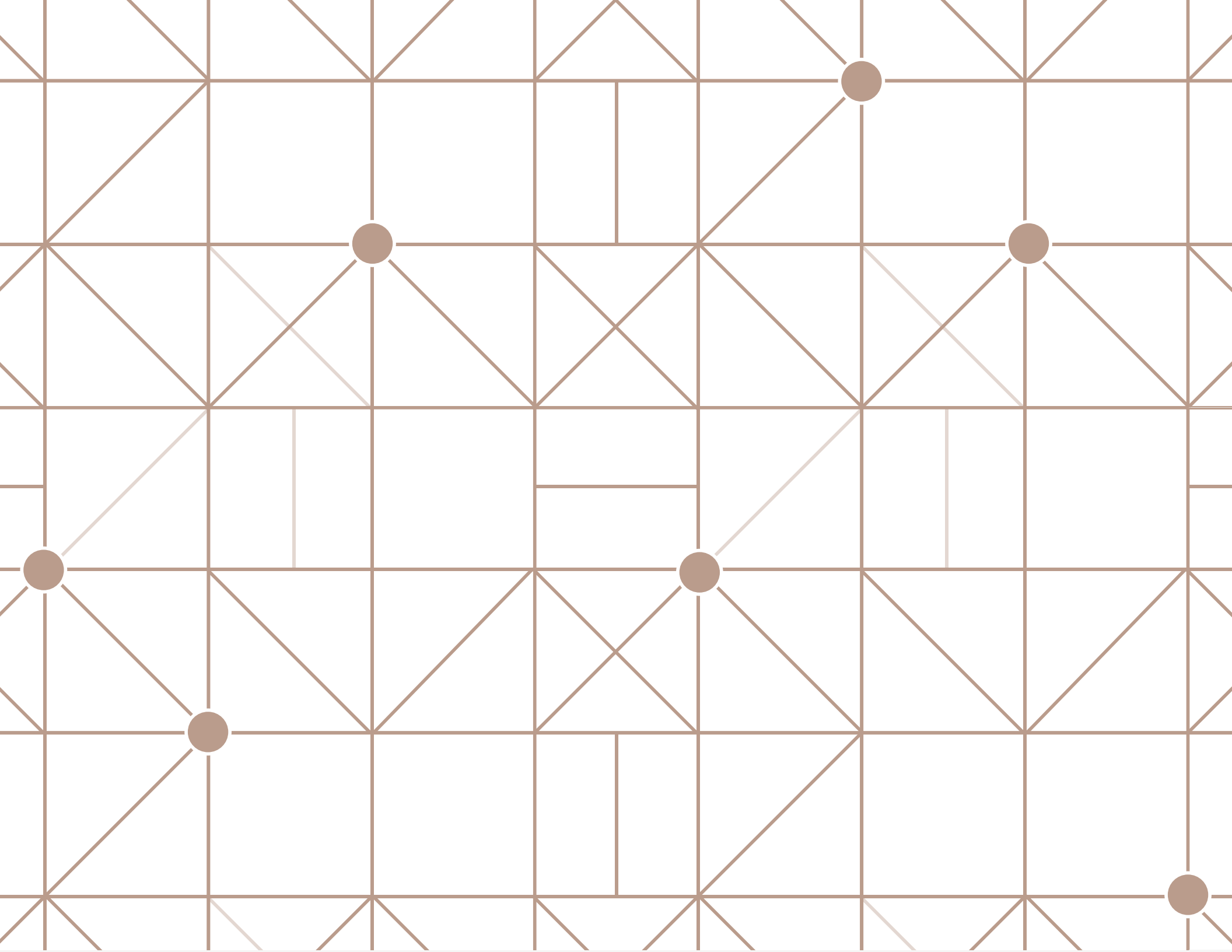
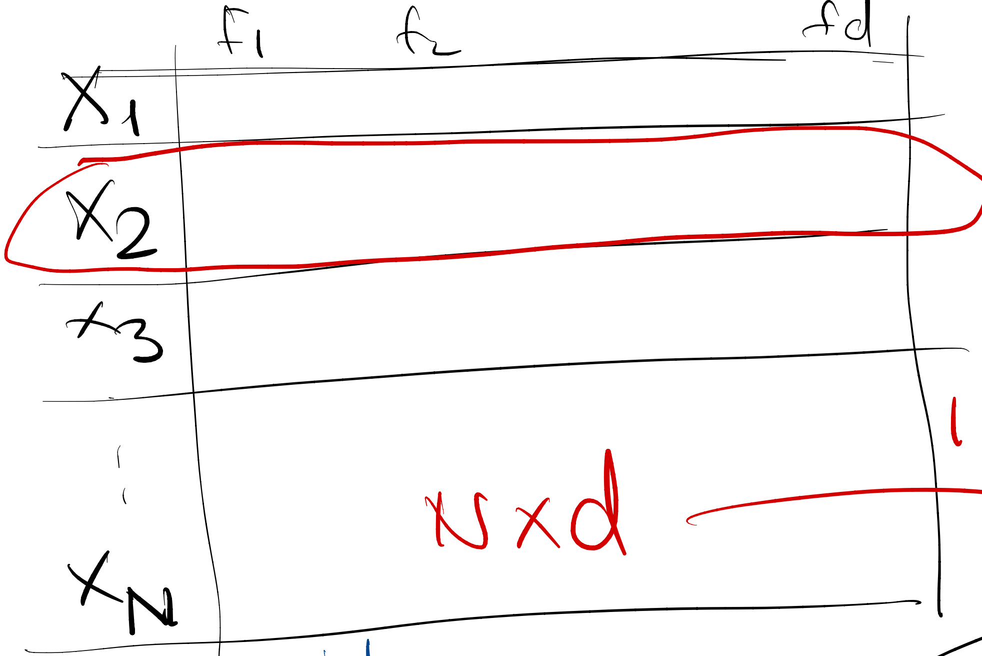


- project on t lines $d \rightarrow t$
  line $= \bar{e}$ $\|e\|=1$
- $x_k \xrightarrow{\text{projected}} e^T(x_k - \mu)$ new rep. on line e
- $J(\ ) = SSE$ min $\Rightarrow$ max var $\Rightarrow$ max $e^T \Sigma e$
- solve max $e^T \Sigma e$ s.t. $\|e\|=1$ $\Rightarrow$ Lagrange $\Sigma e = \lambda e$
  e = eigenvector (Σ)
- $\Sigma = (x-\mu)(x-\mu)^T$ $\begin{cases} \text{sim} \\ \text{pos def} \end{cases} \Rightarrow$ spectral decomp
  $[e][\lambda][e]$

