

• 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 e
 line e

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

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

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

Data points (vectors)

	f_1	f_2	f_d
x_1			
x_2			
x_3			
\vdots			
x_N			

new representation on t features

A: $t \ll d$

$N \times d$

$N \times t$

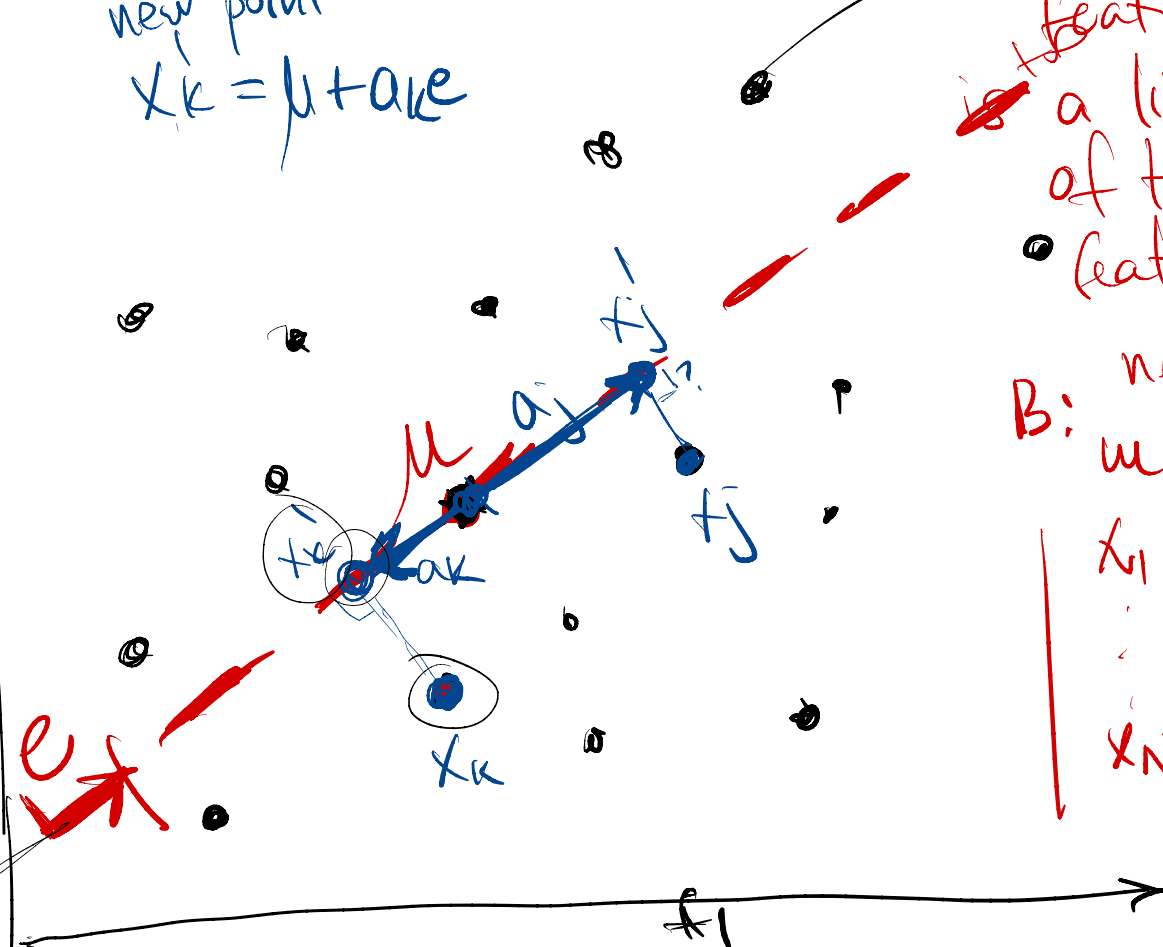
each new feature g_i is a linear comb. of the original features.

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

$N \times$ orig

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

guess: that "one line" new rep

- passes through μ .

- usually correspond to longest-direction stretch of data.

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

SSE / sq loss

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

new point

orig point

$x_k = \mu + a_k e$

$x_k - \mu = a_k e$

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

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

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

projection

new data mean $(a_1, a_2, \dots, a_N) = E[\mu + a_k e] = \mu + E[a_k e] = \mu + e^T E[x_k - \mu]$
 $E[x_k - \mu] = 0 \Rightarrow = \mu$

$J(e)$ 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 Σ

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

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

sigma

minimise $J(e)$

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

max variance (projections)

Most important: what is the best e ? want MAX

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

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

$$= E[e^T (x_k - \mu) \cdot e^T (x_k - \mu)] = E[e^T (x_k - \mu) (x_k - \mu)^T e]$$

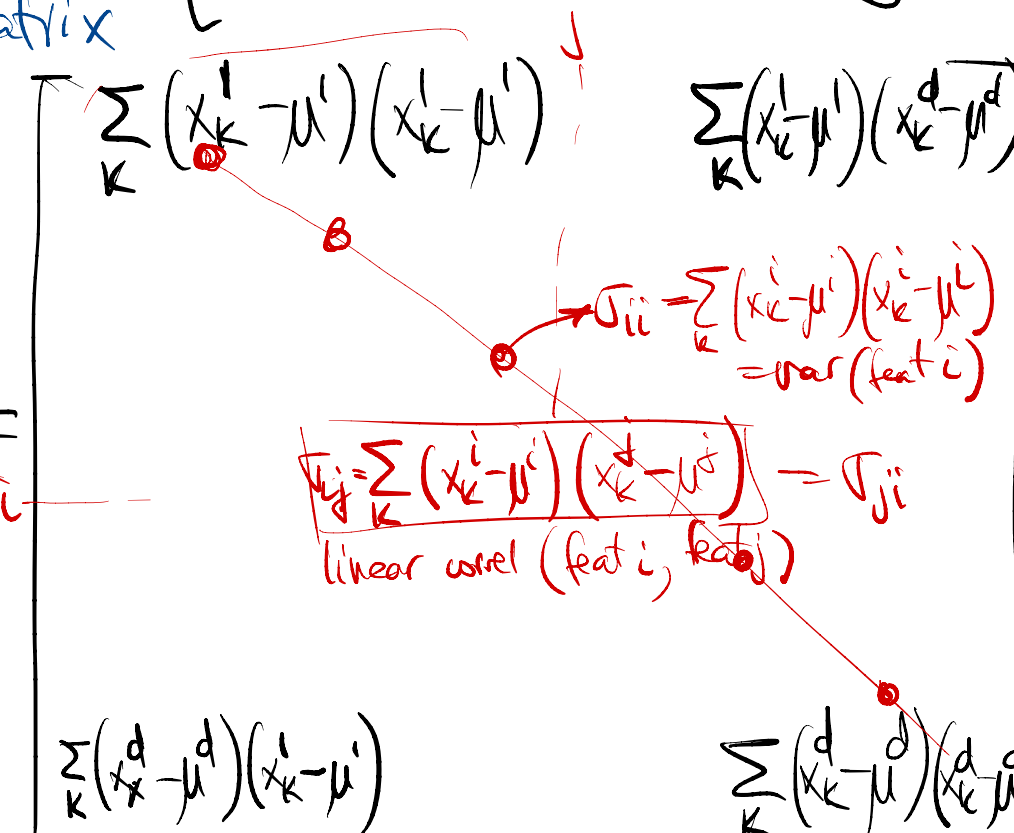
sigma covar matrix

$$= e^T \sum_k e$$

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

$$\sum_{\text{sigma}} = \sum_{\text{sum of matrices}} (x_k - \mu) (x_k - \mu)^T$$

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



Constrained optimization pb

$\Sigma =$ ~~cov~~ matrix = fixed

maximize $e^T \Sigma e$ OBS

constraint

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

Lag Multiplier = ratio of tangent differentials.

