Dimensionality Reduction
PCA

Machine Learning – CSE446
David Wadden (slides provided by Carlos Guestrin)
University of Washington

Feb 22, 2017

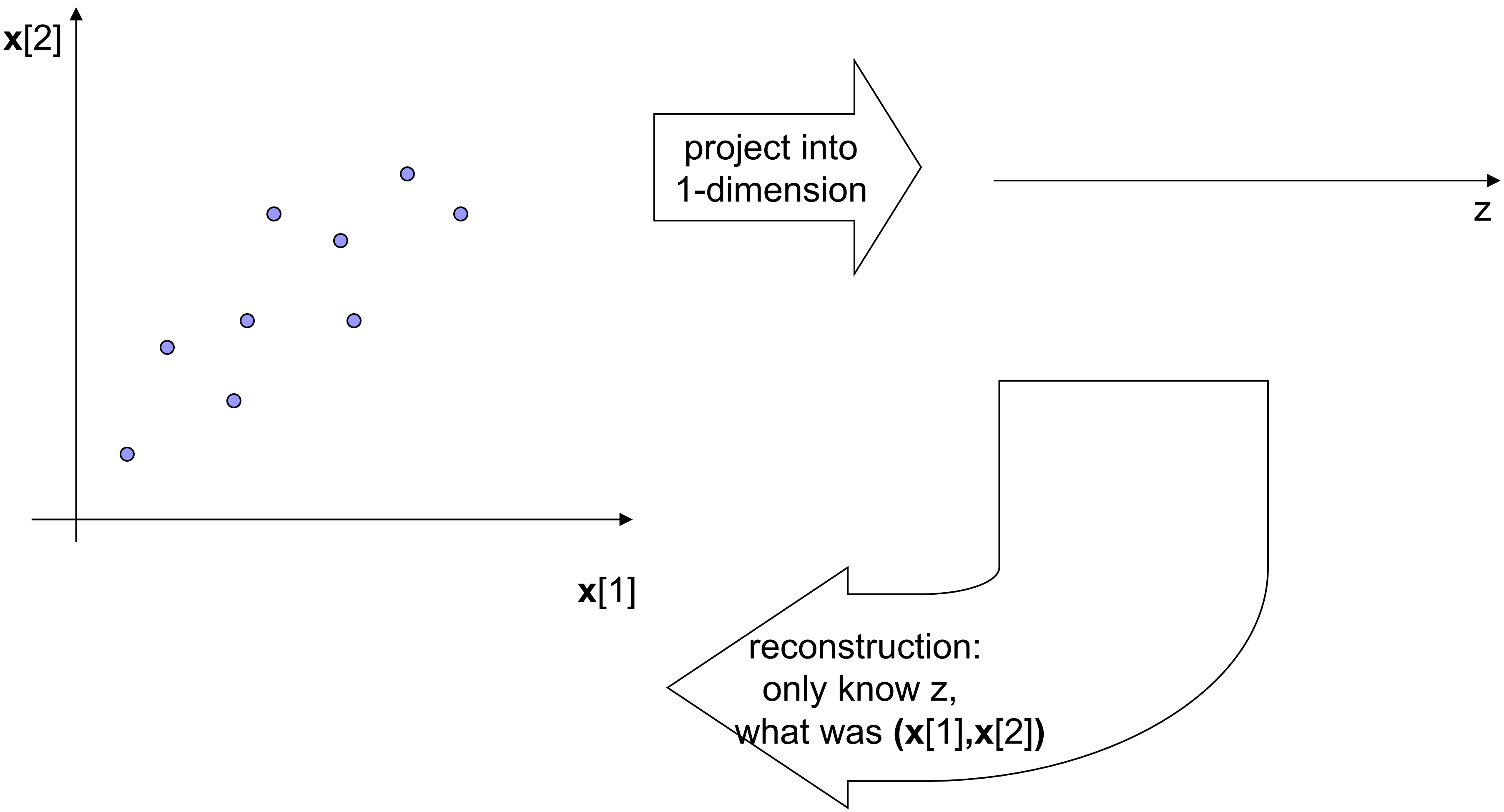
Dimensionality reduction

- Input data may have thousands or millions of dimensions!
  - e.g., text data has
- **Dimensionality reduction**: represent data with fewer dimensions
  - easier learning – fewer parameters
  - visualization – hard to visualize more than 3D or 4D
  - discover "intrinsic dimensionality" of data
    - high dimensional data that is truly lower dimensional

