# 2023

# 22ND ANNUAL

PARIS C. KANELLAKIS

MEMORIAL LECTURE

"Monitoring Health and Diseases Using Radio Signals and Machine Learning"

**Dina Katabi**

Thuan and Nicole Pham Professor
of Electrical Engineering and Computer Science, MIT

4 PM on April 20 • CIT 368

Fill in the mid-term feedback for extra 2 late days!

No quiz this week!

Instructor's office hours cancelled this week!

CSCI 1470/2470
Spring 2023

Ritambhara Singh

April 03, 2023
Monday

# Deep Learning

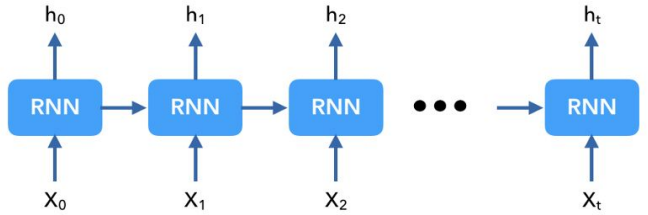
DALL-E 2 prompt "a painting of deep underwater with a yellow submarine in the bottom right corner"

# Interpretation in DL

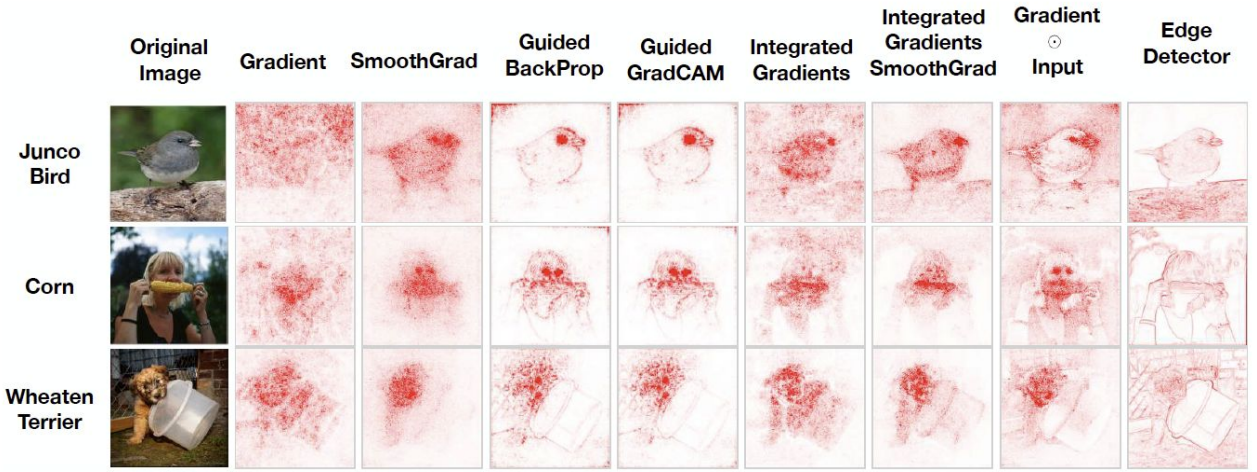
(1) Model architecture based methods

(2) Gradient-based methods

**(3) Model agnostic methods**

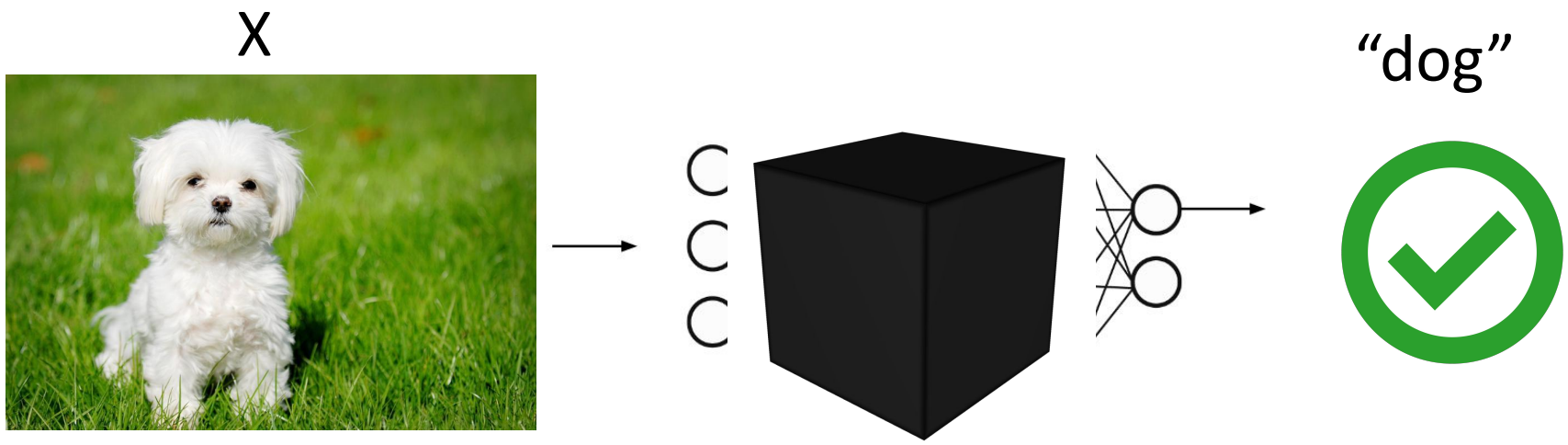






What is the simplest thing that comes to mind?

# Task formulation

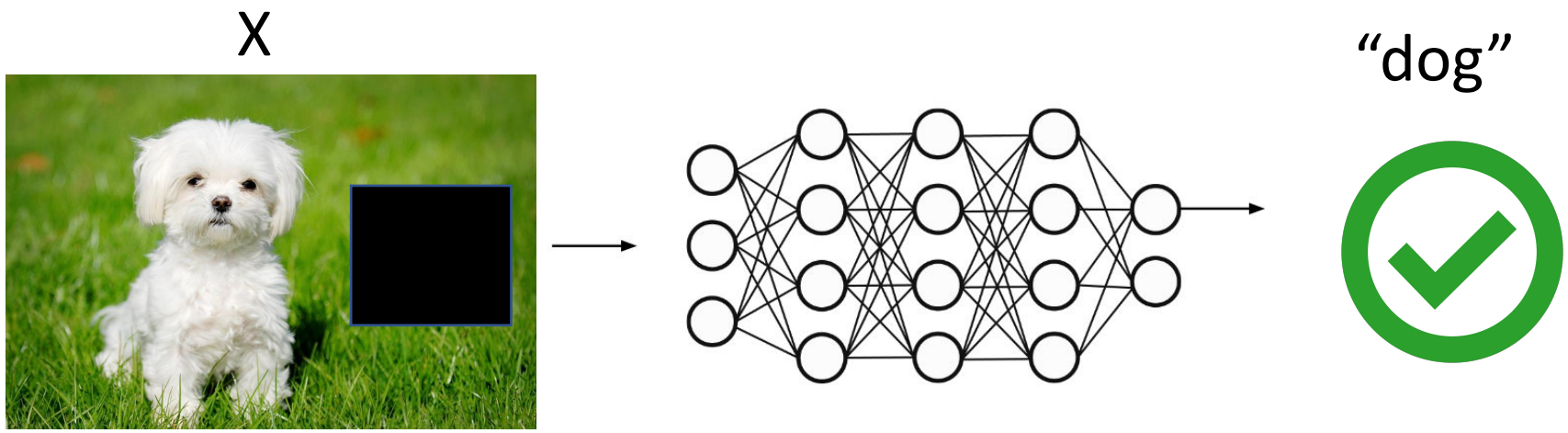



Which pixels are most important for classification?

# Perturbation-based methods

- Let's **perturb inputs**...
- omit or change words/parts of images, change word embedding values, etc.

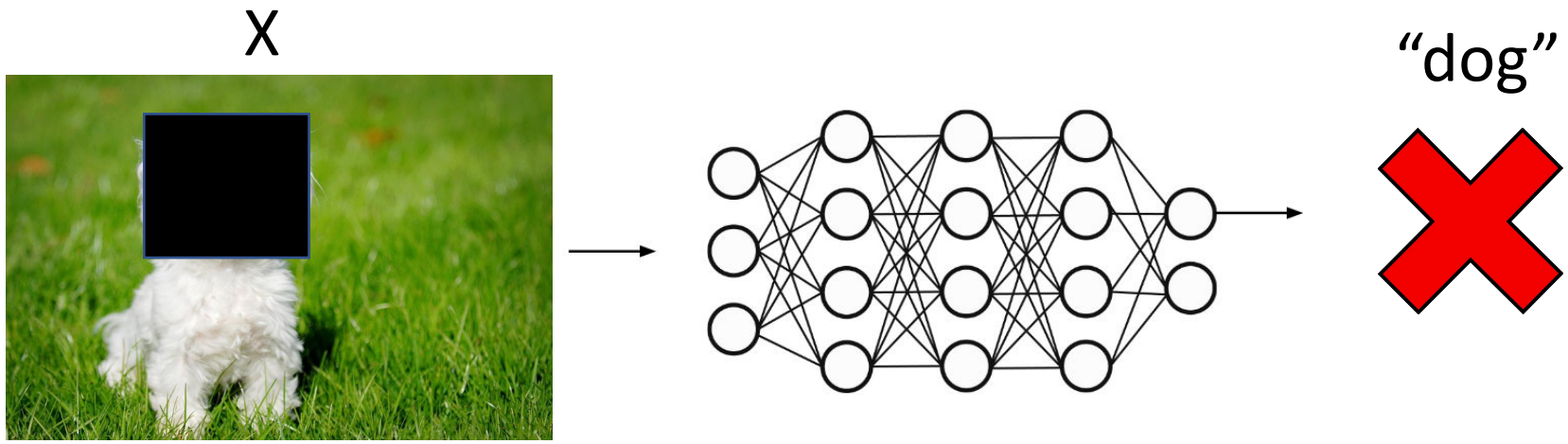
●...**observe changed outputs**...



# Perturbation-based methods

- Let's **perturb inputs**...
- omit or change words/parts of images, change word embedding values, etc.

●...**observe changed outputs**...



# LIME

- Local Interpretable Model-Agnostic Explanations (LIME)
  - “Model-Agnostic”: treats every model as a black-box
- Let’s **perturb inputs**...
  - omit or change words/parts of images, change word embedding values, etc.
- ...**observe changed outputs**...
- ...and **approximate the underlying model** using a simple, interpretable model (like a linear classifier)
  - Interpretable because in $\sum w_i x_i$, the weights say “how much a particular input matters”

# LIME example

- What makes this picture of a tree frog “tree frog”-y to a neural network?

Original Image

# LIME example

- Perform a superpixel segmentation on the image
- Interpretable chunks in the image may be part of multiple superpixels
- But no superpixel will contain multiple interpretable parts

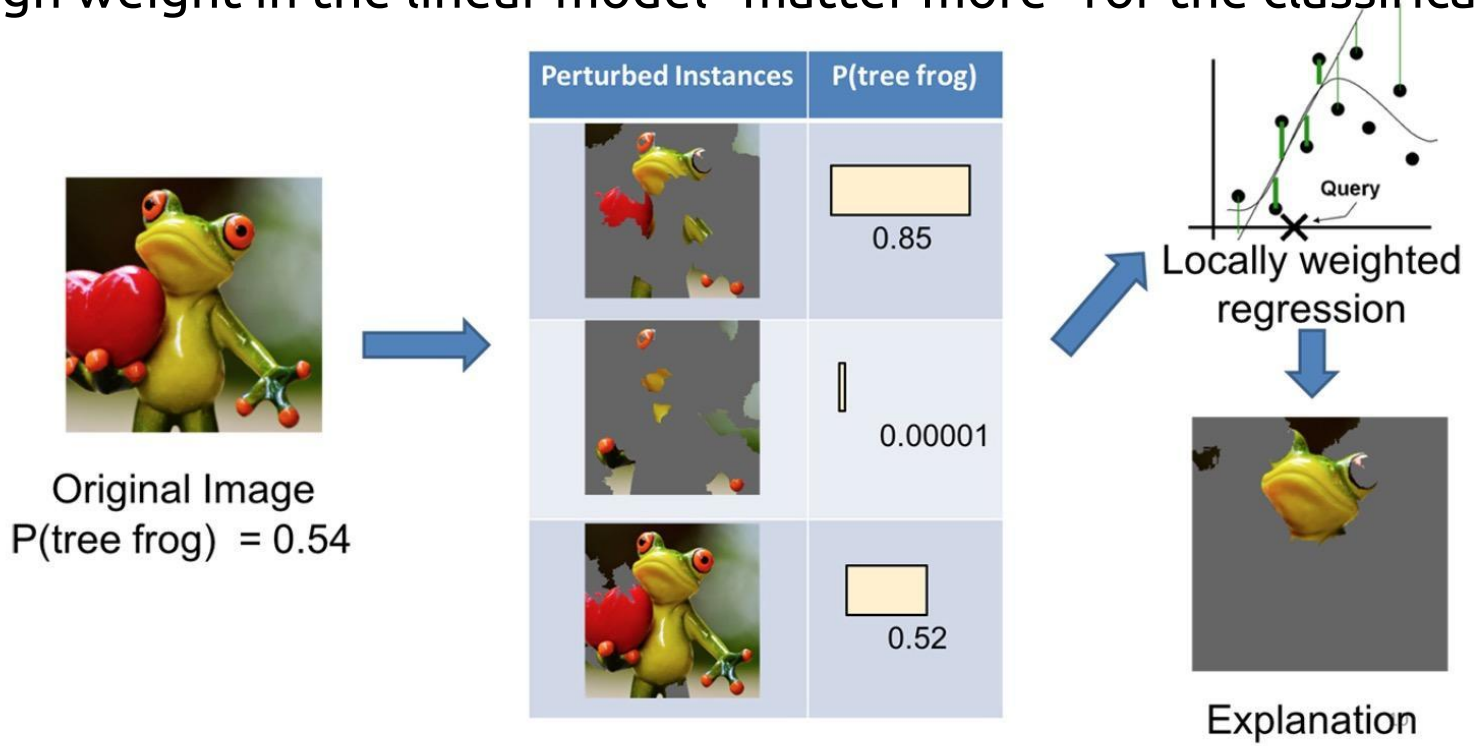
Original Image

Interpretable Components

Can we think of disadvantages of perturbation-based methods?

