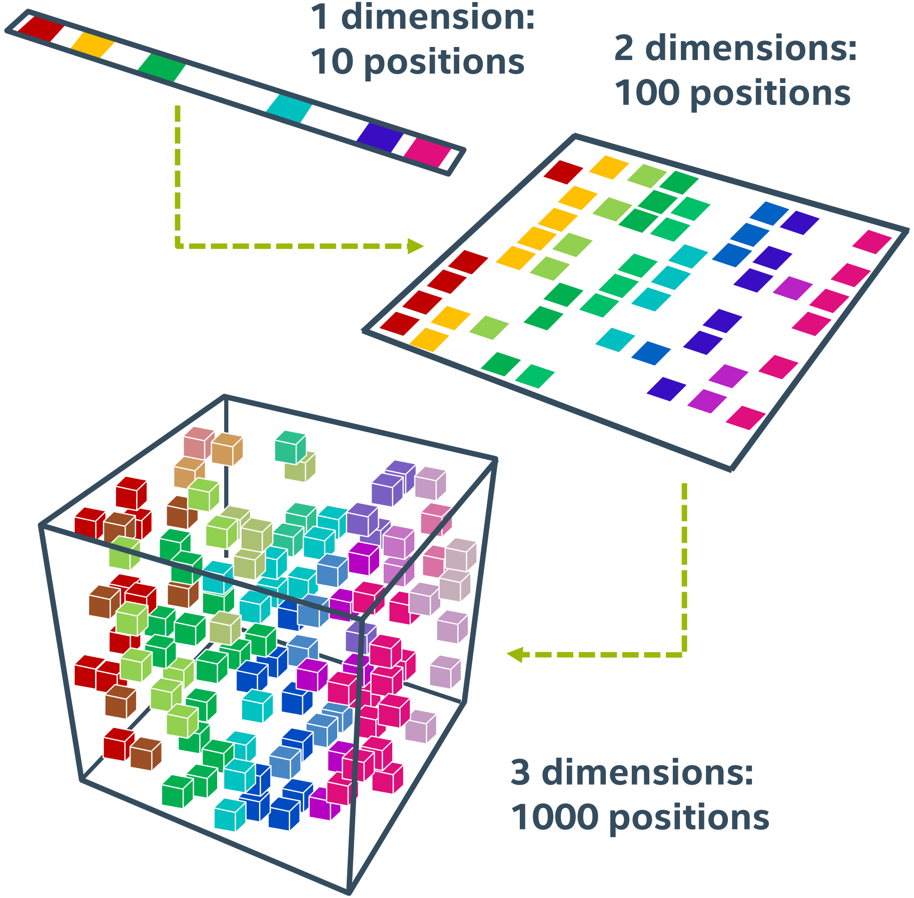
DIMENSIONALITY REDUCTION

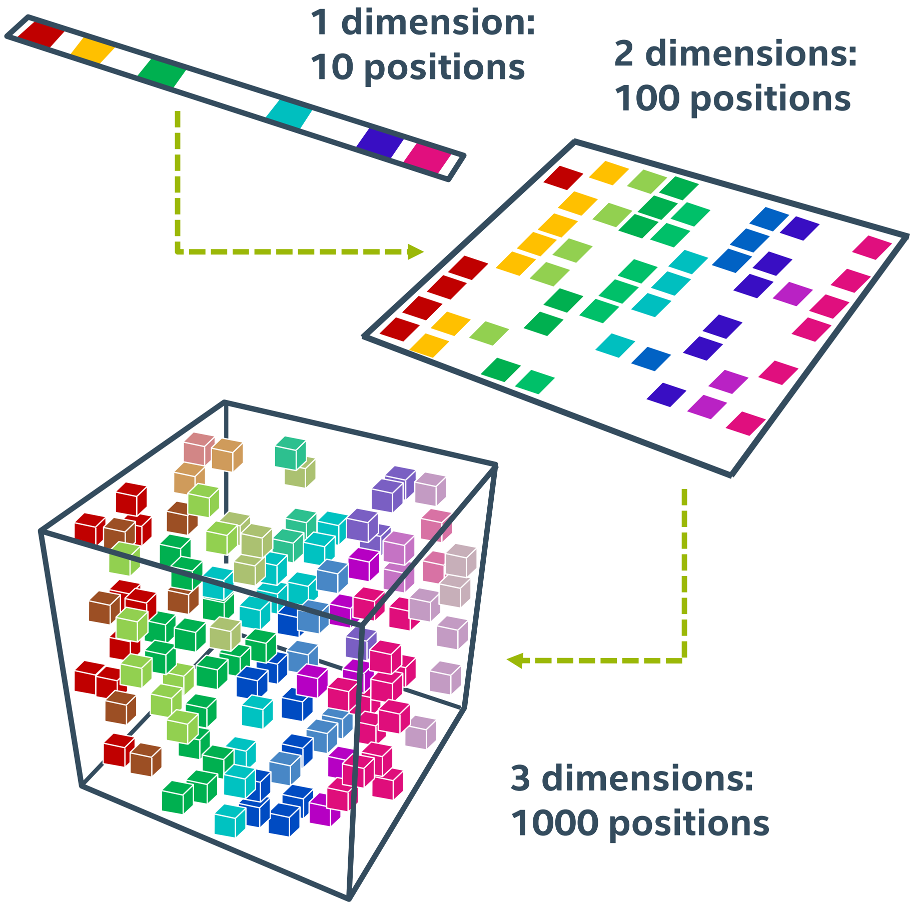
CURSE OF DIMENSIONALITY

- Theoretically, increasing features should improve performance



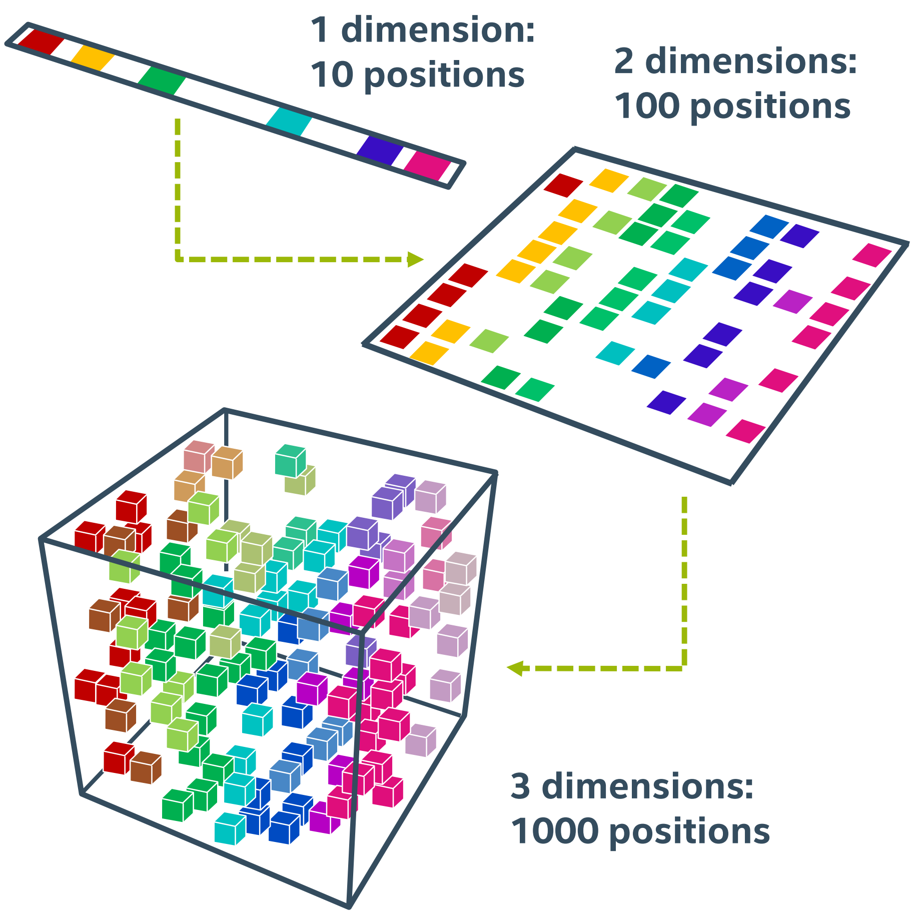
CURSE OF DIMENSIONALITY

- Theoretically, increasing features should improve performance
- In practice, more features leads to worse performance



