CSE 416 Section 9!

Global Pandemic Continues...😵

Just FYI 2021 is 3.5 months away. It’s ok to feel bad for accomplished nothing.

August 13, 2020

HONGJUN JACK WU 😆

Goal for today!

PRINCIPAL COMPONENT ANALYSIS (PCA)!

AS HUMANLY UNDERSTANDABLE AS POSSIBLE!

WHAT WE GOT FOR YOU TODAY

Helpful Notebook:

[Principal Component Analysis]

Helpful Article:

[A One-Stop Shop for Principal Component Analysis].

Critic cat
is not impressed by the artist's utter lack of dimensionality

This presentation:

Has a lot of optional (idk if we have time) but very helpful information.

Feel free to skim through what I prepared in this presentation.

And I will miss ya all. Stay safe and healthy!

Principal Component Analysis: What? Why? When?

(PART I)

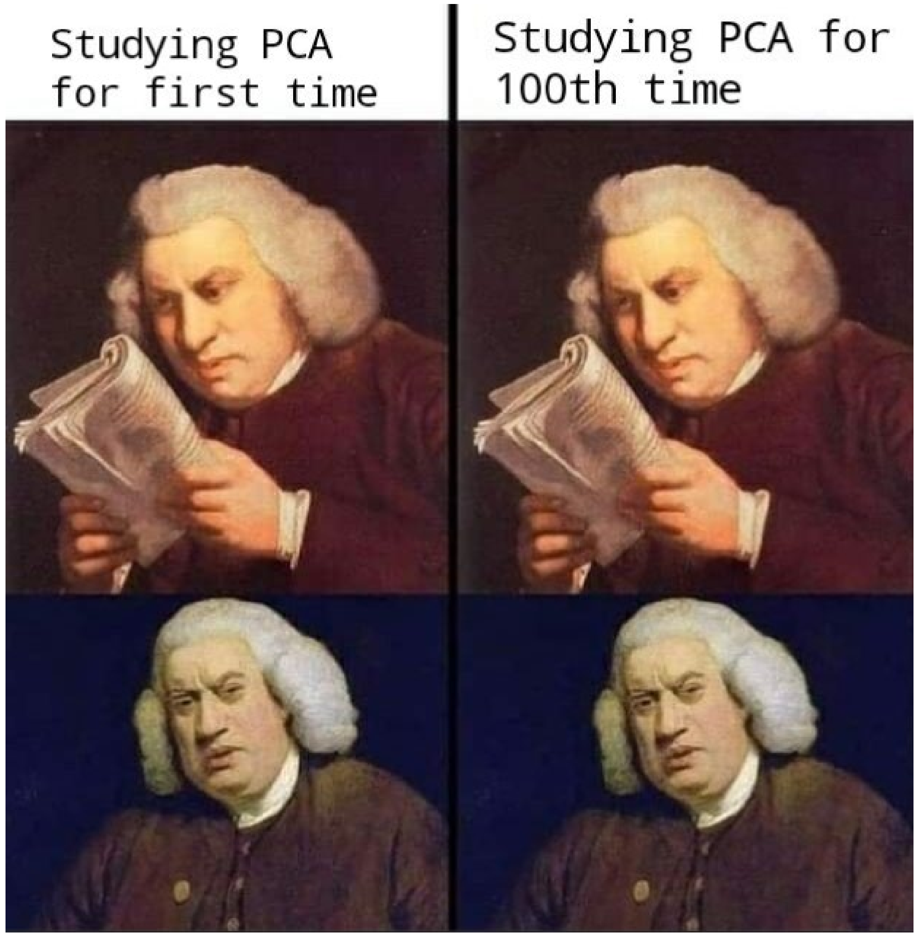
PCA – WHAT IS IT?

It's that time again when I try to convince you why you should even care about what I am about to teach you today…

So why should you even care about PCA?

Well, PCA is a useful tool that sounds like feature elimination, but has a bit more magic than simply eliminating original features.

Introducing… Feature Extraction.

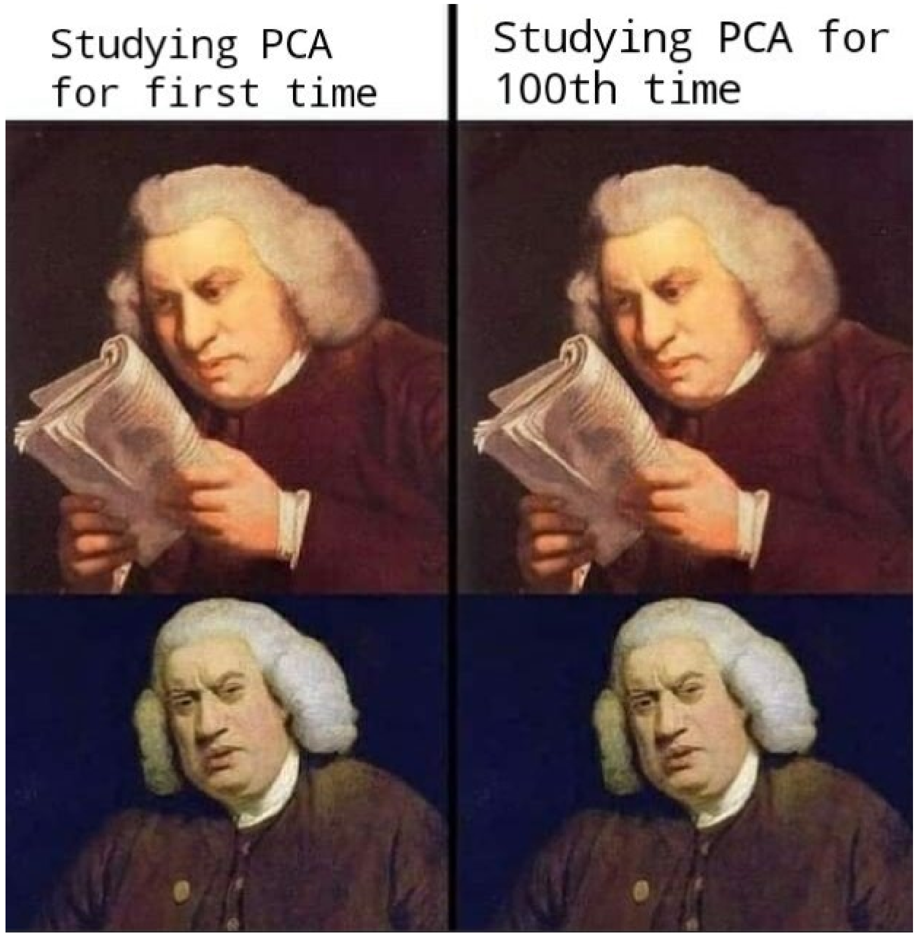


PCA – WHAT IS IT?

I should mention, that even though the material today requires no knowledge of materials that aren't in the prerequisite of this class, further mathematical (linear algebra: eigenvalues, orthogonal matrices, etc) knowledge is needed to understand how PCA works as a **mathematical algorithm**.

Today, however, I'll focus on explaining the mechanics of PCA without getting too much math involved.

You are strongly encouraged to study more linear algebra and dig deeper into PCA beyond this class's scope in the future! It's fun.



PCA – WHY IS IT?

Suppose we are on our usual ML routine:

Step one: We need a problem to solve.

This time, an ambitious one.

We all heard of the gross domestic product (GDP) as a good indicator of how a country is doing, let's imagine we use machine learning to predict: **What the GDP of the United States will be in 2021?**

Now, where should we start?

The Fellowship at 100% strength
The Fellowship at 99% strength

PCA – WHY IS IT?

Now, where should we start?

We will start looking at some data, i.e. analyzing what features are important for us to solve the problem.

You have lots of information available:

The U.S. GDP for the first quarter of 2017, for the entirety of 2016, 2015, etc.

Any publicly-available economic indicator: unemployment rate, inflation rate, and so on.

U.S. Census data from 2010 estimating how many Americans work in each industry and American Community Survey data updating those estimates in between each census.

How many members of the House and Senate belong to each political party.

Stock price data.

Number of IPOs occurring in a year, and how many CEOs seem to be mounting a bid for public office.

PCA – WHY IS IT?

Now, where should we start?

We will start looking at some data, i.e. analyzing what features are important for us to solve the problem.

You have lots of information available:

You got a lot of variables to consider.

PCA – WHY IS IT?

Recall from previous homework and stuff you learned so far:

What is the one word when you have **TOO many variables** involved in your model?

Do you have so many variables that you are in danger of **overfitting** your model to your data?

Do you understand the **relationships** between each variable?

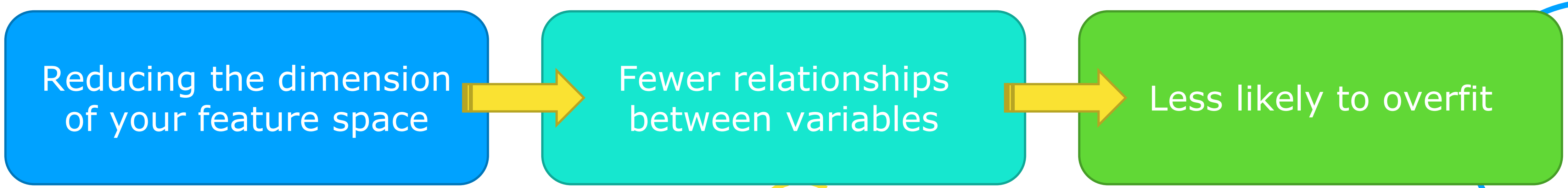
Trouble.

PCA – WHY IS IT?

So, the question becomes:

“How do I take all of the variables I’ve collected and **focus on only a few** of them?”

Aka… reduce the dimension of the features we are using.


Reducing the dimension of your feature space
Fewer relationships between variables
Less likely to overfit

PCA – WHY IS IT?

Somewhat unsurprisingly:
Reducing the ***dimension*** of the feature space is called "***dimensionality reduction***."

There are many ways to achieve dimensionality reduction, but most of these techniques fall into one of two classes:

Feature Elimination

Feature Extraction

FEATURE ELIMINATION

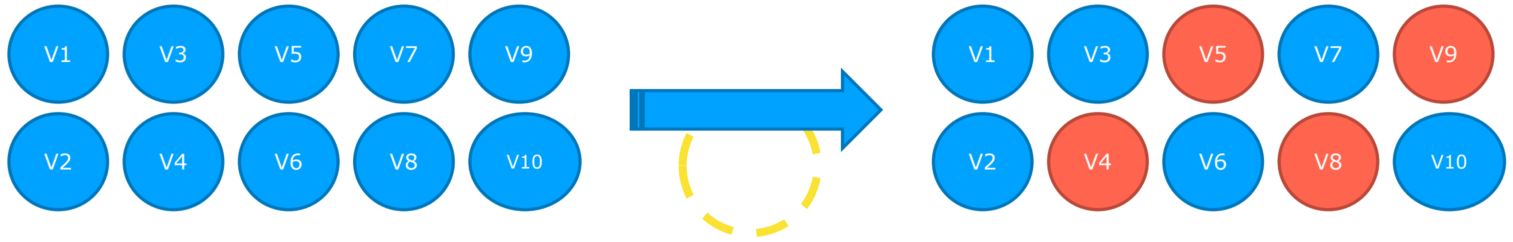
Feature elimination: we reduce the feature space by eliminating features.

Advantages:

It is simple!
It also maintains the interpretability of the variables.

Disadvantages:

You gain no information from those variables you've dropped.
aka it entirely eliminated any benefits those dropped variables would bring.



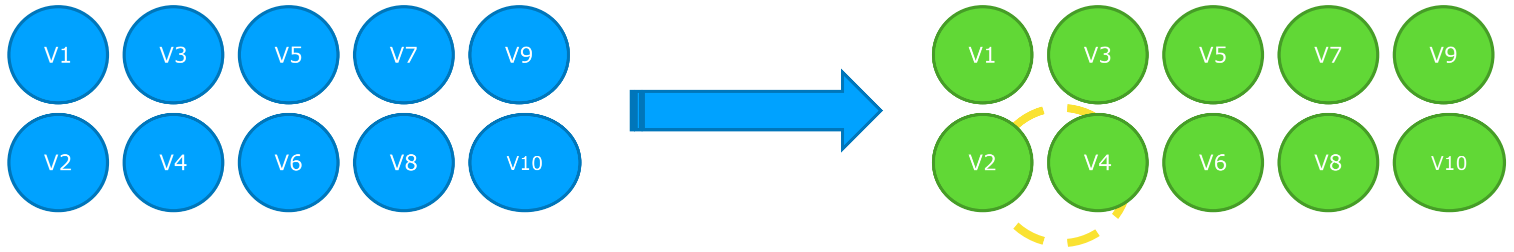
FEATURE EXTRACTION

Feature extraction, however, doesn't run into this problem.