Lagrangian

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

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

same direction as e

$\Sigma e = \lambda e$

sigma

scalar

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

usually changes direction of the vector.

$\begin{matrix} dx \times d & dx \times 1 \\ M & \bullet & v \\ \text{matrix} & & \text{vector} \end{matrix}$

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



[< 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

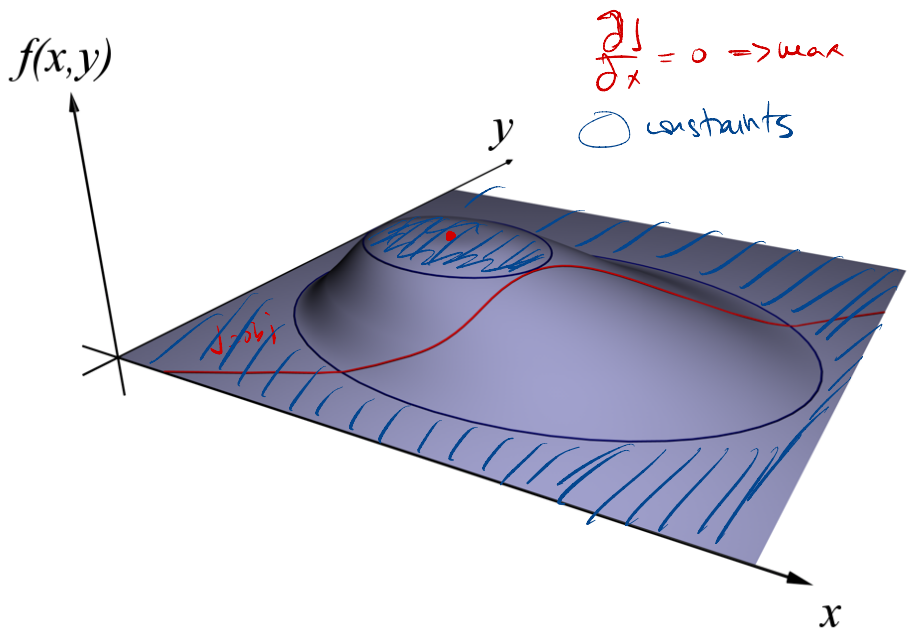


Image credit: By Nexcis (Own work) [Public domain], via [Wikimedia Commons](#)

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 λ , 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 λ is referred to as a "Lagrange multiplier"

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

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

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 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 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

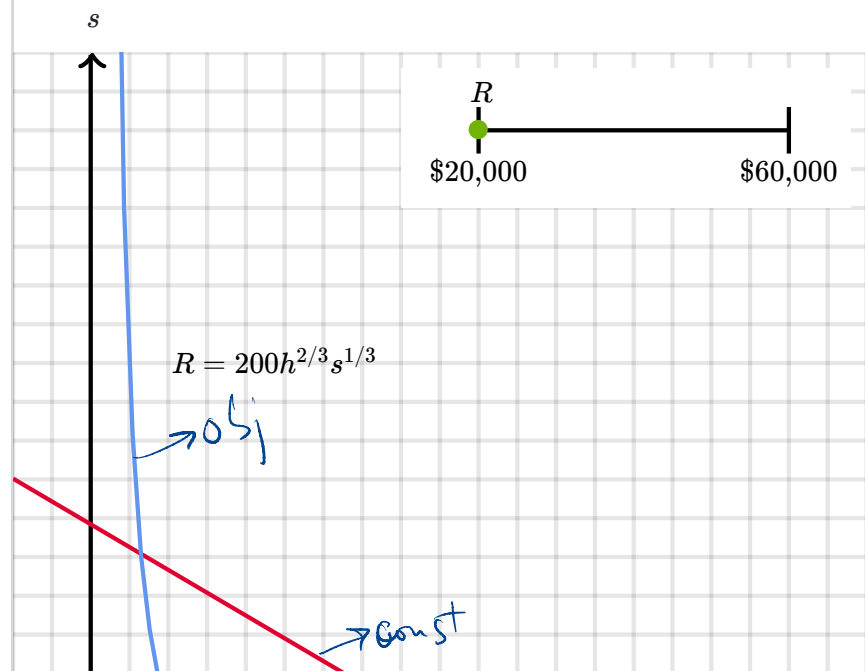
The \$20 per hour labor costs and \$170 per ton steel costs tell us that the total cost of production, in terms of h and s , is

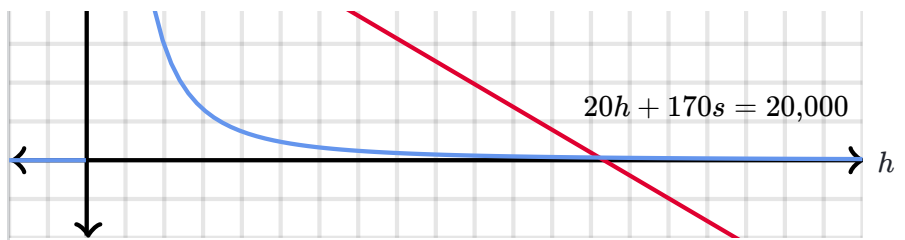
$$20h + 170s$$

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 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.

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

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

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

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

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](#)

