

Dimensionality Reduction

Projection methods:
PCA

Objective

- Gain an understanding of what DR projection methods entail
- Learn the ins and outs of PCA
- Learn the ins and outs of ICA
- Learn to discern when to use PCA and when to use ICA
- Get acquainted with the corresponding code in Python

Curriculum

Module:

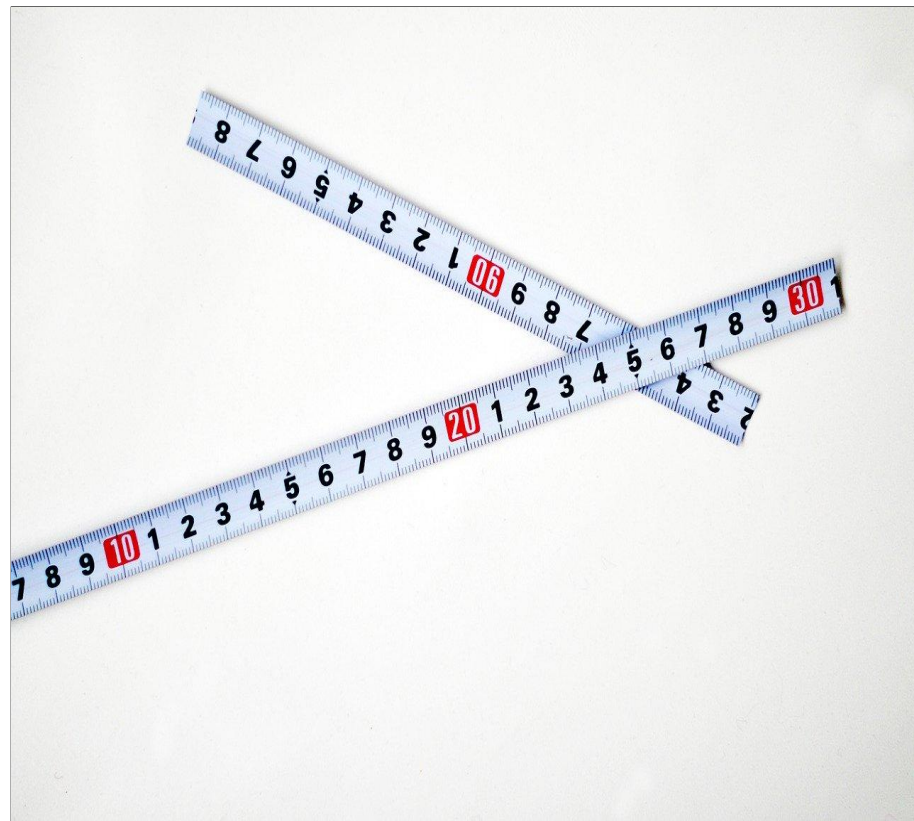
- ➔ Checkpoint 1 Title [linked]
- ➔ Checkpoint 2 Title [linked]

Agenda

- ◆ Warm Up
- ◆ Linear transformations and their usefulness
- ◆ Principal Component Analysis as a linear transformation
- ◆ Use cases and pros & cons of PCA
- ◆ Assignment

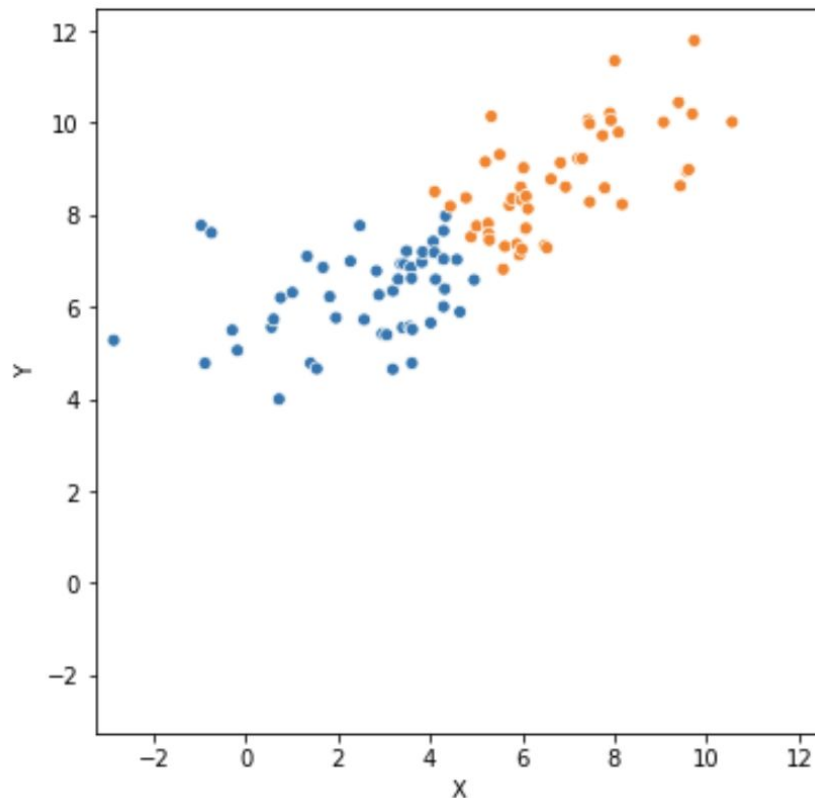
Warm Up

1. If you had to summarize all the features in a couple of meta-features, how would you go about it?
2. How would combining features compare with selecting features?
3. What would be the easiest and most efficient way of combining features in order to obtain a set of meta-features?