Say we have ten independent variables.

In feature extraction, we create ten *"new"* independent variables, where each *"new"* independent variable is a combination of each of the ten *"old"* independent variables.

However, we create these new independent variables in a specific way and order these new variables by how well they predict our dependent variable.

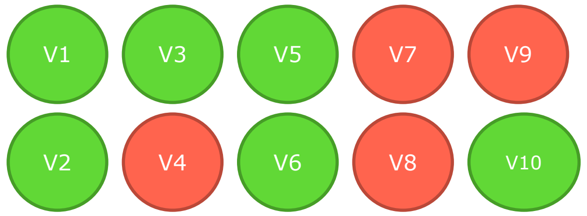


FEATURE EXTRACTION

We create these new independent variables in a specific way and order these new variables by how well they predict our dependent variable.

Now! The big part!!!

We keep as many of the new independent variables as we want, but we drop the "least important ones."



Okay, sounds pretty similar to feature elimination, but!!!

Because these new independent variables are combinations of our old ones, we're still keeping the most valuable parts of our old variables, even when we drop one or more of these "new" variables!

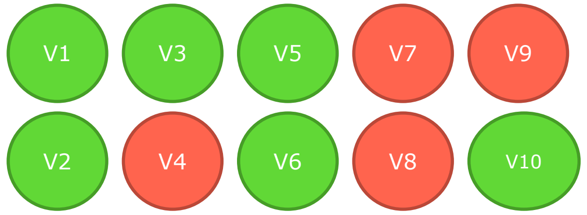
Well, unsurprisingly, **PCA is a technique for *feature extraction!***

FEATURE EXTRACTION

As an added benefit, ***each of the "new" variables after PCA are all independent of one another.***

This is a benefit because the assumptions of a linear model require our independent variables to be independent of one another.

If we decide to fit a linear regression model with these "new" variables, this assumption will necessarily be satisfied.



WHEN SHOULD I USE PCA?

1. Do you want to reduce the number of variables, but aren’t able to identify variables to completely remove from consideration?

2. Do you want to ensure your variables are independent of one another?

3. Are you comfortable making your independent variables less interpretable?

If you answered “yes” to all three questions, then PCA is a good method to use. If you answered “no” to question 3, you **should not** use PCA.

PCA in 2D

(PART II)

(COVERED IN CLASS, REVIEW IF HAVE TIME)

HOW DOES PCA WORK…IN A NUTSHELL?

Well, turns out I can only explain the big idea of PCA without going too deep into linear algebra.

Here’s how it works, in one slide:

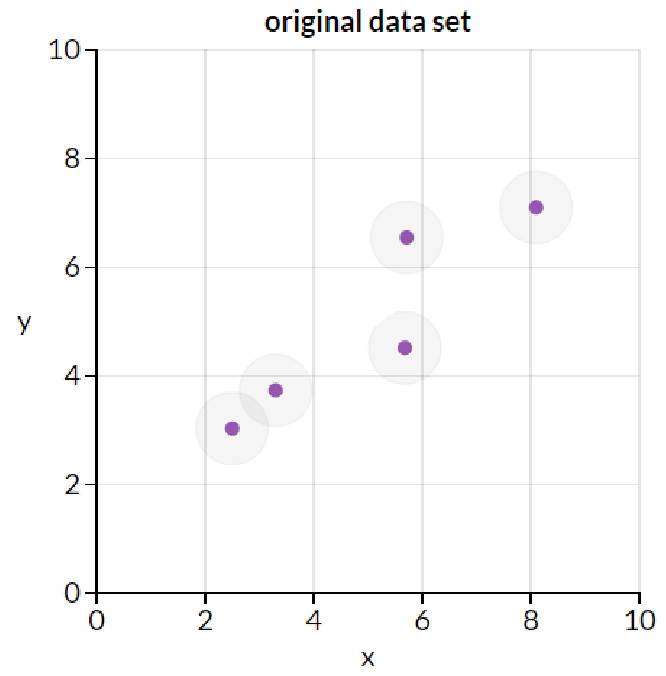
1. We take some data, we compute a Covariance Matrix (just know that it describes how variables relate to one another).
2. We take that matrix, and break this matrix down into two separate components: direction and magnitude.
3. We find the “directions” that is the most important because it gives the lowest reconstruction error.
4. We will transform our original data to align with the important directions.

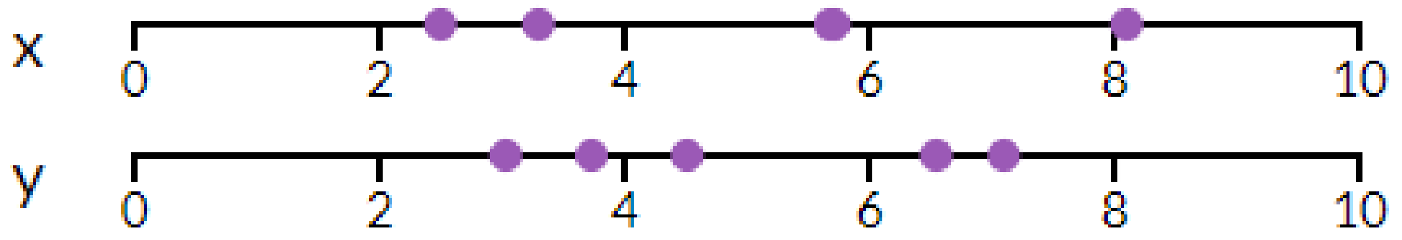
HOW DOES PCA WORK…IN A NUTSHELL?

Say I have a dataset like this:

If we are going to simply do dimensionality elimination (aka from 2D to 1D, to a line), we will simply project the dataset along its x-axis or y-axis.

Does this capture most of the information in the original dataset? Not really, right?

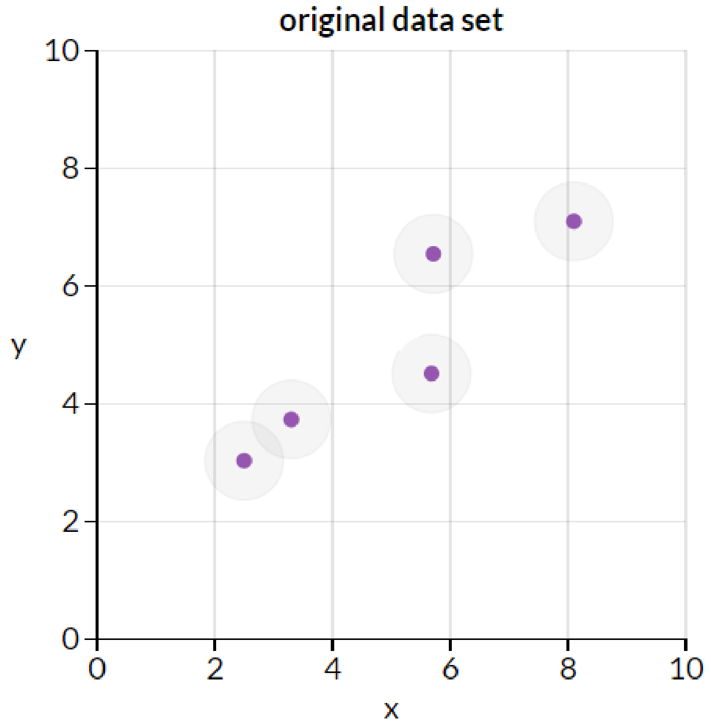




HOW DOES PCA WORK…IN A NUTSHELL?

Again, say I have a dataset like this:

Using what I just said, if you are going to look for a "important direction", where would you want the directions to be?

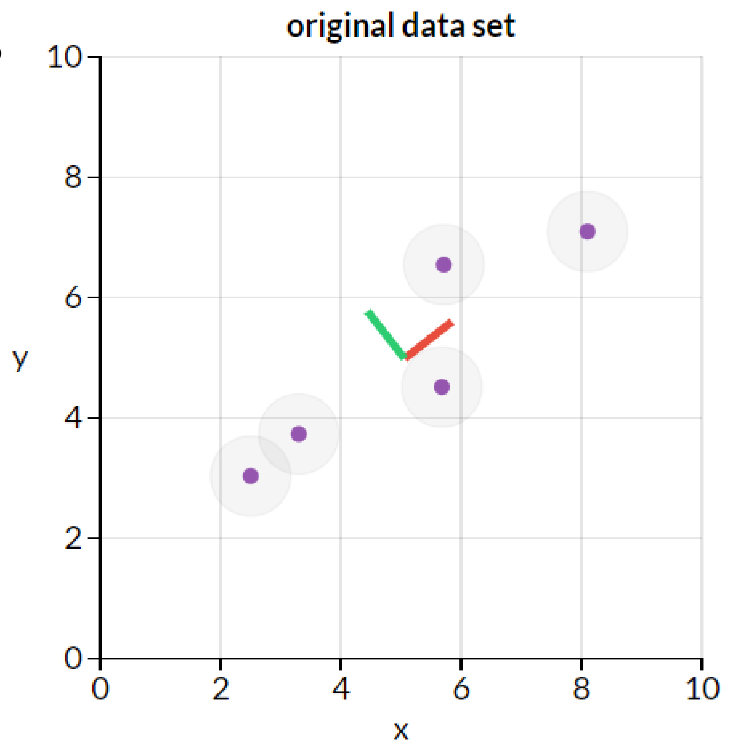


HOW DOES PCA WORK…IN A NUTSHELL?

Say I have a dataset like this:

And I also marked out two directions, the “red direction” and “green direction”.

Which one is more important?

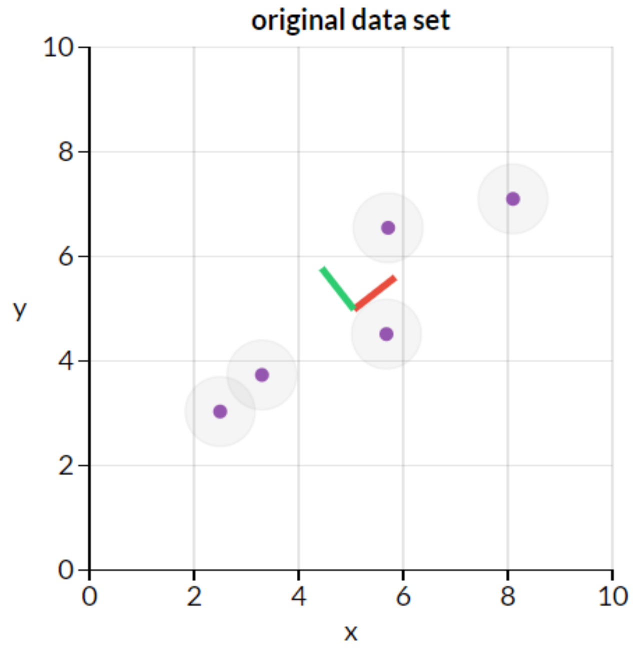


HOW DOES PCA WORK…IN A NUTSHELL?

Which one is more important?

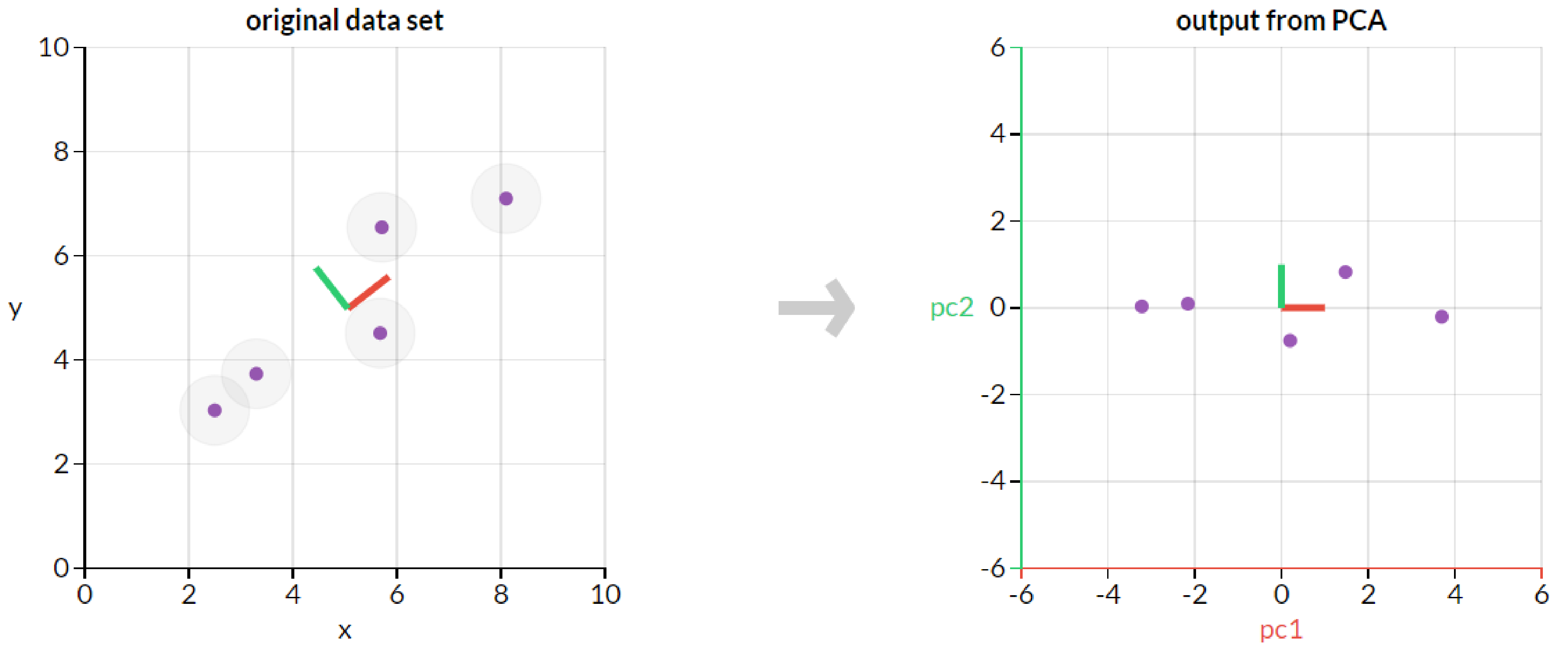
It’s the red direction.

