7
Low-Dimensional Representation

High-dimensional datasets arise in quite a few settings. This chapter concerns a phenomenon that arises frequently: the data points (*i.e.*, vectors) collectively turn out to be "approximately low rank." A running theme in this chapter is that arrays and matrices, which in introductory courses like COS 126 and COS 226 were thought of as data structures (*i.e.*, an abstraction from programming languages), are treated now as objects that we can pass through some remarkable (but simple) mathematical procedures.

If a large dataset of N vectors in $\mathbb{R}^d$ has rank k, then we can think of a natural compression method. Let U be the k-dimensional subspace spanned by the vectors, and identify k basis vectors for U. For each of the N vectors, find k coefficients of their representation in terms of the basis vectors. Following this method, instead of specifying the N vectors using Nd real numbers, we can represent them using $k(N+d)$ real numbers, which is a big win if d is much larger than k.

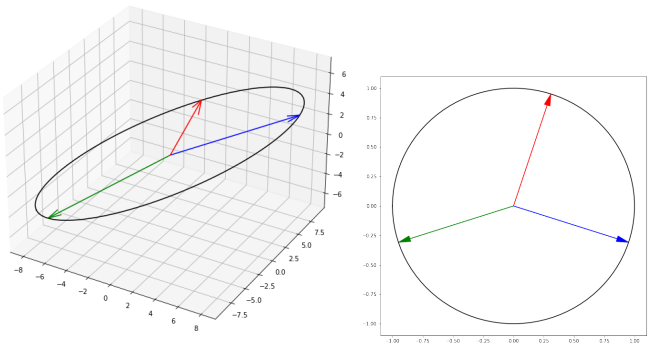

Figure 7.1: $\vec{\mathbf{v}}_1, \vec{\mathbf{v}}_2, \vec{\mathbf{v}}_3 \in \mathbb{R}^3$ (left) and their 2-dimensional representations $\widehat{\vec{\mathbf{v}}}_1, \widehat{\vec{\mathbf{v}}}_2, \widehat{\vec{\mathbf{v}}}_3 \in \mathbb{R}^2$.

Example 7.0.1. *Figure 7.1 shows three vectors* $\vec{\mathbf{v}}_1 = (3.42, -1.33, 6.94), \vec{\mathbf{v}}_2 = (7.30, 8.84, 1.95), \vec{\mathbf{v}}_3 = (-7.92, -6.37, -5.66)$ *in* $\mathbb{R}^3$. *The three vectors have rank* 2 *— they are all in the* 2*-dimensional linear subspace generated*

by $\vec{\mathbf{u}}_1 = (8, 8, 4)$ *and* $\vec{\mathbf{u}}_2 = (1, -4, 6)$. *Specifically,*

$$\vec{\mathbf{v}}_1 = 0.31\vec{\mathbf{u}}_1 + 0.95\vec{\mathbf{u}}_2$$
$$\vec{\mathbf{v}}_2 = 0.95\vec{\mathbf{u}}_1 - 0.31\vec{\mathbf{u}}_2$$
$$\vec{\mathbf{v}}_3 = -0.95\vec{\mathbf{u}}_1 - 0.31\vec{\mathbf{u}}_2$$

Therefore, we can represent these vectors in a 2*-dimensional plane, as* $\widehat{\vec{\mathbf{v}}}_1 = (0.31, 0.95), \widehat{\vec{\mathbf{v}}}_2 = (0.95, -0.31), \widehat{\vec{\mathbf{v}}}_3 = (-0.95, -0.31)$

7.1 Low-Dimensional Representation with Error

Of course, in general, high dimensional datasets are not exactly low rank. We're interested in datasets which have low-dimension representations *once we allow some error*.

