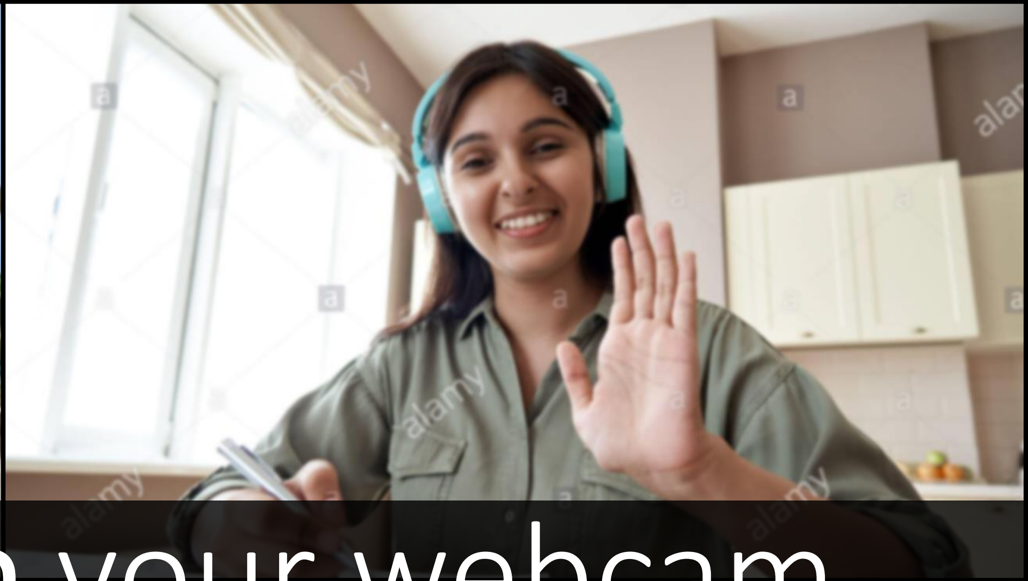
Un-supervised Learning

12

A

CS 3244
Machine Learning

NUS National University of Singapore | Computing



Please **turn on** your webcam

Mystery Student

Recap from W11A

Exercise E11a.1 Solution

How would you interpret?

Linear Regression

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \sum_{r=0}^{n} w_r x_r$$
$$= \boldsymbol{w} \cdot \boldsymbol{x} = \boldsymbol{w}^\top \boldsymbol{x}$$

$\hat{y} \propto w_r x_r, \forall r$

If $w_1 = k w_2$, then x_1 is k times more important than x_2

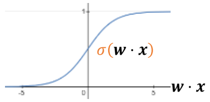
Weighted Sum Interpretation	Gradient Interpretation
Bigger w_r means • **Larger** weight • More **importance** for to x_r • Direction? Supportive (positive) or opposing (negative) **influence**	Bigger w_r means • **Steeper** slope for x_r axis • Changes in x_r lead to bigger in $\hat{y}$ changes • More **importance** for x_r • Direction indicates increasing or decreasing **influence**

Exercise E11a.2 Solution

How would you interpret?

Logistic Regression

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}$$
$$f = \boldsymbol{w} \cdot \boldsymbol{x} = \sum_{r=0}^{n} w_r x_r$$



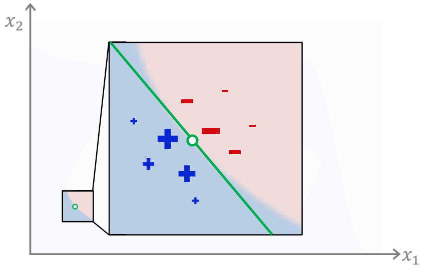
$\mathrm{logit}(P(\hat{y})) \propto w_r x_r, \forall r$

If $w_1 = k w_2$, then log odds ratio of x_1 is k times bigger than of x_2

Weighted Sum Interpretation	Gradient Interpretation
Bigger w_r means • **Larger** importance • Direction indicates **influence**	Bigger w_r means? • **Steeper** slope for x_r near decision boundary • **Decision boundary** more *perpendicular* to x_r • Weight sign indicates direction of **pos**/**neg** prediction

LIME

Local Interpretable Model-agnostic Explanations



Prediction Model

$f(\boldsymbol{x})$

- Non-linear model of $\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix}$
- Shown as curvy decision boundary

Explanation: **LIME**

1. Starting with **instance** $\boldsymbol{x}$ to explain
2. Focus on **Local** region
3. Training set as **neighbors** $\boldsymbol{x}^{\langle\eta\rangle} \in X^{\langle\eta\rangle}$
4. Train **surrogate model**, e.g., linear:

$$g(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} = \sum_{r=0}^{n} w_r x_r$$

Image Credit: https://santiagof.medium.com/model-interpretability-making-your-model-confess-lime-89db7f70a72b

LIME

Find "best" explainer g that minimizes $\xi(\boldsymbol{x})$

Python API: https://github.com/marcotcr/lime

$$\xi(\boldsymbol{x}) = \underset{g \in G}{\mathrm{argmin}}\left(\overbrace{L(f, g, \pi_x)}^{\text{"Faithful"}} + \overbrace{\Omega(g)}^{\text{"Simple"}}\right)$$

f is the **predictor** function (model)
g is the **explainer** function (model)
$\pi_x(\boldsymbol{x}^{\langle\eta\rangle})$ is the neighbor **proximity** function

- E.g., exponential decay $\exp\left(-\left(d(\boldsymbol{x}, \boldsymbol{x}^{\langle\eta\rangle})\right)^2\right)$

$L(f, g, \pi_x)$ is the ***locally-weighted* error loss** function between predictor f and explainer g

$$L(f, g, \pi_x) = \sum_{\boldsymbol{x}^{\langle\eta\rangle} \in X^{\langle\eta\rangle}} \pi_x(\boldsymbol{x}^{\langle\eta\rangle}) \left(f(\boldsymbol{x}^{\langle\eta\rangle}) - g(\boldsymbol{x}^{\langle\eta\rangle})\right)^2$$

$\Omega(g)$ is the **sparsity regularization**

- Want simpler explanation
  - ⇒ *fewer weights*
  - ⇒ Lasso (L1 norm)
- Penalizes if total weights is too large
- λ is hyperparameter on how much to penalize

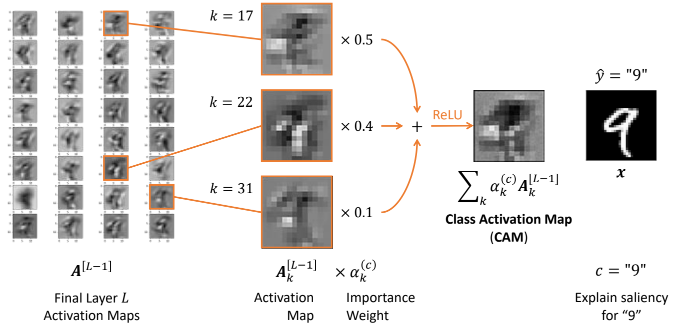
$$\Omega(g) = \lambda \|w_r\|_1 = \lambda \sum_{r=1}^{n} w_r$$

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). "Why should i trust you?" Explaining the predictions of any classifier. *KDD'16*.

Grad-CAM Steps

1. Compute Activation Maps $\boldsymbol{A}^{[L]}$ of last conv layer L
   1. via Forward Propagation
2. Choose class label c to explain about (e.g., predict "9", "car")
3. Filter prediction $\hat{y}$ to be about class c
   1. Given: $\hat{\boldsymbol{y}} = \begin{pmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \hat{y}^{(c)} \\ \hat{y}^{(n)} \end{pmatrix}$, $\boldsymbol{e}^{(c)} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$, then $\hat{\boldsymbol{y}}^{(c)} = \hat{\boldsymbol{y}} \circ \boldsymbol{e}^{(c)} = \begin{pmatrix} 0 \\ 0 \\ y^c \\ 0 \end{pmatrix}$
   2. To generate explanation only for that class c
4. Compute importance weight $\alpha_k^{(c)}$ for each Activation Map $\boldsymbol{A}_k^{[L]}$
   1. Backprop from $\hat{\boldsymbol{y}}^{(c)}$ to get gradients (relative to activations) at last conv layer
5. Compute weighted sum with ReLU to get **Class Activation Map**

Grad-CAM example: Why did the CNN predict "9"?



Final Exam

NUS National University of Singapore
Department of Computer Science
School of Computing

2-hour Final Assessment

Open Book Policy: same as in the midterm assessment:

- Open lecture notes, Colab notebooks and FAQs from our class.
- Any printed physical notes are also admissible
- No other lecture notes, or internet sites admissible.
- No online calculation (Colab, Wolfram Alpha) ; only allowed physical calculators.

There will be programming and calculation questions.

- Where needed, we provide suitable function call and library prototypes.
- Pseudocode ok!

Final Exam Topic Coverage

All W01–W13 topics covered in Lecture Slides, Tutorials or Colab

- 80% on Weeks 07–13
- 20% on Weeks 01–06

Most of the previous exam questions in the exam archive cover different topics in ML, so if in doubt, ask us on [#assessments].

Week 07 27 Sep	Midterm and Evaluation Metrics
Week 08 4 Oct	Data Processing and Feature Engineering *T05: Evaluation Metrics*
Week 09 11 Oct	Perceptron and Neural Networks *T06: Data Processng and Feature Engineering*
Week 10 18 Oct	Intro to Deep Learning *T07: Perceptron and Neural Networks*
Week 11 25 Oct	Deep Learning and Explainable AI *T08: Deep Learning*
Week 12 1 Nov	Unsupervised ML *T09: Deep Learning and Explainable AI*
Week 13 8 Nov	ML Ethics and Revision *T10: Unsupervised ML*


06
UNSUPERVISED
LEARNING

Learning Outcomes

1. k-Means clustering
   1. Describe its intuition and objective
   2. Understand how it is trained
2. Perform and **interpret** clustering on data
3. Auto-Encoders
   1. Understand how they compute and are trained
   2. Describe their different types
   3. Describe their applications

Lecture Outline

1. Unsupervised Learning introduction
2. K-Means Clustering
3. Clustering Interpretation
4. Auto-Encoders

k-Means Clustering

Clustering Applications

Customer Segmentation

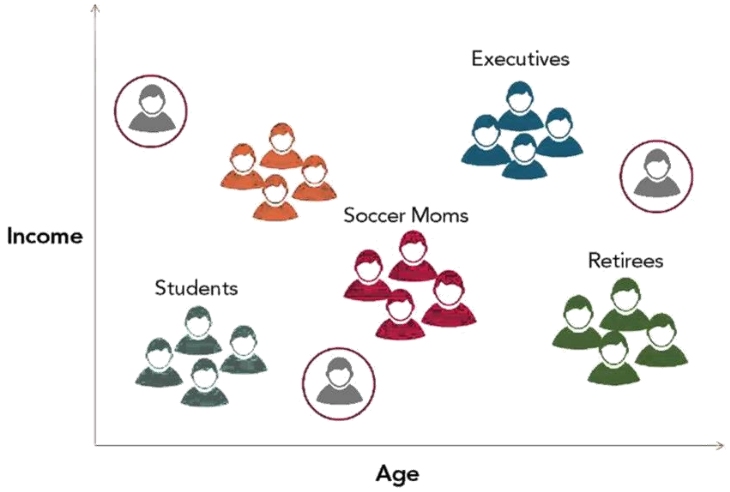
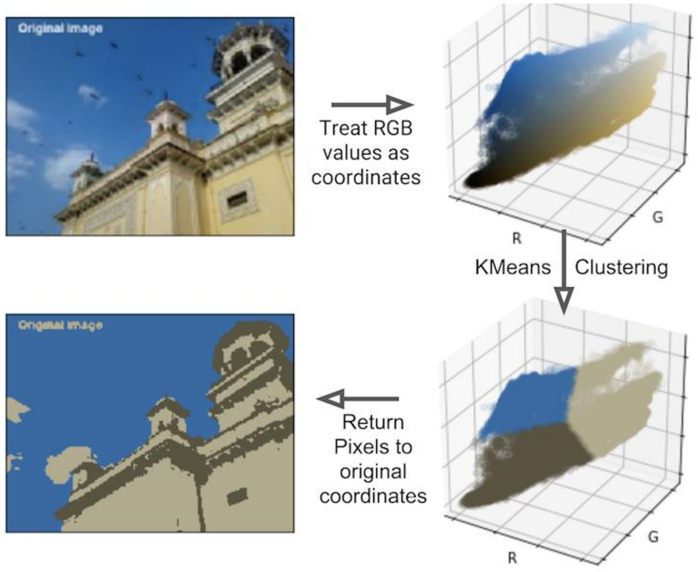


Image Segmentation



Behavior Segmentation

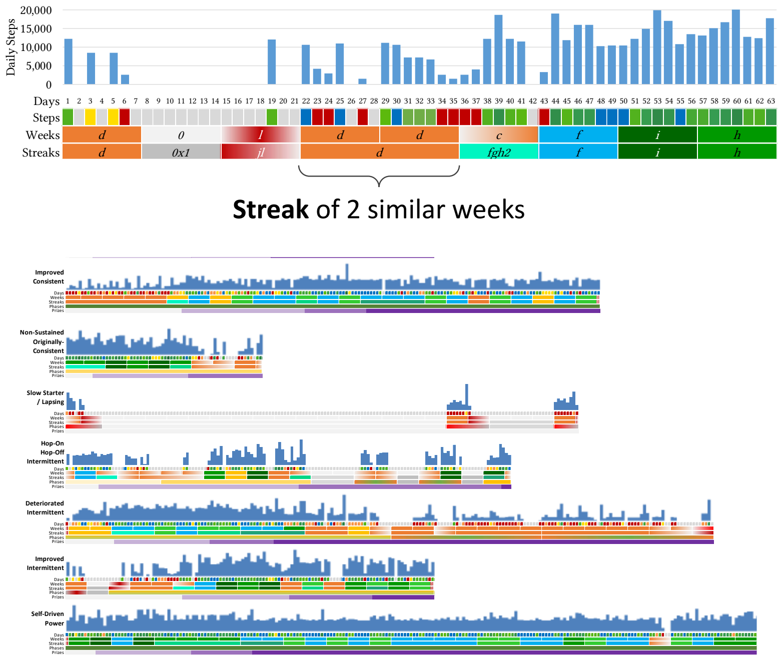


Image credit:

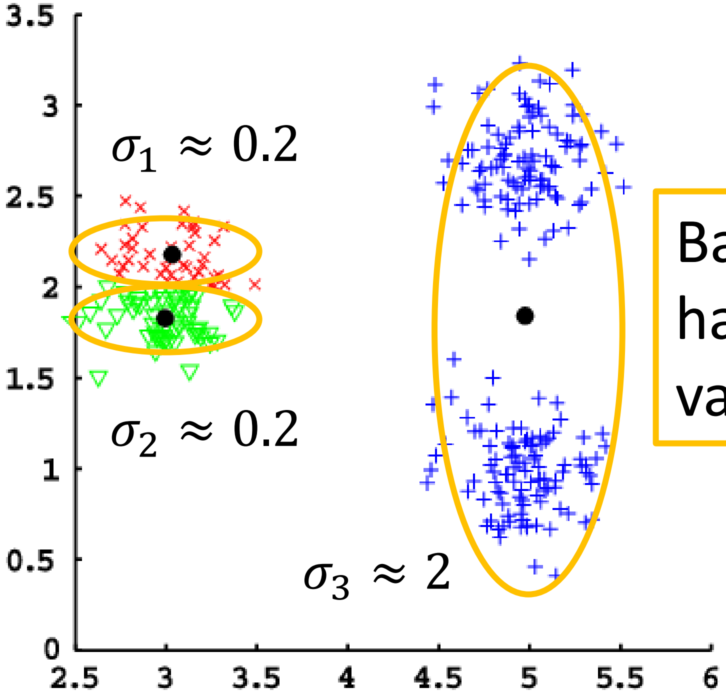
- https://youtu.be/zPJtDohab-g
- https://medium.com/@michael.francis.gray/a-visual-demo-of-kmeans-66f7132427ad
- Lim, B. Y., Kay, J., and Liu, W. 2019. **How does a nation walk? Interpreting large-scale step count activity with weekly streak patterns.** *IMWUT*.

k-Means Intuition

Better clustering has **smaller Total Within-Cluster Variance**

Difference?