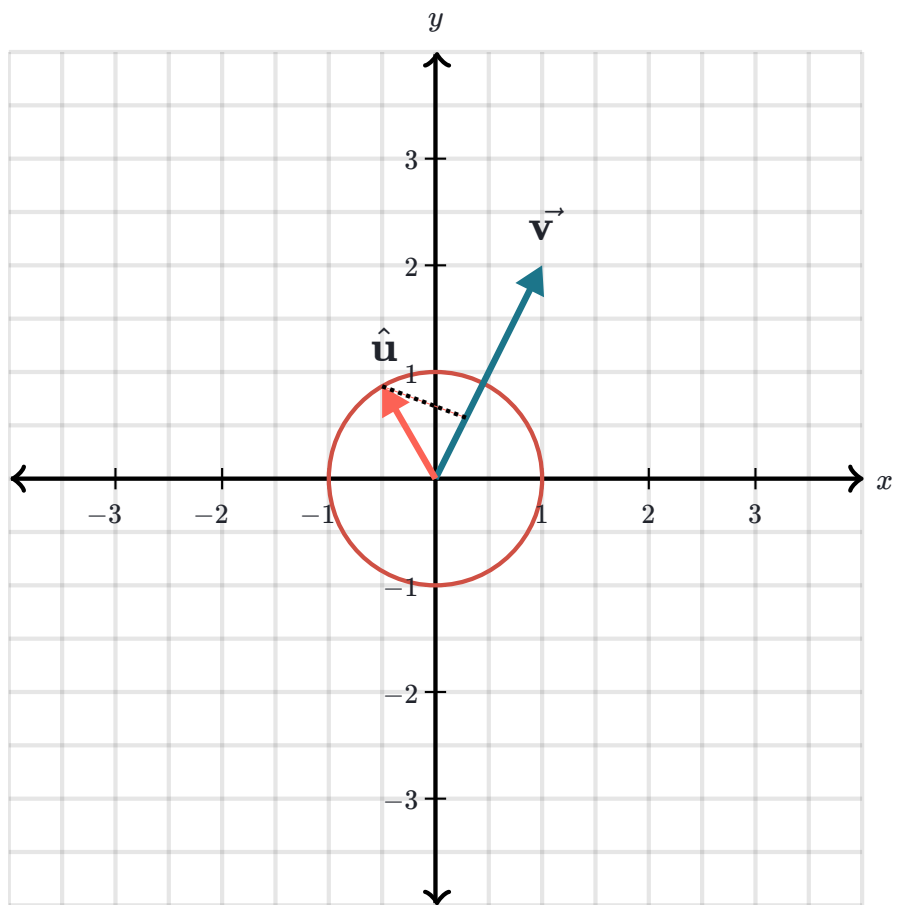
Example 2: Maximizing dot product

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 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

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

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

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

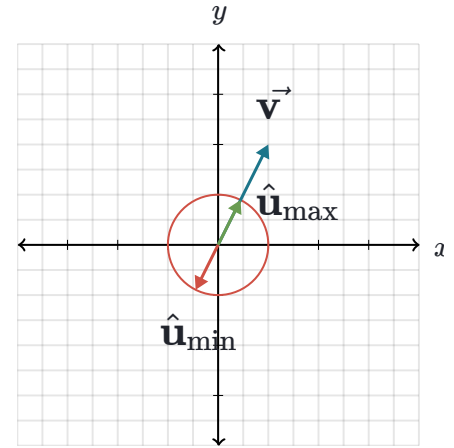
Ah, what beautiful symmetry. Each of these expressions has the same $\frac{1}{2\lambda}$ factor, and the coefficients 2, 3 and 1 match up with the coordinates of \vec{v} . Being good math students as we are, we won't let good symmetry go to waste. In this case, combining the three equations above into 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  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 ▶](#)

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](#)



$M \cdot v = \text{direction of } v \iff v - \text{eigen vector for } M$

$M = \text{fixed}$

λv
↓
scalar

$\lambda = \text{eigen value}$

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

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

very special

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

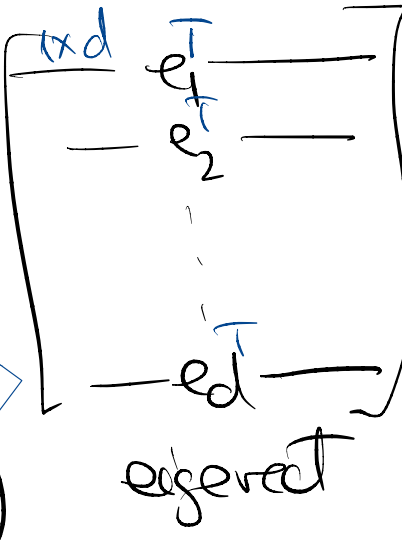
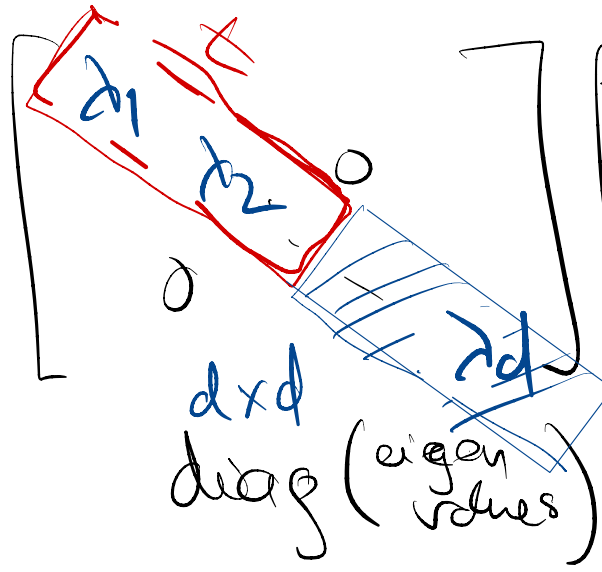
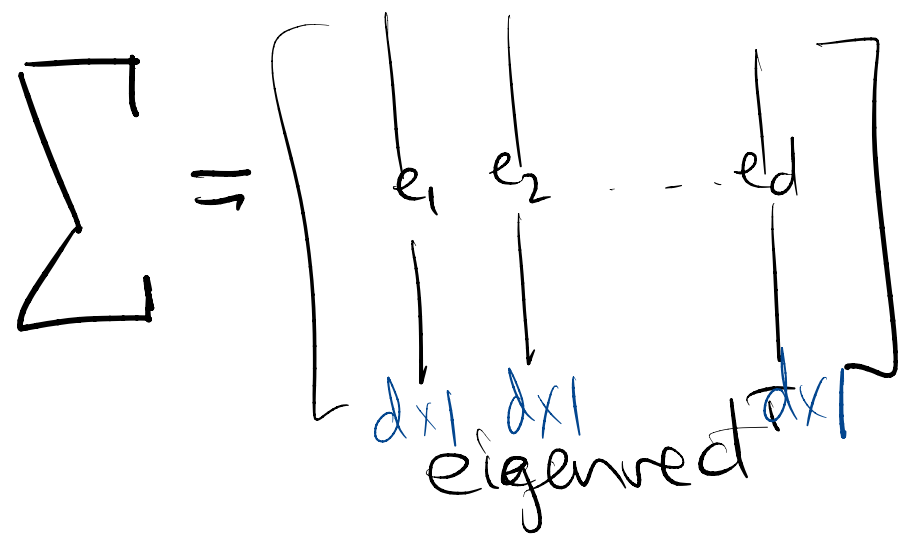
(2) pos def

$$v^T \Sigma v \geq 0$$

$v^T(AA^T)v = (Av)(Av)^T \geq 0$

allows spectral decomposition

$e_i =$ eigen vectors of Σ
normalized



$$\alpha_1 \geq \alpha_2 \geq \dots \geq \alpha_d \geq 0$$

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

3.8.1 Principal Component Analysis (PCA)

We begin by considering the problem of representing all of the vectors in a set of n d -dimensional samples $\mathbf{x}_1, \dots, \mathbf{x}_n$ by a single vector \mathbf{x}_0 . To be more specific, suppose that we want to find a vector \mathbf{x}_0 such that the sum of the squared distances between \mathbf{x}_0 and the various \mathbf{x}_k is as small as possible. We define the squared-error criterion function $J_0(\mathbf{x}_0)$ by