Definition 7.1.1 (Low-dimensional Representation with Error)**.** *We say a set of vectors* $\vec{\mathbf{v}}_1, \vec{\mathbf{v}}_2, \ldots, \vec{\mathbf{v}}_N \in \mathbb{R}^d$ *has* ***rank*** k ***with mean-squared error*** ϵ *if there exist some basis vectors* $\vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2, \ldots, \vec{\mathbf{u}}_k \in \mathbb{R}^d$ *and* N *vectors* $\widehat{\vec{\mathbf{v}}}_1, \widehat{\vec{\mathbf{v}}}_2, \ldots, \widehat{\vec{\mathbf{v}}}_N \in \mathit{span}(\vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2, \ldots, \vec{\mathbf{u}}_k)$ *such that*

$$\frac{1}{N}\sum_i \left\|\vec{\mathbf{v}}_i - \widehat{\vec{\mathbf{v}}}_i\right\|_2^2 \leq \epsilon \tag{7.1}$$

We say that $\widehat{\vec{\mathbf{v}}}_1, \ldots, \widehat{\vec{\mathbf{v}}}_N$ *are the* ***low-rank*** *or* ***low-dimensional*** *approximation of* $\vec{\mathbf{v}}_1, \ldots, \vec{\mathbf{v}}_N$*. Typically we will assume without loss of generality that the basis vectors are orthonormal (i.e., have* ℓ_2 *norm equal to* 1 *and are pairwise orthogonal).*

Definition 7.1.1 can be thought of as a *lossy* compression of the dataset of vectors since the low-dimensional representation of vectors is roughly correct, but with a bound of ϵ on the MSE. This compression view will be used in Section 7.3.

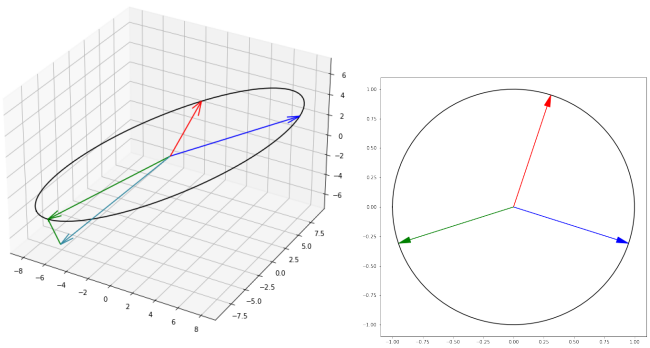

Figure 7.2: $\vec{\mathbf{v}}_1, \vec{\mathbf{v}}_2, \vec{\mathbf{v}}_3 \in \mathbb{R}^3$ (left) and their 2-dimensional approximations $\widehat{\vec{\mathbf{v}}}_1, \widehat{\vec{\mathbf{v}}}_2, \widehat{\vec{\mathbf{v}}}_3$ represented in the 2-dimensional subspace spanned by $\vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2$.

Example 7.1.2. *Figure 7.2 shows three vectors* $\vec{\mathbf{v}}_1 = (3.42, -1.33, 6.94), \vec{\mathbf{v}}_2 = (7.30, 8.84, 1.95), \vec{\mathbf{v}}_3 = (-6.00, -7.69, -6.86)$ *in* $\mathbb{R}^3$*. The three vectors have rank* 2 *with mean-squared error 2.5. If you choose the basis vectors* $\vec{\mathbf{u}}_1 = (8, 8, 4), \vec{\mathbf{u}}_2 = (1, -4, 6)$ *and the low-rank approximations* $\widehat{\vec{\mathbf{v}}}_1 = \vec{\mathbf{v}}_1, \widehat{\vec{\mathbf{v}}}_2 = \vec{\mathbf{v}}_2, \widehat{\vec{\mathbf{v}}}_3 = (-7.92, -6.37, -5.66) \in span(\vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2)$ *then,*

$$\frac{1}{3}\sum_i \left\|\vec{\mathbf{v}}_i - \widehat{\vec{\mathbf{v}}}_i\right\|_2^2 \simeq 2.28 \leq 2.5$$

Note that the basis vectors in this example are only orthogonal and not orthonormal, but it is easy to set them as orthonormal by normalizing them.

Problem 7.1.3. *Show that if* $\vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2, \ldots, \vec{\mathbf{u}}_k \in \mathbb{R}^d$ *is any set of orthonormal vectors and* $\vec{\mathbf{v}} \in \mathbb{R}^d$ *then the vector* $\widehat{\vec{\mathbf{v}}}$ *in* $span(\vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2, \ldots, \vec{\mathbf{u}}_k)$ *that minimizes* $\left\|\vec{\mathbf{v}} - \widehat{\vec{\mathbf{v}}}\right\|_2^2$ *is*

$$\sum_{j=1}^{k} (\vec{\mathbf{v}} \cdot \vec{\mathbf{u}}_j)\vec{\mathbf{u}}_j \tag{7.2}$$

