Principal Component Analysis (PCA)

Salvador Dalí, "Galatea of the Spheres"

CSC411/2515: Machine Learning and Data Mining, Winter 2018

Michael Guerzhoy and Lisa Zhang

Some slides from Derek Hoiem and Alysha Efros

Announcements

- Graduate Project Proposals were reviewed
- Project 3 due March 19th
- Midterm Review Tutorial:
  - Wednesday 10am-12pm (SS2105)
  - Friday 4pm-6pm (BA2185)
- Remark Request to Tracy / Alex in BA4208

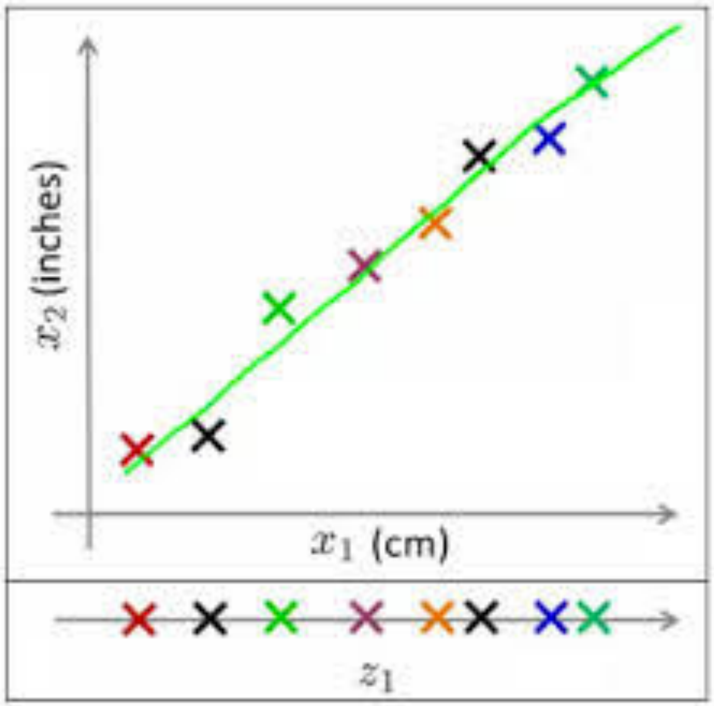
Principal Component Analysis (PCA)

Salvador Dalí, “Galatea of the Spheres”

Michael Guerzhoy and Lisa Zhang

Some slides from Derek Hoiem and Alysha Efros

Dimensionality Reduction

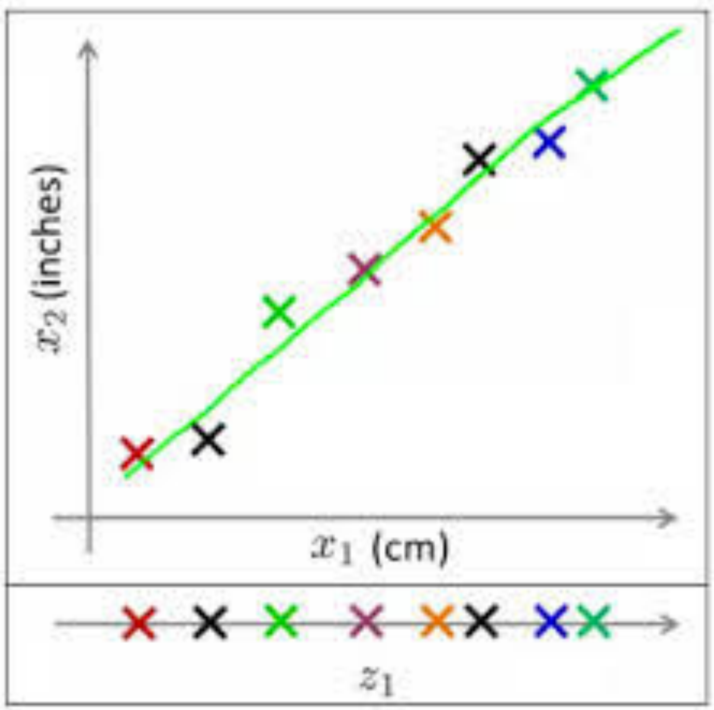


High Dimensional → Original data; Low Dimensional → Transformed

Original data	Transformed
(1, 1.2)	1.15
(2, 2)	2
(3, 3.3)	3.1
…	…

Dimensionality Reduction

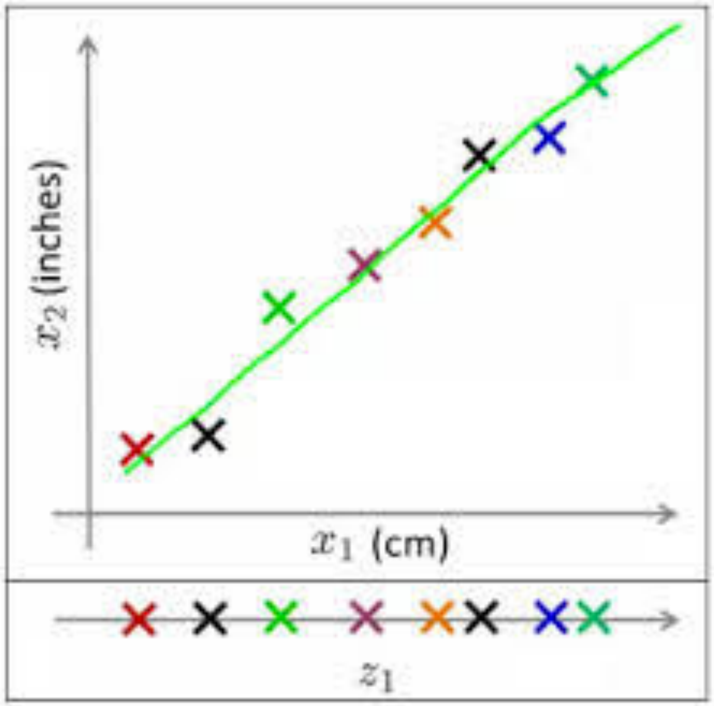
- Want to represent data in a new coordinate system with fewer dimensions
  - Cannot easily visualize n-D data, but can plot 2D or 3D data
  - Want to extract features from the data
    - Similar to extracting features with a ConvNet – easier to classify extracted features than original data
  - Want to compress the data while preserving most of the information