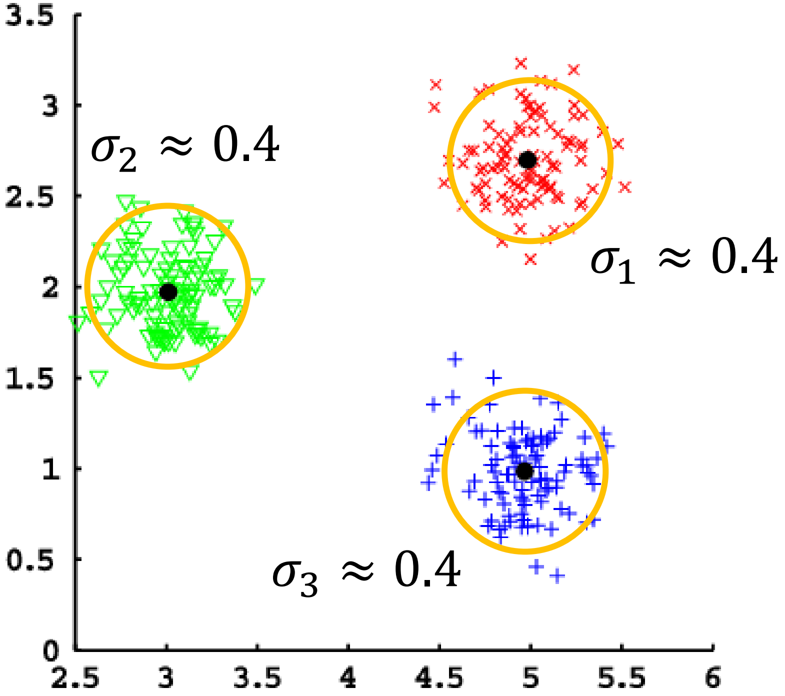
Bad Clusters



Bad cluster has very large variance

$$\sigma_1^2 + \sigma_2^2 + \sigma_3^2 = 0.2^2 + 0.2^2 + 2^2 \approx 4$$

Good Clusters



$$\sigma_1^2 + \sigma_2^2 + \sigma_3^2 = 0.4^2 + 0.4^2 + 0.4^2 \approx 0.5$$

Image Credit: https://www.semanticscholar.org/paper/The-MinMax-k-Means-clustering-algorithm-Tzortzis-Likas/e005940c7c758ea6ba830d1db4cf0f74c3c10514/figure/0

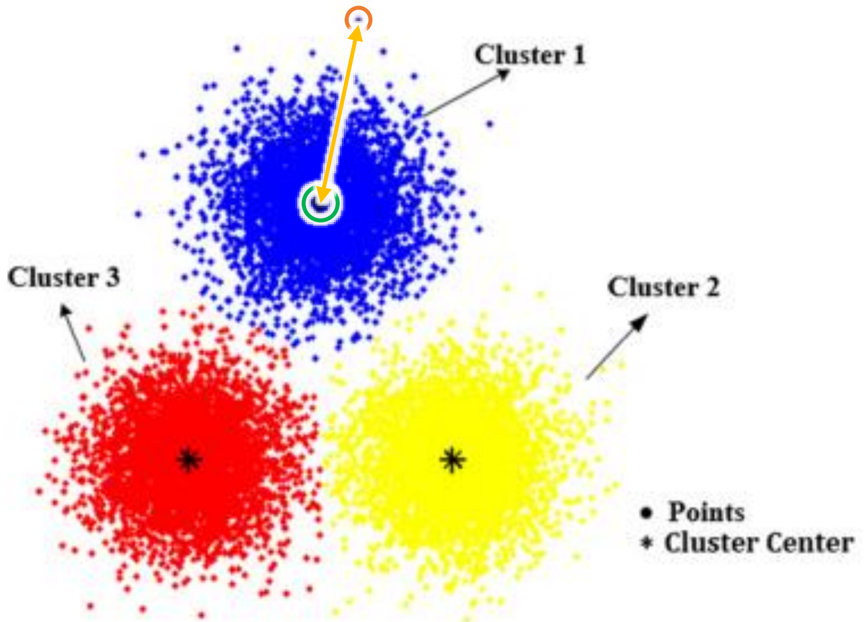
k-Means Objective

Minimize **Within-Cluster Sum-of-Squares (WCSS)** (i.e. variance)

$$L = \arg\min_{S} \sum_{c=1}^{k} \sum_{x \in S_c} \|\boldsymbol{x} - \boldsymbol{\mu}_c\|^2$$

Image credit

- $S = \{S_1, S_2, \dots, S_c, \dots, S_k\}$ is the set of all clusters
    - k is the total number of clusters
- S_c is the cth cluster of points
    - Note that c refers to cluster, not class
    - $\boldsymbol{x} \in S_c$ refers to a point in cluster S_c
    - $\boldsymbol{\mu}_c = \frac{1}{|S_c|} \sum_{x \in S_c} \boldsymbol{x}$ is the centroid point in cluster S_c
    - $\|\boldsymbol{x} - \boldsymbol{\mu}_c\|^2$ refers to the squared Euclidean distance from $\boldsymbol{x}$ to $\boldsymbol{\mu}_c$



k-Means clustering algorithm

for $c = 1$ to k:
    $\boldsymbol{\mu}_c \leftarrow \text{Random}()$

1) Initialize cluster centroids

while not $\text{Converged}()$:

Iterative refinement

for $j = 1$ to m:
    $y^{(j)} \leftarrow c = \underset{c}{\text{argmin}} \left\| \boldsymbol{x}^{(j)} - \boldsymbol{\mu}_c \right\|^2$
    $\boldsymbol{x} \leftarrow S_c$

2) Assign datapoints to clusters

for $c = 1$ to k:
    $\boldsymbol{\mu}_c \leftarrow \frac{1}{|S_c|} \sum_{\boldsymbol{x} \in S_c} \boldsymbol{x}$

3) Update cluster centroids

$t \mathrel{+}= 1$

return $\boldsymbol{y}$

Cluster labels of all datapoints

k-Means clustering algorithm

for $c = 1$ to k:

  $\boldsymbol{\mu}_c \leftarrow \text{Random}()$

while not $\boxed{\text{Converged}()}$:

  for $j = 1$ to m:

    $\boxed{y^{(j)} \leftarrow c = \underset{c}{\operatorname{argmin}} \left\| \boldsymbol{x}^{(j)} - \boldsymbol{\mu}_c \right\|^2}$

    $\boldsymbol{x} \leftarrow S_c$

  for $c = 1$ to k:

    $\boxed{\boldsymbol{\mu}_c \leftarrow \frac{1}{|S_c|} \sum_{x \in S_c} \boldsymbol{x}}$

  $t \mathrel{+}= 1$

return $\boldsymbol{y}$

$\text{Converged}() = \left(\left(\sum_c \left\| \boldsymbol{\mu}_c^{(t)} - \boldsymbol{\mu}_c^{(t-1)} \right\| < \tau_\mu \right) \textbf{ or } (t > \tau_t) \right)$

$d_{min}^{(j)} = \text{Big Number}$

for $c = 1$ to k:

  $d = \left\| \boldsymbol{x}^{(j)} - \boldsymbol{\mu}_c \right\|^2$

  if $d_{min}^{(j)} > d$:

    $d_{min}^{(j)} = d$

    $y^{(j)} \leftarrow c$

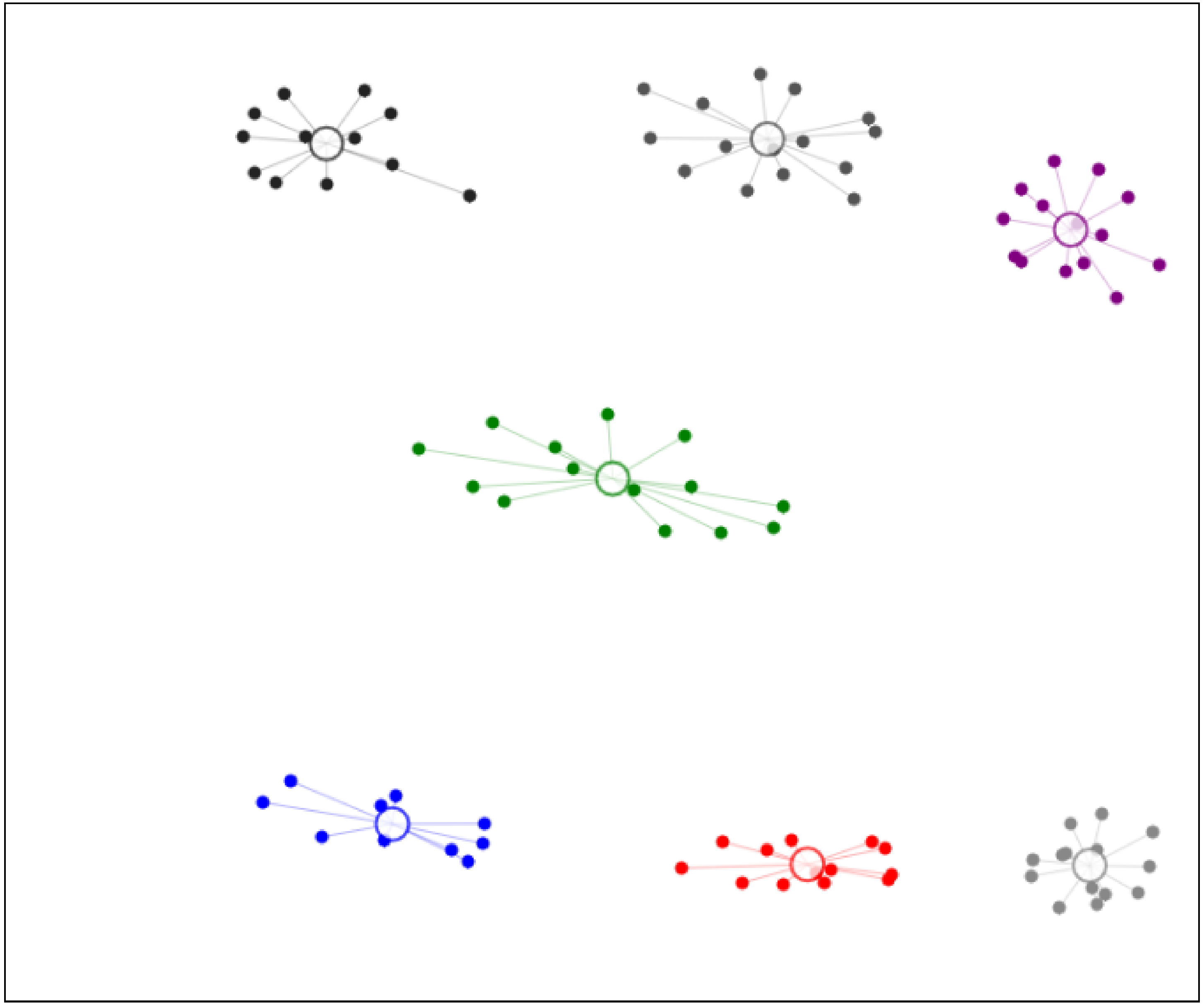
$\boldsymbol{\Sigma} = 0$

for $j_c = 1$ to m_c:

  $\boldsymbol{\Sigma} \mathrel{+}= \boldsymbol{x}^{(j_c)}$

$\boldsymbol{\mu}_c \leftarrow \boldsymbol{\Sigma}/m_c$

This app is ultimately interactive. You can add more points or select template points from the right panel. More hints are available at the bottom.



Draw a point distribution:

1 2 3 4 5 6

Iterations: 5

7 clusters

General Statistics:

Delta position: 0.00

Source: https://user.ceng.metu.edu.tr/~akifakkus/courses/ceng574/k-means/

How to choose k (number of clusters)?

- Use domain knowledge
- Note the k with *diminishing* return
  - When k is too high, marginal decrease in **within-cluster sum-of-squares** (WCSS)
  - How to see?
    - "Elbow" method

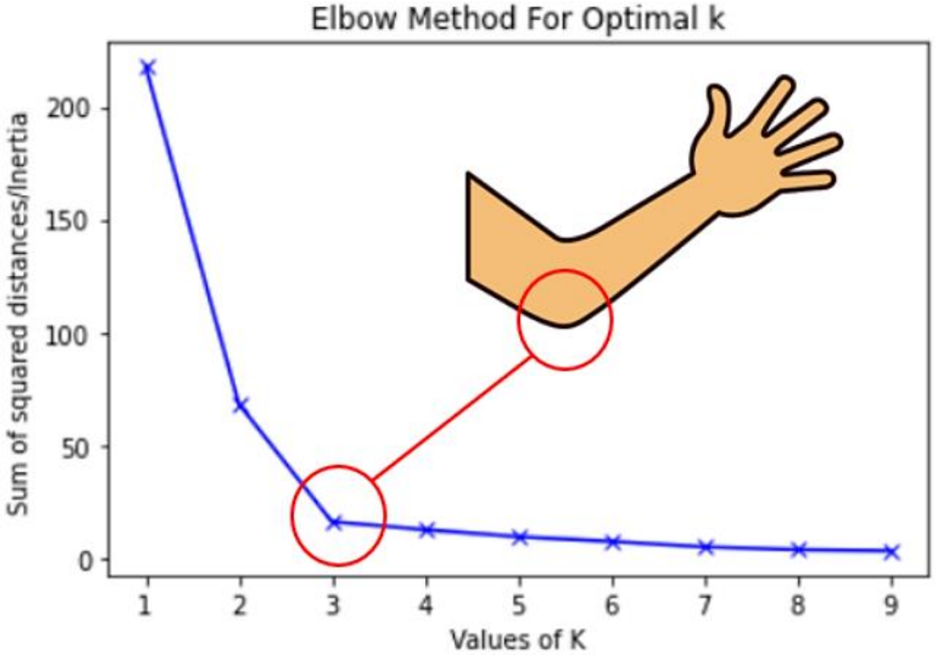


Image credit: https://www.analyticsvidhya.com/blog/2021/05/k-mean-getting-the-optimal-number-of-clusters/

k-Means is not kNN

- What's the difference?

	k-means Clustering	k-Nearest Neighbors (kNN)
Learning Paradigm	Unsupervised	Supervised
Purpose	Group neighbors	Label based on neighbors
k is …	Number of clusters	Number of neighbors
Distance metric	Only squared Euclidean (to match Variance)	Any distance metric (e.g., Euclidean, Manhattan, Cosine)
Measures distance between	Training set $\boldsymbol{x}$ points and cluster centroids	Test set $\boldsymbol{x}$ points and training set neighbors
Need model training?	Yes	No

k-Means clustering for images
Color Quantization to reduce image size

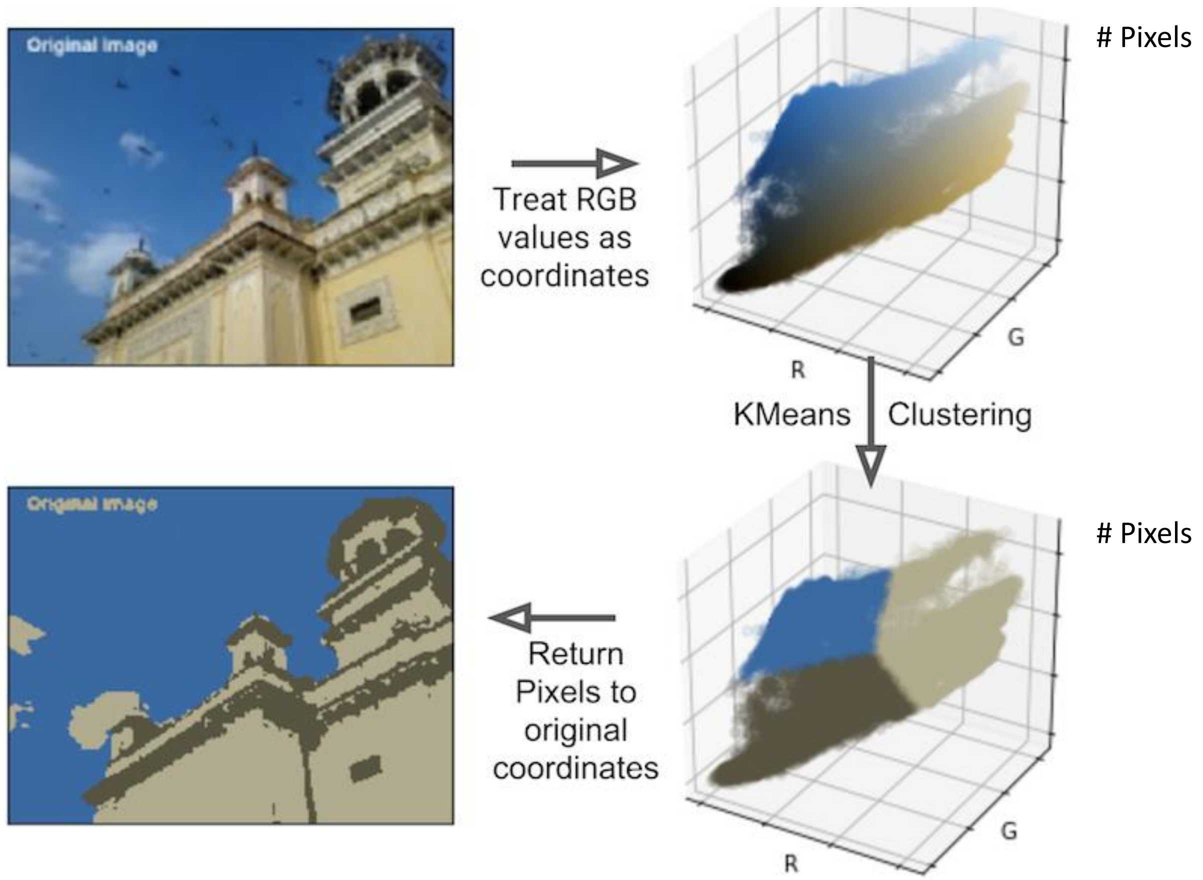
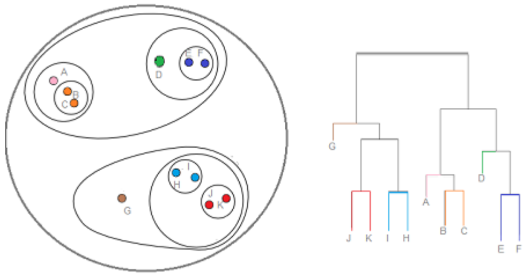