(Hint: If $\alpha_1, \alpha_2, \ldots, \alpha_k$ *minimize* $\left\|\vec{\mathbf{v}} - \sum_j \alpha_j \vec{\mathbf{u}}_j\right\|_2^2$ *then the gradient of this expression with respect to* $\alpha_1, \alpha_2, \ldots, \alpha_k$ *must be zero.)*

Problem 7.1.3 illustrates how to find the low-dimensional representation of the vectors, once we specify the k basis vectors. Notice that (7.2) is the *vector projection* of $\vec{\mathbf{v}}$ onto the subspace U spanned by the vectors $\vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2, \ldots, \vec{\mathbf{u}}_k$. Therefore, the approximation error $\left\|\vec{\mathbf{v}} - \widehat{\vec{\mathbf{v}}}\right\|_2^2$ is the squared norm of the component of $\vec{\mathbf{v}}$ that is orthogonal to U. [1]

[1] Also known as the *vector rejection* of $\vec{\mathbf{v}}$ from U.

Problem 7.1.4 is only for more advanced students but all students should read its statement to understand the main point. It highlights how miraculous it is that real-life datasets have low-rank representations. It shows that generically one would expect ϵ in (7.1) to be $1 - k/n$, which is almost 1 when $k \ll n$. And yet in real life ϵ is small for fairly tiny k.

Problem 7.1.4. *Suppose the* $\vec{\mathbf{v}}_i$*'s are unit vectors* [2] *and the vectors* $\vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2, \ldots, \vec{\mathbf{u}}_k$ *were the basis vectors of a random k-dimensional subspace in* $\mathbb{R}^d$*. (That is, chosen without regard to the* $\vec{\mathbf{v}}_i$*'s.) Heuristically argue that the* ϵ *one would need in* (7.1) *would be* $1 - k/n$*.*

[2] Note: the maximum possible value of ϵ when $\vec{\mathbf{v}}_i$'s are unit vectors is 1. Convince yourself!

7.1.1 Computing the Low-Dimensional Representation with Error

In Problem 7.1.3, we have already seen how to find the low-dimension representation with error, once we are given the basis vectors. All there remains is to identify a suitable value of k and find the corresponding basis vectors that will minimize the error.

There is a simple linear algebraic procedure, the *Singular Value Decomposition (SVD)*. Given a set of vectors $\vec{\mathbf{v}}_i$ and a positive integer

k, SVD can output the best orthonormal basis in sense of Definition 7.1.1 that has the lowest possible value of ϵ. In practice, we treat k as a hyperparameter and use trial and error to find the most suitable k. Problem 7.1.5 shows that the accuracy of the low-dimensional representation will decrease when we choose a smaller number of dimensions. So we are making a choice between the accuracy of the representations against how condensed our compression is.

Problem 7.1.5. *Show that as we decrease k in Definition 7.1.1, the corresponding ϵ can only increase (i.e., cannot decrease).*

Formally, SVD takes a matrix as its input; the rows of this matrix are the vector $\vec{\mathbf{v}}_i$'s. The procedure operates on this matrix to output a low-rank approximation. We discuss details in Section 20.3. To follow the rest of this chapter, you do not need to understand details of the procedure. You just to remember the fact that the best k-dimensional representation is computable for each k. In practice, programming languages have packages that will do the calculations for you. Below is a Python code snippet that will calcuate the SVD.

```
import sklearn.decomposition.TruncatedSVD

# n * n matrix
data = ...
# prepare transform on dataset matrix "data"
svd = TruncatedSVD(n_components=k)
svd.fit(data)
# apply transform to dataset and output a n * k matrix
transformed = svd.transform(data)
```

Now we see some fun applications.

7.2 *Application 1: Stylometry*

In many cases in old literature, the identity of the author is disputed. For instance, the King James Bible (*i.e.*, the canonical English bible from the 17th century) was written by a team whose identities and work divisions are not completely known. Similarly the *Federalist Papers*, an important series of papers explicating finer points of the US government and constitution, were published in the early days of the republic with the team of authors listed as Alexander Hamilton, James Madison, and John Jay. But it was not revealed which paper was written by whom. In such cases, can machine learning help identify who wrote what?