Lower dimensional projections

- Rather than picking a subset of the features, we can create new features that are combinations of existing features
- Let's see this in the unsupervised setting
  - just **X**, but no Y

Linear projection and reconstruction



Principal component analysis – basic idea

- Project d-dimensional data into k-dimensional space while preserving information:
  - e.g., project space of 10000 words into 3-dimensions
  - e.g., project 3-d into 2-d

- Choose projection with minimum reconstruction error

Linear projections, a review

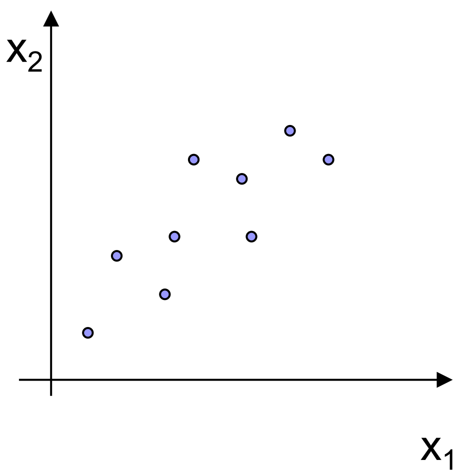
- Project a point into a (lower dimensional) space:
  - **point**: $\mathbf{x}_i = (\mathbf{x}_i[1],\ldots,\mathbf{x}_i[D])$
  - **select a basis** – set of basis vectors – $(\mathbf{u}_1,\ldots,\mathbf{u}_K)$
    - we consider orthonormal basis:
      - $\mathbf{u}_i \bullet \mathbf{u}_i = 1$, and $\mathbf{u}_i \bullet \mathbf{u}_j = 0$ for $i \neq j$
  - **select a center** – $\bar{\mathbf{x}}$, defines offset of space
  - **best coordinates** in lower dimensional space defined by dot-products: $(\mathbf{z}_i[1],\ldots,\mathbf{z}_i[K])$, $\mathbf{z}_i[j] = (\mathbf{x}_i - \bar{\mathbf{x}}) \cdot \mathbf{u}_j$

PCA finds projection that minimizes reconstruction error

- Given N data points: $\mathbf{x}_i = (\mathbf{x}_i[1],\ldots,\mathbf{x}_i[D])$, i=1…N
- Will represent each point as a projection:
  - $\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + \sum_{j=1}^{K} \mathbf{z}_i[j]\mathbf{u}_j$ where: $\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$ and $\mathbf{z}_i[j] = (\mathbf{x}_i - \bar{\mathbf{x}}) \cdot \mathbf{u}_j$

- PCA:
  - Given K<D, find $(\mathbf{u}_1,\ldots,\mathbf{u}_K)$ minimizing reconstruction error:

$$\text{error}_K = \sum_{i=1}^{N}(\mathbf{x}_i - \hat{\mathbf{x}}_i)^2$$



Understanding the reconstruction error

$$\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + \sum_{j=1}^{K} \mathbf{z}_i[j]\mathbf{u}_j$$

$$\mathbf{z}_i[j] = (\mathbf{x}_i - \bar{\mathbf{x}}) \cdot \mathbf{u}_j$$

- Given K<D, find ($\mathbf{u}_1,\ldots,\mathbf{u}_K$) minimizing reconstruction error:

