

***Curse of Dimensionality,
Dimensionality Reduction with PCA***

Curse of Dimensionality: Overfitting

- If the number of features d is large, the number of samples n , may be too small for accurate parameter estimation.

- For example, covariance matrix has d^2 parameters:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \cdots & \sigma_{1d} \\ \vdots & \ddots & \vdots \\ \sigma_{d1} & \cdots & \sigma_d^2 \end{bmatrix}$$

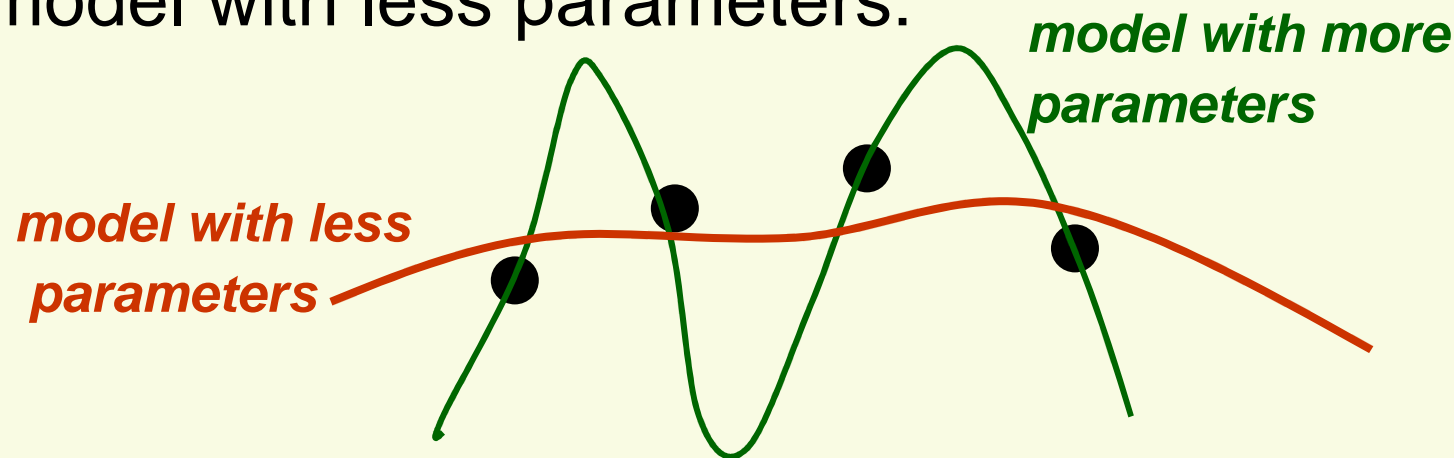
- For accurate estimation, n should be much bigger than d^2 , otherwise model is too complicated for the data, **overfitting**:

Curse of Dimensionality: Overfitting

- Paradox: If $n < d^2$ we are better off assuming that features are uncorrelated, even if we know this assumption is wrong
- In this case, the covariance matrix has only d parameters:

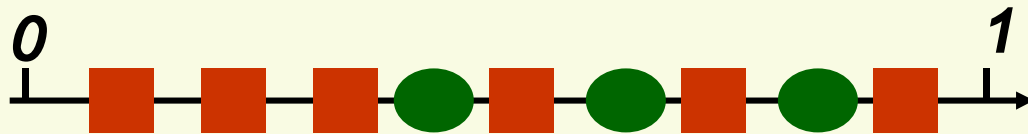
$$\Sigma = \begin{bmatrix} \sigma_1^2 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \sigma_d^2 \end{bmatrix}$$

- We are likely to avoid overfitting because we fit a model with less parameters:



Curse of Dimensionality: Number of Samples

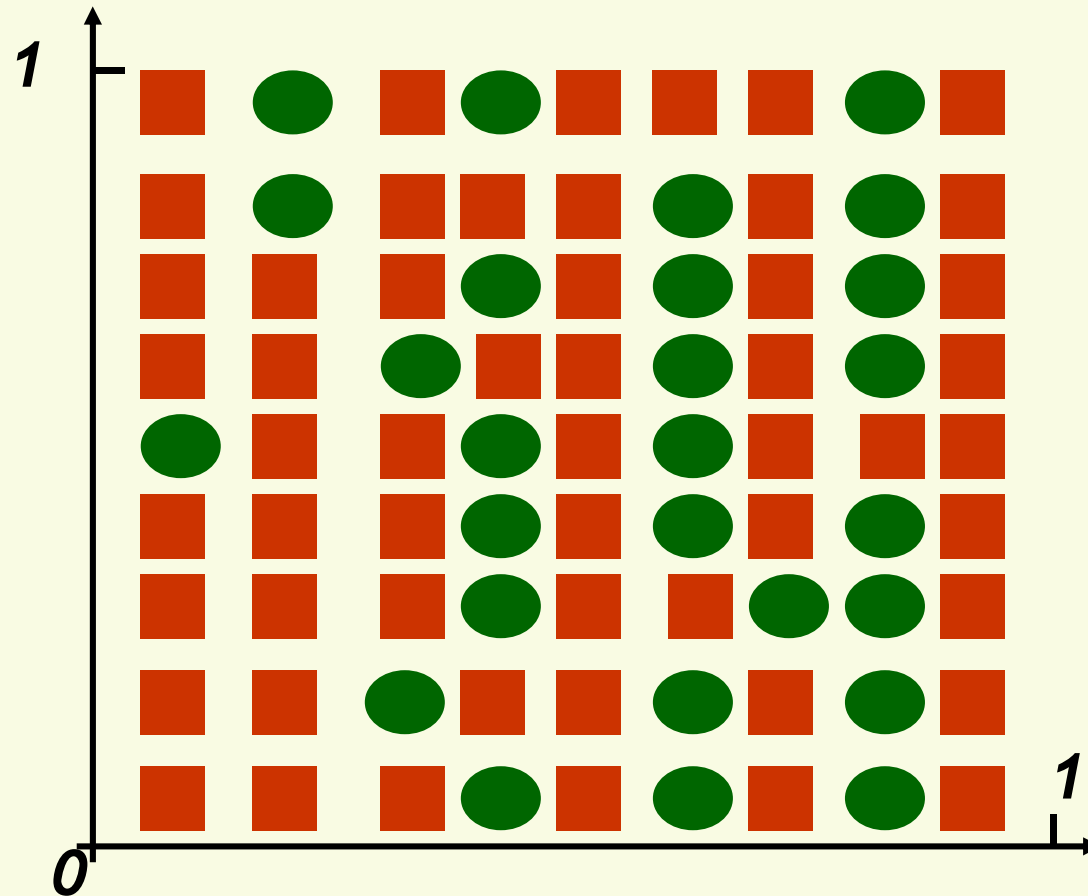
- Suppose we want to use the nearest neighbor approach with $k = 1$ (**1NN**)
- Suppose we start with only one feature



- This feature is not discriminative, i.e. it does not separate the classes well
- We decide to use 2 features. For the 1NN method to work well, need a lot of samples, i.e. samples have to be dense
- To maintain the same density as in 1D (9 samples per unit length), how many samples do we need?