Here we present a fun example about the books in the Wizard of Oz series. [3] L. Frank Baum was the author of the original *Wonderful Wizard of Oz*, which was a best-seller in its day and remains highly popular to this day. The publisher saw a money-making opportunity

[3] Original paper at http://dh.obdurodon.org/Binongo-Chance.pdf. A survey paper by Erica Klarreich in Science News Dec 2003: *Statistical tests are unraveling knotty literary mysteries* at http://web.mit.edu/allanmc/www/stylometrics.pdf

and convinced Baum to also write 15 follow-up books. After Baum's death the publisher managed to pass on the franchise to Ruth Plumly Thompson, who wrote many other books.

However, the last of the Baum books, *Royal Book of Oz* (RBOO), always seemed to Oz readers closer in style to Thompson's books than to Baum's. But with all the principals in the story now dead, there seemed to be no way to confirm the suspicion. Now we describe how simple machine learning showed pretty definitively that this book was indeed written by Ruth Plumly Thompson. The main idea is to represent the books vectors in some way and then find their low-rank representations.

Figure 7.3: *Royal Book of Oz* (1921). Cover image from https://en.wikipedia.org/wiki/The_Royal_Book_of_Oz

The key idea is that different authors use English words at different frequencies. Surprisingly, the greatest difference lies in frequencies of *function words* such as WITH, HOWEVER, UPON, rather than fancy vocabulary words (the ones found on your SAT exam).

Example 7.2.1. *Turns out Alexander Hamilton used* UPON *about* 10 *times more frequently than James Madison. We know this from analyzing their individual writing outside their collaboration on the Federalist Papers. Using these kinds of statistics, it has been determined that Hamilton was the principal author or even the sole author of almost all of the Federalist Papers.*

The statistical analysis of the Oz books consisted of looking at the frequencies of 50 function words. All Oz books except RBOO were divided into text blocks of 5000 words each. For each text block, the frequency (*i.e.*, number of occurrences) of each function word was computed, which allows us to represent the block as a vector in $\mathbb{R}^{50}$. There were 223 text blocks total, so we obtain 223 vectors in $\mathbb{R}^{50}$.

the (6.7%)	with (0.7%)	up (0.3%)	into (0.2%)	just (0.2%)
and (3.7%)	but (0.7%)	no (0.3%)	now (0.2%)	very (0.2%)
to (2.6%)	for (0.7%)	out (0.3%)	down (0.2%)	where (0.2%)
a/an (2.3%)	at (0.6%)	what (0.3%)	over (0.2%)	before (0.2%)
of (2.1%)	this/these (0.5%)	then (0.3%)	back (0.2%)	upon (0.1%)
in (1.3%)	so (0.5%)	if (0.3%)	or (0.2%)	about (0.1%)
that/those (1.0%)	all (0.5%)	there (0.3%)	well (0.2%)	after (0.1%)
it (1.0%)	on (0.5%)	by (0.3%)	which (0.2%)	more (0.1%)
not (0.9%)	from (0.4%)	who (0.3%)	how (0.2%)	why (0.1%)
as (0.7%)	one/ones (0.3%)	when (0.2%)	here (0.2%)	some (0.1%)

Figure 7.4: The top 50 most frequently used function words in the Wizard of Oz series. Their occurrences were counted in 223 text blocks. Figure from Binongo's paper.

Then we compute a rank 2 approximation of these 223 vectors.

Figure 7.5 shows the scatter plot in the 2-dimensional visualization. The two axes correspond to the two basis vectors we found for the rank 2 approximation. It becomes quickly clear that the vectors from the Baum books are in a different part of the space than those from the Thompson books. It is also clear that RBOO vectors fall in the