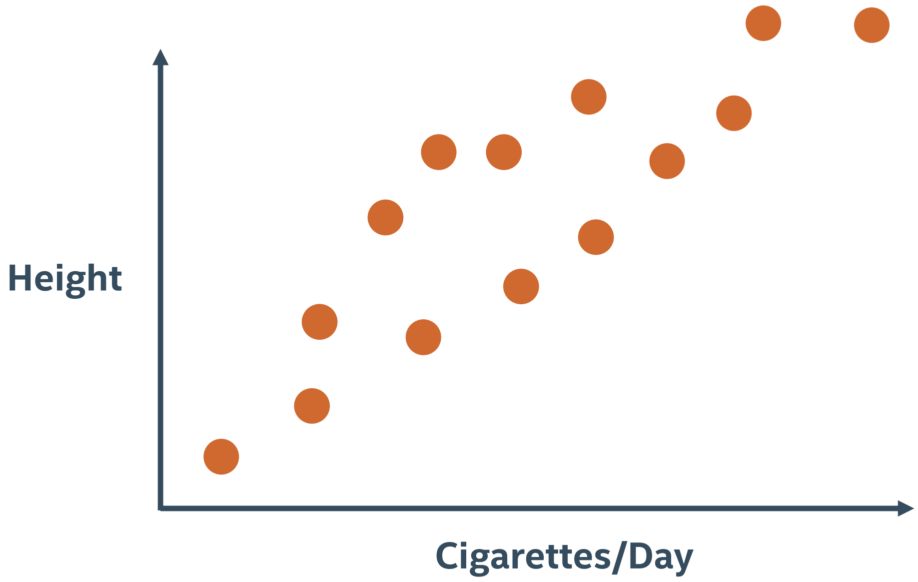
CURSE OF DIMENSIONALITY

- Theoretically, increasing features should improve performance
- In practice, more features leads to worse performance
- Number of training examples required increases exponentially with dimensionality



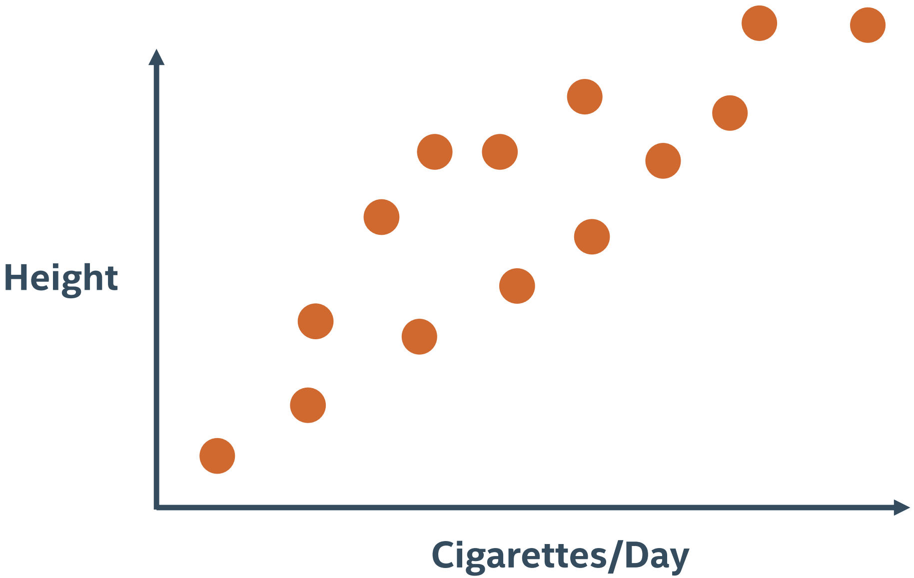
SOLUTION: DIMENSIONALITY REDUCTION

- Data can be represented by fewer dimensions (features)



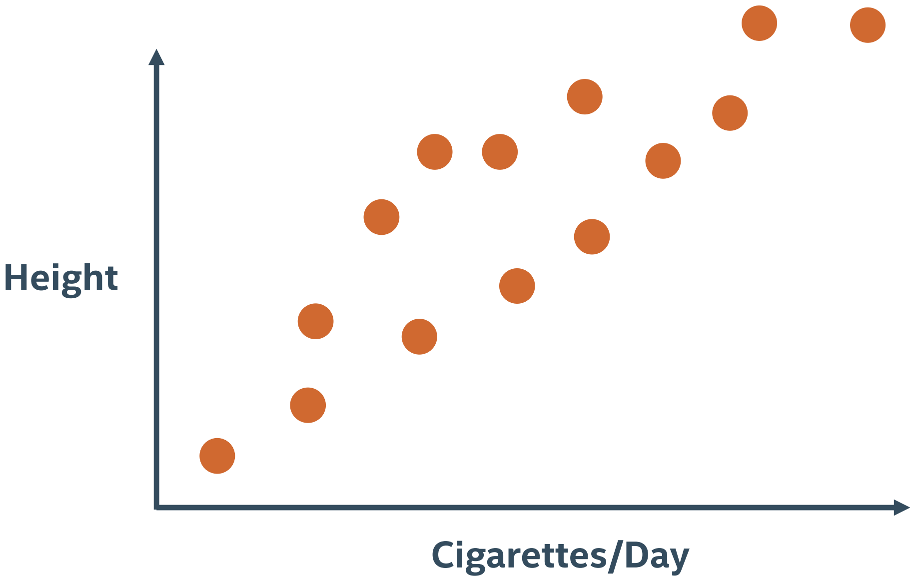
SOLUTION: DIMENSIONALITY REDUCTION

- Data can be represented by fewer dimensions (features)
- Reduce dimensionality by selecting subset (feature elimination)



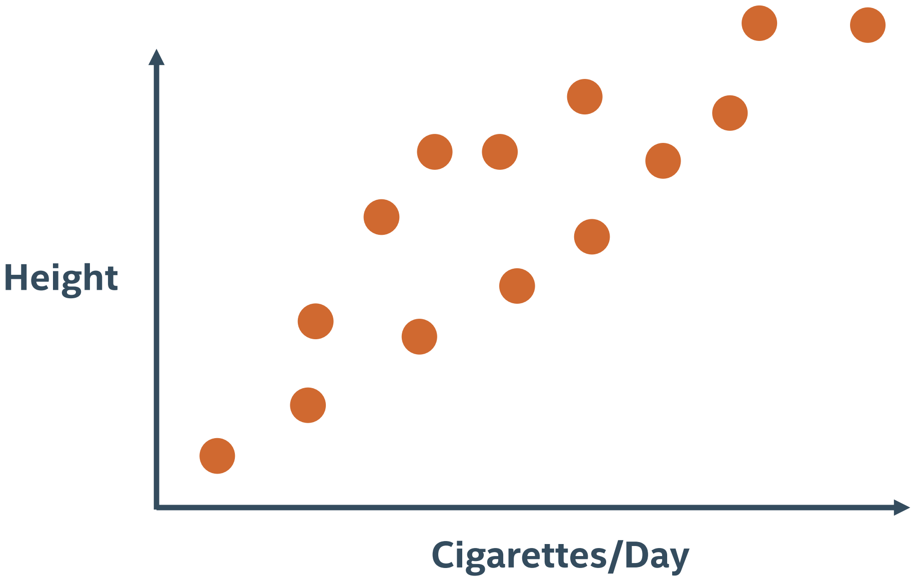
SOLUTION: DIMENSIONALITY REDUCTION

- Data can be represented by fewer dimensions (features)
- Reduce dimensionality by selecting subset (feature elimination)
- Combine with linear and non-linear transformations



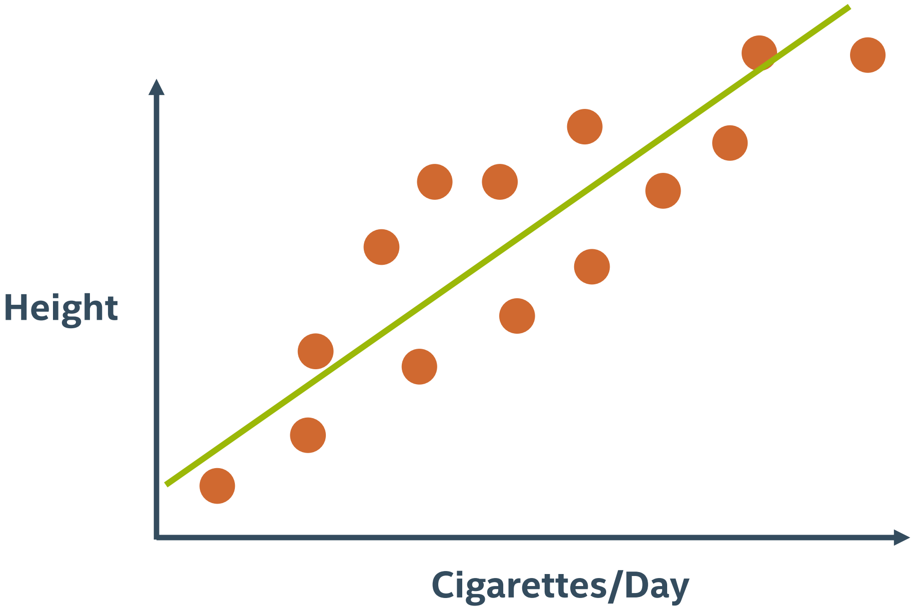
SOLUTION: DIMENSIONALITY REDUCTION

- Two features: height and cigarettes per day
- Both features increase together (correlated)
- Can we reduce number of features to one?



