n-1)\lambda_i} \frac{1}{\sqrt{(n-1)\lambda_i}} X v_i v_i^T \\
 &= X \sum_{i=1}^r v_i v_i^T \\
 &= X,
 \end{aligned}$$

where the last step follows from $I = V^T V = \sum_{i=1}^r v_i v_i^T$. We have thus established the connection between SVD and PCA.

A Few Benefits

Let's explore a few consequences of this correspondence. From the definition of U we already know that it projects and slightly scales the data onto the principal components. It is more satisfying, however, to derive this result as a general consequence of being able to write $X = U\Sigma V^T$, so let's turn our attention to that. Multiplying the SVD of X by V we obtain

$$XV = U\Sigma V^T V = U\Sigma.$$

From the left hand side we see that the i -th column of this matrix is the projection

$$z_i = Xv_i = ((x_1 - \mu)^T v_i, \dots, (x_n - \mu)^T v_i)^T$$

of the data onto the i -th PC. From the right hand side we see that this corresponds to $z_i = u_i \sigma_i$, which is what we were trying to recover.

We can extract another useful morsel by thinking about the scaling of U . Recall that each singular value corresponds to the length of a semi-axis of an ellipse that describes the column space of X . By dividing out the singular value (and multiplying by $\sqrt{n-1}$ to deal with the fact that we started with the covariance) we obtain a transformed dataset which is in some sense spherical:

$$Y = \sqrt{n-1} U \Sigma^{-1}.$$

This process produces a unit variance dataset in a way that can be difficult to achieve by just centering and scaling.

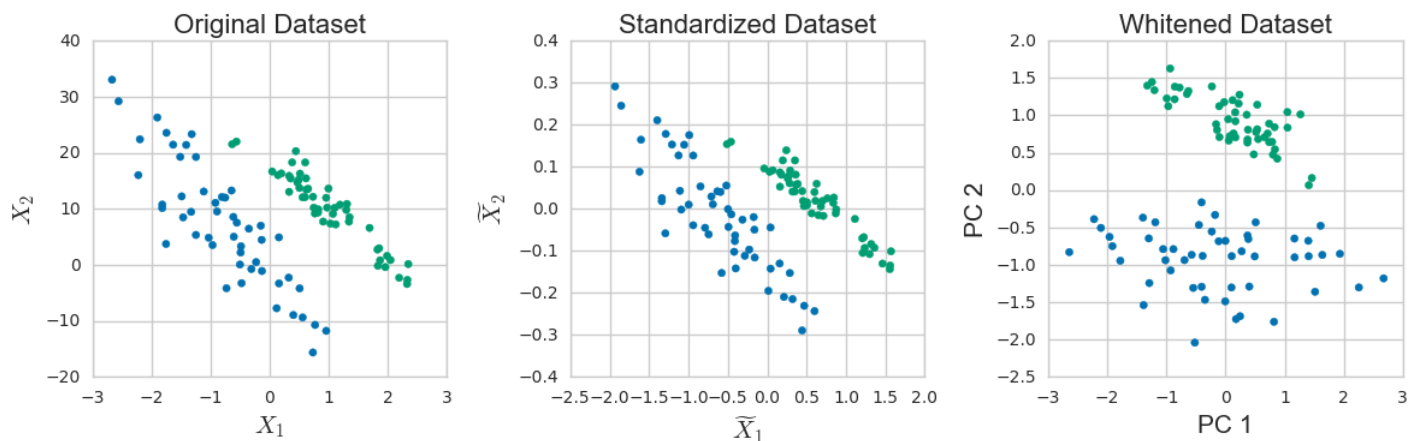


Figure 4: Whitening can be used to easily alleviate the effects of coordinates given in different units. In the left plot you can see that X_2 is on a different scale than X_1 . The middle plot depicts the centered and scaled dataset. In the right plot you see the results of whitening.

As a final remark, let's discuss the *numerical* advantages of using SVD. A basic approach to actually calculating PCA on a computer would be to perform the eigenvalue decomposition of $X^T X$ directly. It turns out that doing so would introduce some potentially serious numerical issues that could be avoided by using SVD.

