Principal Component Analysis

Machine Learning
CSx824/ECEx242
Bert Huang
Virginia Tech

	Example 1	Example 2	Example 3	Example 4	Example 5	Example 6	
Feature 1	0.3	20.1	-2.3	-6.4	0.2	-32.9	
Feature 2	200.4	108.2	428.3	352.8	722.0	50.3	
Feature 3	0.6	40.2	-4.6	-12.8	0.4	-65.8	2 (Feature 1)
Feature 4	1	1	1	1	1	1	vacuous
Feature 5	0	0	0	0	0	0	vacuous
Feature 6	-200.4	-108.2	-428.3	-352.8	-722	-50.3	- (Feature 2)
Feature 7	200.7	128.3	426.0	346.4	722.2	17.4	F1+F2

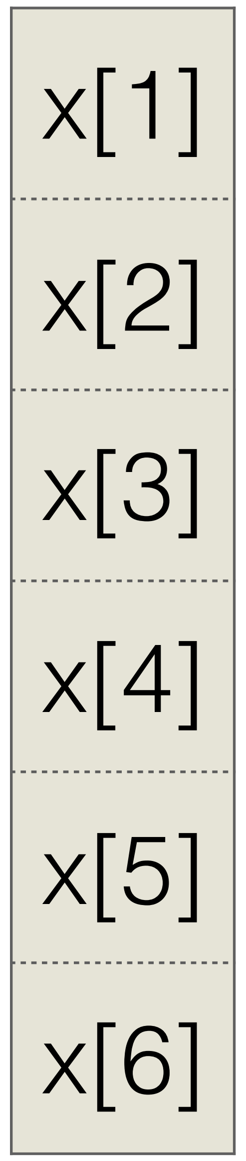
Outline

- Intuition behind principal component analysis (PCA)
- PCA recipe
- How PCA works