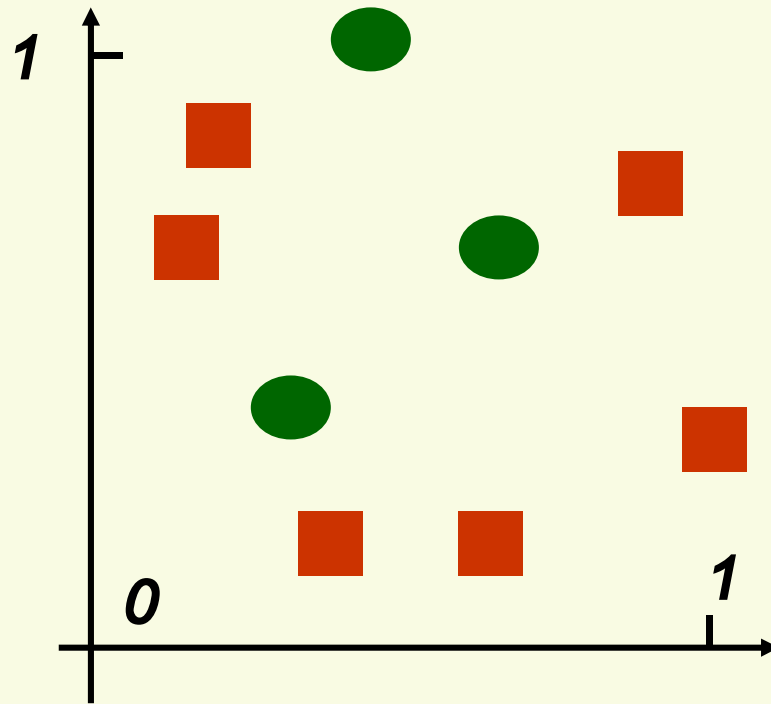
Curse of Dimensionality: Number of Samples

- We need 9^2 samples to maintain the same density as in $1D$



Curse of Dimensionality: Number of Samples

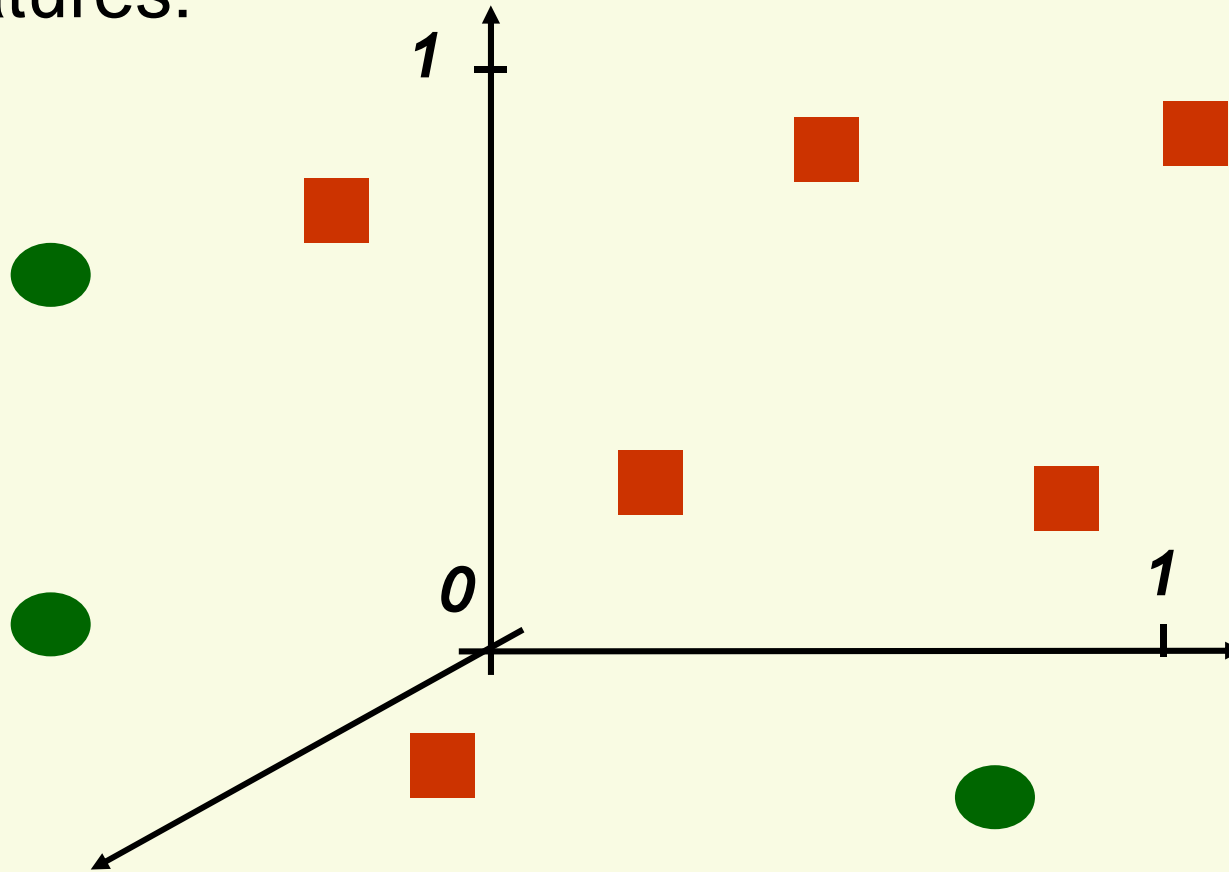
- Of course, when we go from 1 feature to 2, no one gives us more samples, we still have 9



- This is way too sparse for **1NN** to work well

Curse of Dimensionality: Number of Samples

- Things go from bad to worse if we decide to use 3 features:



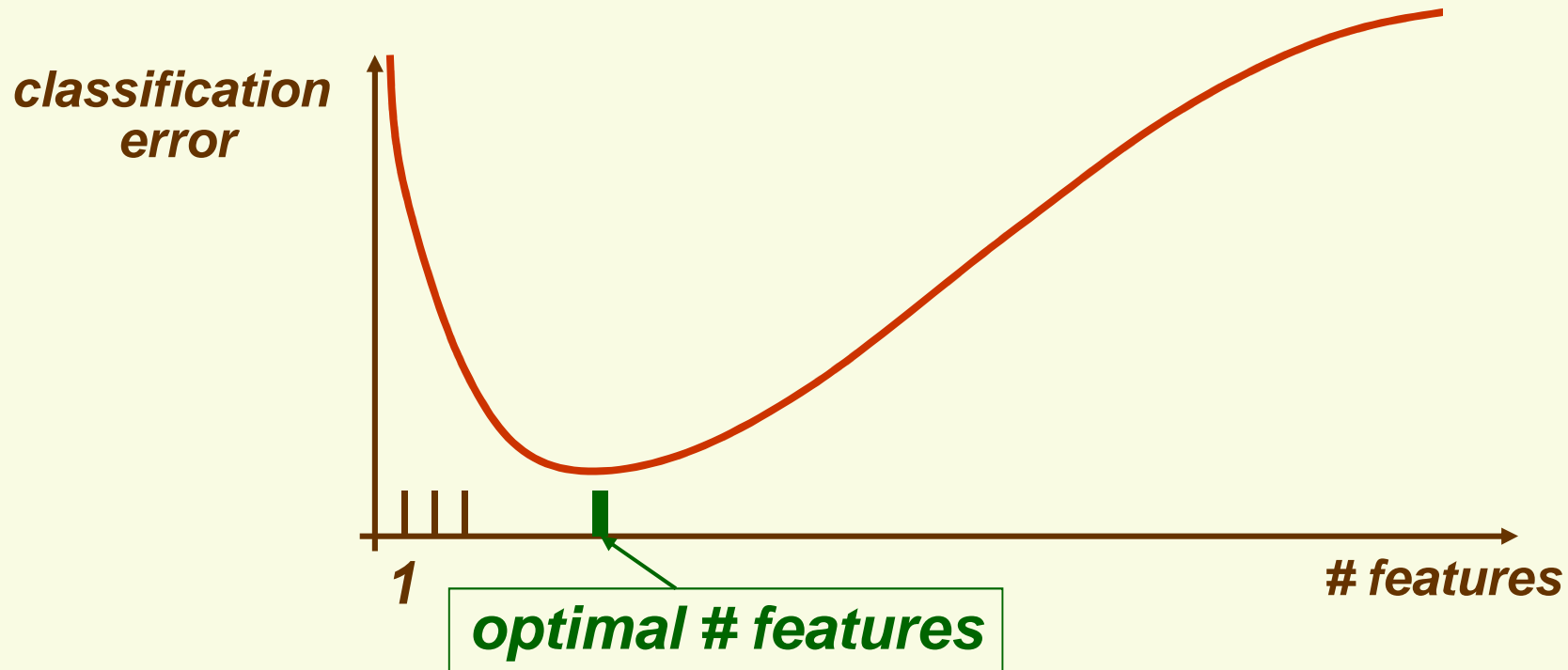
- If **9** was dense enough in 1D, in 3D we need **$9^3=729$** samples!

Curse of Dimensionality: Number of Samples

- In general, if n samples is dense enough in **1D**
- Then in d dimensions we need n^d samples!
- And n^d grows really really fast as a function of d
- Common pitfall:
 - If we can't solve a problem with a few features, adding more features seems like a good idea
 - However the number of samples usually stays the same
 - The method with more features is likely to perform worse instead of expected better

Curse of Dimensionality: Number of Samples

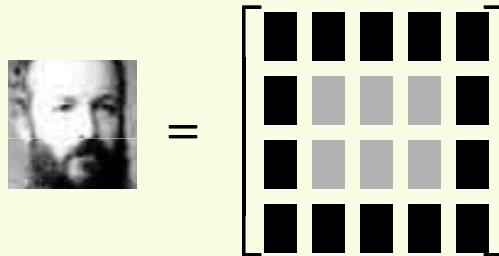
- For a fixed number of samples, as we add features, the graph of classification error:



- Thus for each fixed sample size n , there is the optimal number of features to use

The Curse of Dimensionality

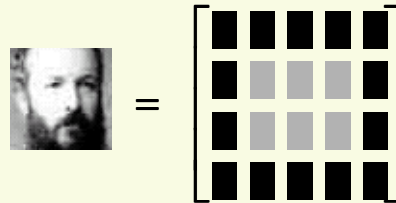
- We should try to avoid creating lot of features
- Often no choice, problem starts with many features
- Example: Face Detection
 - One sample point is k by m array of pixels



- Feature extraction is not trivial, usually every pixel is taken as a feature
- Typical dimension is 20 by 20 = 400
- Suppose **10** samples are dense enough for 1 dimension. Need only 10^{400} samples

The Curse of Dimensionality

- Face Detection, dimension of one sample point is km



- The fact that we set up the problem with km dimensions (features) does not mean it is really a km -dimensional problem
- Space of all k by m images has km dimensions
- Space of all k by m faces must be much smaller, since faces form a tiny fraction of all possible images
- Most likely we are not setting the problem up with the right features
- If we used better features, we are likely need much less than km -dimensions

Dimensionality Reduction

- High dimensionality is challenging and redundant
- It is natural to try to reduce dimensionality
- Reduce dimensionality by feature combination: combine old features \mathbf{x} to create new features \mathbf{y}

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_d \end{bmatrix} \rightarrow f\left(\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_d \end{bmatrix}\right) = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_k \end{bmatrix} = \mathbf{y} \quad \text{with } k < d$$

- For example,
$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{x}_1 + \mathbf{x}_2 \\ \mathbf{x}_3 + \mathbf{x}_4 \end{bmatrix} = \mathbf{y}$$

- Ideally, the new vector \mathbf{y} should retain from \mathbf{x} all information important for classification

Dimensionality Reduction

- The best $f(\mathbf{x})$ is most likely a non-linear function
- Linear functions are easier to find though
- For now, assume that $f(\mathbf{x})$ is a linear mapping
- Thus it can be represented by a matrix \mathbf{W} :

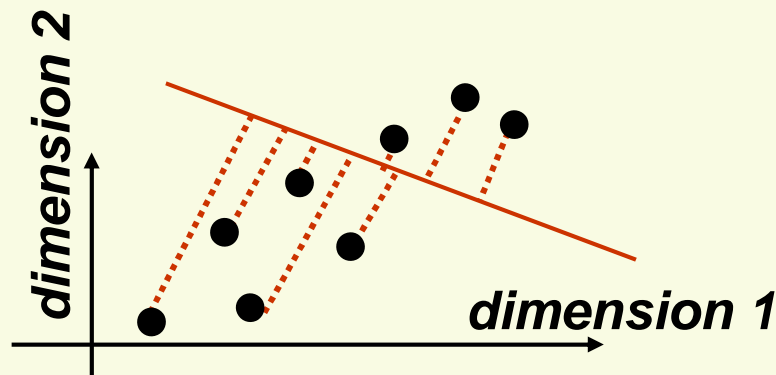
$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_d \end{bmatrix} \Rightarrow \mathbf{W} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_d \end{bmatrix} = \begin{bmatrix} \mathbf{w}_{11} & \cdots & \mathbf{w}_{1d} \\ \vdots & & \vdots \\ \mathbf{w}_{k1} & \cdots & \mathbf{w}_{kd} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_d \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_k \end{bmatrix} \quad \text{with } k < d$$