Why? Because it provides the least reconstruction error.



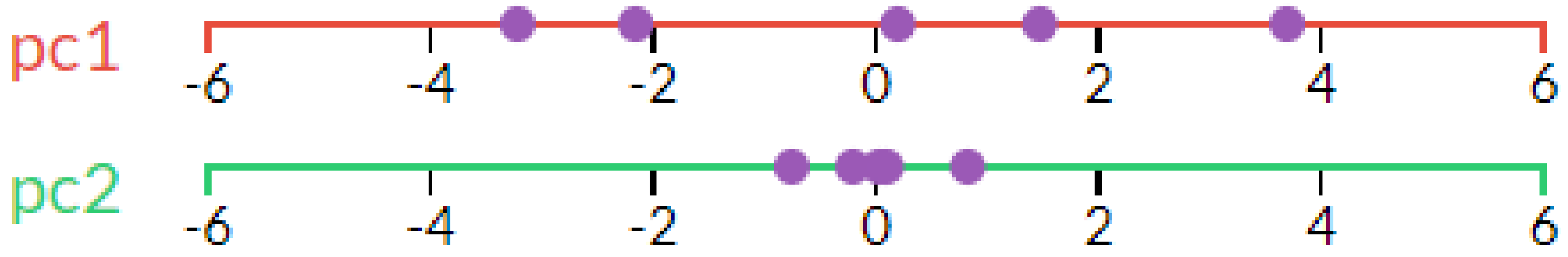
HOW DOES PCA WORK…IN A NUTSHELL?

We will transform our original data to align with these important directions (which are combinations of our original variables).



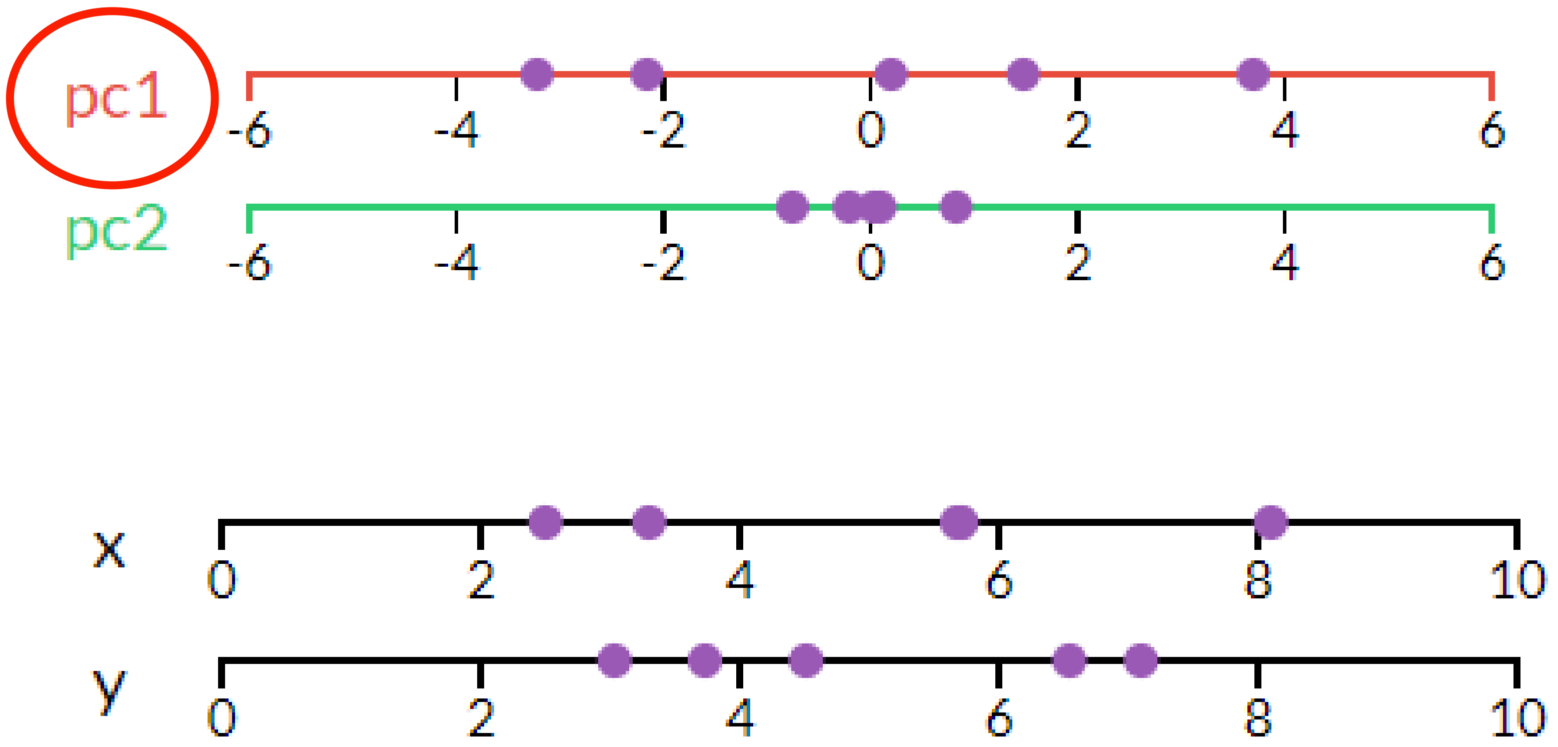
HOW DOES PCA WORK…IN A NUTSHELL?

Now, if we do dimensionality reduction on this aligned data (aka turn 2D into 1D), We get:



HOW DOES PCA WORK…IN A NUTSHELL?

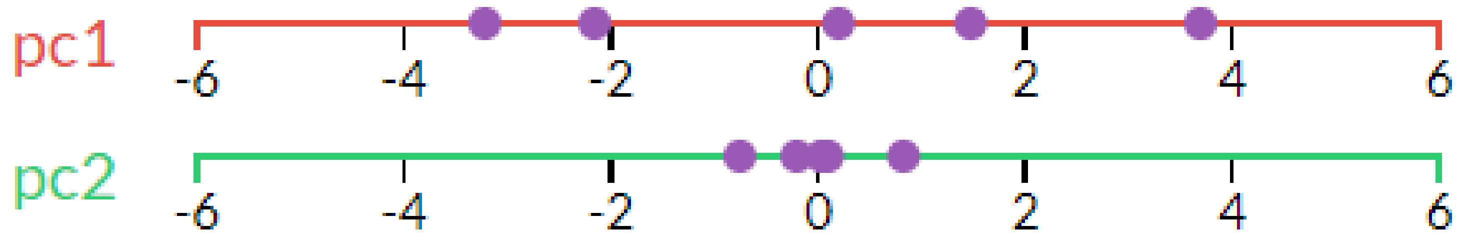
Compare:



HOW DOES PCA WORK…IN A NUTSHELL?

If we are going to reconstruct the original data, because pc1 captures more information of the spread of the original data than pc2.

We will have lower reconstruction error compare to using the original data and do feature elimination.



HOW DOES PCA WORK…IN A NUTSHELL?

By identifying which “directions” are most “important,” we can compress or project our data into a smaller space by dropping the “directions” that are the “least important.”

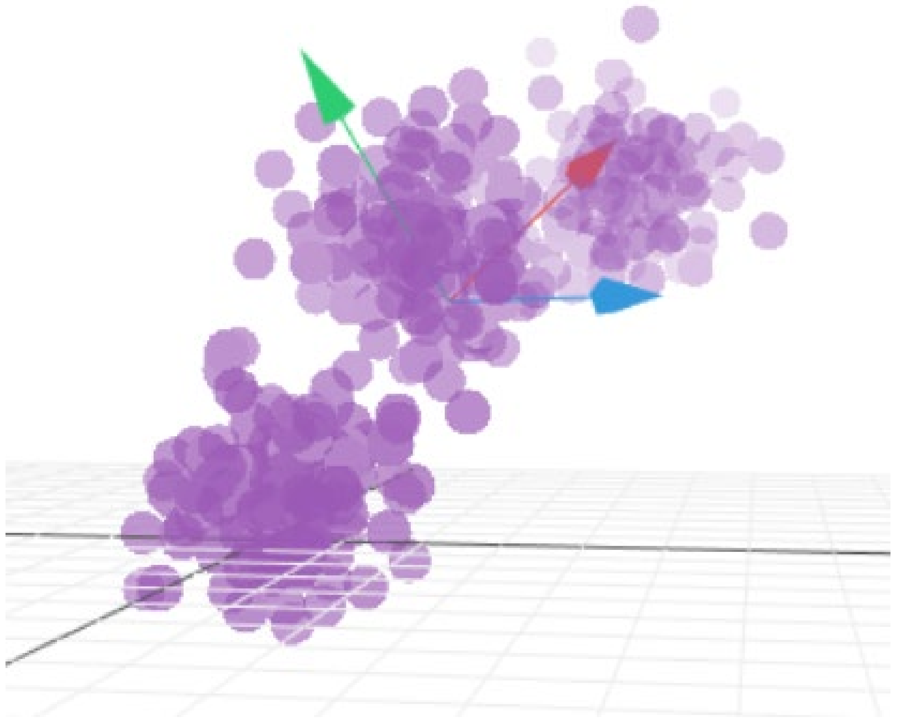
By projecting our data into a smaller space, we’re reducing the dimensionality of our feature space…

But because we’ve transformed our data in these different “directions,” we’ve made sure to keep all useful information in our model!

HOW DOES PCA WORK…IN A NUTSHELL?

PCA Visualization in 2D & 3D:

Principal Component Analysis Explained Visually, Victor Powell and Lewis Lehe



PCA in Higher Dimension

Eating in the UK (a 17D example)

(PART III)

PCA IN HIGHER DIMENSION

PCA can help us quickly gain insight on a very complex dataset.

Say we have a dataset of 17 types of food in grams per person per week for every country in the UK.

