

• project on  $t$  line  $\vec{e}$   $d \rightarrow t$   
 line  $= |\vec{e}|$   $\|\vec{e}\|=1$

•  $x_k$   $\xrightarrow{\text{projected}}$   $\vec{e}^T (x_k - \mu)$  new rep. on line  $\vec{e}$   
 line  $\vec{e}$

•  $J(\vec{e}) = \text{SSE min} \rightarrow \text{max } \vec{e}^T \Sigma \vec{e}$   
 max var

• solve  $\text{max } \vec{e}^T \Sigma \vec{e}$  }  $\xrightarrow{\text{Lagrange}}$   $\Sigma \vec{e} = \lambda \vec{e}$   
 sub;  $\|\vec{e}\|=1$  }  $\Rightarrow \vec{e} = \text{eigen vector}$   
 $(\Sigma)$

•  $\Sigma = (x - \mu)(x - \mu)^T$   $\left. \begin{array}{l} \text{sym} \\ \text{pos def} \end{array} \right\} \Rightarrow \text{spectral decomp}$   
 $[\vec{e}] [\Lambda] [\vec{e}]$

Data points (vectors)

	$f_1$	$f_2$	$f_d$
$x_1$			
$x_2$			
$x_3$			
$\vdots$			
$x_N$			

new representation on  $t$  features

A:  $t \ll d$

$N \times d$

$N \times t$

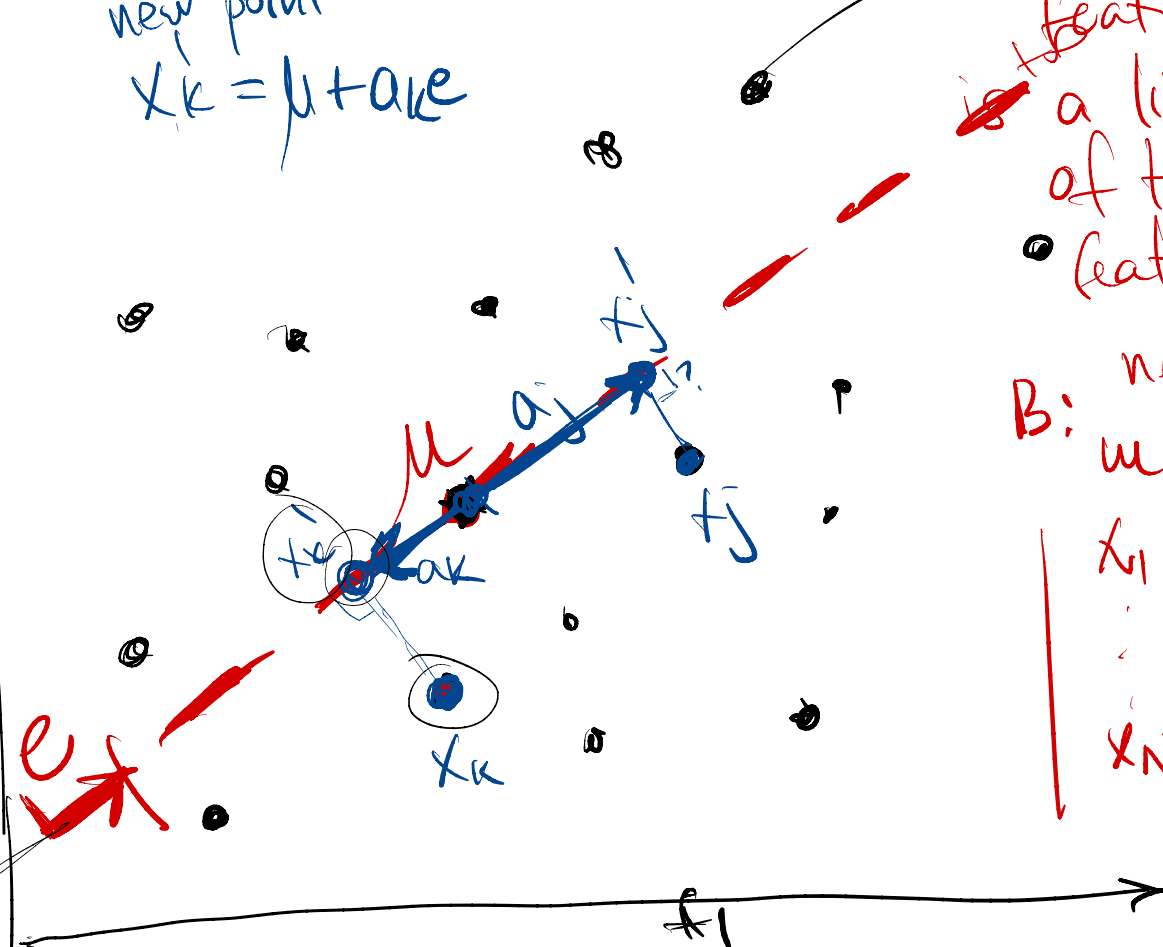
each new feature  $g_i$  is a linear comb. of the original features.

B: new data matrix

	$t$ feat
$x_1$	
$\vdots$	
$x_N$	

$d=2$   
visualization

new point  $x_k = \mu + a_k e$



(1)  $t=0$  NO FEAT  
represent whole  $X$  on 1 point

$\mu = \text{mean}(x)$   
 $= E[x]$

$|e|=1$  direction of line

$N \times t$   
orig

$t=1 \Rightarrow 1$  <sup>new</sup> dim  $\Rightarrow$  new representation will be on 1 line

guess: that "one line" new rep

- passes through  $\mu$ .

- usually correspond to longest-direction stretch of data.

geom line:  $e, a_1, a_2, \dots, a_N$   
new coordinates

SSE / sq loss

error  $J(a_1, a_2, \dots, a_N, e) = \sum_{k=1}^N \left\| \mu + a_k e - x_k \right\|^2 = \sum_k \left\| a_k e - (x_k - \mu) \right\|^2$

new point

orig point

$x_k = \mu + a_k e$

$x_k - \mu = a_k e$

$(x_k - \mu) e = a_k e^T e$

$$= \sum_k a_k^2 \|e\|^2 - 2 \sum_k a_k e^T (x_k - \mu) + \sum_k \|x_k - \mu\|^2$$

$J = \min \Rightarrow \frac{\partial J}{\partial a_k} = 0 \Leftrightarrow a_k - e^T (x_k - \mu) = 0 \Rightarrow a_k = e^T (x_k - \mu)$

new data mean  $(a_1, a_2, \dots, a_N) = E[\mu + a_k e] = \mu + E[a_k e] = \mu + e^T E[x_k - \mu]$

$E[x_k - \mu] = 0$

projection

$= \mu$

$J(e)$  implicit projections  $x \rightarrow e$   
 $= a$

$$= \sum_k a_k^2 - 2 \sum_k a_k e^T (x_k - \mu) + \sum_k \|x_k - \mu\|^2$$

$$= - \sum_k (e^T (x_k - \mu))^2 + \sum_k \|x_k - \mu\|^2$$

$$= - \sum_k e^T (x_k - \mu) (x_k - \mu)^T e + \sum_k \|x_k - \mu\|^2$$

covar matrix  $\Sigma$

$$= -e^T \left[ \sum_k (x_k - \mu)(x_k - \mu)^T \right] e$$

$$= -e^T \Sigma e$$

sigma

minimise  $J(e)$

$$\min -e^T \Sigma e \quad \max e^T \Sigma e$$

max variance (projections)

Most important: what is the best e? want MAX

~~Var~~ Var [projections = new representations]

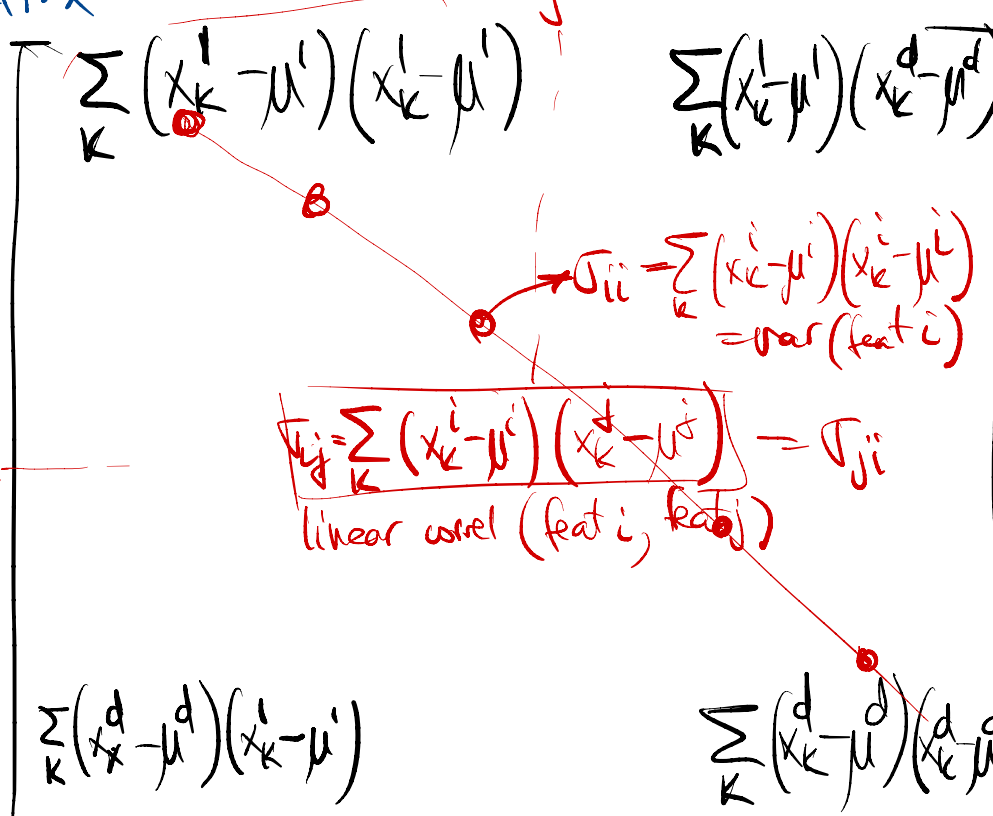
$$E[(\underbrace{\mu + a_{\text{ave}}}_{\text{proj}} - E[\mu + a_{\text{ave}}])^2] = E[(\mu + a_{\text{ave}} - \mu)^2] = E[(a_{\text{ave}})^2]$$

$$= E[e^T \underbrace{(x_k - \mu)}_{\text{sigma covar matrix}} \cdot e^T (x_k - \mu)] = E[e^T (x_k - \mu) (x_k - \mu)^T e]$$

$$= e^T \sum_k e$$

$(1 \times d) (d \times d) (d \times 1)$  MAX

$$\underbrace{\sum}_{\text{sigma}} = \underbrace{\sum}_{\text{sum of matrices}} \underbrace{(x_k - \mu)}_{d \times 1} \underbrace{(x_k - \mu)^T}_{1 \times d}$$