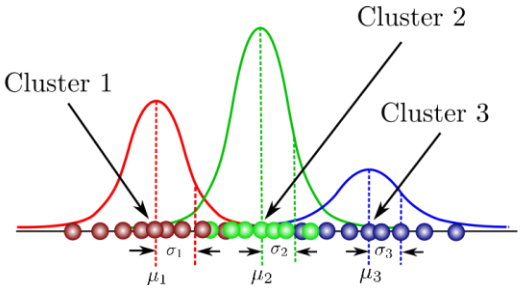
Image credit: https://medium.com/@michael.francis.gray/a-visual-demo-of-kmeans-66f7132427ad

Other Clustering Methods

- k-Means Clustering (in exam)
  - Need to specify k before computing
  - Features need to be numeric
  - Only handles Euclidean Distance
- K-Medoids Clustering
  - Can use any dissimilarity (distance) metric
- Hierarchical Clustering [sklearn]
  - Clusters bases on hierarchy of clusters
  - Produces dendrogram
- Gaussian Mixture Model (GMM) [sklearn]
  - Estimates assumed normally distributed clusters of points
- Density-Based Clustering (DBSCAN) [sklearn]



Hierarchical Clustering



Gaussian Mixture Model

DBSCAN

Image credits: https://www.statisticshowto.com/hierarchical-clustering/, https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95, https://towardsdatascience.com/understanding-dbscan-algorithm-and-implementation-from-scratch-c256289479c5

Evaluating Clustering

- Internal (measures of within- and between-cluster distances)
  - Davies–Bouldin index
  - Dunn Index
  - Silhouette Coefficient
- External (compare with benchmarks or actual labels)
  - Cluster Purity
  - F1 Score (like classification, but need to label clusters first)
- Cluster Tendency
  - Hopkins statistic
    - Good to check whether your data has natural clusters
    - Otherwise, cannot trust clustering

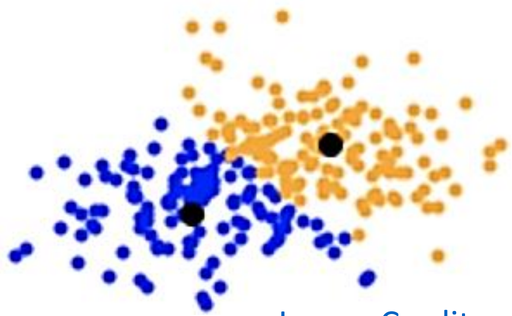


Image Credit

Further Reading: https://en.wikipedia.org/wiki/Cluster_analysis#Evaluation_and_assessment

Questions!

Clustering Real-World Data

W12 Pre-Lecture Task (due before next Mon)

Read

1. Clustering With More Than Two Features? Try This To Explain Your Findings by Mauricio Letelier

Task

1. Describe other use cases where you need to **apply domain knowledge** with data-driven **unsupervised learning** to better understand your business or engineering problem

   Tip: you can your own projects too; you don't have to be correct

2. Post a 1–2 sentence answer to the topic in your tutorial group: #tg-xx

Examples with specific domain concepts (not generic)

In biostatistics, when learning about disease incidence or understanding treatment, there are many variables that can be collected regarding each patient. In such cases, domain knowledge of the disease or treatment would be useful in extracting important features for models. Eg: cancer markers for specific cancers

When classifying molecules based on effects, grouping them based on size, weight, elements used, and individual element count may not be enough. By identifying functional groups within the molecules using domain knowledge, the agent can better approximate the effects of the molecules.

Our project is about Reddit comments which makes an interesting case for where domain knowledge is needed if it were an unsupervised learning problem. There are many different kinds of comments on Reddit, from users simply commenting "cat" on cat pictures, making one-liner jokes, starting controversial arguments, or sharing sections of news articles. It would be interesting to cluster these comments into different types based on the text features. We could also cluster users based on the kinds of comments they make. (edited)

Another use case where I need to apply domain knowledge with data-driven unsupervised learning is analysing user stickiness in video platforms. Data of time of logging in, average length of watched video and types of most often watched videos can be collected and used by the platforms to decide what types/length of videos they want to focus on based on their different user behaviour.

When using unsupervised learning to detect suspicious activities in the shipping industry, it is important to have some domain knowledge on what type of ships and what characteristics of these ships are more likely to exhibit suspicious activity. Domain knowledge could include the countries of origin of the ships, their last port of call, their cargo, etc.

Important Take-Away

Use domain knowledge to

- Do **feature engineering**, then cluster
- **Inspect clusters** to check if they "make sense"
  - Consistent with domain knowledge
  - Or are spurious


National Steps™ Challenge
MOVE IT
WITH NATIONAL STEPS CHALLENGE™ SEASON 4
Health Promotion Board
Clock
10,000 steps
and earn
40 Healthpoints
daily!

Not in Exam

Not in Exam

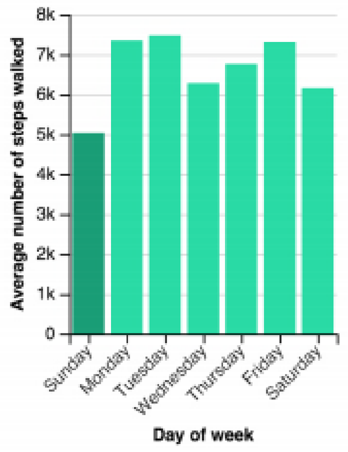
Understanding Data: Single-User to Hundreds of Thousands of Users

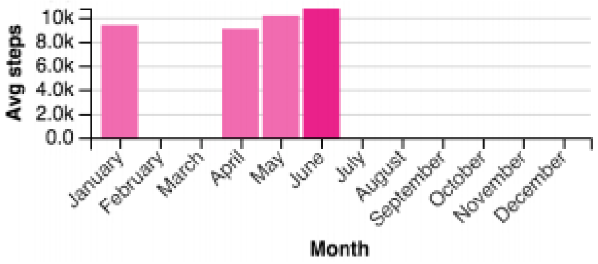
Personal Informatics

Population Analytics

Not in Exam

Understanding Data: Single-User to Hundreds of Thousands of Users





Big Data

140 Thousand users

305 days (10 months)

9 Million total days

74 Billion steps

Can we identify **common behaviors** and **segment** users?

Not in Exam

Research Approach

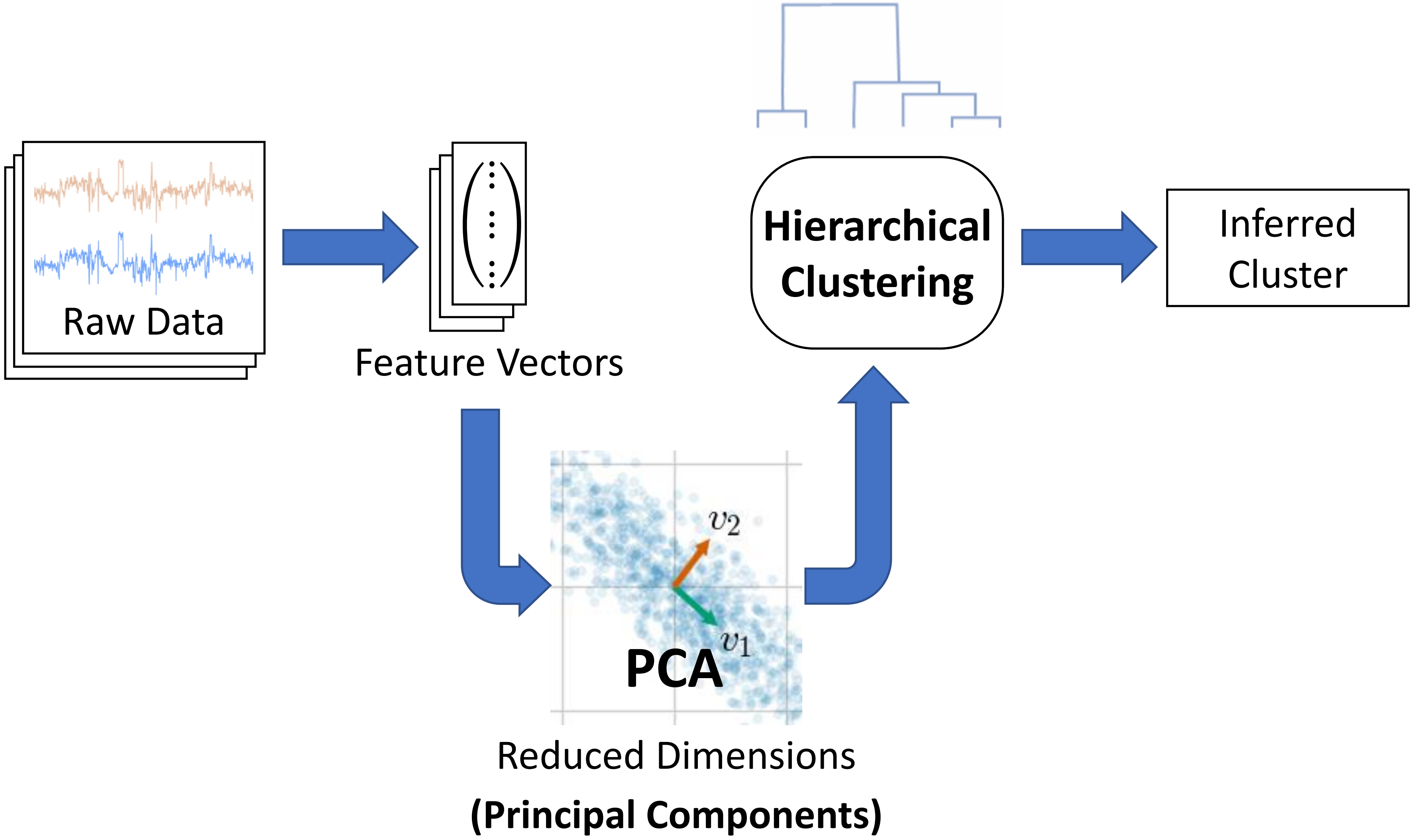
- Use **feature and data clustering**
- To identify **common temporal patterns** in step count
- Describe steps behaviors with patterns as **semantic units**

$$Mean(Steps_{Week1}) = 2k \; \& \; Mean(Steps_{Week5}) = 8k$$

“Slow starter”

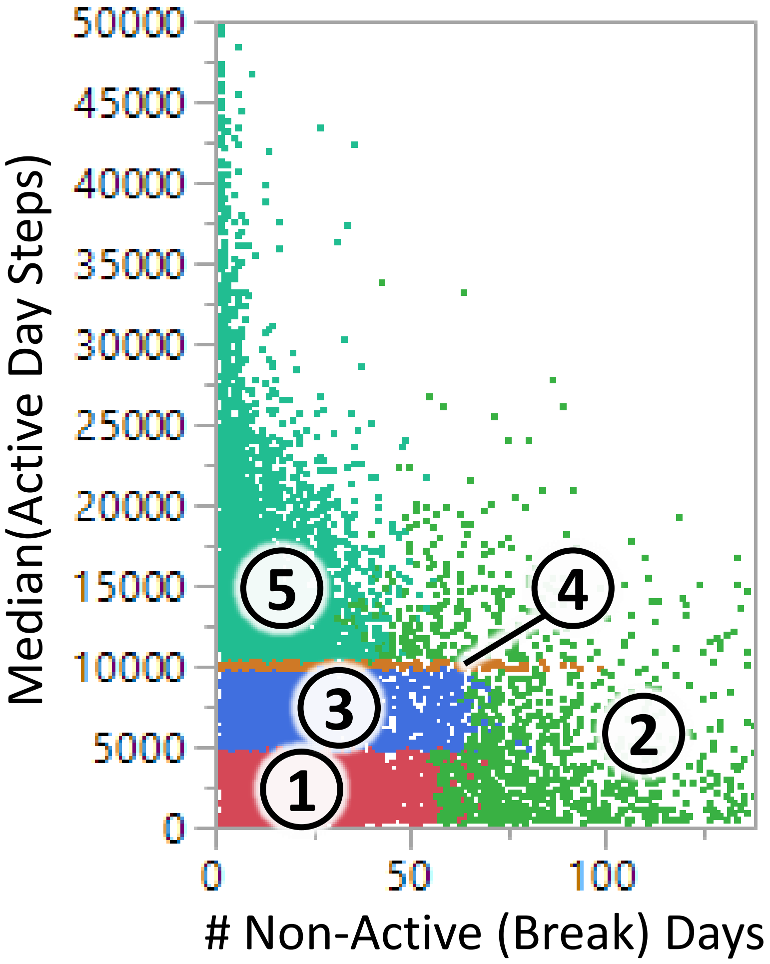
Lim, B. Y., Kay, J., and Liu, W. 2019. **How does a nation walk? Interpreting large-scale step count activity with weekly streak patterns.** In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT).*

Unsupervised Machine Learning Pipeline



Describe and Label Clustered Streaks

Streak = Consecutive Active Days → Non-Active (Break) Days



In Slack #general

1. **Write** to thread to **describe/label** each cluster
2. **Emote** (👍 :+1:) to vote

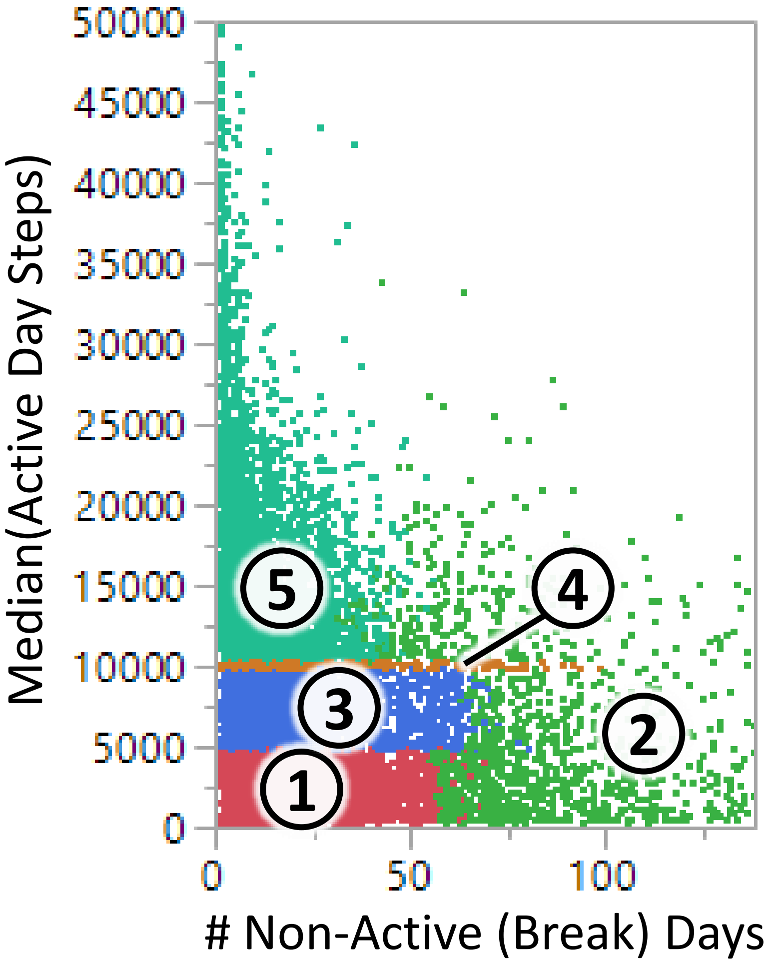
Total campaign duration: 300 days

Active Day is a day with >500 steps

Break Day is a day with ≤500 steps

Describe and Label Clustered Streaks

Streak = Consecutive Active Days → Non-Active (Break) Days



In Slack #general
1. **Write** to thread to **describe/label** each cluster
2. **Emote** (👍 :+1:) to vote

1. Low Steps Effort → Short Break → Quit
2. Low/Moderate Effort → Long Break → Resume
3. Moderate Effort → Short Break → Resume
4. Incentive "just enough" Effort
5. Enthusiastic Effort → Short Break → Resume

Total campaign duration: 300 days
Active Day is a day with >500 steps
Break Day is a day with ≤500 steps

Not in Exam

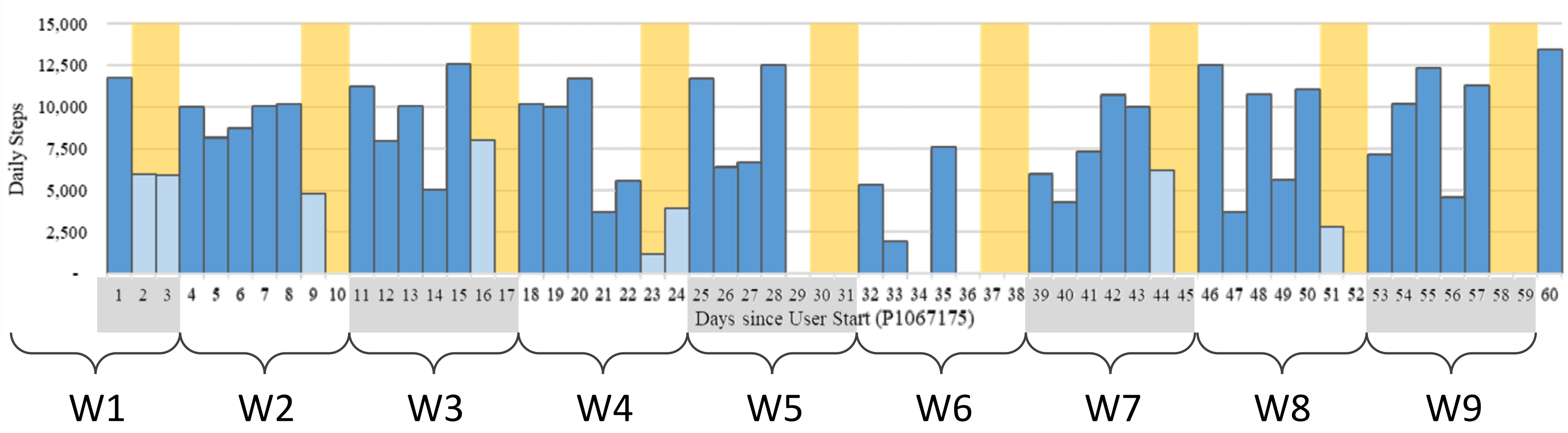
Challenges in Analyzing

1. Handling Imbalanced Data
2. Handling Cyclic Patterns
3. Detecting Routine Habits and Changes
4. Describing Wearing Behavior and Breaks
5. Handling Traits (Accounting for the Influence of Demographics)

Lim, B. Y., Kay, J., and Liu, W. 2019. **How does a nation walk? Interpreting large-scale step count activity with weekly streak patterns.** In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)*.