$$J_0(\mathbf{x}_0) = \sum_{k=1}^n \|\mathbf{x}_0 - \mathbf{x}_k\|^2, \quad (78)$$

and seek the value of \mathbf{x}_0 that minimizes J_0 . It is simple to show that the solution to this problem is given by $\mathbf{x}_0 = \mathbf{m}$, where \mathbf{m} is the sample mean,

$$\mathbf{m} = \frac{1}{n} \sum_{k=1}^n \mathbf{x}_k. \quad (79)$$

This can be easily verified by writing

$$\begin{aligned} J_0(\mathbf{x}_0) &= \sum_{k=1}^n \|(\mathbf{x}_0 - \mathbf{m}) - (\mathbf{x}_k - \mathbf{m})\|^2 \\ &= \sum_{k=1}^n \|\mathbf{x}_0 - \mathbf{m}\|^2 - 2 \sum_{k=1}^n (\mathbf{x}_0 - \mathbf{m})^t (\mathbf{x}_k - \mathbf{m}) + \sum_{k=1}^n \|\mathbf{x}_k - \mathbf{m}\|^2 \\ &= \sum_{k=1}^n \|\mathbf{x}_0 - \mathbf{m}\|^2 - 2(\mathbf{x}_0 - \mathbf{m})^t \sum_{k=1}^n (\mathbf{x}_k - \mathbf{m}) + \sum_{k=1}^n \|\mathbf{x}_k - \mathbf{m}\|^2 \\ &= \sum_{k=1}^n \|\mathbf{x}_0 - \mathbf{m}\|^2 + \underbrace{\sum_{k=1}^n \|\mathbf{x}_k - \mathbf{m}\|^2}_{\text{independent of } \mathbf{x}_0}. \end{aligned} \quad (80)$$

Since the second sum is independent of \mathbf{x}_0 , this expression is obviously minimized by the choice $\mathbf{x}_0 = \mathbf{m}$.

The sample mean is a zero-dimensional representation of the data set. It is simple, but it does not reveal any of the variability in the data. We can obtain a more interesting, one-dimensional representation by projecting the data onto a line running through the sample mean. Let \mathbf{e} be a unit vector in the direction of the line. Then the equation of the line can be written as

$$\mathbf{x} = \mathbf{m} + a\mathbf{e}, \quad (81)$$

where the scalar a (which takes on any real value) corresponds to the distance of any point \mathbf{x} from the mean \mathbf{m} . If we represent \mathbf{x}_k by $\mathbf{m} + a_k\mathbf{e}$, we can find an "optimal" set of coefficients a_k by minimizing the squared-error criterion function

$$\begin{aligned} J_1(a_1, \dots, a_n, \mathbf{e}) &= \sum_{k=1}^n \|(\mathbf{m} + a_k\mathbf{e}) - \mathbf{x}_k\|^2 = \sum_{k=1}^n \|a_k\mathbf{e} - (\mathbf{x}_k - \mathbf{m})\|^2 \\ &= \sum_{k=1}^n a_k^2 \|\mathbf{e}\|^2 - 2 \sum_{k=1}^n a_k \mathbf{e}^t (\mathbf{x}_k - \mathbf{m}) + \sum_{k=1}^n \|\mathbf{x}_k - \mathbf{m}\|^2. \end{aligned} \quad (82)$$

Recognizing that $\|\mathbf{e}\| = 1$, partially differentiating with respect to a_k , and setting the derivative to zero, we obtain

$$a_k = \mathbf{e}'(\mathbf{x}_k - \mathbf{m}). \quad (83)$$

Geometrically, this result merely says that we obtain a least-squares solution by projecting the vector \mathbf{x}_k onto the line in the direction of \mathbf{e} that passes through the sample mean.

SCATTER MATRIX

This brings us to the more interesting problem of finding the *best* direction \mathbf{e} for the line. The solution to this problem involves the so-called *scatter matrix* \mathbf{S} defined by

$$\mathbf{S} = \sum_{k=1}^n (\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})'. \quad (84)$$

The scatter matrix should look familiar—it is merely $n - 1$ times the sample covariance matrix. It arises here when we substitute a_k found in Eq. 83 into Eq. 82 to obtain

$$\begin{aligned} J_1(\mathbf{e}) &= \sum_{k=1}^n a_k^2 - 2 \sum_{k=1}^n a_k^2 + \sum_{k=1}^n \|\mathbf{x}_k - \mathbf{m}\|^2 \\ &= - \sum_{k=1}^n [\mathbf{e}'(\mathbf{x}_k - \mathbf{m})]^2 + \sum_{k=1}^n \|\mathbf{x}_k - \mathbf{m}\|^2 \\ &= - \sum_{k=1}^n \mathbf{e}'(\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})'\mathbf{e} + \sum_{k=1}^n \|\mathbf{x}_k - \mathbf{m}\|^2 \\ &= -\mathbf{e}'\mathbf{S}\mathbf{e} + \sum_{k=1}^n \|\mathbf{x}_k - \mathbf{m}\|^2. \end{aligned} \quad (85)$$

Clearly, the vector \mathbf{e} that minimizes J_1 also maximizes $\mathbf{e}'\mathbf{S}\mathbf{e}$. We use the method of Lagrange multipliers (described in Section A.3 of the Appendix) to maximize $\mathbf{e}'\mathbf{S}\mathbf{e}$ subject to the constraint that $\|\mathbf{e}\| = 1$. Letting λ be the undetermined multiplier, we differentiate

$$u = \mathbf{e}'\mathbf{S}\mathbf{e} - \lambda(\mathbf{e}'\mathbf{e} - 1) \quad (86)$$

with respect to \mathbf{e} to obtain

$$\frac{\partial u}{\partial \mathbf{e}} = 2\mathbf{S}\mathbf{e} - 2\lambda\mathbf{e}. \quad (87)$$

Setting this gradient vector equal to zero, we see that \mathbf{e} must be an eigenvector of the scatter matrix:

$$\mathbf{S}\mathbf{e} = \lambda\mathbf{e}. \quad (88)$$

In particular, because $\mathbf{e}'\mathbf{S}\mathbf{e} = \lambda\mathbf{e}'\mathbf{e} = \lambda$, it follows that to maximize $\mathbf{e}'\mathbf{S}\mathbf{e}$, we want to select the eigenvector corresponding to the largest eigenvalue of the scatter matrix. In other words, to find the best one-dimensional projection of the data (best in the least-