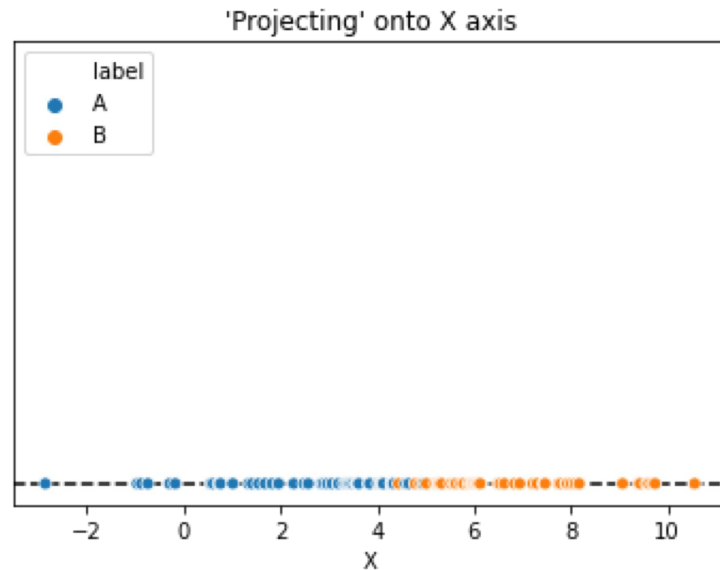
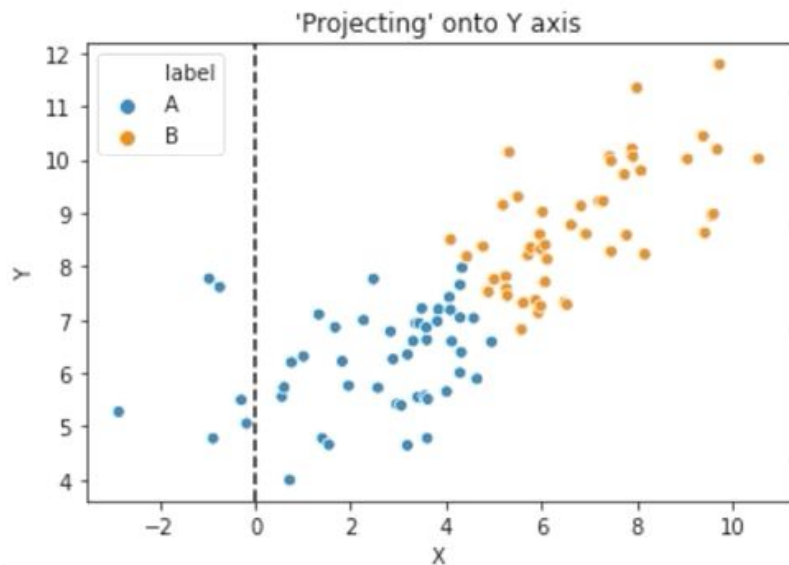
Warm Up

1. If you had to choose only 1 feature (X or Y) to perform this classification task, which would you choose? Why?



Warm Up

These 2 options can also be referred to as 'projections'



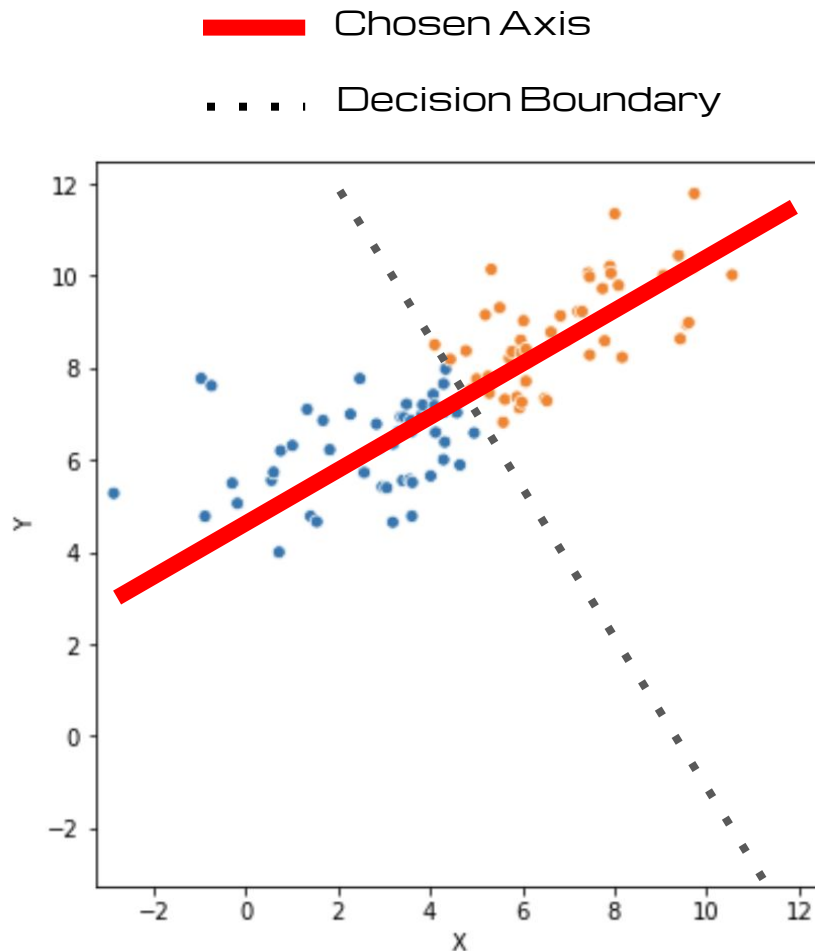
Warm Up

1. If you had to choose only 1 feature (X or Y) to perform this classification task, which would you choose? Why?
2. In the first question you could only choose the horizontal or vertical axis, what if you could make up your own axis at any angle? Would this be a better feature for the classification task? Why?



Warm Up

1. If you had to choose only 1 feature (X or Y) to perform this classification task, which would you choose? Why?
2. In the first question you could only choose the horizontal or vertical axis, what if you could make up your own axis at any angle? Would this be a better feature for the classification task? Why?
3. Can we make this axis with a formula using X and Y? How might that look?