Datapoints (vectors)



new representation on t features

A: t << d

N x t

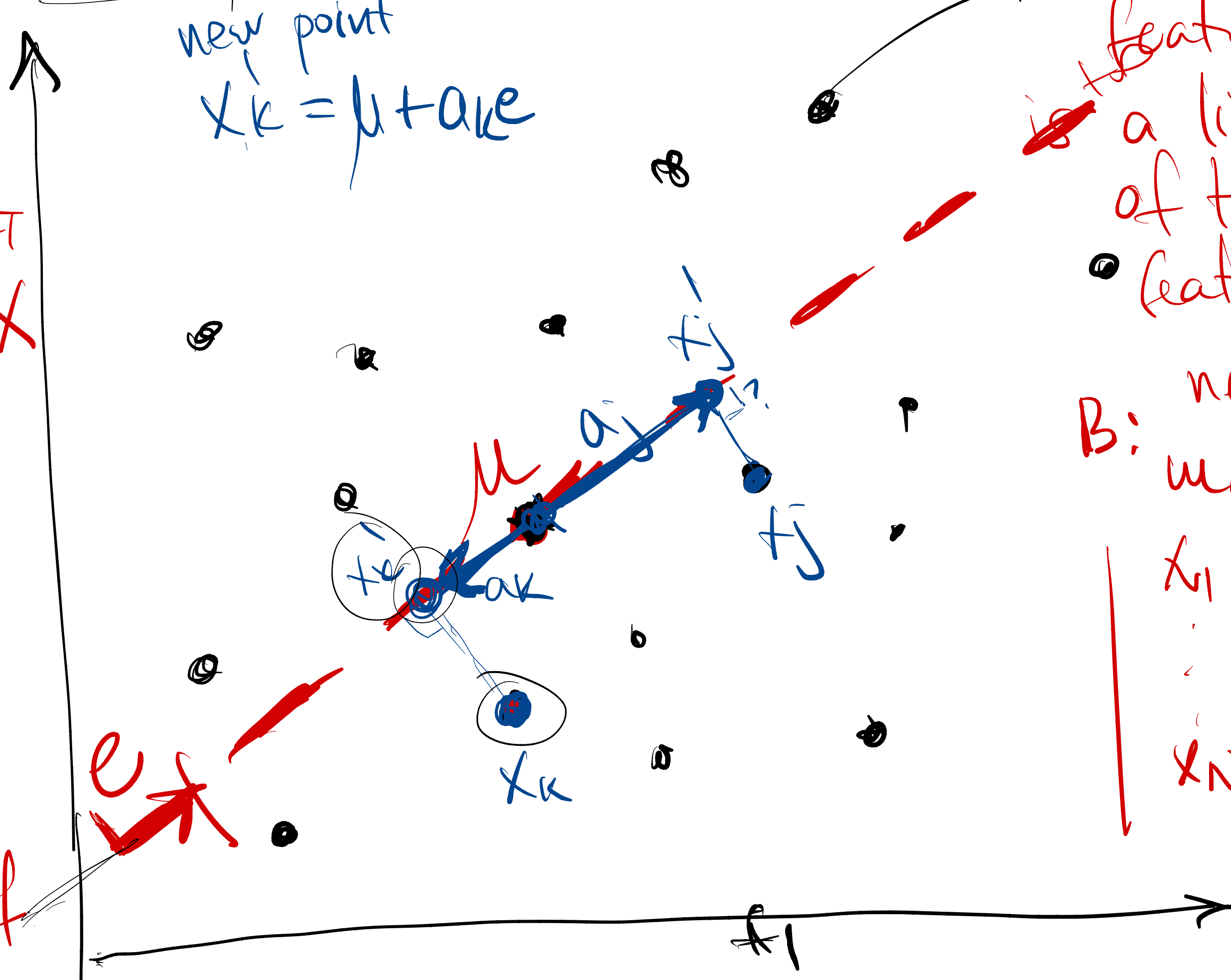
each new feature gt is a linear comb. of the original features

$d=2$ visualization

① t=0 NOFEAT
→ represent whole X on 1 point

$\mu = mean(x)$
$= E[x]$

$|e|=1$ direction of line



B: new data matrix

	t feat
x1	
xN	

$\sim X_{orig}$

$t=1 \Rightarrow 1$ dim $\xrightarrow{\text{new}}$ $\Rightarrow$ new representation will be on 1 line

guess: that "one line" new rep
- passes through μ
- visually correspond to longest-direction-stretch of data.

geom line: e; $a_1 a_2 \dots a_N$ (new coordinates)

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

$\mu + a_k e$: new point; x_k: orig point

$x_k = \mu + a_k e$

$x_k - \mu = a_k \cdot e$

$(x_k - \mu)e = a_k$

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

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

projection

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

$E[x_k - \mu] = 0$

$= \mu$

$$J(e) \quad \text{implicit projections } x \to e = a \Big) = \sum_k a_k^2 - 2\sum_k a_k^2 + \sum_k \|x_k - \mu\|^2$$

$a_k \cdot e^T (x_k - \mu)$, a_k

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

a_k^2

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

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

$$- e^T \Sigma e$$

sigma

covar matrix Σ

minimize $J(e)$

$\Updownarrow$

$\min -e^T \Sigma e$ / $\text{MAX } e^T \Sigma e$

$\Updownarrow$ $\Updownarrow$

max variance (projections)

MOST IMPORTANT: what is the best e? want MAX

MAX Var[projections = new representations]

$$E\left[\left(\underbrace{\mu + a_k e}_{\text{proj}} - E[\mu + a_k e]\right)^2\right] = E\left[(\mu + a_k e - \mu)^2\right] = E\left[(a_k e)^2\right]$$

$$= E\left[\underbrace{e^T(x_k - \mu)}_{a_k e} \cdot e^T(x_k - \mu)\right] = E\left[e^T (x_k - \mu)(x_k - \mu)^T e\right]$$

$$= e^T \underbrace{\sum}_{\text{sigma covar matrix}} e \qquad \sum = \sum_k (x_k - \mu)(x_k - \mu)^T$$

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

sum of matrices

$d \times 1 \quad 1 \times d$

sigma

$$\sum = \begin{bmatrix} \sum_k (x_k^1 - \mu^1)(x_k^1 - \mu^1) & \cdots & \sum_k (x_k^1 - \mu^1)(x_k^d - \mu^d) \\ \vdots & & \\ \sum_k (x_k^d - \mu^d)(x_k^1 - \mu^1) & \cdots & \sum_k (x_k^d - \mu^d)(x_k^d - \mu^d) \end{bmatrix}$$

$\sigma_{ii} = \sum_k (x_k^i - \mu^i)(x_k^i - \mu^i)$ = var(feat i)

$\sigma_{ij} = \sum_k (x_k^i - \mu^i)(x_k^j - \mu^j) = \sigma_{ji}$

linear covel (feat i, feat j)

Constrained optimization pb

Σ = cov matrix = fixed

$$\max_{e} \; e^T \Sigma e \quad \text{OBJ}$$

constraint: subject to $\|e\| = 1 \Leftrightarrow e^T e = 1$

Lagrangian

LagMultip = ratio scalar of tangent differentials.

$$L = \max \left[\underbrace{e^T \Sigma e}_{\text{OBJ}} - \underbrace{\lambda (e^T e - 1)}_{\text{CONSTRAINED}} \right]$$

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

$$\Sigma e = \lambda e$$

sigma; λ scalar; same direction as e

e does not change direction when multip by Σ !