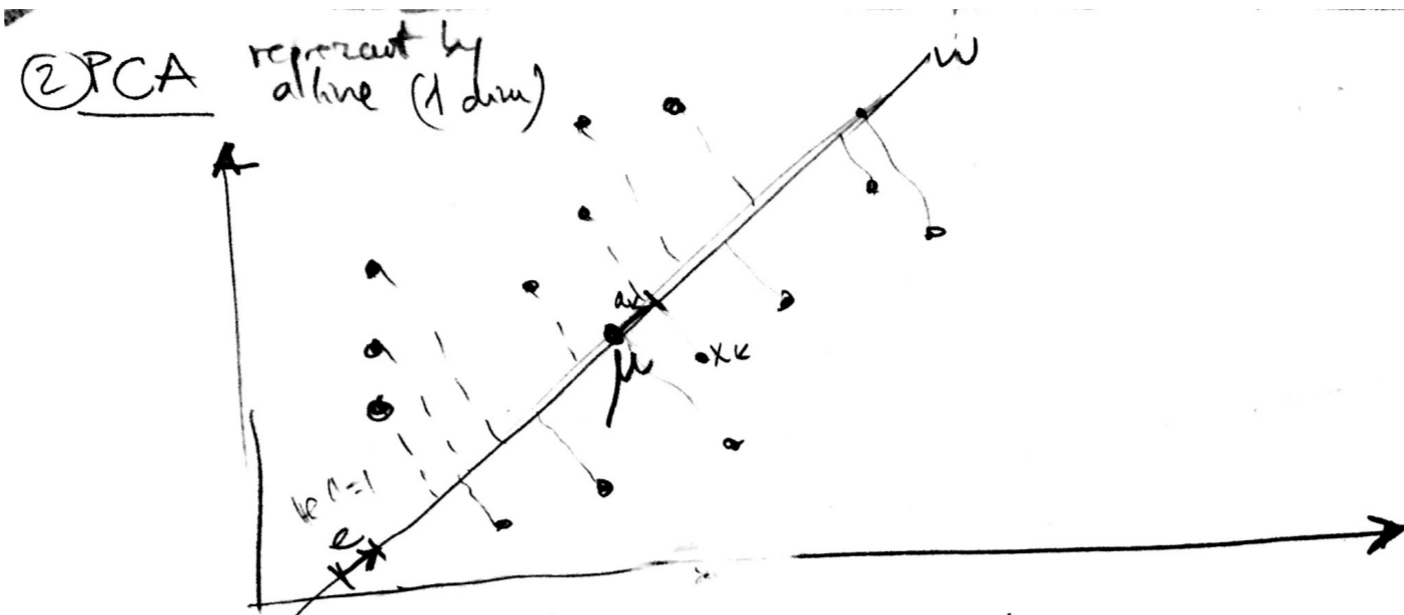
PCA

① REPRESENT BY A POINT (0 dimensional)

- that would be the mean

$$\mu = \frac{1}{n} \sum_k x_k$$

$$\mu = \underset{x}{\operatorname{argmin}} \sum_k \|x - x_k\|^2$$



the line will pass through the mean

we need to define and find a good w .
 $x_k = \mu + a_k e$ $\|e\|=1$ is the direction of the line
 $a_k = \text{scalar}$

$$J(a_1, a_2, \dots, e) = \sum_k \|\mu + a_k e - x_k\|^2 = \sum_k \|a_k e - (x_k - \mu)\|^2 =$$

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

$$\frac{\partial J}{\partial a_i} = 0 \Rightarrow a_k = e^T (x_k - \mu) \quad \left\| \quad E[\mu + a_k e] = \mu + E[a_k e] = \mu + e^T E[x_k - \mu] e = \mu$$

MOST IMPORTANT: what \boxed{e} is a good direction?

- minimize J
- maximize the variance of the projections on e .

Variance of projections

$$E\left[\left(\mu + a_k e - E[\mu + a_k e]\right)^2\right] = E\left[\mu + a_k e - \mu\right] = E\left[(a_k e)^2\right]$$

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

$$= e^T \Sigma e$$

where $\Sigma = \sum_k (x_k - \mu)(x_k - \mu)^T = \begin{bmatrix} \sum_k (x_k^1 - \mu^1)(x_k^1 - \mu^1) & \sum_k (x_k^1 - \mu^1)(x_k^d - \mu^d) \\ \sum_k (x_k^d - \mu^d)(x_k^1 - \mu^1) & \sum_k (x_k^d - \mu^d)(x_k^d - \mu^d) \end{bmatrix}$

$$= \begin{bmatrix} \sigma_{11} & \sigma_{1d} \\ \sigma_{d1} & \sigma_{dd} \end{bmatrix} = \text{covariance matrix}$$

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

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

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

$$= e^T \Sigma e$$

So minimizing the error \Leftrightarrow maximize the variance of projections.

maximize $e^T \Sigma e$

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

$$\text{Lagrangian } L = \max \left[e^T \Sigma e - \alpha (e^T e - 1) \right]$$

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

$$\begin{array}{l} \downarrow \\ e = \text{eigen vector of } \Sigma \\ \alpha = \text{eigen value of } \Sigma \end{array}$$

$$e^T \Sigma e = e^T \alpha e = \alpha$$

\downarrow
we need to choose the (eigen vector, eigen value) pair with the biggest eigen value.

Say we want a second "biggest" dimension.

- constrained that is orthogonal on the first dim $e_1 = e$
so that measures a diff variance component

$$\text{Lagrangian: } \max \left[e_2^T \Sigma e_2 - \alpha (e_2^T e_2 - 1) - \beta (e_2^T e_1 - 0) \right]$$

$$\frac{\partial L}{\partial e_2} = 0 \Rightarrow 2\Sigma e_2 - 2\alpha e_2 - \beta e_1 = 0$$

$$\begin{array}{c} \Downarrow \\ \underbrace{2e_2^T \Sigma e_2}_{e_2^T e_2 = 0} - \underbrace{2e_2^T \alpha e_2}_0 - e_1^T \beta e_1 = 0 \Rightarrow \beta = 0 \end{array}$$

$$\begin{array}{c} \downarrow \\ \Sigma e_2 = \alpha e_2 \Rightarrow e_2 \text{ eigen vector} \\ \alpha = \lambda_2 = \text{second eigenvalue} \end{array}$$

Spectral decomposition Σ symmetric, pos def? $\Rightarrow e_i$ orthogonal.

$$C = \left(\begin{array}{c|c|c} e_1 & e_2 & \dots & e_d \end{array} \right) \quad e_i = \text{eigenvectors of } \Sigma, \text{ normalized}$$

$e_i^T e_j = 0$ if $i \neq j$, $\|e_i\| = 1$. Then.

$$C \cdot C^T = I_d$$

$$\begin{aligned} \Sigma &= \Sigma C C^T = \Sigma \left(\begin{array}{c|c|c} e_1 & e_2 & \dots & e_d \end{array} \right) C^T = \left(\begin{array}{c} \Sigma e_1 \\ \Sigma e_2 \\ \dots \\ \Sigma e_d \end{array} \right) C^T \\ &= \left(\lambda_1 e_1, \lambda_2 e_2, \dots, \lambda_d e_d \right) C^T = \lambda_1 e_1 e_1^T + \lambda_2 e_2 e_2^T + \dots + \lambda_d e_d e_d^T \\ &= C D C^T \text{ where } D = \begin{pmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \lambda_d \end{pmatrix} \end{aligned}$$

$$\Sigma = \left[\begin{array}{c|c|c} e_1 & e_2 & \dots & e_d \end{array} \right] \cdot \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \lambda_d \end{bmatrix} \left[\begin{array}{c} \text{---} e_1 \text{---} \\ \text{---} e_2 \text{---} \\ \text{---} e_d \text{---} \end{array} \right]$$