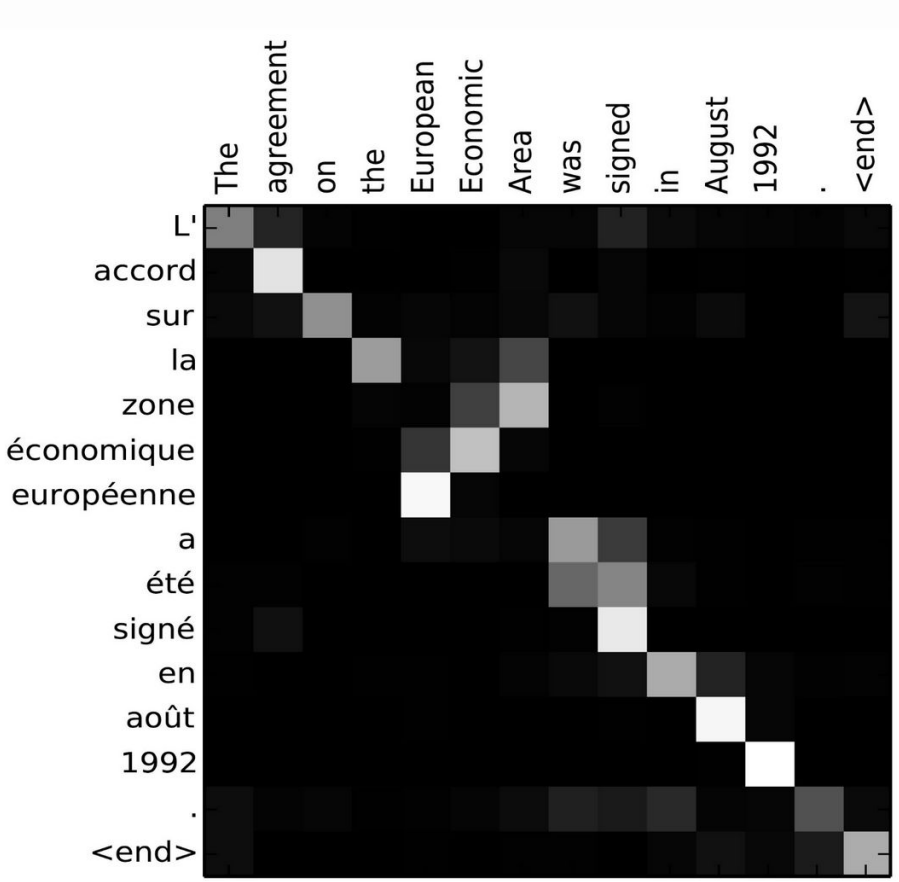
# LIME example

- Different combinations of chunks put through the net yield different probabilities
- Learn a linear model to predict the probability from these different combos
- Chunks with high weight in the linear model “matter more” for the classification result



Any questions?

# Attention as interpretation method




A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

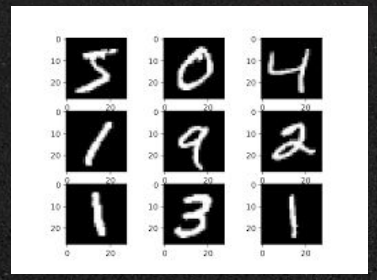

A stop sign is on a road with a mountain in the background.

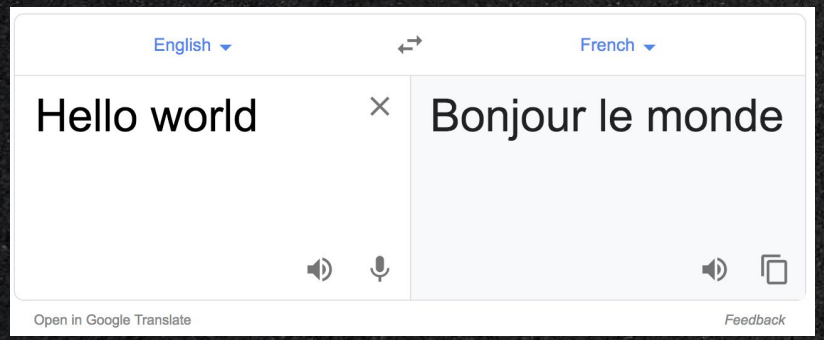
A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

# Roadmap




Supervised machine learning

Perceptron

Fully Connected Neural Networks

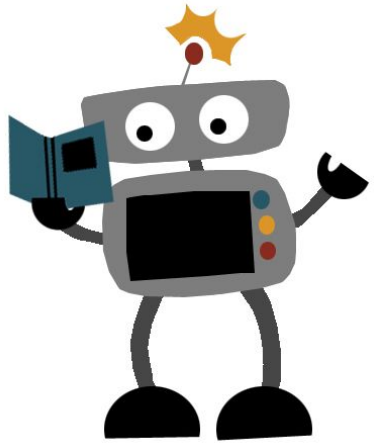
Convolutional Neural Networks

Language models

Recurrent Neural Networks

Transformers (Seq2seq)

# Recap: What is Machine Learning?



Input: X

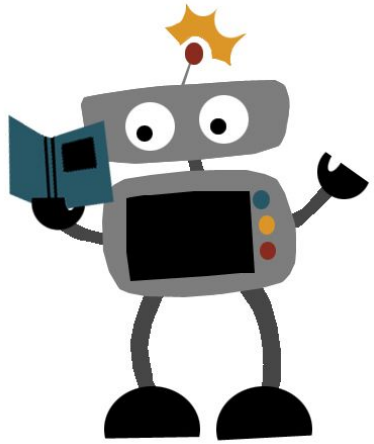
Output: Y

”Cooking?”

Function: f

f(X)□Y

# Recap: What is Machine Learning?



**Supervised Learning**

Input: X

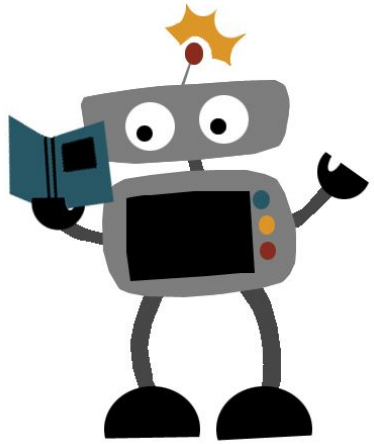
Learned function: f

Output: Y

”Cooking?”

$f(X) \rightarrow Y$

# What if you don’t have any labels?



Input: X

Unsupervised Learning

What can you learn from just input data without labels?

# Today’s goal – learn about unsupervised learning using deep learning models

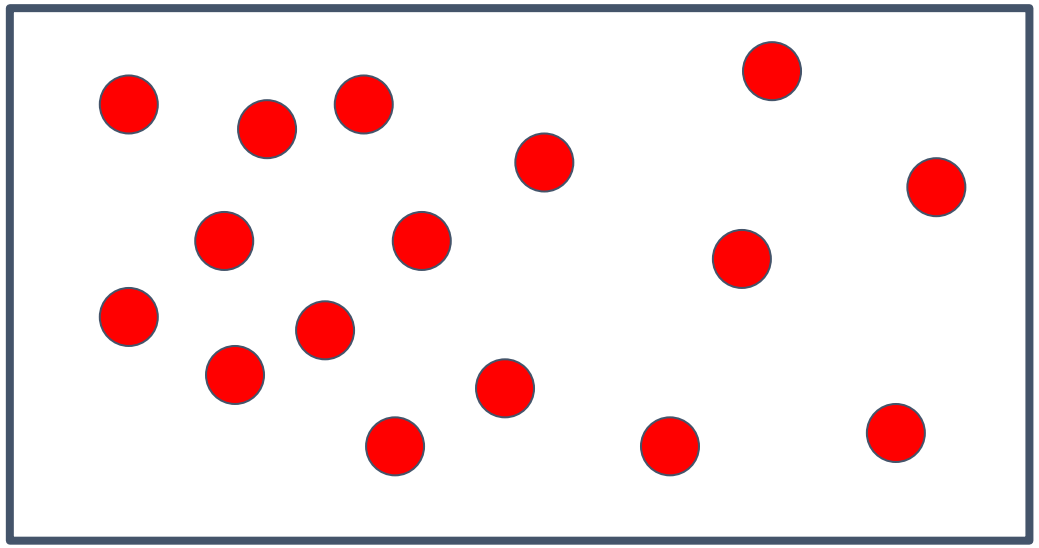
(1) Unsupervised Learning

(2) Auto-encoders (AE)

# Unsupervised Learning

- What can we learn from input data when there are no labels?
  - We can only analyze the structure of the data itself

Data:

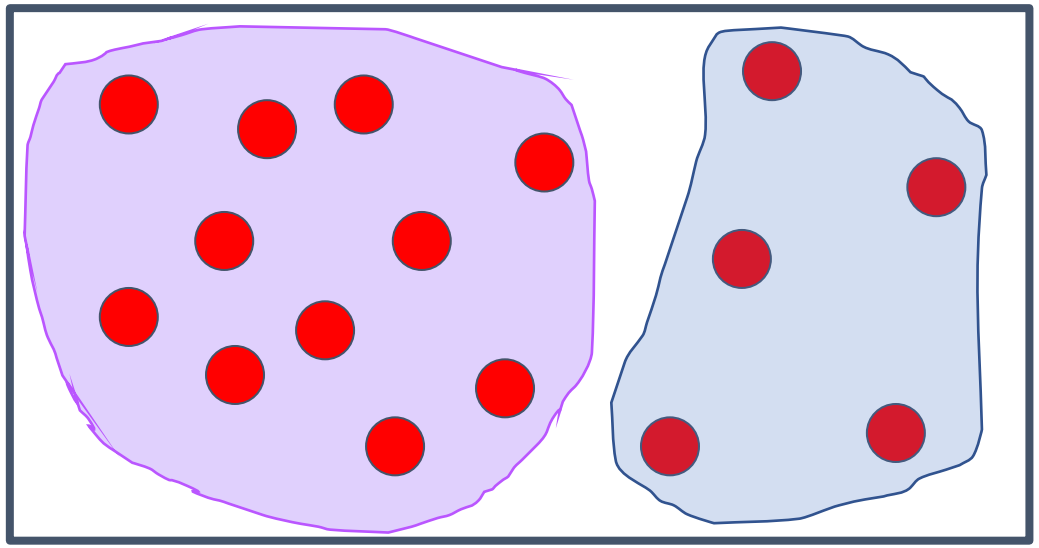


# Clustering

The organization of unlabeled data into similarity groups called “clusters.”

A cluster is a collection of data items which are “similar” between them, and “dissimilar” to data items in other clusters.

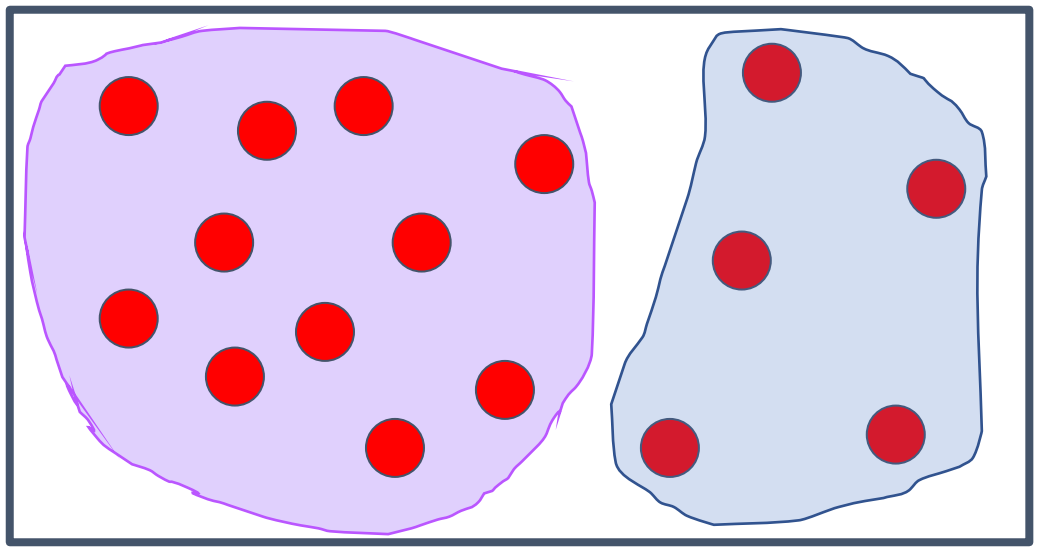
Data:



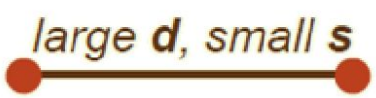
?

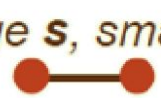
# How does the machine do the clustering?