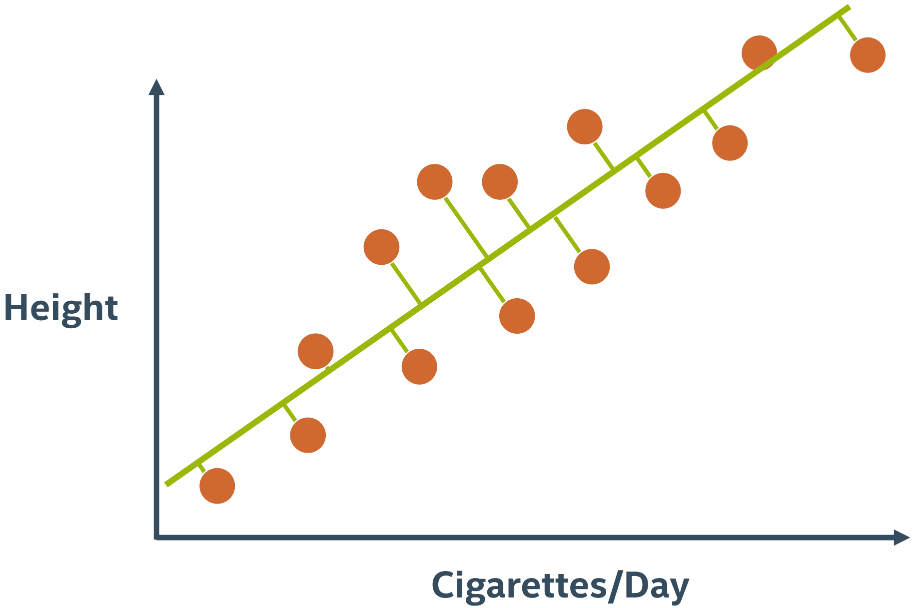
SOLUTION: DIMENSIONALITY REDUCTION

- Two features: height and cigarettes per day
- Both features increase together (correlated)
- Can we reduce number of features to one?



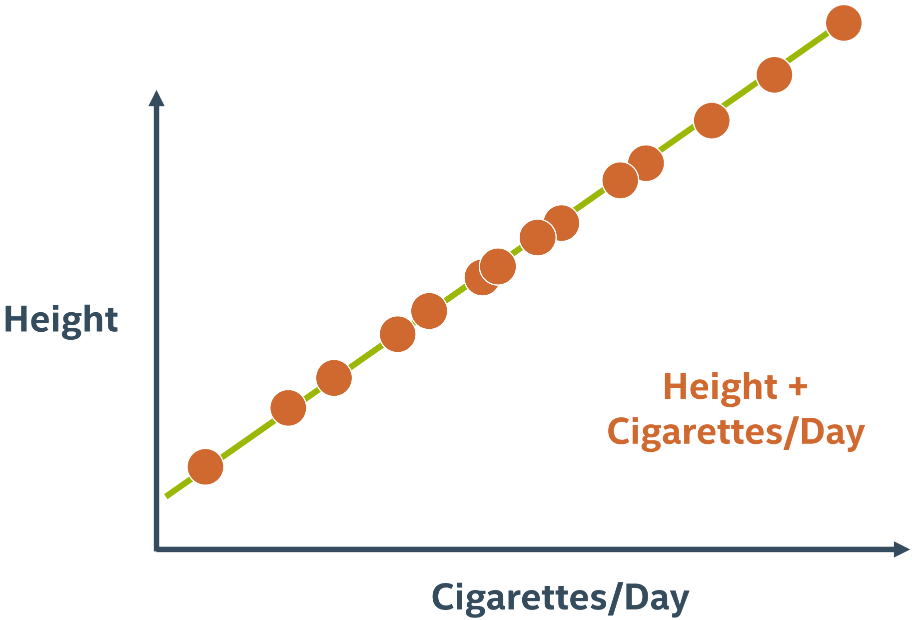
SOLUTION: DIMENSIONALITY REDUCTION

- Two features: height and cigarettes per day
- Both features increase together (correlated)
- Can we reduce number of features to one?



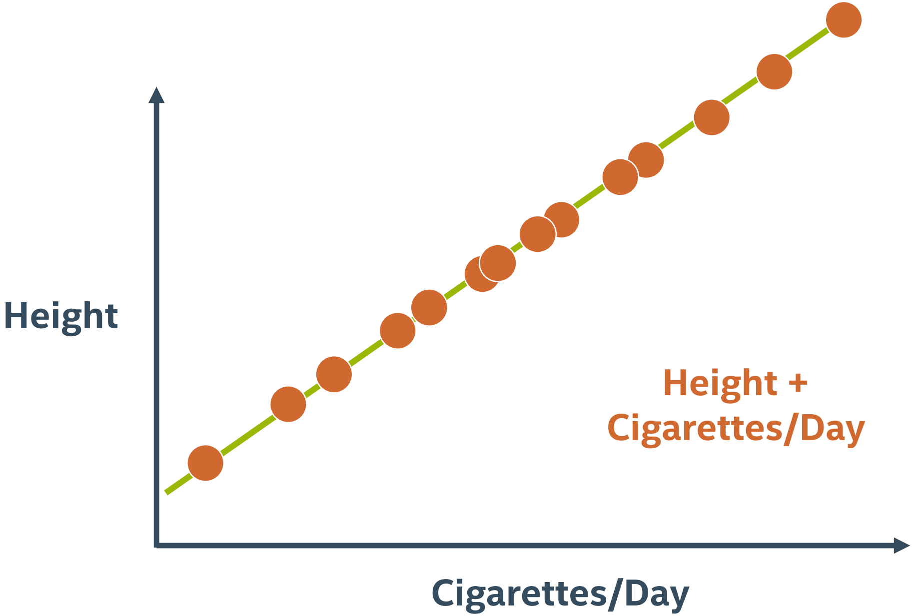
SOLUTION: DIMENSIONALITY REDUCTION

- Two features: height and cigarettes per day
- Both features increase together (correlated)
- Can we reduce number of features to one?



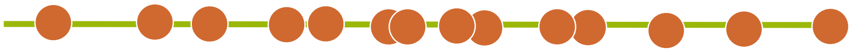
SOLUTION: DIMENSIONALITY REDUCTION

- Create single feature that is combination of height and cigarettes
- This is Principal Component Analysis (PCA)



SOLUTION: DIMENSIONALITY REDUCTION

- Create single feature that is combination of height and cigarettes
- This is Principal Component Analysis (PCA)