The problem lies in the fact that on a computer real numbers are represented with finite precision. Since each number is stored in a finite amount of memory there is necessarily a gap between consecutive representable numbers. For double precision floating point numbers, for instance, the relative error between a real number and its closest floating point approximation is on the order of $\varepsilon \approx 10^{-16}$. Algorithms that take this limitation into account are called *backward stable*. The basic idea is that such an algorithm produces the correct output for an input value that's within ε of the value you actually asked about.

If $\tilde{\sigma}_i$ is the output of a backward stable algorithm calculating singular values for X and σ_i is the true singular value, it can be shown that

$$|\tilde{\sigma}_i - \sigma_i| = O(\varepsilon \|X\|),$$

where $\|X\|$ is a reasonable measure of the size of X . On the other hand, a backward stable algorithm that calculates the eigenvalues l_i of $X^T X$ can be shown to satisfy

$$|\tilde{l}_i - l_i| = O(\varepsilon \|X^T X\|) = O(\varepsilon \|X\|^2),$$

which when given in terms of the singular values of X becomes

$$|\tilde{\sigma}_i - \sigma_i| = O\left(\varepsilon \frac{\|X\|^2}{\sigma_i}\right).$$

The end result here is that if you are interested in relatively small singular values, e.g. $\sigma_i \ll \|X\|$, working directly with X will produce much more accurate results than working with $X^T X$. Luckily, SVD lets us do exactly that.

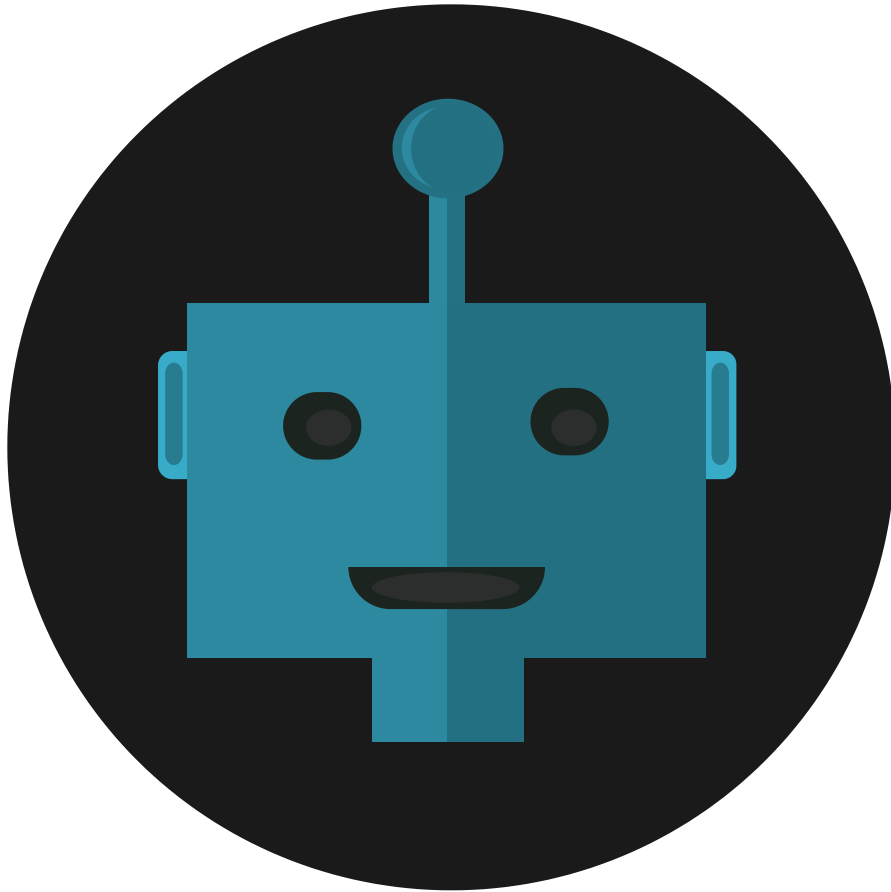
Conclusion

There is obviously a lot more to say about SVD and PCA. If you'd like to know more about ways to extend these ideas and adapt them to different scenarios, a few of the references that I have found useful while writing this article are Numerical Linear Algebra (<http://bookstore.siam.org/ot50/>) by Trefethen and Bau (1997, SIAM) and Principal Component Analysis (<http://www.springer.com/us/book/9780387954424>) by Jolliffe (2002, Springer).