Data:

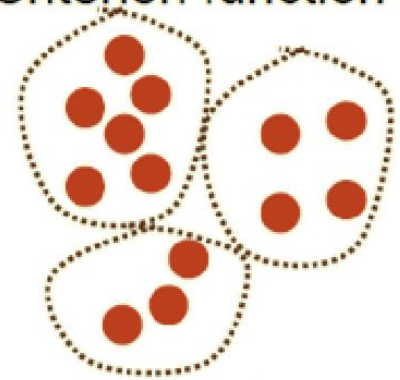


1. Proximity measure, *either*
   - similarity measure $s(x_i,x_k)$: large if $x_i,x_k$ are similar
   - dissimilarity(or distance) measure $d(x_i,x_k)$: small if $x_i,x_k$ are similar

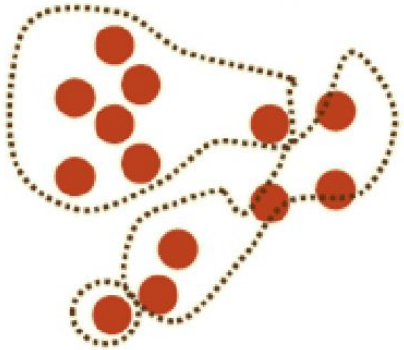



2. Criterion function to evaluate a clustering



*good clustering*



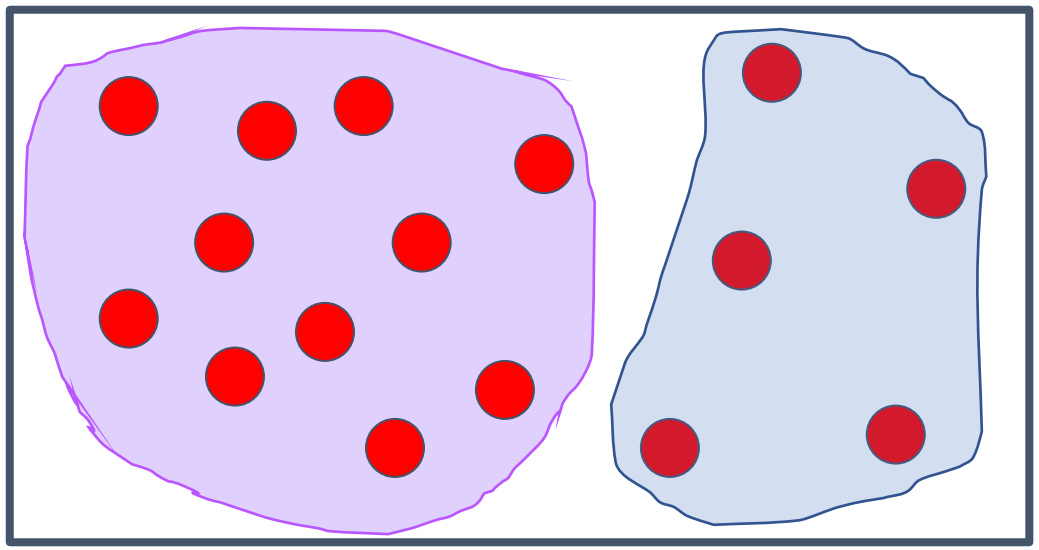
*bad clustering*

3. Algorithm to compute clustering
   - For example, by optimizing the criterion function

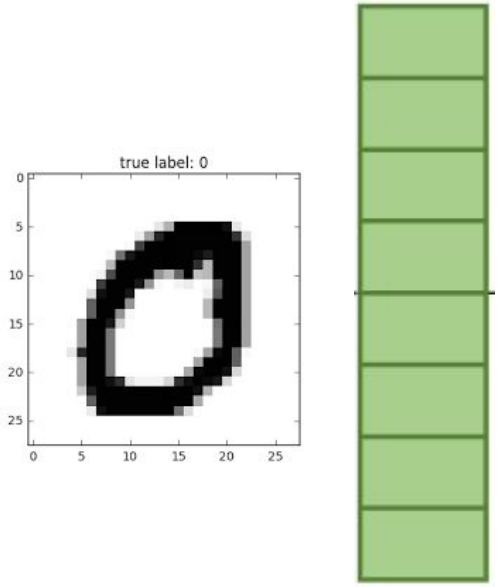
http://www.mit.edu/~9.54/fall14/slides/Class13.pdf

# Data in high dimension

Data:



What about an image?



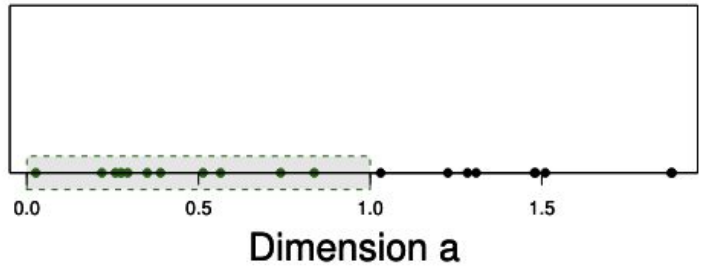

$\mathbb{R}^{784}$

http://www.mit.edu/~9.54/fall14/slides/Class13.pdf

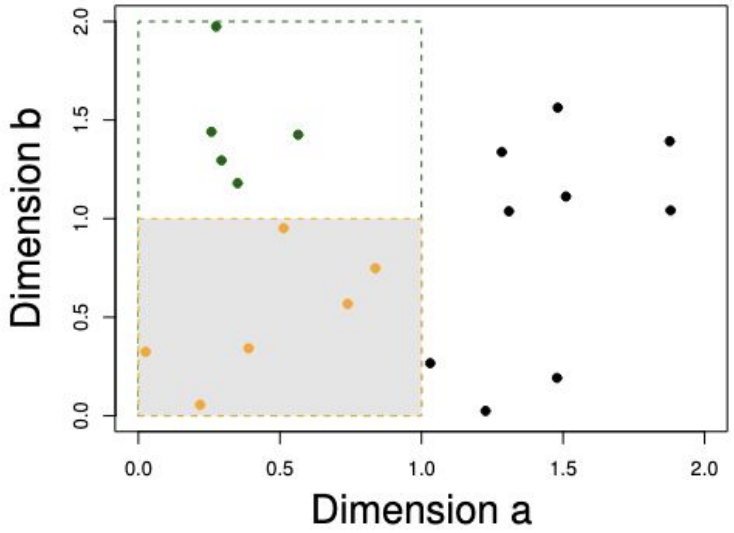
# Curse of dimensionality in clustering

What can we do?

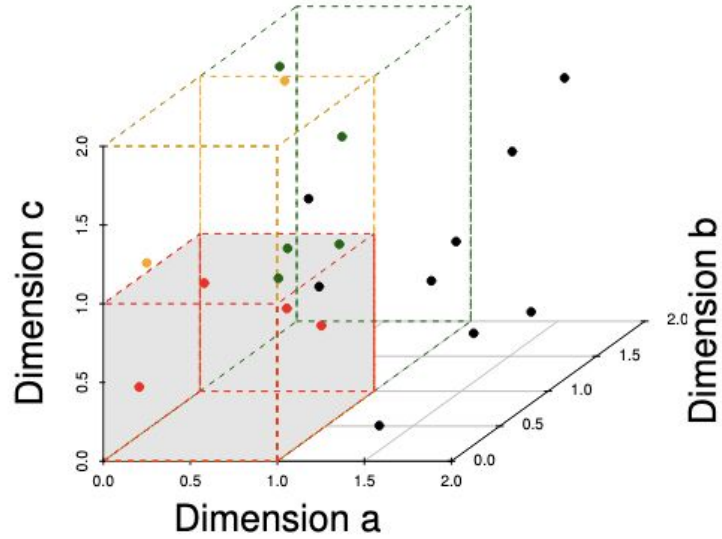
Adding a dimension stretches the points across that dimension, pushing them further apart.




(a) 11 Objects in One Unit Bin




(b) 6 Objects in One Unit Bin




(c) 4 Objects in One Unit Bin

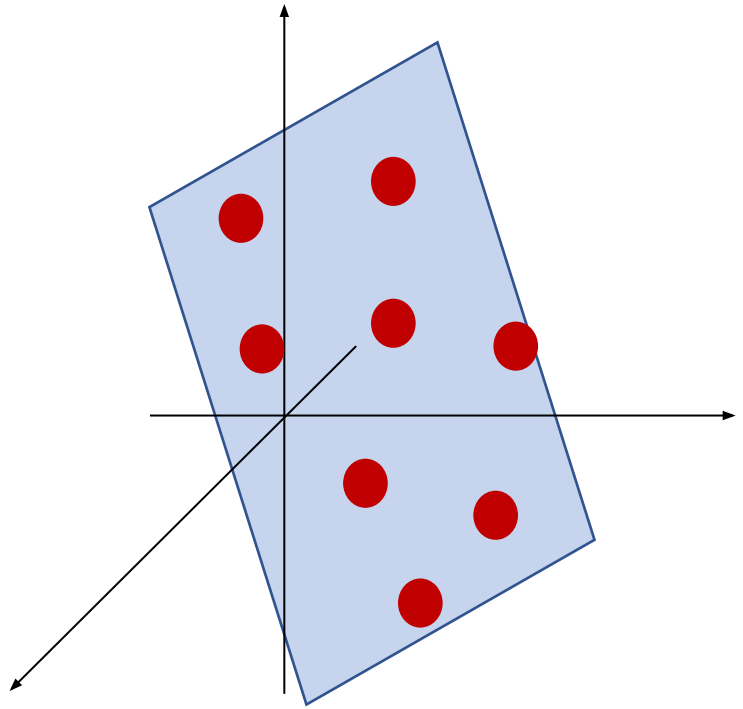
The points continue to spread out, when more dimensions are being added, until they are equidistant from each other and distance is not very meaningful.

https://www.kdd.org/exploration_files/parsons.pdf

https://delta1epsilon.github.io/2019/Gotchas-with-clustering/

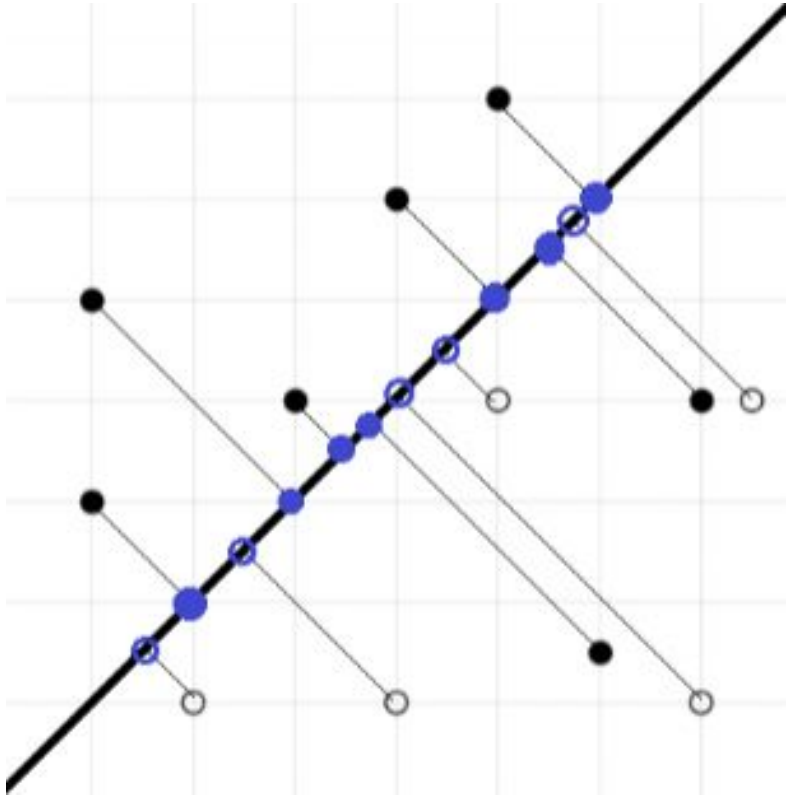
# Dimensionality Reduction

- Represent the data with fewer dimensions
- The key idea: While the data may exist in a high dimensional space, it may actually lie along a lower dimensional subspace
  - Ex: data in $\mathbb{R}^3$ may lie along a plane
  - i.e. the ***intrinsic dimensionality*** of the data is actually 2 (not 3)



# Dimensionality Reduction using projection

- Data may not lie exactly on a lower-dimensional subspace
- Can still represent it fairly well (with some degree of error)



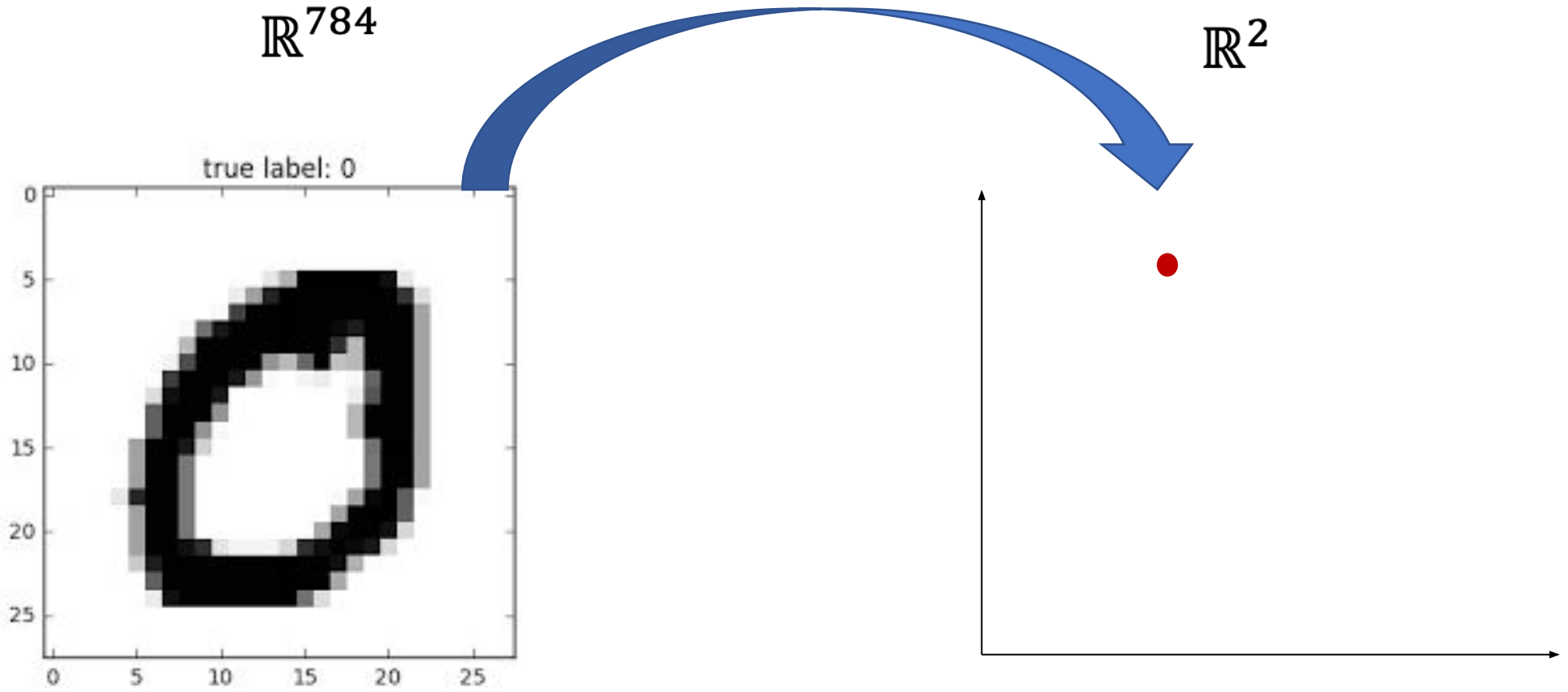
2D data projected to 1 dimension

Any questions?

