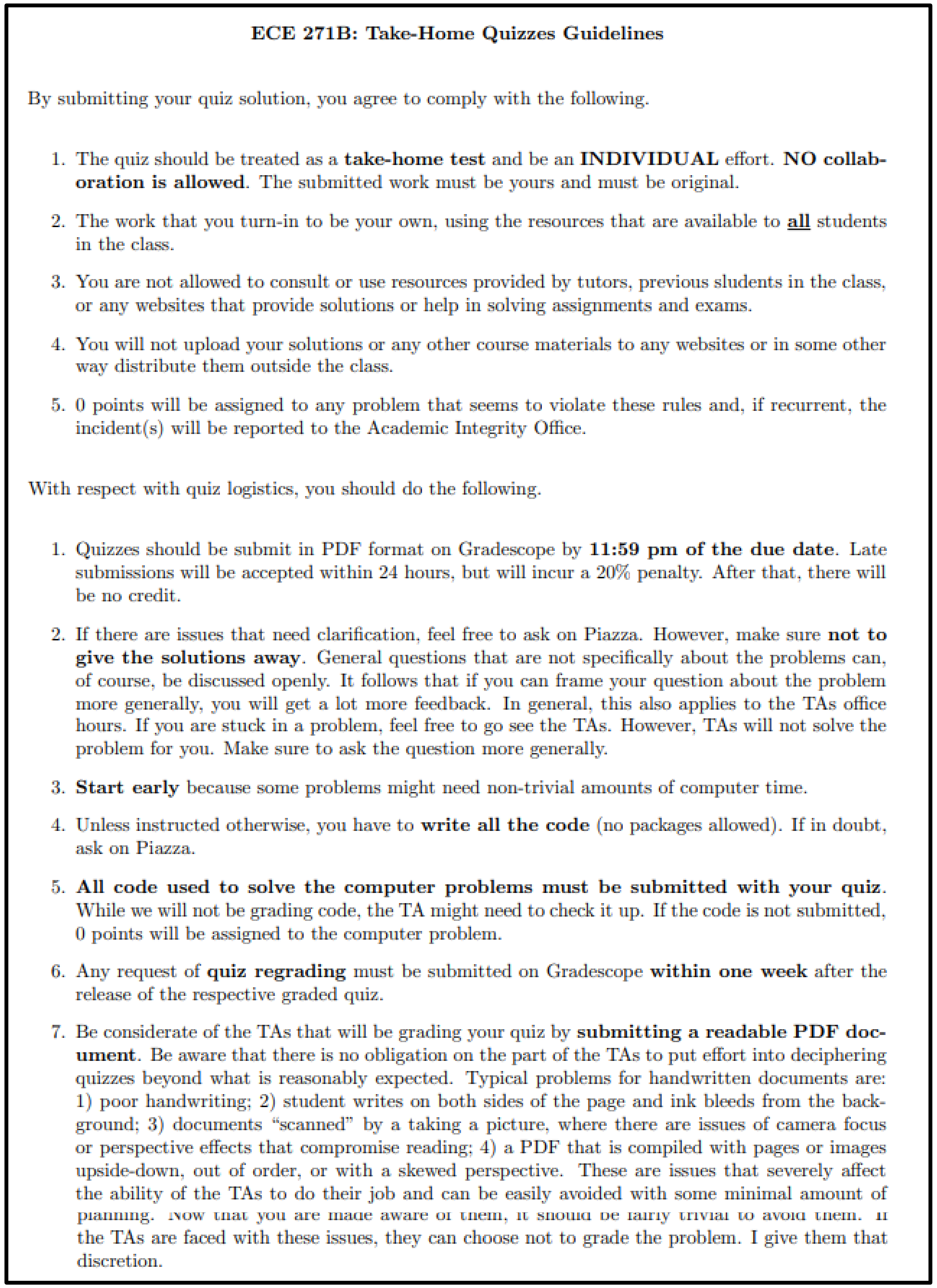


ECE 271B: Take-Home Quizzes Guidelines

By submitting your quiz solution, you agree to comply with the following.

1. The quiz should be treated as a **take-home test** and be an **INDIVIDUAL** effort. **NO collaboration is allowed**. The submitted work must be yours and must be original.
2. The work that you turn-in to be your own, using the resources that are available to **all** students in the class.
3. You are not allowed to consult or use resources provided by tutors, previous students in the class, or any websites that provide solutions or help in solving assignments and exams.
4. You will not upload your solutions or any other course materials to any websites or in some other way distribute them outside the class.
5. 0 points will be assigned to any problem that seems to violate these rules and, if recurrent, the incident(s) will be reported to the Academic Integrity Office.

With respect with quiz logistics, you should do the following.

1. Quizzes should be submit in PDF format on Gradescope by **11:59 pm of the due date**. Late submissions will be accepted within 24 hours, but will incur a 20% penalty. After that, there will be no credit.
2. If there are issues that need clarification, feel free to ask on Piazza. However, make sure **not to give the solutions away**. General questions that are not specifically about the problems can, of course, be discussed openly. It follows that if you can frame your question about the problem more generally, you will get a lot more feedback. In general, this also applies to the TAs office hours. If you are stuck in a problem, feel free to go see the TAs. However, TAs will not solve the problem for you. Make sure to ask the question more generally.
3. **Start early** because some problems might need non-trivial amounts of computer time.
4. Unless instructed otherwise, you have to **write all the code** (no packages allowed). If in doubt, ask on Piazza.
5. **All code used to solve the computer problems must be submitted with your quiz**. While we will not be grading code, the TA might need to check it up. If the code is not submitted, 0 points will be assigned to the computer problem.
6. Any request of **quiz regrading** must be submitted on Gradescope **within one week** after the release of the respective graded quiz.
7. Be considerate of the TAs that will be grading your quiz by **submitting a readable PDF document**. Be aware that there is no obligation on the part of the TAs to put effort into deciphering quizzes beyond what is reasonably expected. Typical problems for handwritten documents are: 1) poor handwriting; 2) student writes on both sides of the page and ink bleeds from the background; 3) documents "scanned" by a taking a picture, where there are issues of camera focus or perspective effects that compromise reading; 4) a PDF that is compiled with pages or images upside-down, out of order, or with a skewed perspective. These are issues that severely affect the ability of the TAs to do their job and can be easily avoided with some minimal amount of planning. Now that you are made aware of them, it should be fairly trivial to avoid them. If the TAs are faced with these issues, they can choose not to grade the problem. I give them that discretion.

Take–Home Quiz #1
posted on Canvas

Due date: **Tuesday, 1/25**

Project

- project groups
  - groups of 3–4
  - if needed, feel free to use "**Search for Teammates!**" feature on Piazza (pinned)
  - send me an email (mvasconcelos@eng.ucsd.edu) stating who are the group **members** (please use your official UCSD name) as soon as you know it, with deadline Tuesday, 1/18

- project proposal
  - due Tuesday, 2/1 @ 11:59pm
  - one–page maximum stating:
    - problem
    - data you will use
    - draft of proposed solution (can be updated later)
    - experiments you will run (can be updated later)
    - references (you can use an additional page for this)

ECE 271B – Winter 2021

Dimensionality and Dimensionality Reduction

Disclaimer:
This class will be recorded
and made available to students asynchronously.

Manuela Vasconcelos

ECE Department, UCSD

Plan for Today

- data and dimensionality
- dimensionality reduction is needed because high dimensional spaces are STRANGE!!!
- introduction to dimensionality reduction
- Principal Component Analysis (PCA)
- **detour**: functional derivatives

Data and Dimensionality

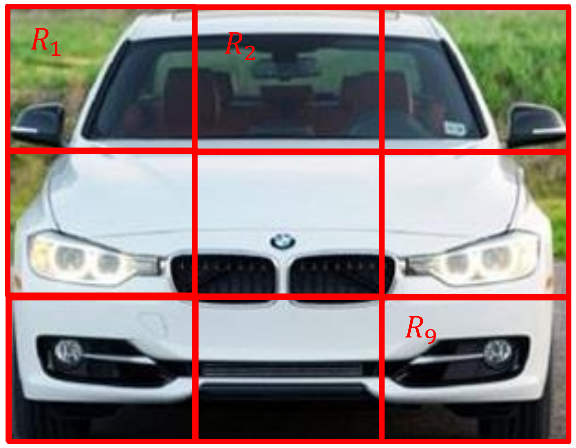
- machine learning is about processing data
  - first step is to **represent data** in a vector space S
  - examples are **mapped into vectors**
  - this makes a dataset a collection of points in S
- assume we need to build a system to distinguish between images of faces and images of cars