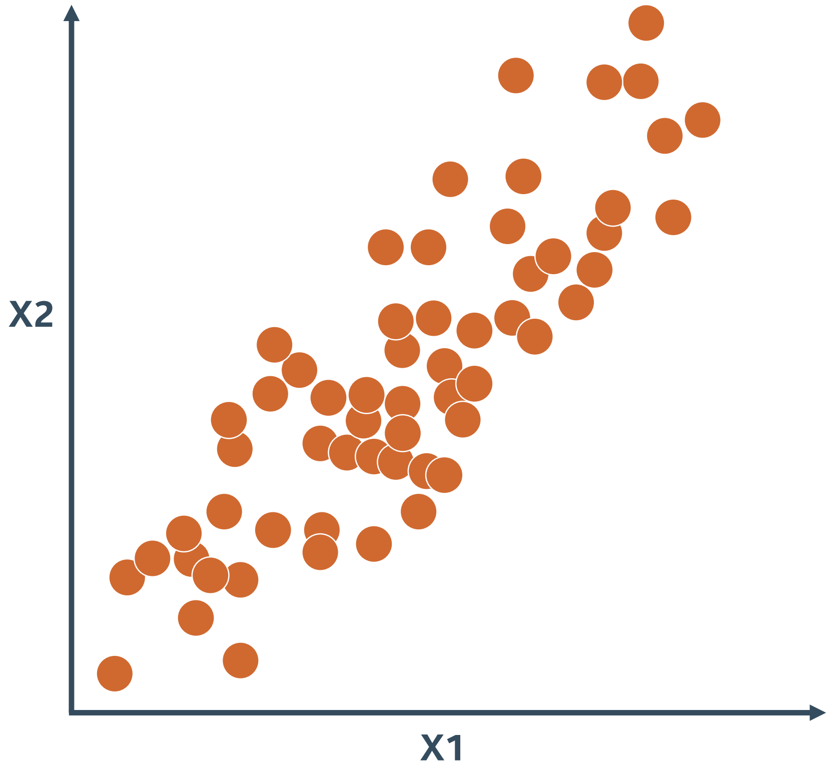
DIMENSIONALITY REDUCTION

Given an N-dimensional data set (x), find a $N \times K$ matrix (U):

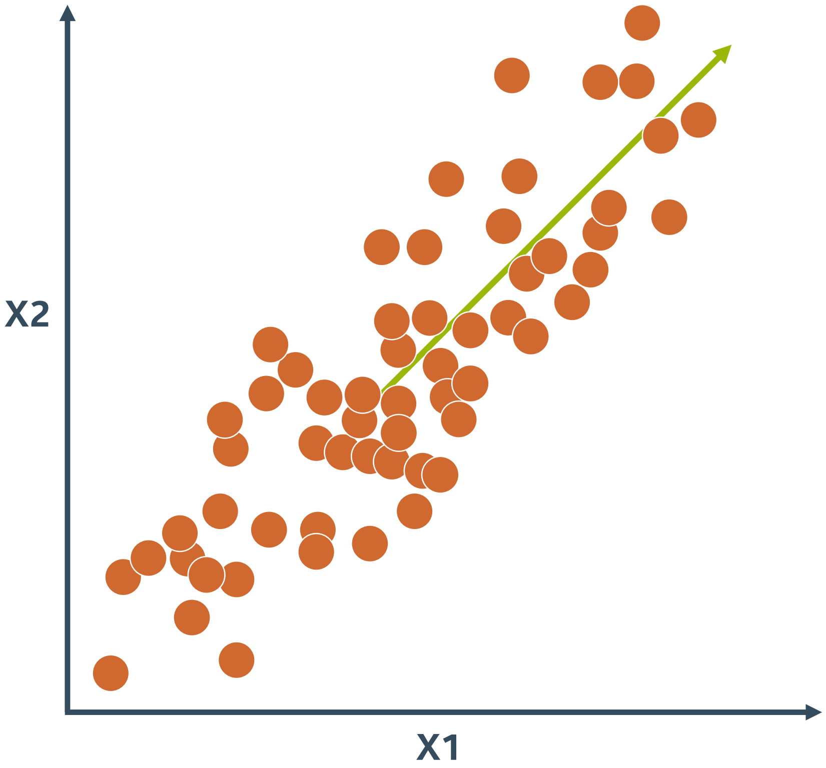
$y = U^T x$, **where** y **has** K **dimensions and** $K < N$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \xrightarrow{U^T} y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_k \end{bmatrix} (K < N)$$

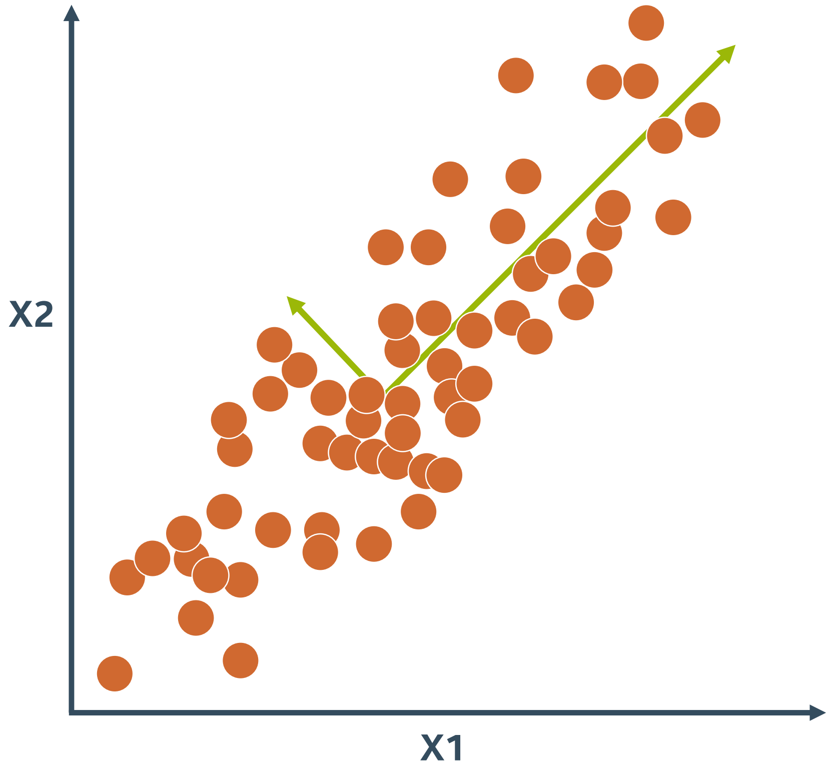
PRINCIPAL COMPONENT ANALYSIS (PCA)



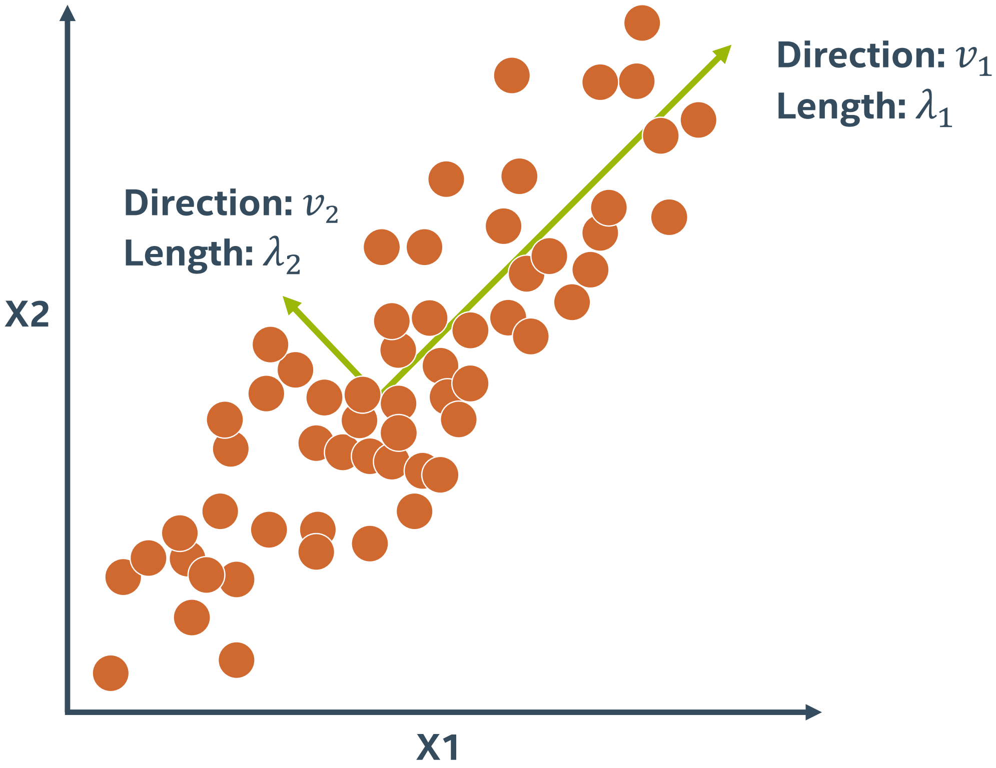
PRINCIPAL COMPONENT ANALYSIS (PCA)



PRINCIPAL COMPONENT ANALYSIS (PCA)



PRINCIPAL COMPONENT ANALYSIS (PCA)



SINGLE VALUE DECOMPOSITION (SVD)