Constrained optimization pb

$\Sigma =$  cov matrix = fixed

maximize  $e^T \Sigma e$  OBS

subject to  $\|e\|=1 \iff e^T e = 1$  constraint

Lag Multiplier = ratio of tangent differentials  
CONSTRAINED

Lagrangian

$L = \max \left[ e^T \Sigma e - \lambda (e^T e - 1) \right]$  OBS

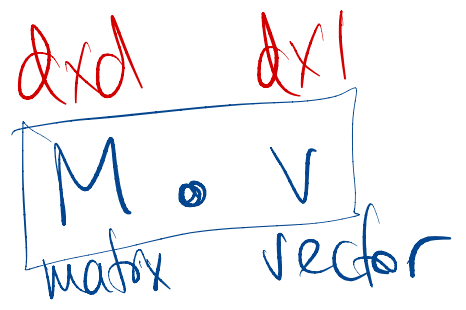
$\frac{\partial L}{\partial e} = 0 \iff 2 \Sigma e = 2 \lambda e = 0$

same direction as e

$\Sigma e = \lambda e$   
sigma scalar

$\|e\|$  does not change direction when  $\|e\|$  multip by  $\Sigma$ !

usually changes direction of the vector.



$\Rightarrow e =$  eigenvector for  $\Sigma$  covar  
 $\lambda =$  eigen val

$M \cdot v = \text{direction of } v \iff v - \text{eigen vector for } M$   
 $M = \text{fixed}$   
 $\lambda v$   
 $\downarrow$   
scalar  
 $\lambda = \text{eigen value}$



$\Sigma = \text{covar}(X)$  (1) symmetric

$\sum_{i,j} = \sum_{j,i}$

Very special

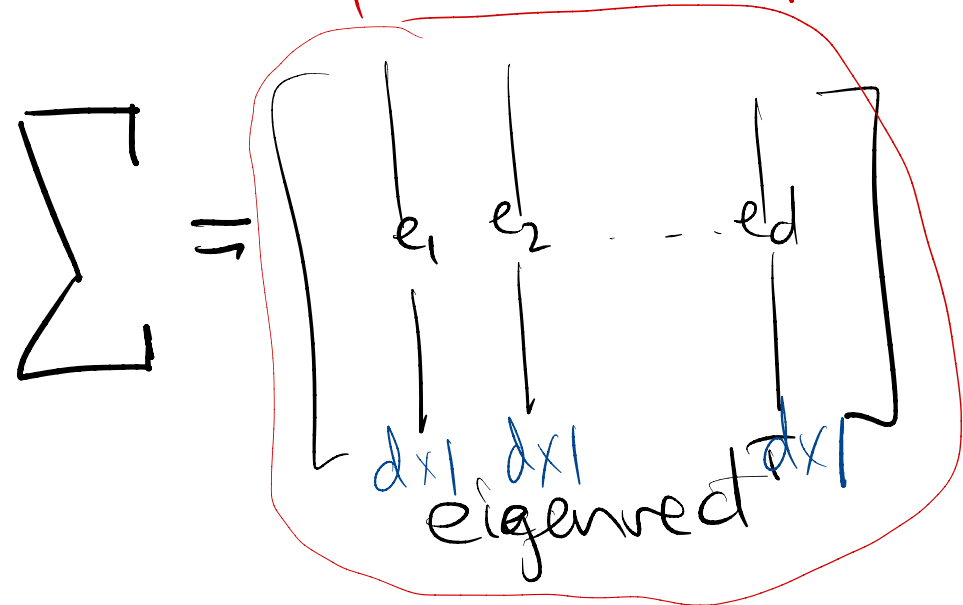
any  $M = A \cdot A^T \Rightarrow$

(2) pos def

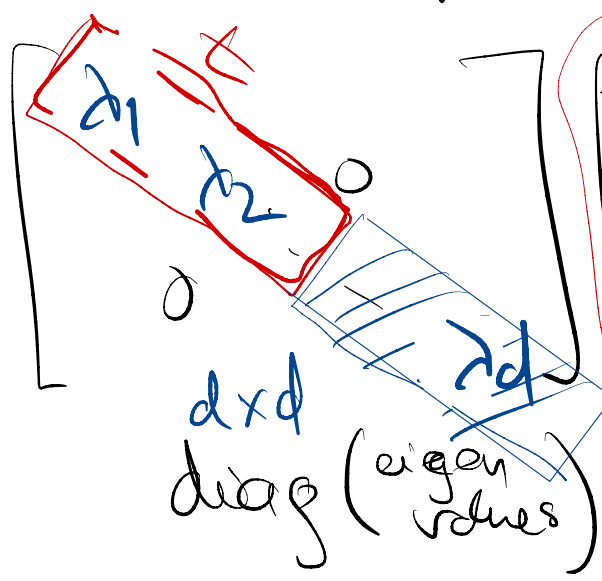
$v^T \Sigma v \geq 0$   
 $v^T (A \cdot A^T) v = (A \cdot v)^T (A \cdot v) \geq 0$

allows spectral decomposition

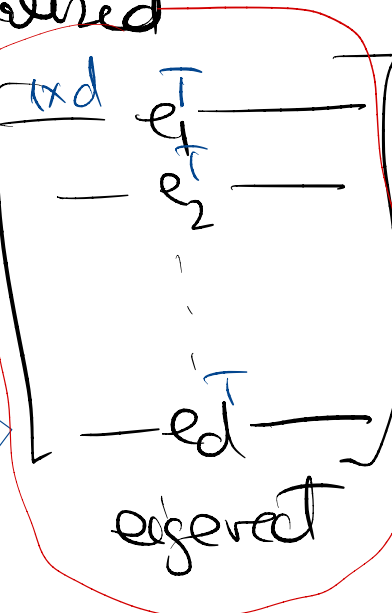
$e_i =$  eigen vectors of  $\Sigma$   
 Normalized



$E$



$d_1 \geq d_2 \geq \dots \geq d_d \geq 0$



$E^T$

WANT  $t$  dim  $\Rightarrow$  take  $(\text{top } t)$  eigen values  
 make the rest 0

Why  $\Sigma = E \cdot D \cdot E^T$  (spectral Theorem)? where  $E$  is the matrix of col-eigenvectors and  $D = \text{diag}$ -matrix of eigenvalues

$$E = \begin{bmatrix} | & | & & | \\ e_1 & e_2 & \dots & e_D \\ | & | & & | \end{bmatrix} \quad D = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \dots & \\ 0 & & & \lambda_D \end{bmatrix} \quad \lambda_1 \geq \lambda_2 \geq \lambda_3 \dots \geq \lambda_D$$

•  $E$  is orthonormal:  $\langle e_i, e_i \rangle = \|e_i\|^2 = 1$  and  $\langle e_i, e_j \rangle = 0$  ( $i \neq j$ )

$$(\lambda_i e_i)^T e_j = (\sum e_i)^T e_j = e_i^T \sum e_j = e_i^T (\sum e_j) = e_i^T (\lambda_j e_j)$$

$$\lambda_i \neq \lambda_j \Rightarrow e_i^T e_j = \langle e_i, e_j \rangle = 0$$

$$\text{Thus } E^T E = I_D = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \dots & \\ & & & 1 \end{bmatrix}_{D \times D} \Rightarrow E^T = E^{-1} \Rightarrow E \cdot E^T = I_D$$

$$\bullet \Sigma = \sum E E^T = \sum \begin{bmatrix} | & | & & | \\ e_1 & e_2 & \dots & e_D \\ | & | & & | \end{bmatrix} \begin{bmatrix} - \\ e_1^T \\ - \\ e_2^T \\ - \\ \dots \\ - \\ e_D^T \\ - \end{bmatrix} = \begin{bmatrix} \sum e_1^T & \sum e_2^T & \dots & \sum e_D^T \end{bmatrix}$$

$$\times \begin{bmatrix} - \\ e_1 \\ - \\ e_2 \\ - \\ \dots \\ - \\ e_D \end{bmatrix} = \begin{bmatrix} | & | & & | \\ \lambda_1 e_1 & \lambda_2 e_2 & \dots & \lambda_D e_D \\ | & | & & | \end{bmatrix} \times \begin{bmatrix} - \\ e_1 \\ - \\ e_2 \\ - \\ \dots \\ - \\ e_D \end{bmatrix} = \lambda_1 e_1 e_1^T + \lambda_2 e_2 e_2^T + \dots + \lambda_D e_D e_D^T$$

$$= \begin{bmatrix} | & | & & | \\ e_1 & e_2 & \dots & e_D \\ | & | & & | \end{bmatrix} \times \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \dots & \\ 0 & & & \lambda_D \end{bmatrix} \times \begin{bmatrix} - \\ e_1 \\ - \\ e_2 \\ - \\ \dots \\ - \\ e_D \end{bmatrix} = E \cdot D \cdot E^T$$

take top eigen values  $\lambda_1, \lambda_2, \dots, \lambda_t$ , ignore the rest  $D^t$ :

$$\sum d \approx \sum_t = \begin{bmatrix} e_1 & e_2 & \dots & e_t \\ | & | & & | \\ \hline & & d \times t & \hline \end{bmatrix}_{d \times t} \times \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_t \end{bmatrix}_{t \times t} \begin{bmatrix} e_1 \\ e_2 \\ \dots \\ e_t \end{bmatrix}_{t \times d}$$

$$d = 765$$

$$t = 20$$

$$\begin{bmatrix} e_1^T \sqrt{\lambda_1} & e_2^T \sqrt{\lambda_2} & \dots & e_t^T \sqrt{\lambda_t} \\ | & | & & | \\ \hline & & d \times t & \hline \end{bmatrix} \cdot \begin{bmatrix} e_1^T \sqrt{\lambda_1} \\ e_2^T \sqrt{\lambda_2} \\ \dots \\ e_t^T \sqrt{\lambda_t} \end{bmatrix}_{t \times d}$$

new representation of data  $(X)$  on  $t$  dim

$$\text{data } \bar{x} \quad 1 \times d$$

$$\rightarrow \bar{x} \cdot C_r \quad 1 \times d \cdot d \times r$$

$$C_r = \begin{bmatrix} e_1 & e_2 & \dots & e_r \\ | & | & & | \\ \hline & & d \times r & \hline \end{bmatrix}$$

$$\bar{x}_{\text{new}} = r\text{-dimensions} = \begin{bmatrix} \bar{x} \cdot e_1 & \bar{x} \cdot e_2 & \dots & \bar{x} \cdot e_r \\ \bar{x} \text{ projections on } e_1, e_2, \dots, e_r \end{bmatrix}$$

every eigen vector is a linear combination of  
 "x datapoints"

duality / KKT conditions

$$e_j = \sum_{i=1}^N \beta_j^i x_i = \sum_{i=1}^N \beta_j^i \cdot x_i$$

$$\bar{\beta}^j = (\beta_1^j, \beta_2^j, \dots, \beta_N^j) \text{ dual variable}$$

new rep (x)  
 $\bar{x}$  orig

$$= \bar{x} \cdot \left[ \begin{array}{ccc} \sum_{i=1}^N \beta_1^i x_i & \sum_{i=1}^N \beta_2^i x_i & \dots & \sum_{i=1}^N \beta_N^i x_i \end{array} \right]$$