UP NEXT

Linear transformations and their usefulness



Linear transformations

- ◆ Involve a linear combination of variables / features
 - ◇ Adding and subtracting these variables
 - ◇ Multiplying each one of these variables by a coefficient
- ◆ Essential in Linear Algebra
- ◆ Computationally cheap and fast to process
- ◆ Examples, for variables x , y , and z :
 - ◇ $x + y + z$
 - ◇ $14.0x - 3.1y + 0.5z$
 - ◇ $4x - 7y$
 - ◇ $2x + 6z$

Linear transformations (cont.)

1. Lines can approximate any shape
2. Linear transformations can do the same with meta-features
3. They are also easy to interpret making the meta-features deriving from them transparent



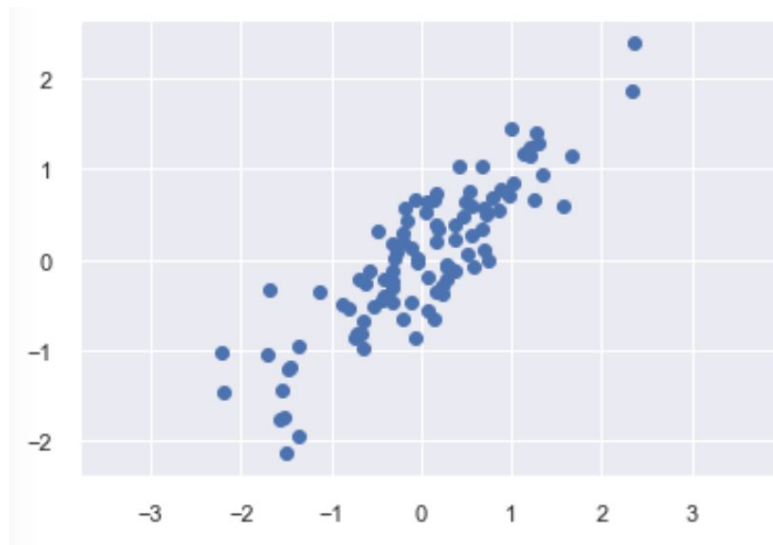
UP NEXT

Principal Component Analysis as a linear transformation



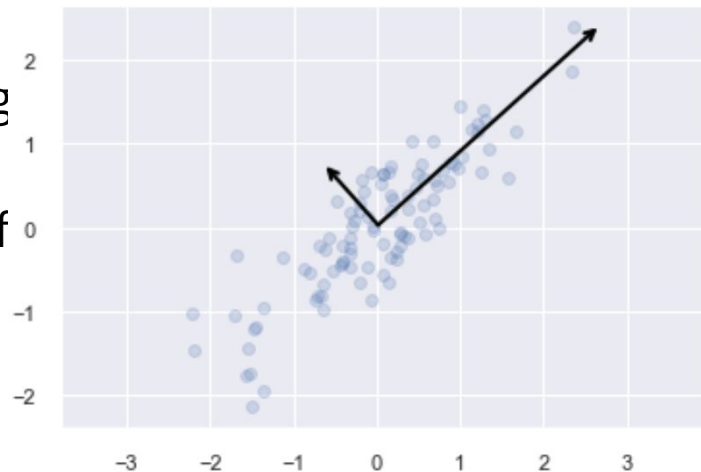
PCA - Intuition and general info

- ◆ A linear transformation that aims to provide the best way to combine features
- ◆ It relies on the **variance** as a proxy of information content
- ◆ The **covariances** of the features are also explored and analyzed
- ◆ Covariance is a proxy of the relationship between two variables. In this example, x and y are two standardized variables exhibiting high covariance



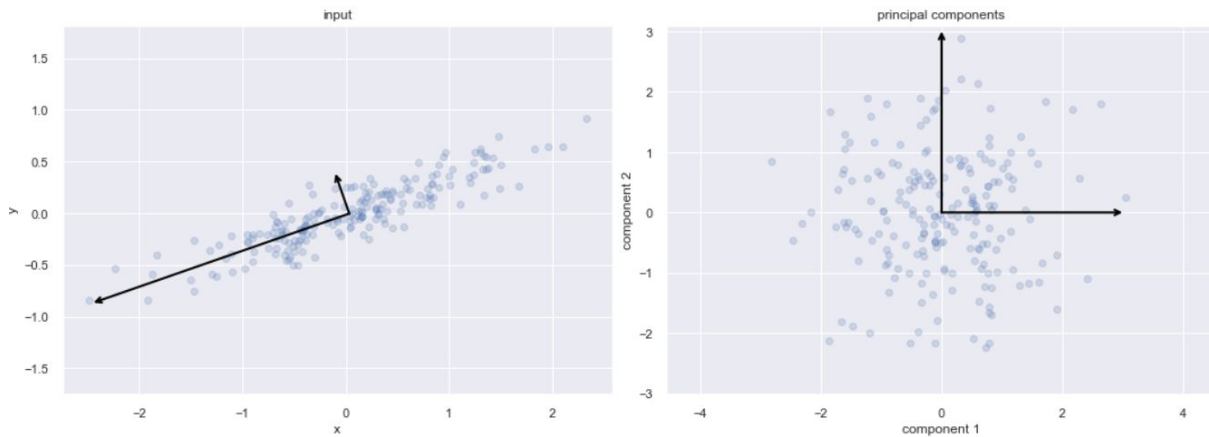
PCA - Intuition and general info (cont.)

- ◆ Figuring out the meta-features that come about from a linear combination of the original features is equivalent to finding a set of **eigenvectors** and their corresponding **eigenvalues**
- ◆ Eigenvectors have to do with the direction of each one of the meta-features PCA creates
- ◆ Eigenvalues have to do with the length of these eigenvectors which relates to the importance of each one of them
- ◆ In this example the first eigenvector (which also has the largest length) is across the part of the dataset that has the most variance.