Original data	Transformed
(1, 1.2)	1.15
(2, 2)	2
(3, 3.3)	3.1
…	…

Dimensionality Reduction

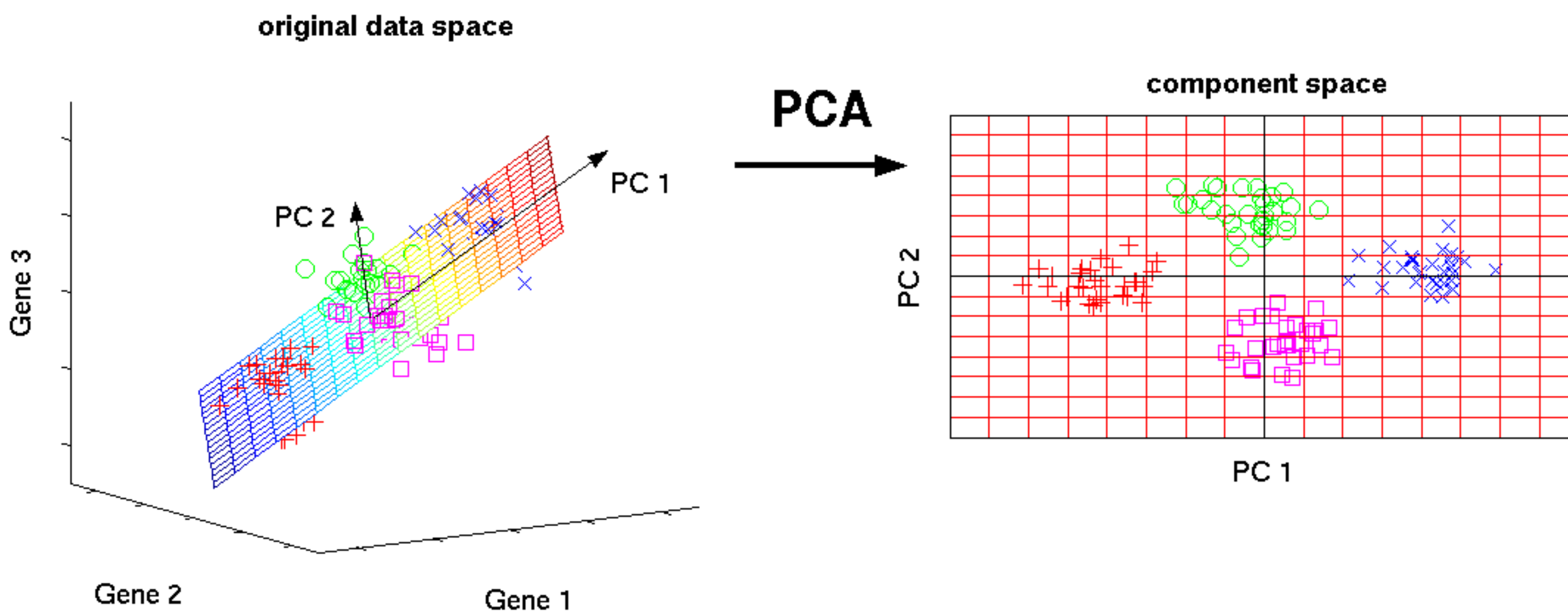
- Goal: preserve as much information as we can about the data in the new coordinate system
  - Preserve **distance** between data points
  - Preserve **variation** between data points



Original data	Transformed
(1, 1.2)	1.15
(2, 2)	2
(3, 3.3)	3.1
…	…

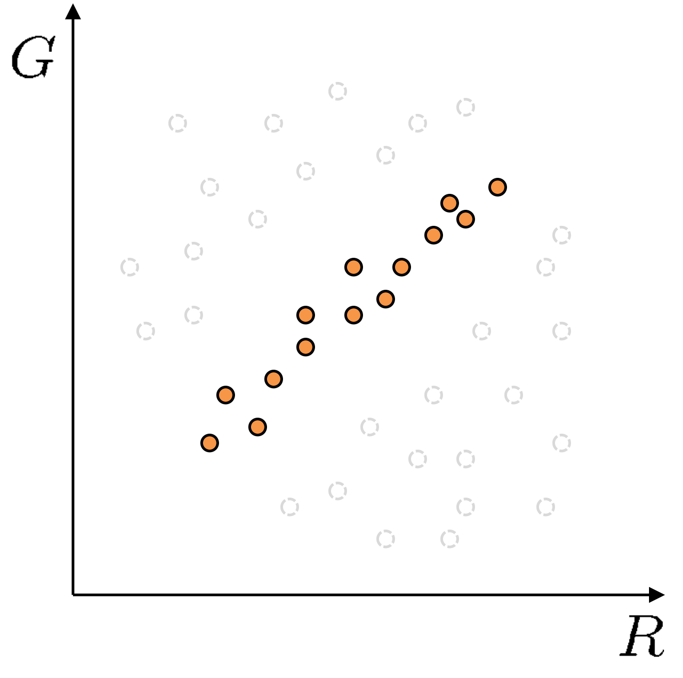
Principal Component Analysis

- **Linear** dimensionality reduction
  - The transformed data is a linear transformation of the original data
- Find a ~~subspace~~ hyperplane that the data lies in and project the data onto that ~~subspace~~ hyperplane
  - Usually we center the data first



Principal Component Analysis

- If all the data lies in a subspace, we can represent the data using the coordinates in that subspace

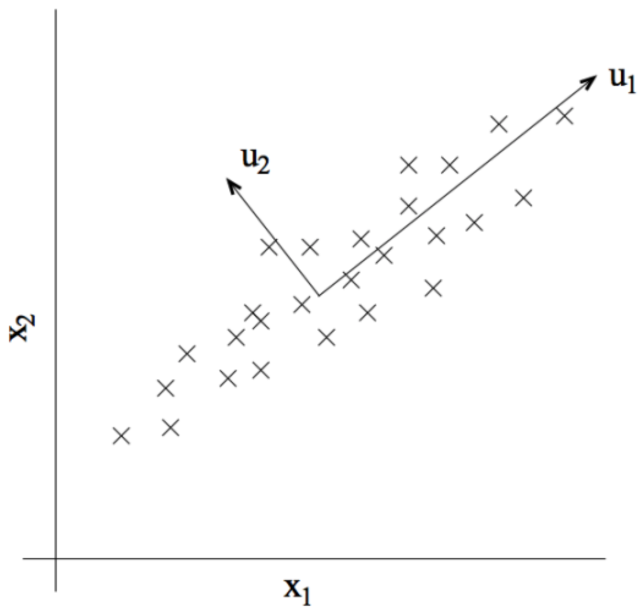


- Here: a 1D subspace arguably suffices
  - Just keep information about where in the orange cloud the point is – one dimension suffices

Principal Component Analysis

Key Idea

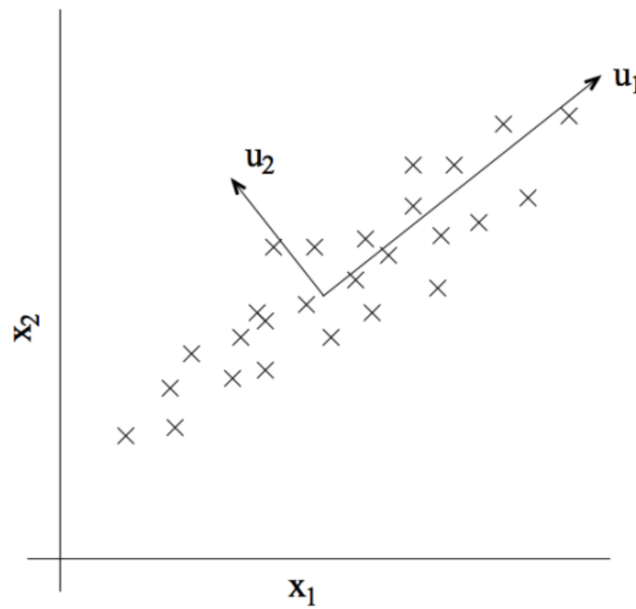
1. **Rotate** the data with some rotation matrix **R** (change of basis) so that the new features are **uncorrelated**
2. **Keep** the dimension with **the highest variance** (assumed to be most information)