Feature Combination

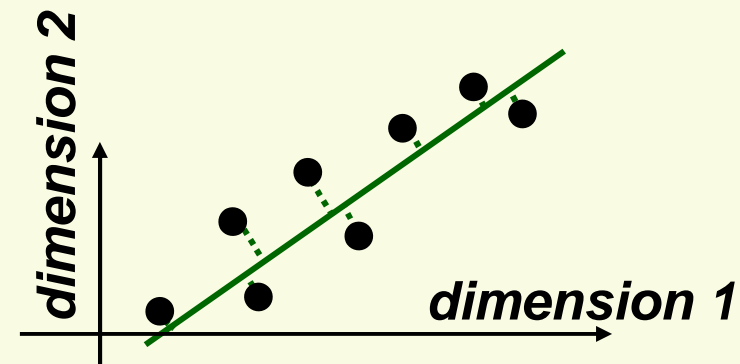
- We will look at 2 methods for feature combination
 - Principle Component Analysis (PCA)
 - Fischer Linear Discriminant (next lecture)

Principle Component Analysis (PCA)

- **Main idea:** seek most accurate data representation in a lower dimensional space
- Example in 2-D
 - Project data to 1-D subspace (a line) which minimize the projection error



*large projection errors,
bad line to project to*

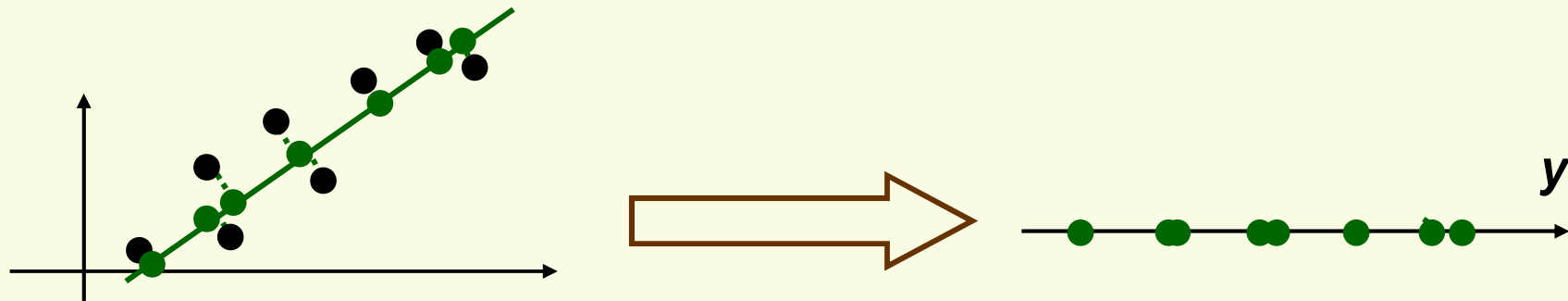


*small projection errors,
good line to project to*

- Notice that the good line to use for projection lies in the direction of largest variance

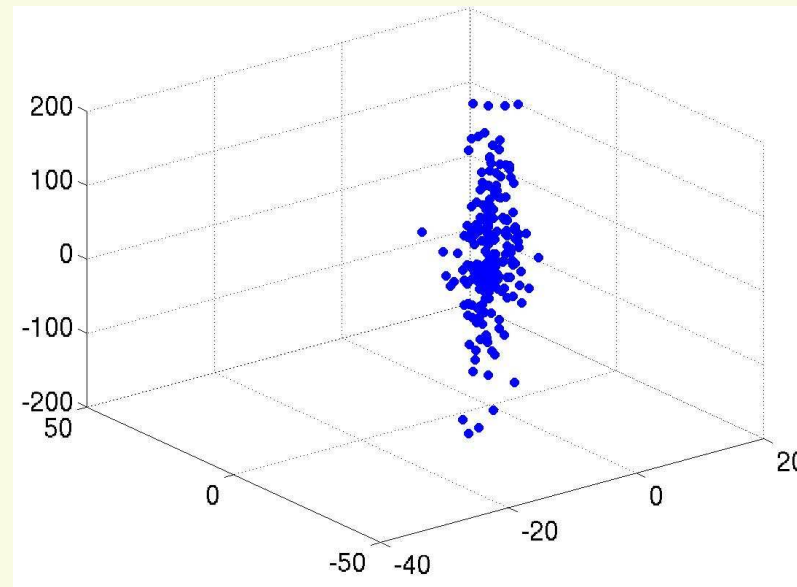
PCA

- After the data is projected on the best line, need to transform the coordinate system to get 1D representation for vector \mathbf{y}

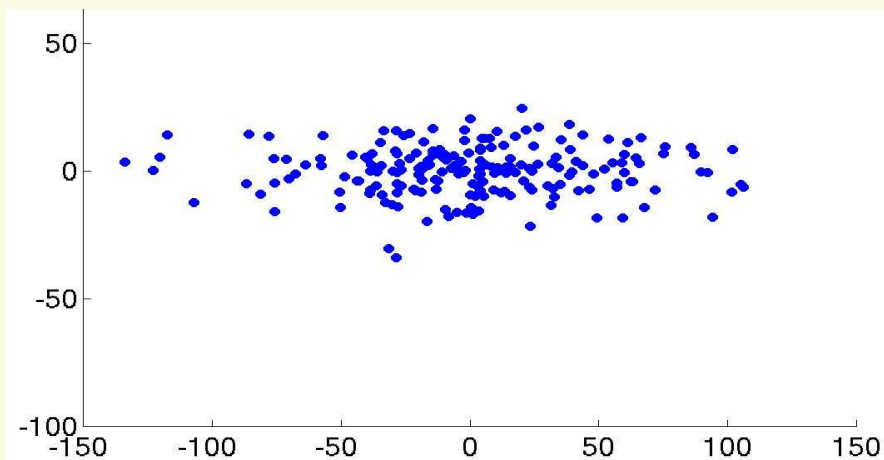


- Note that new data \mathbf{y} has the same variance as old data \mathbf{x} in the direction of the green line
- PCA preserves largest variances in the data. We will prove this statement, for now it is just an intuition of what PCA will do