Data and Dimensionality

- we need a vector representation for images
  - this is obtained by extracting various features from an image
  - these could be many things, e.g.
    - $f_1(\mathbf{x})$ = average color region R_1
    - $f_2(\mathbf{x})$ = average color region R_2

      $\vdots$
    - $f_9(\mathbf{x})$ = average color region R_9
    - the image is mapped into a $9-$ dimensional feature vector

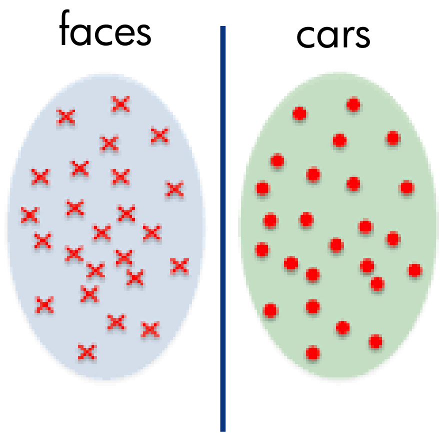
$$\mathbf{x} \rightarrow (f_1, \cdots, f_9)$$



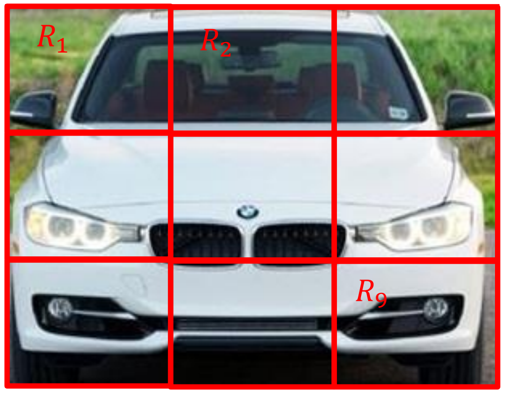
  - the **space of images** is mapped into a **9−dimensional feature space**
  - classification reduces to finding a class boundary in this space

Data and Dimensionality

- this
  - is easy to do in small dimensional spaces
    - e.g. in 2–dimensions we can do it by inspection $\longrightarrow$
  - but the difficulty **increases exponentially** with the dimension of the feature space S



- one property in our favor is that features are usually **redundant**
  - e.g. adjacent image regions tend to have the same average color
- hence, we can eliminate features without sacrificing performance
- this is known as **dimensionality reduction**



Data and Dimensionality

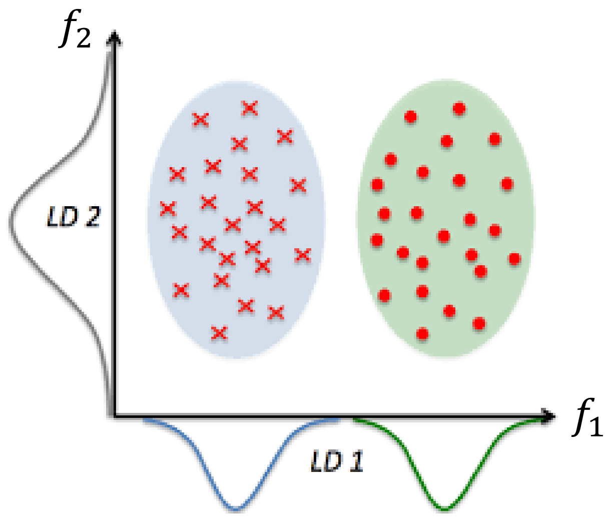
- problem:
  - in general, we do <u>not</u> know what is a <u>good</u> small set of features

- solution:

1) start from a set as <u>large</u> as you can, so as to make sure that <u>no</u> information is lost
   - for example, represent the image by the vector of its pixel colors
   - in this case a $M \times N$ image originates an MN – dimensional vector, e.g. 200×200 pixel images create a $40{,}000$ – dimensional space
   - the dimension of this space can be **huge**
2) use data to learn which <u>dimensions</u> are important
3) <u>project</u> the data on these important dimensions
4) <u>disregard</u> the remaining ones

Data and Dimensionality

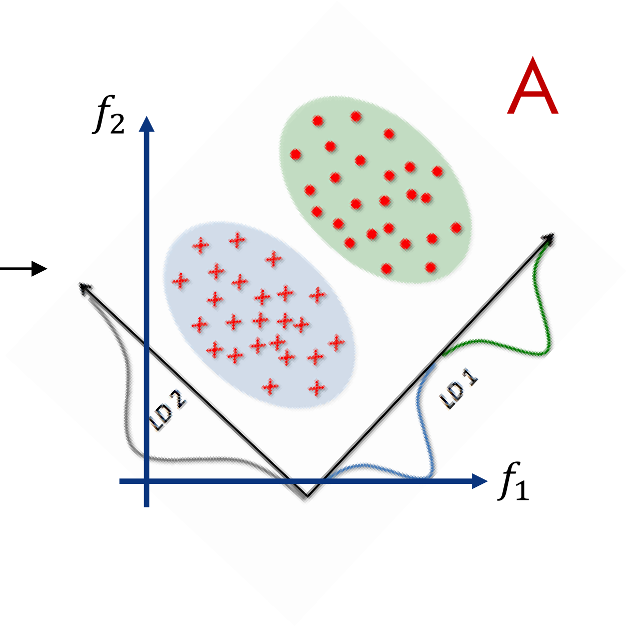
- **Q:** what is a **good** dimension?



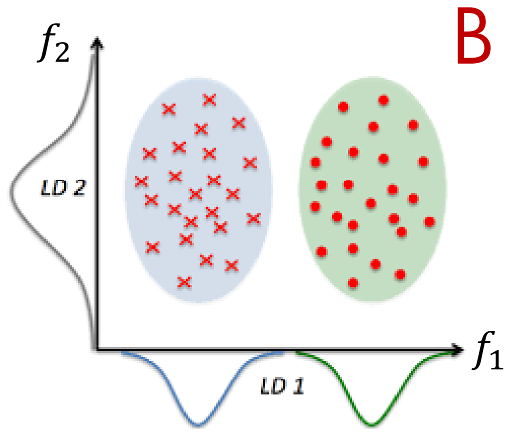
- **A:** a dimension that **<u>separates</u>** the classes
  - this allows classification with <u>no</u> error
  - in the example,
    - feature f_1 is a **good** dimension
      - the data projects on this dimension into two non–overlapping Gaussian distributions
      - f_1 is called a $\boxed{\textbf{discriminant}\text{ dimension or feature}}$
    - feature f_2 is **<u>not</u>** a **good** dimension
      - the two classes have identically distributed projections on this feature, making them impossible to distinguish
      - f_2 is a $\boxed{\textbf{non–discriminant}\text{ dimension or feature}}$

Data and Dimensionality

- **problem**: discriminant features are **not** always obvious
  - for example, a simple rotation of the space makes both f_1 and f_2 non−discriminant
  - neither of them is a projection that separates the classes



- this gives rise to two types of **dimensionality reduction techniques**
  - **feature extraction**:
    - map $\mathbf{f}$ into a new, lower dimension, vector $\mathbf{f}'$
    - useful for problems of type A
  - **feature selection**:
    - select a subset of the original features
    - useful for problems of type B

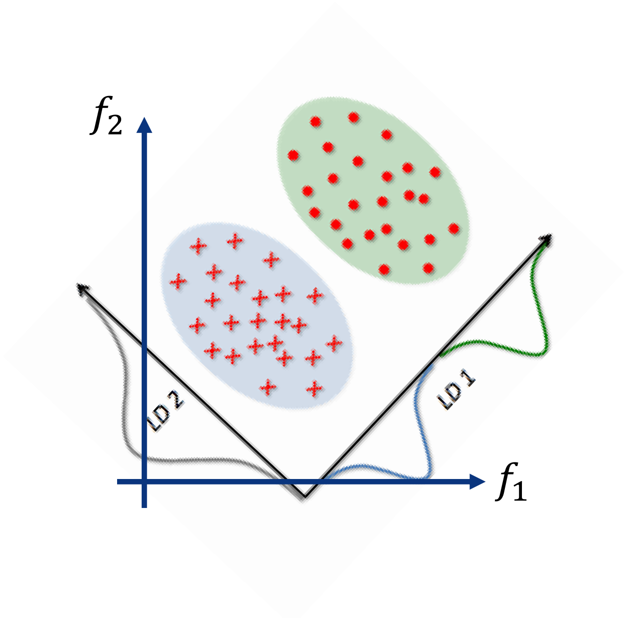


Data and Dimensionality

- in the next lectures, we will consider feature extraction
- this can again be of two types
  - linear feature extraction
    - find a linear projection of the data, i.e. a matrix $\mathbf{\Phi}$ such that

      $$\mathbf{f}' = \mathbf{\Phi}\mathbf{f}$$

      is a discriminant feature vector
    - note that $\mathbf{f}'$ is the vector of projections of $\mathbf{f}$ on the rows of $\mathbf{\Phi}$



  - non−linear feature extraction
    - in this case, the projection can be non−linear, i.e. $\mathbf{f}' = \mathbf{\Phi}(\mathbf{f})$, where $\mathbf{\Phi}$ is a non−linear operator (e.g. a neural network)
- for now, we consider linear feature extraction

High–Dimensional Spaces

- are strange!
- first thing to know:

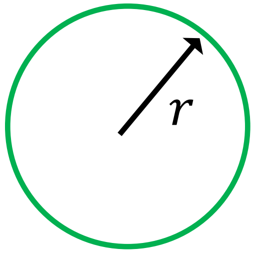
  "**<u>never</u>** trust your intuition in high–dimensions!"