Vectors

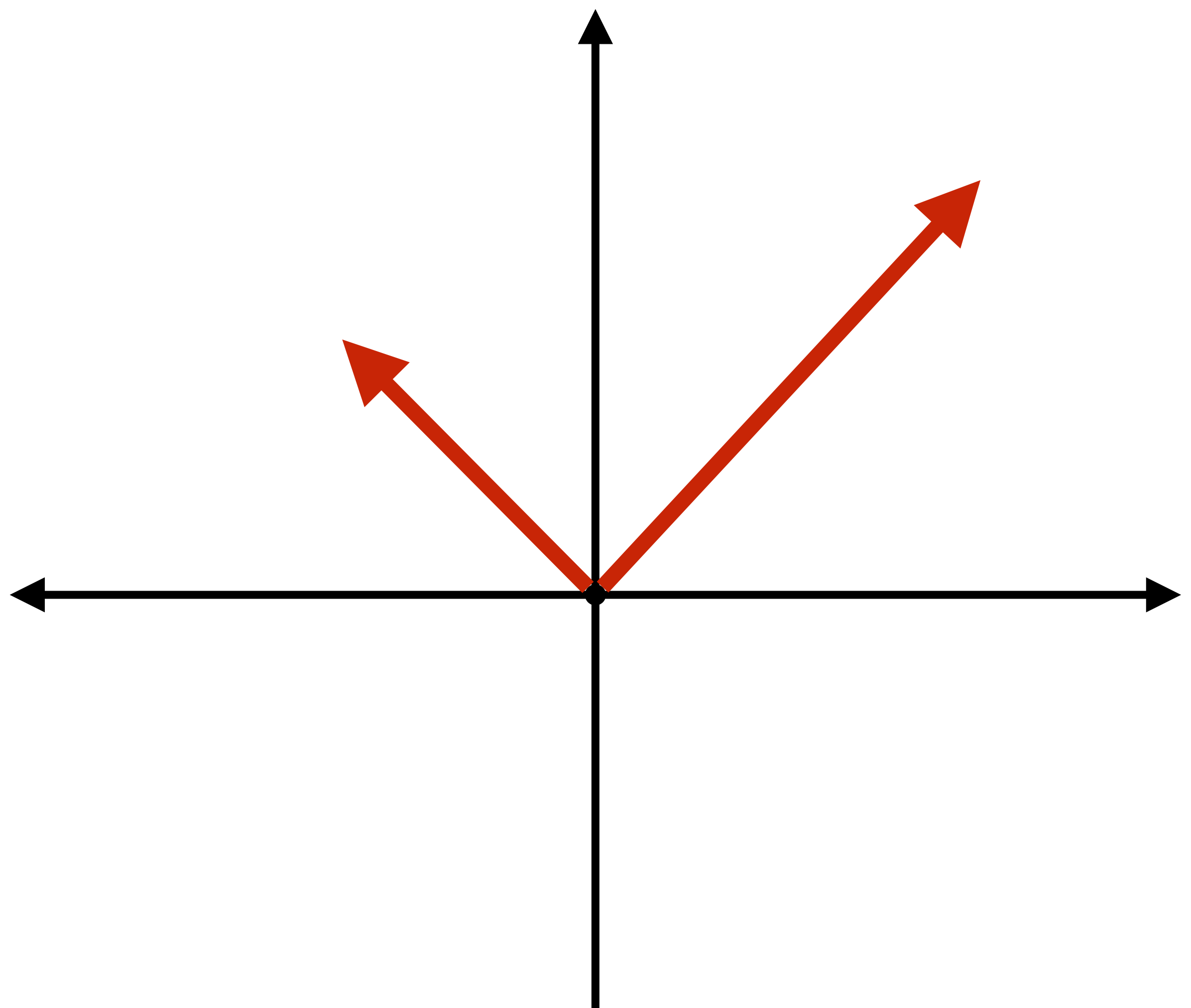
Programmers

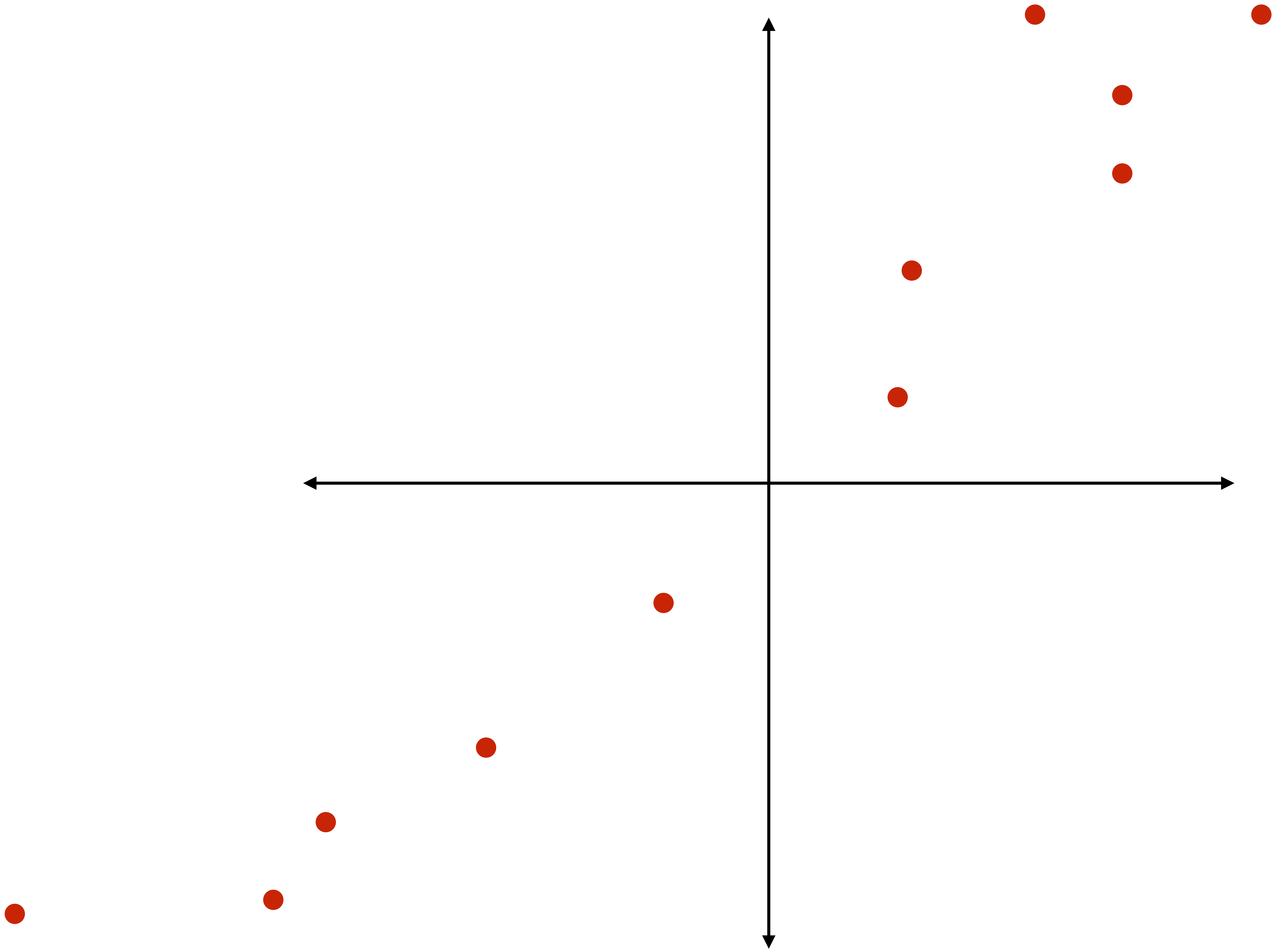
- Arrays
- Fixed basis

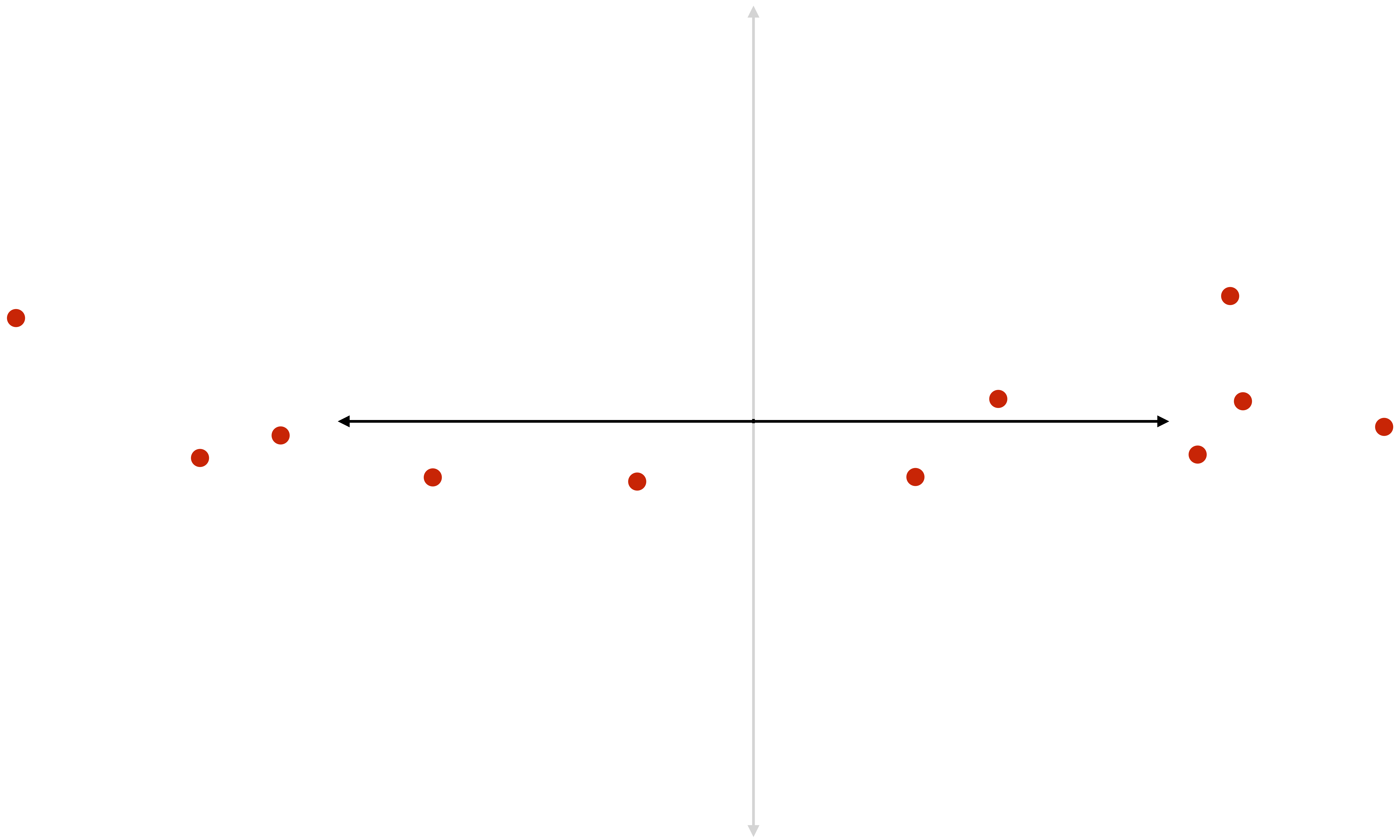


Mathematicians

- Direction in space and magnitude
- No fixed basis







PCA Intuition

- Find low-dimensional principal directions of data
  - low-dimensional representation of data that most accurately reconstructs original data

$$\frac{1}{n}||x_i - \hat{x}_i||^2$$

- Align orthogonal axis with data variance
- Use highest-variance dimensions; discard low-variance dimensions

PCA Intuition

- Find low-dimensional principal directions of data
  - low-dimensional representation of data that most accurately reconstructs original data

$$\frac{1}{n}||x_i - \hat{x}_i||^2$$

$$\min_{W,Z} \; \frac{1}{n}\sum_{i=1}^{n} ||x_i - Wz_i||^2 \qquad W \in \mathbb{R}^{d \times r}$$