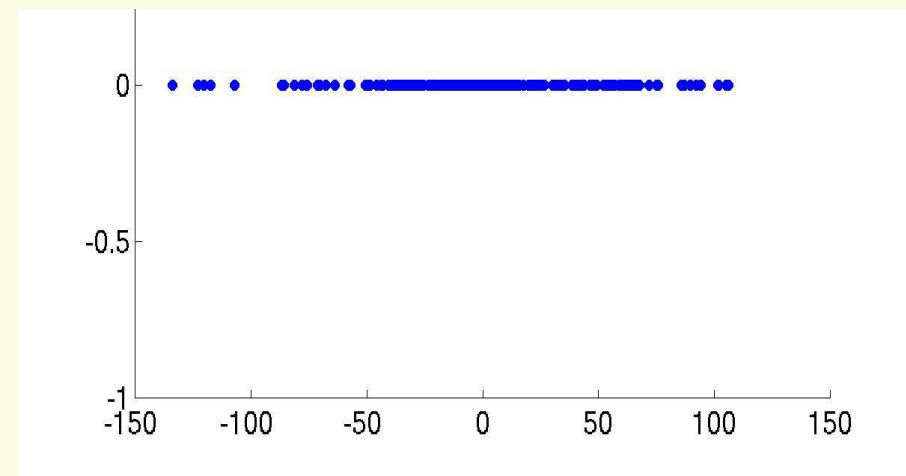
PCA: Approximation of Elliptical Cloud in 3D



best 2D approximation



best 1D approximation



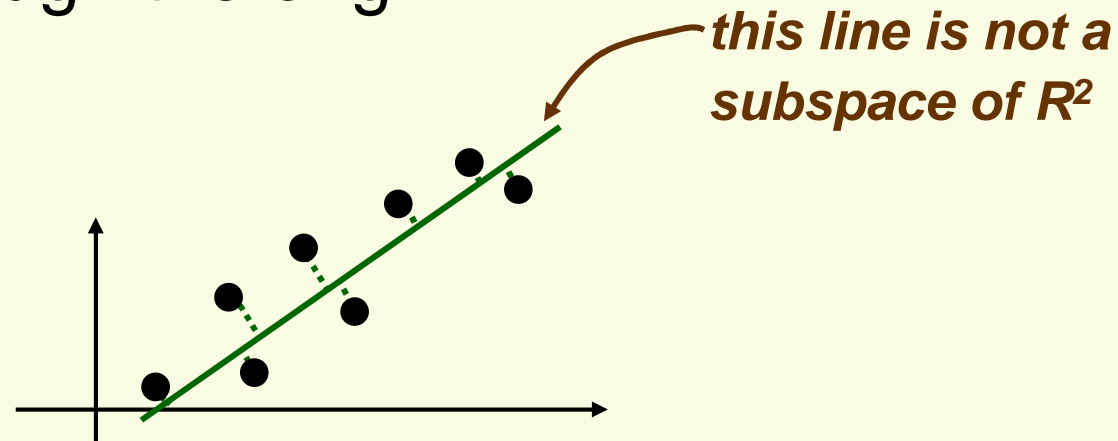
PCA: Linear Algebra for Derivation

- Let V be a d dimensional linear space, and W be a k dimensional linear subspace of V
- We can always find a set of k dimensional vectors $\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_k\}$ which forms an orthonormal basis for W
 - $\langle \mathbf{e}_i, \mathbf{e}_j \rangle = 0$ if i is not equal to j and $\langle \mathbf{e}_i, \mathbf{e}_i \rangle = 1$
- Thus any vector in W can be written as

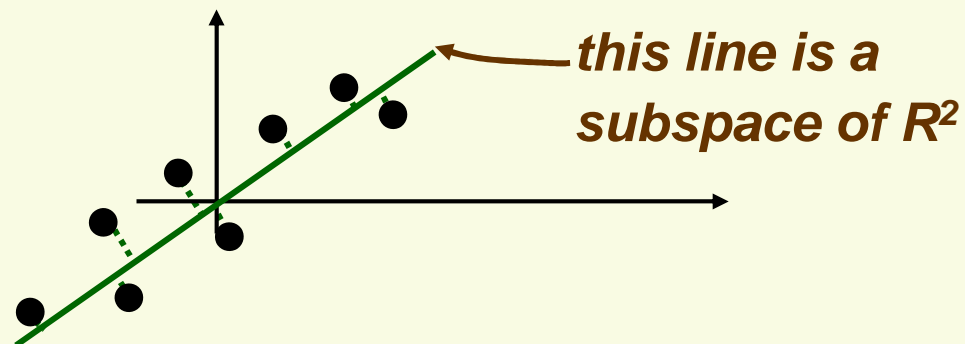
$$\alpha_1 \mathbf{e}_1 + \alpha_2 \mathbf{e}_2 + \dots + \alpha_k \mathbf{e}_k = \sum_{i=1}^k \alpha_i \mathbf{e}_i \quad \text{for scalars } \alpha_1, \dots, \alpha_k$$

PCA: Linear Algebra for Derivation

- Recall that subspace W contains the zero vector, i.e. it goes through the origin



- For derivation, it will be convenient to project to subspace W : thus we need to shift everything

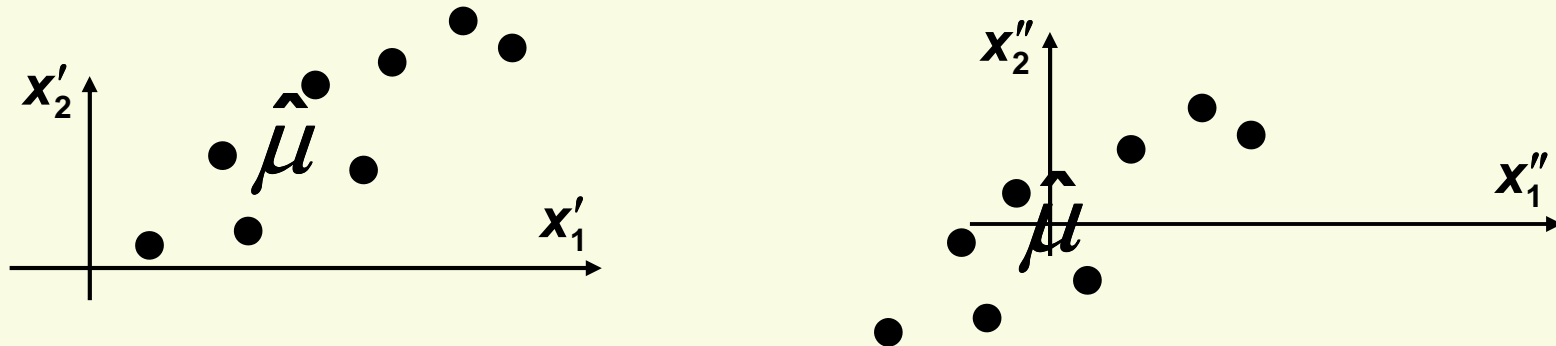


PCA Derivation: Shift by the Mean Vector

- Before PCA, subtract sample mean from the data

$$\mathbf{x} - \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i = \mathbf{x} - \hat{\boldsymbol{\mu}}$$