Principal Component Analysis

In the picture:

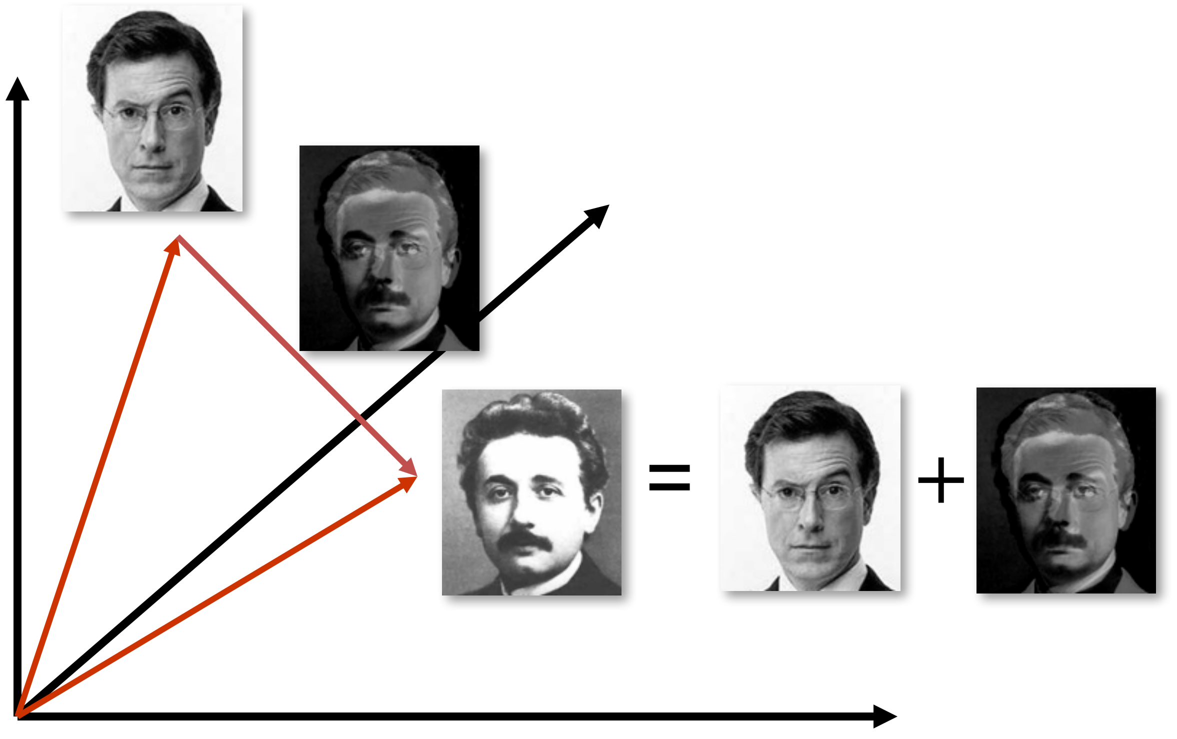
1. **R:** $x_1, x_2 \rightarrow u_1, u_2$
   - This is a change of basis
   - …which is a rotation
2. Can just keep u_1 and drop u_2 and keep most information about the data



Dataset of faces as an example

- When viewed as vectors of pixel values, face images are extremely high-dimensional
  - 100x100 image => 10,000 dimensions
- But very few 10,000-dimensional vectors are valid face images
- We want to effectively model the subspace of face images
  - Need a lot fewer dimensions than 10,000 dimensions for that

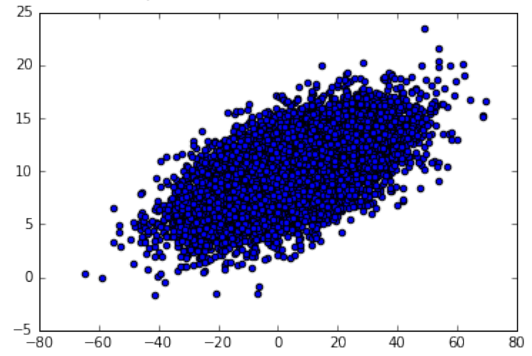
The space of faces



- Each images is a point in space
- Valid faces should lie in a low-dim subspace

Rotating a Cloud to Be Axis-Aligned

- Consider the covariance matrix of all the points in a cloud



- $\Sigma = \sum_i \left(x^{(i)} - \mu\right)\left(x^{(i)} - \mu\right)^T$
- The Spectral Theorem says we can diagonalize Σ (not covered in detail):

$$R^T \Sigma R = D = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_d \end{bmatrix},$$

R's columns are the Eigenvectors of Σ

No need to divide the sum by n when computing the covariance in this context

- Now:

$$\sum_i R\left(x^{(i)} - \mu\right)\left(R\left(x^{(i)} - \mu\right)^T\right) =$$

$$R\left(\sum_i \left(x^{(i)} - \mu\right)\left(x^{(i)} - \mu\right)^T\right)R^T$$

$$= R\Sigma R^T = D$$

- So if we rotate the $\left(x^{(i)} - \mu\right)$ using R, the covariance matrix of the transformed data will be diagonal!

Intuition

- If the covariance of the data is diagonal, the data lies in an axis-aligned cloud
- R, the matrix of the eigenvector of Σ, is the rotation matrix that needs to be applied to the data $\left(x^{(i)} - \mu\right)$ to make the cloud of datapoints axis aligned
  - Because we proved the covariance of the result will be diagonal

Change of Basis

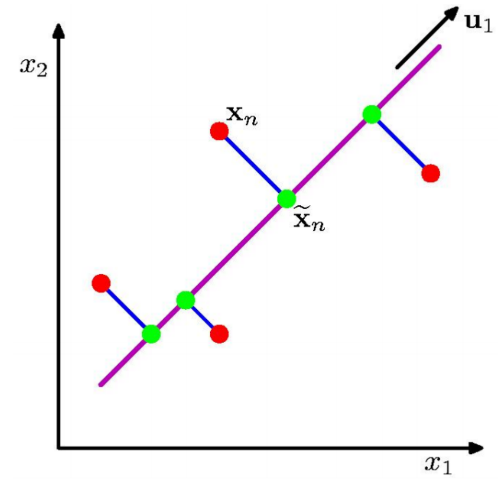
- (On the board)
- Main points
  - Review of change of basis
  - A rotation around 0 can be interpreted as a change of basis
  - If the data is centred at 0, rotating it means changing its basis
  - We can make data be centred at 0 by subtracting the mean from it

Reconstruction

- For a subspace with the orthonormal basis $V_k = \{v_0, v_1, v_2, \ldots v_k\}$, the best (i.e., closest) reconstruction of *x* in that subspace is:

$$\hat{x}_k = (x \cdot v_0)v_0 + (x \cdot v_1)v_1 + \cdots + (x \cdot v_k)v_k$$

  - If x is in the span of V_k, this is an exact reconstruction
  - If not, this is the projection of *x* on *V*
- Squared reconstruction error: $|(\hat{x}_k - x)|^2$