PCA - Intuition and general info (cont.)

- ◆ The meta-features PCA yields come about by projecting the original data onto each one of these eigenvectors
- ◆ A dataset having 10 features will be transformed to another one having 10 features again (aka PCs or meta-features). However, we don't need all 10 of the latter as the first few of them carry in them enough information (variance) for all intents and purposes



Using PCA for DR

- In order to use PCA for dimensionality reduction we need to select a subset of the meta-features (PCs) it creates.
- The simplest way to do that is through the `n_components` parameters which stands for “number of (principal) components” or meta-features.
- Here is an example of using PCA as a dimensionality reduction model to obtain a single meta-feature:

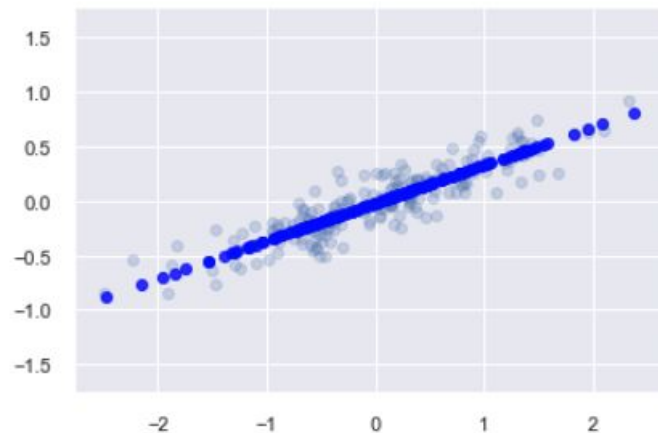
```
pca = PCA(n_components=1)
pca.fit(X)
X_pca = pca.transform(X)
print("original shape: ", X.shape)
print("transformed shape:", X_pca.shape)
```

```
original shape: (200, 2)
transformed shape: (200, 1)
```


Using PCA for DR (cont.)

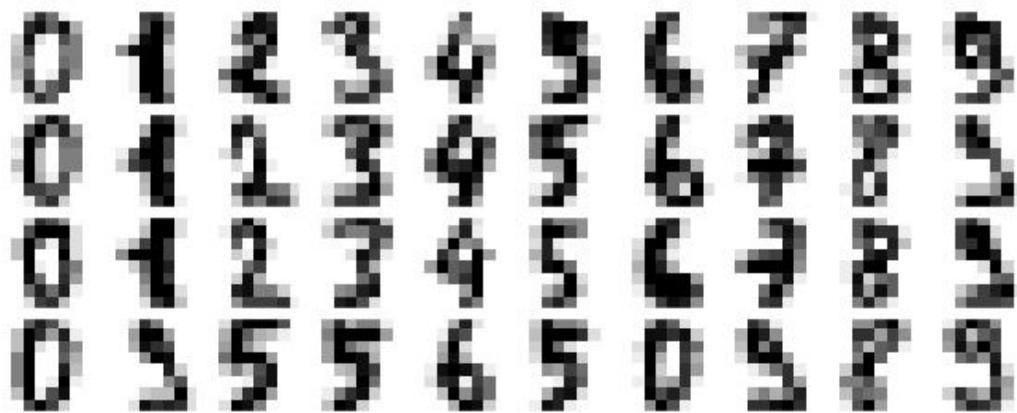
- Usually we'd opt for a larger number of meta-features, but since the aforementioned example has only two features to begin with, a single meta-feature is a viable option.
- But how would the dataset look if we were to rebuild it using that one PC that PCA has yielded? To do that we'd need to inverse the process as follows:

```
X_new = pca.inverse_transform(X_pca)
plt.scatter(X[:, 0], X[:, 1], alpha=0.2)
plt.scatter(X_new[:, 0], X_new[:, 1], alpha=0.8, color="blue")
plt.axis('equal');
```



Inverse Transformation as a sanity test for PCA

- The inverse transformation process of PCA is useful as a way to test how much information was lost through the dimensionality reduction process
- This is not something we have to do, but in some cases, as in the cases of image data, it makes sense. For example, let's say we have the following 64 dimensional image data of handwritten digits:



Inverse Transformation as a sanity test for PCA (cont.)

- Should we reduce the dimensionality to 2 (e.g. for plotting the data) and then back to 64 (the original dimensionality) the image data would look something like this:



Inverse Transformation as a sanity test for PCA (cont.)

- Clearly this isn't very good as there is a lot of information lost
- In most cases the inverse transformation of the reduced feature set (using PCA) may not be as easy to observe
- Because of the inevitable information loss of PCA, you can think of it like jpeg encoding for images. The original high-quality image is compressed in a way that's irreversible (i.e. some of the clarity of the original image will be lost).

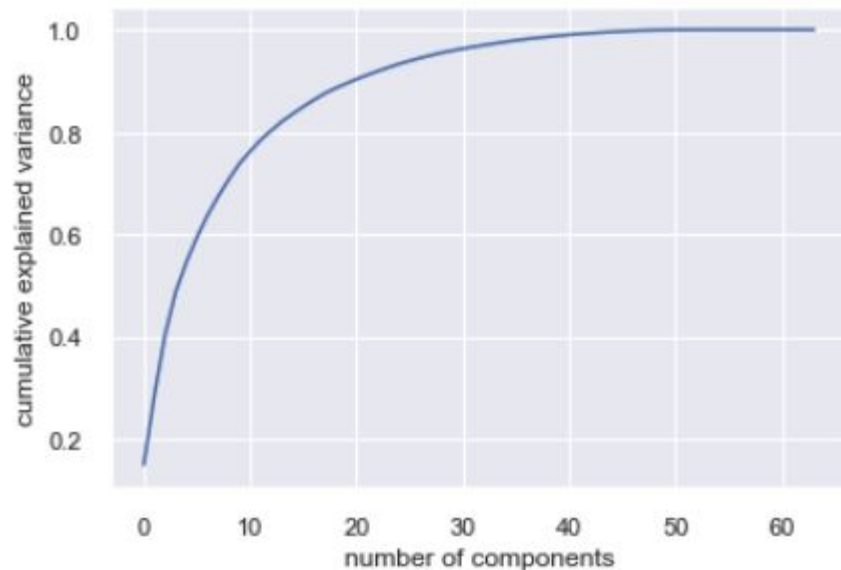
QUESTION

How many components would be sufficient in a PCA setting?