usually changes direction of the vector

$d \times d$ $\quad$ $d \times 1$

$M \cdot v$

matrix $\quad$ vector

$\Rightarrow e$ = eigenvector for Σ covar

λ : eigenval

$M \cdot v$ = direction of v $\Longleftrightarrow$ v = eigen vector for M

λ = eigen value

M = fixed

λv

λ = scalar

$\Sigma = \text{covar}(X)$ (1) symmetric $\Sigma_{ij} = \Sigma_{ji}$ very special

any $M = A \cdot A^T \Rightarrow$ (2) pos def $\forall v$ $v^T \Sigma v \geq 0$

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

allows spectral decomposition

e_i = eigen vectors of Σ normalized

$$\Sigma = \underbrace{\begin{bmatrix} | & | & & | \\ e_1 & e_2 & \cdots & e_d \\ | & | & & | \end{bmatrix}}_{E} \underbrace{\begin{bmatrix} \lambda_1 & 0 & & \\ 0 & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_d \end{bmatrix}}_{d \times d \text{ diag}(\text{eigen values})} \underbrace{\begin{bmatrix} - & e_1^T & - \\ - & e_2^T & - \\ & \vdots & \\ - & e_d^T & - \end{bmatrix}}_{E^T}$$

$d \times 1$ $d \times 1$ $d \times 1$

eigenvec eigenvec

$d \times d$

$\lambda_1 \geq \lambda_2 \geq \cdots \lambda_d \geq 0$

WANT t dim $\Rightarrow$ take (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 = diag-matrix of eigenvalues

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

- E is orthonormal: $\langle e_i \cdot e_i \rangle = \|e_i\| = 1$ and $\langle e_i e_j \rangle = 0 \ (i \neq j)$

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

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

Thus $E^T E = I_D = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix}_{D \times D} \Rightarrow E^T = E^{-1} \Rightarrow E \cdot E^T = I_D$

- $\Sigma = \Sigma E E^T = \Sigma \begin{bmatrix} | & | & & | \\ e_1 & e_2 & \cdots & e_D \\ | & | & & | \end{bmatrix} \begin{bmatrix} — & e_1 & — \\ — & e_2 & — \\ & \vdots & \\ — & e_D & — \end{bmatrix} = \begin{bmatrix} | & | & & | \\ \Sigma e_1 & \Sigma e_2 & \cdots & \Sigma e_D \\ | & | & & | \end{bmatrix}$

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

$$= \begin{bmatrix} | & | & & | \\ e_1 & e_2 & \cdots & e_D \\ | & | & & | \end{bmatrix} \times \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & 0 \\ & & \ddots & \\ & 0 & & \lambda_D \end{bmatrix} \times \begin{bmatrix} — & e_1 & — \\ — & e_2 & — \\ & \vdots & \\ — & e_D & — \end{bmatrix} = E \cdot D \cdot E^T$$

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

$$\sum_{d\times d} \overset{\text{approx}}{\simeq} \sum_{t} = \underbrace{\begin{bmatrix} | & | & & | \\ e_1 & e_2 & \cdots & e_t \\ | & | & & | \end{bmatrix}}_{d\times t} \times \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \ddots \\ & & & \lambda_t \end{bmatrix}_{t\times t} \begin{bmatrix} - & e_1^T & - \\ - & e_2^T & - \\ & \vdots & \\ - & e_t^T & - \end{bmatrix}_{t\times d}$$

C_t

$d = 765$

$t = 20$

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

new representation of data $(X)_{n\times d}$ on t dim $n\times$

datapoint $\bar{x}_{1\times d} \longrightarrow \bar{x}\cdot C_r$

$1\times d \quad d\times r \Rightarrow 1\times r$

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

$\bar{x}_{\text{new}}$ = r-dimensions $= \left[\bar{x}\cdot e_1, \bar{x}\cdot e_2, \ldots, \bar{x}\cdot e_r\right]$

$\bar{x}$ projected on $e_1, e_2, \ldots e_r$

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

duality / KKT conditions

$$e_j = \sum_{i=1}^{N} \vec{\beta}^j x_i = \sum_{i=1}^{N} \beta_i^j \cdot x_i$$

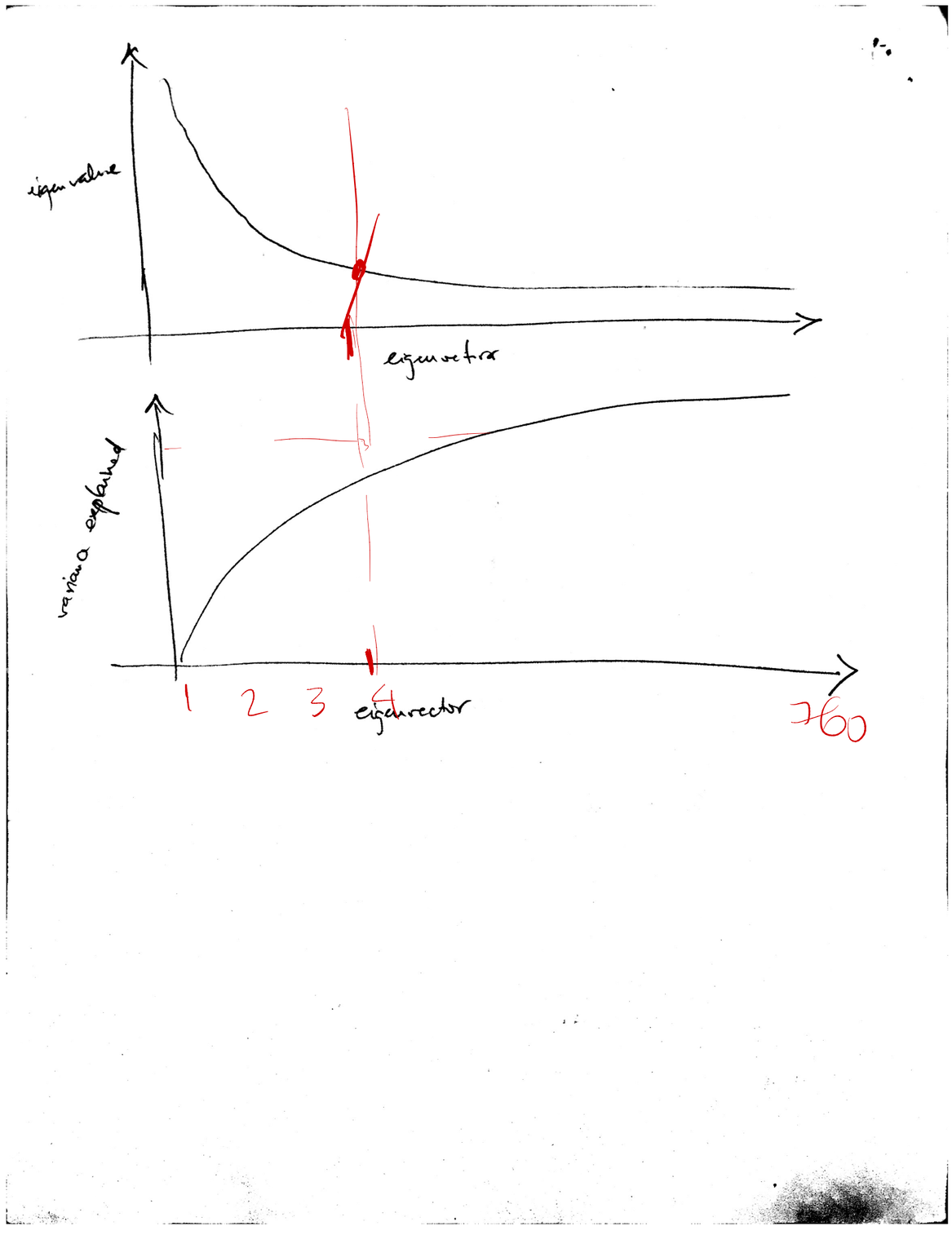
$\vec{\beta}^j = (\beta_1^j, \beta_2^j \ldots \beta_N^j)$ dual variable

new rep(x) $= \bar{x}_{orig} \cdot [$ eigen vectors in dual form $]$

$$\sim \sum B_j \cdot \bar{x} \sum_{i=1}^{N} x_i$$

$$\bar{x} = \sum \beta_j \sum_i \bar{x} x_i \rightarrow sim(\bar{x}, x_i)$$

new rep(x) = sim vector with other points


eigenvalue
eigenvector
variance explained
1
2
3
eigenvector
760

KERNEL PCA

- assume data centered on each column. $\mu_x = 0$

- $K = X \cdot X^T$ kernel matrix ($N \times N$)
- $K_{ij} = \langle x_i, x_j \rangle = x_i \cdot x_j^T = sim(x_i, x_j)$
- K also symmetric, pos-definite