Reconstruction cont'd

- $\hat{x}_k = (x \cdot v_0)v_0 + (x \cdot v_1)v_1 + \cdots + (x \cdot v_k)v_k$

- Note: in $(x \cdot v_0)v_0$,
  - $(x \cdot v_0)$ is a measure of how similar x is to v_0
  - The more similar x is to v_0, the larger the contribution from v_0 is to the sum

Representation and reconstruction

- Face **x** in "face space" coordinates:

$$\mathbf{x} \rightarrow [\mathbf{u}_1^{\mathrm{T}}(\mathbf{x} - \mu), \ldots, \mathbf{u}_k^{\mathrm{T}}(\mathbf{x} - \mu)]$$
$$= \quad w_1, \ldots, w_k$$

- Reconstruction:

$\hat{x} \quad = \quad \mu \quad + \quad w_1u_1+w_2u_2+w_3u_3+w_4u_4+ \ldots$

slide by Derek Hoiem

Reconstruction

k = 4

k = 200

k = 400

After computing eigenfaces using 400 face images from ORL face database

slide by Derek Hoiem

Principal Component Analysis (PCA)

- Suppose the columns of a matrix $X_{N \times K}$ are the datapoints (N is the size of each image, K is the size of the dataset), and we would like to obtain an orthonormal basis of size k that produces the smallest sum of squared reconstruction errors for all the columns of $X - \bar{X}$
  - $\bar{X}$ is the average column of X
- Answer: the basis we are looking for is the *k* eigenvectors of $(X - \bar{X})(X - \bar{X})^T$ that correspond to the *k* largest eigenvalues

PCA – cont'd

- If x is the datapoint (obtained after subtracting the mean), and V an orthonormal basis, $V^T x$ is a column of the dot products of *x* and the elements of *x*
- So the reconstruction for the **centered** x is $\hat{x} = V(V^T x))$
- PCA is the procedure of obtaining the k eigenvectors V_k

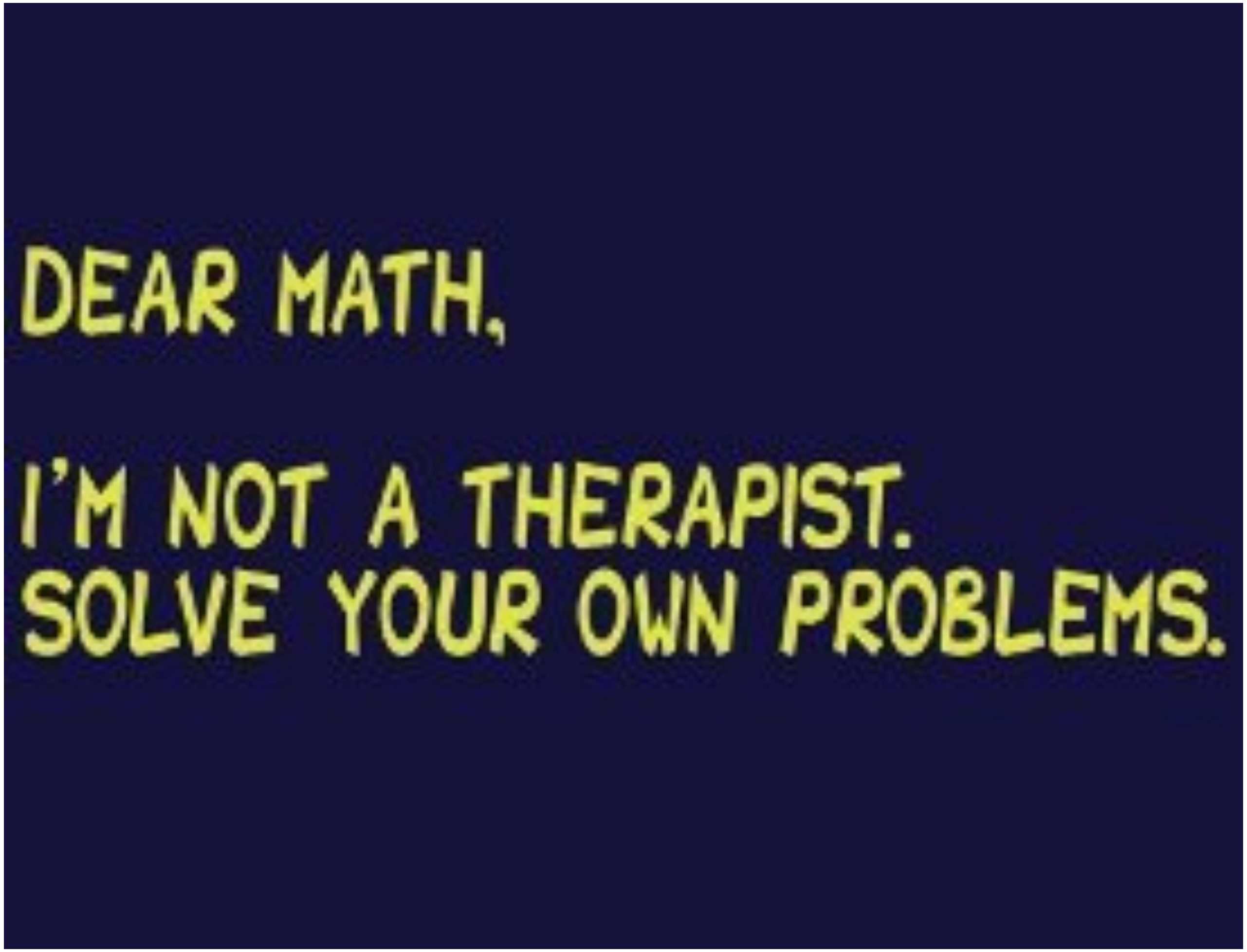
NOTE: centering

- If the image x is *not centred* (i.e., $\bar{X}$ was not subtracted from all the images), the reconstruction is:

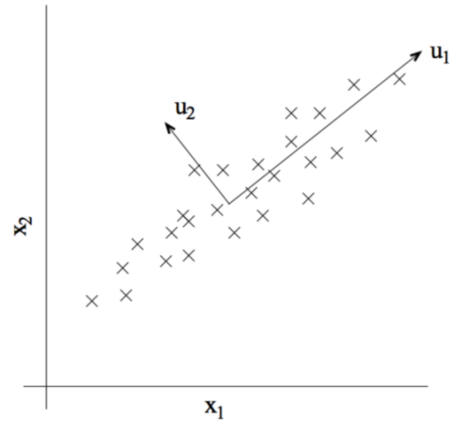
$$\hat{x} = \bar{X} + V(V^T(x - \bar{X}))$$