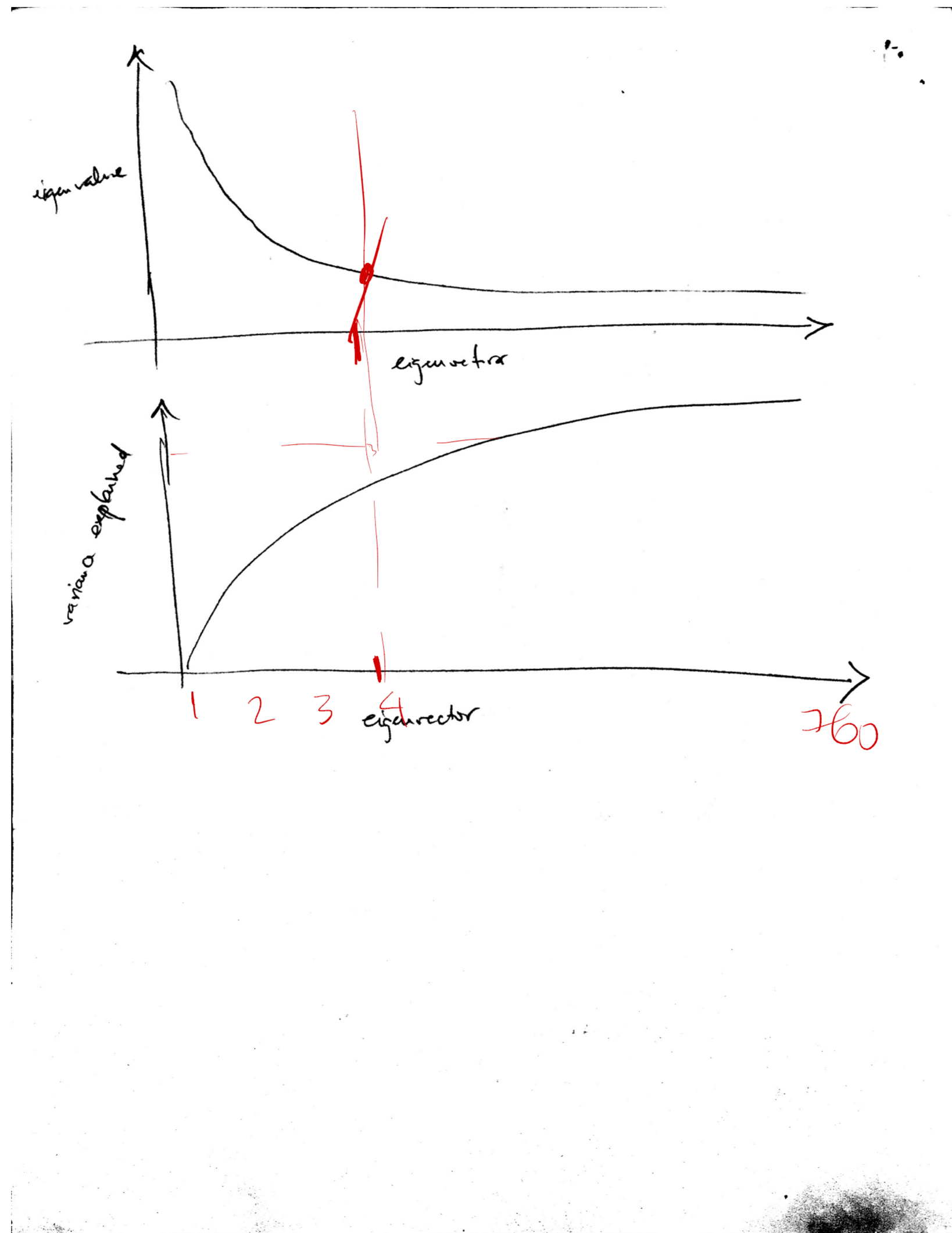
dual  
 eigen vectors in dual form

$$\approx \sum \beta_{ij} \cdot \sum_{i=1}^N x_i$$

$\rightarrow \text{sim}(\bar{x}, x_i)$

$$\bar{x} = \sum \beta_j^i \sum_c \boxed{x_i}$$

new rep(x) = sim vector with other points



**KERNEL PCA** • assume data centered  $\mu_x = 0$  on each column.

$N \times N$

•  $K = X \cdot X^T$  kernel matrix

$K_{ij} = \langle x_i, x_j \rangle = x_i \cdot x_j^T = \text{sim}(x_i, x_j)$

$K$  also symmetric, pos-definite

•  $\Sigma = \text{covar}(X) = X^T X \cdot \frac{1}{N}$   
 $\Sigma$  is  $d \times d$  matrix  
 $= \frac{1}{N} \sum_{i=1}^N x_i^T \cdot x_i$

•  $(e, \lambda)$  eigen vector/value pair for  $\Sigma \Rightarrow \lambda e = \Sigma e = \left( \frac{1}{N} \sum_{i=1}^N x_i^T \cdot x_i \right) e$   
 $\Rightarrow e = \frac{1}{\lambda N} \sum_{i=1}^N x_i^T x_i e = \sum_{i=1}^N \frac{1}{\lambda N} x_i^T (x_i e)$

$e = X^T \cdot \beta^e = \sum_i \left( \beta_{1i} x_{i1}, \beta_{2i} x_{i2}, \dots, \beta_{di} x_{id} \right)$

$e = \beta$ -linear combination of datapoints  $x_i^T$

•  $\beta^e = \begin{bmatrix} \beta_1^e \\ \beta_2^e \\ \vdots \\ \beta_N^e \end{bmatrix}$

are dual variables for  $e$

- PCA representation on new dimension  $e$  for datapoint  $x_j$  is  $x_j \cdot e = x_j \sum_{i=1}^N \beta_i^e x_i^T = \sum_{i=1}^N \beta_i^e x_j x_i^T = \sum_{i=1}^N \beta_i^e K_{ij} = \sum_{i=1}^N \beta_i^e \sin(x_i, x_j)$

- for all datapoints  $X$

$$X \cdot e = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \cdot \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_N \end{bmatrix} = X \cdot X^T \cdot \beta^e = K \cdot \beta^e$$

- for all datapoints  $x$  on all axis  $e_1, e_2, \dots, e_r$  ( $r$ -dim)

$$X \cdot \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_r \end{bmatrix} = X \cdot X^T \cdot \begin{bmatrix} \beta_1^{e_1} & \beta_1^{e_2} & \dots & \beta_1^{e_r} \\ \beta_2^{e_1} & \beta_2^{e_2} & \dots & \beta_2^{e_r} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_N^{e_1} & \beta_N^{e_2} & \dots & \beta_N^{e_r} \end{bmatrix} = K \cdot \begin{bmatrix} \beta^{e_1} \\ \beta^{e_2} \\ \vdots \\ \beta^{e_r} \end{bmatrix}$$

- $K \cdot \beta^e =$  coordinates of all points on new-axis  $e$   
 $N \times N \quad N \times 1$

Calculate  $\beta_i^e$  dual variables for eigenvector  $(e, \lambda)$  of  $\Sigma$

$$\sum_i \beta_i x_i^T = \sum_i \left( \beta_1 x_1^T, \beta_2 x_2^T, \dots, \beta_N x_N^T \right) = X^T \bar{\beta}$$

$$\bar{\beta} = \bar{\beta}^e = (\beta_1, \beta_2, \dots, \beta_N)^T$$

$$\Sigma e = \lambda e \Rightarrow \frac{1}{N} X^T X \cdot X^T \bar{\beta} = \lambda X^T \bar{\beta}$$

Multiply by X to left

$$X \cdot X^T \cdot (X \cdot X^T) \bar{\beta} = X \cdot N \cdot \lambda \cdot X^T \bar{\beta}$$

$$K \cdot K \cdot \bar{\beta} = N \cdot \lambda \cdot (X \cdot X^T) \cdot \bar{\beta}$$

$$K \cdot K \cdot \bar{\beta} = N \lambda \cdot K \cdot \bar{\beta}$$

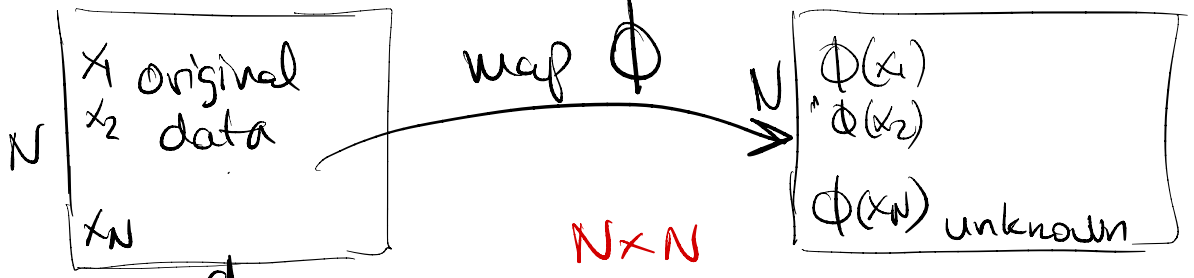
can "divide" by K for all  $\bar{\beta}$  with eigenvals to  $\Rightarrow$

$$K \bar{\beta} = N \lambda \bar{\beta} \Rightarrow \bar{\beta} \text{ is eigenvector for } K \text{ with eigenval } (N \cdot \lambda)$$



# KERNEL TRICK

K can be the dot product  $\text{sim}(x_i, x_j)$



but in a new mapped space  $\phi(x)$  :  $K_{ij} = \langle \phi(x_i) \cdot \phi(x_j) \rangle$

• this is saying K is still  $x \cdot x^T$  like before, but data  $x$  is now  $\phi(x)$

•  $k = x x^T$  is called linear kernel

•  $K_{ij} = e^{-\|x_i - x_j\|^2 / \sigma^2}$  gaussian kernel

•  $K_{ij} = (a \langle x_i x_j \rangle + b)^d$  polynomial kernel

• Laplacian, exponential, circular, quadratic kernels (see slides below)

• all these kernels have an implicit map  $\phi$  so that  $K = \phi(x) \phi(x)^T$   
*we don't know  $\phi$  explicit*

• the theory we developed for PCA with  $K = x x^T$  work for any other kernel  $K$

•  $\beta = \text{eigen vectors}(K)$  we need for KPCA new dim computed as  $K \cdot \beta$

• KERNEL TRICK: we use  $K$  matrix ( $K_{ij}$ ) or  $K$  function

$k_{ij} = \text{function}(x_i, x_j)$  assuming  $K = \phi(x) \cdot \phi(x)^T$

• but without explicit knowing  $\phi$

# CENTERING DATA inside $\mathbf{K}$

$$\mu = \mu(\phi(x)) = \frac{1}{N} \sum_{i=1}^N \phi(x)$$

• if  $\mu \neq 0$  we need to update  $\mathbf{K}$  to match our theory.