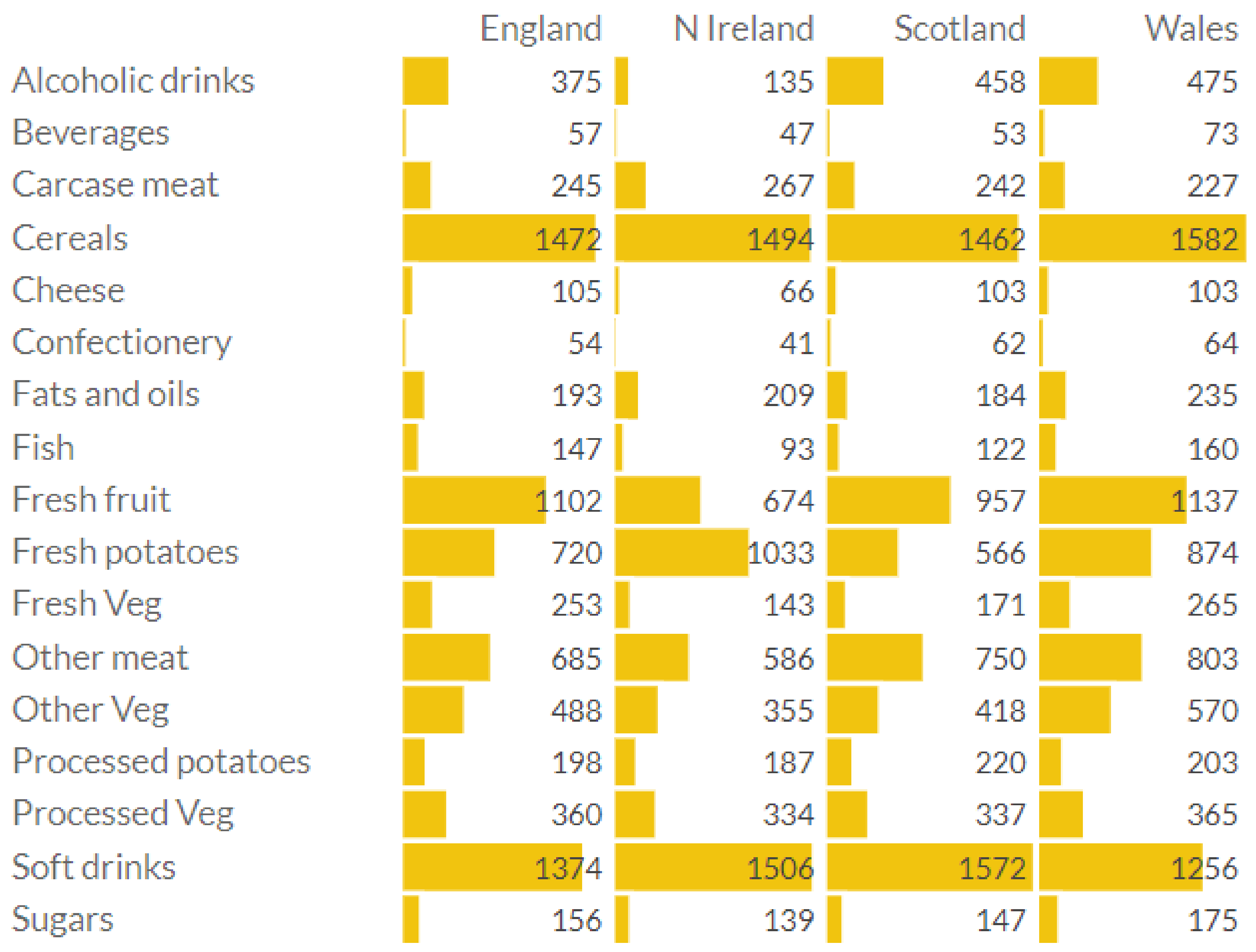
Our goal: Explore how countries in UK differ.

PCA IN HIGHER DIMENSION

Here's the data:

Nothing too obvious, right?

Let's run a PCA to reduce the data from 17D to 1D.



	England	N Ireland	Scotland	Wales
Alcoholic drinks	375	135	458	475
Beverages	57	47	53	73
Carcase meat	245	267	242	227
Cereals	1472	1494	1462	1582
Cheese	105	66	103	103
Confectionery	54	41	62	64
Fats and oils	193	209	184	235
Fish	147	93	122	160
Fresh fruit	1102	674	957	1137
Fresh potatoes	720	1033	566	874
Fresh Veg	253	143	171	265
Other meat	685	586	750	803
Other Veg	488	355	418	570
Processed potatoes	198	187	220	203
Processed Veg	360	334	337	365
Soft drinks	1374	1506	1572	1256
Sugars	156	139	147	175

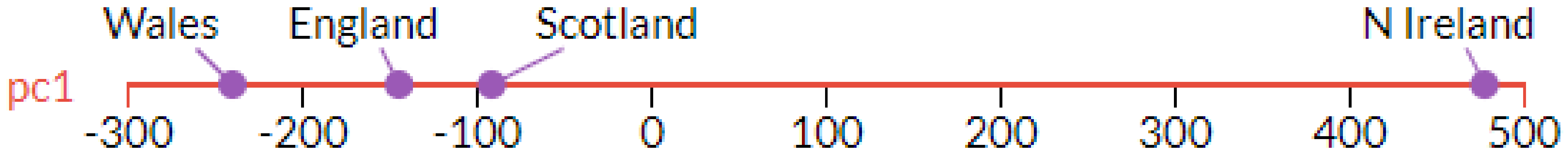
PCA IN HIGHER DIMENSION

Let’s run a PCA to reduce the data from 17D to 1D.

Immediately we can see N. Ireland differs a lot from the rest three countries.

Note:

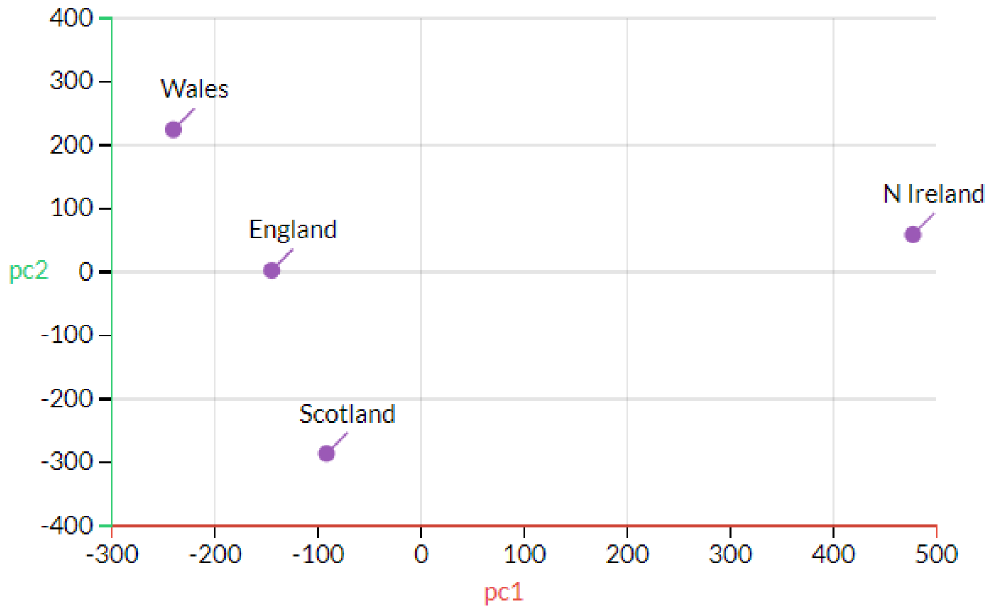
pc1 is a new PCA feature that we can’t necessarily interpret precisely (Recall the third question I asked a while ago: Are you comfortable making your independent variables less interpretable?)



PCA IN HIGHER DIMENSION

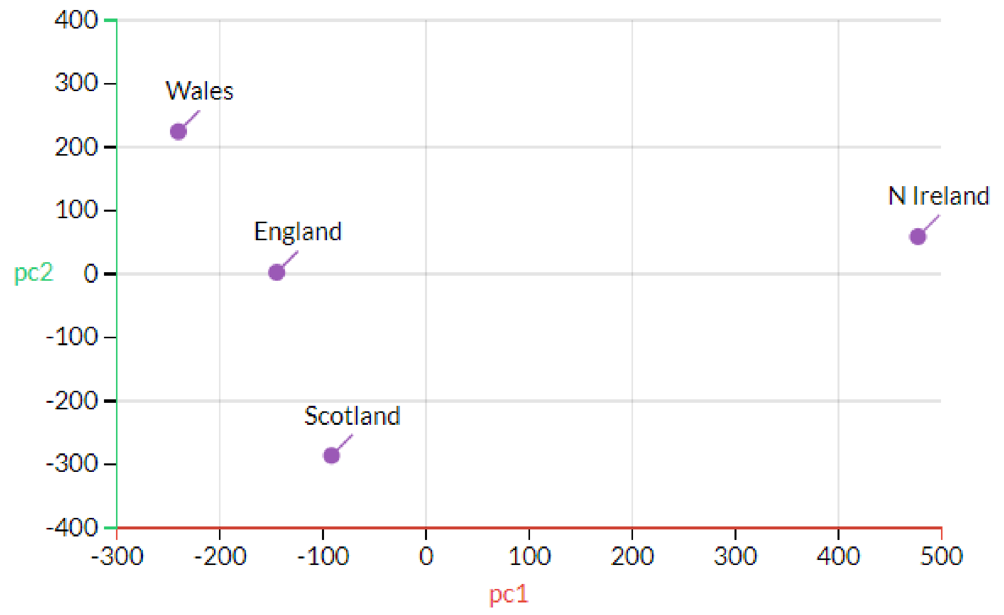
The first PCA feature gives us a big direction:
N. Ireland must have something very different from the other 3 countries in UK in terms of diet.

Now, this time, we run PCA and reduce the dimensionality into 2D.



PCA IN HIGHER DIMENSION

Clearly, we realize that Northern Ireland is a major outlier.

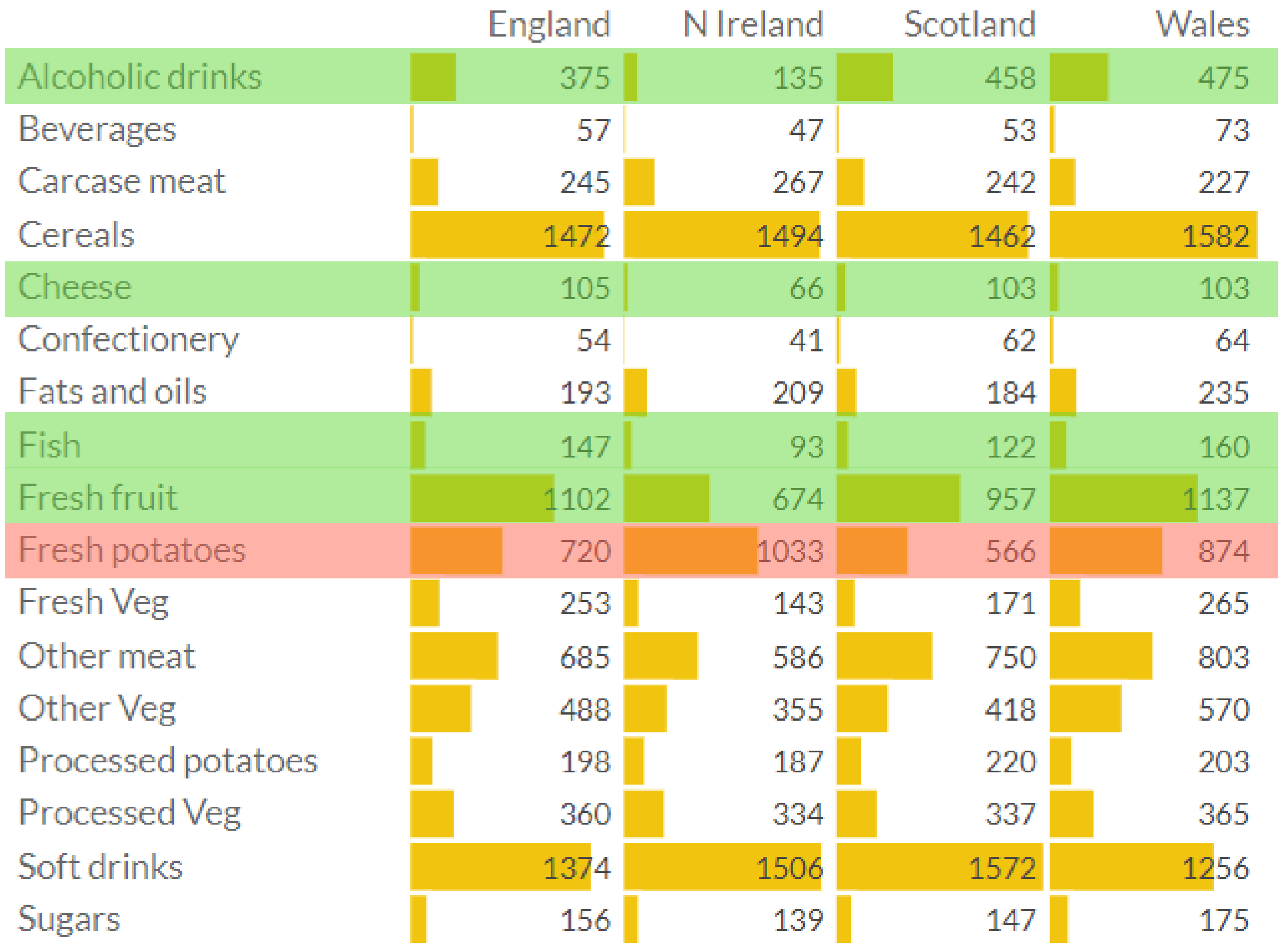


PCA IN HIGHER DIMENSION

Once we go back and look at the data in the table, this makes sense: the Northern Irish eat way more grams of fresh potatoes and way fewer of fresh fruits, cheese, fish and alcoholic drinks.

It's a good sign that structure we've visualized reflects a big fact of real-world geography:

Northern Ireland is the only of the four countries not on the island of Great Britain.