- Σ (sigma, $d \times d$) $= covar(X) = X^T X \cdot \frac{1}{N}$
$= \frac{1}{N} \sum_{i=1}^{N} x_i^T \cdot x_i$ ($d \times 1$, $1 \times d$)

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

$$\Rightarrow e = \frac{1}{\lambda N} \sum_{i=1}^{N} x_i^T x_i e = \sum_{i=1}^{N} \frac{1}{\lambda N} x_i^T \underbrace{(x_i e)}_{1 \times 1} = \sum_{i=1}^{N} \underbrace{\frac{x_i e}{\lambda N}}_{\beta_i} \cdot x_i^T = \sum_{i=1}^{N} \beta_i x_i^T$$

($x_i e$: $1 \times d$)

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

$$e = X^T \cdot \vec{\beta}^e = \sum_i \left(\beta_1 x_1, \beta_2 x_2 \cdots \beta_N x_N\right)$$

- $\vec{\beta}^e = \begin{bmatrix} \beta_1^e \\ \beta_2^e \\ \vdots \\ \beta_N^e \end{bmatrix}$ ($N \times 1$) are dual variables for e

- PCA representation on new dimension e for datapoint x_j is

$$x_j \cdot e = x_j \sum_{i=1}^{N} \beta_i^e x_i^T = \sum_{i=1}^{N} \beta_i^e \cdot x_j x_i^T = \sum_{i=1}^{N} \beta_i^e K_{ij} = \sum_{i=1}^{N} \beta_i^e \cdot sim(x_i, x_j)$$

- for all datapoints X

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

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

$$X \cdot \begin{bmatrix} | & | & & | \\ e_1 & e_2 & \cdots & e_r \\ | & | & & | \end{bmatrix} = X \cdot X^T \cdot \begin{bmatrix} \beta_1^{e_1} & \beta_1^{e_2} & \cdots & \beta_1^{e_r} \\ \beta_2^{e_1} & \beta_2^{e_2} & & \beta_2^{e_r} \\ \vdots & \vdots & & \vdots \\ \beta_N^{e_1} & \beta_N^{e_2} & & \beta_N^{e_r} \end{bmatrix} = \underset{N \times N}{K} \cdot \underbrace{\begin{bmatrix} | & | & & | \\ \beta^{e_1} & \beta^{e_2} & \cdots & \beta^{e_r} \\ | & | & & | \end{bmatrix}}_{N \times r}$$

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

$N \times N$ $N \times 1$

Calculate β_i^e dual variables for eigenvector (e, λ) of Σ

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

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

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

multiply by X to left

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

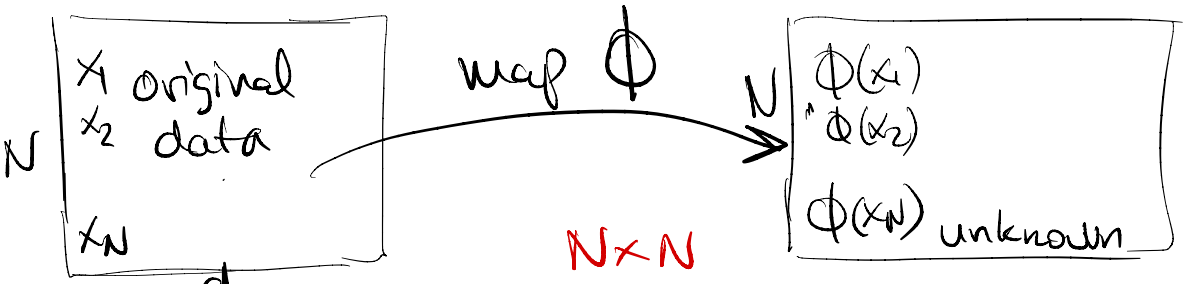
$$K \cdot K \cdot \bar{\beta} = N \cdot \lambda \cdot (X \cdot X^T) \cdot \bar{\beta}$$

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

can "divide" by K for all $\bar{\beta}$ with eigenvals $\neq 0$ $\Rightarrow$

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

KERNEL TRICK



K can be the dot product $sim(x_i, x_j)$ but in a new mapped space $\phi(x)$: $K_{ij} = \langle \phi(x_i) \cdot \phi(x_j) \rangle$ $N \times N$

- this is saying K is still $X \cdot X^T$ like before, but data X is now $\phi(x)$
- $K = XX^T$ 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 ϕ so that $K = \phi(x)\,\phi(x)^T$ — we don't know ϕ explicit
- the theory we developed for PCA with $K = XX^T$ work for any other kernel K
- β = eigenvectors(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} = function (x_i, x_j) assuming $K = \phi(x) \cdot \phi(x)^T$
- but without explicit knowing ϕ

CENTERING DATA inside K

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

- if $\mu \neq 0$ we need to update K to watch our theory.
- $K = \phi(x)\cdot\phi^T(x)$; want new $\bar{K} = (\phi(x)-\mu)(\phi(x)-\mu)^T$
- $\phi_i = \phi(x_i) =$ 1x unknown wapped vector(x_i
- $\bar{K}_{ij} = (\phi_i - \mu)\times(\phi_j - \mu)^T$

$$= \left(\phi_i - \frac{1}{N}\sum_{t=1}^{N}\phi_t\right)\times\left(\phi_j - \frac{1}{N}\sum_{e=1}^{N}\phi_e\right)^T =$$

$$= \phi_i\times\phi_j - \frac{1}{N}\sum_{t=1}^{N}\phi_t\times\phi_j^T - \frac{1}{N}\sum_{e=1}^{N}\phi_i\times\phi_e^T + \frac{1}{N^2}\sum_{et}\phi_t\times\phi_e^T$$

$$= K_{ij} - \frac{1}{N}\sum_{t}K_{tj} - \frac{1}{N}\sum_{e}K_{ie} + \frac{1}{N^2}\sum_{et}K_{te}$$

- matrix form update for $K \to \bar{K}$

$$\bar{K} = K - UK - KU - UKU$$

$$U_{N\times N} = \frac{1}{N}\cdot\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}_{N\times N}$$

Lagrange multipliers, examples

Examples of the Lagrangian and Lagrange multiplier technique in action.