Not in Exam

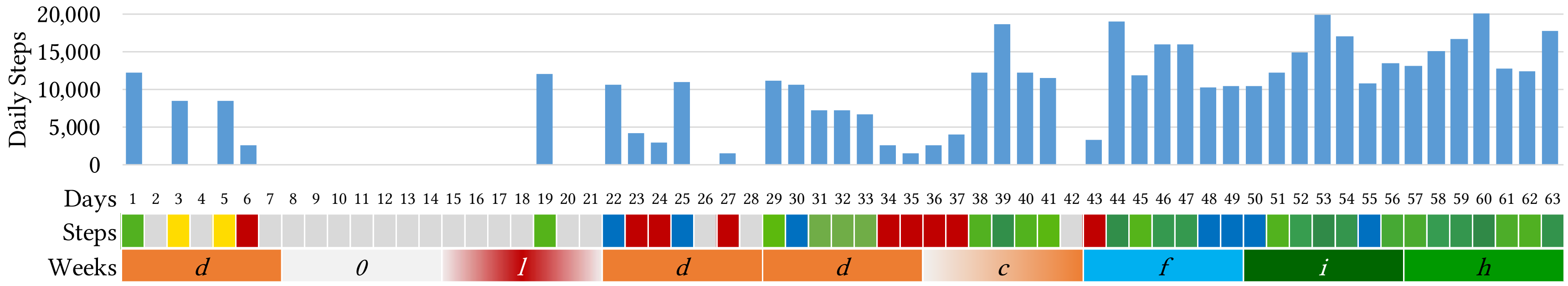
Challenge 2: Handling Cyclic Patterns



Not in Exam

Challenge 3: Detecting Routine Habits and Changes

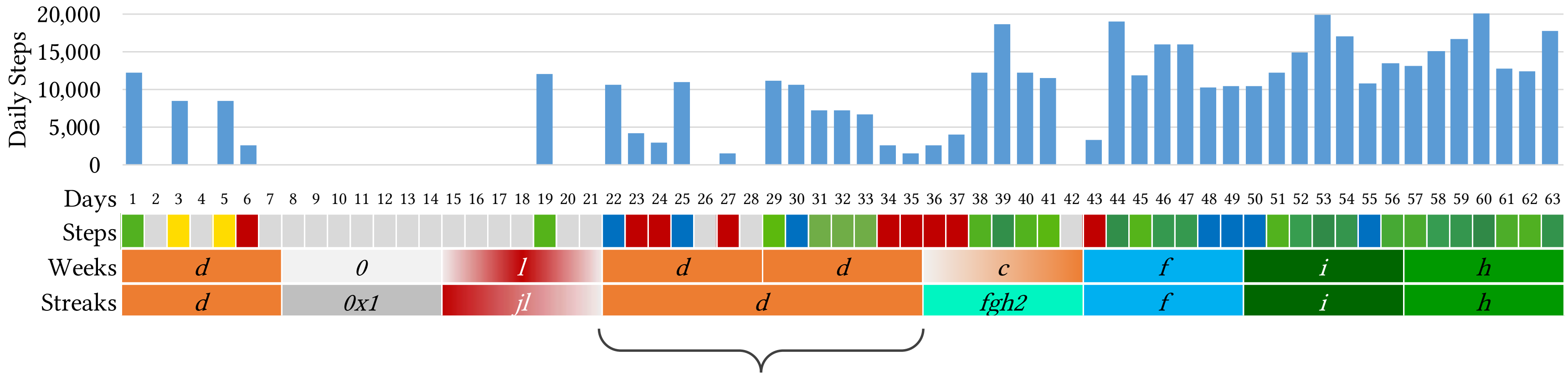
- Some **weeks** are more similar than others



Not in Exam

Challenge 3: Detecting Routine Habits and Changes

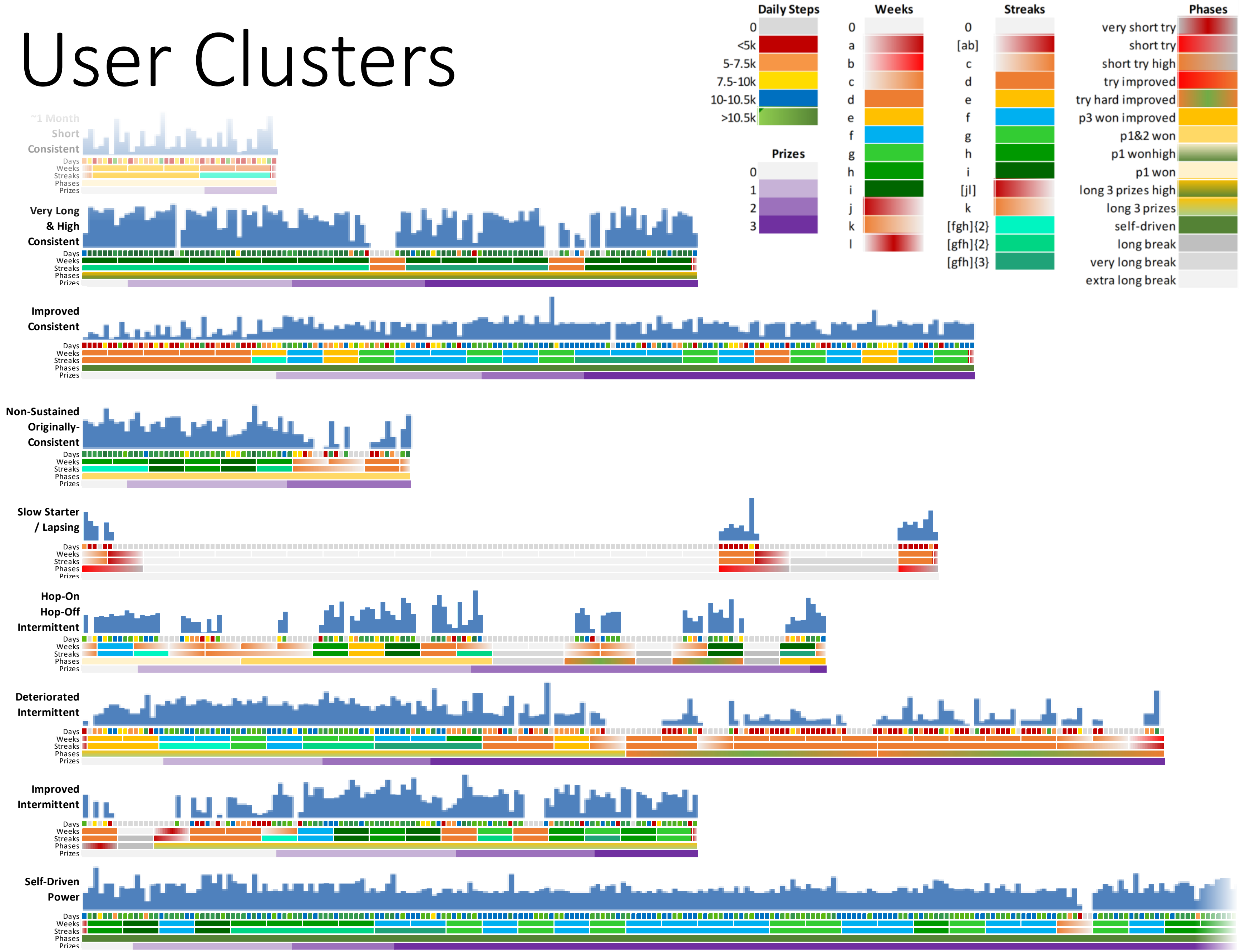
- Some **weeks** are more similar than others
- Some weeks continuously repeat as **streaks**



Streak of 2 similar weeks

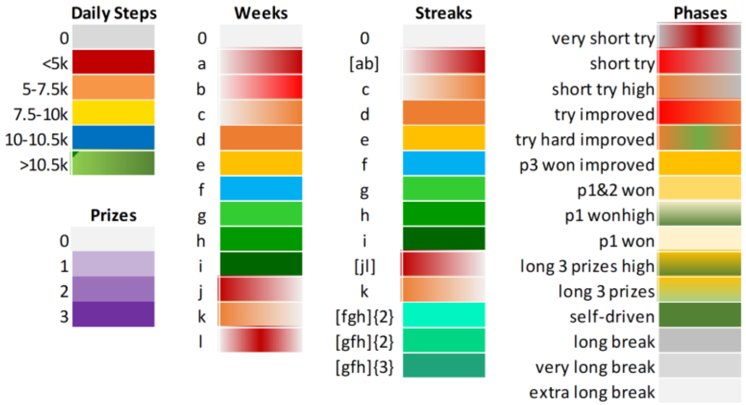
User Clusters

Not in Exam

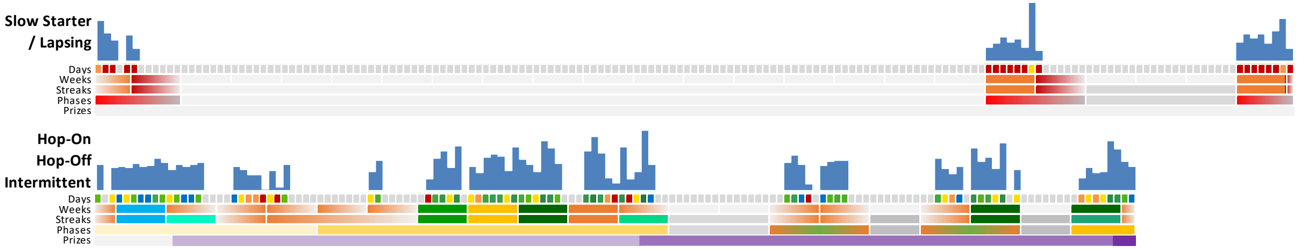


User Clusters

Not in Exam



- *Inconsistent **Slow Starter** users* who lapsed for months after initially tracking (*SS*)



- *Inconsistent **Hop-On Hop-Off*** users alternated between long non-active breaks and active days with moderate to high steps

Lim, B. Y., Kay, J., and Liu, W. 2019. **How does a nation walk? Interpreting large-scale step count activity with weekly streak patterns.** In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT).*


?
Questions!

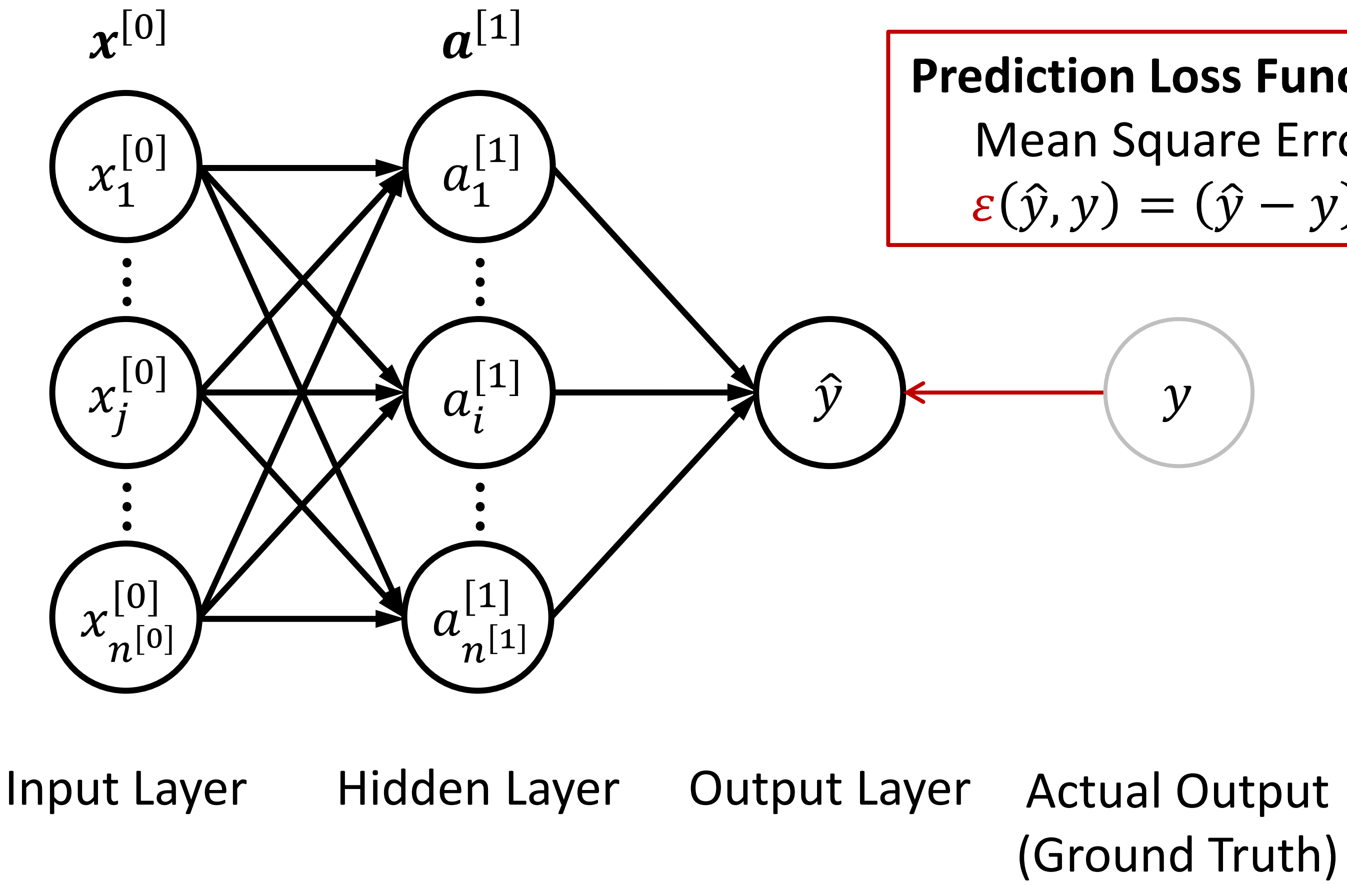
Auto-Encoders

Auto-Encoders

- What are auto-encoders (AE)?
- Types of Auto-Encoders
- How to train them?
- Applications

