Dimension reduction finds patterns in data, and uses this to compress data

This is efficient storage and computation

A useful use case for dimension reduction is to remove noise

Principal Component Analysis firstly decorrelates, then reduces dimensions

- rotate data samples to be aligned with axis
- shift data samples so they have mean 0

```
from sklearn.decomposition import PCA
model = PCA()
model.fit(samples)
transformed = model.transform(samples)
```

Through a PCA transform, features are no longer correlated

In short, PCA learns the principal components of data = direction of vectors

Intrinsic Dimension

Intrinsic dimension = number of features needed to approximate the dataset

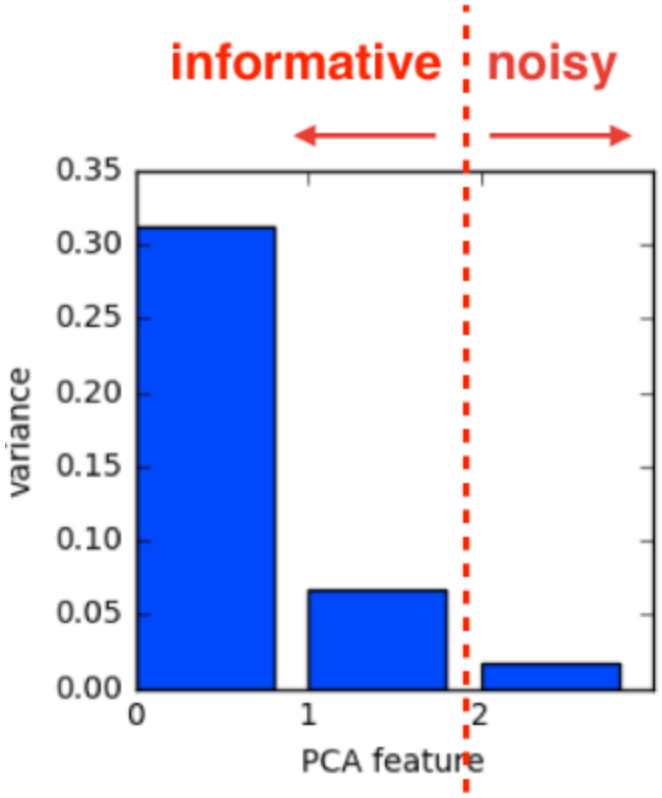
PCA identifies intrinsic dimension by counting the high variance PCA features

```
pca = PCA()
pca.fit(samples)
plt.bar(features, pca.explained_variance_)
plt.xticks(features)
plt.ylabel('variance')
plt.xlabel('PCA feature')
plt.show()
```

Dimension reduction with PCA

Represents same data, using less features

Important for ML pipelines




```
pca = PCA(n_components=2)
pca.fit(samples)
transformed = pca.transform(samples)

xs = transformed[:,0]
ys = transformed[:,1]
plt.scatter(xs, ys, c=species)
plt.show()
```

Sparse arrays mean most entries are zero

We can use a csr_matrix because it only remembers non-zero values

In sci-kit learn, we use TruncatedSVD

```
from sklearn.decomposition import TruncatedSVD
model = TruncatedSVD(n_components=3)
model.fit(documents)
transformed = model.transform(documents)
```