- more often than not, you will be wrong!
- there are **many examples** of this
- we will do a couple

The Hyper–Sphere

- Consider the sphere of radius r on a space of dimension d

$$S = \left\{ \mathbf{x} \;\middle|\; \sum_{i=1}^{d} x_i^2 \leq r^2 \right\}$$



- **Homework:** show that its volume is

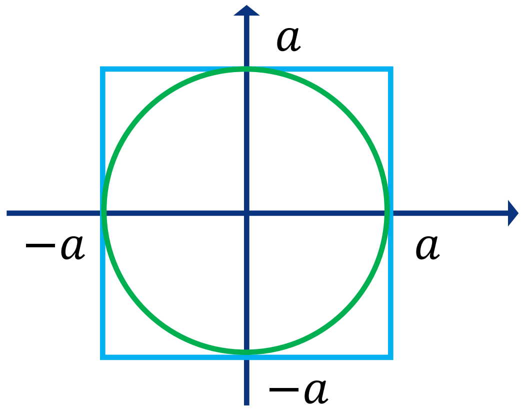
$$V_d(r) = \frac{r^d \pi^{d/2}}{\Gamma\left(\frac{d}{2} + 1\right)}$$

where $\Gamma(n)$ is the Gamma function

$$\Gamma(n) = \int_0^{\infty} e^{-x} x^{-n-1}\, dx$$

Hyper–Cube vs Hyper–Sphere

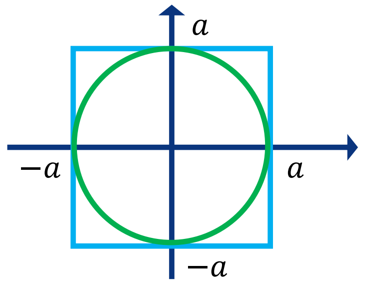
- Next, consider the hyper–cube $[-a, a]^d$ and the inscribed hyper–sphere, i.e.



- Q: what does **your intuition** tell you about the relative sizes of these two objects?

1. volume of sphere $\gg$ volume of cube
2. volume of sphere $\ll$ volume of cube
3. volume of sphere $\cong$ volume of cube

Answer



- we can just compute

$$f_d = \frac{\text{Vol(sphere)}}{\text{Vol(cube)}} = \frac{\frac{a^d \pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)}}{(2a)^d} = \frac{\pi^{d/2}}{2^d \Gamma\left(\frac{d}{2}+1\right)}$$

- f_d that does **not** depend on a, just on the dimension d!

d	1	2	3	4	5	6	7
f_d	1	.785	.524	.308	.164	.08	.037

- it goes to zero and goes to zero <u>fast</u>!

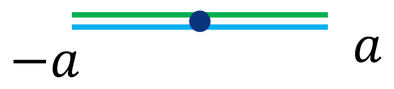
Hyper–Cube vs Hyper–Sphere

- this means that:

  "as the dimension of the space increases, the volume of the sphere is much smaller (infinitesimal) than that of the cube!"

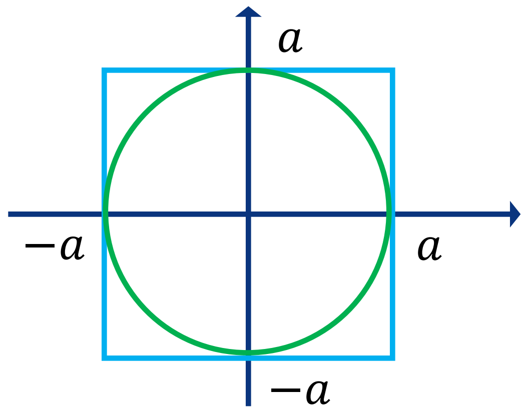
- how is this going against intuition?
- it is actually not very surprising: we can see it even in low–dimensions

1) $d = 1$



volume is the same

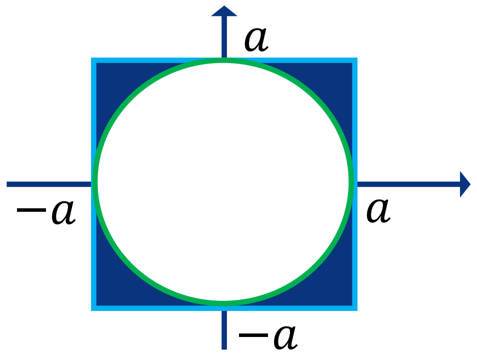
2) $d = 2$



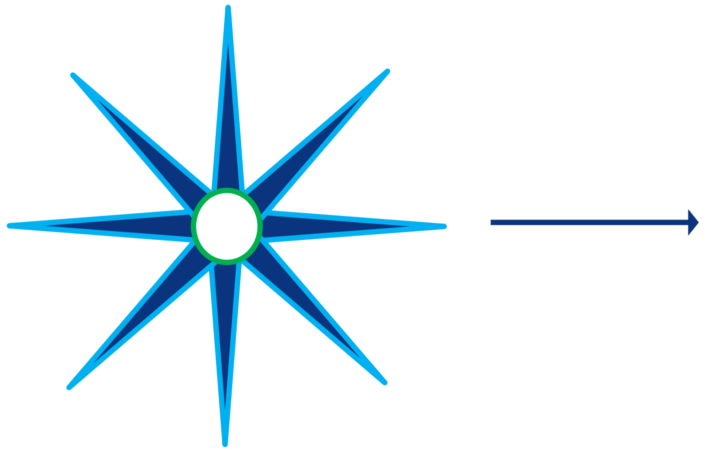
volume of sphere is already smaller

Hyper–Cube vs Hyper–Sphere

- as the dimension increases, the volume of the shaded corners becomes larger



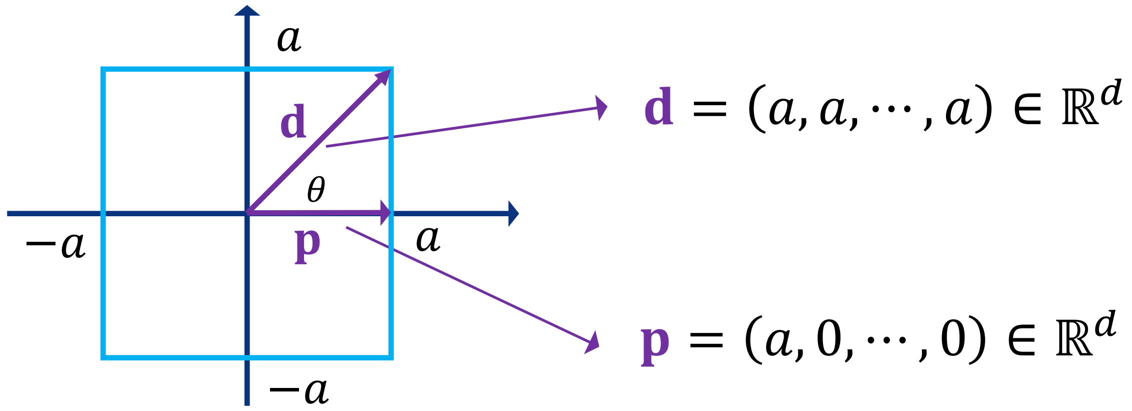
- in high–dimensions, the picture you should have in mind is



all the volume of the cube
is in these spikes!