$$\text{error}_K = \sum_{i=1}^{N} (\mathbf{x}_i - \hat{\mathbf{x}}_i)^2$$

- Note that $\mathbf{x}_i$ can be represented exactly by d-dimensional projection:

$$\mathbf{x}_i = \bar{\mathbf{x}} + \sum_{j=1}^{D} \mathbf{z}_i[j]\mathbf{u}_j$$

- Rewriting error:

$$\text{error}_K = \sum_{i=1}^{N} (\mathbf{x}_i - \hat{\mathbf{x}}_i)^2 = \sum_{i=1}^{N} \left[\bar{\mathbf{x}} + \sum_{j=1}^{D} \mathbf{z}_i[j]\mathbf{u}_j - \left(\bar{x} + \sum_{j=1}^{K} \mathbf{z}_i[j]\mathbf{u}_j\right)\right]^2$$

$$= \sum_{i=1}^{N} \left[\sum_{j=K+1}^{D} \mathbf{z}_i[j]\mathbf{u}_j\right]^2 = \sum_{i=1}^{N} \left[\sum_{j=K+1}^{D} \mathbf{z}_i[j]\mathbf{u}_j \cdot \mathbf{u}_j \mathbf{z}_i[j] + 2 \sum_{j=K+1}^{D} \sum_{\ell>j}^{D} \mathbf{z}_i[j]\mathbf{u}_j \cdot \mathbf{u}_\ell \mathbf{z}_i[\ell]\right]$$

$$= \sum_{i=1}^{N} \sum_{j=K+1}^{D} (\mathbf{z}_i[j])^2$$

Reconstruction error and covariance matrix

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

$$\sigma_{m\ell} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i[m] - \bar{\mathbf{x}}[m])(\mathbf{x}_i[\ell] - \bar{\mathbf{x}}[\ell])$$

$$\begin{aligned}
\text{error}_K &= \sum_{i=1}^{N}\sum_{j=K+1}^{D}\left[\mathbf{u}_j \cdot (\mathbf{x}_i - \bar{\mathbf{x}})\right]^2 \\
&= \sum_{i=1}^{N}\sum_{j=K+1}^{D}\mathbf{u}_j^T(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T\mathbf{u}_j \\
&= \sum_{j=K+1}^{D}\mathbf{u}_j^T\left[\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T\right]\mathbf{u}_j \\
&= N\sum_{j=K+1}^{D}\mathbf{u}_j^T\boldsymbol{\Sigma}\mathbf{u}_j
\end{aligned}$$

Minimizing reconstruction error and eigen vectors

- Minimizing reconstruction error equivalent to picking (ordered) orthonormal basis ($\mathbf{u}_1,\ldots,\mathbf{u}_D$) minimizing:

$$\text{error}_K = N \sum_{j=k+1}^{D} \mathbf{u}_j^T \mathbf{\Sigma} \mathbf{u}_j$$

- Eigen vector:

$$\mathbf{\Sigma}\mathbf{u} = \lambda \mathbf{u}$$

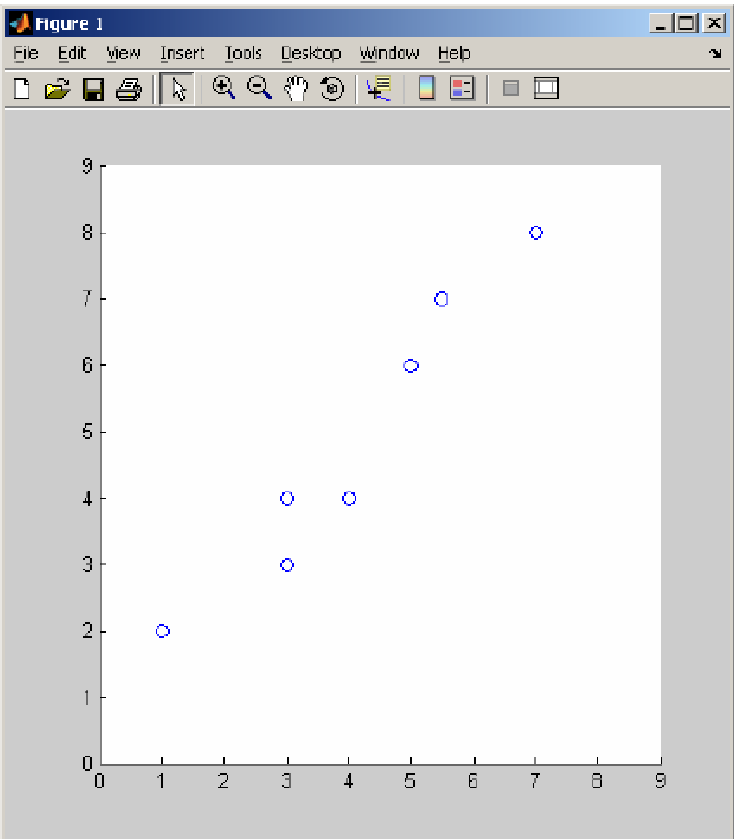
- Minimizing reconstruction error equivalent to picking ($\mathbf{u}_{K+1},\ldots,\mathbf{u}_D$) to be eigen vectors with smallest eigen values

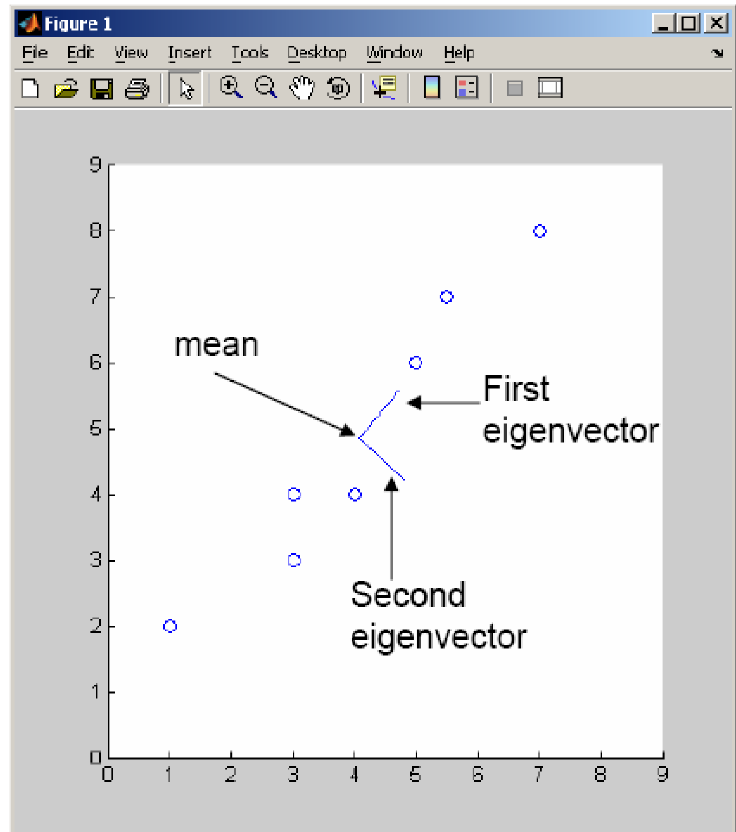
Basic PCA algoritm

- Start from m by n data matrix **X**
- **Recenter**: subtract mean from each row of **X**
  - $\mathbf{X}_c \leftarrow \mathbf{X} - \overline{\mathbf{X}}$
- **Compute covariance matrix**:
  - $\Sigma \leftarrow 1/N\ \mathbf{X}_c^T\ \mathbf{X}_c$
- Find **eigen vectors and values** of Σ
- **Principal components:** k eigen vectors with highest eigen values

PCA example

$$\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + \sum_{j=1}^{K} \mathbf{z}_i[j]\mathbf{u}_j$$