•  $\mathbf{K} = \phi(x) \cdot \phi(x)^T$ ; want new  $\bar{\mathbf{K}} = (\phi(x) - \mu)(\phi(x) - \mu)^T$

•  $\bar{K}_{ij} = (\phi_i - \mu) \cdot (\phi_j - \mu)^T$   $\phi_i = \phi(x_i) = \text{ix unknown mapped vector}(x_i)$

$$\begin{aligned} &= \left( \phi_i - \frac{1}{N} \sum_{t=1}^N \phi_t \right) \cdot \left( \phi_j - \frac{1}{N} \sum_{e=1}^N \phi_e \right)^T \\ &= \phi_i \cdot \phi_j - \frac{1}{N} \sum_{t=1}^N \phi_t \cdot \phi_j^T - \frac{1}{N} \sum_{e=1}^N \phi_i \cdot \phi_e^T + \frac{1}{N^2} \sum_{e,t} \phi_t \cdot \phi_e^T \\ &= K_{ij} - \frac{1}{N} \sum_t K_{tj} - \frac{1}{N} \sum_e K_{ie} + \frac{1}{N^2} \sum_{e,t} K_{te} \end{aligned}$$

• matrix form update for  $\mathbf{K} \rightarrow \bar{\mathbf{K}}$

$$\bar{\mathbf{K}} = \mathbf{K} - \mathbf{U}\mathbf{K} - \mathbf{K}\mathbf{U} - \mathbf{U}\mathbf{K}\mathbf{U}$$

$$\mathbf{U}_{N \times N} = \frac{1}{N} \cdot$$

$$\begin{matrix} & \begin{matrix} 1 & 1 & \dots & \dots & 1 \end{matrix} \\ \begin{matrix} 1 \\ 1 \\ \vdots \\ 1 \end{matrix} & \begin{bmatrix} 1 & 1 & \dots & \dots & 1 \\ 1 & 1 & \dots & \dots & 1 \\ \vdots & \vdots & \dots & \dots & \vdots \\ 1 & 1 & \dots & \dots & 1 \end{bmatrix} \end{matrix}$$



[< MATH](#) · [MULTIVARIABLE CALCULUS](#) · [APPLICATIONS OF MULTIVARIABLE DERIVATIVES](#) · [CONSTRAINED OPTIMIZATION \(ARTICLES\)](#)

# Lagrange multipliers, examples

Examples of the Lagrangian and Lagrange multiplier technique in action.

 [Google Classroom](#)

 [Facebook](#)



 [Twitter](#)

[Email](#)

## Background

- [Introduction to Lagrange multipliers](#)
- [Gradient](#)

## Lagrange multiplier technique, quick recap

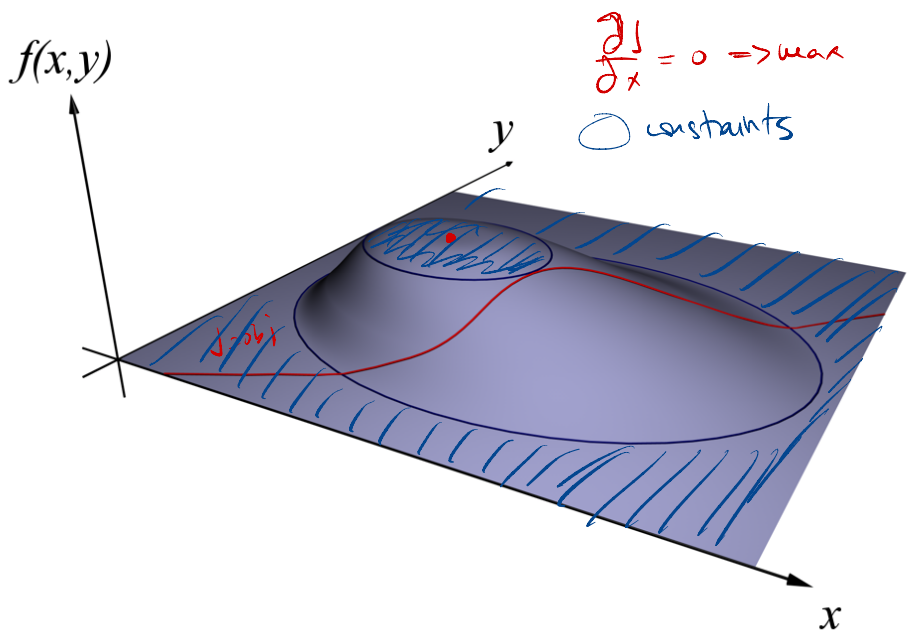


Image credit: By Nexcis (Own work) [Public domain], via [Wikimedia Commons](#)

When you want to maximize (or minimize) a multivariable function  $f(x, y, \dots)$  subject to the constraint that another multivariable function equals a constant,  $g(x, y, \dots) = c$ , follow these steps:

- **Step 1:** Introduce a new variable  $\lambda$ , and define a new function  $\mathcal{L}$  as follows:

$$\mathcal{L}(x, y, \dots, \lambda) = f(x, y, \dots) - \lambda(g(x, y, \dots) - c)$$

This function  $\mathcal{L}$  is called the "Lagrangian", and the new variable  $\lambda$  is referred to as a "Lagrange multiplier"

- **Step 2:** Set the gradient of  $\mathcal{L}$  equal to the zero vector.

$$\nabla \mathcal{L}(x, y, \dots, \lambda) = \mathbf{0} \quad \leftarrow \text{Zero vector}$$

In other words, find the **critical points** of  $\mathcal{L}$ .

- **Step 3:** Consider each solution, which will look something like  $(x_0, y_0, \dots, \lambda_0)$ . Plug each one into  $f$ . Or rather, first remove the  $\lambda_0$  component, then plug it into  $f$ , since  $f$  does not have  $\lambda$  as an input. Whichever one gives the greatest (or smallest) value is the maximum (or minimum) point you are seeking.

## Example 1: Budgetary constraints

### Problem

Suppose you are running a factory, producing some sort of widget that requires steel as a raw material. Your costs are predominantly human labor, which is \$20 per hour for your workers, and the steel itself, which runs for \$170 per ton. Suppose your revenue  $R$  is loosely modeled by the following equation:

$$R(h, s) = 200h^{2/3}s^{1/3}$$

- $h$  represents hours of labor
- $s$  represents tons of steel

If your budget is \$20,000, what is the maximum

possible revenue?

## Solution

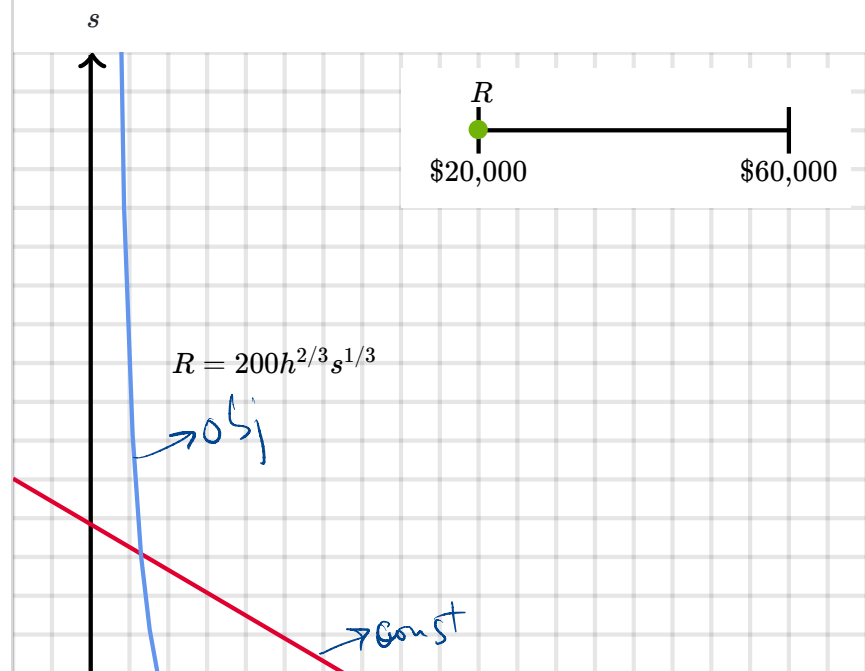
The \$20 per hour labor costs and \$170 per ton steel costs tell us that the total cost of production, in terms of  $h$  and  $s$ , is

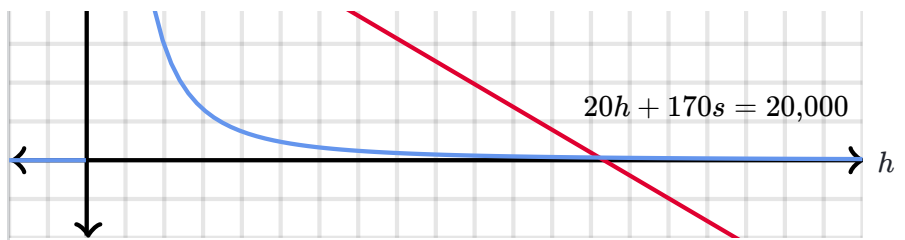
$$20h + 170s$$

Therefore the budget of \$20,000 can be translated to the constraint

$$20h + 170s = 20,000$$

Before we dive into the computation, you can get a feel for this problem using the following interactive diagram. You can see which values of  $(h, s)$  yield a given revenue (blue curve) and which values satisfy the constraint (red line).