- The new data has zero mean.
- All we did is change the coordinate system



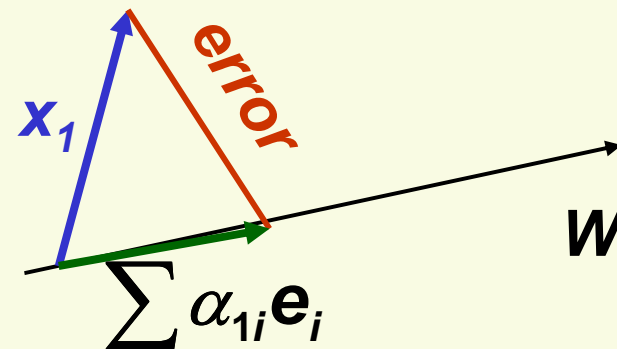
PCA: Derivation

- We want to find the most accurate representation of data $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ in some subspace W which has dimension $k < d$
- Let $\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_k\}$ be the orthonormal basis for W . Any vector in W can be written as $\sum_{i=1}^k \alpha_i \mathbf{e}_i$
- Thus \mathbf{x}_1 will be represented by some vector in W

$$\sum_{i=1}^k \alpha_{1i} \mathbf{e}_i$$

- Error of this representation:

$$\mathbf{error} = \left\| \mathbf{x}_1 - \sum_{i=1}^k \alpha_{1i} \mathbf{e}_i \right\|^2$$



PCA: Derivation

- To find the total error, we need to sum over all \mathbf{x}_j 's
- Any \mathbf{x}_j can be written as $\sum_{i=1}^k \alpha_{ji} \mathbf{e}_i$
- Thus the total error for representation of all data \mathbf{D} is:

sum over all data points

$$\underbrace{J(\mathbf{e}_1, \dots, \mathbf{e}_k, \alpha_{11}, \dots, \alpha_{nk})}_{\text{unknowns}} = \sum_{j=1}^n \left\| \mathbf{x}_j - \sum_{i=1}^k \alpha_{ji} \mathbf{e}_i \right\|^2$$

error at one point

PCA: Derivation

- To minimize J , need to take partial derivatives and also enforce constraint that $\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_k\}$ are orthogonal

$$J(\mathbf{e}_1, \dots, \mathbf{e}_k, \alpha_{11}, \dots, \alpha_{nk}) = \sum_{j=1}^n \left\| \mathbf{x}_j - \sum_{i=1}^k \alpha_{ji} \mathbf{e}_i \right\|^2$$

- Let us simplify J first:

$$J(\mathbf{e}_1, \dots, \mathbf{e}_k, \alpha_{11}, \dots, \alpha_{nk}) = \sum_{j=1}^n \|\mathbf{x}_j\|^2 - 2 \sum_{j=1}^n \sum_{i=1}^k \alpha_{ji} \mathbf{x}_j^t \mathbf{e}_i + \sum_{j=1}^n \sum_{i=1}^k \alpha_{ji}^2$$

PCA: Derivation

$$\mathbf{J}(\mathbf{e}_1, \dots, \mathbf{e}_k, \alpha_{11}, \dots, \alpha_{nk}) = \sum_{j=1}^n \|\mathbf{x}_j\|^2 - 2 \sum_{j=1}^n \sum_{i=1}^k \alpha_{ji} \mathbf{x}_j^t \mathbf{e}_i + \sum_{j=1}^n \sum_{i=1}^k \alpha_{ji}^2$$

- First take partial derivatives with respect to α_{ml}

$$\frac{\partial}{\partial \alpha_{ml}} \mathbf{J}(\mathbf{e}_1, \dots, \mathbf{e}_k, \alpha_{11}, \dots, \alpha_{nk}) = -2 \mathbf{x}_m^t \mathbf{e}_l + 2 \alpha_{ml}$$

- Thus the optimal value for α_{ml} is

$$-2 \mathbf{x}_m^t \mathbf{e}_l + 2 \alpha_{ml} = 0 \Rightarrow \alpha_{ml} = \mathbf{x}_m^t \mathbf{e}_l$$

PCA: Derivation

$$\mathbf{J}(\mathbf{e}_1, \dots, \mathbf{e}_k, \alpha_{11}, \dots, \alpha_{nk}) = \sum_{j=1}^n \|\mathbf{x}_j\|^2 - 2 \sum_{j=1}^n \sum_{i=1}^k \alpha_{ji} \mathbf{x}_j^t \mathbf{e}_i + \sum_{j=1}^n \sum_{i=1}^k \alpha_{ji}^2$$

- Plug the optimal value for $\alpha_{mi} = \mathbf{x}_m^t \mathbf{e}_i$ back into \mathbf{J}

$$\mathbf{J}(\mathbf{e}_1, \dots, \mathbf{e}_k) = \sum_{j=1}^n \|\mathbf{x}_j\|^2 - 2 \sum_{j=1}^n \sum_{i=1}^k (\mathbf{x}_j^t \mathbf{e}_i) \mathbf{x}_j^t \mathbf{e}_i + \sum_{j=1}^n \sum_{i=1}^k (\mathbf{x}_j^t \mathbf{e}_i)^2$$

- Can simplify \mathbf{J}

$$\mathbf{J}(\mathbf{e}_1, \dots, \mathbf{e}_k) = \sum_{j=1}^n \|\mathbf{x}_j\|^2 - \sum_{j=1}^n \sum_{i=1}^k (\mathbf{x}_j^t \mathbf{e}_i)^2$$

PCA: Derivation

$$J(\mathbf{e}_1, \dots, \mathbf{e}_k) = \sum_{j=1}^n \|\mathbf{x}_j\|^2 - \sum_{j=1}^n \sum_{i=1}^k (\mathbf{x}_j^t \mathbf{e}_i)^2$$

- Rewrite J using $(\mathbf{a}^t \mathbf{b})^2 = (\mathbf{a}^t \mathbf{b})(\mathbf{a}^t \mathbf{b}) = (\mathbf{b}^t \mathbf{a})(\mathbf{a}^t \mathbf{b}) = \mathbf{b}^t (\mathbf{a} \mathbf{a}^t) \mathbf{b}$

$$\begin{aligned} J(\mathbf{e}_1, \dots, \mathbf{e}_k) &= \sum_{j=1}^n \|\mathbf{x}_j\|^2 - \sum_{i=1}^k \mathbf{e}_i^t \left(\sum_{j=1}^n (\mathbf{x}_j \mathbf{x}_j^t) \right) \mathbf{e}_i \\ &= \sum_{j=1}^n \|\mathbf{x}_j\|^2 - \sum_{i=1}^k \mathbf{e}_i^t \mathbf{S} \mathbf{e}_i \end{aligned}$$