Proof that PCA produces the best reconstruction

Proof that PCA produces the best reconstruction



Reminder: Intuition for PCA

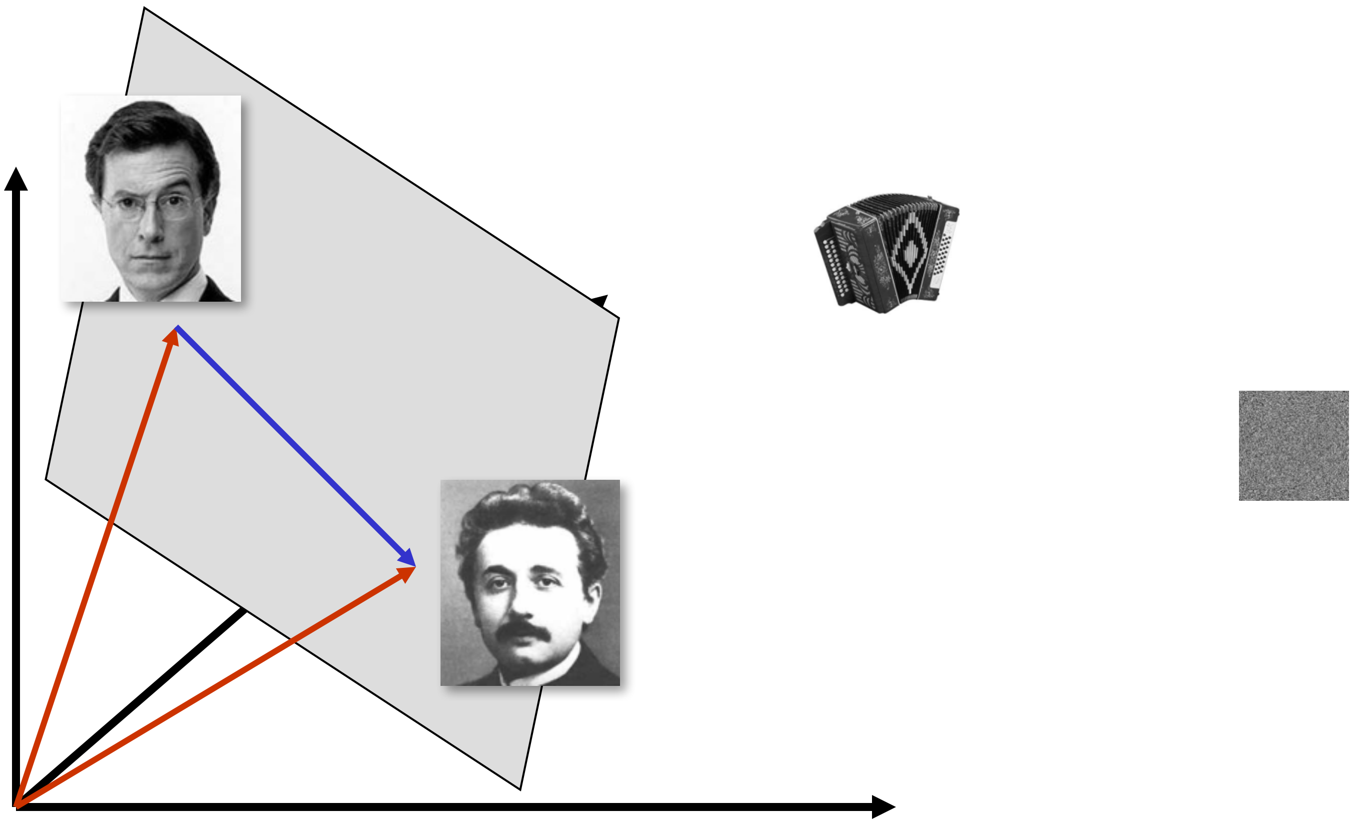


- The subspace where the reconstruction that will be the best is the major axis of the cloud
- We've shown that we are making the cloud axis-aligned by rotating it
  - So we know *one* of the basis elements will correspond to the best one-dimensional subspace
- The major axis is the Eigenvector which corresponds to the largest Eigenvalue
  - We haven't shown that (but we could)

Obtaining the Principal Components

- XX^T can be *huge*
- There are tricks to still compute the eigenvalues
  - Look up Singular Value Decomposition (SVD) if you're interested

EigenFace as dimensionality reduction



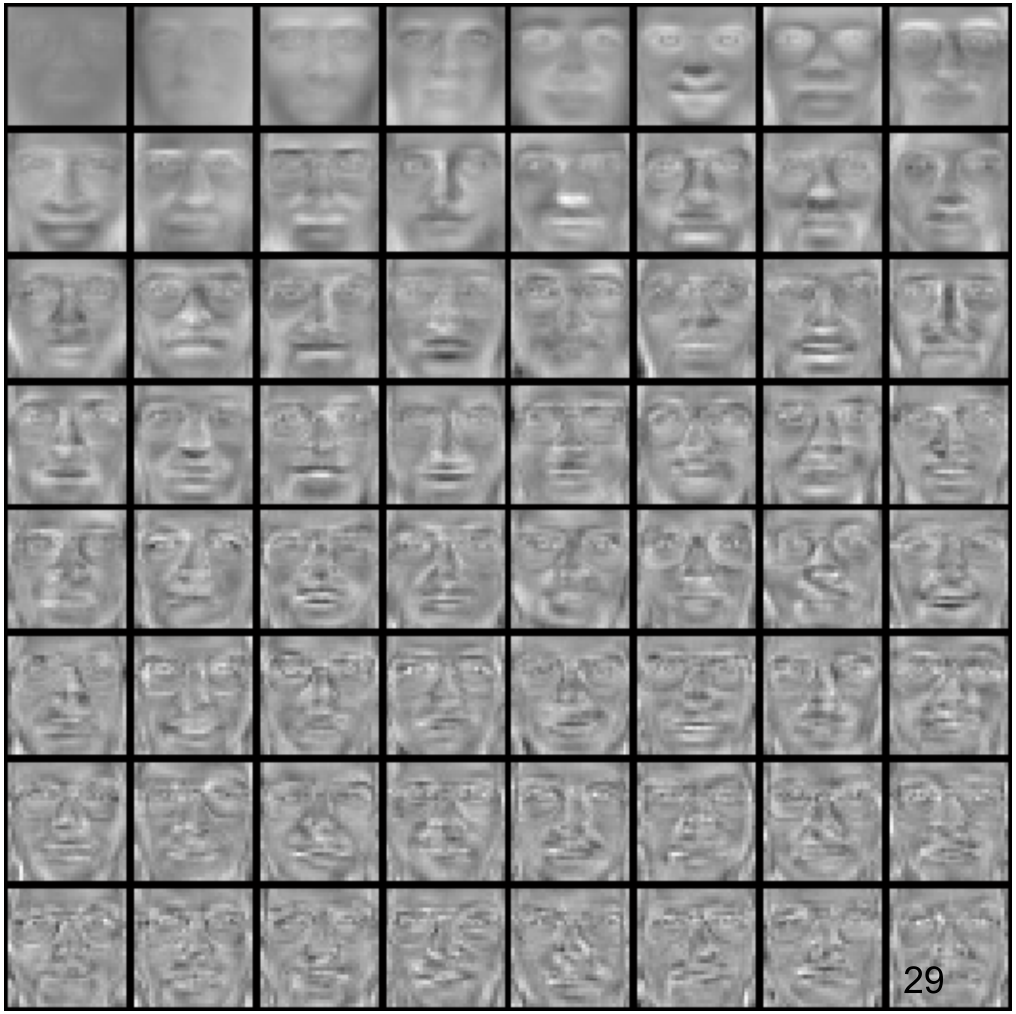
The set of faces is a "subspace" of the set of images