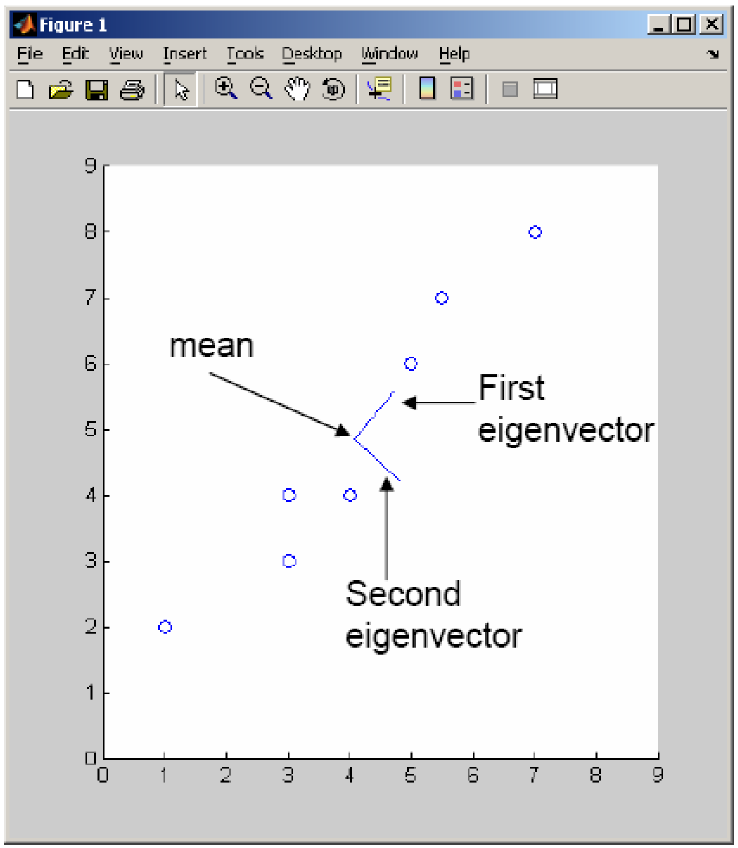


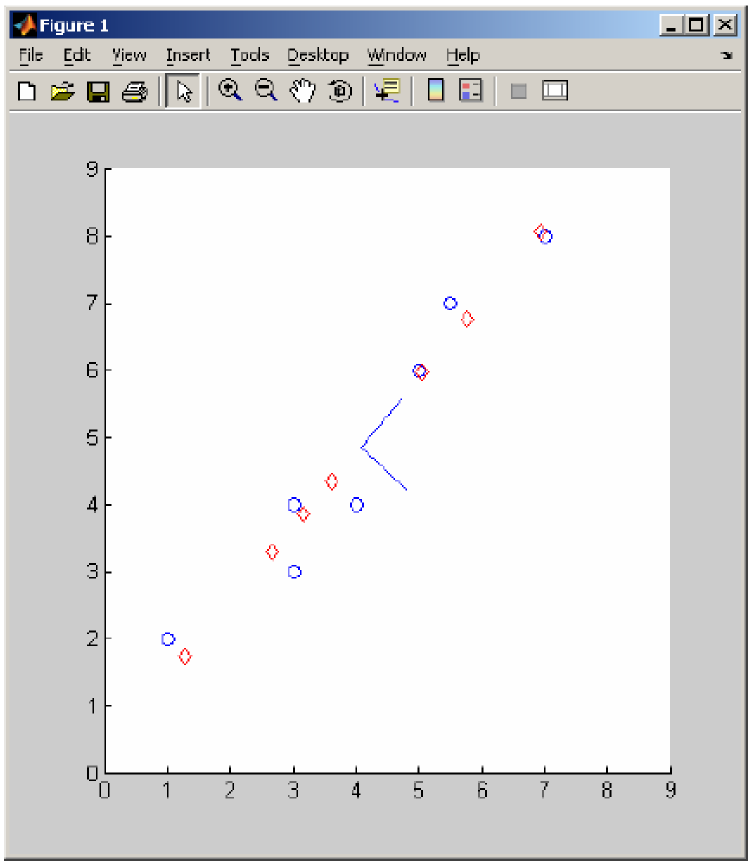


PCA example – reconstruction

$$\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + \sum_{j=1}^{K} \mathbf{z}_i[j]\mathbf{u}_j$$