$$\text{s.t.} \; W^\top W = I \qquad Z \in \mathbb{R}^{r \times n}$$

- Align orthogonal axis with data variance
- Use highest-variance dimensions; discard low-variance dimensions

Eigenvectors

eigenvector (unit vector)

$$Av = \lambda v$$

linear transformation
(stretching, shearing, rotating)

eigenvalue

Eigenvectors

$$Av = \lambda v$$

For symmetric matrices

$$Au = \lambda' u$$

$$v^\top A u = \lambda' v^\top u$$

$$u^\top A v = \lambda u^\top v$$

$$0 = (\lambda - \lambda') u^\top v$$

eigenvectors are orthogonal*

$$AV = VD$$

$$A = VDV^{-1}$$

$$A = VDV^{\top}$$

$$D = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_d \end{bmatrix}$$


docs.scipy.org

SciPy.org Sponsored By ENTHOUGHT

Scipy.org | Docs | NumPy v1.13 Manual | NumPy Reference | Routines | Linear algebra (`numpy.linalg`)

index | next | previous

numpy.linalg.eig

`numpy.linalg.eig`(*a*) [source]

Compute the eigenvalues and right eigenvectors of a square array.

Parameters: **a** : *(..., M, M) array*

Matrices for which the eigenvalues and right eigenvectors will be computed

Returns: **w** : *(..., M) array*

The eigenvalues, each repeated according to its multiplicity. The eigenvalues are not necessarily ordered. The resulting array will be of complex type, unless the imaginary part is zero in which case it will be cast to a real type. When *a* is real the resulting eigenvalues will be real (0 imaginary part) or occur in conjugate pairs

v : *(..., M, M) array*

The normalized (unit "length") eigenvectors, such that the column `v[:,i]` is the eigenvector corresponding to the eigenvalue `w[i]`.