- SVD is a matrix factorization method normally used for PCA
- Does not require a square data set
- SVD is used by Scikit-learn for PCA

$$\underset{A_{m\times n}}{\begin{bmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}} = \underset{U_{m\times m}}{\begin{bmatrix} \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \end{bmatrix}} \underset{S_{m\times n}}{\begin{bmatrix} \star & 0 & 0 \\ 0 & \star & 0 \\ 0 & 0 & \star \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}} \underset{V^T_{n\times n}}{\begin{bmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}}$$

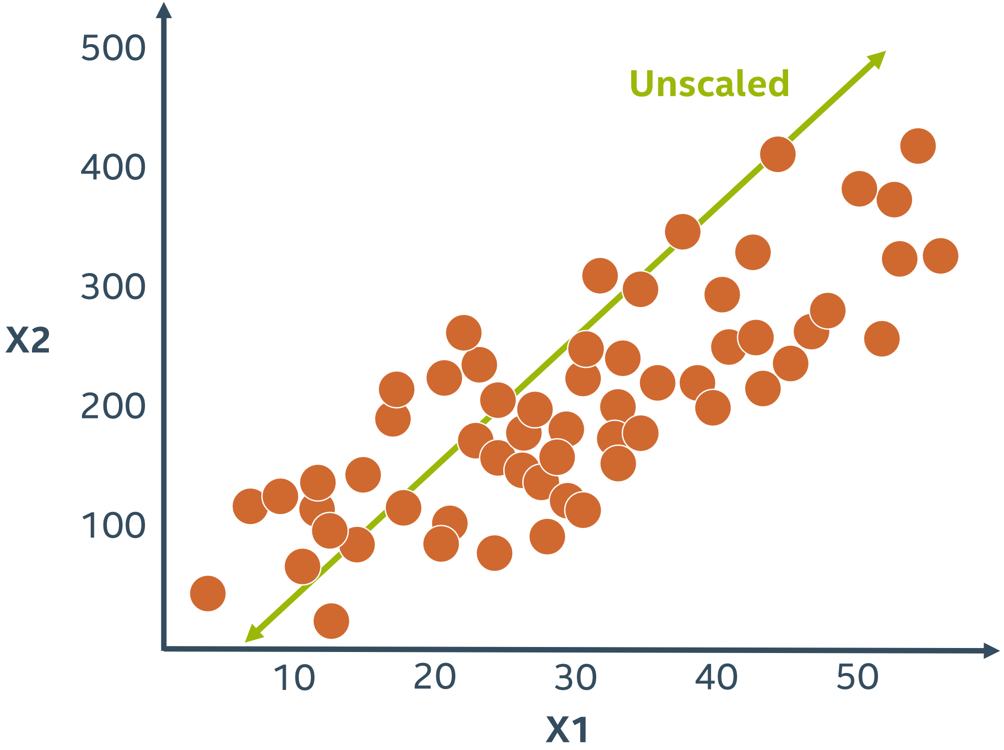
TRUNCATED SINGLE VALUE DECOMPOSITION

- How can SVD be used for dimensionality reduction?
- Principal components are calculated from US
- "Truncated SVD" used for dimensionality reduction ($n \rightarrow k$)

$$\underset{A_{m\times n}}{\begin{bmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}} = \underset{U_{m\times m}}{\begin{bmatrix} \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \end{bmatrix}} \underset{S_{m\times n}}{\begin{bmatrix} \star & 0 & 0 \\ 0 & \star & 0 \\ 0 & 0 & \star \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}} \underset{V^T_{n\times n}}{\begin{bmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}}$$

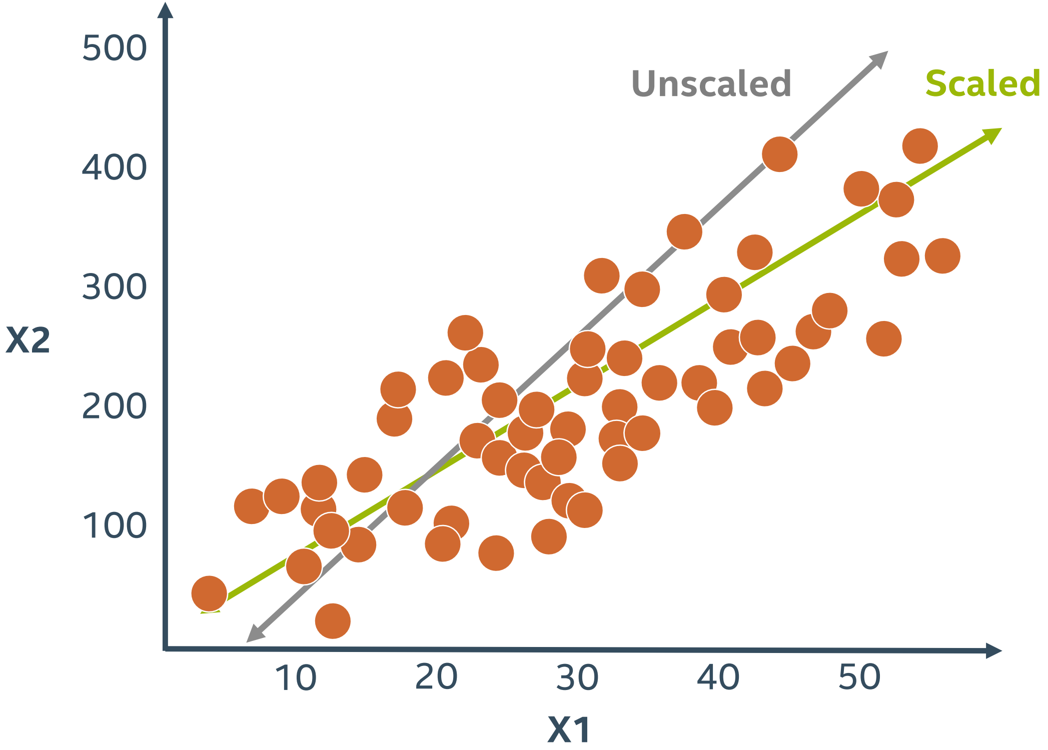
IMPORTANCE OF FEATURE SCALING

- PCA and SVD seek to find the vectors that capture the most variance
- Variance is sensitive to axis scale



IMPORTANCE OF FEATURE SCALING

- PCA and SVD seek to find the vectors that capture the most variance
- Variance is sensitive to axis scale
- Must scale data!



PCA: THE SYNTAX

Import the class containing the dimensionality reduction method.

```
from sklearn.decomposition import PCA
```

PCA: THE SYNTAX

Import the class containing the dimensionality reduction method.

```
from sklearn.decomposition import PCA
```

Create an instance of the class.

```
PCAinst = PCA(n_components=3, whiten=True)
```

PCA: THE SYNTAX

Import the class containing the dimensionality reduction method.

```
from sklearn.decomposition import PCA
```

Create an instance of the class.

```
PCAinst = PCA(n_components=3, whiten=True)
```

final number of dimensions

PCA: THE SYNTAX

Import the class containing the dimensionality reduction method.

```
from sklearn.decomposition import PCA
```

Create an instance of the class.

```
PCAinst = PCA(n_components=3, whiten=True)
```

whiten = scale and center data

PCA: THE SYNTAX

Import the class containing the dimensionality reduction method.

```
from sklearn.decomposition import PCA
```

Create an instance of the class.

```
PCAinst = PCA(n_components=3, whiten=True)
```

Fit the instance on the data and then transform the data.

```
X_trans = PCAinst.fit_transform(X_train)
```

PCA: THE SYNTAX

Import the class containing the dimensionality reduction method.

```
from sklearn.decomposition import PCA
```

Create an instance of the class.

```
PCAinst = PCA(n_components=3, whiten=True)
```

Fit the instance on the data and then transform the data.

```
X_trans = PCAinst.fit_transform(X_train)
```

Does not work with sparse matrices.

TRUNCATED SVD: THE SYNTAX

Import the class containing the dimensionality reduction method.

```
from sklearn.decomposition import TruncatedSVD
```

Create an instance of the class.

```
SVD = TruncatedSVD(n_components=3)
```

Fit the instance on the data and then transform the data.

```
X_trans = SVD.fit_transform(X_sparse)
```

Works with sparse matrices—used with text data for Latent Semantic Analysis (LSA).