Believe It or Not

- we can check mathematically: consider $\mathbf{d}$ and $\mathbf{p}$



a
$\mathbf{d}$
θ
$\mathbf{p}$
a
$-a$
$-a$
$\mathbf{d} = (a, a, \cdots, a) \in \mathbb{R}^d$
$\mathbf{p} = (a, 0, \cdots, 0) \in \mathbb{R}^d$

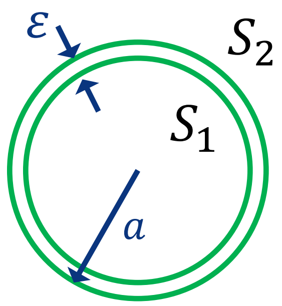
- note that

$$\frac{\|\mathbf{d}\|^2}{\|\mathbf{p}\|^2} = \frac{da^2}{a^2} = d \to \infty \qquad \cos\theta = \frac{\mathbf{d}^T\mathbf{p}}{\sqrt{\|\mathbf{d}\|^2\|\mathbf{p}\|^2}} = \frac{a^2}{\sqrt{da^2a^2}} = \frac{1}{\sqrt{d}} \to 0$$

- $\mathbf{d}$ orthogonal to $\mathbf{p}$ as d increases and infinitely larger!!!

But There Is More...

- consider the crust of unit sphere of thickness ε
- we can compute its volume



$$\mathrm{Vol(crust)} = \mathrm{Vol}(S_2) - \mathrm{Vol}(S_1) = \left[1 - \frac{\mathrm{Vol}(S_1)}{\mathrm{Vol}(S_2)}\right] \mathrm{Vol}(S_2)$$

$$\frac{\mathrm{Vol}(S_1)}{\mathrm{Vol}(S_2)} = \frac{\dfrac{(a-\varepsilon)^d \pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)}}{\dfrac{a^d \pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)}} = \frac{a^d\left(1-\frac{\varepsilon}{a}\right)^d}{a^d} = \left(1-\frac{\varepsilon}{a}\right)^d$$

$$V_d(r) = \frac{r^d \pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)}$$

- no matter how small ε is, ratio goes to zero as d increases
- i.e. "all the volume is in the crust!"

High–Dimensional Gaussian

- **Homework**: show that if

$$\mathbf{X} \sim N(\mathbf{0}, \mathbf{I}), \mathbf{x} \in \mathbb{R}^n$$

and one considers the hyper–sphere where the probability density drops to 1% of peak value

$$S_{0.01}(\mathbf{x}) = \left\{ \mathbf{x} \;\middle|\; \frac{G(\mathbf{x}, \mathbf{0}, \mathbf{I})}{G(\mathbf{0}, \mathbf{0}, \mathbf{I})} \le 0.01 \right\}$$

the probability mass **outside** this sphere is

$$P_n = P[\chi^2(n) \ge 9.21]$$

where $\chi^2(n)$ is a chi–squared random variable with n degrees of freedom

High–Dimensional Gaussian

- if you evaluate this, you will find out that

inside sphere

n	1	2	3	4	5	6	10	15	20
$1 - P_n$	.998	.99	.97	.94	.89	.83	.48	.134	.02

- as the dimension increases, **all** probability mass is on the **tails**
- the point of maximum density is still the mean
- really strange: in high–dimensions, the Gaussian is a very **heavy tailed** distribution
- take–home message:

  "in high dimensions never trust your intuition!"

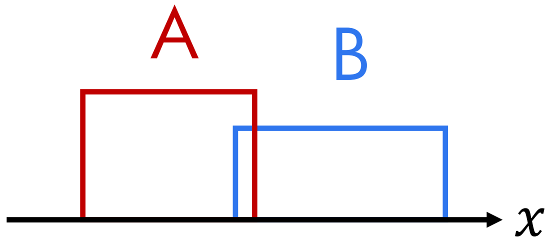
- Q: how does all this **affect** decision rules?

The Curse of Dimensionality

- typical observation in **Bayes decision theory**:
  - error **increases** when number of features is **large**
  - **harder** to build a classifier in high−dimensional spaces

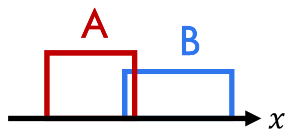
- highly **unintuitive** since, theoretically,
  - if we have a problem in n − dimension, we can always generate a problem in $(n+1)$ − dimension with smaller probability of error

- e.g. two uniform classes in 1D

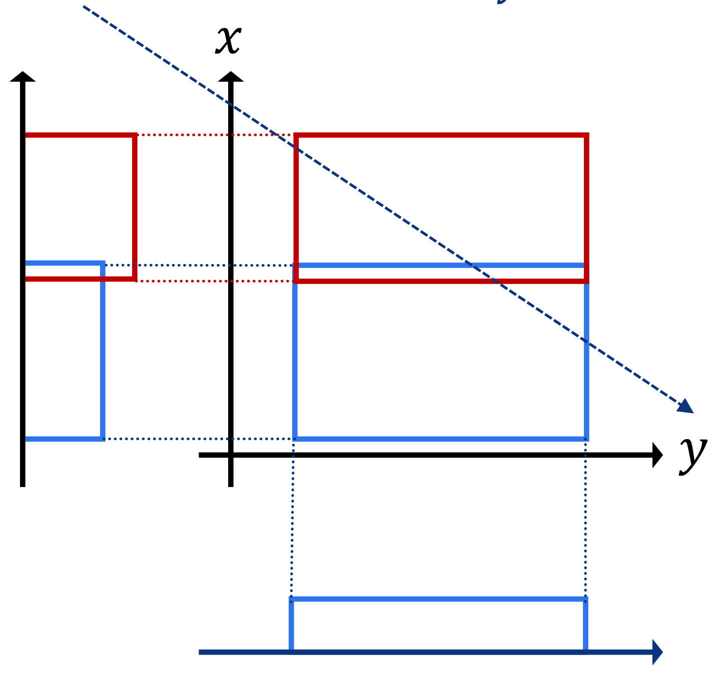


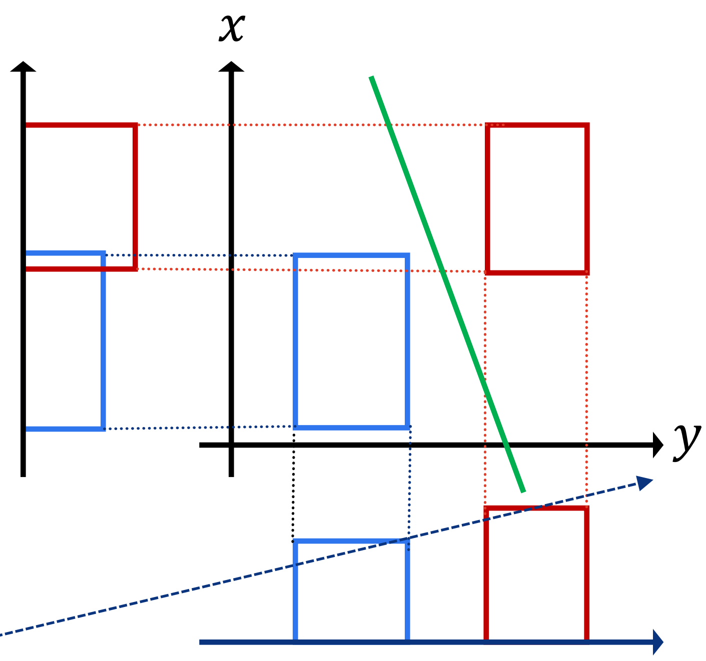
can be transformed into a 2D problem with same error by just adding a **non−informative** variable y

The Curse of Dimensionality



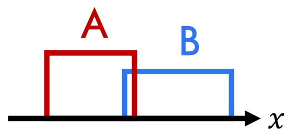
- can be transformed into a 2D problem with same error by just adding a **non−informative** variable y



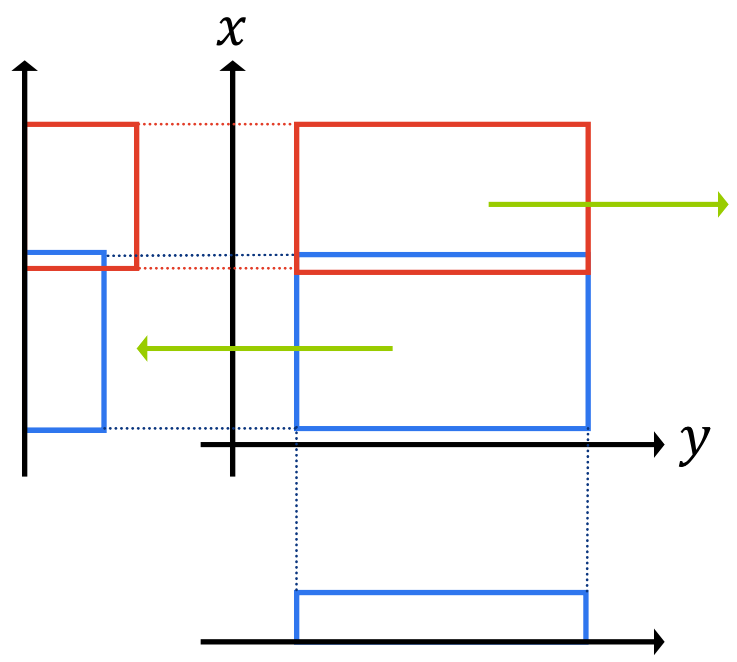


- but it is also possible to reduce the error by adding a second variable which is **informative**
- on the left, there is <u>**no**</u> decision boundary that will achieve zero error
- on the right, the decision boundary shown has zero error