only used first principal component

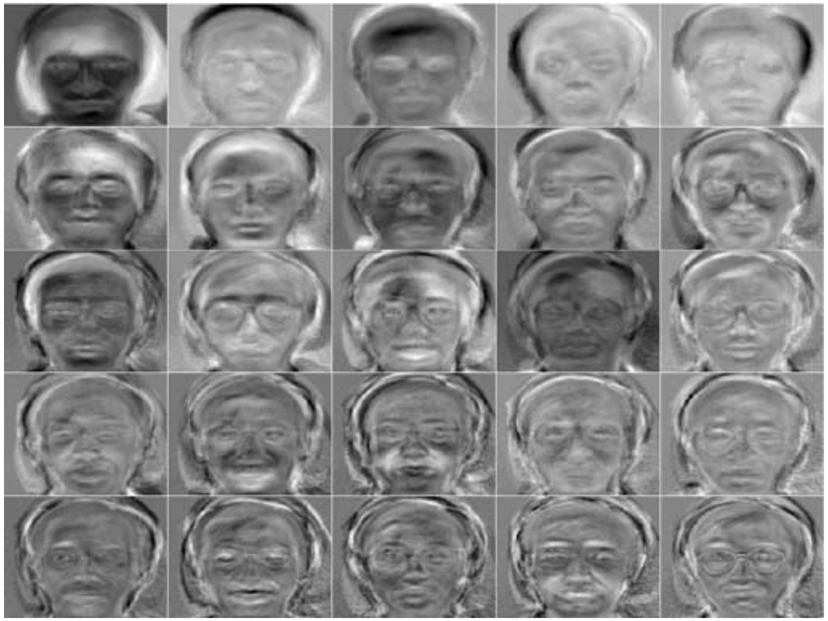




Eigenfaces [Turk, Pentland '91]

- Input images:

- Principal components:



Eigenfaces reconstruction

- Each image corresponds to adding 8 principal components:

Scaling up

- Covariance matrix can be really big!
  - Σ is D by D
  - Say, only 10000 features
  - finding eigenvectors is very slow…
- Use singular value decomposition (SVD)
  - finds to K eigenvectors
  - great implementations available, e.g., scipy.linalg.svd

SVD

- Write $\mathbf{X} = \mathbf{W}\ \mathbf{S}\ \mathbf{V}^T$
  - $\mathbf{X} \leftarrow$ data matrix, one row per datapoint
  - $\mathbf{W} \leftarrow$ weight matrix, one row per datapoint – coordinate of $\mathbf{x}_i$ in eigenspace
  - $\mathbf{S} \leftarrow$ singular value matrix, diagonal matrix
    - in our setting each entry is eigenvalue λ_j
  - $\mathbf{V}^T \leftarrow$ singular vector matrix
    - in our setting each row is eigenvector $\mathbf{v}_j$

PCA using SVD algoritm

- Start from m by n data matrix **X**
- **Recenter**: subtract mean from each row of **X**
  - $\mathbf{X_c} \leftarrow \mathbf{X} - \overline{\mathbf{X}}$
- Call SVD algorithm on $\mathbf{X_c}$ – ask for k singular vectors
- **Principal components:** k singular vectors with highest singular values (rows of $\mathbf{V}^T$)
  - **Coefficients** become:

What you need to know

- Dimensionality reduction
  - why and when it's important
- Simple feature selection
- Principal component analysis
  - minimizing reconstruction error
  - relationship to covariance matrix and eigenvectors
  - using SVD