Since we need to maximize a function  $R(h, s)$ , subject to a constraint,  $20h + 170s = 20,000$ , we begin by writing the Lagrangian function for this setup:

$$\mathcal{L}(h, s, \lambda) = 200h^{2/3}s^{1/3} - \lambda(20h + 1$$

Next, set the gradient  $\nabla \mathcal{L}$  equal to the  $\mathbf{0}$  vector. This is the same as setting each partial derivative equal to 0. First, we handle the partial derivative with respect to  $h$ .

$$0 = \frac{\partial \mathcal{L}}{\partial h}$$

$$0 = \frac{\partial}{\partial h} (200h^{2/3}s^{1/3} - \lambda(20h + 170s - 20,0$$

$$0 = 200 \cdot \frac{2}{3}h^{-1/3}s^{1/3} - 20\lambda$$

Next, we handle the partial derivative with respect to  $s$ .

$$0 = \frac{\partial \mathcal{L}}{\partial s}$$

$$0 = \frac{\partial}{\partial s} (200h^{2/3}s^{1/3} - \lambda(20h + 170s - 20,000))$$

$$0 = 200 \cdot \frac{1}{3}h^{2/3}s^{-2/3} - 170\lambda$$

Finally we set the partial derivative with respect to  $\lambda$  equal to 0, which as always is just the same thing as the constraint. In practice, you can of course just write the constraint itself, but I'll write out the partial derivative here just to make things clear.

$$0 = \frac{\partial \mathcal{L}}{\partial \lambda}$$

$$0 = \frac{\partial}{\partial \lambda} (200h^{2/3}s^{1/3} - \lambda(20h + 170s - 20,000))$$

$$0 = -20h - 170s + 20,000$$

$$20h + 170s = 20,000$$

Putting it together, the system of equations we need to solve is



$$0 = 200 \cdot \frac{2}{3} h^{-1/3} s^{1/3} - 20\lambda$$

$$0 = 200 \cdot \frac{1}{3} h^{2/3} s^{-2/3} - 170\lambda$$

$$20h + 170s = 20,000$$

In practice, you should almost always use a computer once you get to a system of equations like this. Especially because the equation will likely be more complicated than these in real applications.

Once you do, you'll find that the answer is

$$h = \frac{2,000}{3} \approx 666.667$$

$$s = \frac{2,000}{51} \approx 39.2157$$

$$\lambda = \sqrt[3]{\frac{8,000}{459}} \approx 2.593$$

This means you should employ about 667 hours of labor, and purchase 39 tons of steel, which will give a maximum revenue of

$$R(667, 39) = 200(667)^{2/3} (39)^{1/3} \approx \boxed{\$51,777}$$

MULTIVARIABLE  
CALCULUS > APPLICATIONS OF  
MULTIVARIABLE DERIVATIVES

Constrained optimization  
(articles)



Lagrange multipliers,  
introduction



Lagrange multipliers,  
examples



Interpretation of Lagrange  
multipliers

The interpretation of this constant  $\lambda = 2.593$  is left to the [next article](#)

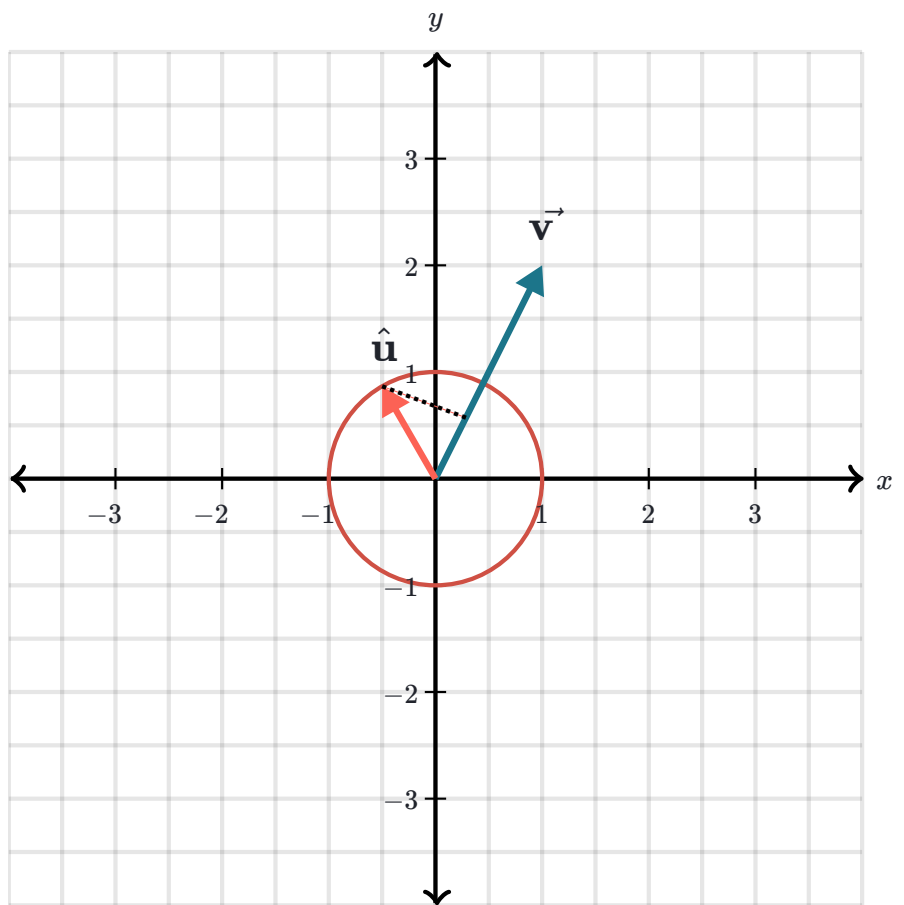
## Example 2: Maximizing dot product

**Problem:** Let the three-dimensional vector  $\vec{\mathbf{v}}$  be defined as follows.

$$\vec{\mathbf{v}} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

Consider every possible unit vector  $\hat{\mathbf{u}}$  in three-dimensional space. For which one is the dot product  $\hat{\mathbf{u}} \cdot \vec{\mathbf{v}}$  the greatest?

The diagram below is two-dimensional, but not much changes in the intuition as we move to three dimensions.



Two-dimensional analogy to the three-dimensional problem we have. Which unit vector  $\hat{\mathbf{u}}$  maximizes the dot product  $\hat{\mathbf{u}} \cdot \mathbf{v}$ ?

If you are fluent with dot products, you may already know the answer. It's one of those mathematical facts worth remembering. If you don't know the answer, all the better! Because we will now find and prove the result using the Lagrange multiplier method.

**Solution:**

First, we need to spell out how exactly this is a constrained optimization problem. Write the coordinates of our unit vectors as  $x$ ,  $y$  and  $z$ :

$$\hat{\mathbf{u}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

The fact that  $\hat{\mathbf{u}}$  is a **unit vector** means its magnitude is 1:

$$\|\hat{\mathbf{u}}\| = \sqrt{x^2 + y^2 + z^2} = 1$$
$$\Downarrow$$
$$x^2 + y^2 + z^2 = 1$$

This is our constraint.

Maximizing  $\hat{\mathbf{u}} \cdot \vec{\mathbf{v}}$  means maximizing the following quantity:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} = 2x + 3y + z$$

The Lagrangian, with respect to this function and the constraint above, is

$$\mathcal{L}(x, y, z, \lambda) = 2x + 3y + z - \lambda(x^2 + y^2 + z^2 - 1)$$

We now solve for  $\nabla \mathcal{L} = \mathbf{0}$  by setting each partial derivative of this expression equal to 0.

$$\frac{\partial}{\partial x}(2x + 3y + z - \lambda(x^2 + y^2 + z^2 - 1))$$

$$\frac{\partial}{\partial y}(2x + 3y + z - \lambda(x^2 + y^2 + z^2 - 1))$$

$$\frac{\partial}{\partial z}(2x + 3y + z - \lambda(x^2 + y^2 + z^2 - 1))$$

Remember, setting the partial derivative with respect to  $\lambda$  equal to 0 just restates the constraint.

$$\frac{\partial}{\partial \lambda}(2x + 3y + z - \lambda(x^2 + y^2 + z^2 - 1)) = -x$$

Solving for  $x$ ,  $y$  and  $z$  in the first three equations above, we get

$$x = 2 \cdot \frac{1}{2\lambda}$$

$$y = 3 \cdot \frac{1}{2\lambda}$$

$$z = 1 \cdot \frac{1}{2\lambda}$$

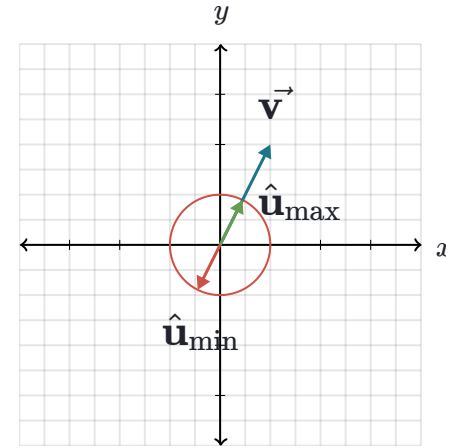
Ah, what beautiful symmetry. Each of these expressions has the same  $\frac{1}{2\lambda}$  factor, and the coefficients 2, 3 and 1 match up with the coordinates of  $\vec{v}$ . Being good math students as we are, we won't let good symmetry go to waste. In this case, combining the three equations above into a single vector equation, we can relate  $\hat{u}$  and  $\vec{v}$  as follows:

$$\hat{\mathbf{u}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{1}{2\lambda} \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} = \frac{1}{2\lambda} \mathbf{v}^{\rightarrow}$$

Therefore  $\hat{\mathbf{u}}$  is proportional to  $\mathbf{v}^{\rightarrow}$ !

Geometrically, this means  $\hat{\mathbf{u}}$  points in the same direction as  $\mathbf{v}^{\rightarrow}$ . There are two unit vectors proportional  $\mathbf{v}^{\rightarrow}$ ,

- One which points in the same direction, this is the vector that **maximizes**  $\hat{\mathbf{u}} \cdot \mathbf{v}^{\rightarrow}$ .
- One which points in the opposite direction. This one **minimizes**  $\hat{\mathbf{u}} \cdot \mathbf{v}^{\rightarrow}$ .



Two-dimensional analogy showing the two unit vectors which maximize and minimize the quantity  $\hat{\mathbf{u}} \cdot \mathbf{v}^{\rightarrow}$ .