Background

- Introduction to Lagrange multipliers
- Gradient

Lagrange multiplier technique, quick recap

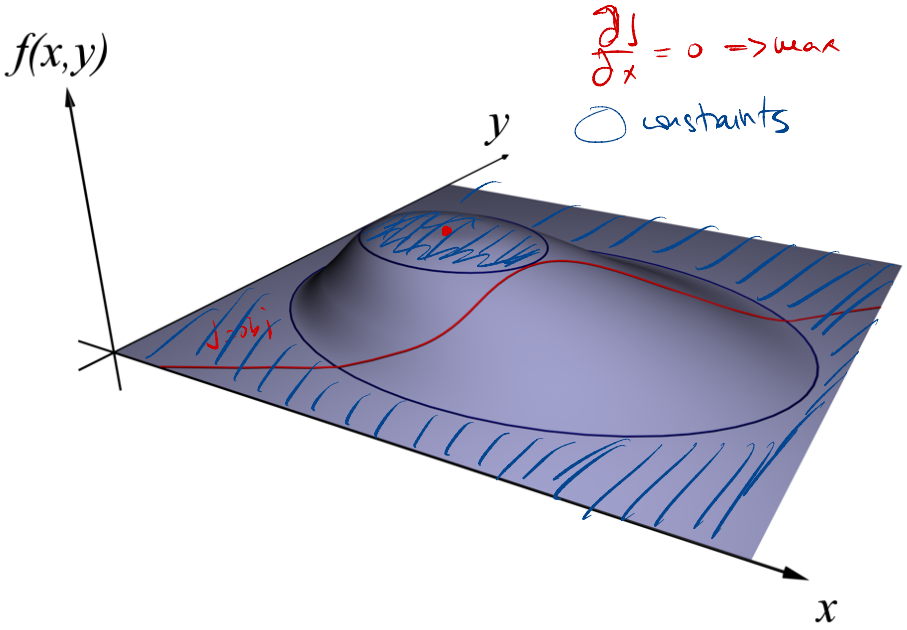


Image credit: By Nexcis (Own work) [Public domain], via Wikimedia Commons

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

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

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

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

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

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

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

- **Step 3**: Consider each solution, which will look something like $(x_0, y_0, \ldots, \lambda_0)$. Plug each one into f. Or rather, first remove the λ_0 component, then plug it into f, since f does not have λ as an input. Whichever one gives the greatest (or smallest) value is the maximum (or minimum) point your 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

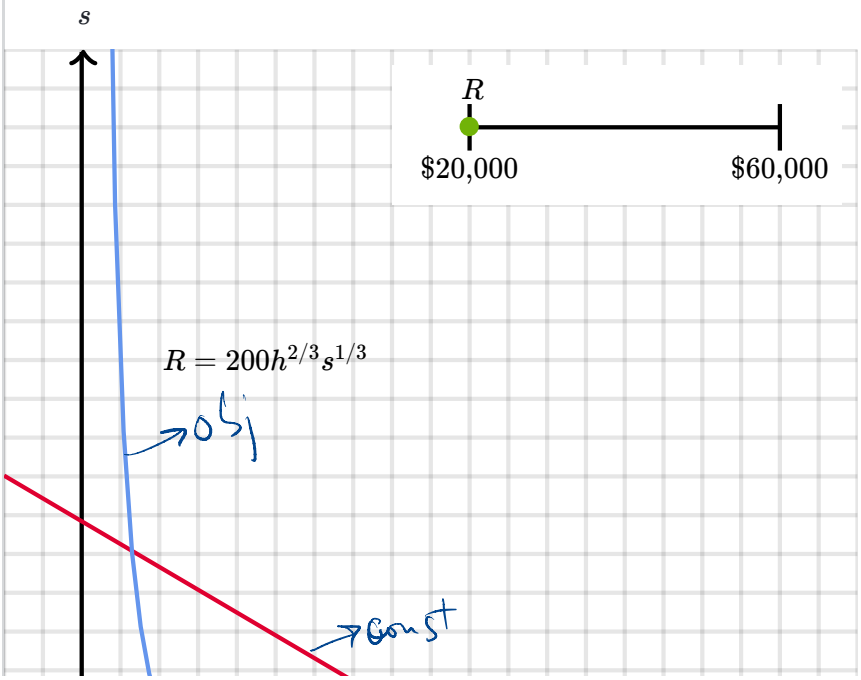
The $\$20$ per hour labor costs and $\$170$ per ton steel costs tell us that the total cost of production, in terms of h and s, is

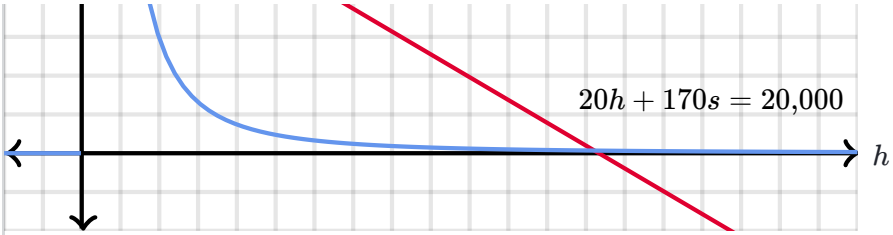
$$20h + 170s$$

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.

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

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

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

Finally we set the partial derivative with respect to λ 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.

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

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

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

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

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

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

The interpretation of this constant $\lambda = 2.593$ is left to the next article

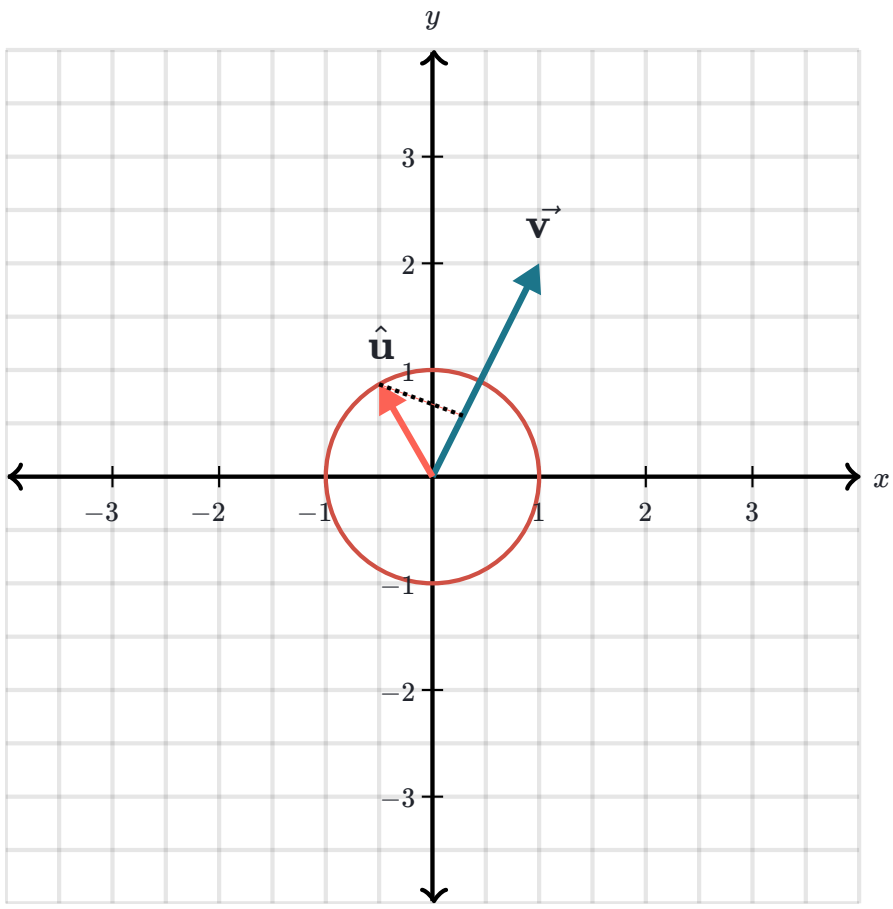
Example 2: Maximizing dot product

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 $\hat{\mathbf{u}} \cdot \vec{\mathbf{v}}$?