Choosing the number of PCs in PCA

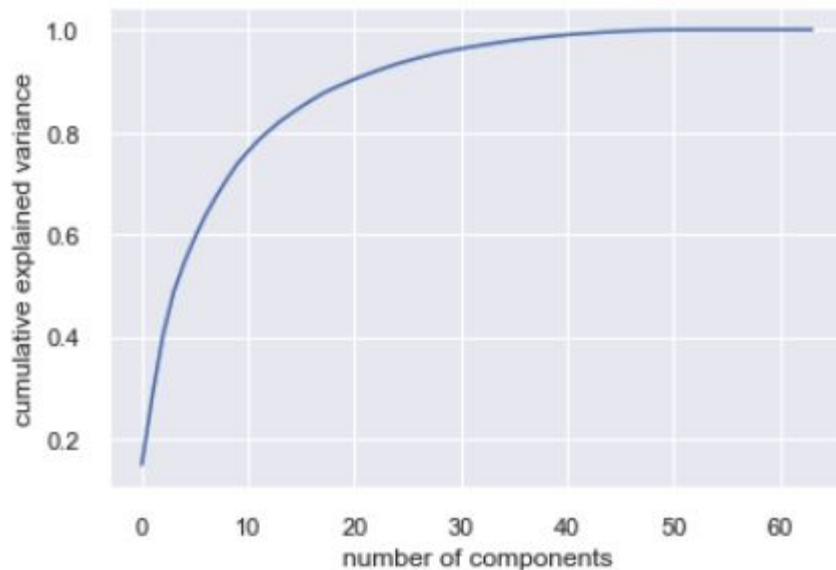
- Guesswork won't cut it as we'll need something more scientific!
- One great way to do that is to explore how much variance is explained by different subsets of components. This is possible through plotting the cumulative variance explained as a function of the number of PCs used. To obtain the data for this plot, we can use the following command:

```
print(model.explained_variance_ratio_)
```



Choosing the number of PCs in PCA (cont.)

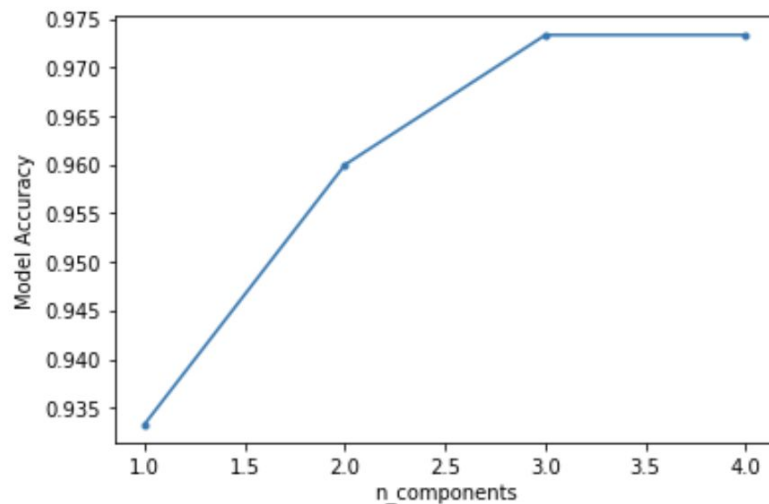
- Usually we set a threshold like 0.75 or 0.9. Naturally, the higher the threshold, the more PCs we'll end up needing
- For this example, if we were to use the threshold 0.9, we'd need around 20 PCs. This is around 70% reduction in the dimensionality!



Choosing the number of PCs in PCA (cont.)

- If you're using PCA in a supervised learning pipeline, you can also consider using model accuracy for guidance on the correct number of components.
- Think of this as another hyperparameter that can be optimized.

Model Accuracy by n_components: [0.93 0.96 0.97 0.97]



UP NEXT

Use cases and
pros & cons of
PCA

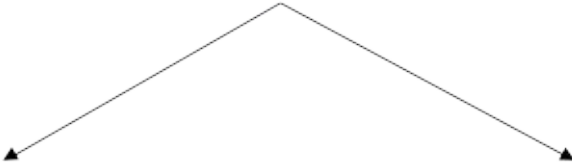


Use cases for PCA

- ◆ In general, any scenario involving lots of variables that are correlated to each other
- ◆ Cases where there are linear relationships among the features at hand
- ◆ Cases where there are too many features to handle with other, more sophisticated DR methods

Pros and cons of PCA

Principle Components Analysis



```
graph TD; PCA[Principle Components Analysis] --> Pros[Pros:]; PCA --> Cons[Cons:];
```

Pros:

- easy to use
- makes figuring out the number of components to use straight-forward
- easier to comprehend and explain to others

Cons:

- doesn't handle non-linear relationships in the data
- yields components that although uncorrelated, they are not truly independent to each other

Summary

- Linear transformation are important and find a very practical application in dimensionality reduction through methods like PCA
- PCA creates meta-features using the original features, efficiently reducing dimensionality
- PCA makes finding the best number of components (meta-features) easier through the variance explained metric. Usually a threshold of 0.75 or so yields satisfactory results