We can write these two unit vectors by normalizing  $\mathbf{v}^{\rightarrow}$ , which just means dividing  $\mathbf{v}^{\rightarrow}$  by its magnitude:

$$\hat{\mathbf{u}}_{\max} = \frac{\mathbf{v}^{\rightarrow}}{\|\mathbf{v}^{\rightarrow}\|}$$

$$\hat{\mathbf{u}}_{\min} = -\frac{\mathbf{v}^{\rightarrow}}{\|\mathbf{v}^{\rightarrow}\|}$$

The magnitude  $\|\mathbf{v}^{\rightarrow}\|$  is  $\sqrt{2^2 + 3^2 + 1^2} = \sqrt{14}$ , so we can write the maximizing unit vector  $\hat{\mathbf{u}}_{\max}$  explicitly

as like this:

$$\hat{\mathbf{u}}_{\max} = \begin{bmatrix} 2/\sqrt{14} \\ 3/\sqrt{14} \\ 1/\sqrt{14} \end{bmatrix}$$

## Just skip the Lagrangian

If you read the [last article](#), you'll recall that the whole point of the Lagrangian  $\mathcal{L}$  is that setting  $\nabla\mathcal{L} = 0$  encodes the two properties a constrained maximum must satisfy:

- Gradient alignment between the target function and the constraint function,

$$\nabla f(x, y) = \lambda \nabla g(x, y)$$

- The constraint itself,

$$g(x, y) = c$$

When working through examples, you might wonder why we bother writing out the Lagrangian at all. Wouldn't it be easier to just start with these two equations rather than re-establishing them from  $\nabla\mathcal{L} = 0$  every time? The short answer is yes, it would be easier. If you find yourself solving a constrained

optimization problem by hand, and you remember the idea of gradient alignment, feel free to go for it without worrying about the Lagrangian.

In practice, it's often a computer solving these problems, not a human. Given that there are many highly optimized programs for finding when the gradient of a given function is 0, it's both clean and useful to encapsulate our problem into the equation  $\nabla \mathcal{L} = 0$ .

Furthermore, the Lagrangian itself, as well as several functions deriving from it, arise frequently in the theoretical study of optimization. In this light, reasoning about the single object  $\mathcal{L}$  rather than multiple conditions makes it easier to see the connection between high-level ideas. Not to mention, it's quicker to write down on a blackboard.

In either case, whatever your future relationship with constrained optimization might be, it is good to be able to think about the Lagrangian itself and what it does. The examples above illustrate how it works, and hopefully help to drive home the point that  $\nabla \mathcal{L} = 0$  encapsulates both  $\nabla f = \lambda \nabla g$  and  $g(x, y) = c$  in a single equation.



Ask a question...

## Questions Tips & Thanks

[Top](#) [Recent](#)


In example 2, why do we put a hat on  $u$ ? Is it because it is a unit vector, or because it is the vector that we are looking for?

6 votes ▲ ▼ • [Comment](#) • [Flag](#) 2 years ago by  clara.vdw

It is because it is a unit vector. Unit vectors will typically have a hat on them.

7 votes ▲ ▼ • [Comment](#) • [Flag](#) 2 years ago by  u.yu16

Use the method of Lagrange multipliers to compute the Optimal investments  $x$  and  $y$  in mutual Funds 1 and 2 respectively. An expressions for  $x$  and  $y$  should not contain the lagrange multiplier

2 votes ▲ ▼ • [Comment](#) • [Flag](#) about a year ago by  Learner

Instead of constraining optimization to a curve on  $x$ - $y$  plane, is there which a method to constrain the optimization to a region/area on the  $x$ - $y$  plane. Like the region  $x^2+y^2 \leq 2$  which r all the points in the unit circle

including the boundary.

1 vote ▲ ▼ • Comment • Flag

9 months ago by  hamadm077

---

For problems where the number of constraints is one less than the number of variables (ie every example we've gone over except the unit vector one), is there a reason why we can't just solve the system of equations of the function and constraint? ie the result is a single-variable function; take its derivative and set to 0.

1 vote ▲ ▼ • Comment • Flag

about a year ago by  David O'Connor

---

how do you maximize this function subject to the constraint

$$f(x,y)=x^2-y^2+3, 2x+y=3$$

1 vote ▲ ▼ • Comment • Flag

10 months ago by  jam008

---

Hello and really thank you for your amazing site. Can you please explain me why we dont use the whole Lagrange but only the first part? Why we dont use the 2nd derivatives

1 vote ▲ ▼ • Comment • Flag

3 months ago by  nikostogas

---

what shuld we do if we have constraints as well as boundaries and we need a local extrima?

1 vote ▲ ▼ • Comment • Flag

2 years ago by 🐼 Garbage can jr.

---

At the start of example 1, it would be good if you mentioned that the problem is very hard to solve completely by hand, so that people don't waste their time.

0 votes ▲ ▼ • Comment • Flag

about a year ago by 🍃 Zaz Brown

---

Its indeed tricky, but I found it usefull and good practice.

1 vote ▲ ▼ • Comment • Flag

10 months ago by 🦋 aflenoir

---

find the temperature  $f(x,y,z)$  at any point in space is  $f=400xyz^2$ .find the highest temperature on the surface of the sphere  $x^2+y^2+z^2=1$

0 votes ▲ ▼ • 1 comment • Flag

2 years ago by 🍃 gakhil1018

---

[◀ Lagrange multipliers, introduction](#)

[Interpretation of Lagrange multipliers ▶](#)

Our mission is to provide a free, world-class education to anyone, anywhere.

Khan Academy is a 501(c)(3) nonprofit organization. **Donate** or **volunteer** today!

About

News

Impact

Our team

Our interns

Our content specialists

Our leadership

Our supporters

Our contributors

Careers

Internships

**Contact**

Help center

Support community

Share your story

Press

**Download our apps**

iOS app

Android app

## Subjects

[Math by subject](#)

[Math by grade](#)

[Science & engineering](#)

[Computing](#)

[Arts & humanities](#)

[Economics & finance](#)

[Test prep](#)

[College, careers, & more](#)

Language [English](#)



# UNSUPERVISED LEARNING 2011

## LECTURE :KERNEL PCA

Rita Osadchy

Some slides are due to Scholkopf, Smola, Muller, and Precup

# Dimensionality Reduction

- Data representation

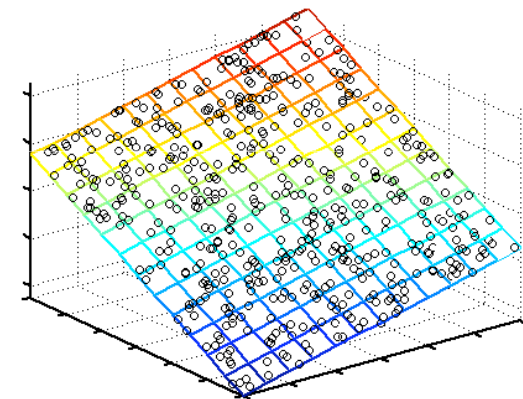
Inputs are real-valued vectors in a high dimensional space.

- Linear structure

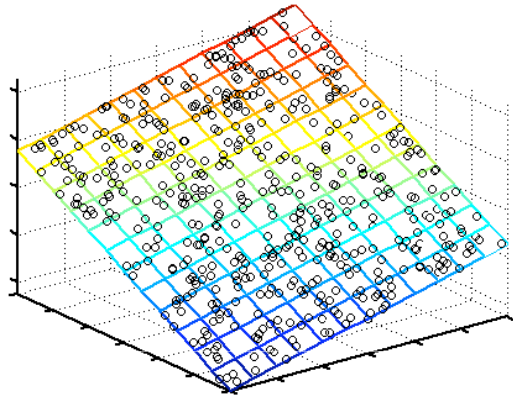
Does the data live in a low dimensional subspace?

- Nonlinear structure

Does the data live on a low dimensional submanifold?



# Dimensionality Reduction so far

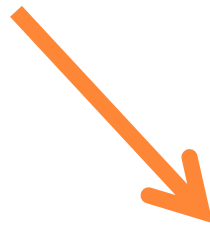


**PCA**

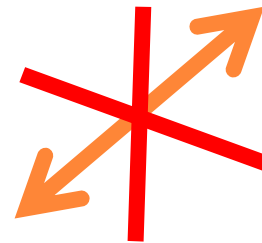


**Manifold learning methods**

for non linear  
structure



**Kernel PCA**



but does not  
unfold the data



# Notations

- Inputs (**high dimensional**)

$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$  points in  $\mathbb{R}^D$

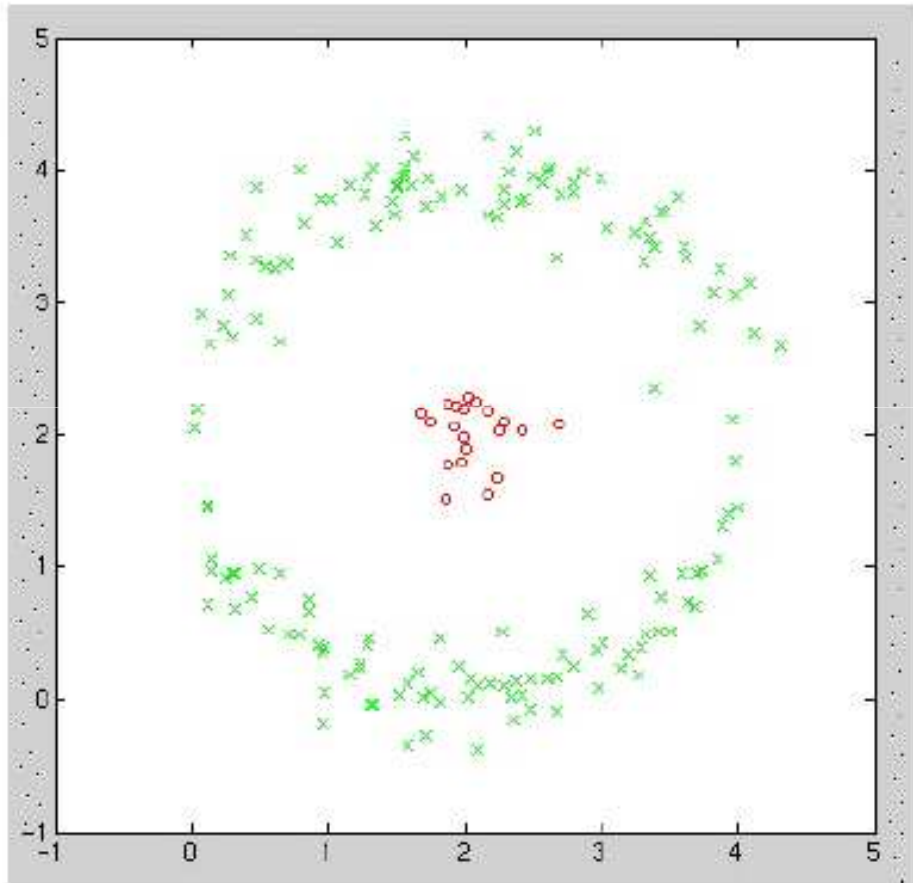
- Outputs (**low dimensional**)