# Dimensionality Reduction: Why?

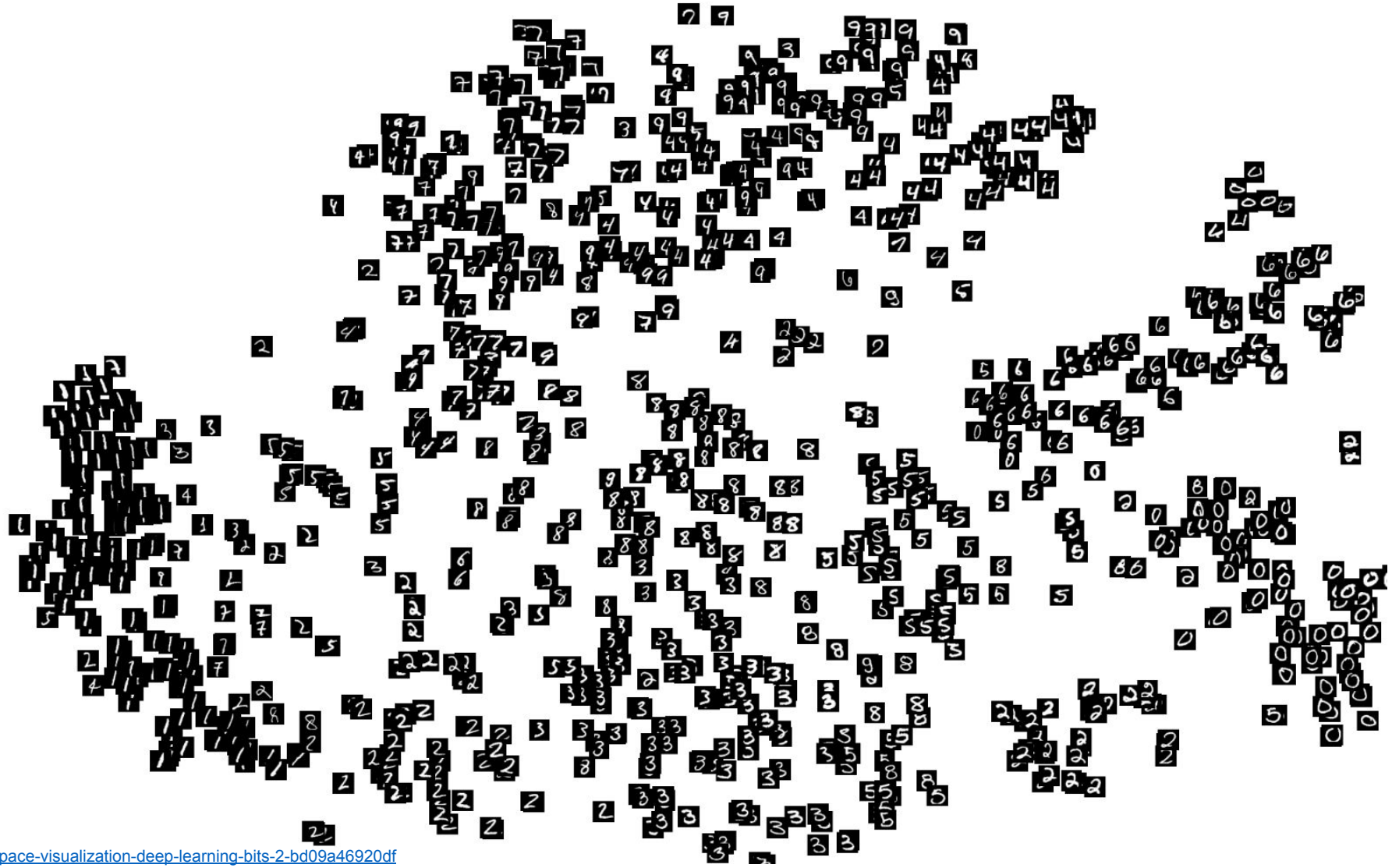
- Lots of benefits to making the data lower-dimensional
  - Many clustering algorithms behave better in lower dimensions
  - Takes less storage/memory  if you're trying to analyze a huge dataset
  - More efficient to search using approximate nearest neighbor algorithms
  - Easier to ***visualize*** (if you reduce the data to 2 or 3 dimensions)

# Dimensionality Reduction: Visualization



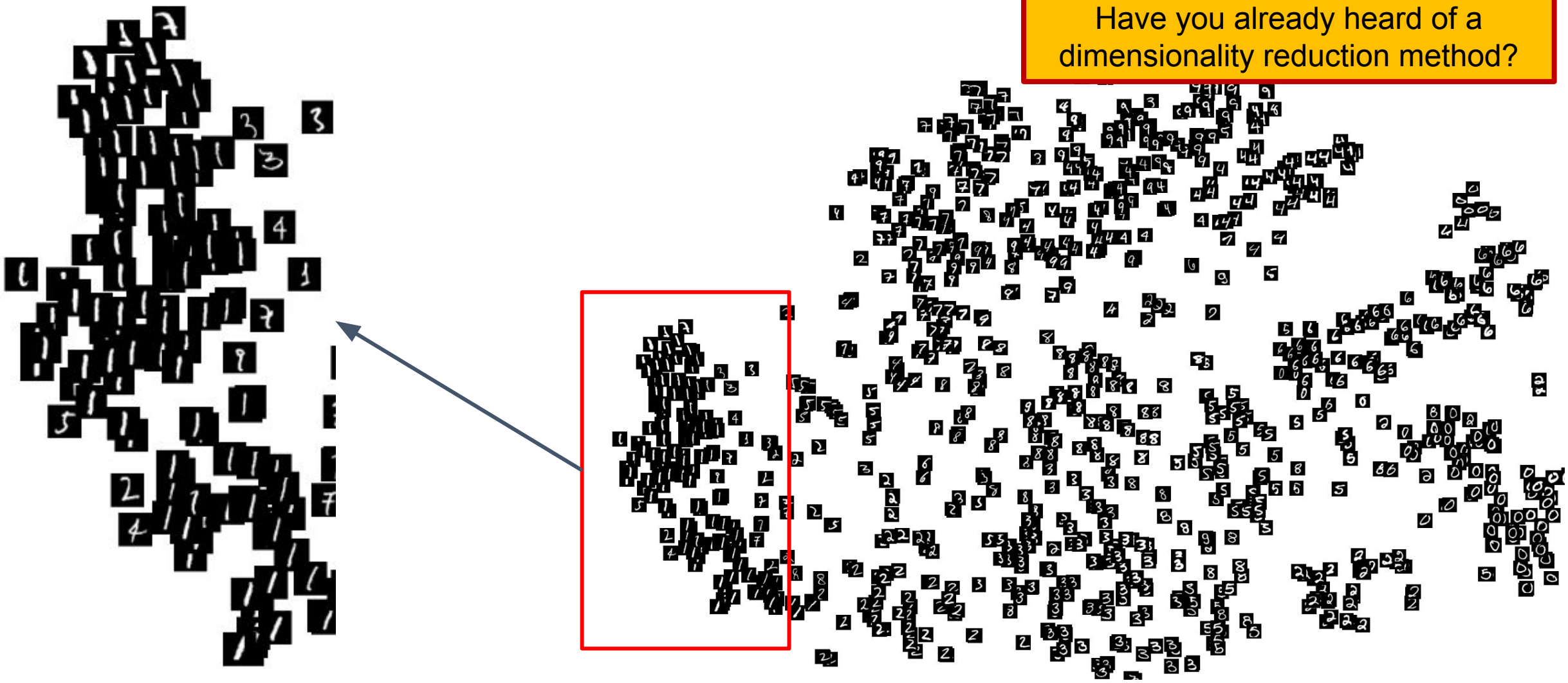

https://hackernoon.com/latent-space-visualization-deep-learning-bits-2-bd09a46920df

# Dimensionality Reduction: Visualization



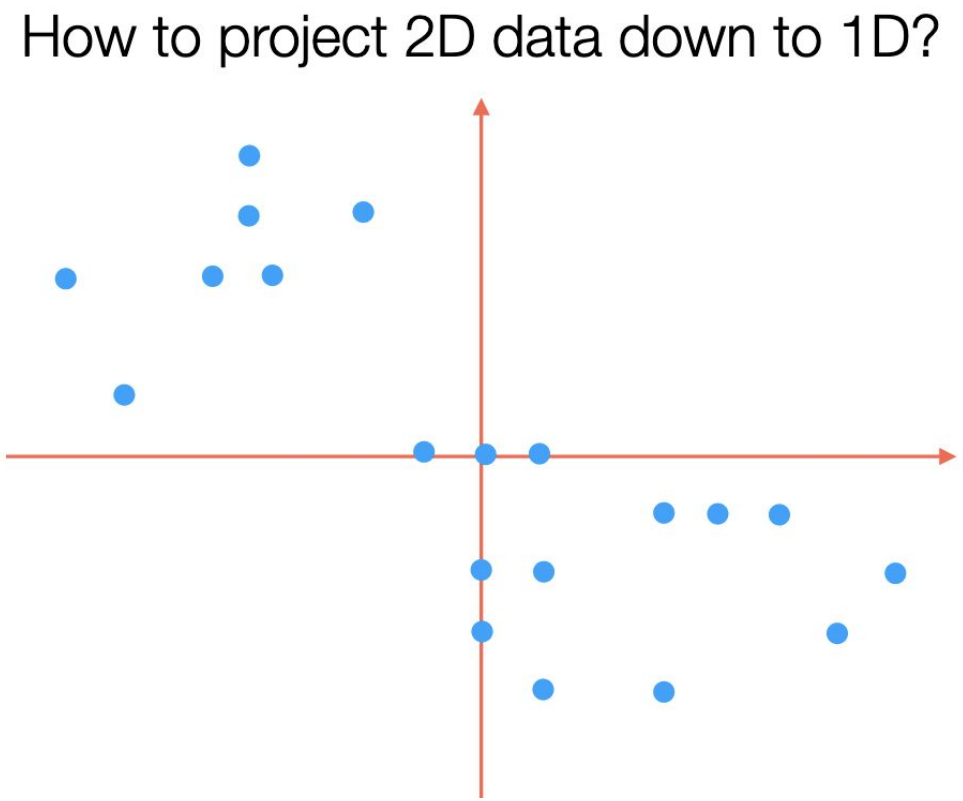
https://hackernoon.com/latent-space-visualization-deep-learning-bits-2-bd09a46920df