Neural Network

$$M(\boldsymbol{x}) = \hat{y}$$



Prediction Loss Function

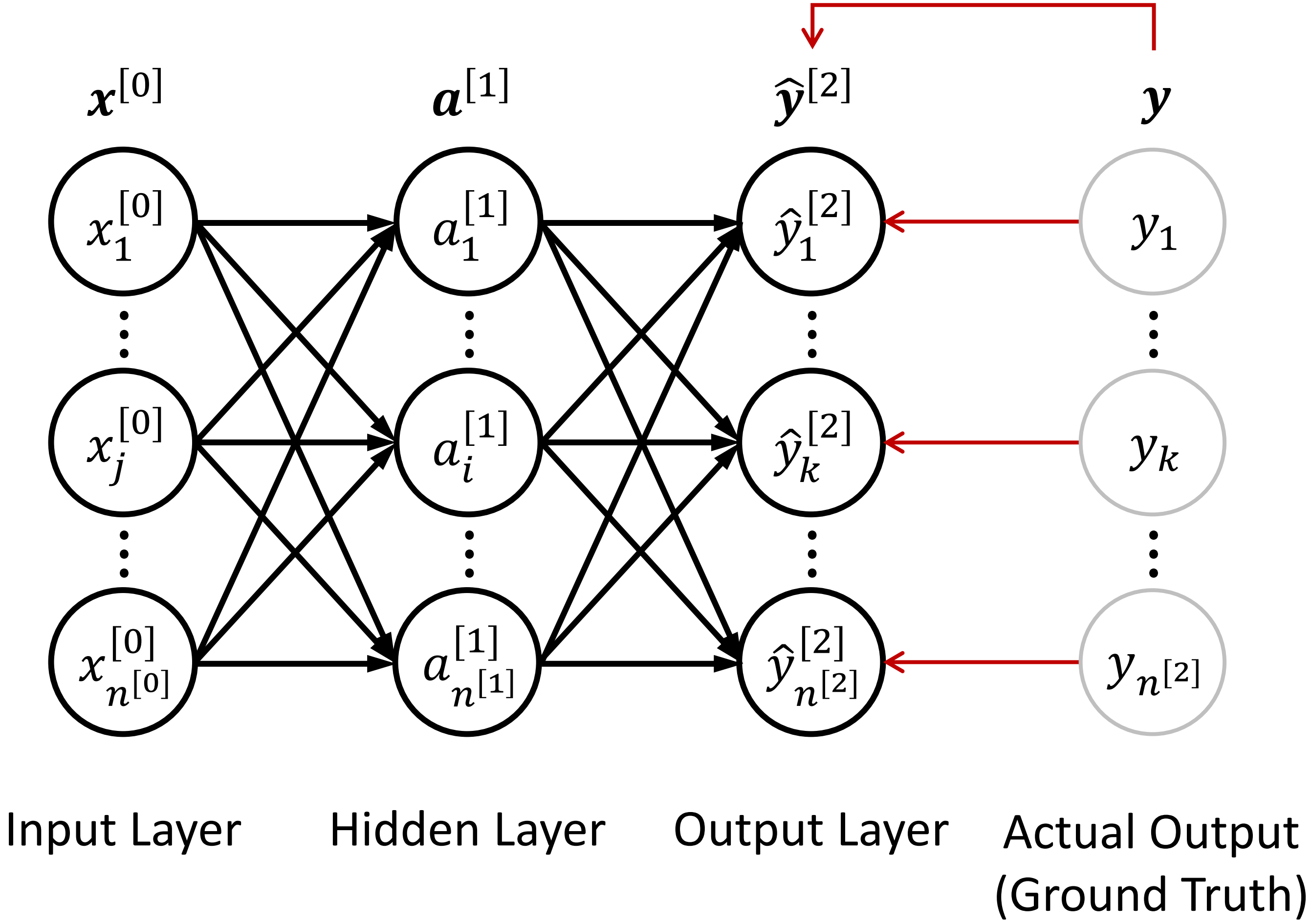
Mean Square Error

$$\varepsilon(\hat{y}, y) = (\hat{y} - y)^2$$

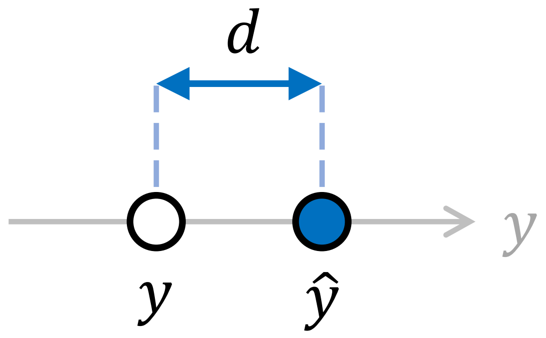
Neural Network (multiple outputs)

Prediction Loss Function
Squared Euclidean Distance
$\varepsilon(\hat{\boldsymbol{y}}, \boldsymbol{y}) = (\hat{\boldsymbol{y}} - \boldsymbol{y})^\top(\hat{\boldsymbol{y}} - \boldsymbol{y})$

$M(\boldsymbol{x}) = \hat{\boldsymbol{y}}$

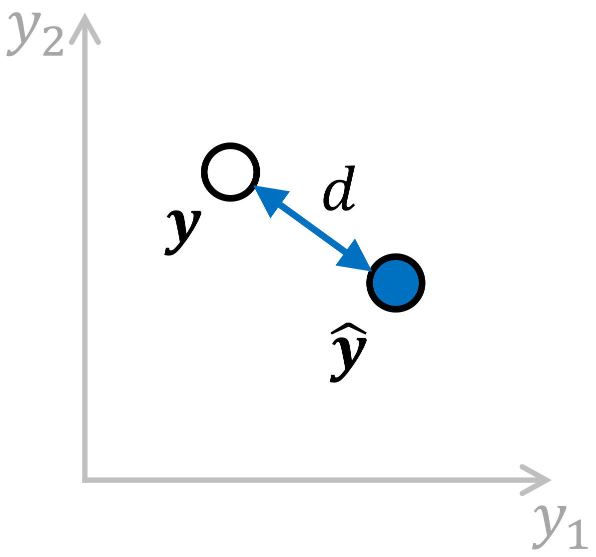


Vector Distances and Similarity



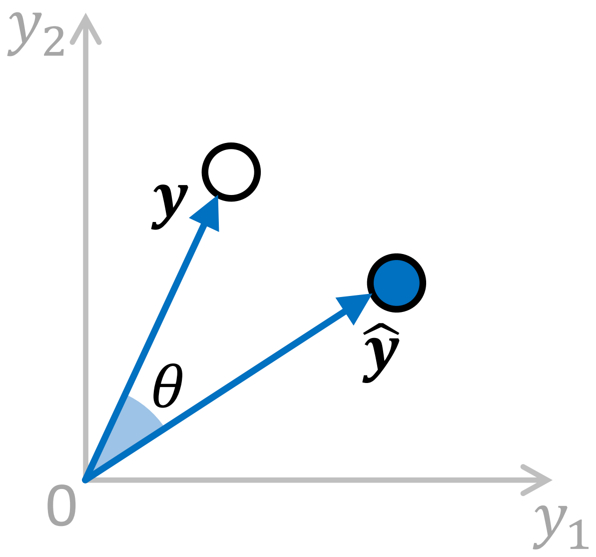
Squared Distance

$$d = (\hat{y} - y)^2$$



Euclidean Distance

$$d = \sqrt{\underbrace{(\hat{\boldsymbol{y}} - \boldsymbol{y})^\top (\hat{\boldsymbol{y}} - \boldsymbol{y})}_{\text{Dot Product}}}$$



Cosine Similarity

$$s = \cos(\theta) = \frac{\hat{\boldsymbol{y}} \cdot \boldsymbol{y}}{\|\hat{\boldsymbol{y}}\| \, \|\boldsymbol{y}\|}$$

Angular Distance

$$\theta = \cos^{-1}(s)$$

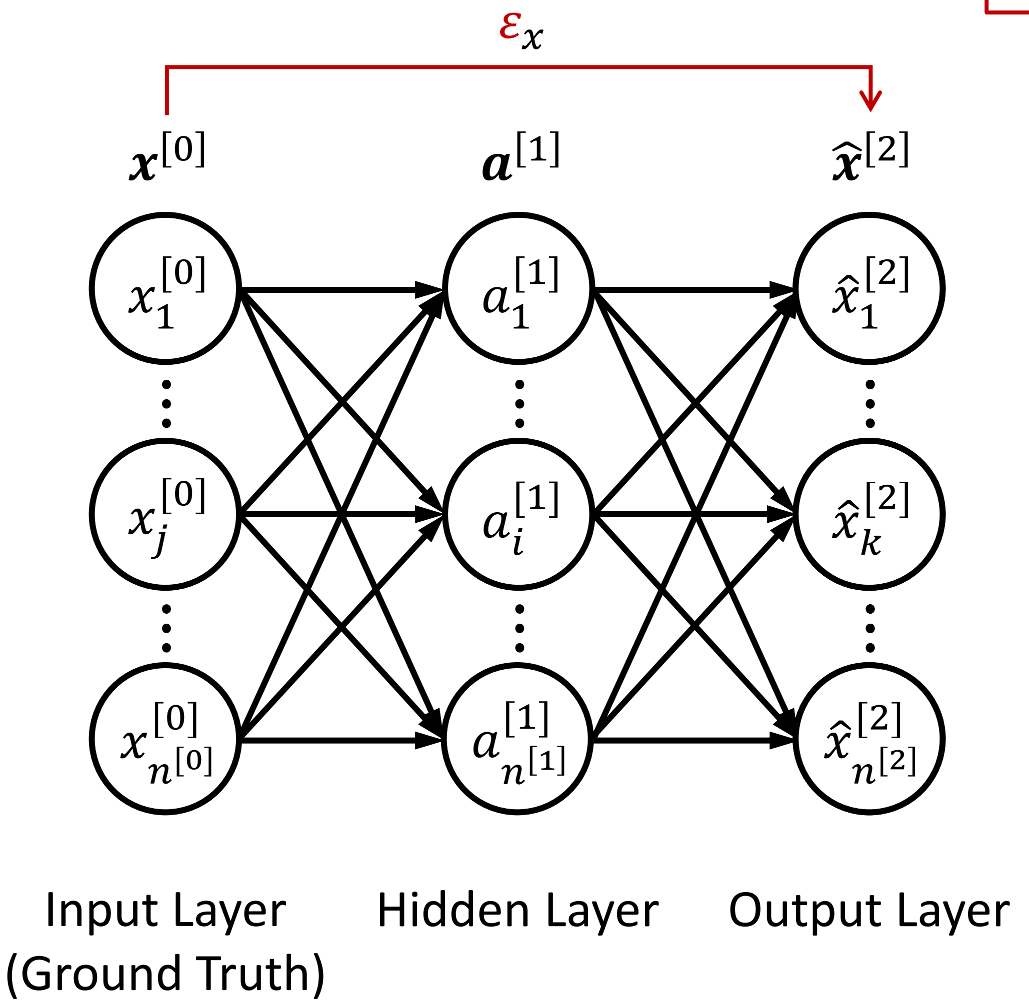
Auto-Encoder (AE)

Reconstruction Loss Function

Squared Euclidean Distance

$\varepsilon_x(\hat{\boldsymbol{x}}, \boldsymbol{x}) = (\hat{\boldsymbol{x}} - \boldsymbol{x})^\top (\hat{\boldsymbol{x}} - \boldsymbol{x})$

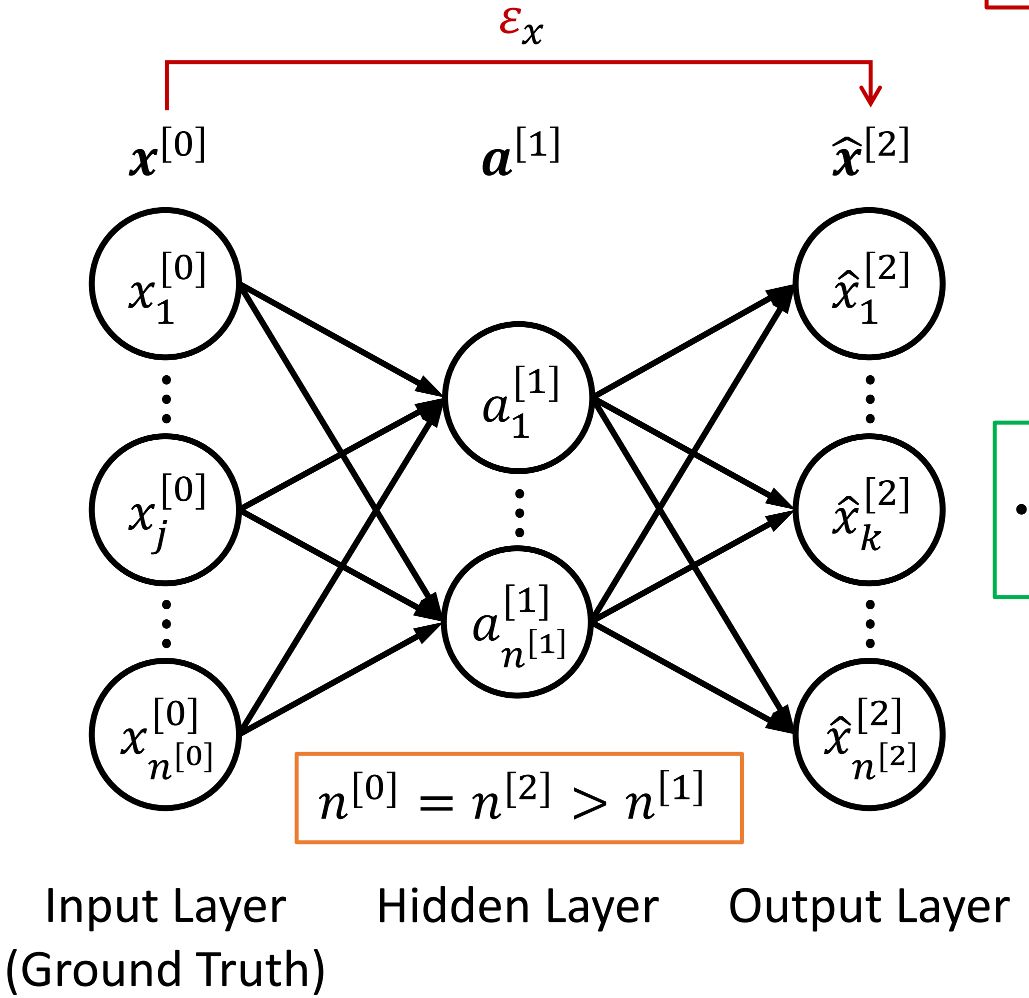
$M(\boldsymbol{x}) = \hat{\boldsymbol{x}}$



Auto-Encoder (with **Bottleneck**)

Reconstruction Loss Function
Squared Euclidean Distance
$\varepsilon_x(\hat{\boldsymbol{x}}, \boldsymbol{x}) = (\hat{\boldsymbol{x}} - \boldsymbol{x})^\top(\hat{\boldsymbol{x}} - \boldsymbol{x})$