TRUNCATED SVD: THE SYNTAX

Import the class containing the dimensionality reduction method.

```
from sklearn.decomposition import TruncatedSVD
```

Create an instance of the class.

```
SVD = TruncatedSVD(n_components=3)
```

does not center data

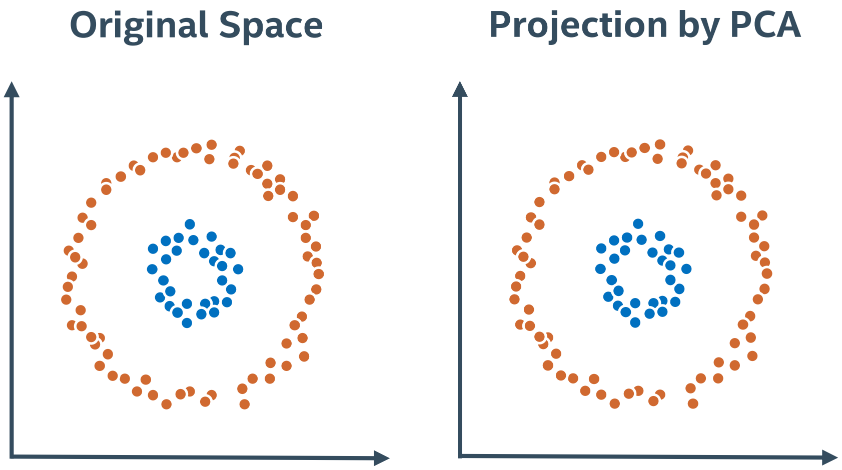
Fit the instance on the data and then transform the data.

```
X_trans = SVD.fit_transform(X_sparse)
```

Works with sparse matrices—used with text data for Latent Semantic Analysis (LSA).

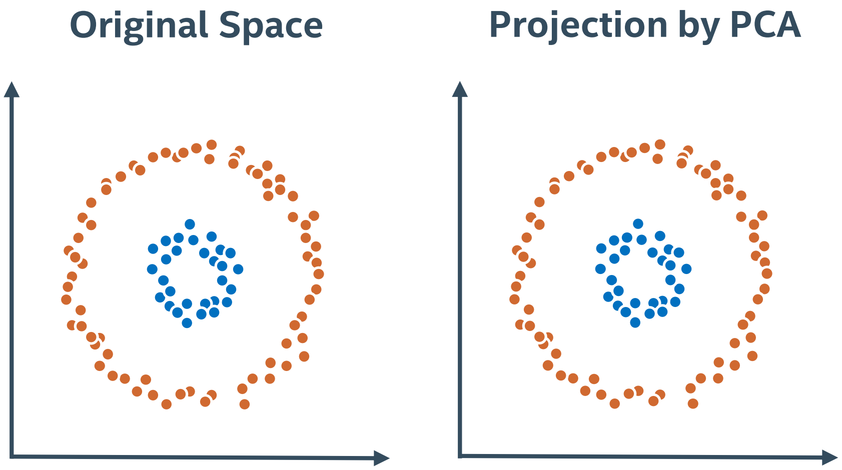
MOVING BEYOND LINEARITY

- Transformations calculated with PCA/SVD are linear



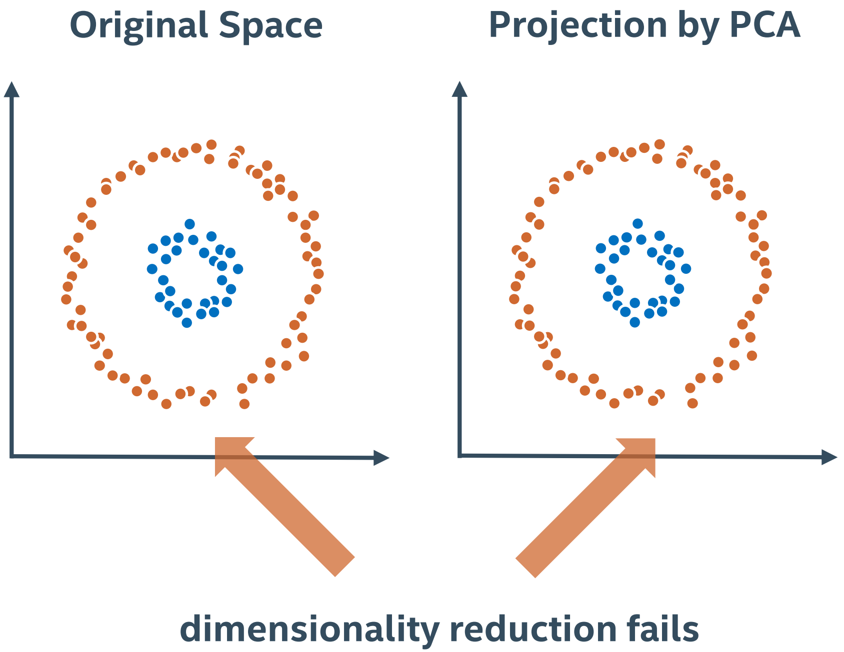
MOVING BEYOND LINEARITY

- Transformations calculated with PCA/SVD are linear
- Data can have non-linear features



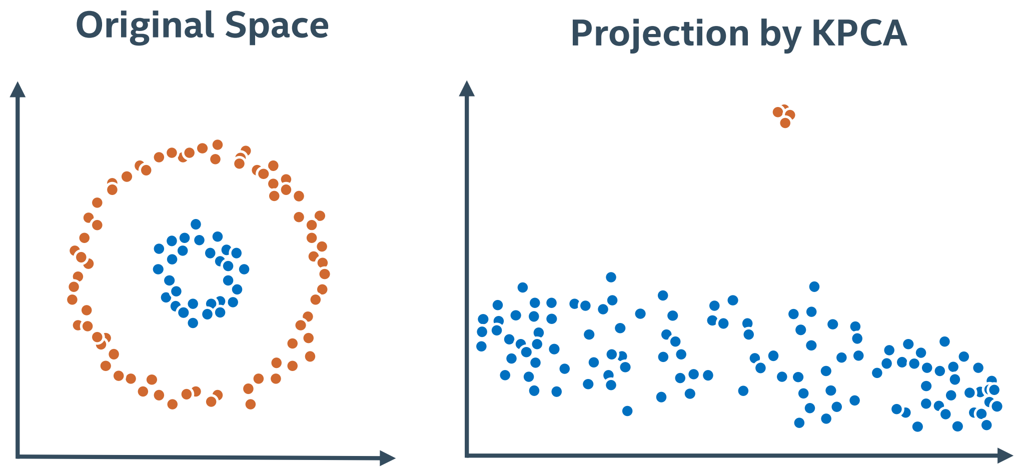
MOVING BEYOND LINEARITY

- Transformations calculated with PCA/SVD are linear
- Data can have non-linear features
- This can cause dimensionality reduction to fail



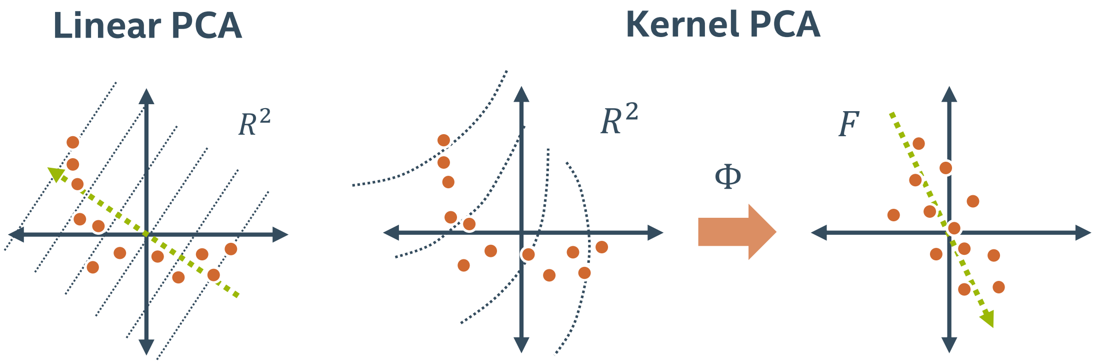
KERNEL PCA

- **Solution:** kernels can be used to perform non-linear PCA



KERNEL PCA

- **Solution:** kernels can be used to perform non-linear PCA
- Like the kernel trick introduced for SVMs



KERNEL PCA: THE SYNTAX

Import the class containing the dimensionality reduction method.

```
from sklearn.decomposition import KernalPCA
```

Create an instance of the class.