# Dimensionality Reduction: Visualization



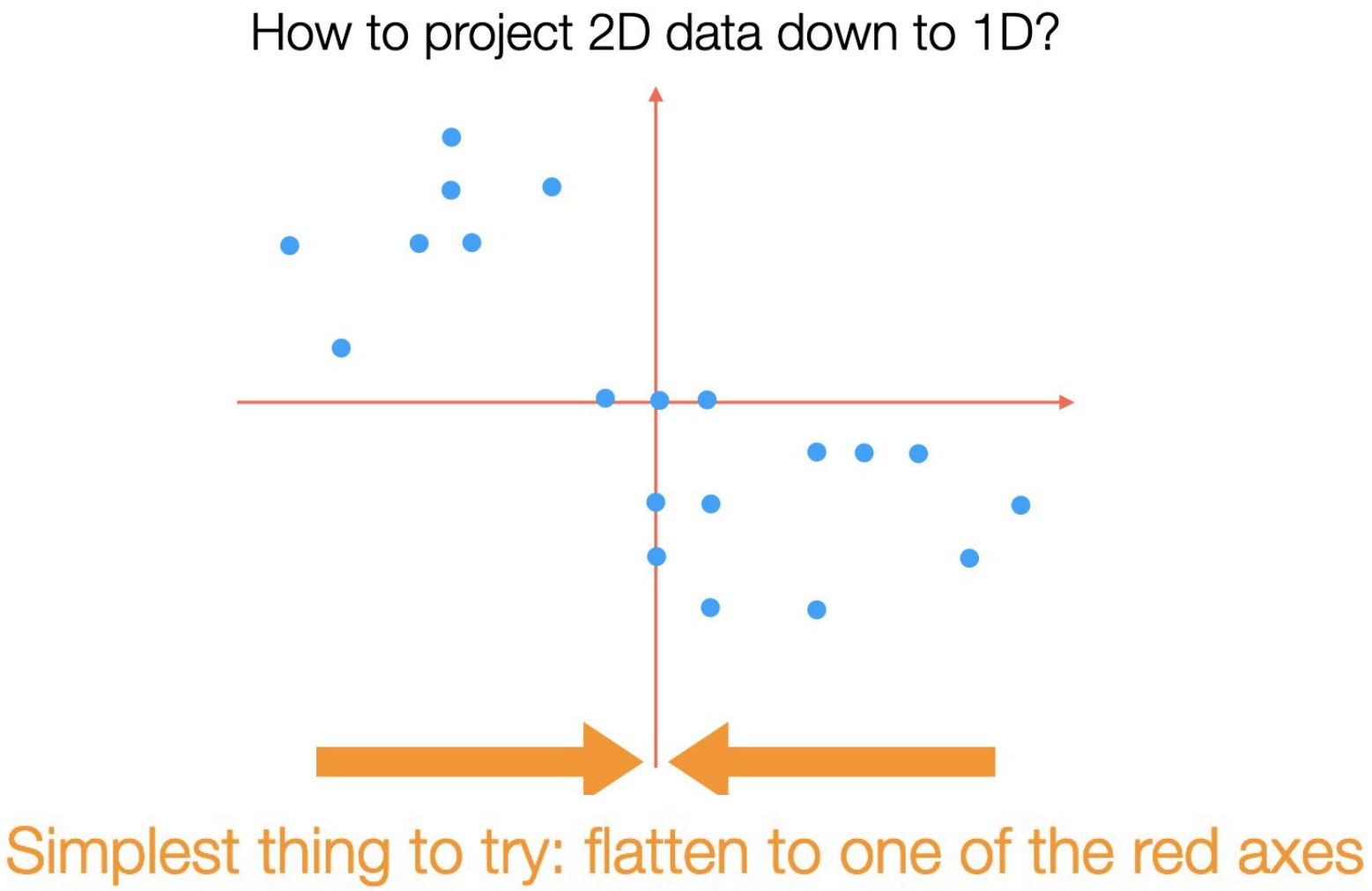

https://hackernoon.com/latent-space-visualization-deep-learning-bits-2-bd09a46920df

# Principal Component Analysis (PCA)




https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

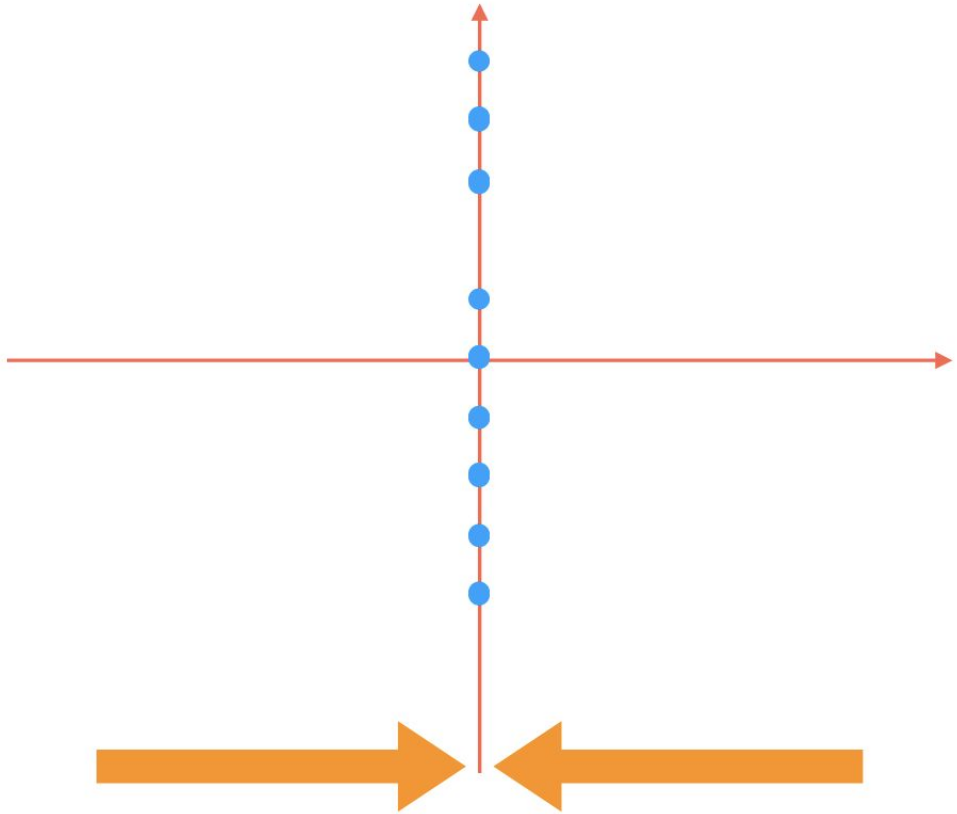
# Principal Component Analysis (PCA)




https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

# Principal Component Analysis (PCA)

How to project 2D data down to 1D?



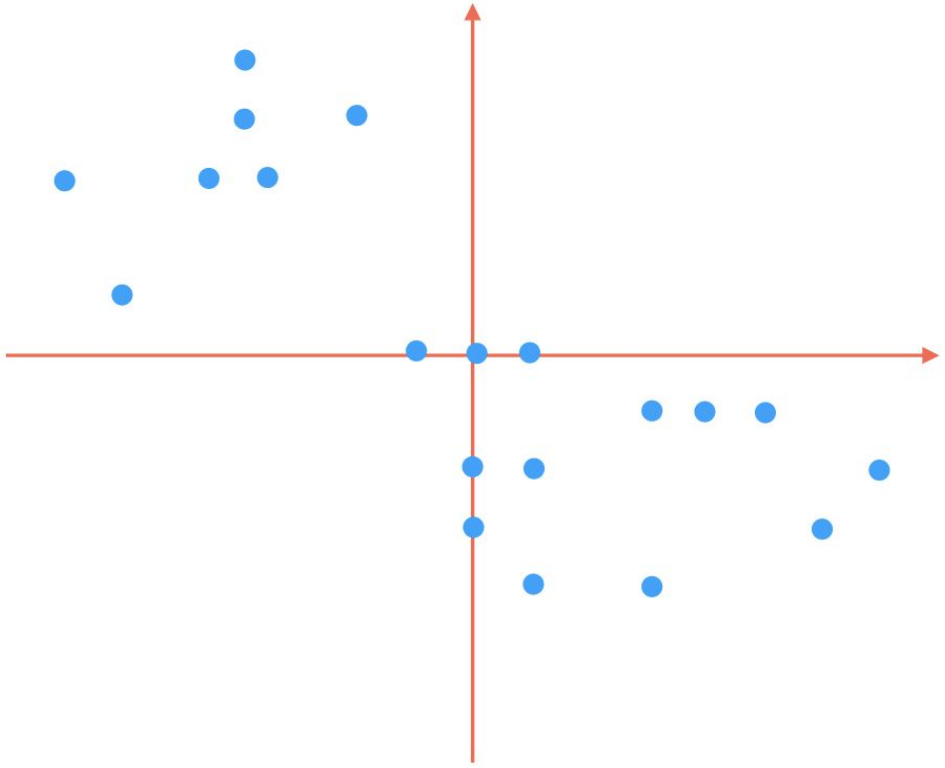
Simplest thing to try: flatten to one of the red axes

(We could of course flatten to the other red axis)

https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

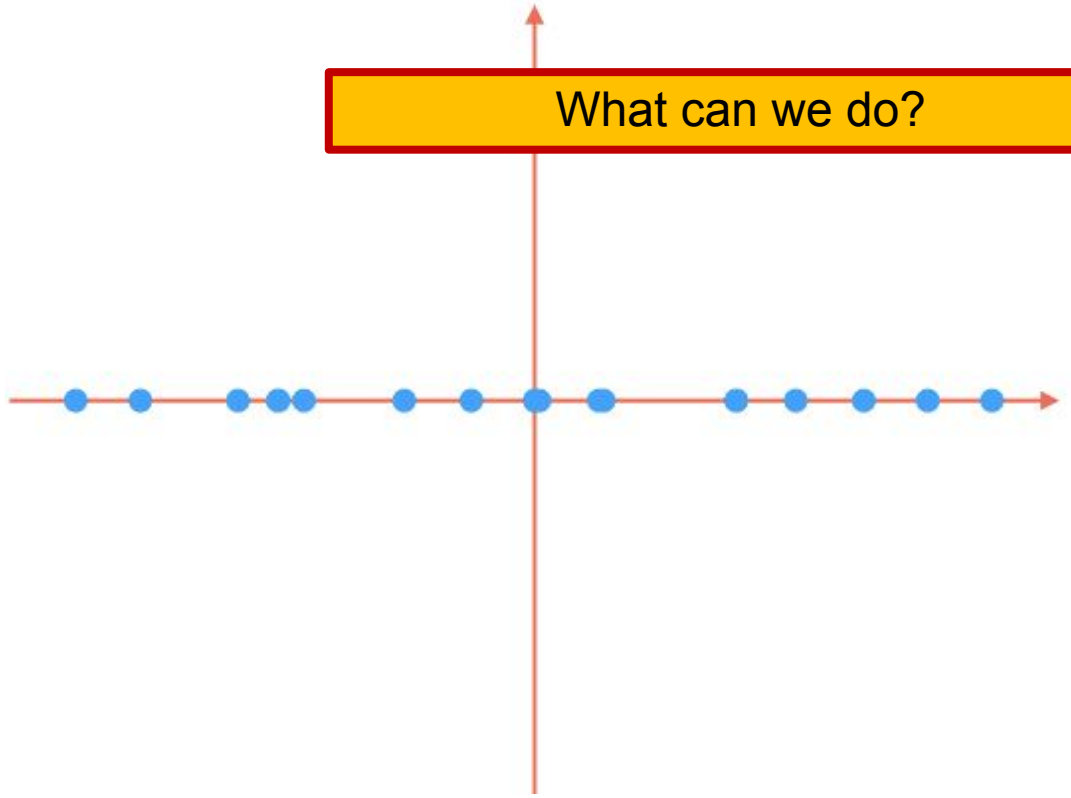
# Principal Component Analysis (PCA)

How to project 2D data down to 1D?



What is the issue here?

What can we do?

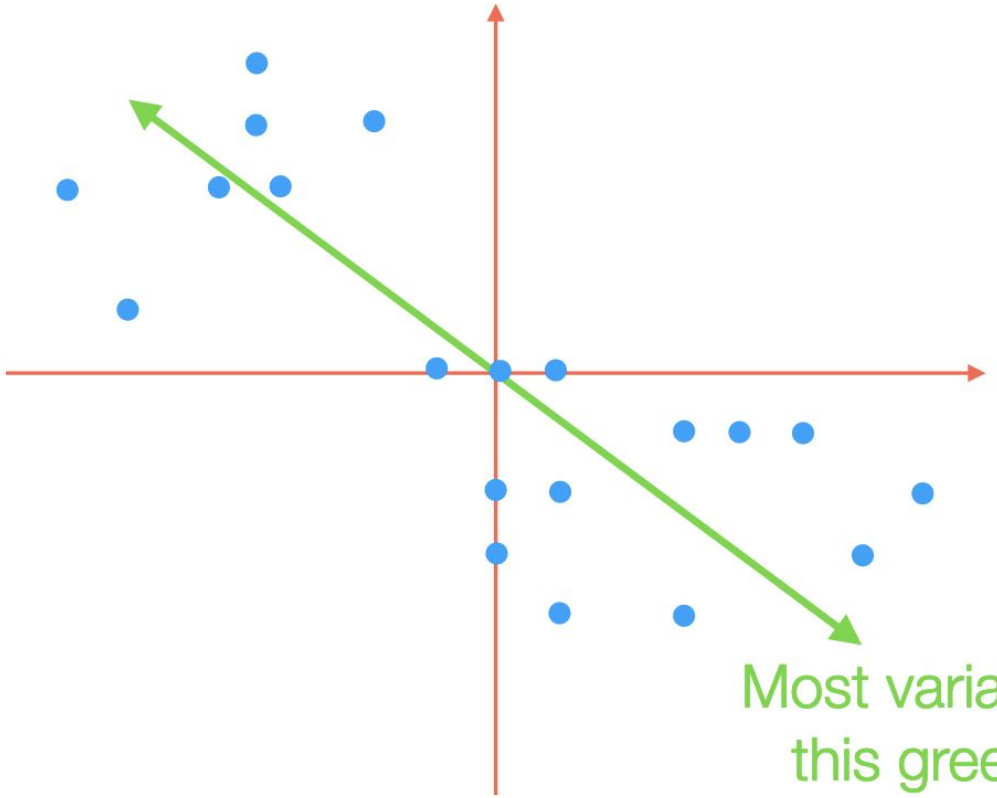


But notice that most of the variability in the data is *not* aligned with the red axes!

https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

# Principal Component Analysis (PCA)

How to project 2D data down to 1D?