The Curse of Dimensionality



- in fact, it is **impossible** to do worse in 2D than 1D



- if we move the classes along the lines shown in green, the error can only go down since there will be less overlap

The Curse of Dimensionality

- so **why** do we observe this curse of dimensionality?
- the problem is the **quality** of the density estimates
- the best way to see this is to think of a **histogram**
  - suppose you have 100 points and you need at least 10 bins per axis in order to get a reasonable quantization
  - for uniform data you get, on average,

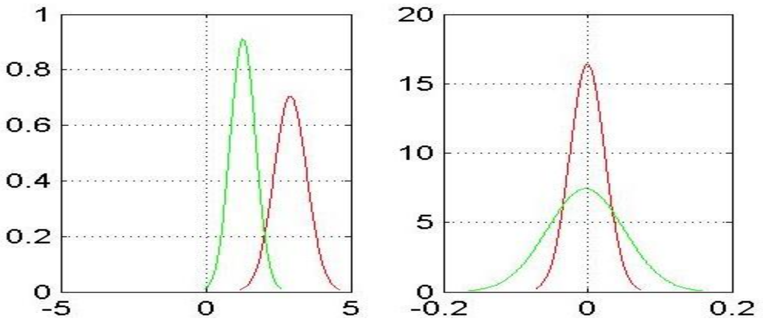
dimension	1	2	3
bins	10	100	1000
points/bin	10	1	0.1

  - decent in1D, bad in 2D, terrible in 3D (9 out of each 10 bins empty)
  - this has **nothing** to do with the probability distribution, just the **lack of data**

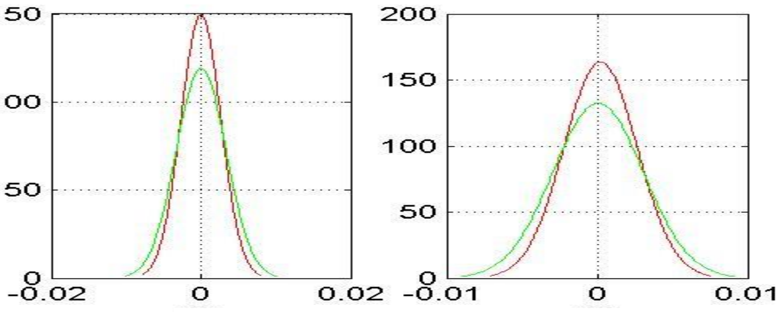
Dimensionality Reduction

- what do we do about this? we avoid <u>unnecessary</u> dimensions
- <u>unnecessary</u> can be measured in **two** ways:
  1) features are **not discriminant**
  2) features are **not independent**
- <u>non–discriminant</u> means that they do <u>not</u> separate the classes well, we talked about this already

discriminant



non–discriminant



Dimensionality Reduction

- dependent features, even if very discriminant, are not needed – **one is enough**!

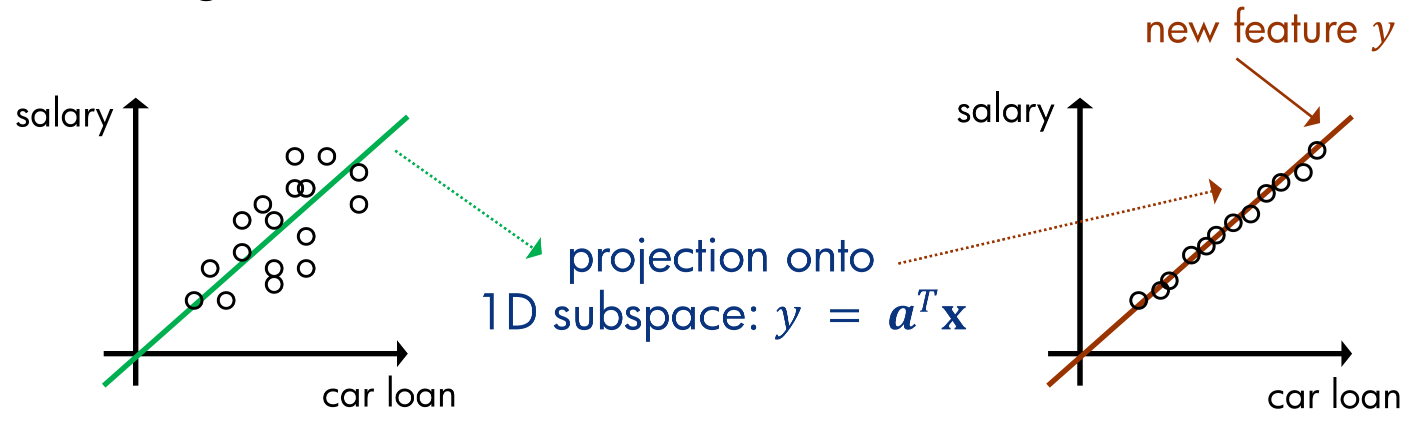
- e.g. data–mining company studying consumer credit card ratings

  X = {salary, mortgage, car loan, # of kids, profession, ...}

  - the first three features tend to be highly correlated:
    - "the more you make, the higher the mortgage, the more expensive the car you drive"
    - from one of these variables, I can predict the others very well
  - including features 2 and 3 does not increase the discrimination, but increases the dimension and leads to poor density estimates

Dimensionality Reduction

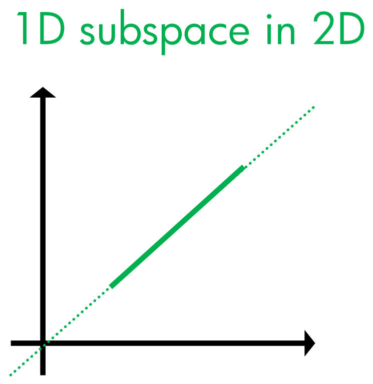
- Q: how do we detect the **presence** of these correlations?
- A: the data “lives” in a low–dimensional subspace (up to some amounts of noise). E.g.

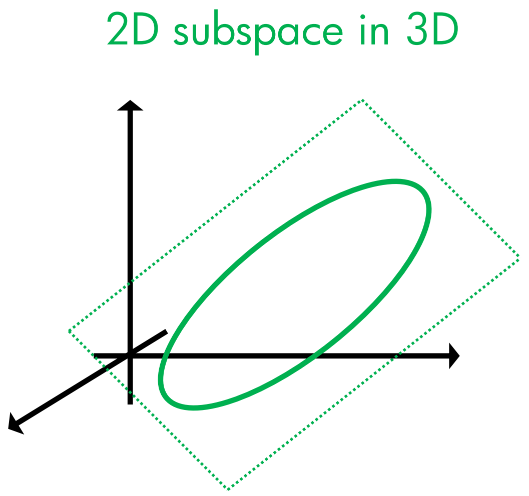


- in our example, we have a 3D hyper–plane in 5D
- if we can find this hyper–plane, we can
  - project the data onto it
  - get rid of ~ half of the dimensions without introducing significant error

Principal Component Analysis

- basic idea:
  - if the data lives in a subspace, it is going to look **very flat** when viewed from the full space, e.g.