	England	N Ireland	Scotland	Wales
Alcoholic drinks	375	135	458	475
Beverages	57	47	53	73
Carcase meat	245	267	242	227
Cereals	1472	1494	1462	1582
Cheese	105	66	103	103
Confectionery	54	41	62	64
Fats and oils	193	209	184	235
Fish	147	93	122	160
Fresh fruit	1102	674	957	1137
Fresh potatoes	720	1033	566	874
Fresh Veg	253	143	171	265
Other meat	685	586	750	803
Other Veg	488	355	418	570
Processed potatoes	198	187	220	203
Processed Veg	360	334	337	365
Soft drinks	1374	1506	1572	1256
Sugars	156	139	147	175

PCA in Image Processing

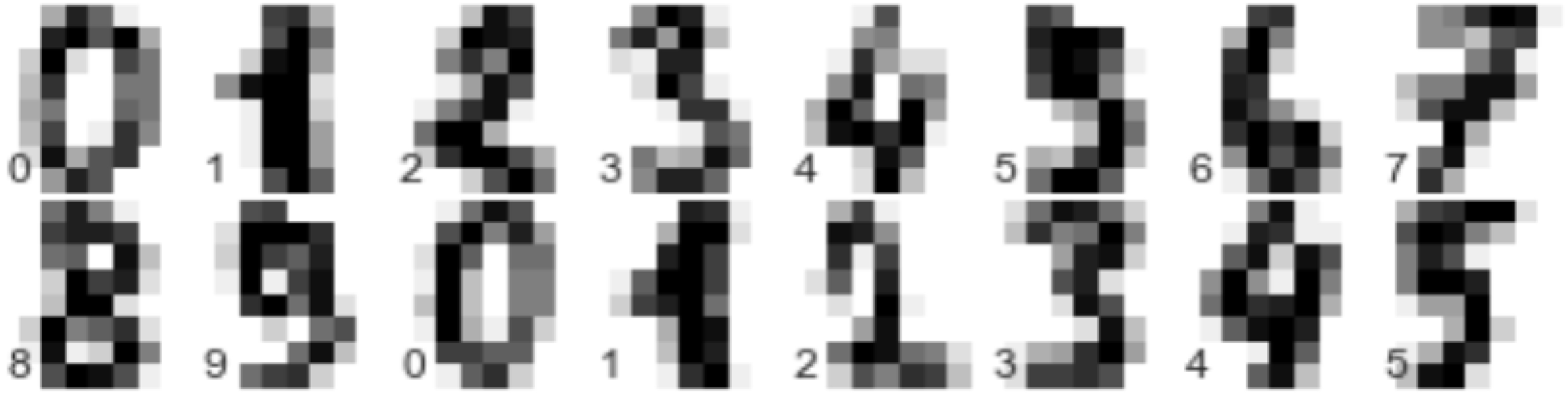
Handwritten digits (a 64D example)

(PART IV)

PCA IN IMAGE PROCESSING

Remember the handwritten digit dataset? (~~aka easy image dataset for noob data scientists~~). Let's explore how running PCA can help us with image processing.

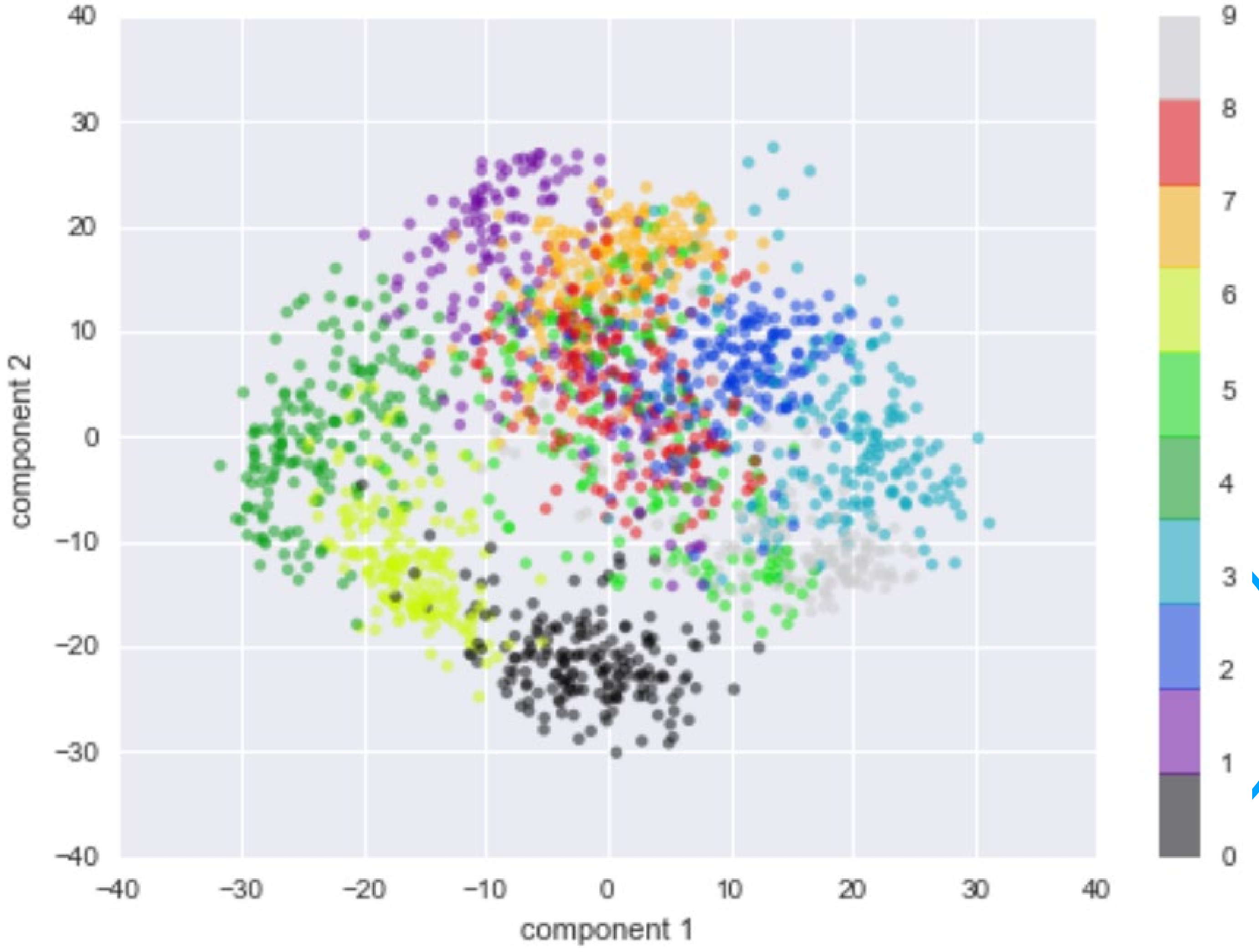
In image processing, the complexity grows EXTREMELY fast as the resolution goes up. For example, what we consider very very low resolution image in the hand written digit dataset (8px by 8px) has 64 dimensions…



PCA IN IMAGE PROCESSING

It is really hard for us, miserable humans who live in a three dimension world (well, 4D if you take time into account) to imagine how our data look like in a 64D space.

So let’s just run PCA and reduce the dimension to 3D, and see what it will provide us.



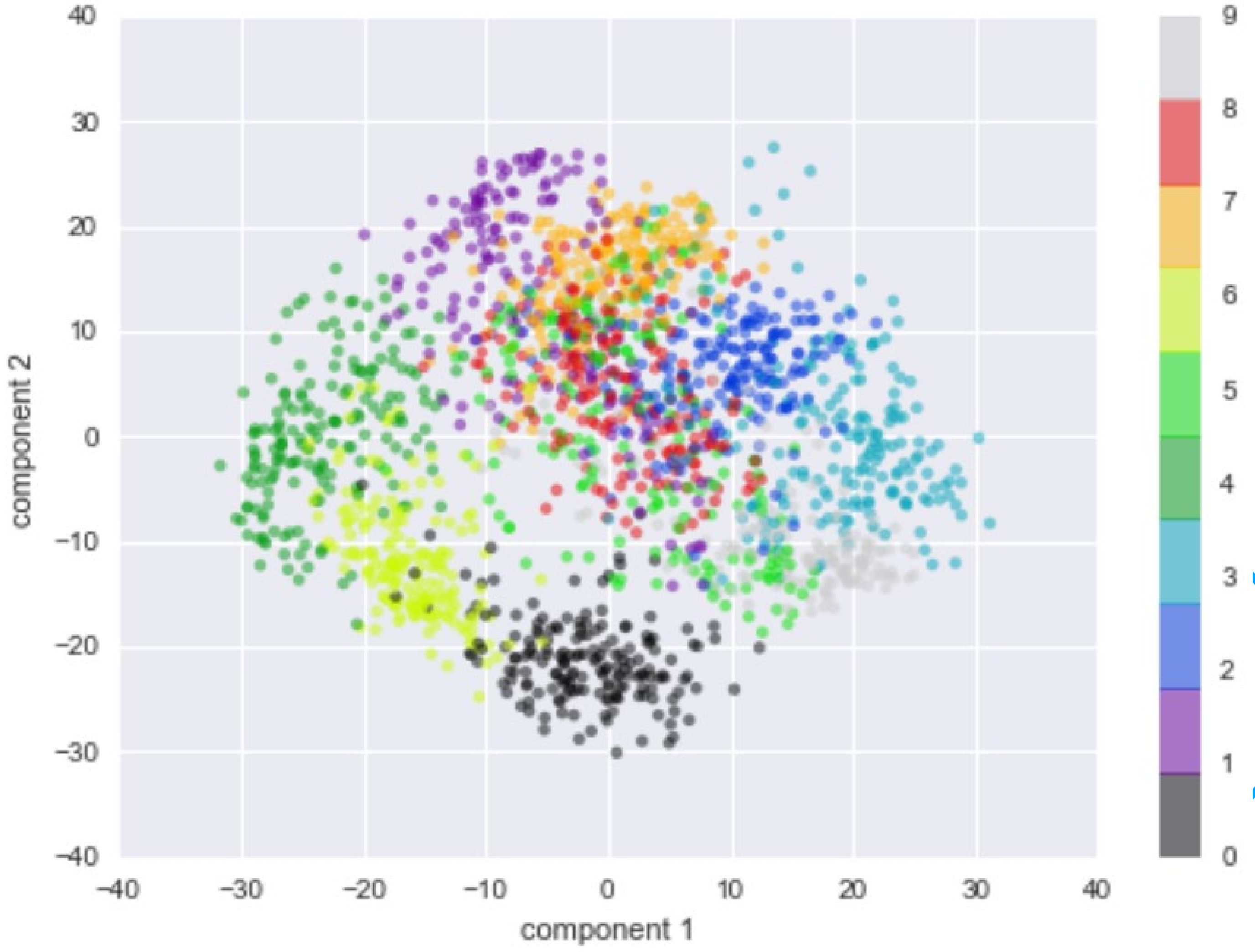
PCA IN IMAGE PROCESSING

Recall what these components mean:

The full data is a 64-dimensional point cloud, and these points are the projection of each data point along the directions with the largest variance.

Essentially, we have found the optimal stretch and rotation in 64-dimensional space.

That allows us to see the layout of the digits in two dimensions, and have done this in an unsupervised manner—that is, without reference to the labels.



PCA IN IMAGE PROCESSING

Let's also take a look at visually how feature elimination differs from feature extraction.

If you think of how we mathematically represent an image:

Say each image in the training set is defined by a collection of 64 pixel values, which we will call the vector x :

$$x = [x_1, x_2, x_3 \cdots x_{64}]$$

To construct the image, we multiply each element of the vector by the pixel it describes, and then add the results together to build the image:

$$\text{image}(x) = x_1 \cdot (\text{pixel } 1) + x_2 \cdot (\text{pixel } 2) + x_3 \cdot (\text{pixel } 3) \cdots x_{64} \cdot (\text{pixel } 64)$$

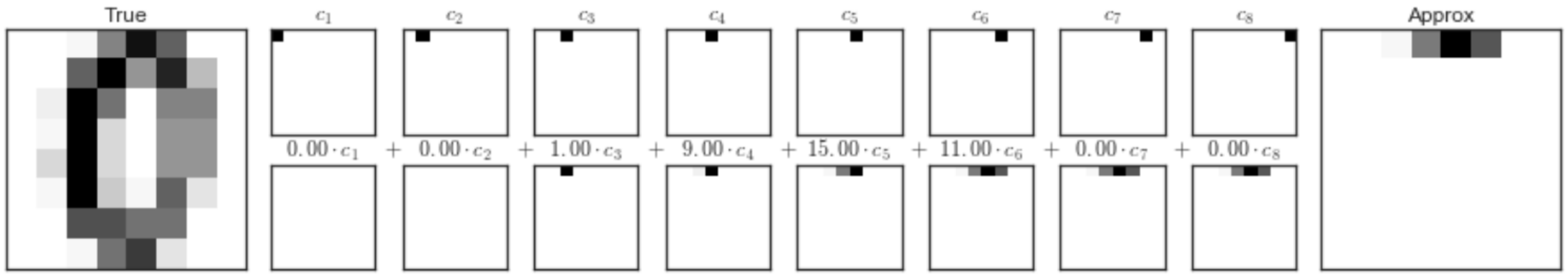
PCA IN IMAGE PROCESSING

If we do feature elimination on a single picture (aka one datapoint) that means we are literally throwing out pixels.