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d \geq 0$$

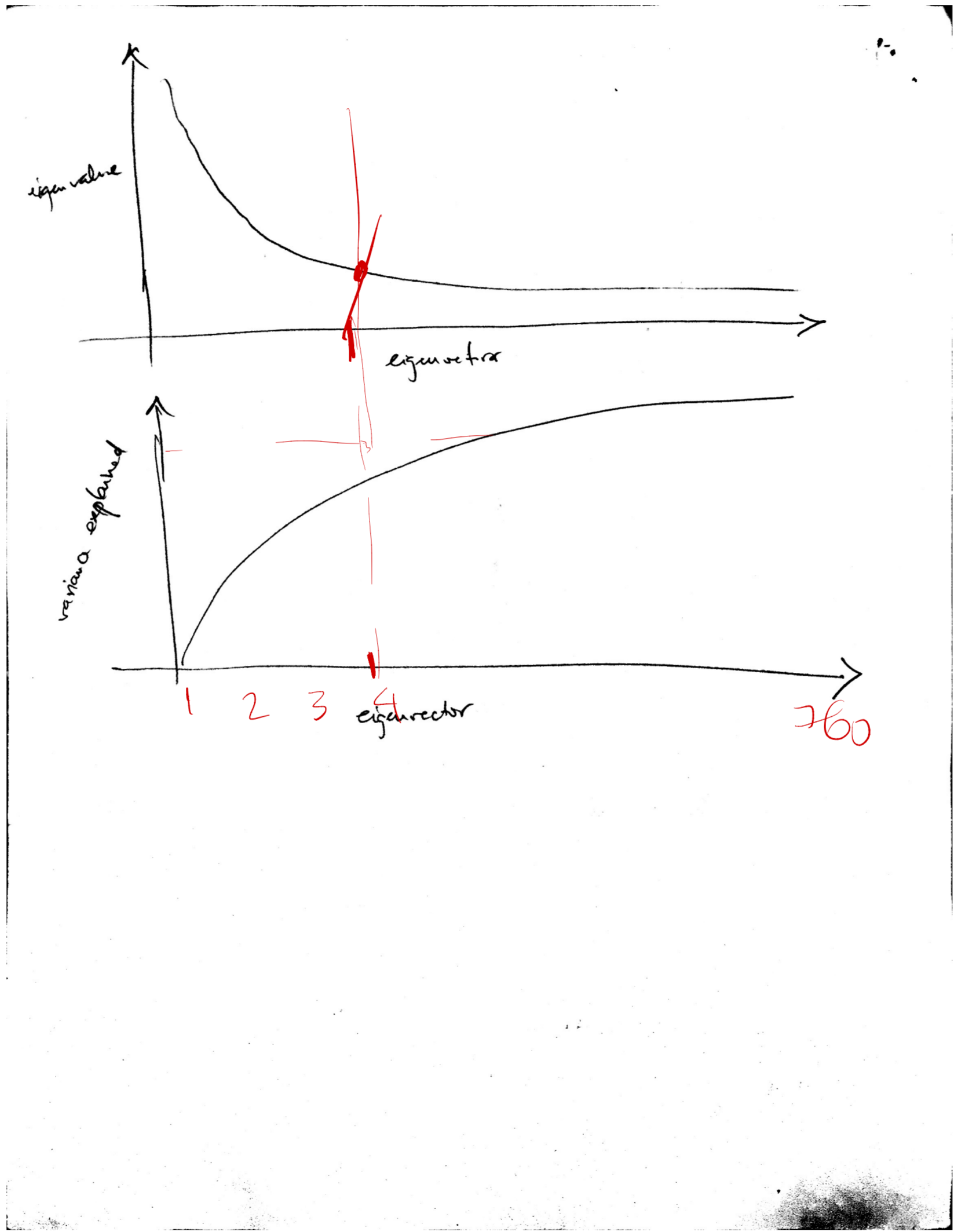
- if some $\lambda_i = \lambda_{i+1} = \dots = \lambda_d = 0$ then we can only keep e_1, e_2, \dots, e_t ; $\lambda_1, \lambda_2, \dots, \lambda_t$ so we reduced to t dimensions.

- if approximate to t dimensions, eliminate $t+1 \rightarrow d$ eigen v. even if they are not 0

Σ symmetric $\Rightarrow e_i$ orthogonal:

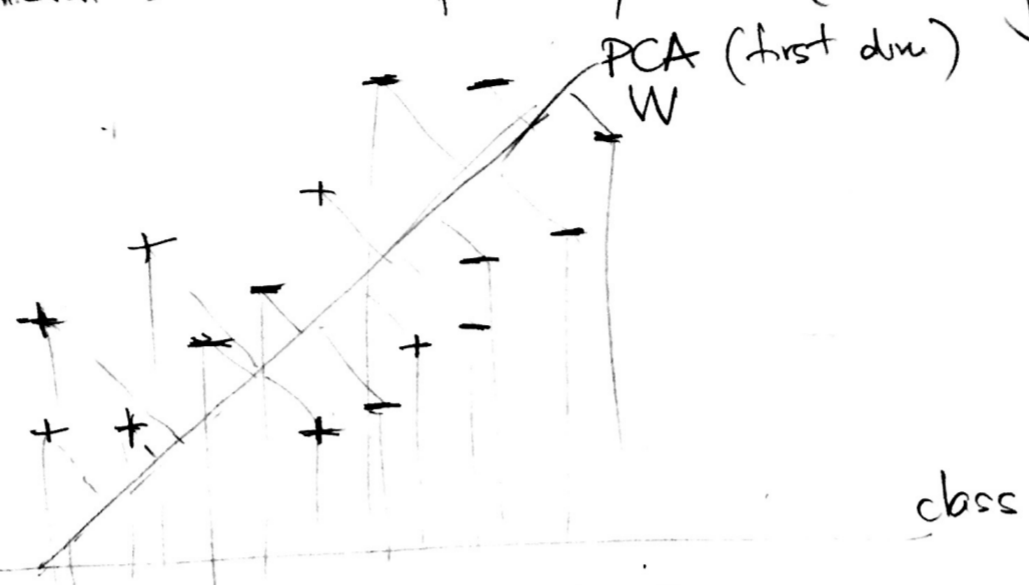
$$(\lambda e_1)^T e_2 = (\lambda e_1)^T e_2 = e_1^T \lambda e_2 = \lambda e_1^T e_2 = \lambda e_1^T e_2 = 0$$

But $\lambda_1 \neq \lambda_2 \Rightarrow e_1^T e_2 = 0$.