$\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n$  points in  $\mathbb{R}^d$  ( $d \ll D$ )

# The “magic” of high dimensions

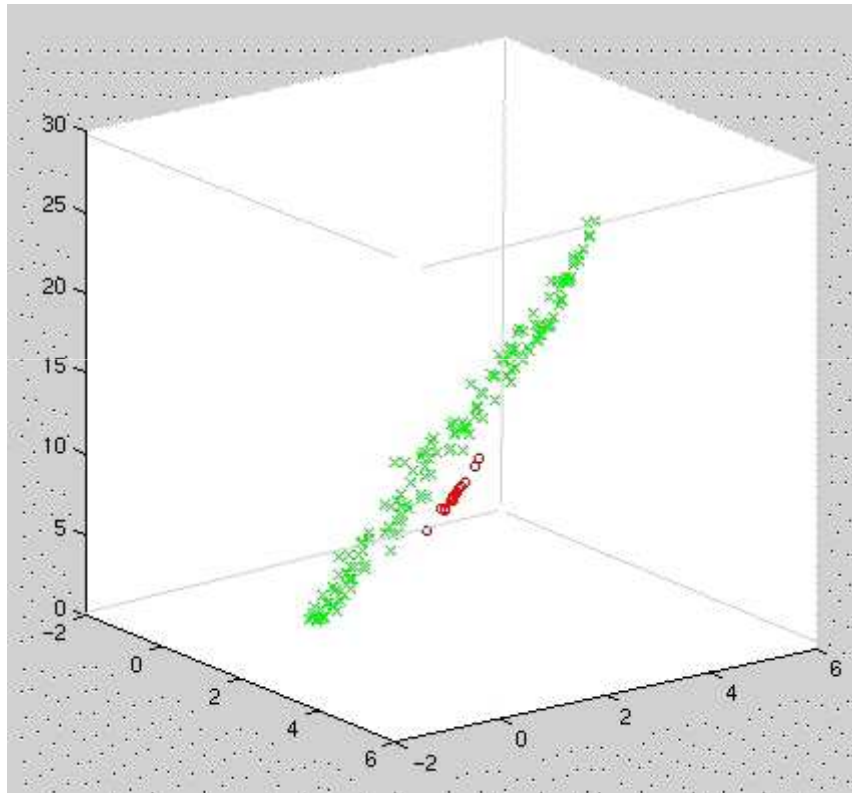
- Given some problem, how do we know what classes of functions are capable of solving that problem?
- VC (Vapnik-Chervonenkis) theory tells us that often mappings which take us into a higher dimensional space than the dimension of the input space provide us with greater classification power.

# Example in $\mathbb{R}^2$



These classes are linearly inseparable in the input space.

# Example: High-Dimensional Mapping



We can make the problem linearly separable by a simple mapping

$$\Phi : \mathbf{R}^2 \rightarrow \mathbf{R}^3$$

$$(x_1, x_2) \mapsto (x_1, x_2, \underline{x_1^2 + x_2^2})$$

Kernel trick : don't need  $\phi$   
just  $k(x, x_i)$   
math  $\Leftarrow$  guarantee  $k$  valid

# Kernel Trick

- High-dimensional mapping can seriously increase computation time.
- Can we get around this problem and still get the benefit of high-D?

Yes! **Kernel Trick**

$$K(x_i, x_j) = \underbrace{\phi(x_i)}_{A^T} \cdot \underbrace{\phi(x_j)}_A$$

Kernel  
dot product in high-dim space  
 $\phi =$  hidden map  
 $\Rightarrow$  sim. pos. def.

- Given *any* algorithm that can be expressed solely in terms of dot products, this trick allows us to construct different nonlinear versions of it.

# Popular Kernels

**Gaussian**  $K(\vec{x}, \vec{x}') = \exp(-\beta \|\vec{x} - \vec{x}'\|^2)$

**Polynomial**  $K(\vec{x}, \vec{x}') = (1 + \vec{x} \cdot \vec{x}')^p$

**Hyperbolic tangent**  $K(\vec{x}, \vec{x}') = \tanh(\vec{x} \cdot \vec{x}' + \delta)$

# Kernel Principal Component Analysis (KPCA)

- Extends conventional principal component analysis (PCA) to a high dimensional feature space using the “kernel trick”.
- Can extract up to  $n$  (number of samples) nonlinear principal components without expensive computations.

# Making PCA Non-Linear

- Suppose that instead of using the points  $x_i$  we would first map them to some nonlinear **feature space**  $\phi(x_i)$ 
  - E.g. using polar coordinates instead of cartesian coordinates would help us deal with the circle.
- Extract principal component in that space (PCA)
- The result will be non-linear in the original data space!



# Derivation

- Suppose that the mean of the data in the feature space is

$$\mu = \frac{1}{n} \sum_{i=1}^n \phi(x_i) = 0$$

- Covariance:

$$C = \frac{1}{n} \sum_{i=1}^n \phi(x_i) \phi(x_i)^T$$

- Eigenvectors

$$Cv = \lambda v$$

# Derivation Cont.

- Eigenvectors can be expressed as linear combination of features:

$$v = \sum_{i=1}^n \alpha_i \phi(x_i)$$

- Proof:

$$Cv = \frac{1}{n} \sum_{i=1}^n \phi(x_i) \phi(x_i)^T v = \lambda v$$

thus

$$v = \frac{1}{\lambda n} \sum_{i=1}^n \phi(x_i) \phi(x_i)^T v = \frac{1}{\lambda n} \sum_{i=1}^n (\phi(x_i) \cdot v) \phi(x_i)^T$$

Showing that  $\mathbf{x}\mathbf{x}^T\mathbf{v} = (\mathbf{x}\cdot\mathbf{v})\mathbf{x}^T$

$$\begin{aligned}(\mathbf{x}\mathbf{x}^T)\mathbf{v} &= \begin{pmatrix} x_1x_1 & x_1x_2 & \dots & x_1x_M \\ x_2x_1 & x_2x_2 & \dots & x_2x_M \\ \vdots & \vdots & \ddots & \vdots \\ x_Mx_1 & x_Mx_2 & \dots & x_Mx_M \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_M \end{pmatrix} \\ &= \begin{pmatrix} x_1x_1v_1 + x_1x_2v_2 + \dots + x_1x_Mv_M \\ x_2x_1v_1 + x_2x_2v_2 + \dots + x_2x_Mv_M \\ \vdots \\ x_Mx_1v_1 + x_Mx_2v_2 + \dots + x_Mx_Mv_M \end{pmatrix}\end{aligned}$$

Showing that  $\mathbf{x}\mathbf{x}^T\mathbf{v} = (\mathbf{x}\cdot\mathbf{v})\mathbf{x}^T$

$$\begin{aligned} &= \begin{pmatrix} (x_1v_1 + x_2v_2 + \dots + x_Mv_M)x_1 \\ (x_1v_1 + x_2v_2 + \dots + x_Mv_M)x_2 \\ \vdots \\ (x_1v_1 + x_2v_2 + \dots + x_Mv_M)x_M \end{pmatrix} \\ &= \begin{pmatrix} x_1v_1 + x_2v_2 + \dots + x_Mv_M \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{pmatrix} \\ &= (\mathbf{x}\cdot\mathbf{v})\mathbf{x} \quad \square \end{aligned}$$

# Derivation Cont.

- So, from before we had,

$$v = \frac{1}{n\lambda} \sum_{i=1}^n \phi(x_i)\phi(x_i)^T v = \frac{1}{n\lambda} \sum_{i=1}^n (\phi(x_i) \cdot v) \phi(x_i)^T$$

just a scalar

- this means that all solutions  $v$  with  $\lambda = 0$  lie in the span of  $\phi(x_1), \dots, \phi(x_n)$ , i.e.,

*eigenvector* ←  $v = \sum_{i=1}^n \alpha_i \phi(x_i)$  → *dual vector*

- Finding the eigenvectors is equivalent to finding the coefficients  $\alpha_i$

# Derivation Cont.

- By substituting this back into the equation we get:

$$\frac{1}{n} \sum_{i=1}^n \phi(x_i) \phi(x_i)^T \left( \sum_{l=1}^n \alpha_{jl} \phi(x_l) \right) = \lambda_j \sum_{l=1}^n \alpha_{jl} \phi(x_l)$$

- We can rewrite it as

$$\frac{1}{n} \sum_{i=1}^n \phi(x_i) \left( \sum_{l=1}^n \alpha_{jl} K(x_i, x_l) \right) = \lambda_j \sum_{l=1}^n \alpha_{jl} \phi(x_l)$$

- Multiply this by  $\phi(x_k)$  from the left:

$$\frac{1}{n} \sum_{i=1}^n \phi(x_k)^T \phi(x_i) \left( \sum_{l=1}^n \alpha_{jl} K(x_i, x_l) \right) = \lambda_j \sum_{l=1}^n \alpha_{jl} \phi(x_k)^T \phi(x_l)$$

# Derivation Cont.

- By plugging in the kernel and rearranging we get:

$$\mathbf{K}^2 \alpha_j = n \lambda_j \mathbf{K} \alpha_j$$

We can remove a factor of  $\mathbf{K}$  from both sides of the matrix (this will only affect the eigenvectors with zero eigenvalue, which will not be a principle component anyway):

$$\mathbf{K} \alpha_j = n \lambda_j \alpha_j$$

- We have a normalization condition for the  $\alpha_j$  vectors:

$$\mathbf{v}_j^T \mathbf{v}_j = 1 \Rightarrow \sum_{k=1}^n \sum_{l=1}^n \alpha_{jl} \alpha_{jk} \phi(x_l)^T \phi(x_k) = 1 \Rightarrow \alpha_j^T \mathbf{K} \alpha_j = 1$$