So say we are only going to use the first eight pixels:

That means we are reducing the image into a 8D projection of the 64D data.

Does this look very informative…? Ughhhh no…


True
c_1
c_2
c_3
c_4
c_5
c_6
c_7
c_8
$0.00 \cdot c_1 + 0.00 \cdot c_2 + 1.00 \cdot c_3 + 9.00 \cdot c_4 + 15.00 \cdot c_5 + 11.00 \cdot c_6 + 0.00 \cdot c_7 + 0.00 \cdot c_8$
Approx

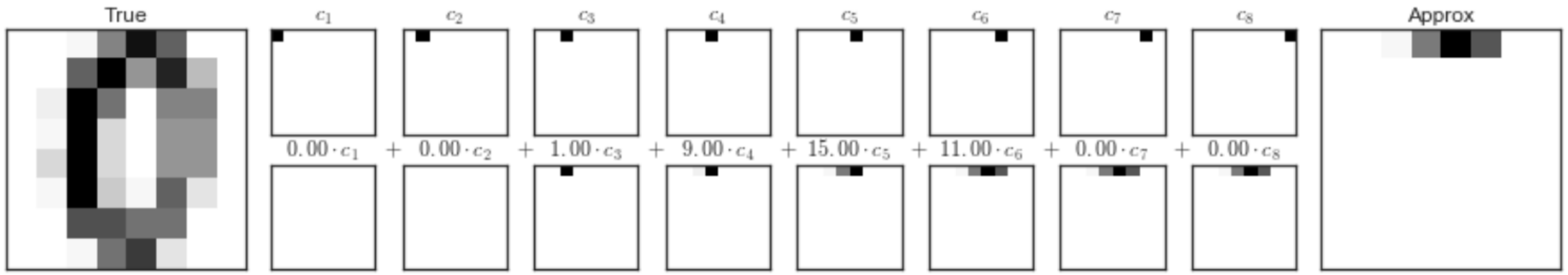
PCA IN IMAGE PROCESSING

The upper row of panels shows the individual pixels.

The lower row shows the cumulative contribution of these pixels to the construction of the image.

Using only eight of the pixel-basis components, we can only construct a small portion of the 64-pixel image.

Obviously, were we to continue this sequence and use all 64 pixels, we would recover the original image.

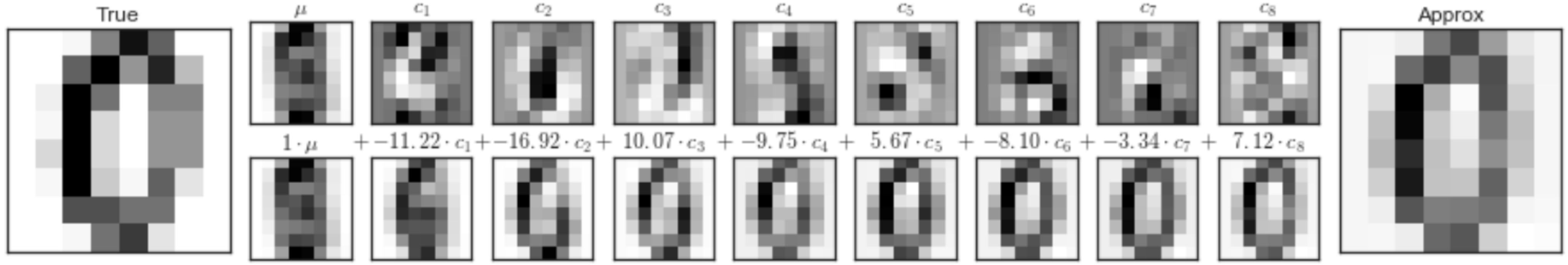

True
c_1
c_2
c_3
c_4
c_5
c_6
c_7
c_8
Approx
$0.00 \cdot c_1 + 0.00 \cdot c_2 + 1.00 \cdot c_3 + 9.00 \cdot c_4 + 15.00 \cdot c_5 + 11.00 \cdot c_6 + 0.00 \cdot c_7 + 0.00 \cdot c_8$

PCA IN IMAGE PROCESSING

In contrary, PCA can be thought of as a process of choosing optimal basis functions, such that adding together just the first few of them is enough to suitably reconstruct the bulk of the elements in the dataset.

$$image(x) = \text{mean} + x_1 \cdot (\text{basis 1}) + x_2 \cdot (\text{basis 2}) + x_3 \cdot (\text{basis 3}) \cdots$$

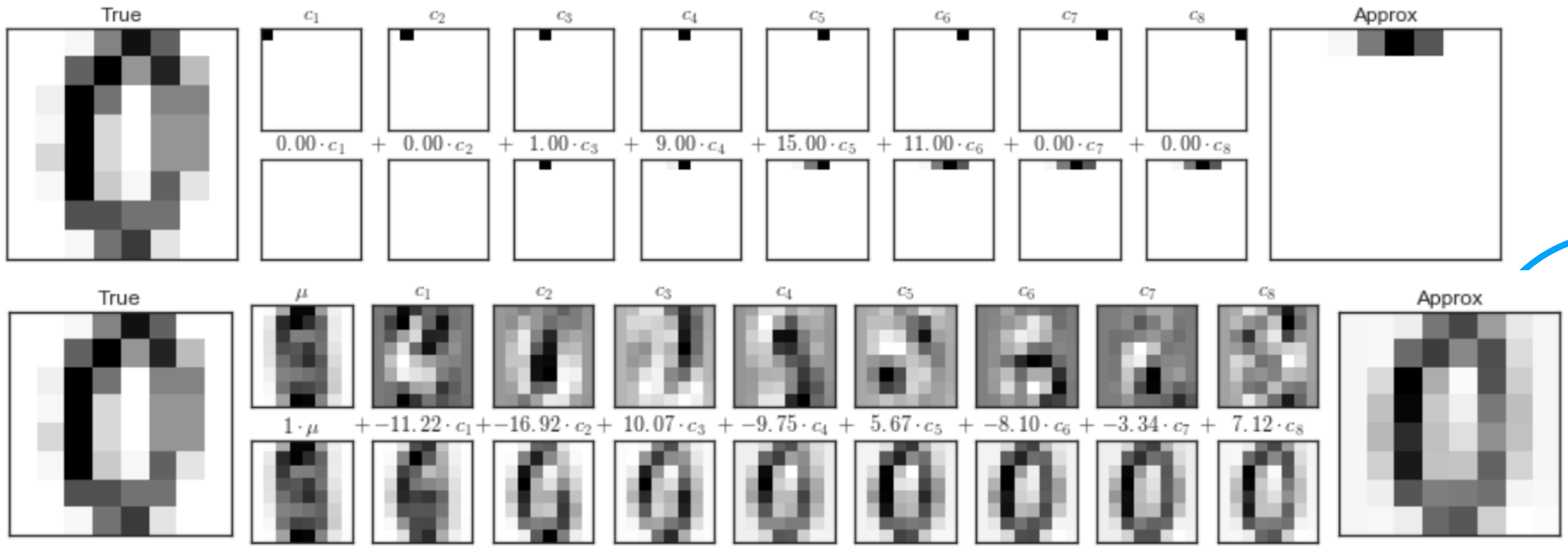
This figure shows a similar depiction of reconstructing this digit using the mean plus the first eight PCA basis functions:



PCA IN IMAGE PROCESSING

Takeaway:

PCA provides a low-dimensional representation of the data: it discovers a set of basis functions that are more efficient than the native pixel-basis of the input data.



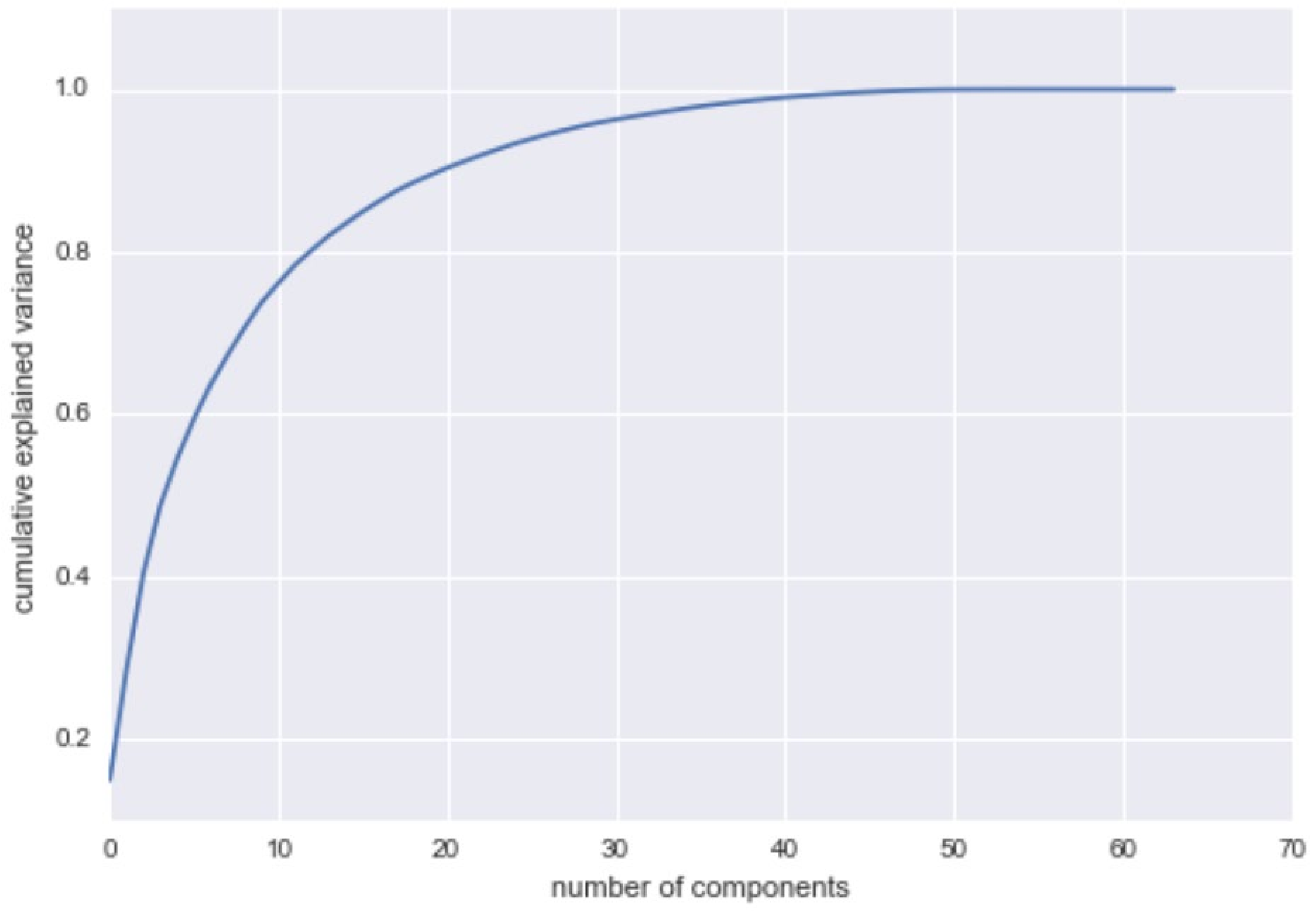
True
c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8
$0.00 \cdot c_1 + 0.00 \cdot c_2 + 1.00 \cdot c_3 + 9.00 \cdot c_4 + 15.00 \cdot c_5 + 11.00 \cdot c_6 + 0.00 \cdot c_7 + 0.00 \cdot c_8$
Approx
True
μ c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8
$1 \cdot \mu + -11.22 \cdot c_1 + -16.92 \cdot c_2 + 10.07 \cdot c_3 + -9.75 \cdot c_4 + 5.67 \cdot c_5 + -8.10 \cdot c_6 + -3.34 \cdot c_7 + 7.12 \cdot c_8$
Approx

CHOOSING THE NUMBER OF COMPONENTS

This is a common question: How many components are needed to describe the data?