FISHER LINEAR DISCRIMINANT

Find dimensions that help classify data. (PCA might lose these)



n_1 points $\in C_1$
 n_2 points $\in C_2$

$$\mu_1 = \frac{1}{n_1} \sum_{x \in C_1} x$$

$$\mu_2 = \frac{1}{n_2} \sum_{x \in C_2} x$$

$\|w\|=1$ w = the line; projected points $z = wx$

projected means: $\bar{\mu}_1 = \frac{1}{n_1} \sum_{C_1} w^T x = w^T \mu_1$

distance between projected means is $|\bar{\mu}_1 - \bar{\mu}_2| = |w^T (\mu_1 - \mu_2)|$

$$\Sigma_i = \sum_{C_i} (x - \mu_i)(x - \mu_i)^T \quad \Sigma_w = \Sigma_1 + \Sigma_2$$

$$J_i^2 = \text{Var}_{C_i} [w^T x] = w^T \Sigma_i w$$

$$\sigma_1^2 + \sigma_2^2 = w^T \Sigma_1 w + w^T \Sigma_2 w = w^T \Sigma_w w$$

$$\Sigma_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

$$(\bar{\mu}_1 - \bar{\mu}_2)^2 = (w^T \mu_1 - w^T \mu_2)^2 = w^T \Sigma_B w$$

Fisher linear discriminant

$$\text{(maximize)} \quad J(w) = \frac{|\bar{\mu}_1 - \bar{\mu}_2|^2}{\sigma_1^2 + \sigma_2^2} =$$

$$= \frac{w^T \Sigma_B w}{w^T \Sigma_W w} \quad \left(\text{sometimes called "Rayleigh" quotient} \right)$$

Solution must satisfy $\Sigma_B w = \lambda \Sigma_W w$
(max J)

(A) if Σ_W nonsingular $\Rightarrow \Sigma_W^{-1} \Sigma_B w = \lambda w \Rightarrow$
 (λ, w) eigens of $\Sigma_W^{-1} \Sigma_B$

(B) for us, we do not need $\Sigma_W^{-1} \Sigma_B$ because $\Sigma_B w$ is in the
direction of $(\mu_1 - \mu_2)$, so $w = \Sigma_W^{-1} (\mu_1 - \mu_2)$

(C) All that is left is to find a threshold on the projections.

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

duality / KKT conditions

$$e_j = \sum_{i=1}^N \bar{\alpha}_j x_i = \sum_{i=1}^N \alpha_{ji} \cdot x_i$$

$\bar{\alpha}_j = (\alpha_{j1}, \alpha_{j2}, \dots, \alpha_{jN})$ dual variable

new rep (x)
 \bar{x} orig

$$= \bar{x} \cdot \begin{bmatrix} \sum_{i=1}^N \bar{\alpha}_1 x_i & \sum_{i=1}^N \bar{\alpha}_2 x_i & \dots & \sum_{i=1}^N \bar{\alpha}_N x_i \end{bmatrix}$$

dual
 eigen vectors in dual form

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

$$\bar{x} = \sum \alpha_j \sum_i \bar{x} x_i$$

new rep(x) = sim vector with other points

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

UNSUPERVISED LEARNING 2011

LECTURE :KERNEL PCA

Rita Osadchy

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

Dimensionality Reduction

- Data representation

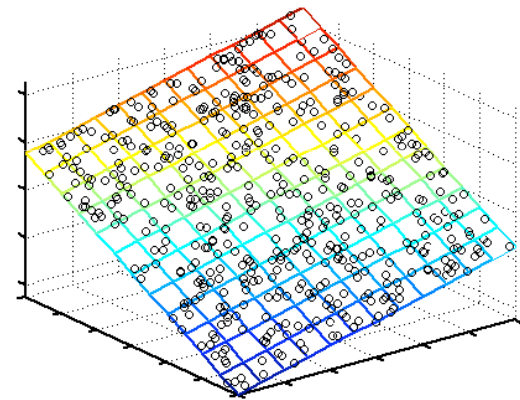
Inputs are real-valued vectors in a high dimensional space.

- Linear structure

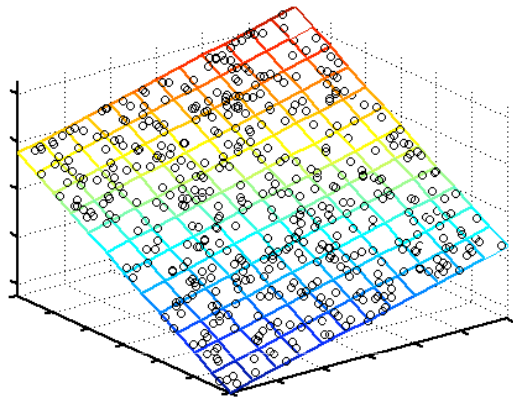
Does the data live in a low dimensional subspace?

- Nonlinear structure

Does the data live on a low dimensional submanifold?



Dimensionality Reduction so far



PCA

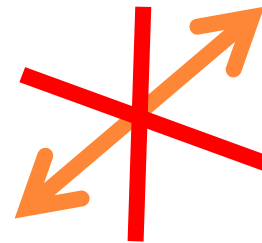


Manifold learning methods

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

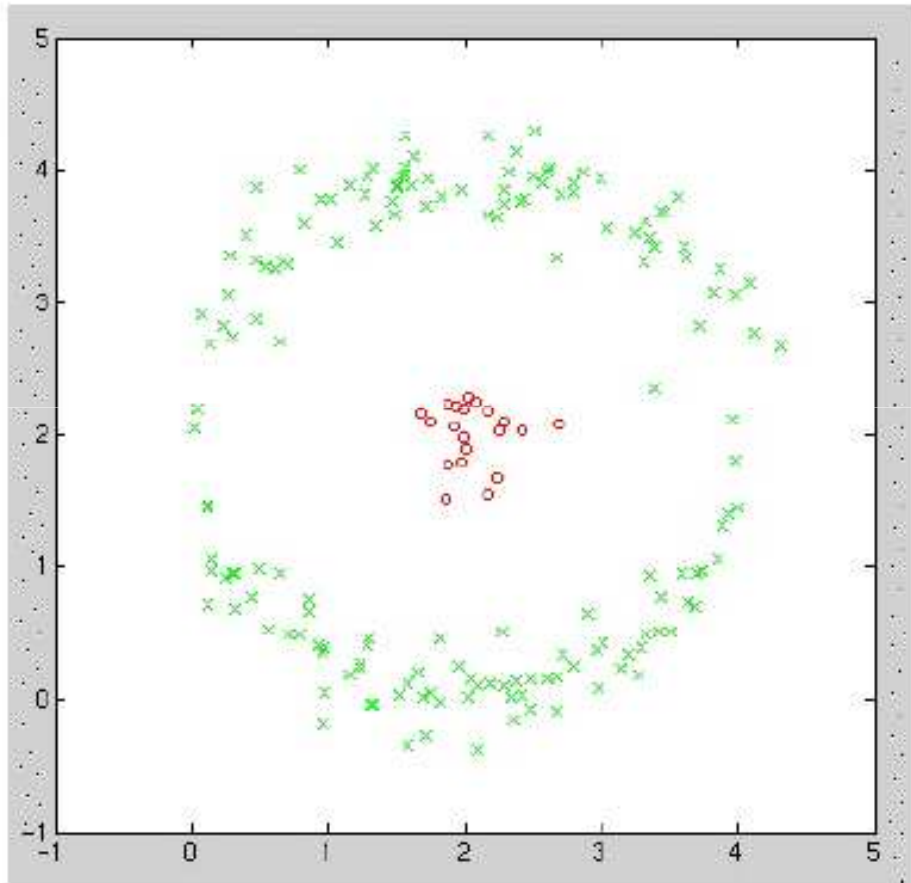
- Outputs (**low dimensional**)