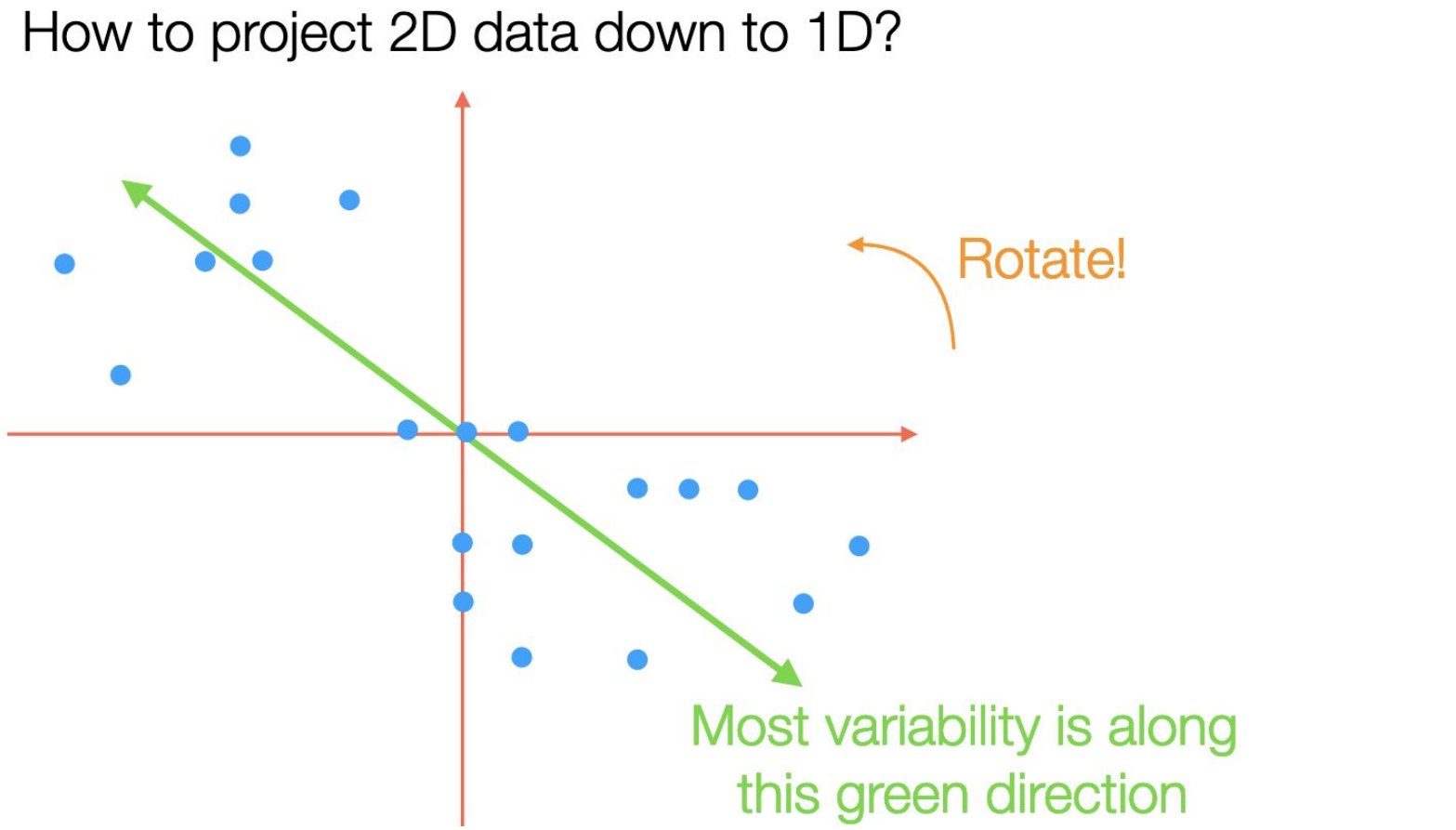


Most variability is along this green direction

But notice that most of the variability in the data is *not* aligned with the red axes!

https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

# Principal Component Analysis (PCA)

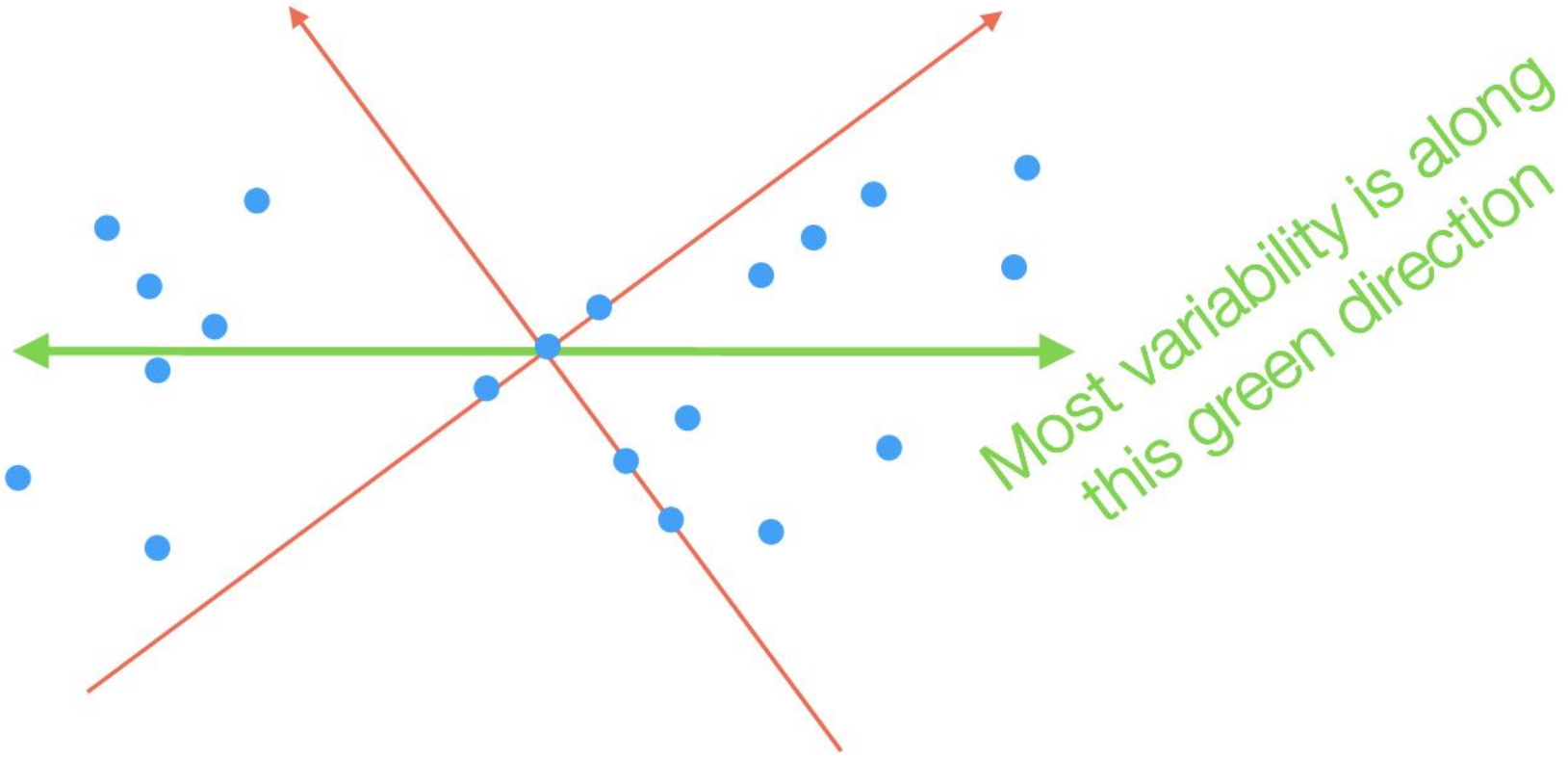



But notice that most of the variability in the data is *not* aligned with the red axes!

https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

# Principal Component Analysis (PCA)

How to project 2D data down to 1D?




https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

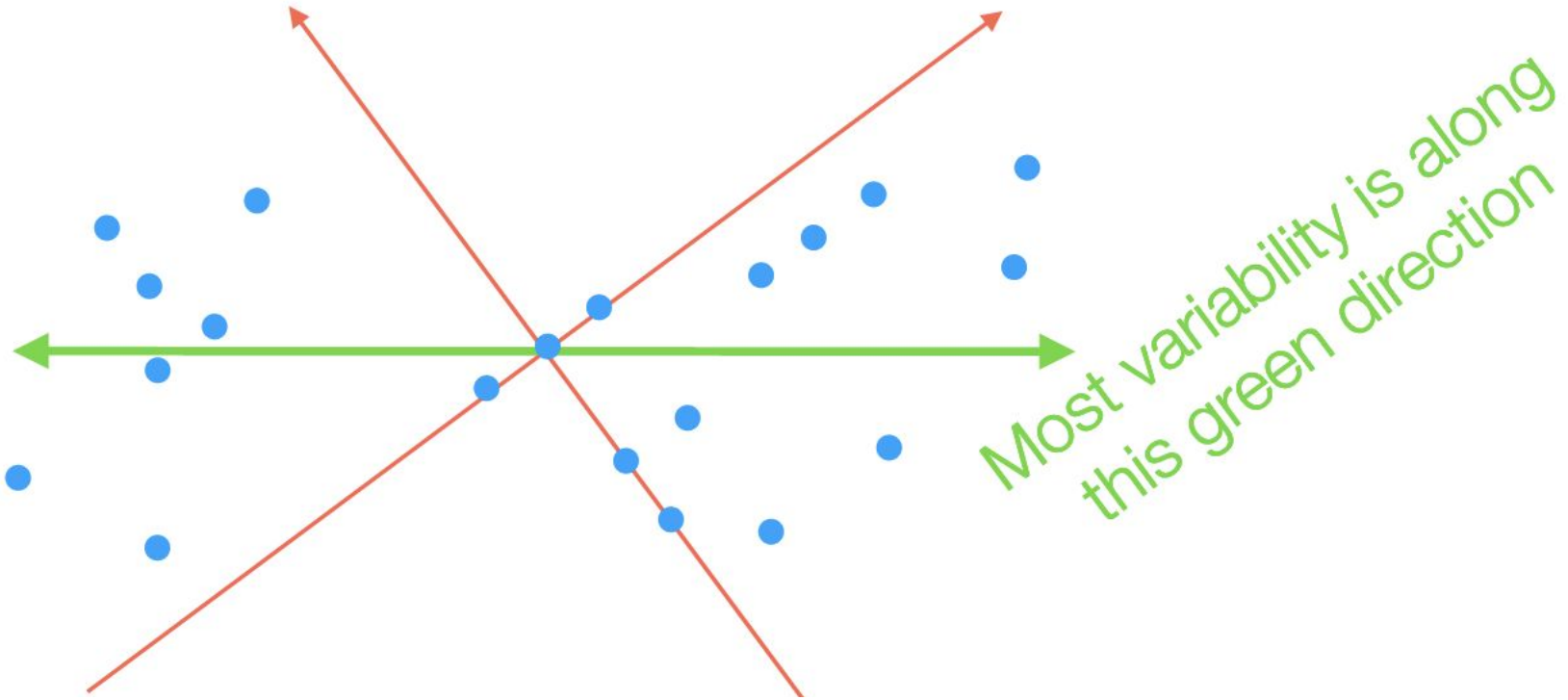
# Principal Component Analysis (PCA)

How to project 2D data down to 1D?


https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

# Principal Component Analysis (PCA)

How to project 2D data down to 1D?



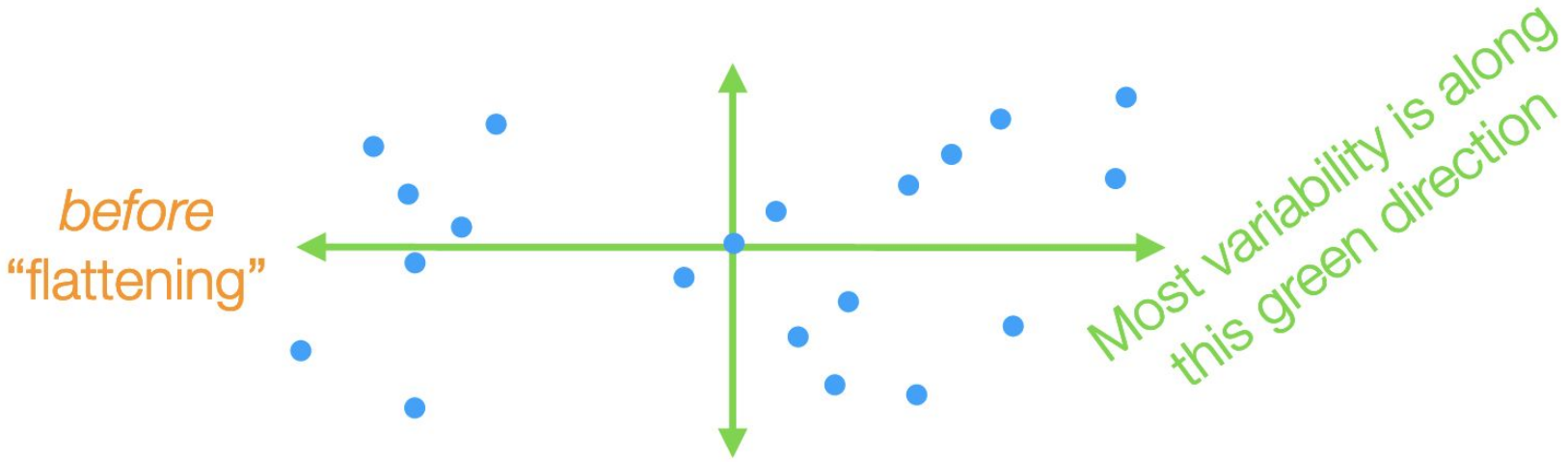

The idea of PCA actually works for 2D ➔ 2D as well
(and just involves rotating, and not "flattening" the data)

https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

# Principal Component Analysis (PCA)

~~How to project 2D data down to 1D?~~

How to rotate 2D data so 1st axis has most variance




The idea of PCA actually works for 2D ➔ 2D as well
(and just involves rotating, and not "flattening" the data)

2nd green axis chosen to be 90° ("orthogonal") from first green axis

https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

# Principal Component Analysis (PCA)

- Finds top *k* orthogonal directions that explain the most variance in the data

https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

# Principal Component Analysis (PCA)

- Finds top *k* orthogonal directions that explain the most variance in the data
  - 1st component: explains most variance along 1 dimension
  - 2nd component: explains most of remaining variance along next dimension that is orthogonal to 1st dimension
  - …

https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

Any questions?