This can be determined by looking at the **cumulative *explained variance ratio*** as a function of the number of components:

```
PCA().fit(digits.data).explained_variance_ratio_
```

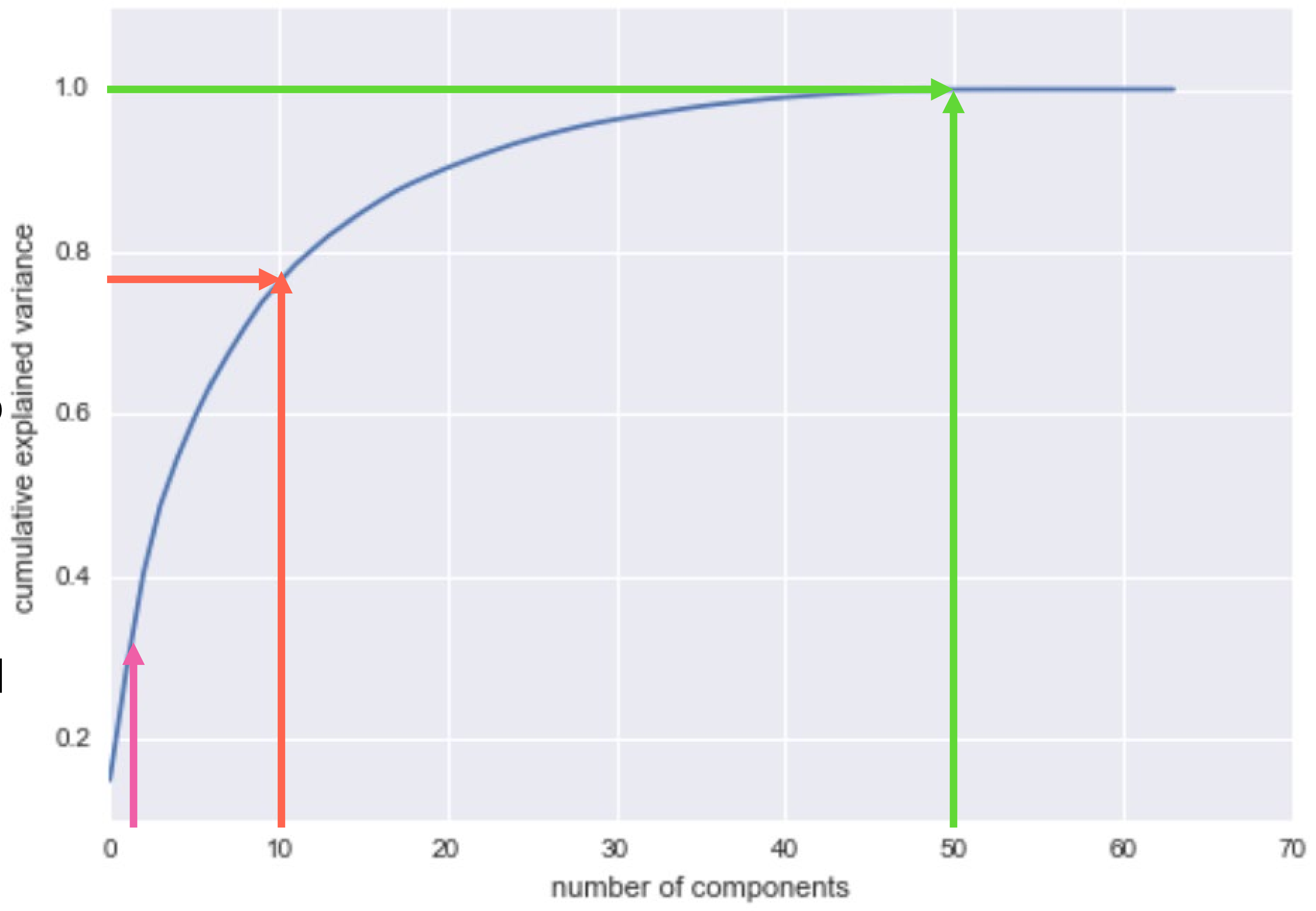


CHOOSING THE NUMBER OF COMPONENTS

This curve quantifies how much of the total, 64-dimensional variance is contained within the first N components.

For example, we see that with the digits the first 10 components contain approximately 75% of the variance, while you need around 50 components to describe close to 100% of the variance.

Here we see that our two-dimensional projection loses a lot of information (as measured by the explained variance) and that we'd need about 20 components to retain 90% of the variance.



PCA in Image Processing

Other Examples:
PCA as Noise Filtering (64D example)

(PART V)

PCA AS NOISE FILTERING

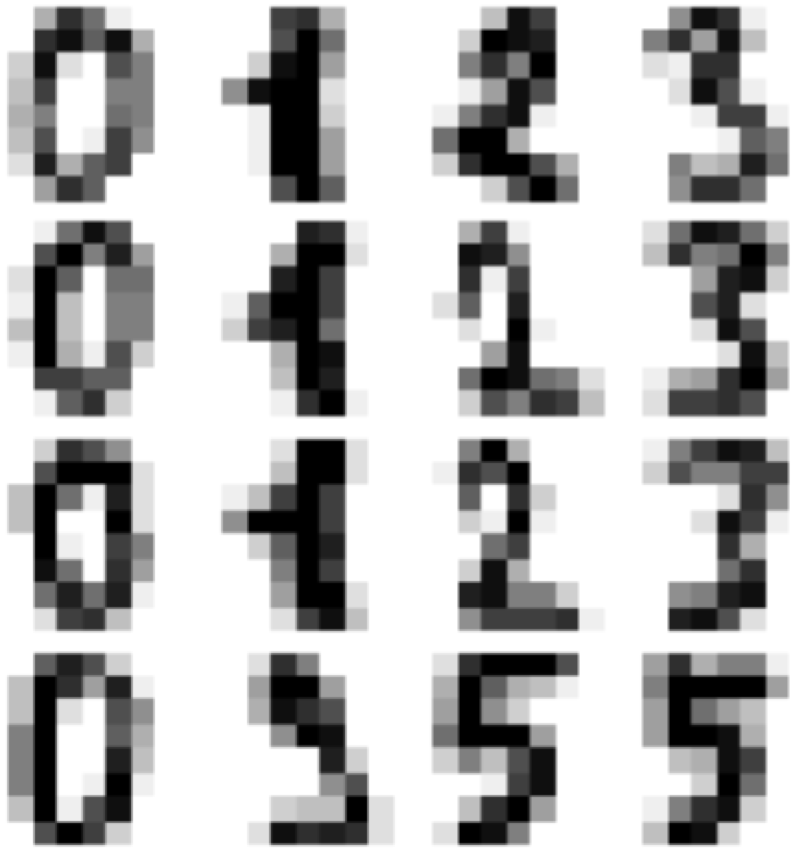
PCA can also be used as a filtering approach for noisy data.

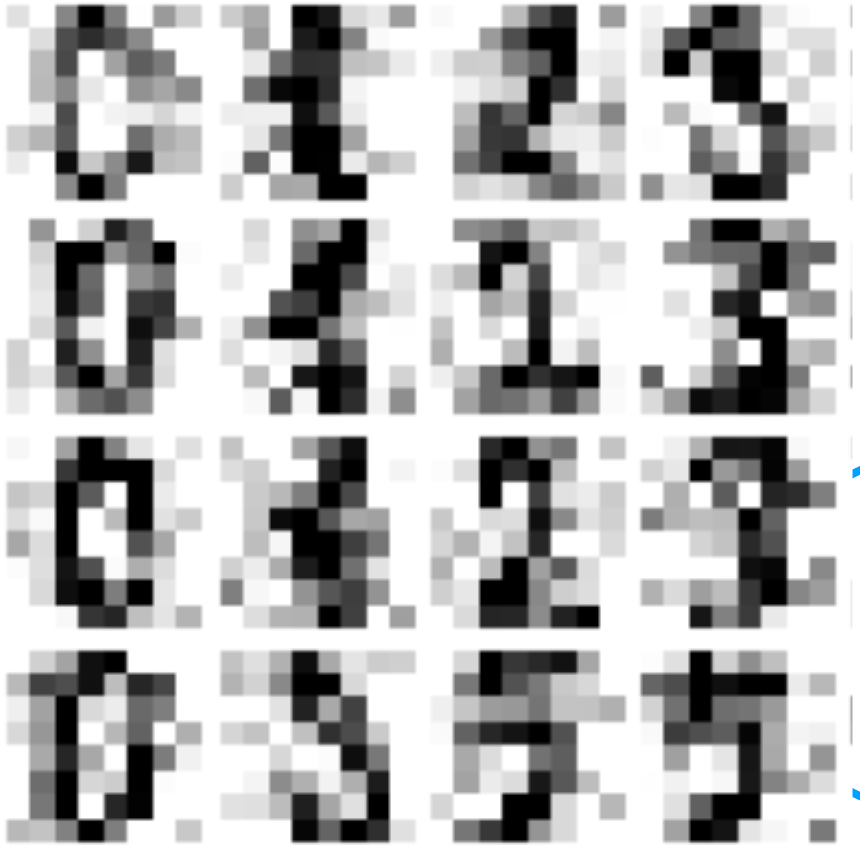
The idea is this:
Any components with variance much larger than the effect of the noise should be relatively unaffected by the noise.

So if you reconstruct the data using just the largest subset of principal components, you should be preferentially keeping the signal and throwing out the noise.

PCA AS NOISE FILTERING

If We add random noise to the data:

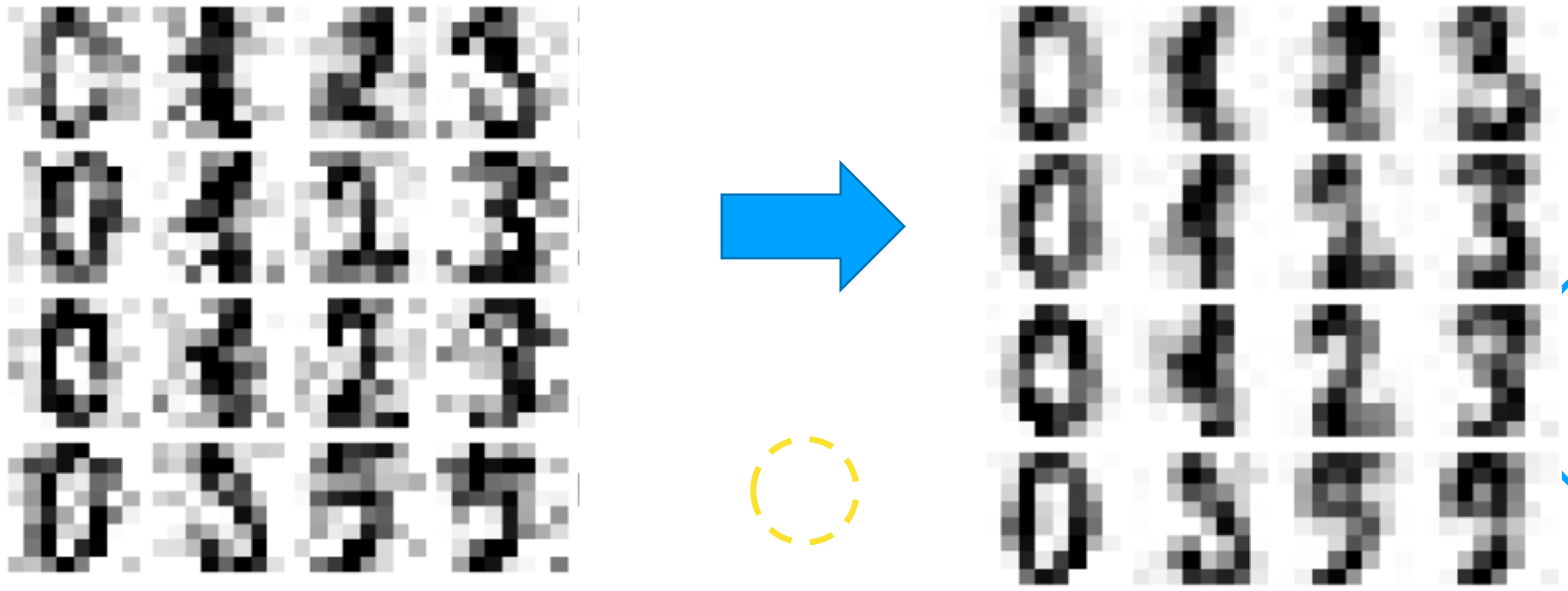




PCA AS NOISE FILTERING

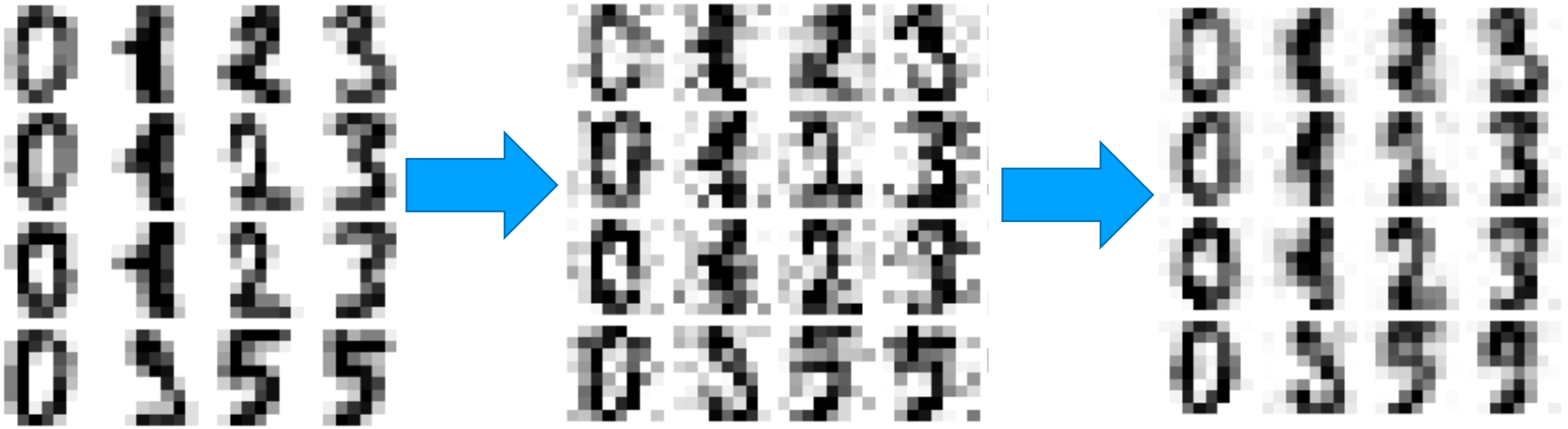
Let's train a PCA on the noisy data, requesting that the projection preserve 50% of the variance (amounts to 12 principal components.)