Assignment

pitchFX dataset

1. Read data with:

```
pd.read_csv("https://docs.google.com/spreadsheets/d/1pmBtSw7v\_tU\_dIX1-4E8\_Q7wC43fDs6LGdQzN49-ffk/export?format=csv")
```

2. **Objective: Use the numeric features and PCA to create a `pitchType` classifier. What is the ideal number of components?**

3. A full data dictionary can be found here:

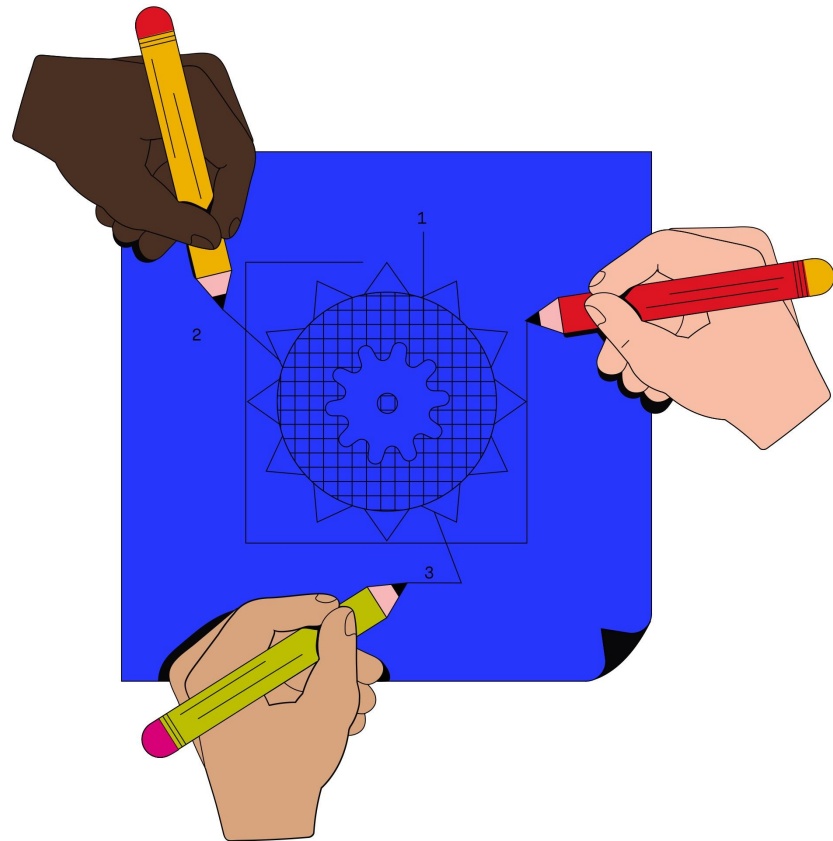
[https://www2.stat.duke.edu/courses/Summer17/sta101.001-2/uploads/project/project.html](\"https://www2.stat.duke.edu/courses/Summer17/sta101.001-2/uploads/project/project.html\")

THANKFUL

Thank You

UP NEXT

Independent Component Analysis as a linear transformation

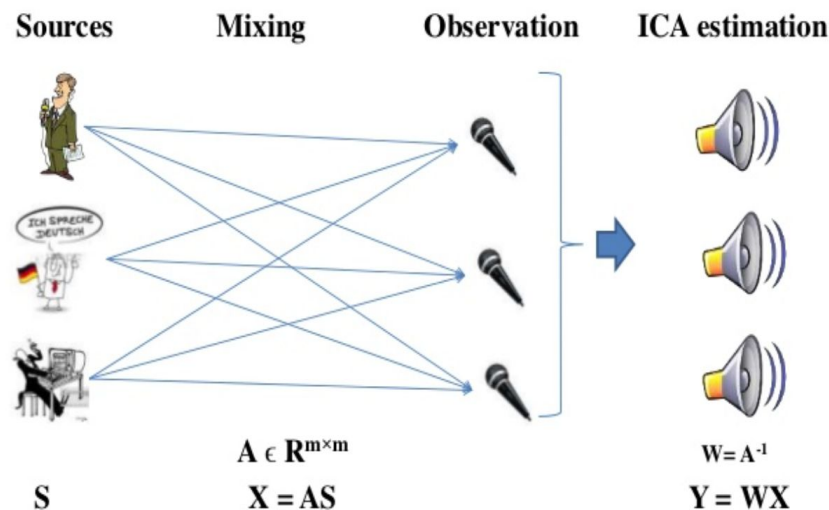


Blind Source Separation (BSS) and ICA as a solution

- ◆ BSS is the separation of a set of source signals from a set of mixed signals
 - Think of it as trying to get to individual spaghetti untangled from a spaghetti bowl
- ◆ Classical application: figuring out the different audio signals based on a single audio channel
 - This is something we do naturally when in a crowded room full of chatter, when we are listening to a particular individual in it. This is known as the *cocktail party effect*