- Where $\mathbf{S} = \sum_{j=1}^n \mathbf{x}_j \mathbf{x}_j^t$
- \mathbf{S} is called the scatter matrix, it is just $n-1$ times the sample covariance matrix we have seen before

$$\hat{\Sigma} = \frac{1}{n-1} \sum_{j=1}^n (\mathbf{x}_j - \hat{\mu})(\mathbf{x}_j - \hat{\mu})^t$$

PCA: Derivation

$$J(\mathbf{e}_1, \dots, \mathbf{e}_k) = \underbrace{\sum_{j=1}^n \|\mathbf{x}_j\|^2}_{\text{constant}} - \sum_{i=1}^k \mathbf{e}_i^t \mathbf{S} \mathbf{e}_i$$

- Minimizing J is equivalent to maximizing $\sum_{i=1}^k \mathbf{e}_i^t \mathbf{S} \mathbf{e}_i$
- We should also enforce constraints $\mathbf{e}_i^t \mathbf{e}_i = 1$ for all i
- Use the method of Lagrange multipliers, incorporate the constraints with undetermined $\lambda_1, \dots, \lambda_k$
- Need to maximize new function u

$$u(\mathbf{e}_1, \dots, \mathbf{e}_k) = \sum_{i=1}^k \mathbf{e}_i^t \mathbf{S} \mathbf{e}_i - \sum_{j=1}^k \lambda_j (\mathbf{e}_j^t \mathbf{e}_j - 1)$$

PCA: Derivation

$$u(\mathbf{e}_1, \dots, \mathbf{e}_k) = \sum_{i=1}^k \mathbf{e}_i^t \mathbf{S} \mathbf{e}_i - \sum_{j=1}^k \lambda_j (\mathbf{e}_j^t \mathbf{e}_j - 1)$$

- Compute the partial derivatives with respect to \mathbf{e}_m

$$\frac{\partial}{\partial \mathbf{e}_m} u(\mathbf{e}_1, \dots, \mathbf{e}_k) = 2\mathbf{S}\mathbf{e}_m - 2\lambda_m \mathbf{e}_m = \mathbf{0}$$

Note: \mathbf{e}_m is a vector, what we are really doing here is taking partial derivatives with respect to each element of \mathbf{e}_m and then arranging them up in a linear equation

- Thus λ_m and \mathbf{e}_m are eigenvalues and eigenvectors of scatter matrix \mathbf{S}

$$\mathbf{S}\mathbf{e}_m = \lambda_m \mathbf{e}_m$$

PCA: Derivation

$$J(\mathbf{e}_1, \dots, \mathbf{e}_k) = \sum_{j=1}^n \|\mathbf{x}_j\|^2 - \sum_{i=1}^k \mathbf{e}_i^t \mathbf{S} \mathbf{e}_i$$

- Let's plug \mathbf{e}_m back into J and use $\mathbf{S} \mathbf{e}_m = \lambda_m \mathbf{e}_m$

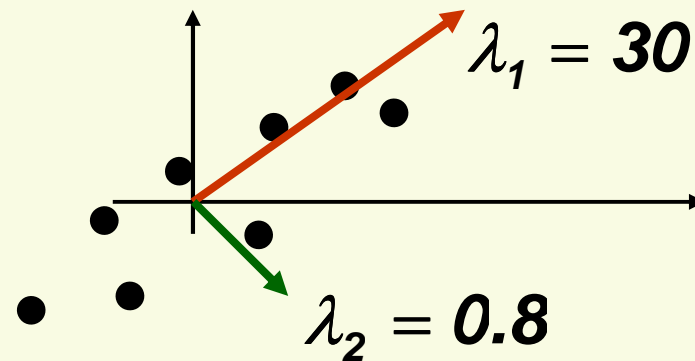
$$J(\mathbf{e}_1, \dots, \mathbf{e}_k) = \sum_{j=1}^n \|\mathbf{x}_j\|^2 - \sum_{i=1}^k \lambda_i \|\mathbf{e}_i\|^2 = \sum_{j=1}^n \|\mathbf{x}_j\|^2 - \sum_{i=1}^k \lambda_i$$

constant

- Thus to minimize J take for the basis of W the k eigenvectors of \mathbf{S} corresponding to the k largest eigenvalues

PCA

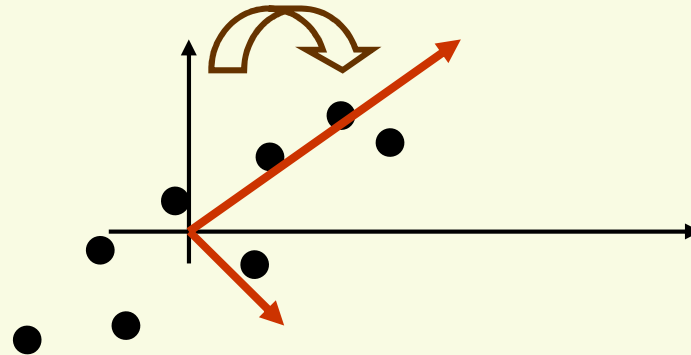
- The larger the eigenvalue of \mathbf{S} , the larger is the variance in the direction of corresponding eigenvector



- This result is exactly what we expected: project \mathbf{x} into subspace of dimension \mathbf{k} which has the largest variance
- This is very intuitive: restrict attention to directions where the scatter is the greatest

PCA

- Thus PCA can be thought of as finding new orthogonal basis by rotating the old axis until the directions of maximum variance are found



PCA as Data Approximation

- Let $\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_d\}$ be all d eigenvectors of the scatter matrix \mathbf{S} , sorted in order of decreasing corresponding eigenvalue

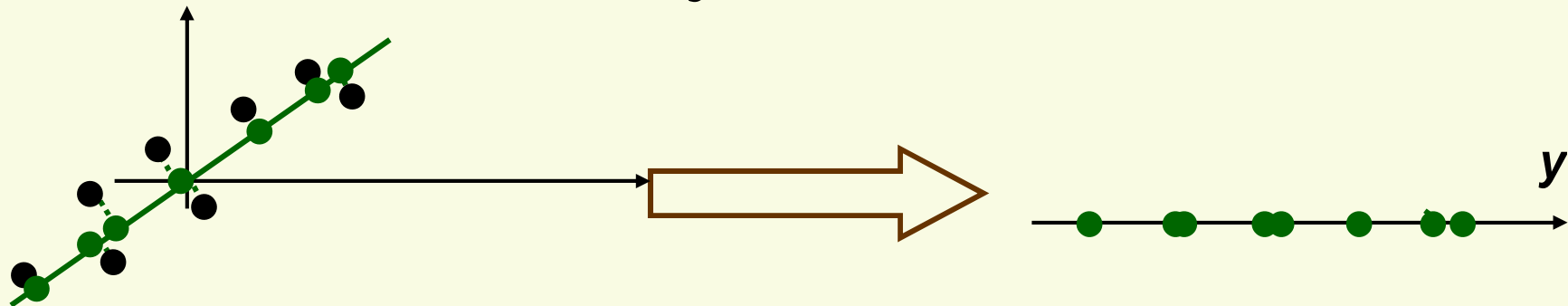
- Without any approximation, for any sample \mathbf{x}_i :
error of approximation

$$\mathbf{x}_i = \sum_{j=1}^d \alpha_j \mathbf{e}_j = \underbrace{\alpha_1 \mathbf{e}_1 + \dots + \alpha_k \mathbf{e}_k}_{\text{approximation of } \mathbf{x}_i} + \underbrace{\alpha_{k+1} \mathbf{e}_{k+1} + \dots + \alpha_d \mathbf{e}_d}_{\text{error of approximation}}$$