If you need a custom data sourcing, transformation and analysis solution, get in touch with us (</contact/>) to see how our experts can help you make the most out of your data.

SUGGESTED ARTICLES

If you enjoyed this article, then you might also enjoy these related ones.

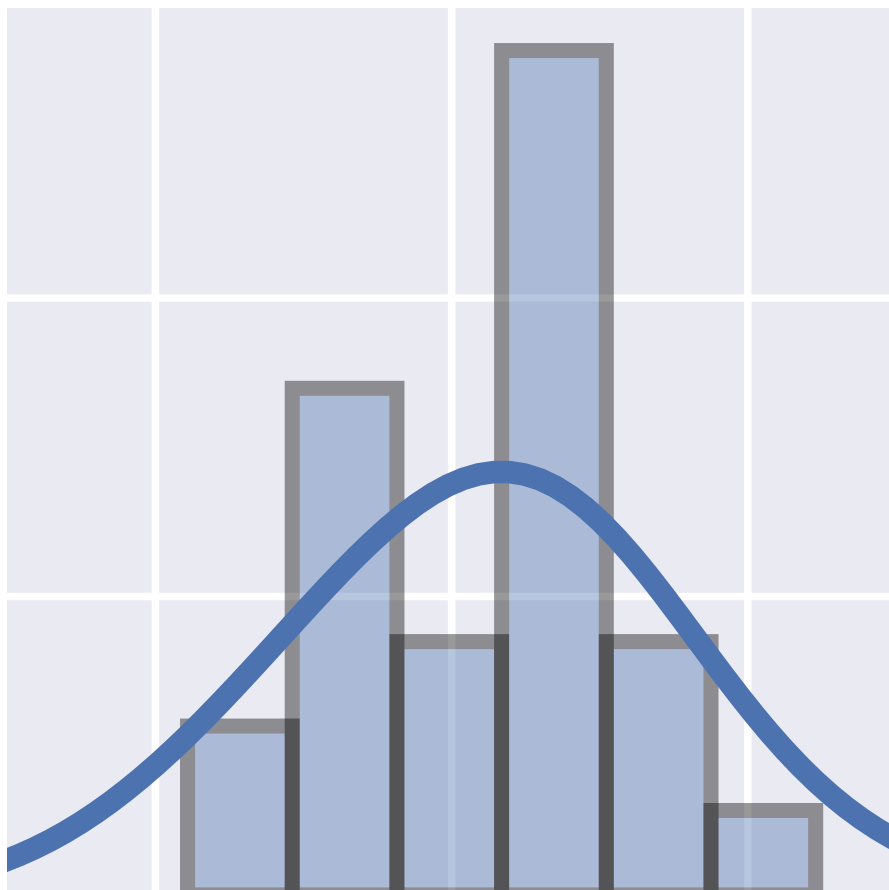


PERFORMING EFFICIENT BROAD CRAWLS WITH THE AOPIC ALGORITHM ([//INTOLI.COM/BLOG/AOPIC-ALGORITHM/](https://intoli.com/blog/aopic-algorithm/))

BY **ANDRE PERUNICIC**
ON SEPTEMBER 16, 2018

Learn how to estimate page importance and allocate bandwidth during a broad crawl.