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

The “magic” of high dimensions

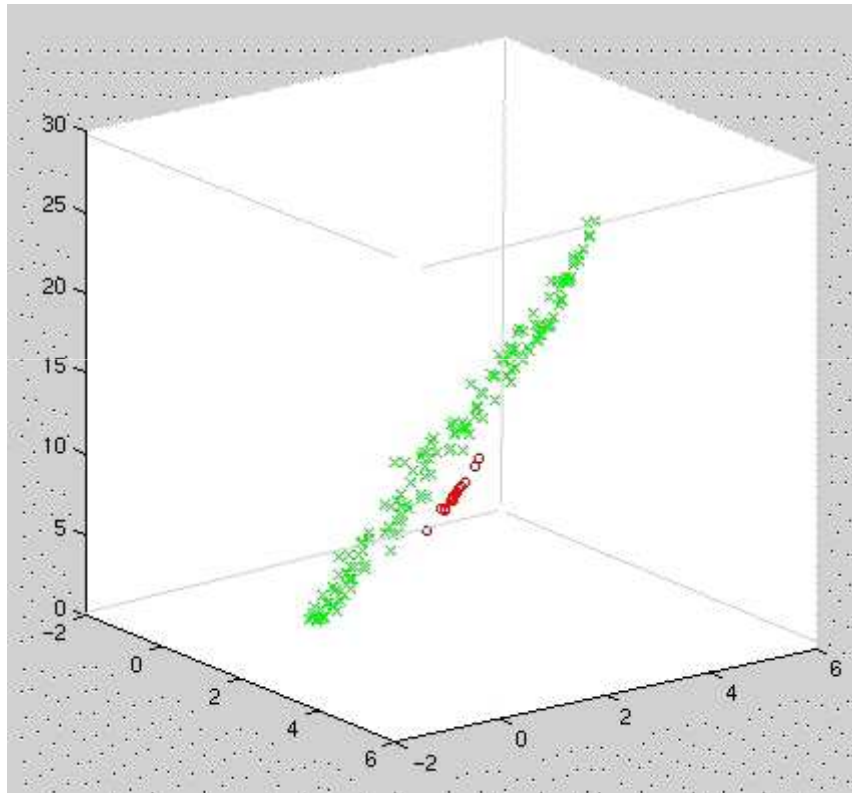
- Given some problem, how do we know what classes of functions are capable of solving that problem?
- VC (Vapnik-Chervonenkis) theory tells us that often mappings which take us into a higher dimensional space than the dimension of the input space provide us with greater classification power.

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 ϕ
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 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:

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

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 α_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 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:

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

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

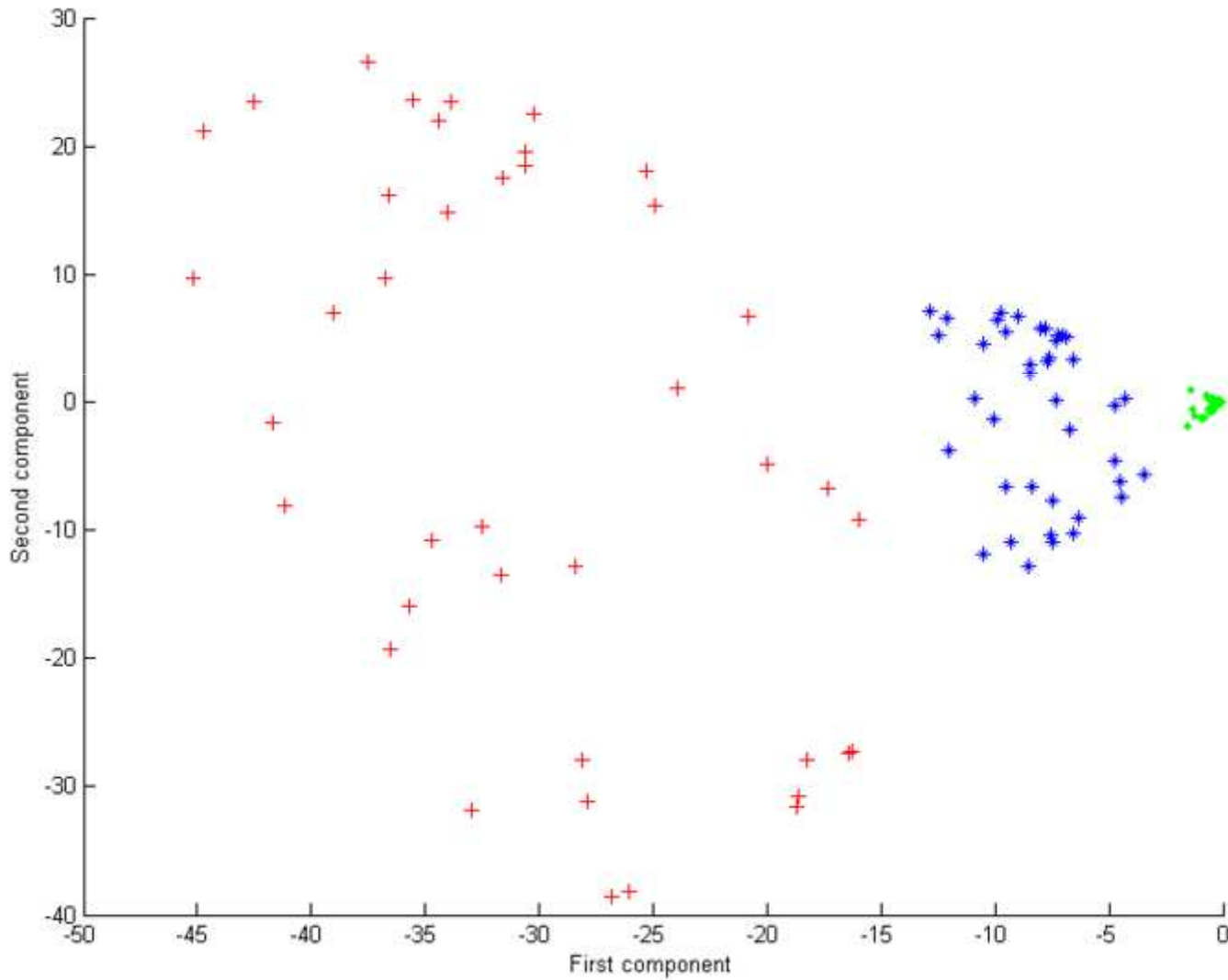
x = orig point

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: KPCA

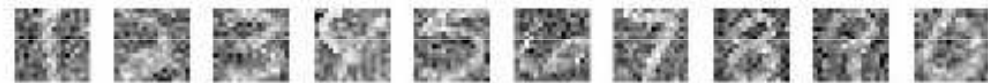


Example: De-noising images

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.