# Principal Component Analysis (PCA)

- Finds top *k* orthogonal directions that explain the most variance in the data
  - 1st component: explains most variance along 1 dimension
  - 2nd component: explains most of remaining variance along next dimension that is orthogonal to 1st dimension
  - …
- "Flatten" data to the top *k* dimensions to get lower dimensional representation (if $k <$ original dimension)

https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

3D example from:

https://setosa.io/ev/principal-component-analysis/

https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

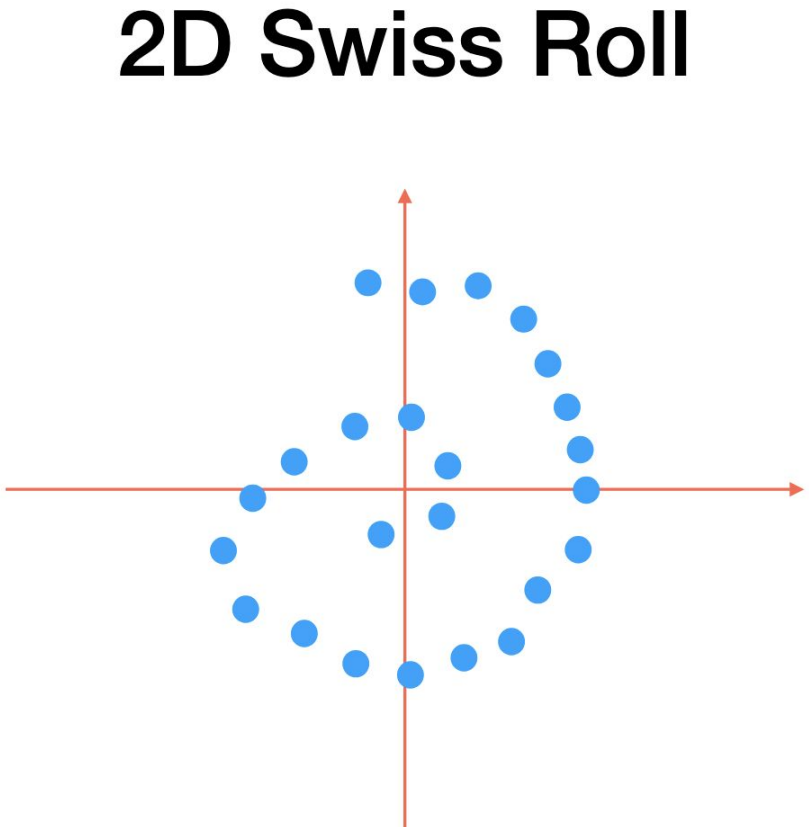
PCA reorients data so axes explain variance in “decreasing order”
→ can “flatten” (*project*) data onto a few axes that captures most variance

https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

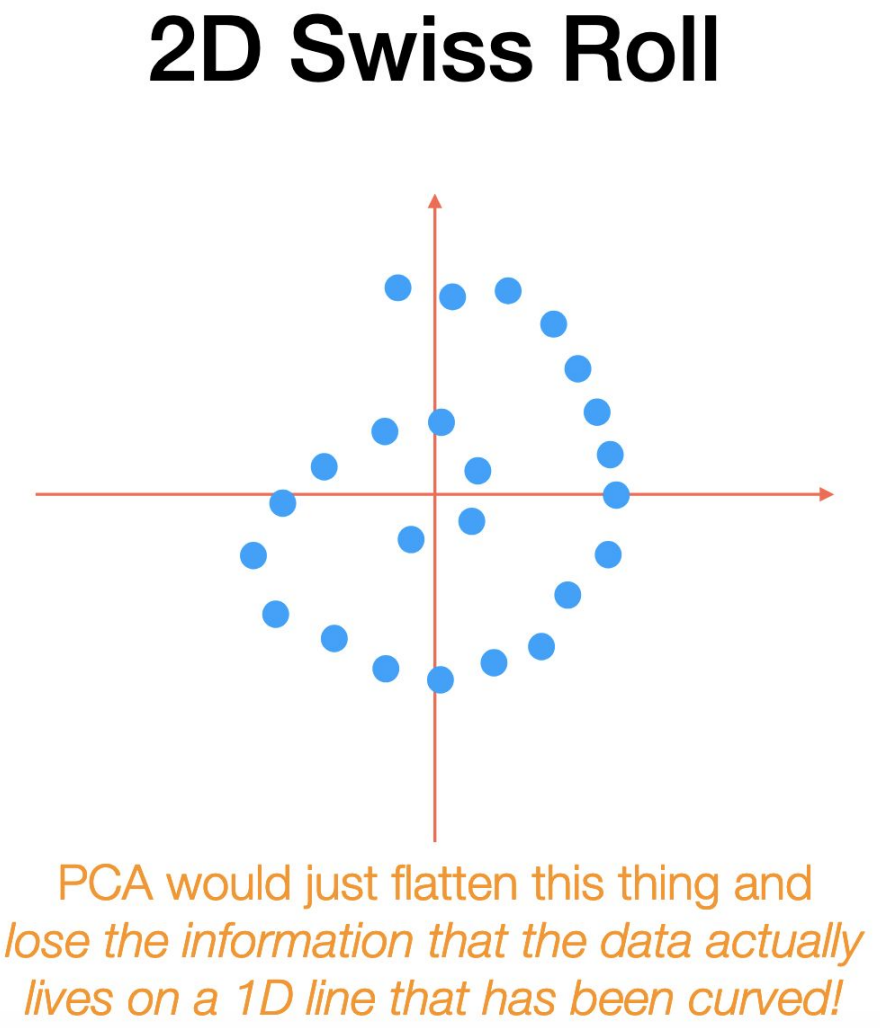
# Limitations of PCA

https://www.foodnetwork.com/recipes/food-network-kitchen/hot-cocoa-cake-roll-4553196

# Limitations of PCA

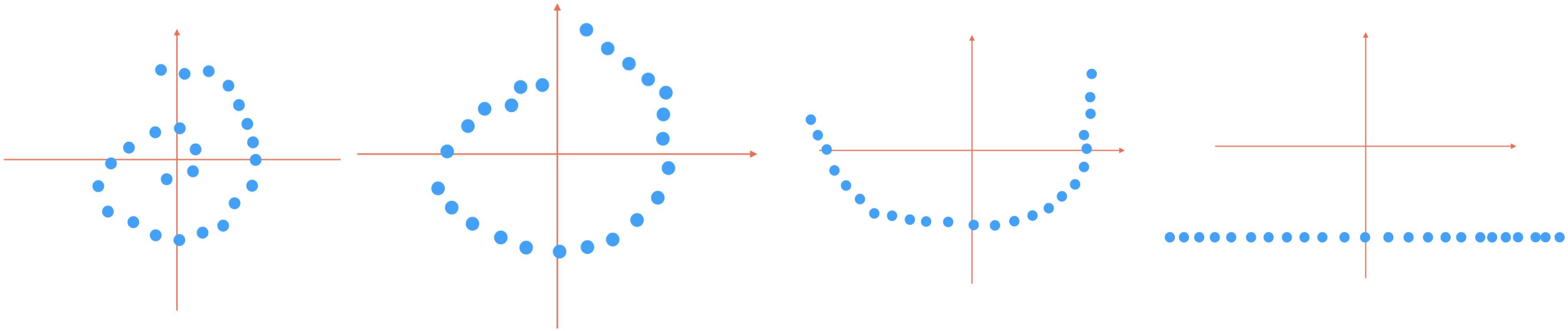


# Limitations of PCA




https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

# Limitations of PCA

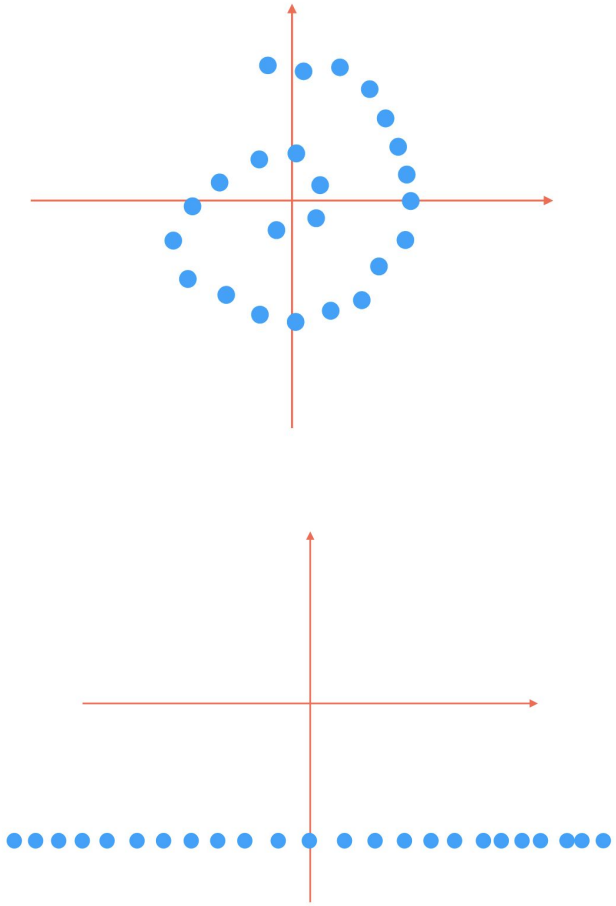


Desired result!

https://www.andrew.cmu.edu/user/georgech/95-865/Lectures/Lecture%20-%2003_essence.pdf

# How to dimension-reduce gnarly datasets?

- The Swiss Roll has an intrinsic dimensionality of 1
  - i.e. “how far along the curve a point is”
- But PCA can’t figure this out because the projection from $\mathbb{R}^2$ to $\mathbb{R}^1$ is ***non-linear***
  - i.e. “unroll” the curve and lay it flat along a number line
- How can we compute non-linear projections?

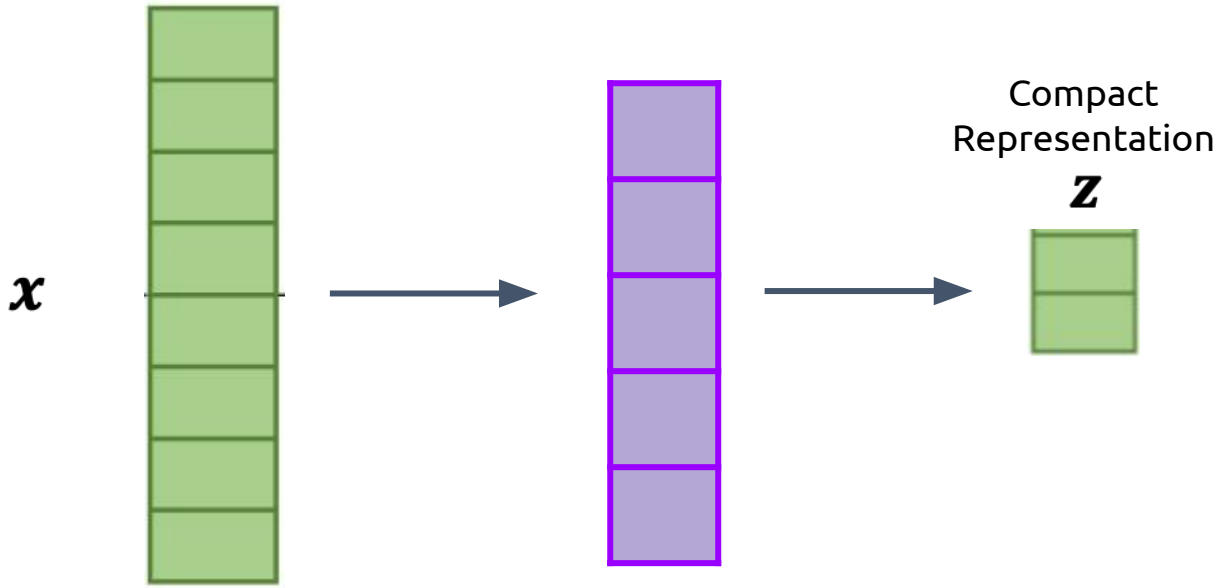


# “I hear these neural nets are pretty good at learning non-linear functions”

Can we use a neural net to learn a non-linear projection to a lower-dimensional space?