We then use the inverse of the transform to reconstruct the filtered digits:



PCA AS NOISE FILTERING

Compare: (Original vs. Noise vs. Reconstructed)

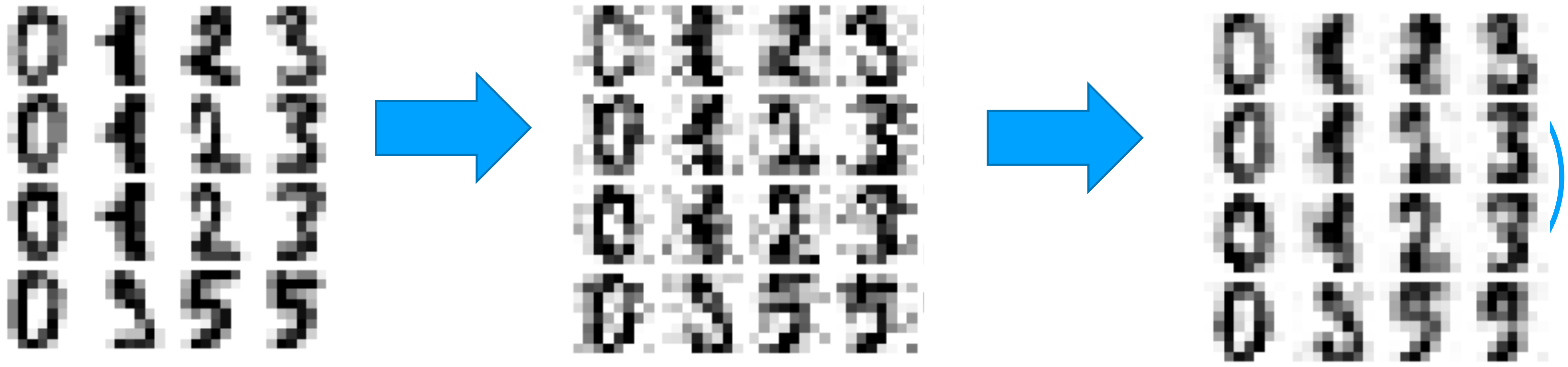


PCA AS NOISE FILTERING

Takeaway:

This signal preserving/noise filtering property makes PCA a very useful feature selection routine.

For example, rather than training a classifier on very high-dimensional data, you might instead train the classifier on the lower-dimensional representation, which will automatically serve to filter out random noise in the inputs.



PCA in Image Processing

Other Examples:
PCA as Eigenfaces (a 3000D example)

(PART VI)

PCA AS EIGENFACES

We did not expect you to know linear algebra, so we won't expect you know what eigenvectors are.

Just some helpful information on more complex image processing. Relax!

Say we have some images of people:

PCA AS EIGENFACES

Term:

We usually call the “**important directions**” principal components (The “PC” in PCA)

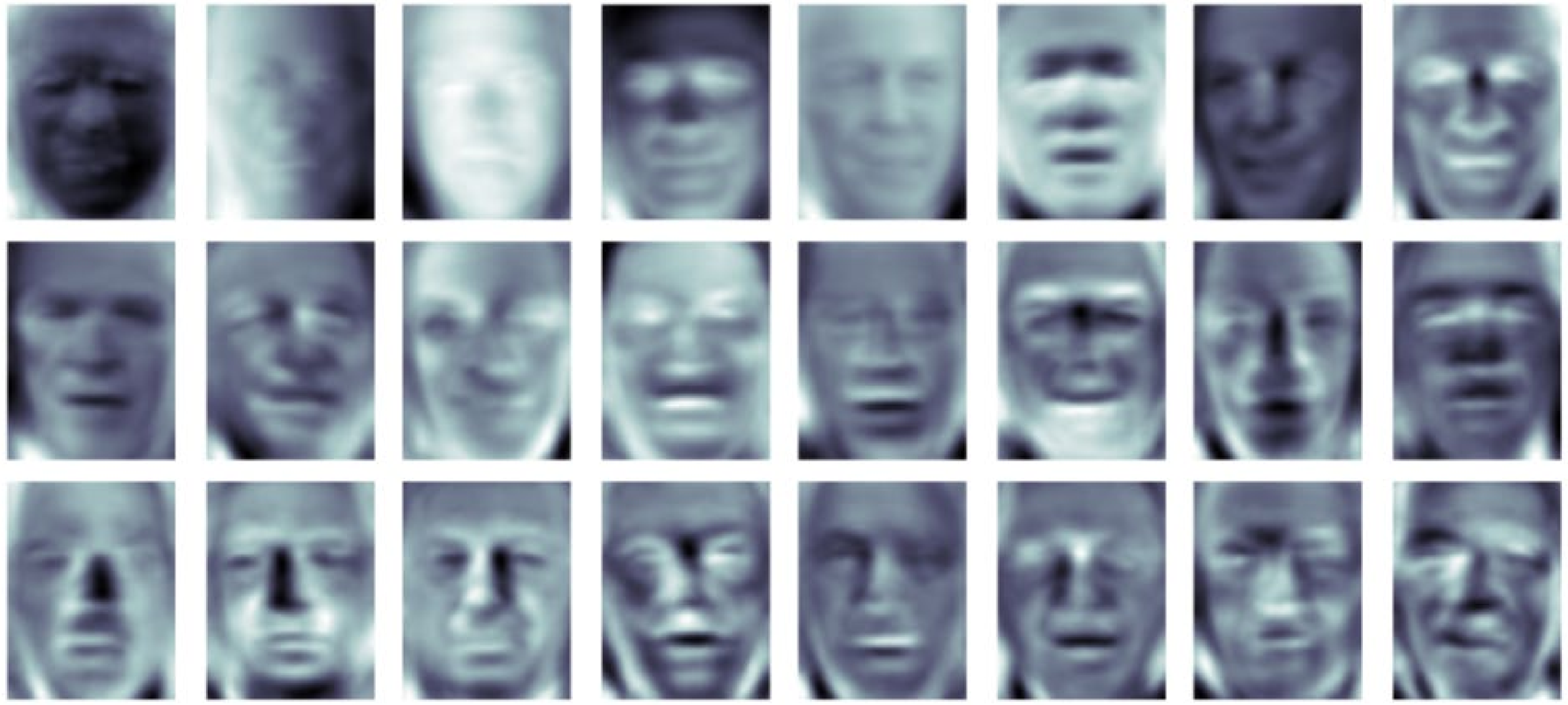
But these components are technically known as "**eigenvectors**" so these types of images are often called "**eigenfaces**‟.

If we visualize the first couple components (aka eigenfaces), they look like this:

Warning:
The next slide contains the image of eigenfaces which is kinda creepy imo but they contain useful information. Not trying to scare you!

PCA AS EIGENFACES

If we visualize the first couple components (aka eigenfaces), they look like this:

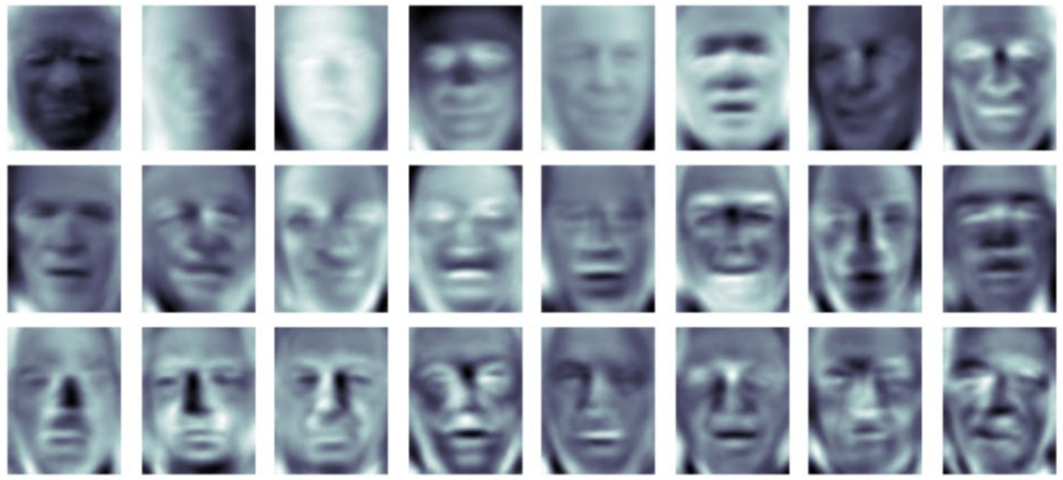


PCA AS EIGENFACES

Some insights:

The first few eigenfaces (from the top left) seem to be associated with the angle of lighting on the face…

Later principal vectors seem to be picking out certain features, such as eyes, noses, and lips.

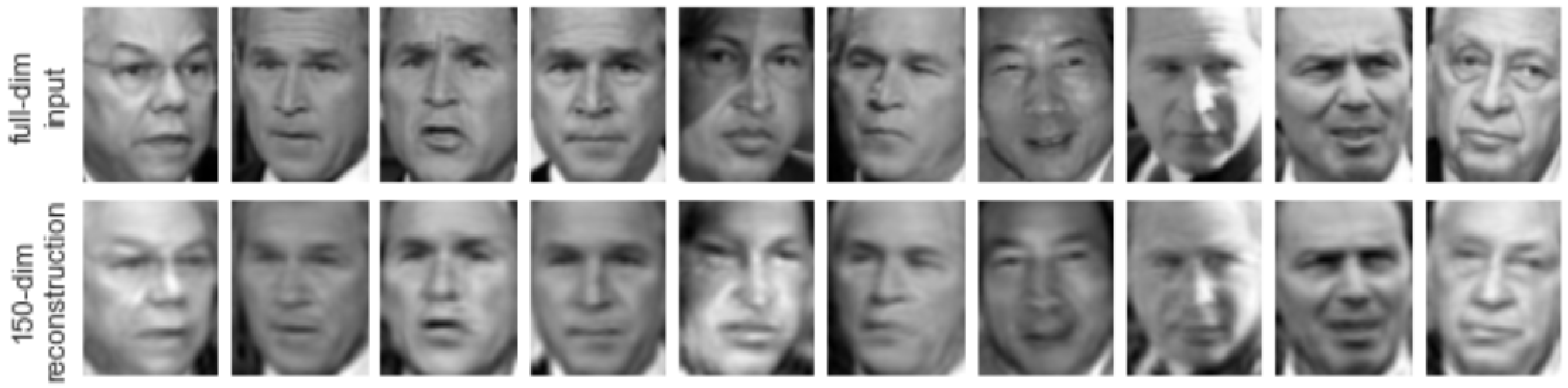


PCA AS EIGENFACES

The top row here shows the input images, while the bottom row shows the reconstruction of the images from just 150 of the ~3,000 initial features.

Although it reduces the dimensionality of the data by nearly a factor of 20, the projected images contain enough information that we might, by eye, recognize the individuals in the image.

What this means is that our classification algorithm needs to be trained on 150-dimensional data rather than 3,000-dimensional data, which depending on the particular algorithm we choose, can lead to a much more efficient classification.



CREDITS & RESOURCES

[1] A One-Stop Shop for Principal Component Analysis.

[2] PCA Visualization and eating in UK.

[3] PCA Notebook.

License