[READ MORE \(//INTOLI.COM/BLOG/AOPIC-ALGORITHM/\)](https://intoli.com/blog/aopic-algorithm/)

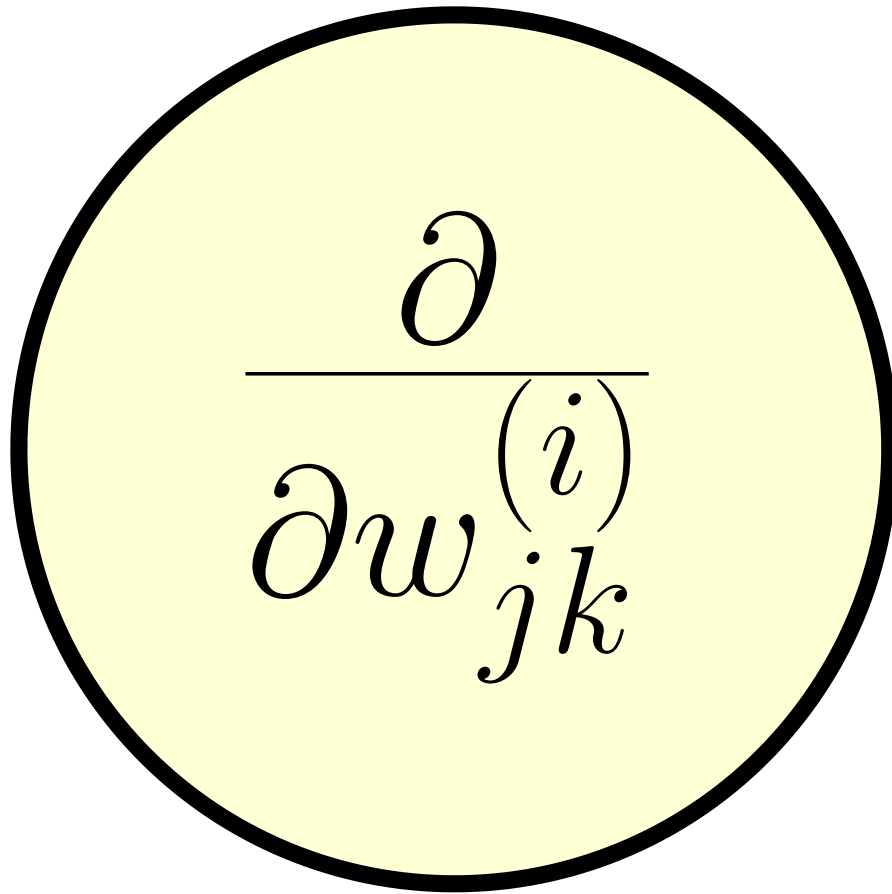


MARKOV'S AND CHEBYSHEV'S INEQUALITIES EXPLAINED ([//INTOLI.COM/BLOG/CHEBYSHEVS- INEQUALITY/](https://intoli.com/blog/chebyshevs-inequality/))

BY EVAN SANGALINE
ON AUGUST 4, 2017

A look at why Chebyshev's Inequality holds true and some potential applications.

READ MORE ([//INTOLI.COM/BLOG/CHEBYSHEVS-INEQUALITY/](https://intoli.com/blog/chebyshevs-inequality/))



UNDERSTANDING NEURAL NETWORK WEIGHT INITIALIZATION ([//INTOLI.COM/BLOG/NEURAL-NETWORK- INITIALIZATION/](https://intoli.com/blog/neural-network-initialization/))

BY ANDRE PERUNICIC
ON JULY 25, 2017

Exploring the effects of neural network weight initialization strategies.

[READ MORE \(//INTOLI.COM/BLOG/NEURAL-NETWORK-INITIALIZ](https://intoli.com/blog/neural-network-initialization/)

COMMENTS

What do you want to say? markdown

▼ **hide preview**

Email - never shown

What's next? verify your email address for reply notifications!

save message

unverified · 3y, 178d ago

Hi, thanks for your post. In the equation following this sentence "This means that the variance of the dataset projected onto the first Principal", could you explain why the expression on the left is equal to the right? Thanks!

remark link collapse [-]



andre mod · 3y, 166d ago · [edited]

You can use the equation for S from the box "What Is the Covariance Matrix?" to expand the right hand side, then use

$$v_1^T(x_i - \mu) = [v_1^T(x_i - \mu)]^T \in \mathbb{R}.$$

remark link parent

unverified · 3y, 144d ago

Helpful

remark link

unverified · 3y, 11d ago

Hi, thanks for your post, very insightful.