- Suppose it is K dimensional
- We can find the best subspace using PCA
- This is like fitting a "hyper-plane" to the set of (centred) faces
  - spanned by vectors $\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_K}$
  - any face $\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_k\mathbf{v_k}$

Eigenfaces example

Mean: μ

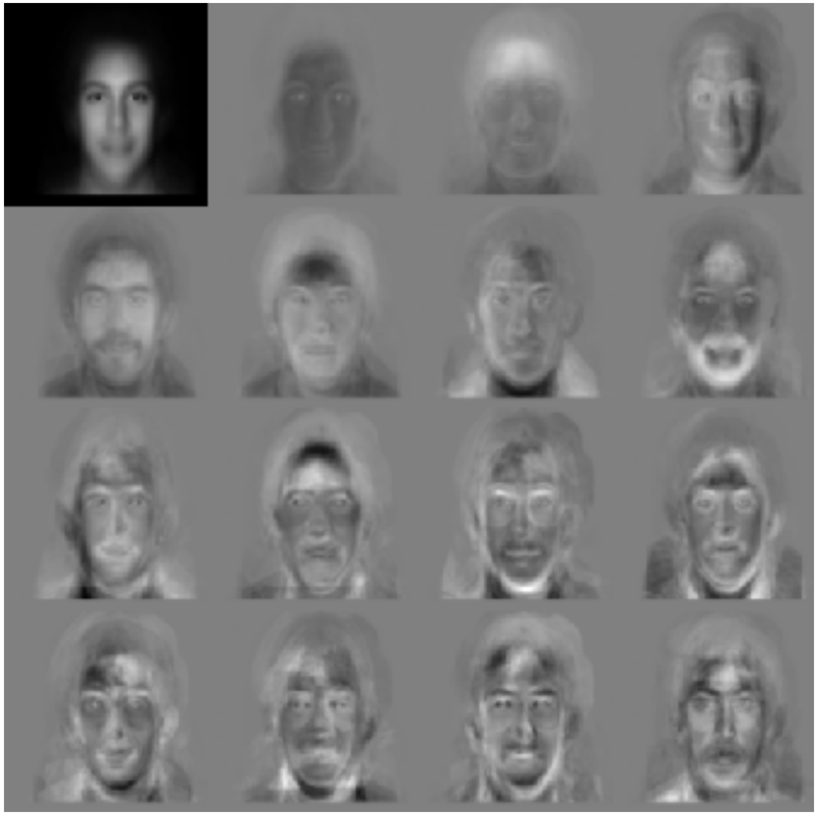
Top eigenvectors: $u_1, \ldots u_k$



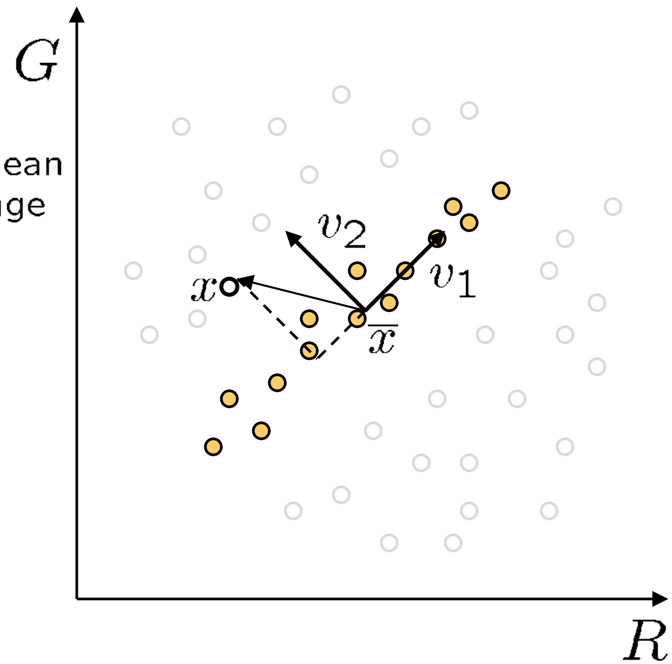
slide by Derek Hoiem

Another Eigenface set

With a smaller training set



PCA: Applications



convert **x** into $\mathbf{v_1}$, $\mathbf{v_2}$ coordinates

$$\mathbf{x} \rightarrow ((\mathbf{x} - \overline{x}) \cdot \mathbf{v_1}, (\mathbf{x} - \overline{x}) \cdot \mathbf{v_2})$$

What does the $\mathbf{v_2}$ coordinate measure?

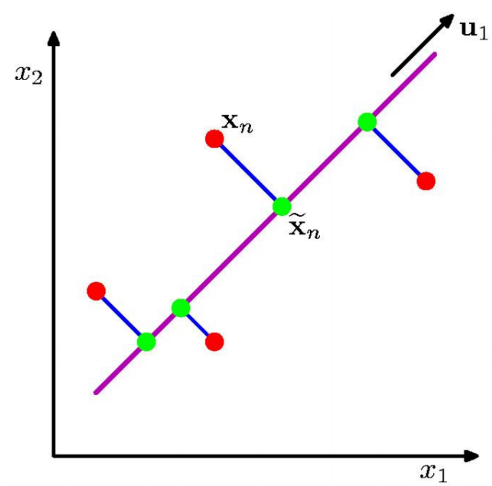
- Distance to line
- Use it for classification—near 0 for orange pts
- Possibly a good feature!

What does the $\mathbf{v_1}$ coordinate measure?

- Position along line
- Use it to specify which orange point it is
- Use $\mathbf{v_1}$ if want to compress data

Two views of PCA

- Want to minimize the squared distance between the original data and the reconstructed data, for a given dimensionality of the subspace k
  - Minimize the red-green distance per data point
  - Our approach so far
- Maximize the variance of the projected data, for a given dimensionality of the subspace k
  - Maximize the “scatter” (i.e., variance) of the green points
  - In general, maximize sum of [illegible] he components



Two views of PCA

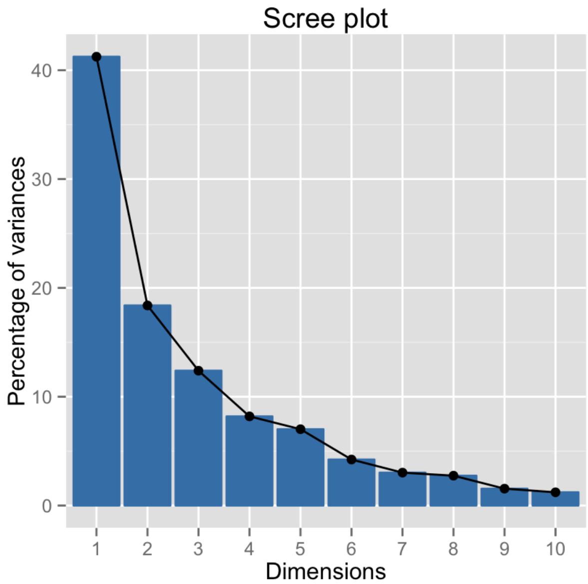
- $R^T \Sigma R = D = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_d \end{bmatrix}$
- D is the covariance matrix of the transformed data
- The variance of the projected data along the j-th component (coordinate) is $\lambda_j = \frac{1}{N} \sum_i \left(x_j^{(i)} - \overline{x_j} \right)^2$
- For a given k, when maximizing the variance of the projected data, we are trying to maximize
  - $\lambda_1 + \lambda_2 + \cdots + \lambda_k$
- We can show (but won't) that the variance maximization formulation and minimum square reconstruction error formulation produce the same k-dimensional bases