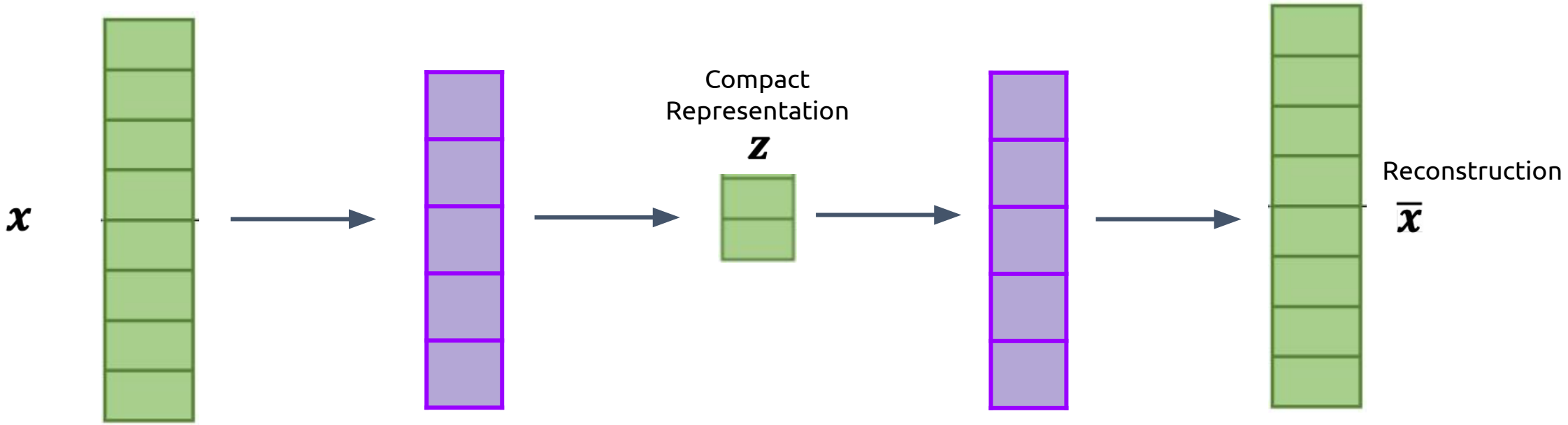
# A nonlinear projection neural net

- We could just use a regular neural net architecture (e.g. fully



# A nonlinear projection neural net

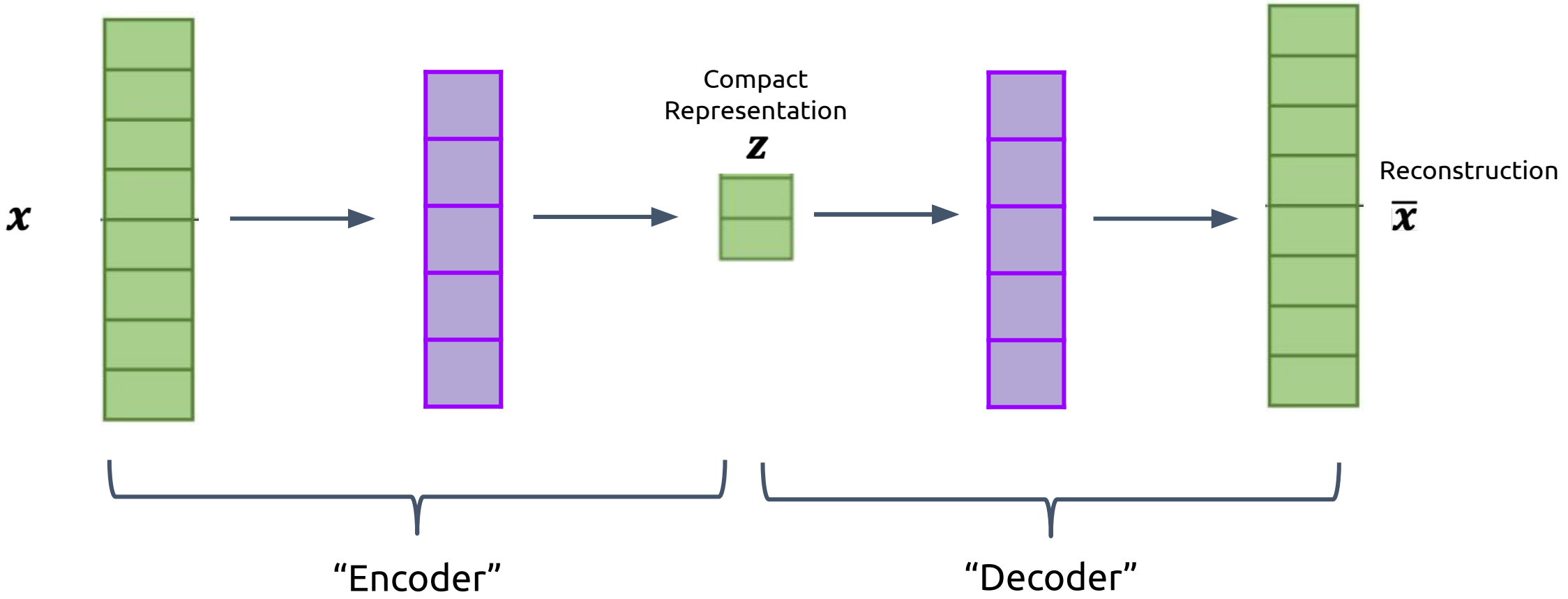
- Idea: a compact representation $z$ is a good non-linear projection of $x$



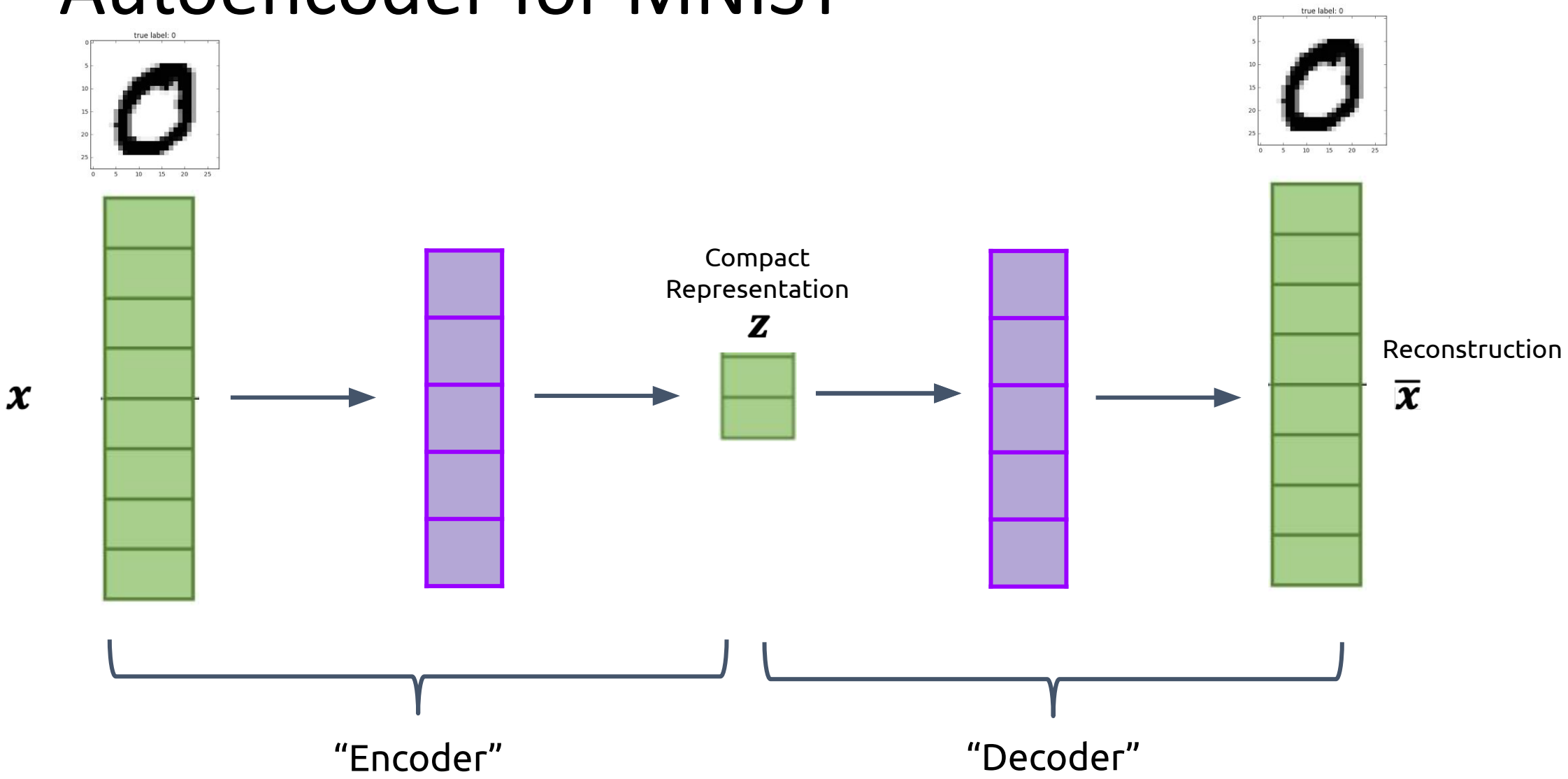
# Autoencoder

- Reconstruction loss: $L(\boldsymbol{x}, \overline{\boldsymbol{x}}) = (\boldsymbol{x} - \overline{\boldsymbol{x}})^2$

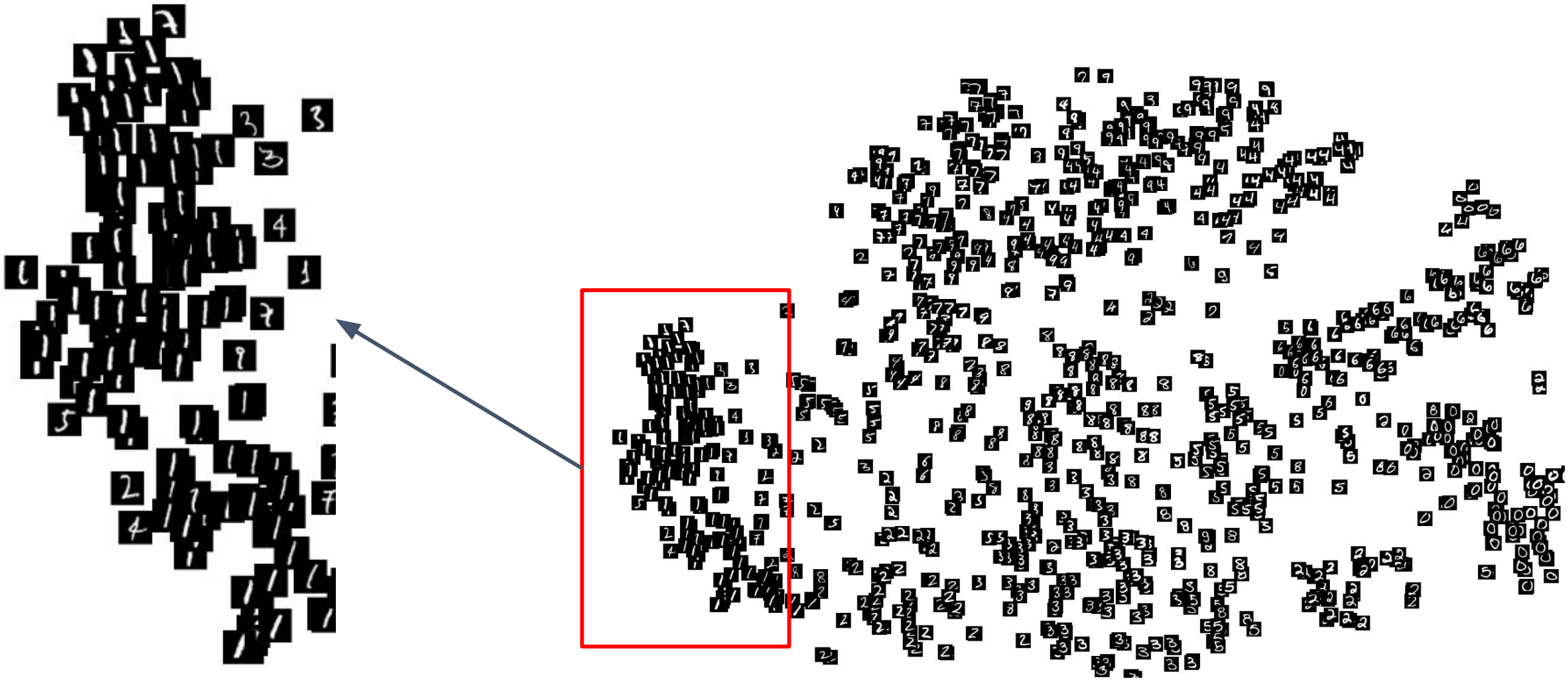
How is this different from the Seq2seq encoder/decoder setup?



# Autoencoder for MNIST



# This visualization? Autoencoder

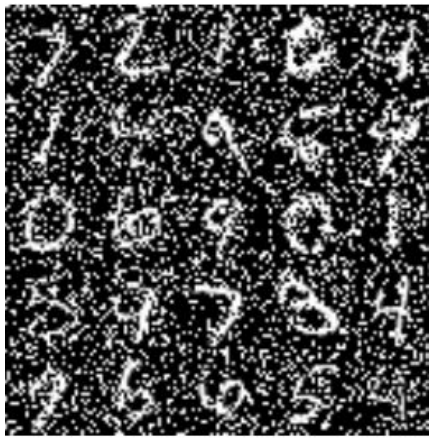


https://hackernoon.com/latent-space-visualization-deep-learning-bits-2-bd09a46920df

# Other Autoencoder applications

## Denoising Autoencoder

Input: Noisy Images



Output: Restored Image

# Recap

Model agnostic interpretation

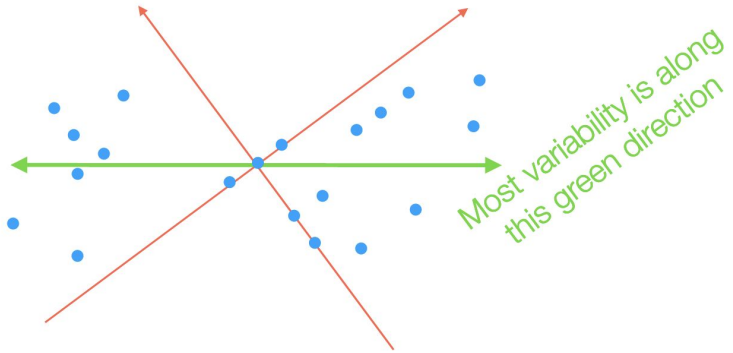
- LIME (or perturbation-based)
- Attention (inbuilt interpretation)

Original Image

Interpretable Components

Unsupervised Learning

- Learning the structure in the data
- Clustering
- Dimensionality reduction using PCA




Auto-encoders (AEs)

- Perform non-linear dimensionality reduction
- Architecture and loss function