Could it be that the U matrix represents the PCA taking variables as vectors instead of observations?

remark link collapse [-]



andre mod · 2y, 364d ago

Hi and thanks. I'm not sure what you mean about U, but I'll think about it if you clarify that for me.

remark link parent

G51p8mGy · 3y, 11d ago

Hi, thanks again for your post.

Could you develop mathematically the affirmation: "By dividing out the singular value (and multiplying it by $(n-1)^{1/2}$ to deal with the fact that we started with the covariance) we obtain a transformed dataset which is in some sense spherical"? How do you get $Y = (n-1)^{1/2} U E^{-1}$?

Thanks

remark link collapse [-]



andre mod · 2y, 364d ago

A way to understand the paragraph it is to write out U in terms of its column vectors and multiply that use the definitions of Σ and u_i . You can also take the covariance of the transformed data set, and it should come out to be the identity matrix!

remark link parent

unverified · 2y, 313d ago

Thanks for the post. There is a mistake in Fig. 3, the dimension of Σ must be $d \times d$

remark link collapse [-]



andre mod · 2y, 270d ago

Thanks. It's actually Fig. 2, and the dimension of U is incorrect in the figure: it should be $n \times n$.

remark link parent

unverified · 2y, 100d ago · [edited]

Hi,

I could not figure out how did you go from

$$|\tilde{l}_i - l_i| = O(\varepsilon \|X^T X\|) = O(\varepsilon \|X\|^2)$$

to this

$$|\tilde{\sigma}_i - \sigma_i| = O\left(\varepsilon \frac{\|X\|^2}{\sigma_i}\right).$$

(i.e. last two equations of the post) I would appreciate if you could explain.

SEARCH



TAGS

[MATH \(4\) \(//INTOLI.COM/TAGS/MATH\)](https://intoli.com/tags/math/)[BROWSE ALL TAGS \(/TAGS/\)](https://intoli.com/tags/)

INTOLI SMART PROXIES

Our residential proxies do what other proxies do, and a whole lot more. We intelligently route your requests through clean residential IPs, automatically retry failed requests, and can even render web pages in remote browsers preloaded with realistic fingerprints that make them difficult to detect!

[NEXT →](#)

ABOUT US

We're a consulting agency with deep expertise in data acquisition, processing, and analysis. From web scrapers to machine learning, we're here to help.

OUR NEW ARTICLE NEWSLETTER

Sign up to receive occasional emails with the best new articles from our blog. You can unsubscribe at any time.

[SUBSCRIBE](#)

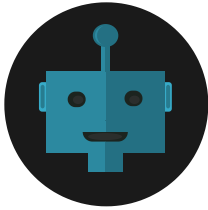
RECENT POSTS



THE RED TIDE AND THE BLUE WAVE: GERRYMANDERING AS A RISK VS. REWARD STRATEGY (//INTOLI.COM/BLOG/GERRYMANDERING/)

[\(//intoli.com/blog/gerrymandering/\)](https://intoli.com/blog/gerrymandering/)

PERFORMING EFFICIENT BROAD CRAWLS WITH THE AOPIC ALGORITHM (//INTOLI.COM/BLOG/AOPIC-ALGORITHM/)



([//intoli.com/blog/aopic-algorithm/](https://intoli.com/blog/aopic-algorithm/))



BREAKING OUT OF THE CHROME/WEBEXTENSION SANDBOX
([//INTOLI.COM/BLOG/SANDBOX-BREAKOUT/](https://intoli.com/blog/sandbox-breakout/))

([//intoli.com/blog/sandbox-breakout/](https://intoli.com/blog/sandbox-breakout/))

CONTACT

Intoli, LLC

725 NW 4th Ave
Gainesville, FL 32601

United States

[GO TO CONTACT PAGE \(/CONTACT\)](https://intoli.com/contact/)

Copyright (c) 2015 - 2017, Intoli, LLC; all rights reserved.

[Privacy Policy \(/privacy/\)](https://intoli.com/privacy/) • [Terms of Service \(/terms/\)](https://intoli.com/terms/)