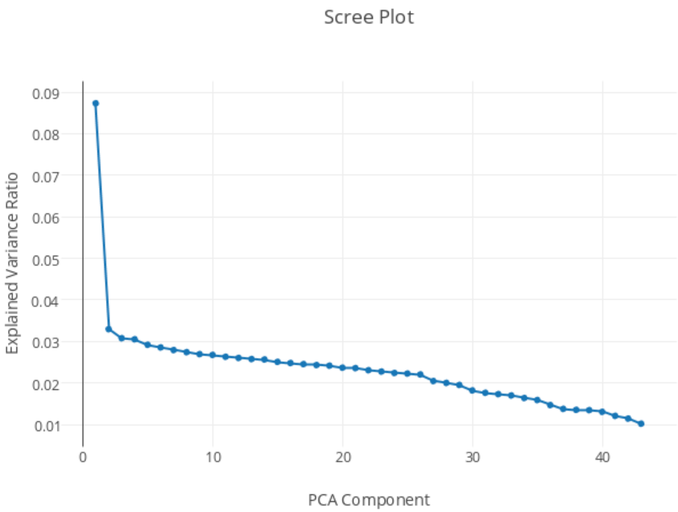
How to choose k?

- If need to visualize the dataset
  - Use k=2 or k=3
- If want to use PCA for feature extraction
  - Transform the data to be k-dimensional
  - Use cross-validation to select the best k

How to choose k?

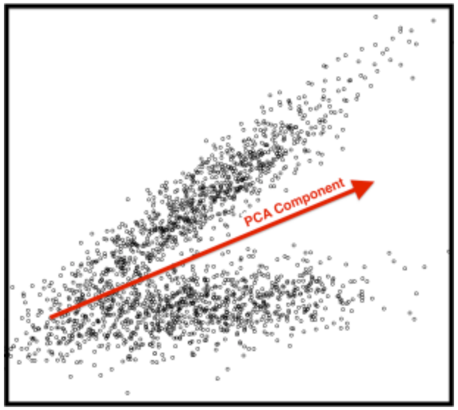
- Pick based on percentage of variance captured / lost
  - Variance captured: the variance of the projected data
- Pick smallest k that explains some % of variance
  - $(\lambda_1 + \lambda_2 + \cdots + \lambda_k)/(\lambda_1 + \lambda_2 + \cdots + \lambda_k + \cdots + \lambda_N)$
- Look for an "elbow" in Scree plot (plot of explained variance or eigenvalues)
  - In practice the plot is never very clean

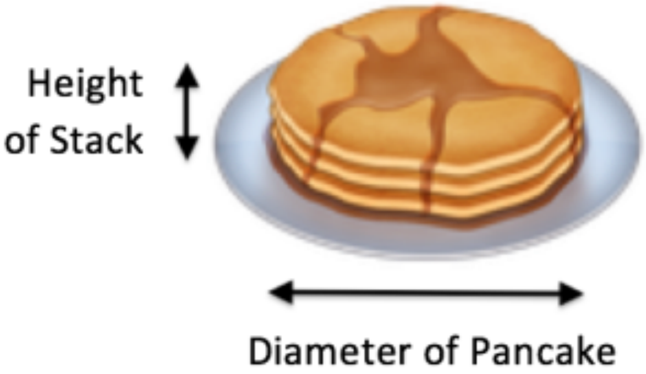


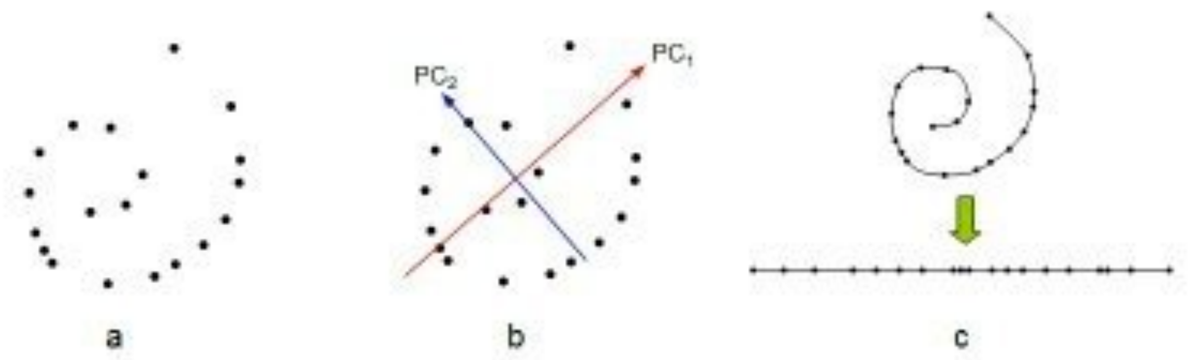


Limitations of PCA

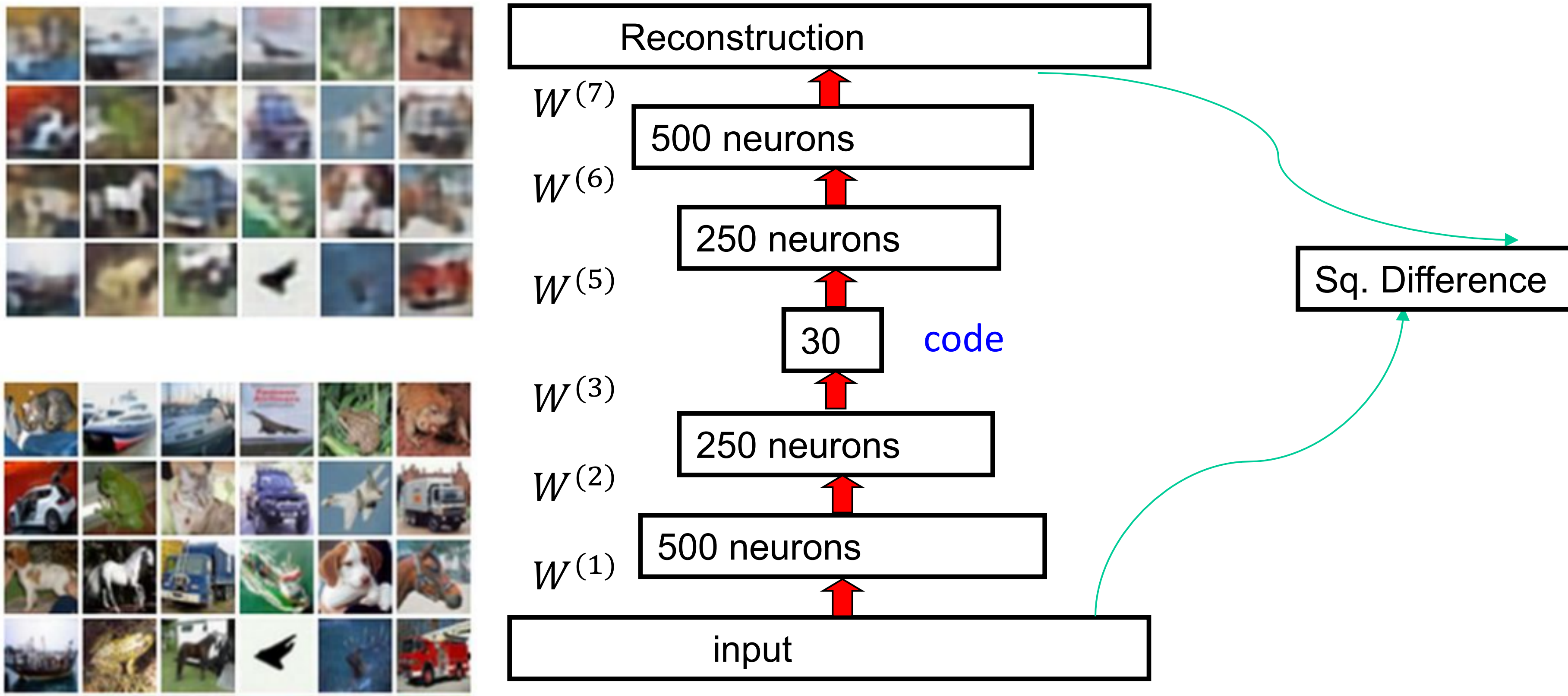
- Assumption: variance == information
- Assumption: the data lies in a linear subspace only