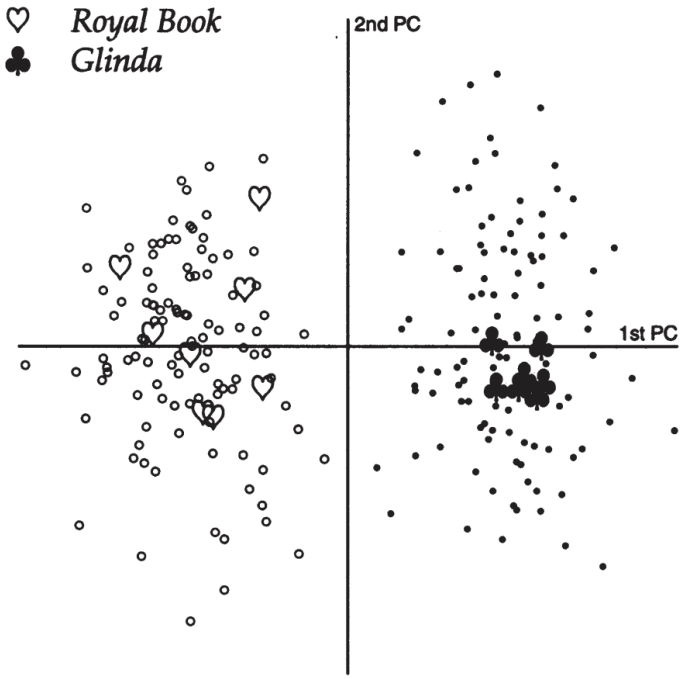

Figure 7.5: Rank-2 visualization of the 223 text block vectors from books of Oz. The dots on the left correspond to vectors from Oz books known to be written by Ruth Plumly Thompson. The hearts on the left correspond to vectors from RBOO. The ones on the right correspond to ones written by L. Frank Baum. Figure from Binongo's paper.

same place as those from other Thompson books. Conclusion: *Ruth Plumly Thompson was the true author of Royal Book of Oz!*

By the way, if one takes the non-Oz writings of Baum and Thompson and plot their vectors in the 2D-visualization in Figure 7.6, they also fall on the appropriate side. So the difference in writing style came across clearly even in non-Oz books!

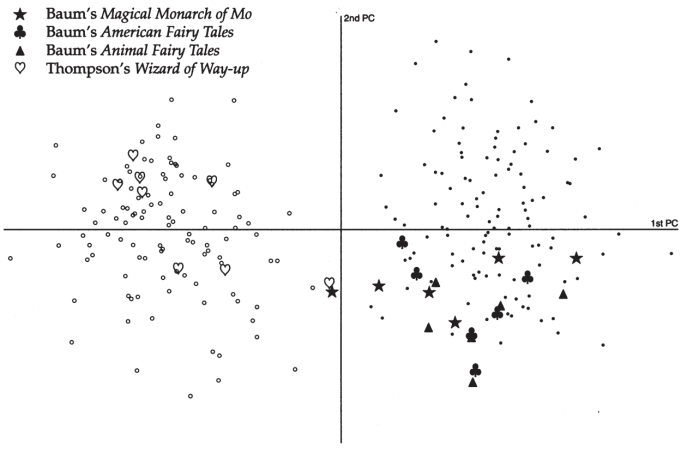

Figure 7.6: Rank-2 visualization of text block vectors from books written by Baum and Thompson outside of the Oz series. Figure from Binongo's paper.

7.3 *Application 2: Eigenfaces*

This section uses the *lossy compression* viewpoint of low-rank representations. As you may remember from earlier computer science courses (*e.g.*, Seam Carver from COS 226), images are vectors of pixel values. In this section, let us only consider grayscale (*i.e.*, B&W) images where each pixel has an integer value in $[-122, 122]$. -122 corresponds to the pixel being pitch black; 0 corresponds to middle gray; and 122 corresponds to total white. We can also reorganize the entries to form a single vector:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \rightarrow (a_{11}, a_{12}, \cdots, a_{1n}, a_{21}, \cdots, a_{2n}, \cdots, a_{mn})$$

Once we have this vectorized form of an image, it is possible to perform linear algebraic operations on the vectors. For example, we can take 0.3 times the first image and add it to -0.8 times the second image, etc. See Figure 7.7 for some of these examples.

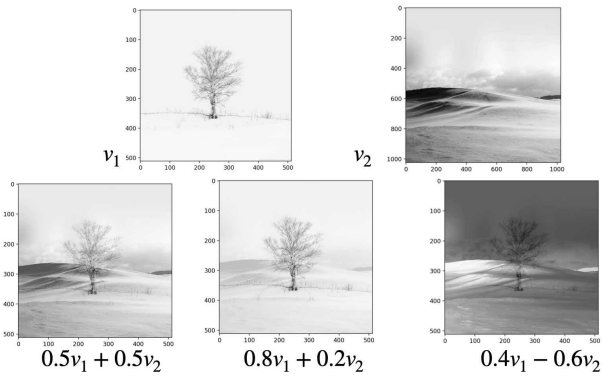

Figure 7.7: Example of linear algebra on images.