$M(\boldsymbol{x}) = \hat{\boldsymbol{x}}$



Benefits
- Compressed representation of $\boldsymbol{x}$ in the middle layer

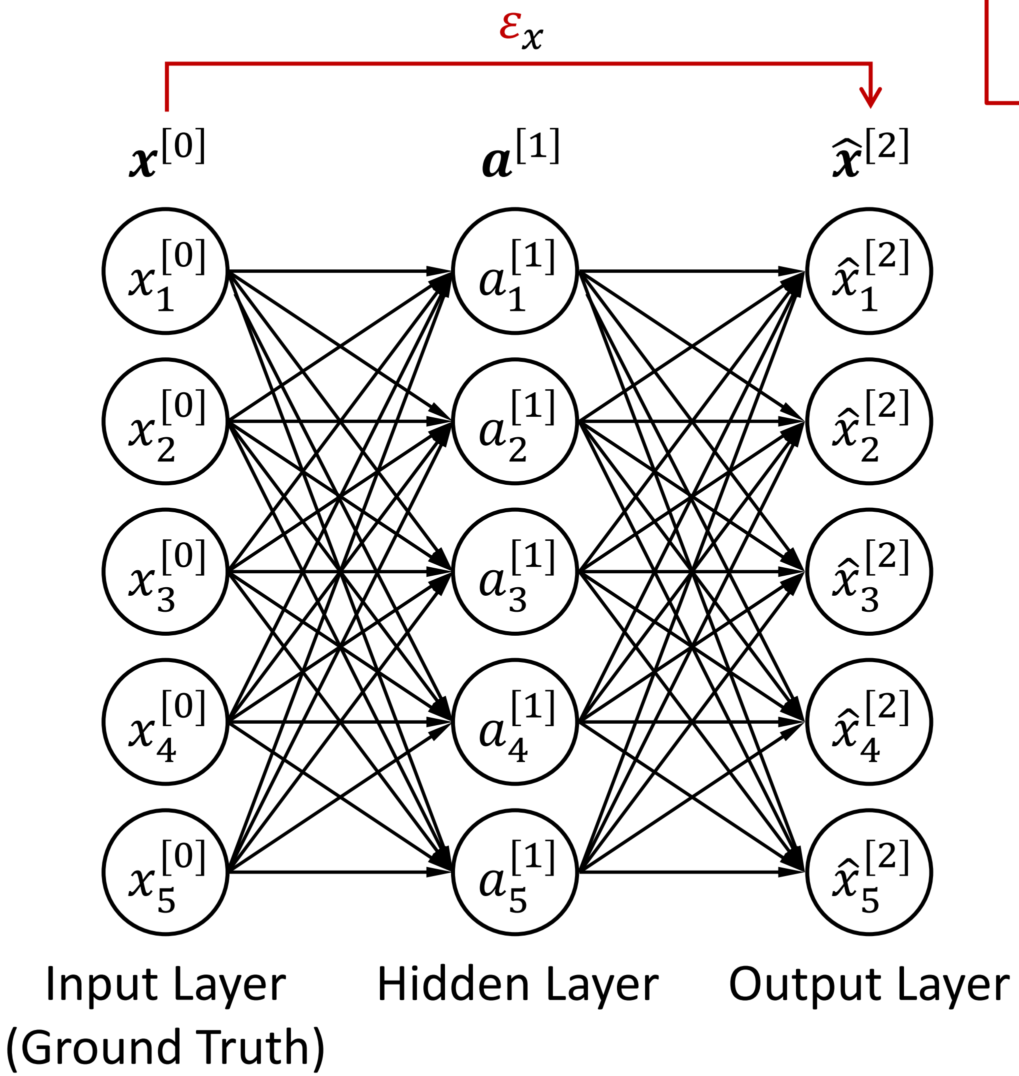
Plain Auto-Encoder

Reconstruction Loss Function

Squared Euclidean Distance

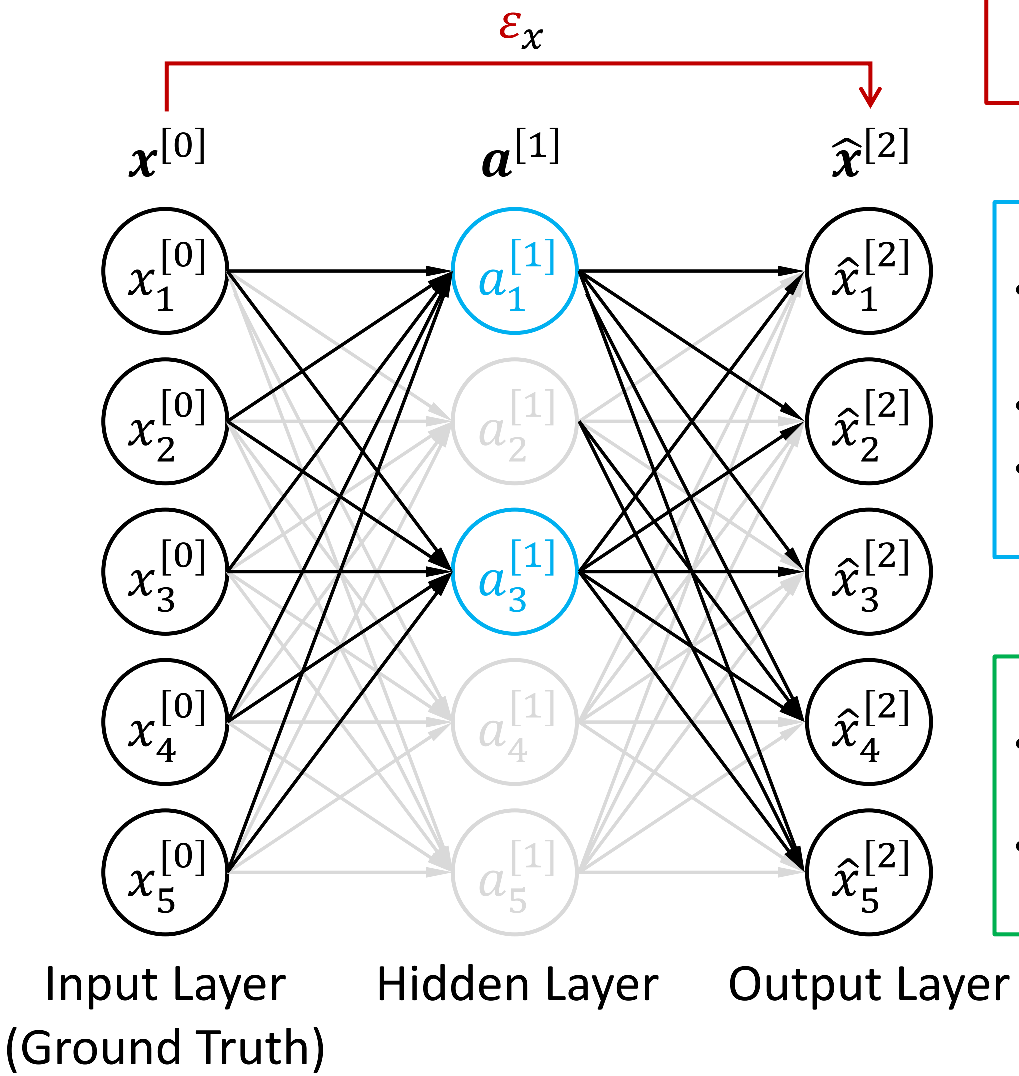
$\varepsilon(\hat{\boldsymbol{x}}, \boldsymbol{x}) = (\hat{\boldsymbol{x}} - \boldsymbol{x})^{\top}(\hat{\boldsymbol{x}} - \boldsymbol{x})$

$M(\boldsymbol{x}) = \hat{\boldsymbol{x}}$



Sparse Auto-Encoder

$$M(\boldsymbol{x}) = \hat{\boldsymbol{x}}$$



Reconstruction Loss Function

Squared Euclidean Distance

$$\varepsilon(\hat{\boldsymbol{x}}, \boldsymbol{x}) = (\hat{\boldsymbol{x}} - \boldsymbol{x})^{\top}(\hat{\boldsymbol{x}} - \boldsymbol{x}) + \lambda \left\| \boldsymbol{a}^{[1]} \right\|_1$$

Sparsity Regularization

- Penalizes having too many activations in hidden layer
- $\left\|\boldsymbol{a}^{[1]}\right\|_1 = \sum_{i=1}^{n^{[1]}} \left|a_i^{[1]}\right|$ is L1 norm
- λ is hyperparameter (higher value means more sparse)

Benefits

- Don't need to explicitly specify how many neurons in bottleneck
- Empirically higher performance than bottleneck AE. **Why?**

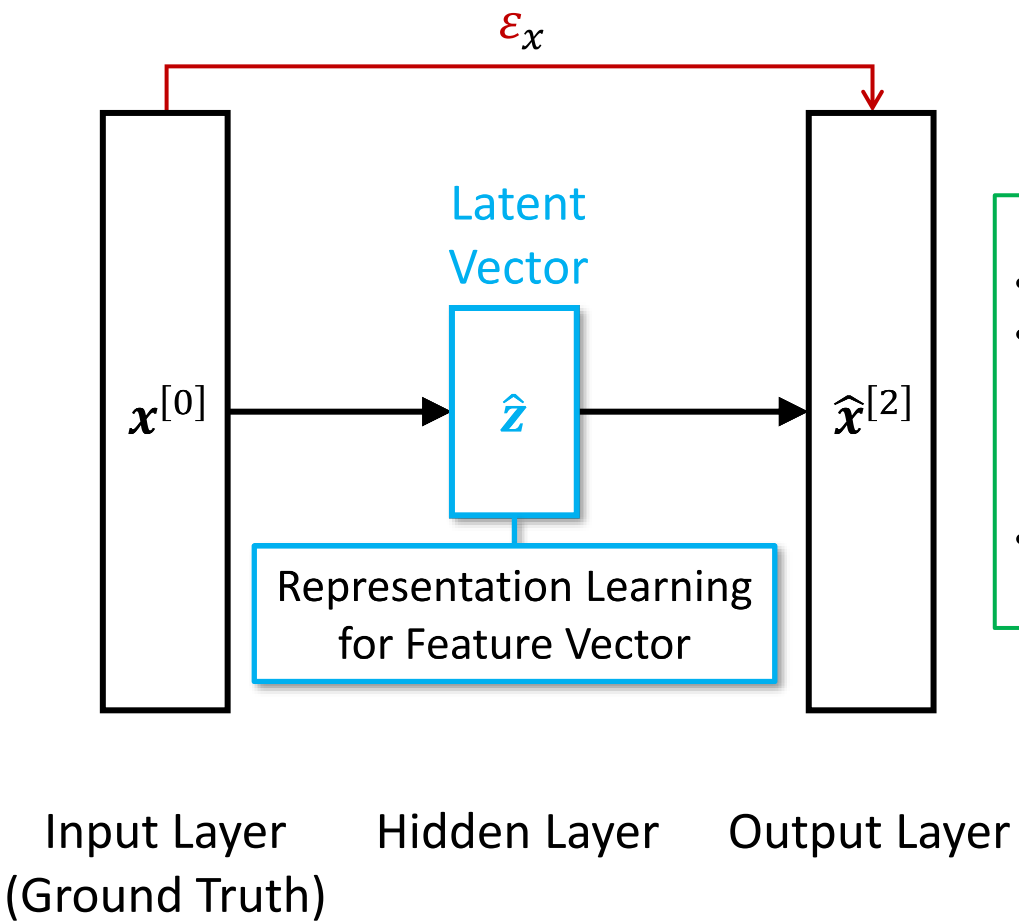
Auto-Encoder (layers as vectors)

Reconstruction Loss Function

Squared Euclidean Distance

$$\varepsilon_x(\hat{\boldsymbol{x}}, \boldsymbol{x}) = (\hat{\boldsymbol{x}} - \boldsymbol{x})^\top (\hat{\boldsymbol{x}} - \boldsymbol{x})$$

$$M(\boldsymbol{x}) = \hat{\boldsymbol{x}}$$



Benefits

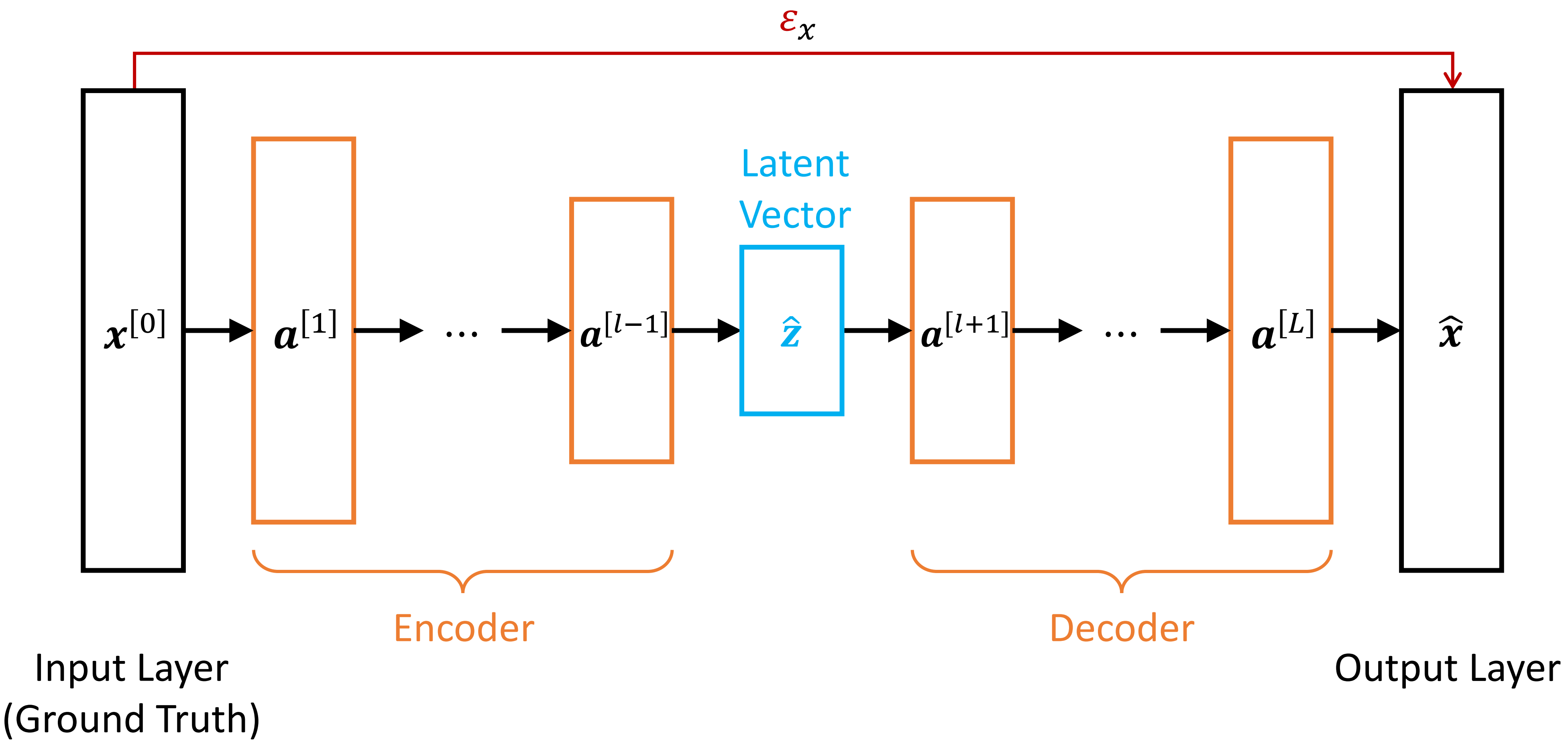
- Dimensionality Reduction
- If linear activations,
  - then similar to PCA,
  - but with non-orthogonal latent features.
- With non-linear activations, it can be better than PCA. **Why?**

Deep Auto-Encoder

Reconstruction Loss Function

Squared Euclidean Distance

$$\varepsilon_x(\hat{\boldsymbol{x}}, \boldsymbol{x}) = (\hat{\boldsymbol{x}} - \boldsymbol{x})^\top(\hat{\boldsymbol{x}} - \boldsymbol{x})$$

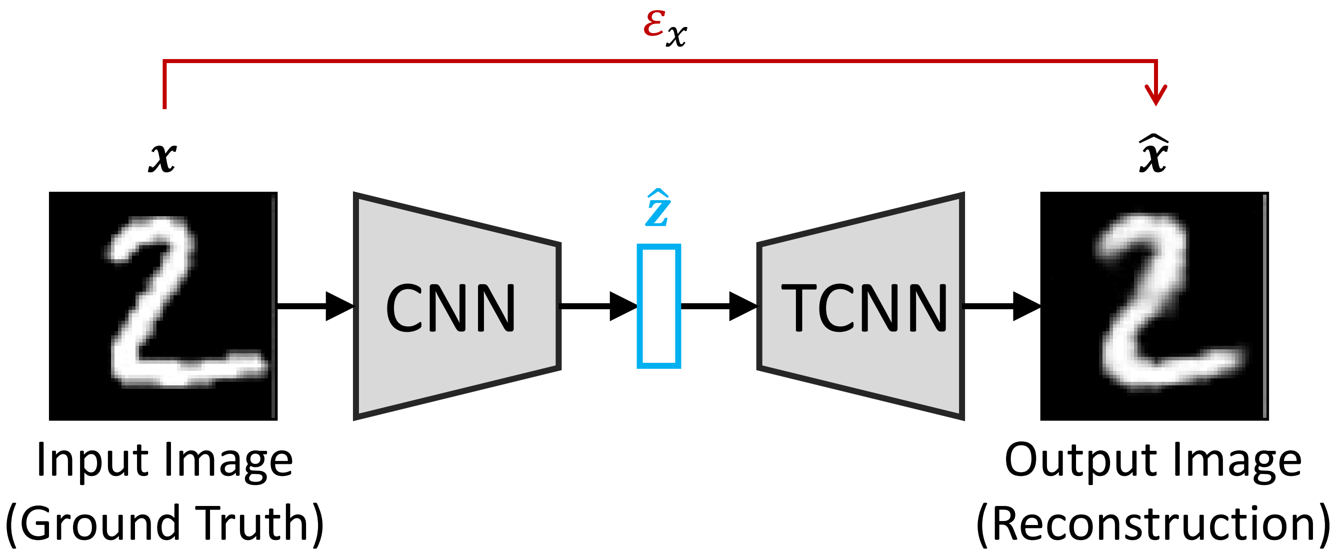


Deep Convolutional Auto-Encoder

Reconstruction Loss Function

Squared Euclidean Distance

$\varepsilon_x(\hat{\boldsymbol{x}}, \boldsymbol{x}) = (\hat{\boldsymbol{x}} - \boldsymbol{x})^\top(\hat{\boldsymbol{x}} - \boldsymbol{x})$

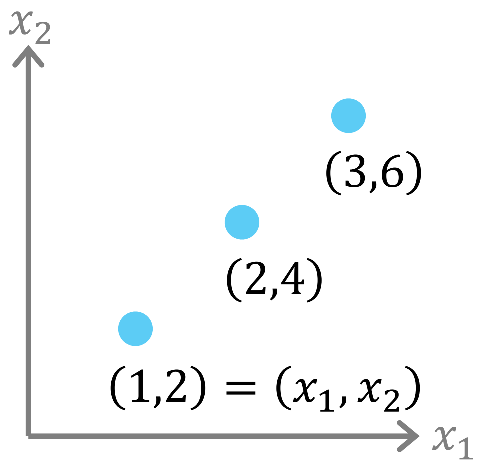


Auto-Encoders

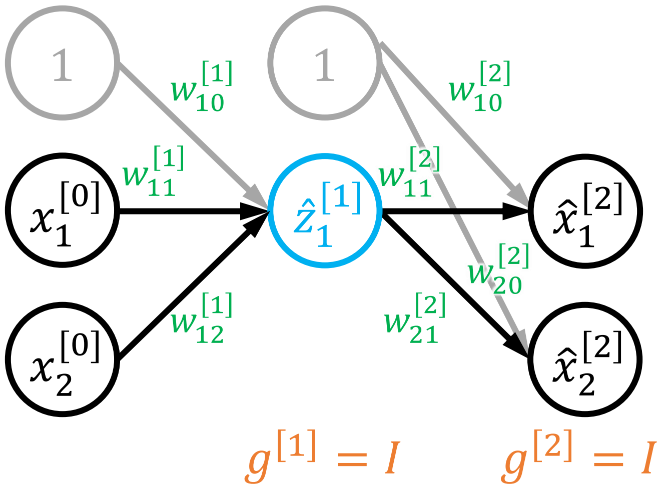
- What are auto-encoders (AE)?
  - Neural Networks to reconstruct the original inputs
- Types of Auto-Encoders
  - Plain Auto-Encoders
  - Auto-Encoders with Bottleneck
  - Sparse Auto-Encoders
  - Deep Auto-Encoders
  - Deep Convolutional Auto-Encoders
- How to train them?