(w is diagonal of ***D***)

PCA Recipe v.1

Input: centered data $X \in \mathbb{R}^{d \times n}$ $x_i \in \mathbb{R}^d$ $X\vec{1} = \vec{0}$

Covariance matrix $$\Sigma = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top = \frac{1}{n} XX^\top$$

Eigendecomposition $\Sigma = VDV^\top$ $V = [v_1, \ldots, v_d]$ $D = \begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_d \end{bmatrix}$

Truncate eigenvectors $V_r = [v_1, \ldots, v_r]$ $\lambda_k \geq \lambda_{k+1}$

Project onto truncated eigenvectors $z_i = V_r^\top x_i$ $Z = V_r^\top X$

$\hat{x}_i = V_r z_i$

Reconstruction

PCA Recipe v.2

Input: centered data $X \in \mathbb{R}^{d \times n}$ $x_i \in \mathbb{R}^d$ $X\vec{1} = \vec{0}$

Singular-value decomposition (SVD) of **transpose** $X^\top = USV^\top$ ***U***, ***V*** unitary & orthogonal
S diagonal

Truncate right singular vectors $V_r = [v_1, \dots, v_r]$

Project onto truncated right singular vectors $z_i = V_r^\top x_i$ $Z = V_r^\top X$

$$XX^\top = (VS^\top U^\top)(USV^\top) = V(S^\top S)V^\top = VDV^\top$$

$$U^\top U = I \qquad S^\top S = D$$

right singular vectors of (n x d) data matrix = eigenvectors of covariance matrix

How PCA Reduces Reconstruction Error

$$\min_{W,Z} \frac{1}{n}\sum_{i=1}^{n} ||x_i - Wz_i||^2 \qquad W \in \mathbb{R}^{d\times r}$$

$$\text{s.t.} \quad W^\top W = I \qquad Z \in \mathbb{R}^{r\times n}$$

$$(x_i - Wz_i)^\top(x_i - Wz_i) = x_i^\top x_i - 2x_i^\top W z_i + z_i^\top W^\top W z_i$$

$$= x_i^\top x_i - 2x_i^\top W z_i + z_i^\top z_i$$

$$\nabla_{z_i} = 2z_i - 2W^\top x_i \qquad z_i = W^\top x_i$$

$$x_i^\top x_i - 2z_i^\top z_i + z_i^\top z_i = x_i^\top x_i - z_i^\top z_i$$

$$\max_{W,Z} \frac{1}{n}\sum_{i=1}^{n} x_i^\top W W^\top x_i \text{ s.t. } W^\top W = I$$

$$\max_{W,Z} \frac{1}{n}\sum_{i=1}^{n} x_i^\top W W^\top x_i \text{ s.t. } W^\top W = I$$

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{r} x_i^\top w_k w_k^\top x_i$$ explicitly write out dimensions

$$\frac{1}{n}\sum_{k=1}^{r}\sum_{i=1}^{n} x_i^\top w_k w_k^\top x_i$$ flip nested summations

Let's focus on 1d case: $$\sum_{i=1}^{n} w_k^\top x_i x_i^\top w_k \quad = w_k^\top \Sigma w_k$$ covariance

$$L(w_k, \lambda_k) = w_k^\top \Sigma w_k + \lambda_k (w_k^\top w_k - 1)$$ Lagrangian for unitary constraint

$$\nabla_{w_k} L = 2\Sigma w_k - 2\lambda_k w_k$$ gradient wrt $\boldsymbol{w_k}$

$$\Sigma w_k = \lambda_k w_k$$ each $\boldsymbol{w_k}$ must be an eigenvector

$$w_k^\top \Sigma w_k = \lambda_k$$ -variance is the eigenvalue; choose greatest eigenvalue

Summary

- PCA is a few lines in numpy
    - roughly the same amount of code as configuring and using built-in PCA
- Eigen-decomposition or SVD equivalent
- Showed proof for 1-D PCA.

Closing Tidbits

- Princip**al** component analysis
- **e**igenvalue, **e**igenvector lowercase
- Intuition of covariance matrix as linear transformation?