Eigenfaces was an idea for face recognition [4]. The dataset in this lecture is from a classic Olivetti dataset from 1990s. Researchers took images of people facing the camera in good light, downsized to 64×64 pixels. This makes them vectors in $\mathbb{R}^{4096}$. Now we can find a 64-rank approximation of the vectors using procedures we will explore more in detail in Section 20.3.

Figure 7.8 shows four basis vectors in the low-rank approximation of the images. The first image looks like a generic human with a ill-defined nose and lips; the second image looks like having glasses and a wider nose; the third image potentially looks like a female face; the fourth image looks like having glasses, a moustache, and a beard. All images in the dataset can be approximated as a linear

[4] L. Sirovich; M. Kirby (1987). *Low-dimensional procedure for the characterization of human faces*. Journal of the Optical Society of America.

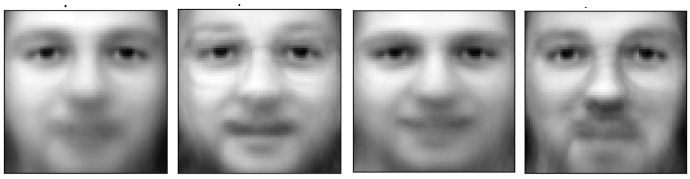

Figure 7.8: Some basis vectors (which turn out to be face-like) in the low-rank approximation of the Olivetti dataset.

combination of 128 of these basis images, and the approximations are surprisingly accurate. Figure 7.9 shows four original images of the dataset, compared with their 64-rank approximations and 128-rank approximations.

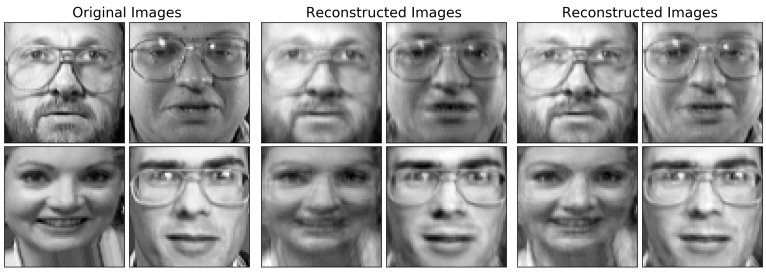

Figure 7.9: 4 original images in the Olivetti dataset (left), compared with their 64-rank approximations (middle) and 128-rank approximations (right).

From Figure 7.9, we also see that the approximations are more accurate when the corresponding value of k is larger. In fact, Figure 7.10 shows the average value of $\frac{\left\|\vec{\mathbf{v}}_i - \widehat{\vec{\mathbf{v}}}_i\right\|_2^2}{\left\|\vec{\mathbf{v}}_i\right\|_2^2}$ as a function of the rank of the approximation. Note that this value roughly represents the *fraction of* $\vec{\mathbf{v}}$ *lost in the approximation*. It can be seen that the error is a decreasing function in terms of k. [5] However, note that doing machine learning — specifically face recognition — on low-rank representations is computationally more efficient particularly because the images are compressed to a lower dimension. With a smaller value of k, we can improve the speed of the learning.

[5] This was also explored in Problem 7.1.5

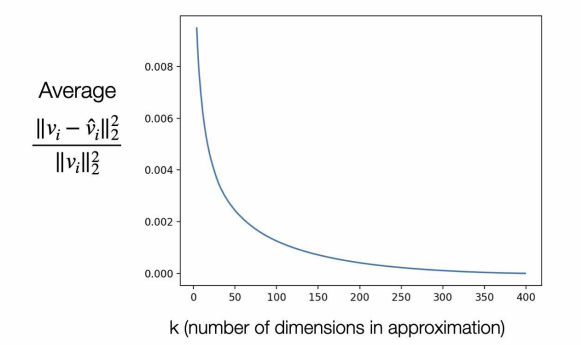

Figure 7.10: What fraction of norm of the image is not captured in the low-dimensional representation, plotted versus the rank k.