Reconstructing with Auto-Encoder
What model weights can model (x_1, x_2)?

In Slack #general
1. **Write** to thread to suggest weights
2. **Emote** (👍 :+1:) to vote for weights



$$\boldsymbol{W}^{[1]} = \begin{pmatrix} w_{10}^{[1]} \\ w_{11}^{[1]} \\ w_{12}^{[1]} \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ ? \end{pmatrix}$$



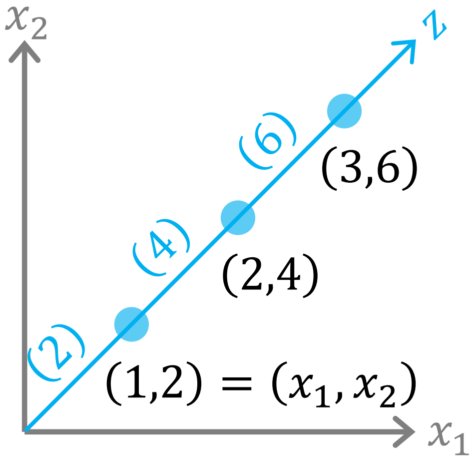
$$\boldsymbol{W}^{[2]} = \begin{pmatrix} w_{10}^{[2]} & w_{20}^{[2]} \\ w_{11}^{[2]} & w_{21}^{[2]} \end{pmatrix} = \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix}$$

E12a.2 Solution

Reconstructing with Auto-Encoder
What model weights can model (x_1, x_2)?

In Slack #general
1. **Write** to thread to suggest weights
2. **Emote** (👍 :+1:) to vote for weights

$$\hat{z} = 0x_0 + x_1 + 0.5x_2$$

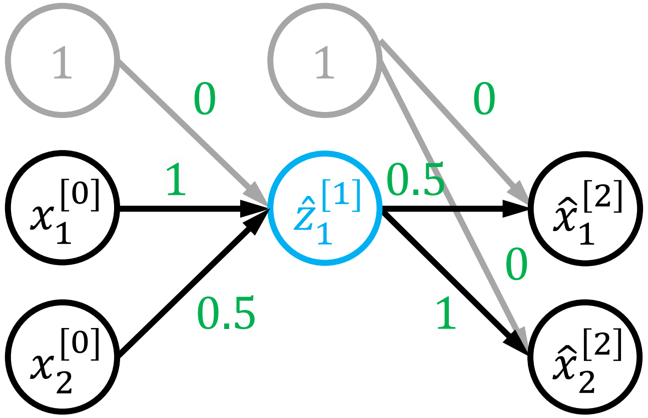


$$\boldsymbol{x}^{[0]} = \begin{pmatrix} 1 \\ x_1^{[0]} \\ x_2^{[0]} \end{pmatrix}$$

$$\hat{\boldsymbol{x}}^{[2]} = \begin{pmatrix} 0.5\hat{z}_1^{[1]} \\ 1\hat{z}_1^{[1]} \end{pmatrix}$$

$$\hat{\boldsymbol{z}}^{[1]} = \begin{pmatrix} 1 \\ \hat{z}_1^{[1]} \end{pmatrix}$$

$$\boldsymbol{W}^{[1]} = \begin{pmatrix} w_{10}^{[1]} \\ w_{11}^{[1]} \\ w_{12}^{[1]} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0.5 \end{pmatrix}$$



$g^{[1]} = I$ $\quad g^{[2]} = I$

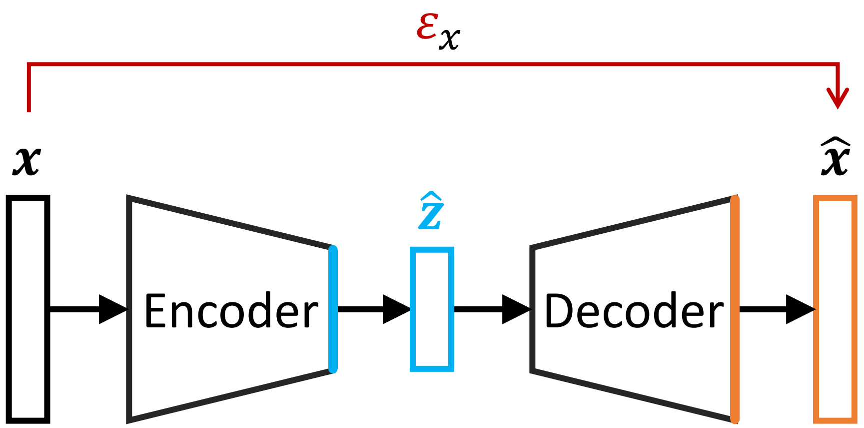
$$\boldsymbol{W}^{[2]} = \begin{pmatrix} w_{10}^{[2]} & w_{20}^{[2]} \\ w_{11}^{[2]} & w_{21}^{[2]} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0.5 & 1 \end{pmatrix}$$

$$\begin{aligned} \hat{z}_1^{[1]} &= \left(\boldsymbol{W}^{[1]}\right)^\top \boldsymbol{x}^{[0]} \\ &= w_{10}^{[1]} + w_{11}^{[1]} x_1^{[0]} + w_{21}^{[1]} x_2^{[0]} \boldsymbol{x}^{[0]} \\ &= 0 + 1x_1^{[0]} + 0.5x_2^{[0]} \end{aligned}$$

$$\begin{aligned} \hat{\boldsymbol{x}}^{[2]} &= \left(\boldsymbol{W}^{[2]}\right)^\top \hat{\boldsymbol{z}}^{[1]} = \begin{pmatrix} w_{10}^{[2]} + w_{11}^{[2]} \hat{z}_1^{[1]} \\ w_{20}^{[2]} + w_{21}^{[2]} \hat{z}_1^{[1]} \end{pmatrix} \\ &= \begin{pmatrix} 0.5\hat{z}_1^{[1]} \\ \hat{z}_1^{[1]} \end{pmatrix} \end{aligned}$$

Auto-Encoder Training

Reconstruction Loss Function
Squared Euclidean Distance
$$\varepsilon_x(\hat{\boldsymbol{x}}, \boldsymbol{x}) = (\hat{\boldsymbol{x}} - \boldsymbol{x})^\top (\hat{\boldsymbol{x}} - \boldsymbol{x})$$



$$E(\boldsymbol{x}) = \hat{\mathbf{z}} \qquad D(\hat{\mathbf{z}}) = \hat{\boldsymbol{x}}$$

$$D\big(E(\boldsymbol{x})\big) = \hat{\boldsymbol{x}}$$

Gradient
$$\frac{\partial \varepsilon_x}{\partial \boldsymbol{W}} = \frac{\partial \hat{\boldsymbol{x}}}{\partial \boldsymbol{W}} \frac{\partial \varepsilon_x}{\partial \hat{\boldsymbol{x}}} \qquad \hat{\boldsymbol{y}} \coloneqq \hat{\boldsymbol{x}}$$

$$\frac{\partial \hat{\boldsymbol{x}}}{\partial \boldsymbol{W}} = \frac{\partial D}{\partial \boldsymbol{W}} = \frac{\partial E}{\partial \boldsymbol{W}} \frac{\partial D}{\partial E} = \frac{\partial E}{\partial \boldsymbol{W}} \frac{\partial D}{\partial \hat{\mathbf{z}}}$$

$$\boxed{\frac{\partial D}{\partial \boldsymbol{W}^{[l_D]}} = \frac{\partial f^{[l_D]}}{\partial \boldsymbol{W}^{[l_D]}} \cdots \frac{\partial f^{[L_D]}}{\partial g^{[L_D-1]}} \frac{\partial g^{[L_D]}}{\partial f^{[L_D]}}}$$

$$l_E < l_D \qquad \frac{\partial E}{\partial \boldsymbol{W}^{[l_E]}} = \frac{\partial f^{[l_E]}}{\partial \boldsymbol{W}^{[l_E]}} \cdots \frac{\partial f^{[L_E]}}{\partial g^{[L_E-1]}} \frac{\partial g^{[L_E]}}{\partial f^{[L_E]}}$$

$$\boxed{\frac{\partial D}{\partial \boldsymbol{W}^{[l_E]}} = \frac{\partial E}{\partial \boldsymbol{W}^{[l_E]}} \frac{\partial D}{\partial \hat{\mathbf{z}}}}$$

Take-Away:
Backprop gradient descent for weight update
$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{\partial \varepsilon_x}{\partial \boldsymbol{W}}$$
Same as all neural networks, but through 2 models

Auto-Encoders

- What are auto-encoders (AE)?
  - Neural Networks to reconstruct the original inputs
- Types of Auto-Encoders
- How to train them?
  - Gradient Descent → Weight Updates
- Applications

Auto-Encoder
for **Feature Representation Learning**

Reconstruction Loss Function

Squared Euclidean Distance

$\varepsilon_x(\hat{\boldsymbol{x}}, \boldsymbol{x}) = (\hat{\boldsymbol{x}} - \boldsymbol{x})^\top (\hat{\boldsymbol{x}} - \boldsymbol{x})$

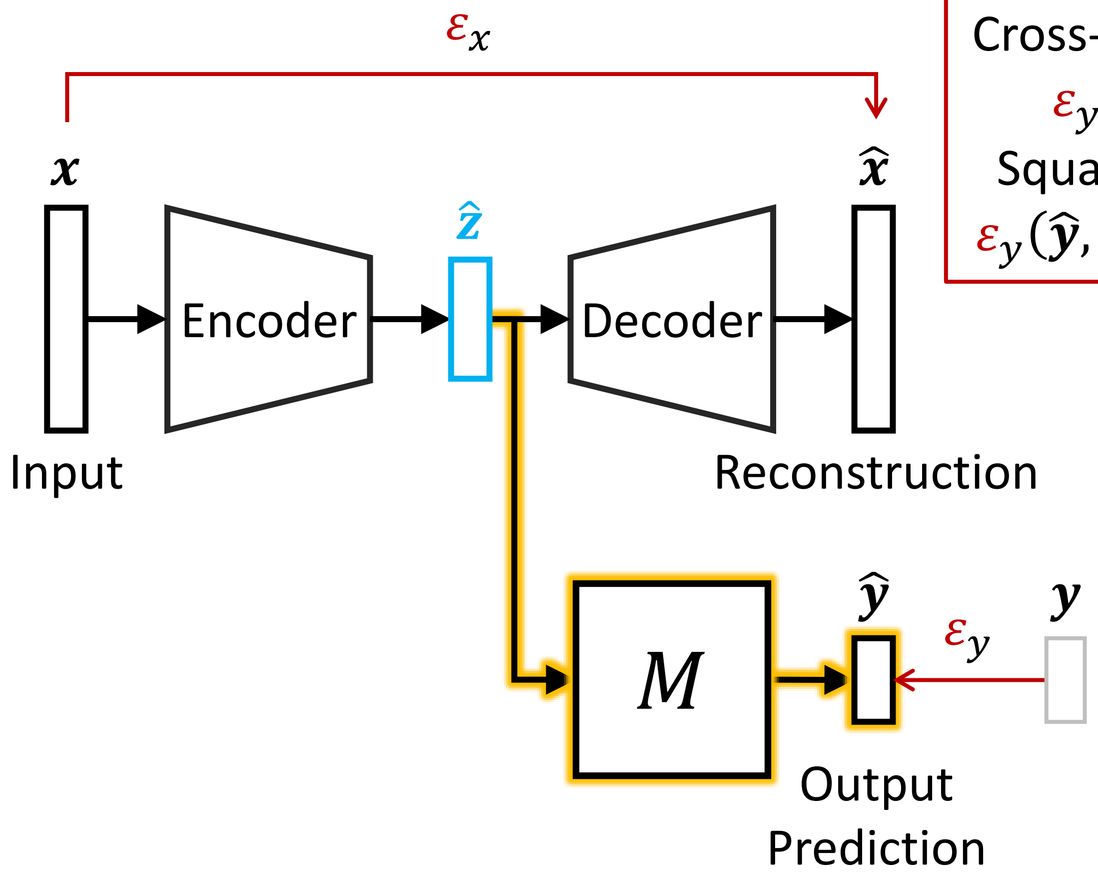
Prediction Loss Function

Cross-Entropy (Classification)

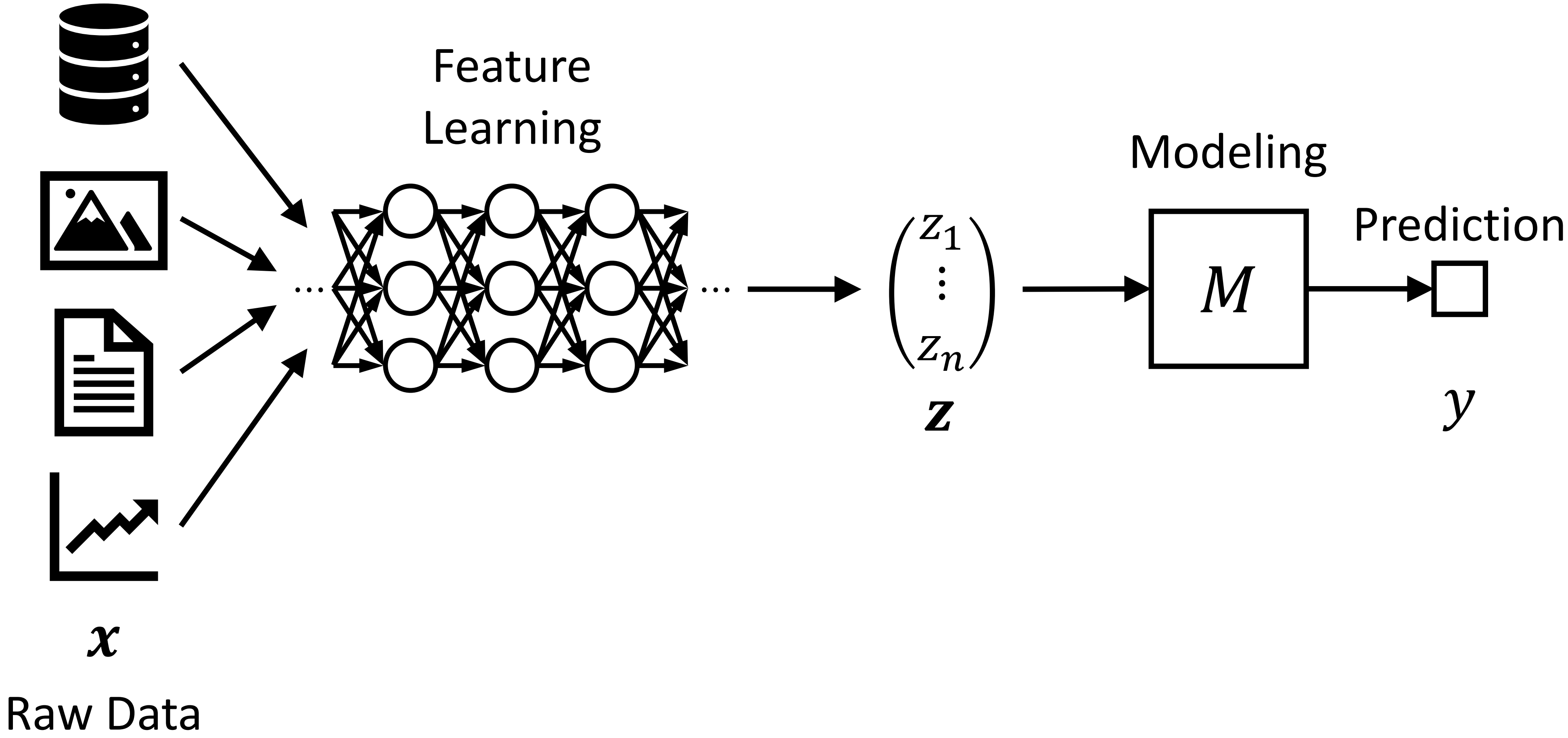
$\varepsilon_y(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\boldsymbol{y}^\top \log \hat{\boldsymbol{y}}$

Squared Error (Regression)