  - this means that if we fit a **Gaussian** to the data, the equiprobability contours are going to be **<u>highly skewed</u> ellipsoids**

Gaussian Review

- the **equiprobability contours** of a Gaussian are the points such that $(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) = K$

- let's consider the change of variable $\mathbf{z} = \mathbf{x} - \boldsymbol{\mu}$, which only moves the origin by $\boldsymbol{\mu}$:

  the equation $\mathbf{z}^T\boldsymbol{\Sigma}^{-1}\mathbf{z} = K$ is the equation of an ellipse

- this is easy to see when $\boldsymbol{\Sigma}$ is **diagonal**

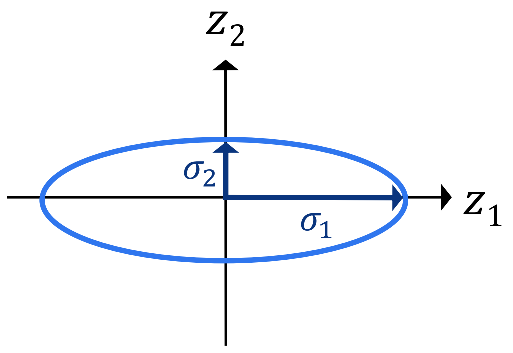
$$\boldsymbol{\Sigma} = \boldsymbol{\Lambda} = diag\left(\sigma_1^2, \cdots, \sigma_d^2\right) \Rightarrow \mathbf{z}^T\boldsymbol{\Sigma}^{-1}\mathbf{z} = \sum_i \frac{z_i^2}{\sigma_i^2} = K$$

this is the equation of an ellipse with principal lengths σ_i

e.g. when $d = 2$

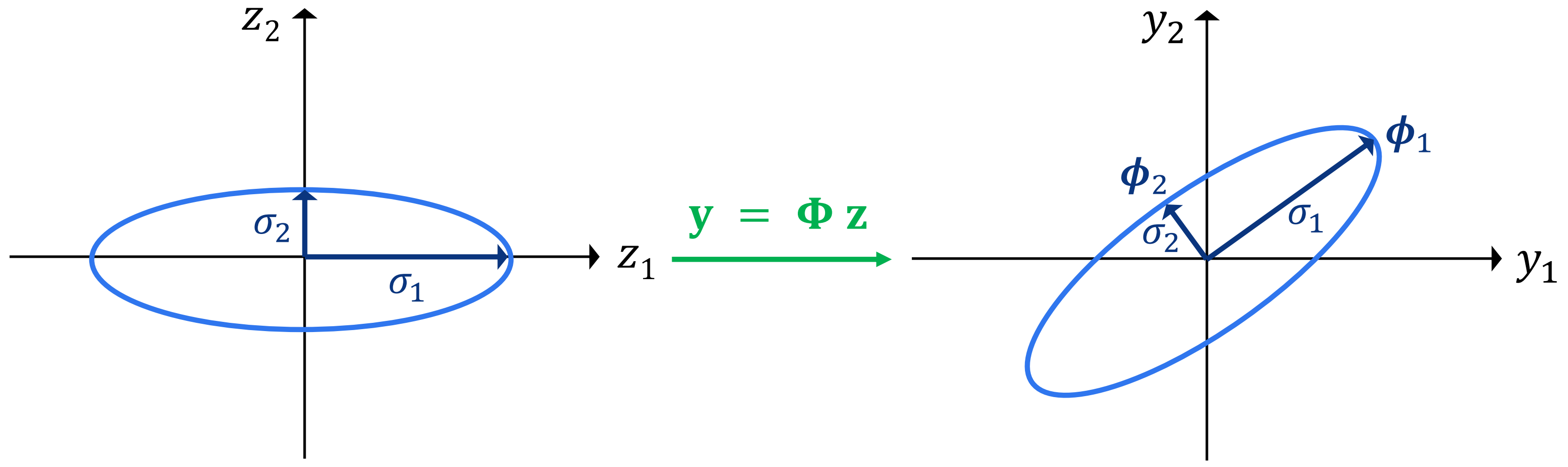
$$\frac{z_1^2}{\sigma_1^2} + \frac{z_2^2}{\sigma_2^2} = 1$$

is the ellipse



Gaussian Review

- introduce the transformation $\mathbf{y} = \mathbf{\Phi}\,\mathbf{z}$
- then $\mathbf{y}$ has covariance $\mathbf{\Sigma}_y = \mathbf{\Phi}\,\mathbf{\Sigma}_z\mathbf{\Phi}^T = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T$
- if $\mathbf{\Phi}$ is orthonormal, this is just a **rotation** and we have

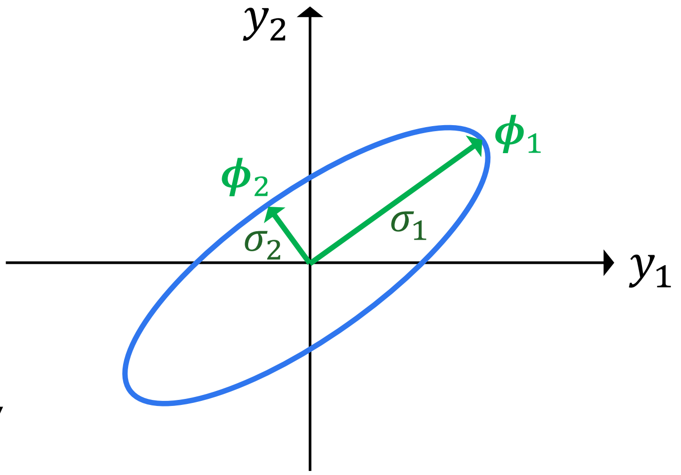


- we obtain a rotated ellipse with principal components $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$, which are the columns of $\mathbf{\Phi}$
- note that $\mathbf{\Sigma}_y = \mathbf{\Phi}\,\mathbf{\Sigma}_z\mathbf{\Phi}^T$ is the eigendecomposition of $\mathbf{\Sigma}_y$

$$\Phi = \begin{bmatrix} | & | \\ \phi_1 & \phi_2 \\ | & | \end{bmatrix}$$

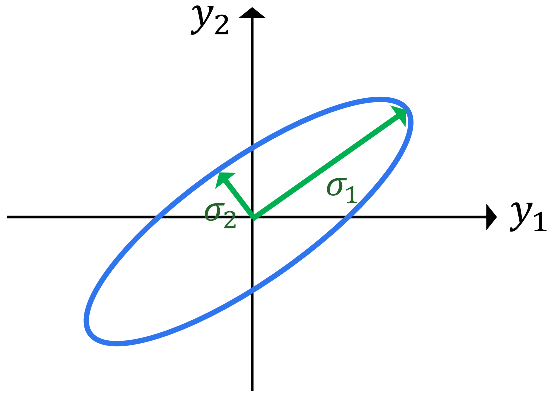
Principal Component Analysis (Learning)

- if $\mathbf{y}$ is Gaussian with covariance $\boldsymbol{\Sigma}$, the equiprobability contours are the **ellipses** whose
  - principal components $\boldsymbol{\phi}_i$ are the eigenvectors of $\boldsymbol{\Sigma}$
  - **principal lengths** σ_i are the **eigenvalues** of $\boldsymbol{\Sigma}$

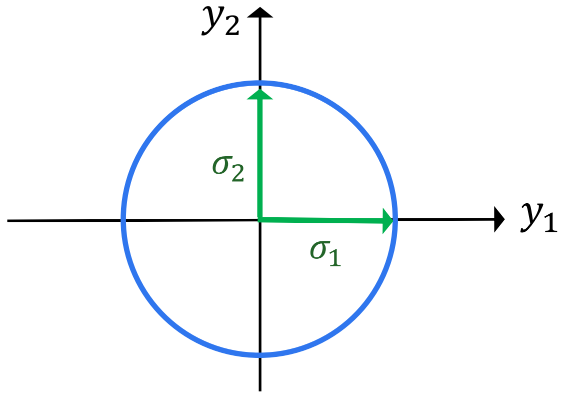


- by **computing the eigenvalues**, we know if **the data is flat**

$\sigma_1 \gg \sigma_2$: flat



$\sigma_1 = \sigma_2$: not flat



Principal Component Analysis (Learning)

Given sample $\mathcal{D} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$

- compute sample mean: $\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_i \mathbf{x}_i$
- compute sample covariance: $\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_i (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T$ (*)
- compute eigenvalues and eigenvectors of $\hat{\boldsymbol{\Sigma}}$

$$\hat{\boldsymbol{\Sigma}} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^T \qquad \boldsymbol{\Lambda} = diag\left(\sigma_1^2, \cdots, \sigma_d^2\right) \qquad \boldsymbol{\Phi}\boldsymbol{\Phi}^T = \mathbf{I}$$

- order eigenvalues: $\sigma_1^2 > \cdots > \sigma_n^2$
- if, for a certain k, $\sigma_k \ll \sigma_1$, eliminate the eigenvalues and eigenvectors above k

(*) Next class, we will see that there is an alternative way to compute PCA which does not require estimating the covariance matrix.