ICA and its assumptions

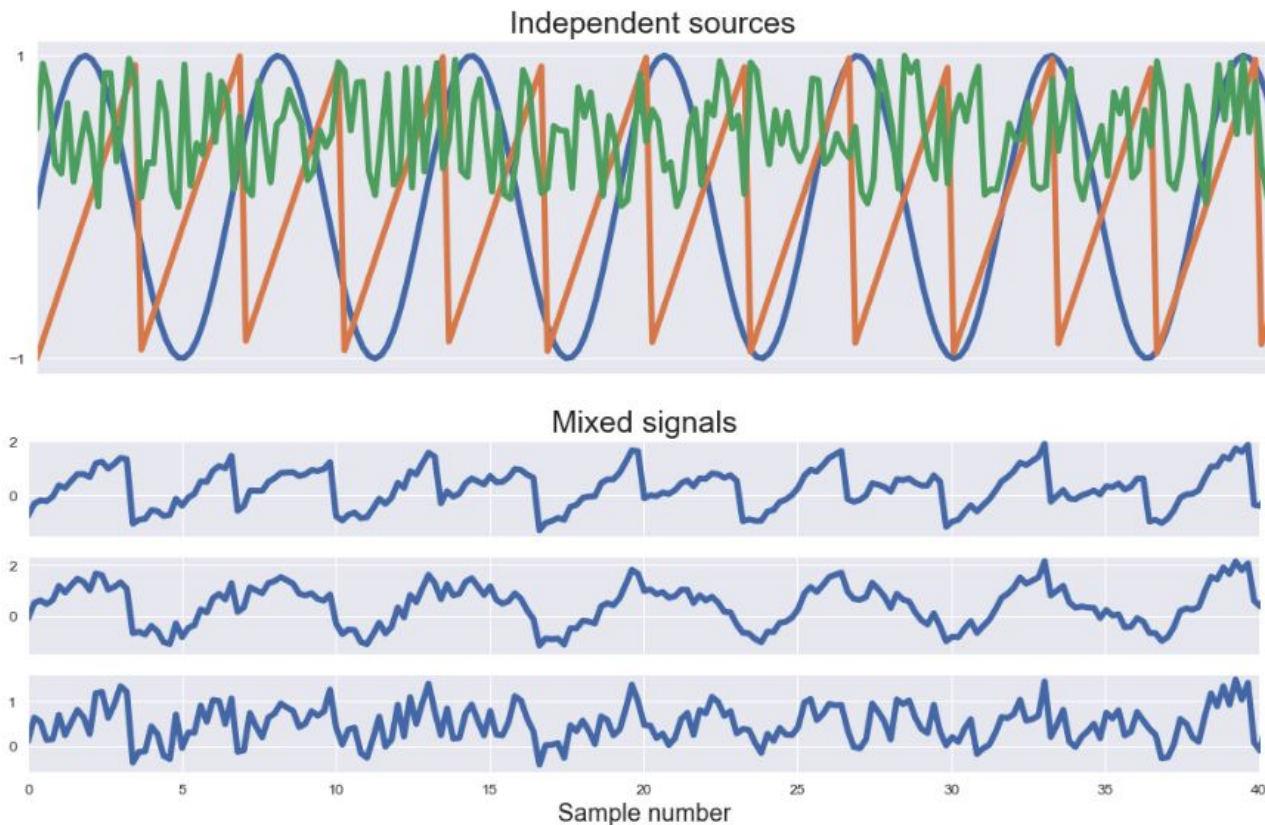
- ◆ Assumptions:
 - Linear projections which are not necessarily orthogonal to each other.
 - Generalization of PCA where principal components are nearly statistically indep.
 - Source data comprises of linear mixtures of non-gaussian unknown latent variables (source signals)



Why not just use PCA instead?

- ◆ Since PCA assumes normal (Gaussian) distributions in all variables, applying PCA on the mixed signal would create interesting meta-features that would however not represent the original (source) signals
- ◆ In the case of the cocktail party sounds, the PCs would be combinations of the audio signals but not nearly as clear as the original voices and music present in the room
- ◆ In other words, PCA would provide uncorrelated meta-features but ICA would provide truly independent ones.

Independent signals in practice



ICA's underlying generative model

- ◆ Generate independent meta-features using mutual information metric (I)
- ◆ Note: two signals s_1 and s_2 are independent if the information in s_1 does not give any information about s_2 and vice versa. This is equivalent to their covariance being 0, as well as their mutual information being 0 too.
- ◆ The joint distribution of Gaussian source signals is completely symmetric just like the joint distribution of the mixed signals. As a result, it doesn't contain any information about the mixing matrix, the inverse of which we want to calculate. In cases like this the ICA algorithm will fail.
- ◆ Ideally, the mutual information of the mixed signals will be as high as possible ($I(s_1, s_2)$ close to 1.0).

ICA's underlying generative model (cont.)

- ◆ Statistical independence is a stronger criterion than merely uncorrelatedness (no covariance). Think of it as two people who arrive at different conclusions because their thinking process is inherently different, not just because they weigh different factors differently. In the first case they are akin to independent meta-feature while in the latter case they are just meta-features that happen to be uncorrelated.
- ◆ ICA accomplishes statistical independence in its meta-features through one of the following approaches:
 - Minimizing their mutual information: $I(x,y)$
 - Maximizing their non-Gaussianity (making them as different to the Gaussian distribution as possible)

The Mutual Information ordeal

- ◆ All of the information theory metrics (including mutual information) are based on probabilities
- ◆ The probability of two events x, y happening at the same time is $P(x, y)$. If we were to take the natural log of this, reverse its sign, and multiply it by the probability itself we'd end up with what is called Conditional Entropy for those events. Naturally, a meta-feature consists of several data points (the aforementioned events), so we'd need to sum up all of their entropies:

$$H(X|Y) = - \sum_{x,y} P(x, y) \log(P(x, y))$$

The Mutual Information ordeal (cont.)