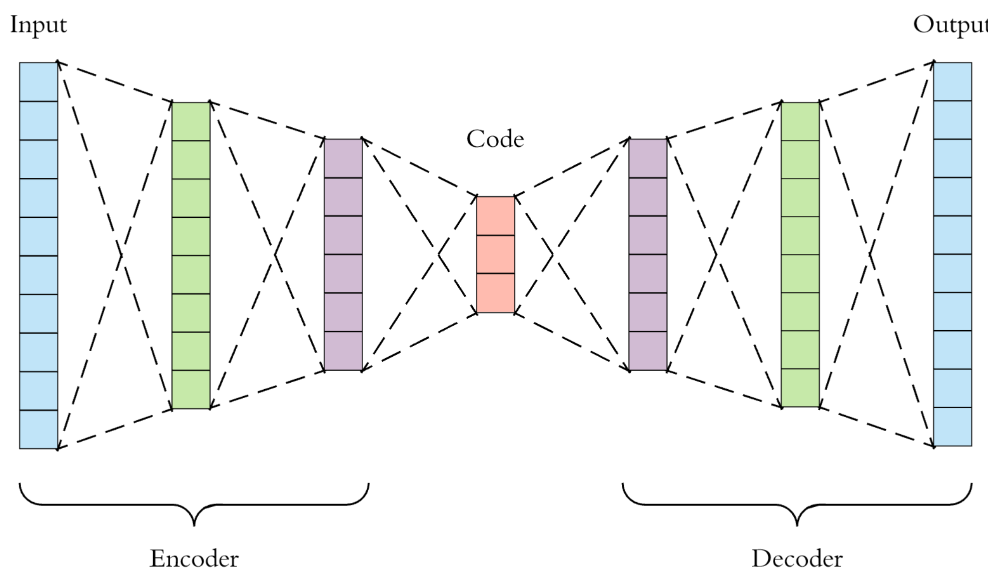
Autoencoders



- Find the weights that produce as small a difference as possible between the input and the reconstruction
- Train using Backprop
- The code layer is a low-dimensional summary of the input

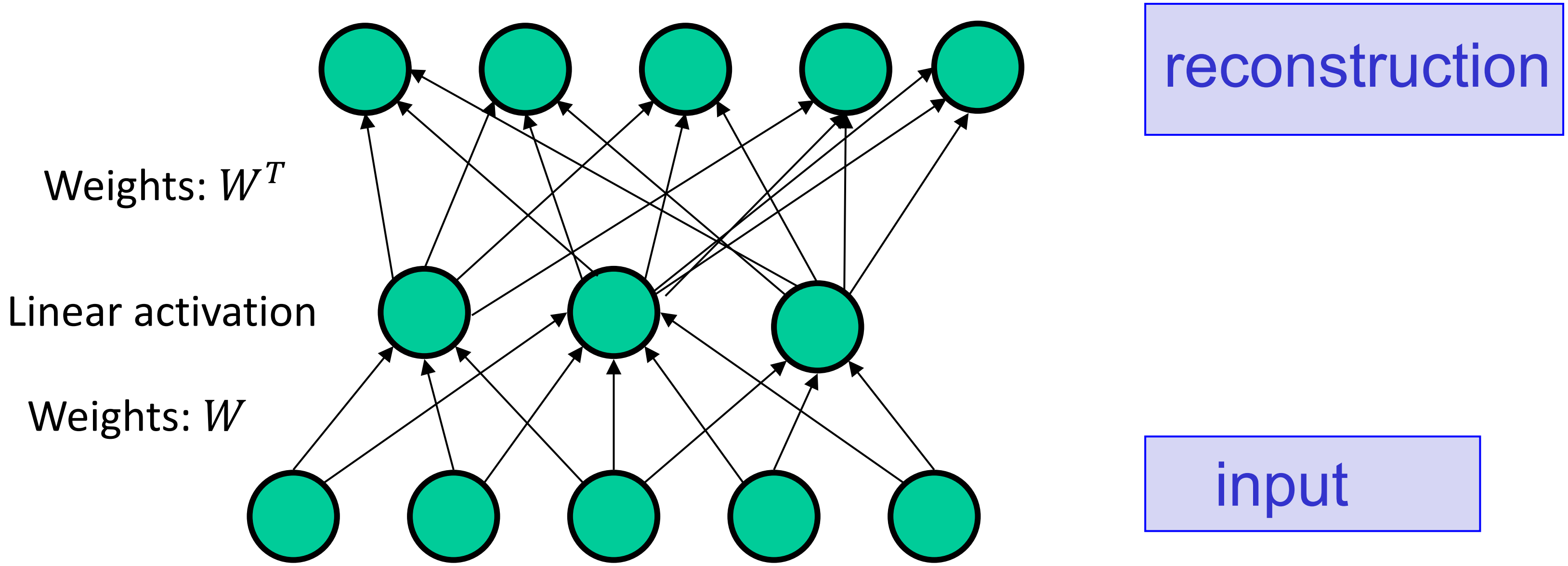
Autoencoders

- The **Encoder** can be used
  - To compress data
  - As a feature extractor
    - If you have lots of unlabeled data and some labeled data, train an autoencoder on unlabeled data, then a classifier on the features (semi-supervised learning)
- The **Decoder** can be used to generate images given a new code



Autoencoders and PCA

- PCA can be viewed as a special case of an autoencoder
- We can "tie" the encoder and decoder weights to make sure the architecture works the same way PCA does
  - This is sometimes useful for regularization



- Input: x
- Hidden layer: $W^T x$
- Reconstruction: $W W^T x$
- Minimize $\sum_i |W W^T x - x|^2$ -- the square reconstruction error, as in PCA
- Just one set of weight parameters, transposed