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

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

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 λ equal to 0 just restates the constraint.

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

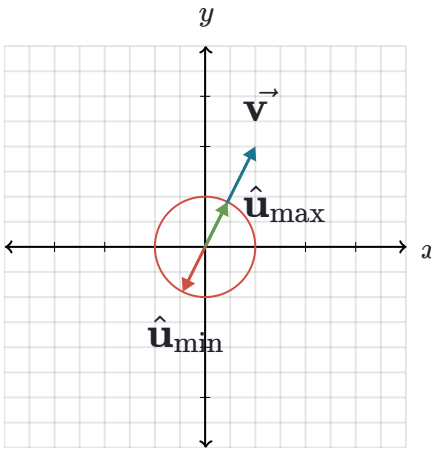
Solving for x, y and z in the first three equations above, we get

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

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 $\vec{\mathbf{v}}$. 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 $\hat{\mathbf{u}}$ and $\vec{\mathbf{v}}$ as follows:

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

Therefore $\hat{\mathbf{u}}$ is proportional to $\vec{\mathbf{v}}$! Geometrically, this means $\hat{\mathbf{u}}$ points in the same direction as $\vec{\mathbf{v}}$. There are two unit vectors proportional $\vec{\mathbf{v}}$,



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

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

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

$$\hat{\mathbf{u}}_{\text{max}} = \frac{\vec{\mathbf{v}}}{||\vec{\mathbf{v}}||}$$

$$\hat{\mathbf{u}}_{\text{min}} = -\frac{\vec{\mathbf{v}}}{||\vec{\mathbf{v}}||}$$

The magnitude $||\vec{\mathbf{v}}||$ is $\sqrt{2^2 + 3^2 + 1^2} = \sqrt{14}$, so we can write the maximizing unit vector $\hat{\mathbf{u}}_{\text{max}}$ 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<=2 which r all the points in the unit circle

including the boundary.

1 vote • Comment • Flag

9 months ago by hamadmo77

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 ›

UNSUPERVISED LEARNING 2011

LECTURE :KERNEL PCA

Rita Osadchy

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

Dimensionality Reduction

- Data representation

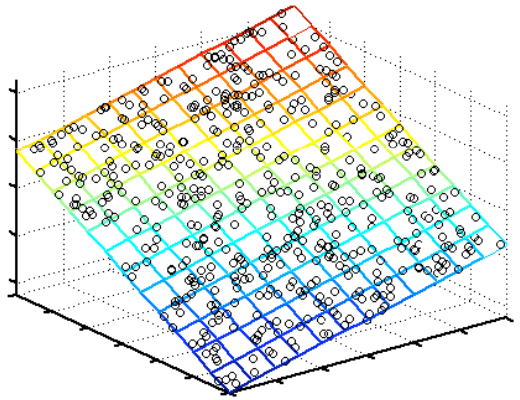
  Inputs are real-valued vectors in a high dimensional space.

- Linear structure

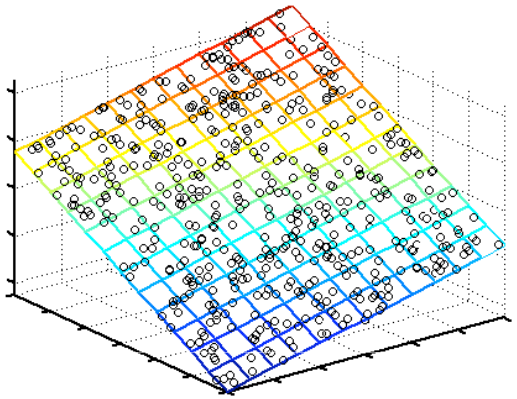
  Does the data live in a low dimensional subspace?

- 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**)

  $\boldsymbol{x_1, x_2, \ldots, x_n}$ points in R^D

- Outputs **(low dimensional)**

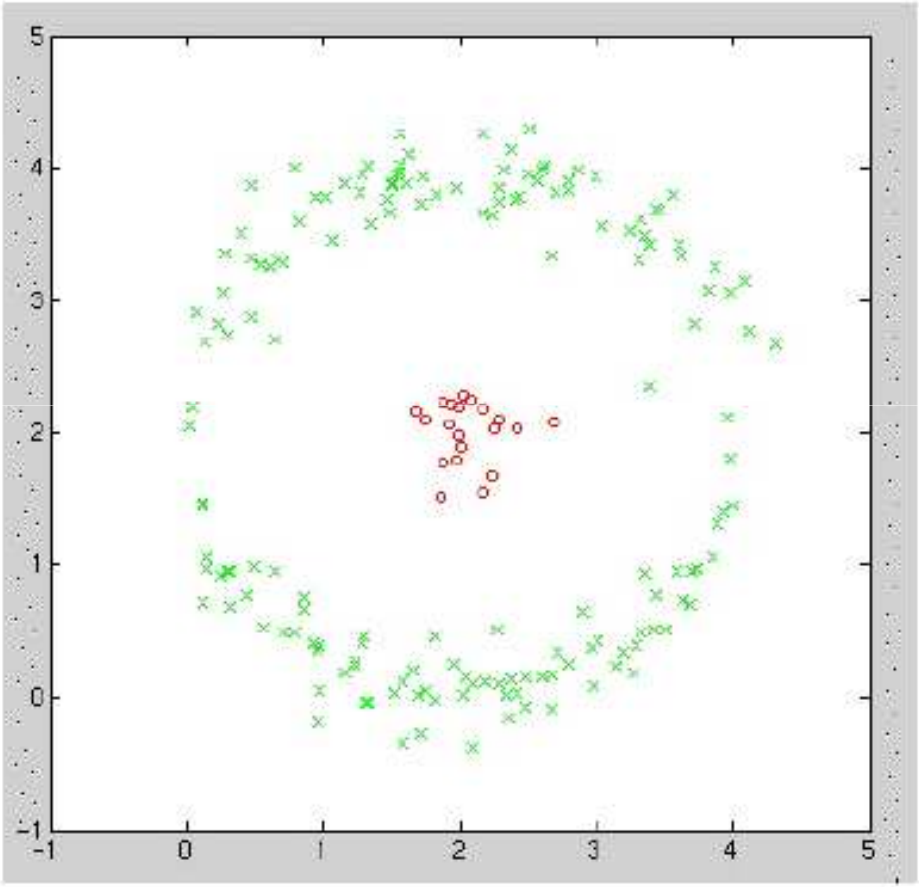
  $\boldsymbol{y_1, y_2, \ldots, y_n}$ points in R^d ($d \ll D$)

The “magic” of high dimensions

- Given some problem, how do we know what classes of functions are capable of solving that problem?