- ◆ The entropy of a single event is defined the same way though instead of $P(x,y)$ we used just $P(x)$. Also, the entropy of a meta-feature is defined similarly:

$$H(X) = - \sum_x P(x) \log(P(x))$$

- ◆ Entropy is often used as a metric of disorder (i.e. lack of order or randomness)
- ◆ Based on the entropy of a meta-feature X and the conditional entropy of this meta-feature in relation to another one, Y , we can define mutual information $I(X,Y)$:

$$I(X, Y) = H(X) - H(X|Y)$$

Mutual Information in ICA

- ◆ As M.I. can be viewed as the reduction of uncertainty regarding X , based on the observation (use) of Y (in a model), we can optimize it for a given set of parameters.
- ◆ In the case of ICA these parameters are a set of weights W
- ◆ For this, the *InfoMax* algorithm is used. This entails starting with random values for W and changing them until $I(X,Y)$ converges (i.e. it doesn't change much any more)
- ◆ The optimization process involves **minimizing** $I(X,Y)$

Maximizing Non-Gaussanity (while maintaining our sanity)

- ◆ This involves making sure that the distribution of the meta-features is as far away from the normal distribution as possible
- ◆ To accomplish this we use another metric from Information Theory called **negentropy**, which is probably the most counter-intuitive name for a metric ever created. Contrary to what you may think, it is not the negative of entropy but instead is defined as follows:

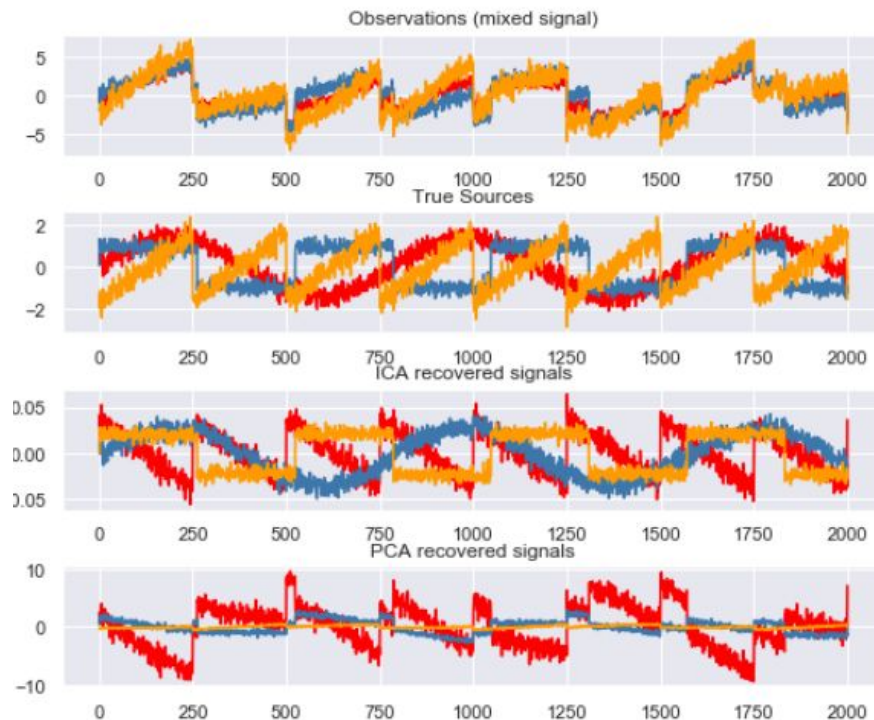
$$N(X) = H(X_{\text{gaussian}}) - H(X)$$

where $H(X_{\text{gaussian}})$ is the entropy (H) of a Gaussian random vector whose covariance matrix is equal to that of X

- To optimize negentropy we use the *FastICA* algorithm which works like InfoMax but instead of minimizing $I(X,Y)$, it maximizes $N(X)$.

ICA in practice

- ◆ For the example on the right, we can use ICA through the code available in SciKit Learn:
<http://bit.ly/2IsECzW>
- ◆ As you can see, the ICA method recovers the original signals quite well, while PCA does a poor job at this



UP NEXT

Use cases and
pros & cons of
PCA and ICA

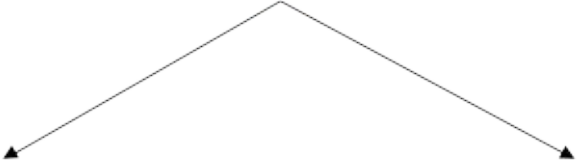


Use cases for PCA and ICA

- ◆ In general, any scenario involving lots of variables that are correlated to each other
- ◆ Specifically for PCA: cases where there are linear relationships among the features at hand
- ◆ Specifically for ICA: cases where the original data comprises of combinations of the meta-features we are looking for
- ◆ For both, cases where there are too many features to handle with other, more sophisticated DR methods

Pros and cons of ICA

Independent Components Analysis



```
graph TD; ICA[Independent Components Analysis] --> Pros[Pros:]; ICA --> Cons[Cons:];
```

Pros:

- easy to use
- ideal for figuring out latent features
- yields independent meta-features, not just uncorrelated ones
- makes use of more sophisticated metrics

Cons:

- doesn't help in selecting the number of components
- harder to comprehend and to explain to others
- fewer use cases where it offers a real advantage over other DR methods

Summary

- Linear transformation are important and find a very practical application in dimensionality reduction through methods like PCA and ICA
- Both PCA and ICA create meta-features using the original features, efficiently reducing dimensionality
- PCA is more user-friendly in terms of the processes involved
- PCA makes finding the best number of components (meta-features) easier through the variance explained metric. Usually a threshold of 0.75 or so yields satisfactory results
- ICA makes use of mutual information or non-Gaussianity for constructing the meta-features it yields
- ICA is great for cases involving Blind Source Separation (BSS) such as the original audio signals behind a mixed signals scenario

Assignment

pitchFX dataset

1. Read data with:

```
pd.read_csv("https://docs.google.com/spreadsheets/d/1pmBtSw7v\_tU\_dIX1-4E8\_Q7wC43fDs6LGDQzN49-ffk/export?format=csv")
```

2. **Objective: Compare PCA and ICA to figure out which works better at identifying components related to `pitchType`**

3. A full data dictionary can be found here:

[https://www2.stat.duke.edu/courses/Summer17/sta101.001-2/uploads/project/project.html](\"https://www2.stat.duke.edu/courses/Summer17/sta101.001-2/uploads/project/project.html\")