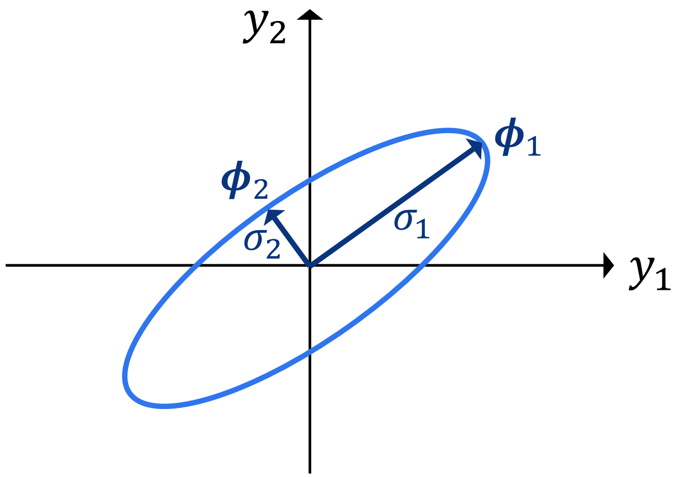
Principal Component Analysis

- Given a vector $\mathbf{y}$ in the **original** space $\mathbb{R}^d$
  - we can obtain the vector $\mathbf{z}$ by applying the transformation $\mathbf{\Phi}^{-1}$
  - since $\mathbf{\Phi}$ is orthogonal, this is just the transpose: $\mathbf{\Phi}^{-1} = \mathbf{\Phi}^T$

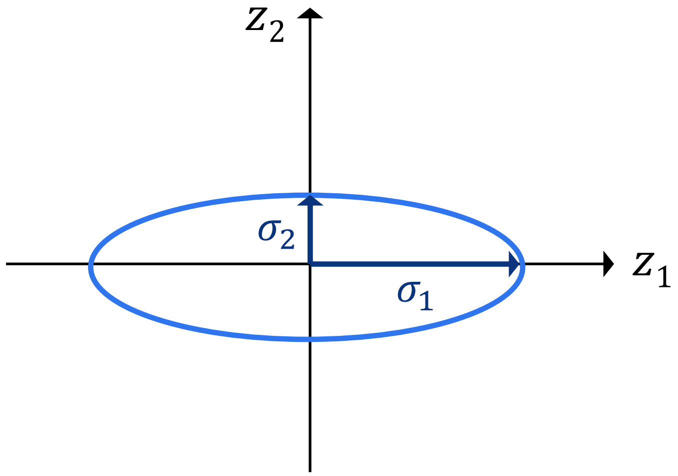
    $$\mathbf{\Phi} = \begin{bmatrix} | & & | \\ \boldsymbol{\phi}_1 & \cdots & \boldsymbol{\phi}_d \\ | & & | \end{bmatrix}$$

    note: $\mathbf{\Phi}^T$ is the matrix with principal components $\boldsymbol{\phi}_i$ as rows
  - elimination of components of small variance (eigenvalue) corresponds to eliminating all but k of these rows
  - in summary, PCA can be implemented with

$$\underbrace{\mathbf{z}}_{k-\text{dimensional}} = \underbrace{\begin{bmatrix} - \boldsymbol{\phi}_1^T - \\ \vdots \\ - \boldsymbol{\phi}_k^T - \end{bmatrix}}_{k \times d} \underbrace{\mathbf{y}}_{d-\text{dimensional}}$$



$$\mathbf{z} = \mathbf{\Phi}^T \mathbf{y}$$



Principal Component Analysis

Given principal components $\boldsymbol{\phi}_i, i \in \{1, \cdots, k\}$ and a test sample $\mathcal{T} = \{\mathbf{t}_1, \cdots, \mathbf{t}_n\}$, $\mathbf{t}_i \in \mathbb{R}^d$

- subtract mean to each point: $\mathbf{t}'_i = \mathbf{t}_i - \hat{\boldsymbol{\mu}}$
- project onto eigenvector space

$$\mathbf{y}_i = \mathbf{A}\,\mathbf{t}'_i \qquad \text{where} \qquad \mathbf{A} = \begin{bmatrix} - & \boldsymbol{\phi}_1^T & - \\ & \vdots & \\ - & \boldsymbol{\phi}_k^T & - \end{bmatrix}$$

- use $\mathcal{T}' = \{\mathbf{y}_1, \cdots, \mathbf{y}_n\}$ to do **all subsequent machine learning**

$$\underbrace{\mathbf{y}_i}_{k-\text{dimensional}} = \underbrace{\begin{bmatrix} - & \boldsymbol{\phi}_1^T & - \\ & \vdots & \\ - & \boldsymbol{\phi}_k^T & - \end{bmatrix}}_{k \times d} \underbrace{[\,\mathbf{t}_i - \hat{\boldsymbol{\mu}}\,]}_{d-\text{dimensional}}$$

Principal Components

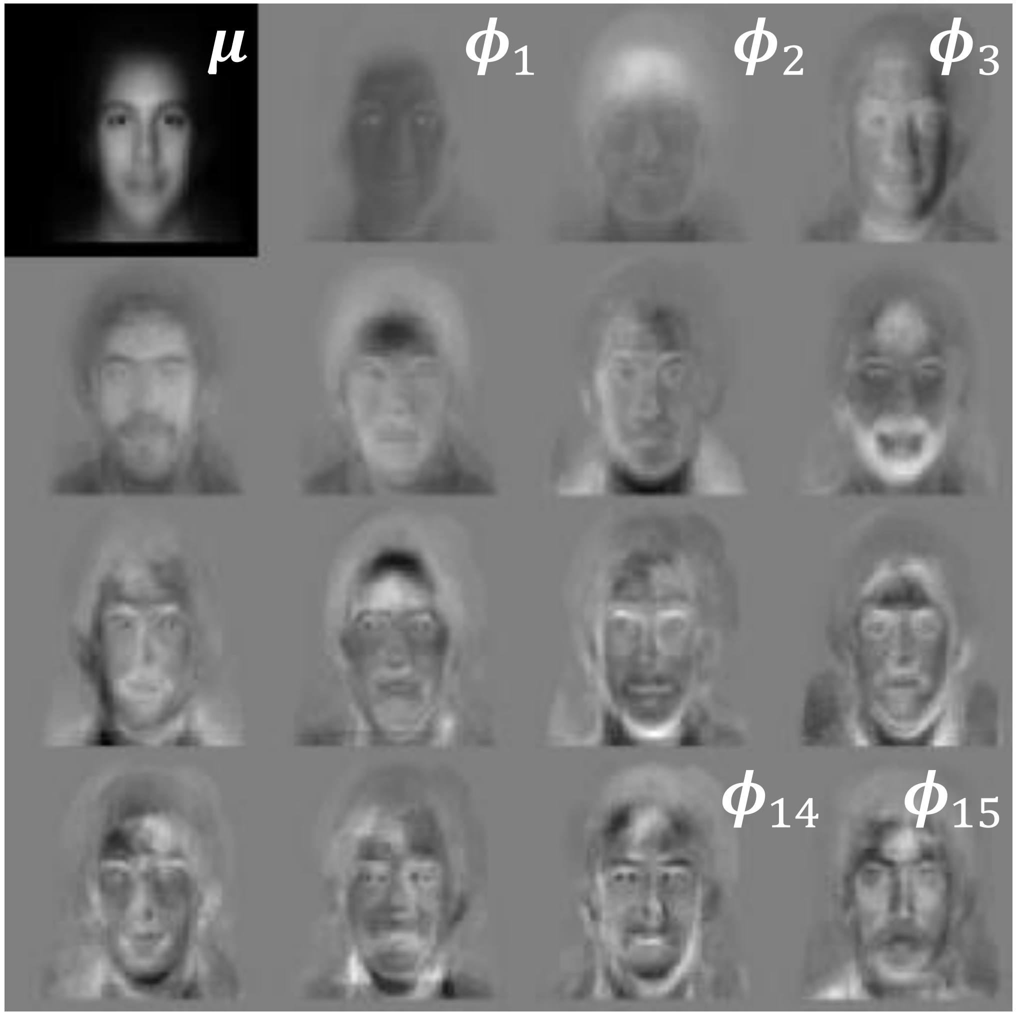
- what are they? in some cases it is possible to see
- example: eigenfaces
  - face recognition problem: can you identify who is the person in this picture?
  - training:
    - assemble examples from people's faces
    - compute the PCA basis
    - project each image into PCA space
    - use image projections to learn a classifier in the PCA space
  - recognition:
    - project image to classify into PCA space
    - apply the classifier in the PCA space