# Derivation Cont.

- By multiplying  $K\alpha_j = n\lambda_j\alpha_j$  by  $\alpha_j$  and using the normalization condition we get:

$$\lambda_j n \alpha_j^T \alpha_j = 1, \quad \forall j$$

- For a new point  $x$ , its projection onto the principal components is:

$$\phi(x)^T v_j = \sum_{i=1}^n \alpha_{ji} \phi(x)^T \phi(x_i) = \sum_{i=1}^n \alpha_{ji} K(x, x_i)$$



# Normalizing the feature space

- In general,  $\phi(x_i)$  may not be zero mean.
- Centered features:

$$\tilde{\phi}(x_k) = \phi(x_i) - \frac{1}{n} \sum_{k=1}^n \phi(x_k)$$

- The corresponding kernel is:

$$\begin{aligned} \tilde{K}(x_i, x_j) &= \tilde{\phi}(x_i)^T \tilde{\phi}(x_j) \\ &= \left( \phi(x_i) - \frac{1}{n} \sum_{k=1}^n \phi(x_k) \right)^T \left( \phi(x_j) - \frac{1}{n} \sum_{k=1}^n \phi(x_k) \right) \\ &= K(x_i, x_j) - \frac{1}{n} \sum_{k=1}^n K(x_i, x_k) - \frac{1}{n} \sum_{k=1}^n K(x_j, x_k) + \frac{1}{n^2} \sum_{l,k=1}^n K(x_l, x_k) \end{aligned}$$

# Normalizing the feature space (cont)

$$\tilde{K}(x_i, x_j) = K(x_i, x_j) - \frac{1}{n} \sum_{k=1}^n K(x_i, x_k) - \frac{1}{n} \sum_{k=1}^n K(x_j, x_k) + \frac{1}{n^2} \sum_{l,k=1}^n K(x_l, x_k)$$

- In a matrix form

$$\tilde{\mathbf{K}} = \mathbf{K} - 2\mathbf{1}_{1/n} \mathbf{K} + \mathbf{1}_{1/n} \mathbf{K} \mathbf{1}_{1/n}$$

- where  $\mathbf{1}_{1/n}$  is a matrix with all elements  $1/n$ .

# Summary of kernel PCA

- Pick a kernel
- Construct the normalized kernel matrix of the data (dimension  $m \times m$ ):

$$\tilde{K} = K - 2\mathbf{1}_{1/n} K + \mathbf{1}_{1/n} K \mathbf{1}_{1/n}$$

- Solve an eigenvalue problem:

$$\tilde{K} \alpha_i = \lambda_i \alpha_i$$

- For any data point (new or old), we can represent it as

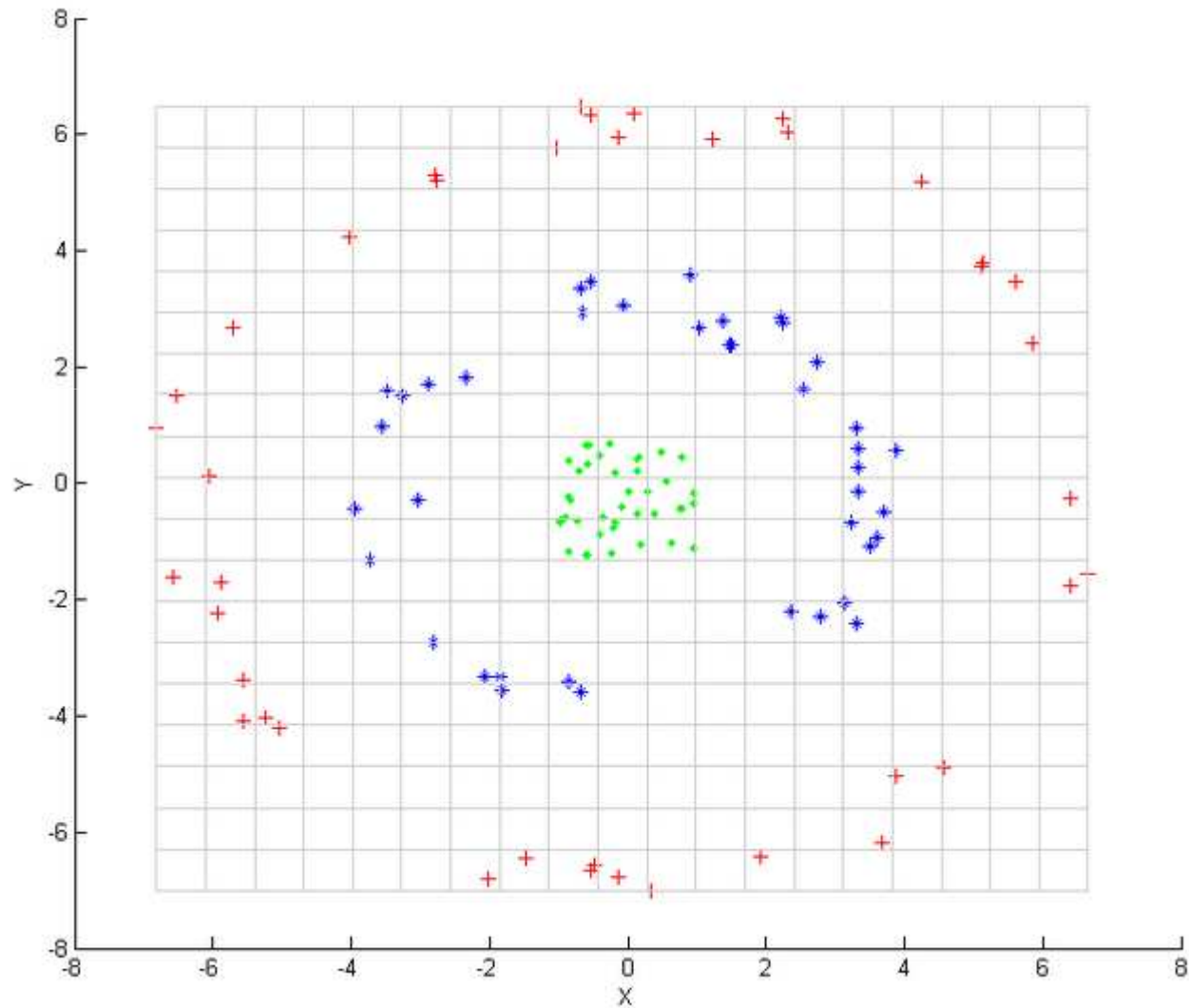
*x = orig point*

*new rep*

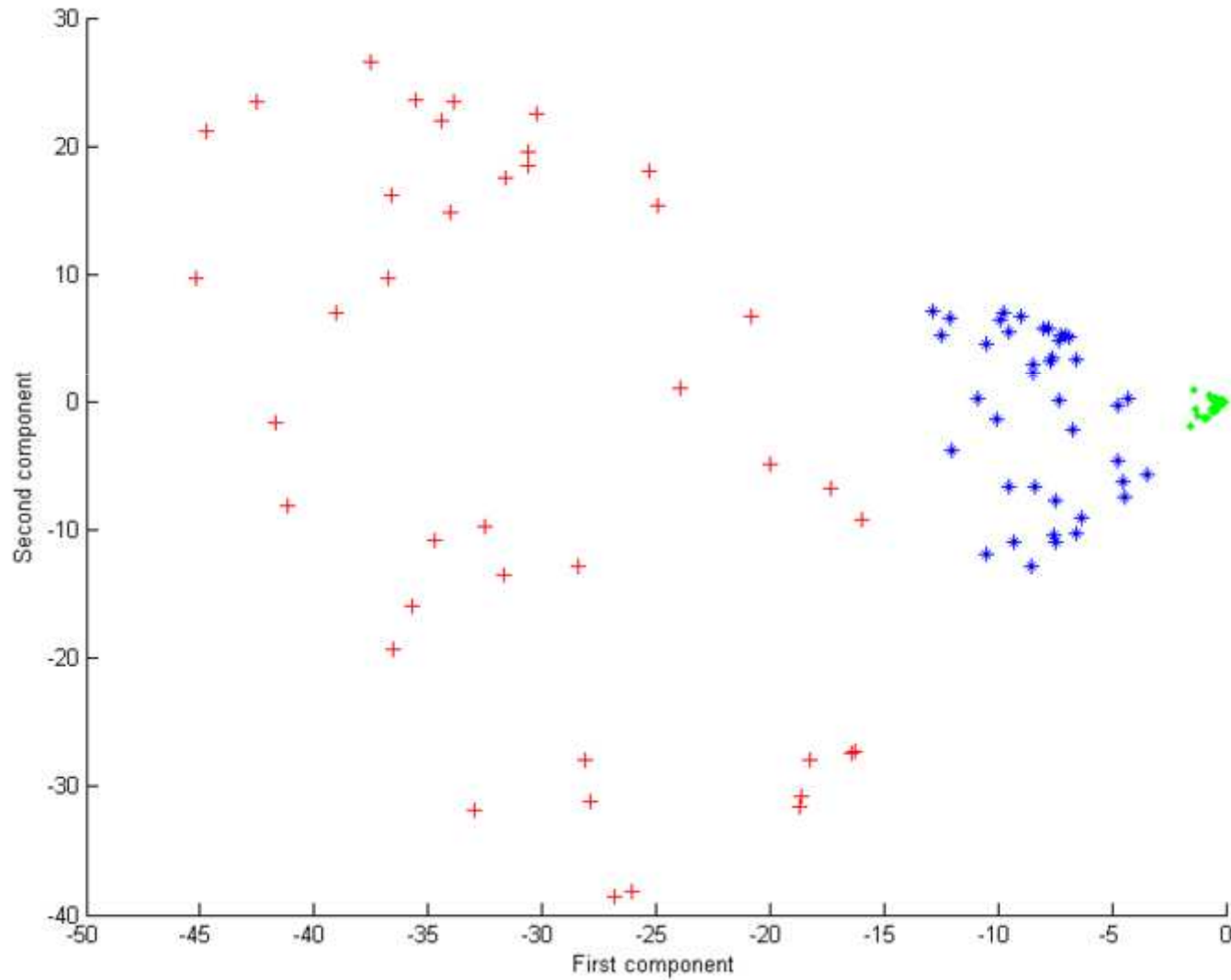
$$y_j = \sum_{i=1}^n \alpha_{ji} K(x, x_i), \quad j = 1, \dots, d$$

*dual feat =  $\sin(x_j x_i)$*

# Example: Input Points



# Example: KPCA



# Example: De-noising images

Original data



Data corrupted with Gaussian noise



Result after linear PCA



Result after kernel PCA, Gaussian kernel



# Properties of KPCA

- Kernel PCA can give a good re-encoding of the data when it lies along a non-linear manifold.
- The kernel matrix is  $n \times n$ , so kernel PCA will have difficulties if we have lots of data points.