- coefficients $\alpha_m = \mathbf{x}_i^t \mathbf{e}_m$ are called *principle components*
 - The larger k , the better is the approximation
 - Components are arranged in order of importance, more important components come first
- Thus PCA takes the first k most important components of \mathbf{x}_i as an approximation to \mathbf{x}_i

PCA: Last Step

- Now we know how to project the data
- Last step is to change the coordinates to get final k -dimensional vector \mathbf{y}



- Let matrix $\mathbf{E} = [\mathbf{e}_1 \cdots \mathbf{e}_k]$
- Then the coordinate transformation is $\mathbf{y} = \mathbf{E}^t \mathbf{x}$

- Under \mathbf{E}^t , the eigenvectors become the standard basis:

$$\mathbf{E}^t \mathbf{e}_i = \begin{bmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_i \\ \vdots \\ \mathbf{e}_k \end{bmatrix} \quad \mathbf{e}_i = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

Recipe for Dimension Reduction with PCA

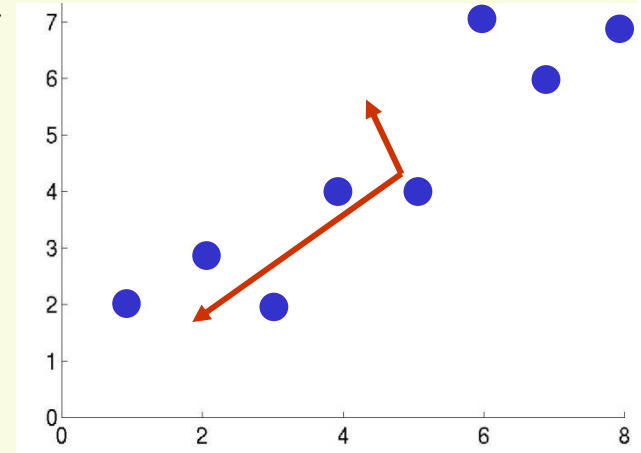
Data $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$. Each \mathbf{x}_i is a d -dimensional vector. Wish to use PCA to reduce dimension to k

1. Find the sample mean $\hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$
2. Subtract sample mean from the data $\mathbf{z}_i = \mathbf{x}_i - \hat{\boldsymbol{\mu}}$
3. Compute the scatter matrix $\mathbf{S} = \sum_{i=1}^n \mathbf{z}_i \mathbf{z}_i^t$
4. Compute eigenvectors $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_k$ corresponding to the k largest eigenvalues of \mathbf{S}
5. Let $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_k$ be the columns of matrix $\mathbf{E} = [\mathbf{e}_1 \cdots \mathbf{e}_k]$
6. The desired \mathbf{y} which is the closest approximation to \mathbf{x} is $\mathbf{y} = \mathbf{E}^t \mathbf{z}$

PCA Example Using Matlab

- Let $D = \{(1,2), (2,3), (3,2), (4,4), (5,4), (6,7), (7,6), (9,7)\}$
- Convenient to arrange data in array

$$X = \begin{bmatrix} 1 & 2 \\ \vdots & \vdots \\ 9 & 7 \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_8 \end{bmatrix}$$



- Mean $\mu = \mathit{mean}(X) = [4.6 \ 4.4]$
- Subtract mean from data to get new data array Z

$$Z = X - \begin{bmatrix} \mu \\ \vdots \\ \mu \end{bmatrix} = X - \mathit{repmat}(\mu, 8, 1) = \begin{bmatrix} -3.6 & -4.4 \\ \vdots & \vdots \\ 4.4 & 2.6 \end{bmatrix}$$

- Compute the scatter matrix S

$$S = 7 * \mathit{cov}(Z) = [-3.6 \ -4.4] \begin{bmatrix} -3.6 \\ -4.4 \end{bmatrix} + \dots + [4.4 \ 2.6] \begin{bmatrix} 4.4 \\ 2.6 \end{bmatrix} = \begin{bmatrix} 57 & 40 \\ 40 & 34 \end{bmatrix}$$

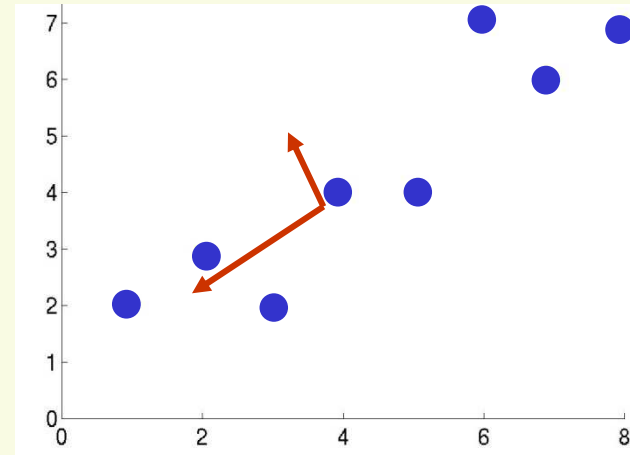
matlab uses unbiased estimate for covariance, so $S=(n-1)\mathit{cov}(Z)$*

PCA Example Using Matlab

- Use $[V,D] = \text{eig}(\mathbf{S})$ to get eigenvalues and eigenvectors of \mathbf{S}

$$\lambda_1 = 87 \text{ and } \mathbf{e}_1 = \begin{bmatrix} -0.8 \\ -0.6 \end{bmatrix}$$

$$\lambda_2 = 3.8 \text{ and } \mathbf{e}_2 = \begin{bmatrix} 0.6 \\ -0.8 \end{bmatrix}$$



- Projection to 1D space in the direction of \mathbf{e}_1

$$\begin{aligned} \mathbf{Y} = \mathbf{e}_1^t \mathbf{Z}^t &= \left(\begin{bmatrix} -0.8 & -0.6 \end{bmatrix} \begin{bmatrix} -3.6 & \cdots & 4.4 \\ -4.4 & \cdots & 2.6 \end{bmatrix} \right) = \begin{bmatrix} 4.3 & \cdots & -5.1 \end{bmatrix} \\ &= \begin{bmatrix} y_1 & \cdots & y_8 \end{bmatrix} \end{aligned}$$