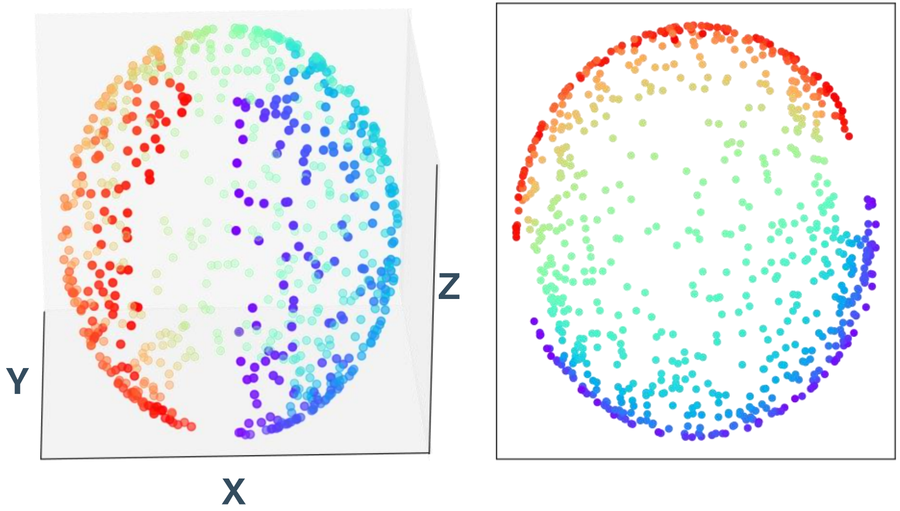
```
kPCA = KernalPCA(n_components=3, kernel='rbf',gamma=1.0)
```

Fit the instance on the data and then transform the data.

```
X_trans = kPCA.fit_transform(X_train)
```

MULTI-DIMENSIONAL SCALING (MDS)

- Non-linear transformation
- Doesn't focus on maintaining overall variance
- Instead, maintains geometric distances between points



MDS: THE SYNTAX

Import the class containing the dimensionality reduction method.

```
from sklearn.manifold import MDS
```

Create an instance of the class.

```
mdsMod = MDS(n_components=2)
```

Fit the instance on the data and then transform the data.

```
X_trans = mdsMod.fit_transform(X_sparse)
```

Many other manifold dimensionality methods exist: `Isomap`, `TSNE`.

USES OF DIMENSIONALITY REDUCTION

- Frequently used for high dimensionality data
- Natural language processing (NLP)—many word combinations
- Image-based data sets—pixels are features

Image Source: https://commons.wikimedia.org/wiki/File:Monarch_In_May.jpg

USES OF DIMENSIONALITY REDUCTION

- Divide image into 12 x 12 pixel sections

Image Source: https://commons.wikimedia.org/wiki/File:Monarch_In_May.jpg

USES OF DIMENSIONALITY REDUCTION

- Divide image into 12 x 12 pixel sections
- Flatten section to create row of data with 144 features

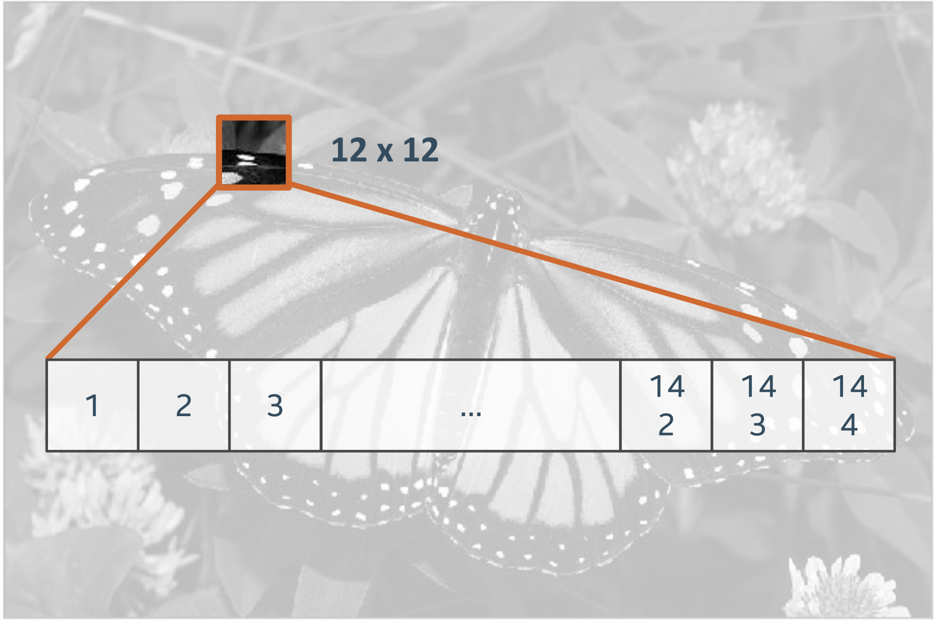


Image Source: https://commons.wikimedia.org/wiki/File:Monarch_In_May.jpg

USES OF DIMENSIONALITY REDUCTION

- Divide image into 12 x 12 pixel sections
- Flatten section to create row of data with 144 features
- Perform PCA on all data points