Principal Components

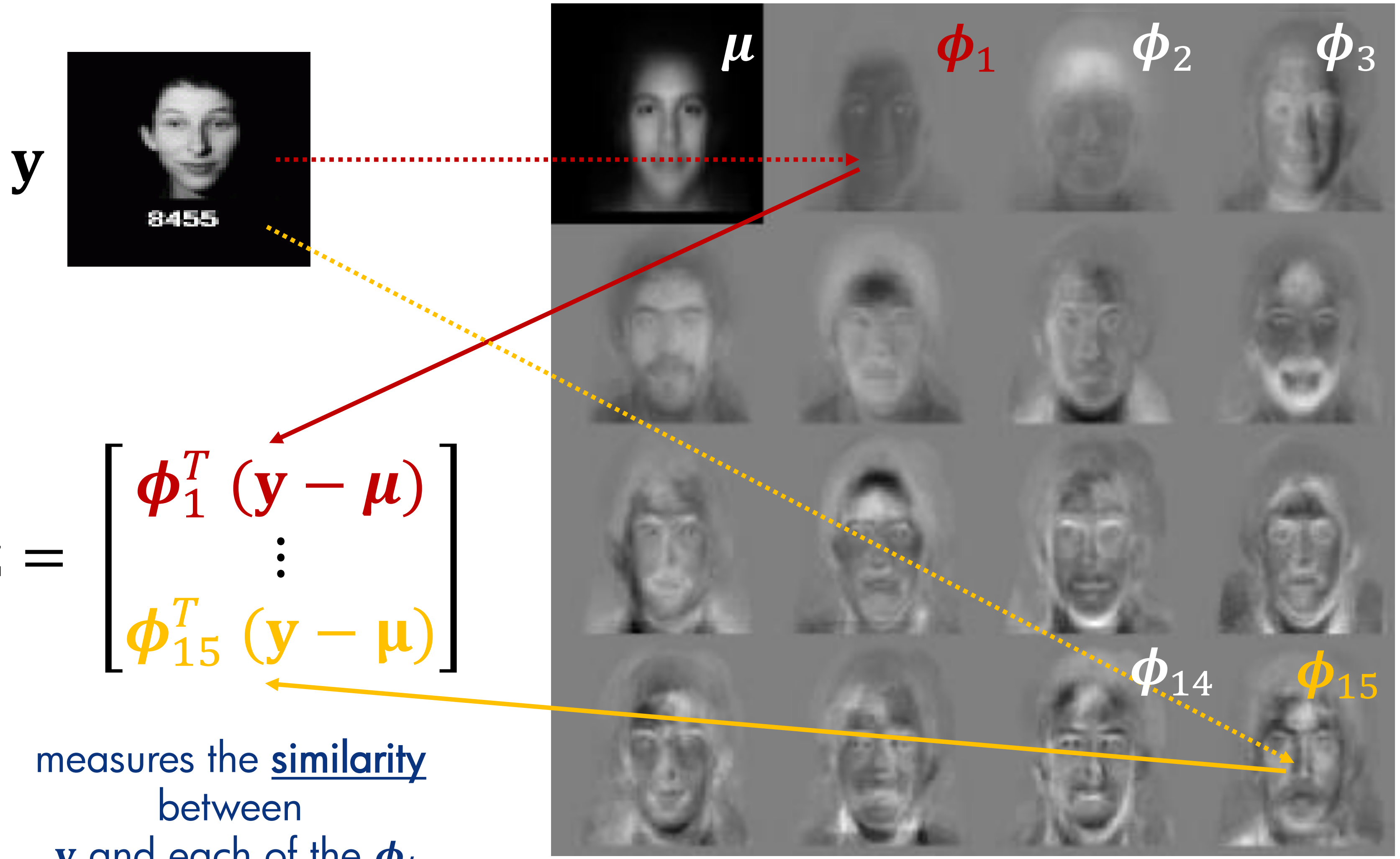
- face examples

Principal Components

- principal components (eigenfaces)
  - high–variance ones tend to have low–frequency
  - capture average face, illumination, etc.
  - at the intermediate–level, we have face detail
  - low–variance tends to be high–frequency noise



Principal Component Analysis



$$\mathbf{z} = \begin{bmatrix} \boldsymbol{\phi}_1^T (\mathbf{y} - \boldsymbol{\mu}) \\ \vdots \\ \boldsymbol{\phi}_{15}^T (\mathbf{y} - \boldsymbol{\mu}) \end{bmatrix}$$

measures the <u>**similarity**</u>
between
$\mathbf{y}$ and each of the $\boldsymbol{\phi}_i$

Functional Derivatives

Recall

- a function $f(w)$ is **differentiable** if it has derivatives for all w
- the derivative at point w is defined as

$$\frac{\partial f}{\partial w} = \lim_{\alpha \to 0} \frac{f(w+\alpha) - f(w)}{\alpha}$$

- for a **multivariate function** $f(\mathbf{w}), \mathbf{w} \in \mathbb{R}^n$, the directional derivative of $f(\mathbf{w})$ at point $\mathbf{w}$, along **direction** $\mathbf{u}$ is

$$D_{\mathbf{u}} f(\mathbf{w}) = \lim_{\alpha \to 0} \frac{f(\mathbf{w}+\alpha\mathbf{u}) - f(\mathbf{w})}{\alpha}$$

- if the function is differentiable, this can be written as

$$D_{\mathbf{d}} f(\mathbf{w}) = \mathbf{d}^T \nabla f(\mathbf{w})$$

where

$$\nabla f(\mathbf{z}) = \left(\frac{\partial f}{\partial w_0}(\mathbf{z}), \cdots, \frac{\partial f}{\partial w_{n-1}}(\mathbf{z}) \right)^T$$

is the gradient of a function $f(\mathbf{w})$ at $\mathbf{z}$

Functions of Functions

- all these concepts extend to functions of functions, if we consider a function as a very long vector
- consider the vector $w = (w_1, \dots, w_n)$ and **increase n until infinity** → this is an infinite−dimensional **vector**
- note that we can write the vector elements as $w(x), x \in \{1, \dots, \infty\}$
- the next step is to **make x continuous**, i.e. consider the infinite−dimensional vector of elements $w(x), x \in \mathbb{R}$
- we call this a function, but there is no fundamental difference
  - in fact, if you define function addition and scalar−function multiplication in the standard way, the space of functions is a vector space
  - we can then define the inner−product between functions as

$$\langle w, u \rangle = \int w(x)u(x)\, dx$$

which generalizes the standard dot−product

$$\langle w, u \rangle = \sum_x w_x u_x$$

Functional Derivative

- consider now a function of a function $F[w(x)]$
- the **directional derivative** of F at point $w(x)$, along (function) direction u is

$$D_u F[w(x)] = \lim_{h\to 0} \frac{F[w(x) + hu(x)] - F[w(x)]}{h}$$

  - this **measures** how much the **function** F **grows** if we give an infinitesimal step along direction defined by function u
  - note that you should think of the function $u(x)$ as a **vector**, not as "the value of u at point x"

- there is an **alternative** definition,

$$\boxed{D_u F[w(x)]} = \left[\lim_{h\to 0} \frac{F[w(x) + (\varepsilon + h)u(x)] - F[w(x) + \varepsilon u(x)]}{h}\right]_{\varepsilon=0}$$

$$= \boxed{\left[\frac{d}{d\varepsilon} F[w(x) + \varepsilon u(x)]\right]_{\varepsilon=0}}$$

that only requires computing a scalar derivative

Example

- for **example**, if

$$F[w(x)] = \int w(x) \log w(x) \, dx$$

then, the derivative of F along the direction $\delta(x-v)$ is

$$\begin{aligned}
D_{\delta(x-v)} F[w(x)] &= \left[\frac{d}{d\varepsilon} F[w(x) + \varepsilon\delta(x-v)]\right]_{\varepsilon=0} \\
&= \left[\frac{d}{d\varepsilon} \int [w(x) + \varepsilon\delta(x-v)] \log[w(x) + \varepsilon\delta(x-v)] \, dx\right]_{\varepsilon=0} \\
&= \left[\int \delta(x-v) \log[w(x) + \varepsilon\delta(x-v)] \, dx + \int [w(x) + \varepsilon\delta(x-v)] \frac{\delta(x-v)}{w(x) + \varepsilon\delta(x-v)} \, dx\right]_{\varepsilon=0} \\
&= \log w(v) + 1
\end{aligned}$$

Functional Derivative

- the derivatives $D_{\delta(x-v)}F(w)$ are important because they measure the variation of $F[w(x)]$ along the directions $\delta(x-v)$
- since these directions are the infinite–dimensional extensions of the canonical vectors, these derivatives are the extensions of the partials $\partial f(w)/\partial w_i$
- hence, the **gradient** is the infinite–dimensional vector of such derivatives
  - e.g. $F[w(x)] = \int w(x) \log w(x) dx$ has gradient $\nabla_w \mathrm{F} = \log w(x) + 1$

- the derivation of the previous page is a bit complicated: there is an **easier way** to think about this
- each the "entry" $D_{\delta(x-v)}F(w)$ of the gradient is the **derivative along** $\delta(x-v)$: it only depends on $w(v)$, not on the values of $w(x), x \neq v$
- hence, we can get rid of the integral, and simply compute

$$\boxed{\frac{\partial F}{\partial w(v)} = \frac{\partial}{\partial w(v)}[w(v) \log w(v)] = \log w(v) + 1}$$

- it follows that $\nabla_w \mathrm{F} = \log w(x) + 1$