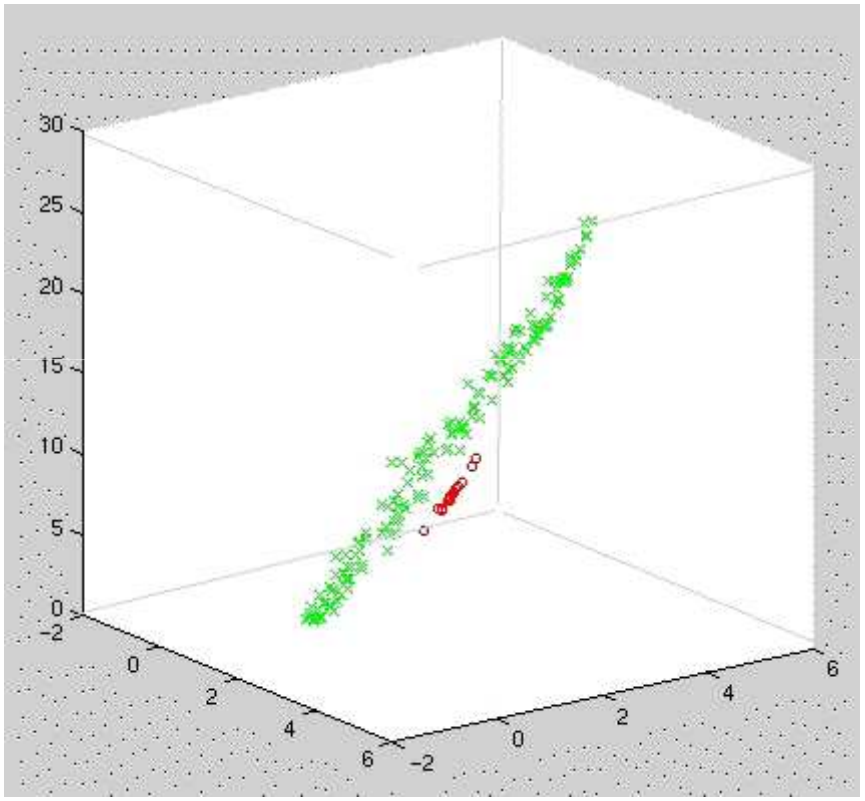
- VC (Vapnik-Chervonenkis) theory tells us that often mappings which take us into a higher dimensional space than the dimension of the input space provide us with greater classification power.

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

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

$$(x_1, x_2) \mapsto (x_1, x_2, x_1^2 + x_2^2)$$

Kernel trick : don't need ϕ

just $K(x_i, x_j)$

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

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

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

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:

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

- Eigenvectors

$$\mathrm{C}v = \lambda v$$

Derivation Cont.

- Eigenvectors can be expressed as linear combination of features:

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

- Proof:

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

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 $xx^T v = (x \cdot v)x^T$

$$(\boldsymbol{x}\boldsymbol{x}^T)\boldsymbol{v} = \begin{pmatrix} x_1x_1 & x_1x_2 & \dots & x_1x_M \\ x_2x_1 & x_2x_2 & \dots & x_2x_M \\ \vdots & \vdots & \ddots & \vdots \\ x_Mx_1 & x_Mx_2 & \dots & x_Mx_M \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_M \end{pmatrix}$$

$$= \begin{pmatrix} x_1x_1v_1 + x_1x_2v_2 + \ldots + x_1x_Mv_M \\ x_2x_1v_1 + x_2x_2v_2 + \ldots + x_2x_Mv_M \\ \vdots \\ x_Mx_1v_1 + x_Mx_2v_2 + \ldots + x_Mx_Mv_M \end{pmatrix}$$

Showing that $xx^T v = (x \cdot v)x^T$

$$= \begin{pmatrix} (x_1v_1 + x_2v_2 + \ldots + x_Mv_M)\, x_1 \\ (x_1v_1 + x_2v_2 + \ldots + x_Mv_M)\, x_2 \\ \vdots \\ (x_1v_1 + x_2v_2 + \ldots + x_Mv_M)\, x_M \end{pmatrix}$$

$$= \begin{pmatrix} x_1v_1 + x_2v_2 + \ldots + x_Mv_M \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{pmatrix}$$

$$= (\boldsymbol{x} \cdot \boldsymbol{v})\boldsymbol{x}$$

□

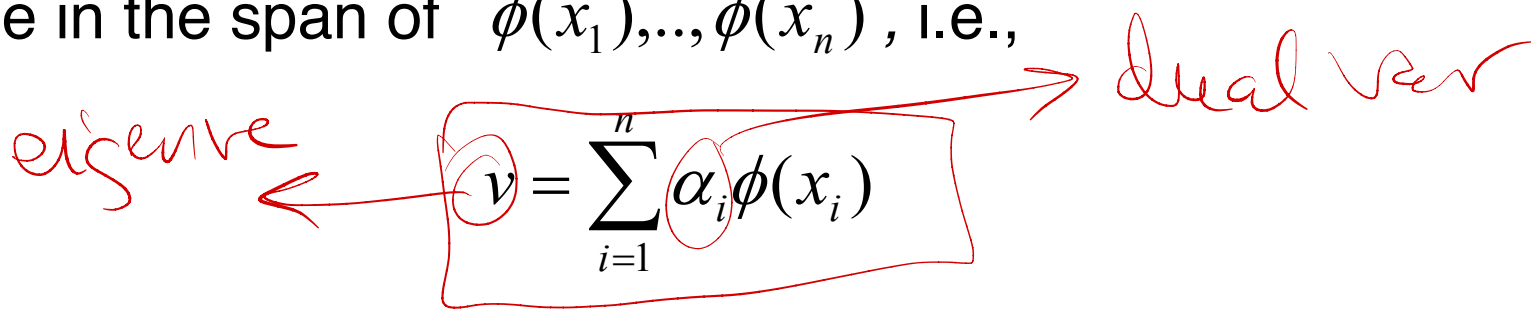
Derivation Cont.

- 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 $\phi(x_1),..,\phi(x_n)$, i.e.,



- Finding the eigenvectors is equivalent to finding the coefficients α_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:

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

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

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

- We have a normalization condition for the α_j vectors:

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

Derivation Cont.

- By multiplying $\mathrm{K}\alpha_j = n\lambda_j\alpha_j$ by α_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:

$$\tilde{K}(x_i, x_j) = \tilde{\phi}(x_i)^T \tilde{\phi}(x_j)$$

$$= \left(\phi(x_i) - \frac{1}{n}\sum_{k=1}^{n}\phi(x_k)\right)^T \left(\phi(x_j) - \frac{1}{n}\sum_{k=1}^{n}\phi(x_k)\right)$$

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

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

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

- where $\mathbf{1}_{1/\mathrm{n}}$ 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 x m):

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

- Solve an eigenvalue problem:

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

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

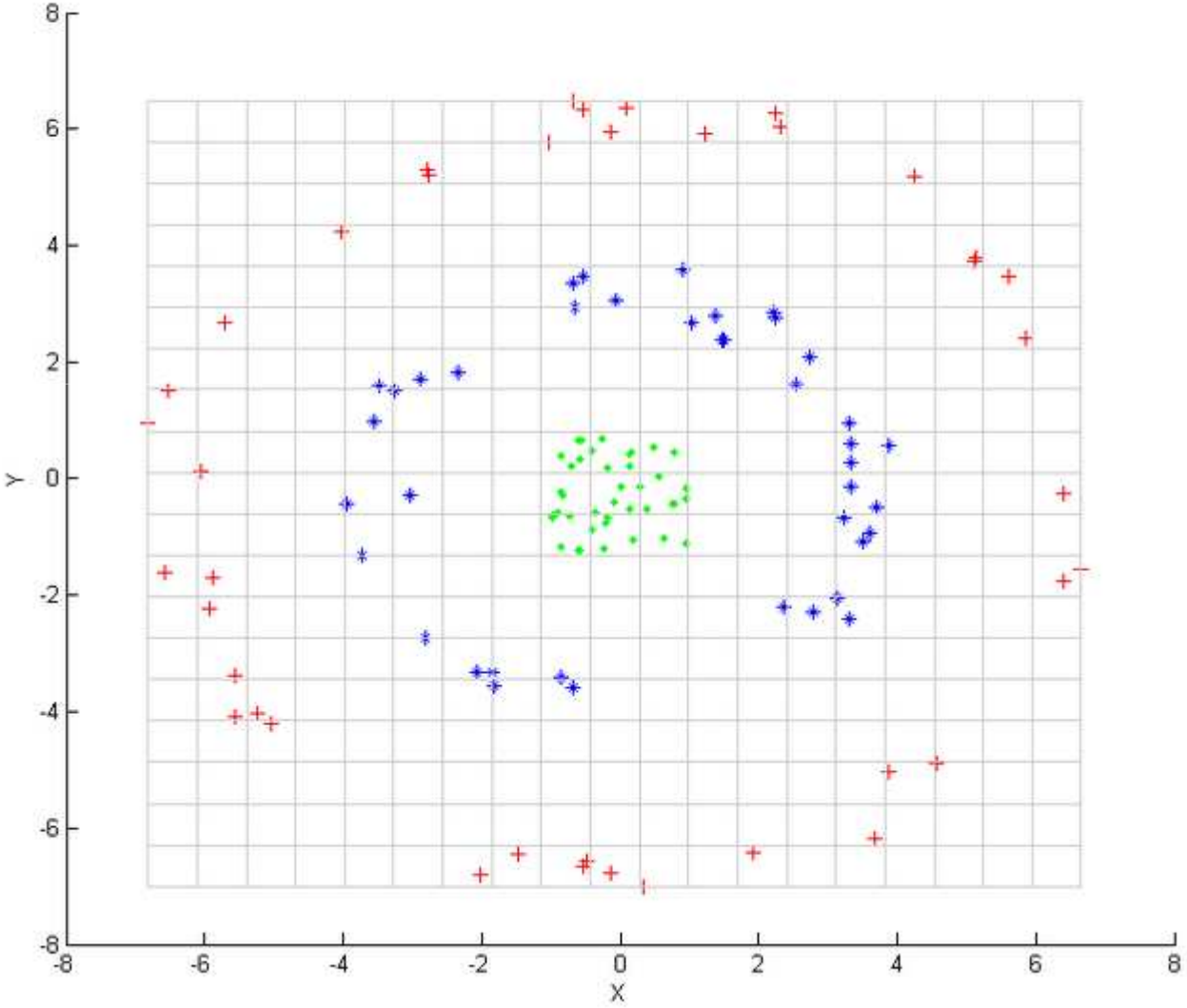
$$y_j = \sum_{i=1}^{n} \alpha_{ji} K(x, x_i), \quad \mathrm{j} = 1,..,d$$

x = any point

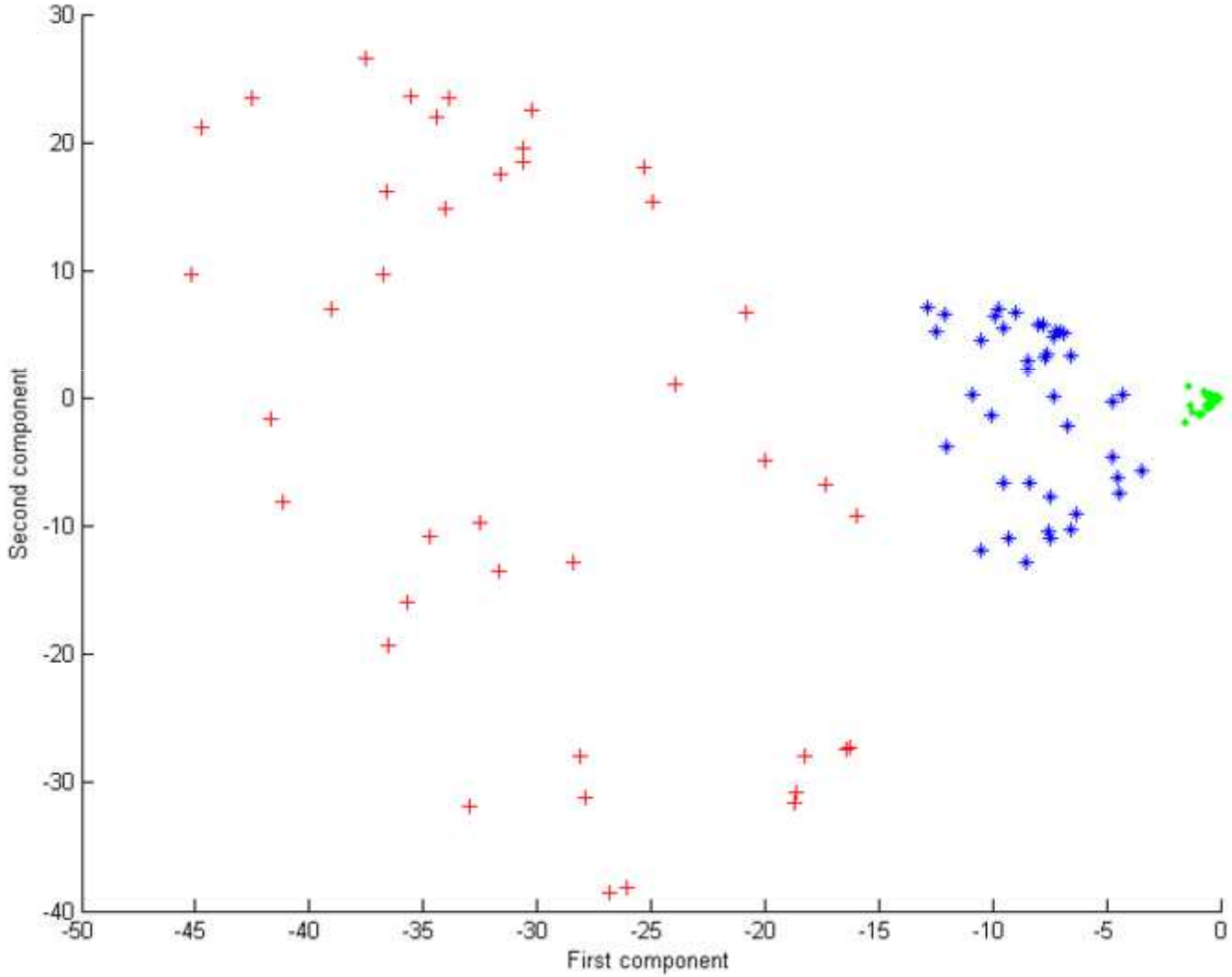
new rep

dual feat = sim(x, 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 x n, so kernel PCA will have difficulties if we have lots of data points.