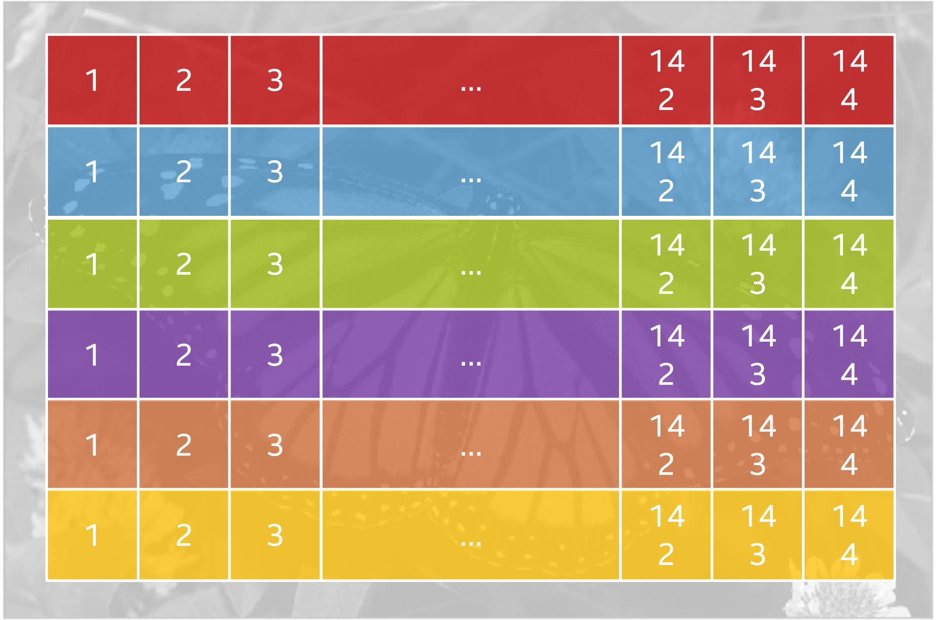


Image Source: https://commons.wikimedia.org/wiki/File:Monarch_In_May.jpg

PCA COMPRESSION: 144 → 60 DIMENSIONS

144 Dimensions

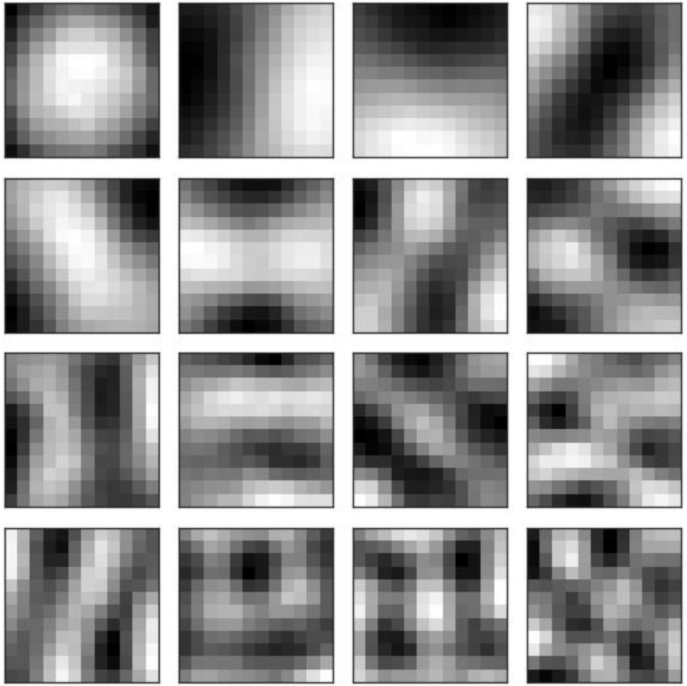
60 Dimensions

PCA COMPRESSION: 144 → 16 DIMENSIONS

144 Dimensions

16 Dimensions

SIXTEEN MOST IMPORTANT EIGENVECTORS

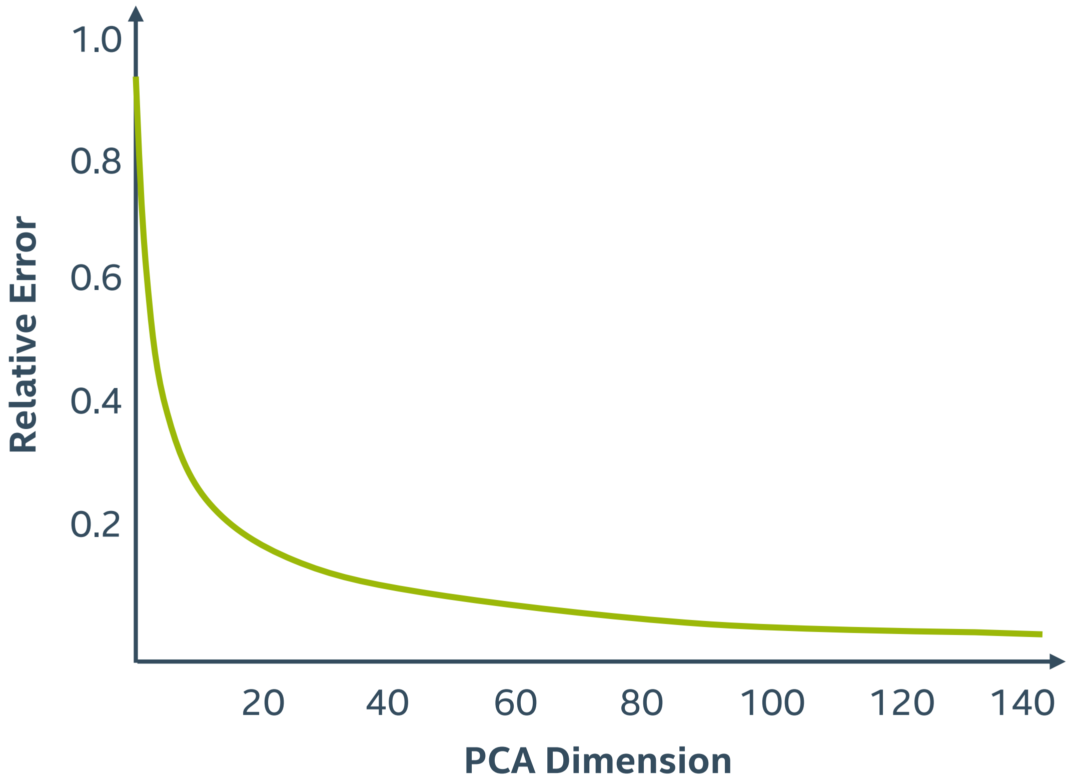


PCA COMPRESSION: 144 → 4 DIMENSIONS

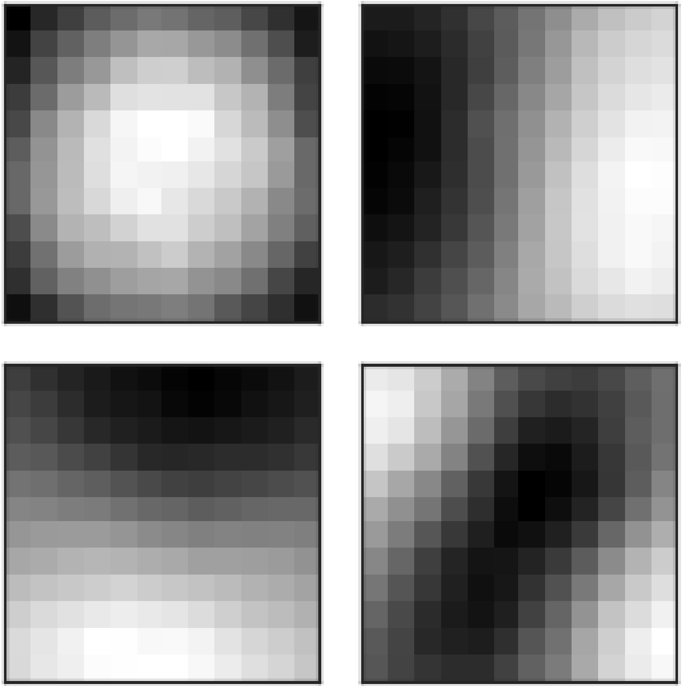
144 Dimensions

4 Dimensions

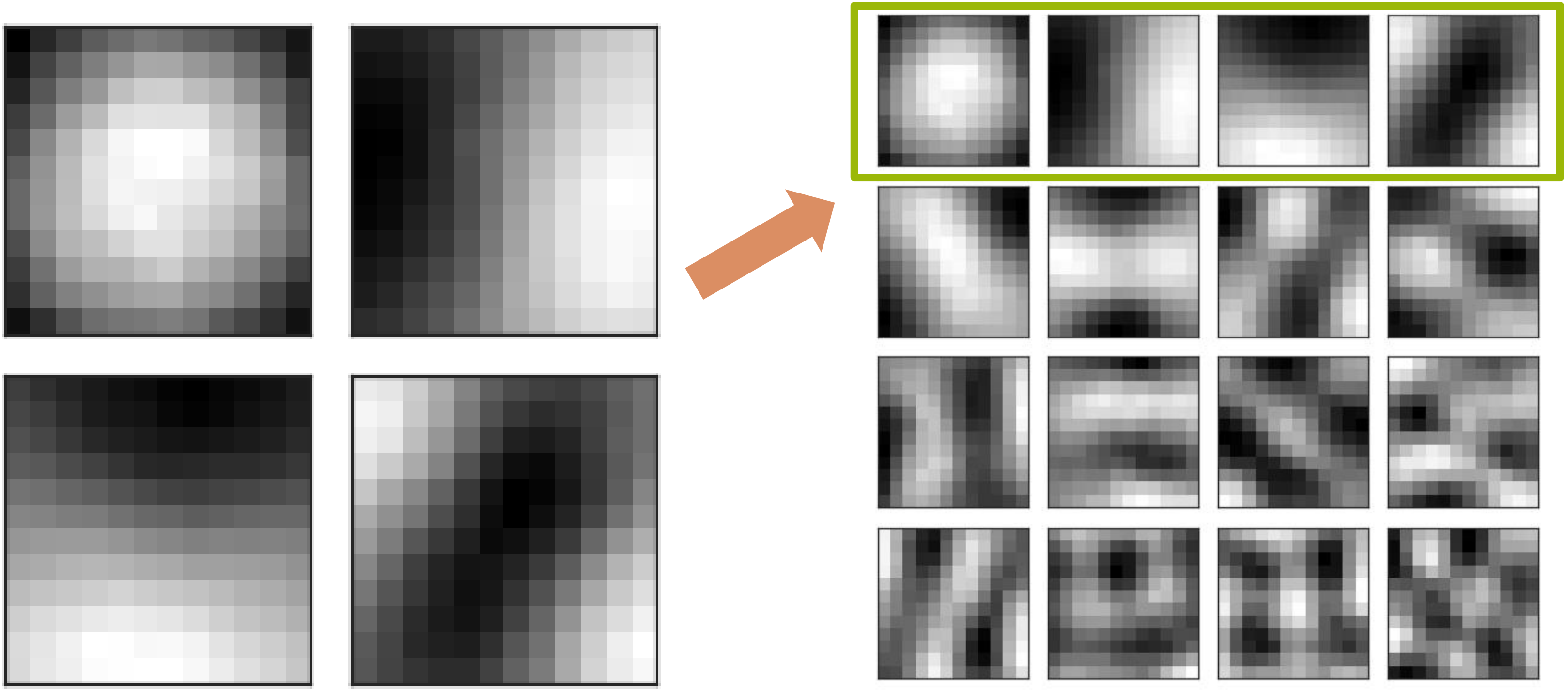
L2 ERROR AND PCA DIMENSION



FOUR MOST IMPORTANT EIGENVECTORS



FOUR MOST IMPORTANT EIGENVECTORS



PCA COMPRESSION: 144 → 1 DIMENSION

144 Dimensions

1 Dimensions

Nervana™ AI Academy