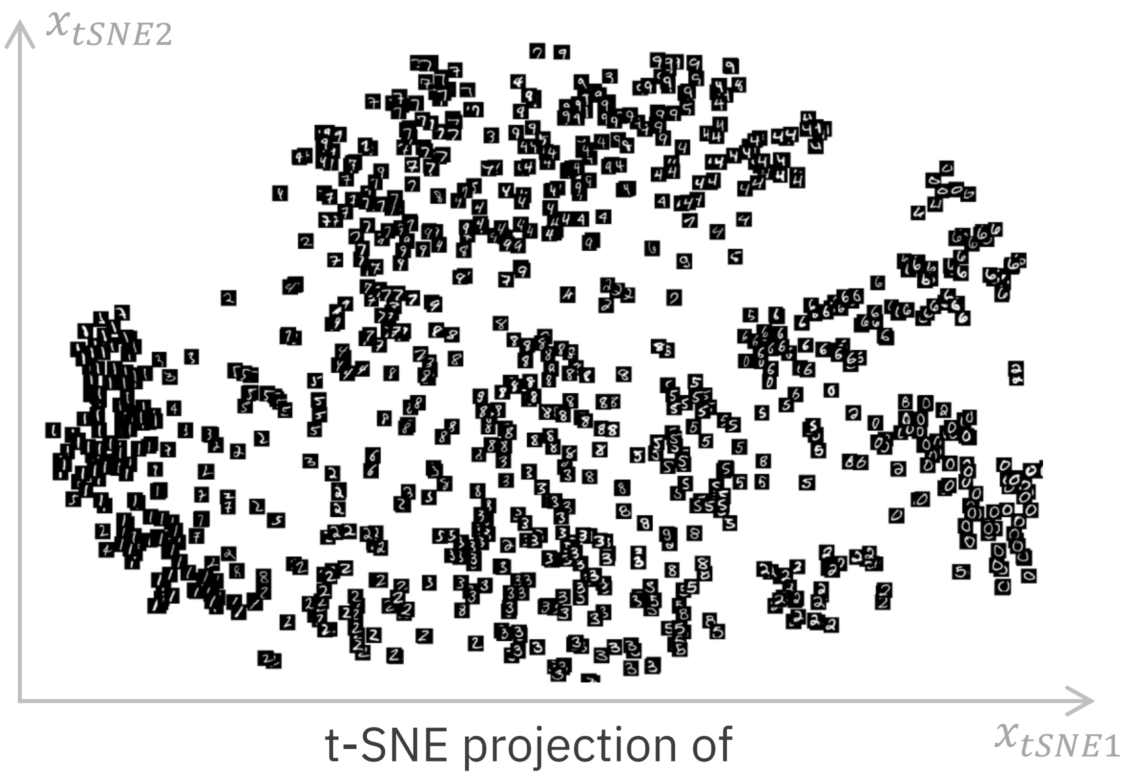
$\varepsilon_y(\hat{\boldsymbol{y}}, \boldsymbol{y}) = (\hat{\boldsymbol{y}} - \boldsymbol{y})^\top (\hat{\boldsymbol{y}} - \boldsymbol{y})$



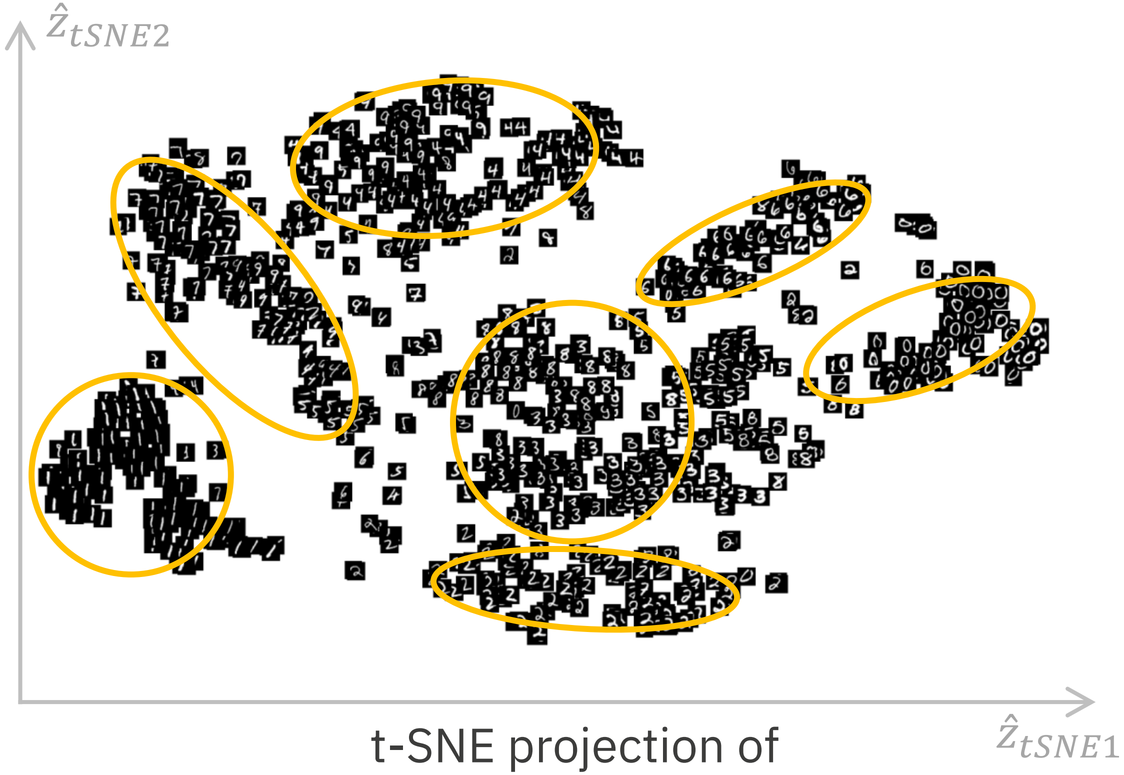
From Manual Feature Engineering To Automatic Feature Learning



Auto-Encoder
for **Feature Representation Learning Clustering**



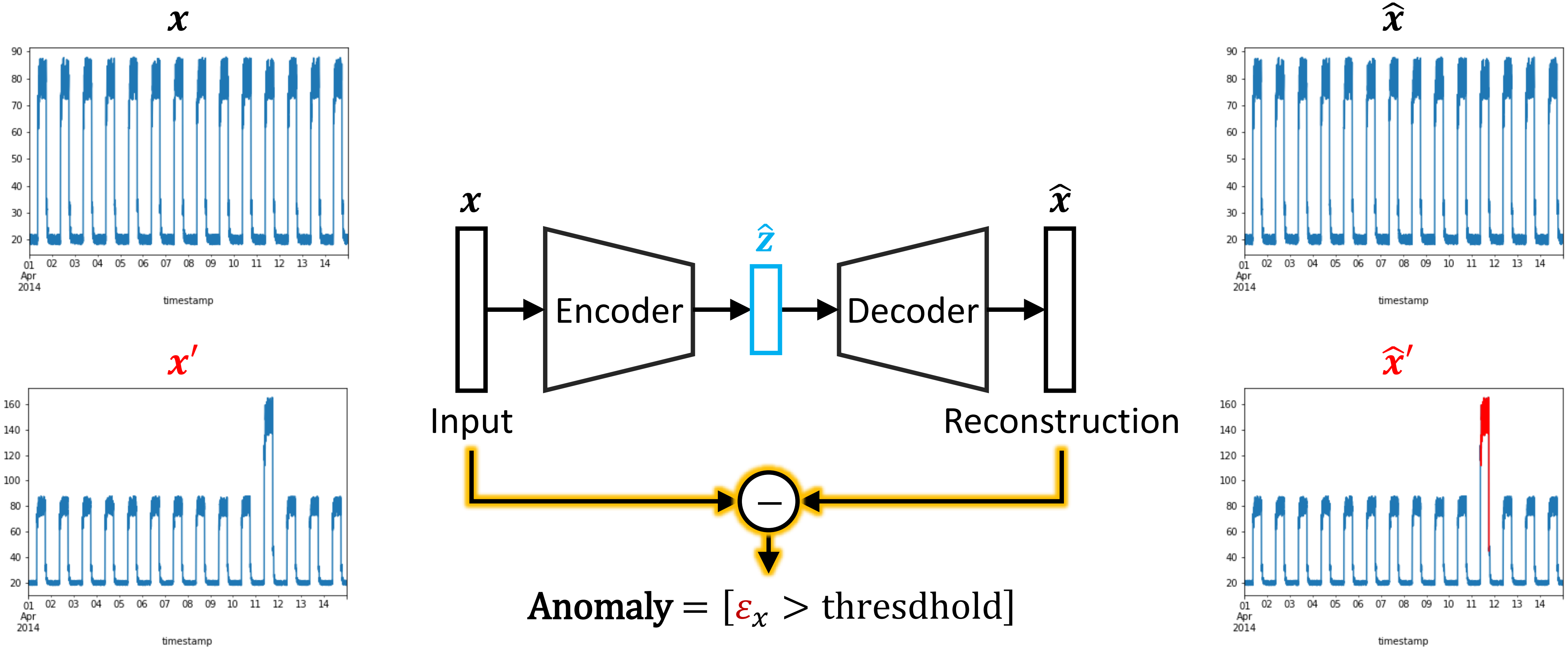
t-SNE projection of
image space x representations



t-SNE projection of
latent space $\hat{z}$ representations

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

Auto-Encoder for **Anomaly Detection**



$$\textbf{Anomaly} = [\varepsilon_x > \text{thresdhold}]$$

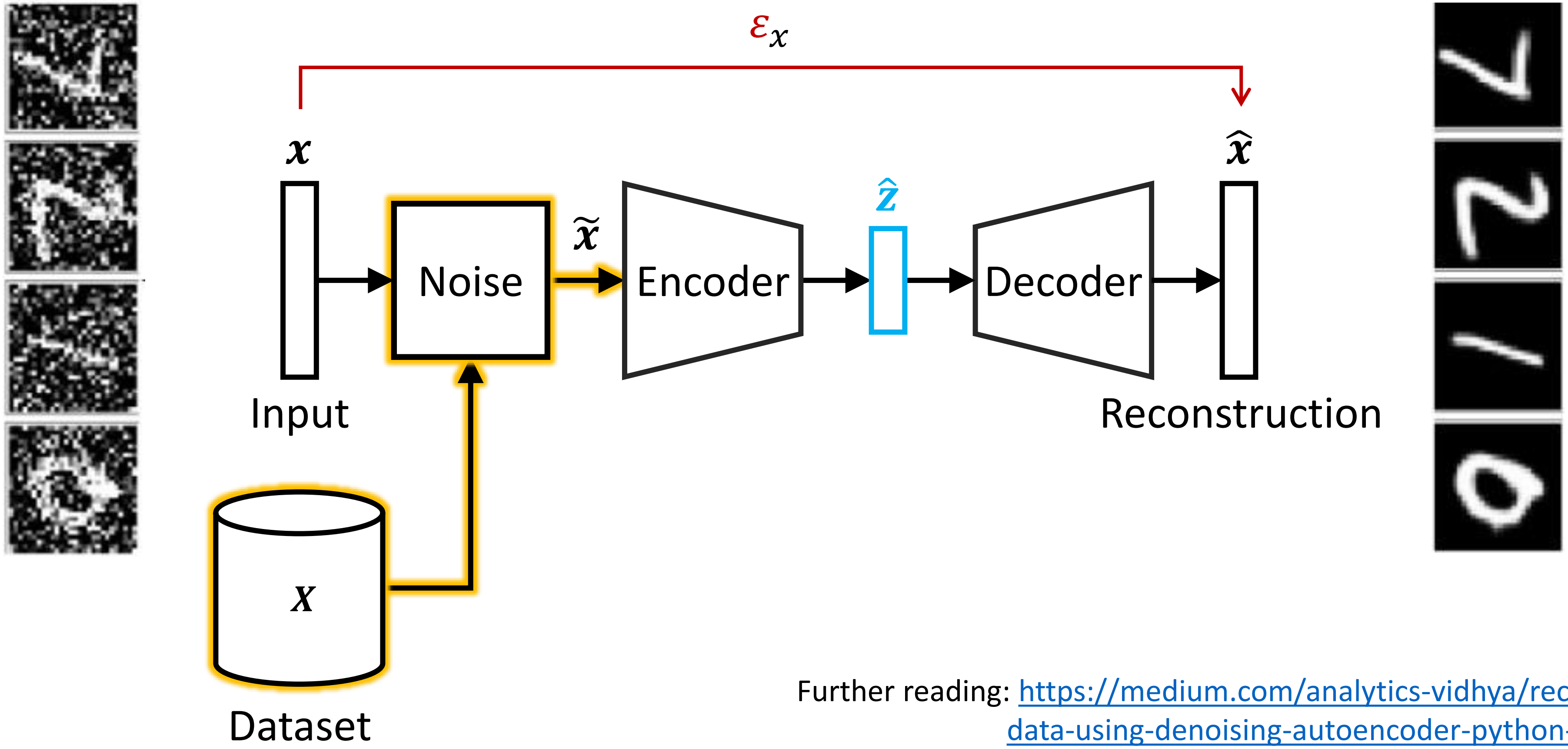
Further Reading: https://keras.io/examples/timeseries/timeseries_anomaly_detection/

Denoising Auto-Encoder for **Robust Learning**

Reconstruction Loss Function

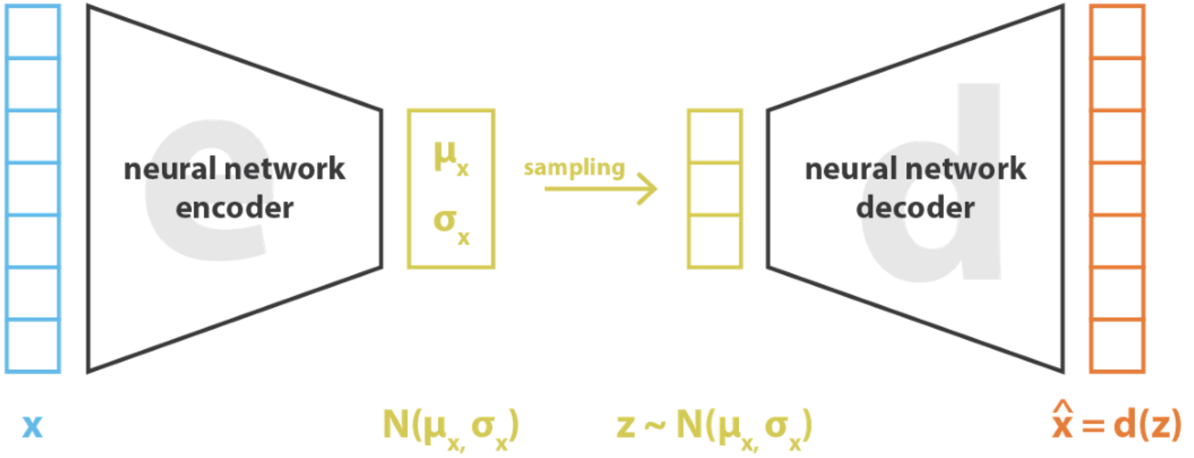
Squared Euclidean Distance

$$\varepsilon_x(\hat{\boldsymbol{x}}, \boldsymbol{x}) = (\hat{\boldsymbol{x}} - \boldsymbol{x})^\top(\hat{\boldsymbol{x}} - \boldsymbol{x})$$

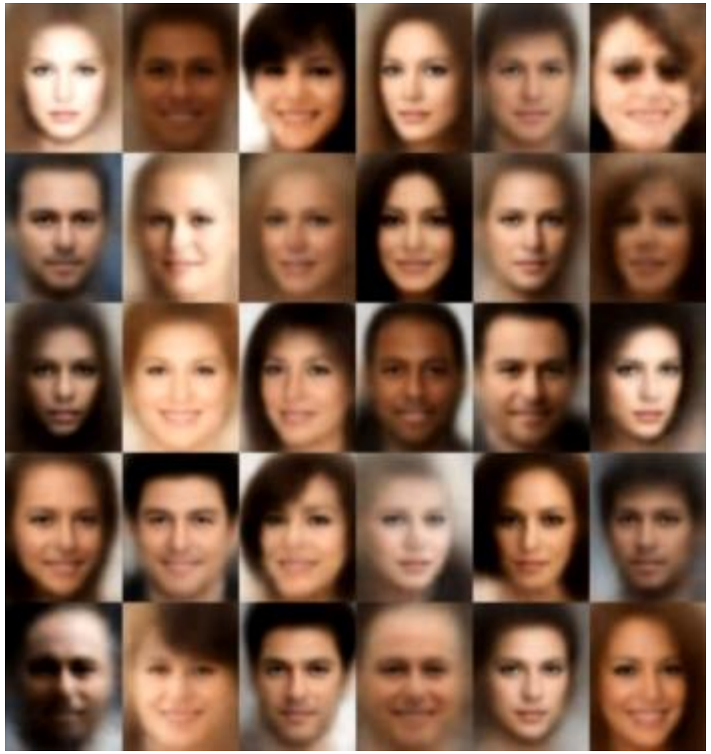


Further reading: https://medium.com/analytics-vidhya/reconstruct-corrupted-data-using-denoising-autoencoder-python-code-aeaff4b0958e

Variational Auto-Encoder for Generative **Data Synthesis**



$$\text{loss} = \| x - \hat{x} \|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,] = \| x - d(z) \|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,]$$



Face images generated with a Variational Autoencoder (source: [Wojciech Mormul on Github](#))

Further Reading: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Auto-Encoders Applications

- Dimensionality Reduction
- Feature Representation Learning
- Anomaly detection
- Noise removal
- Data synthesis


?
Questions!


VOICE CLONING
TWO MINUTE
PAPERS

Wrapping Up

What did we learn?

- Unsupervised Clustering to summarize and group data
- k-Means clustering to group tabular data into k clusters
- Auto-Encoder for unsupervised feature representation learning
  - Can then classify or cluster on latent features
  - Allows clustering on unstructured data

Outlook for next week

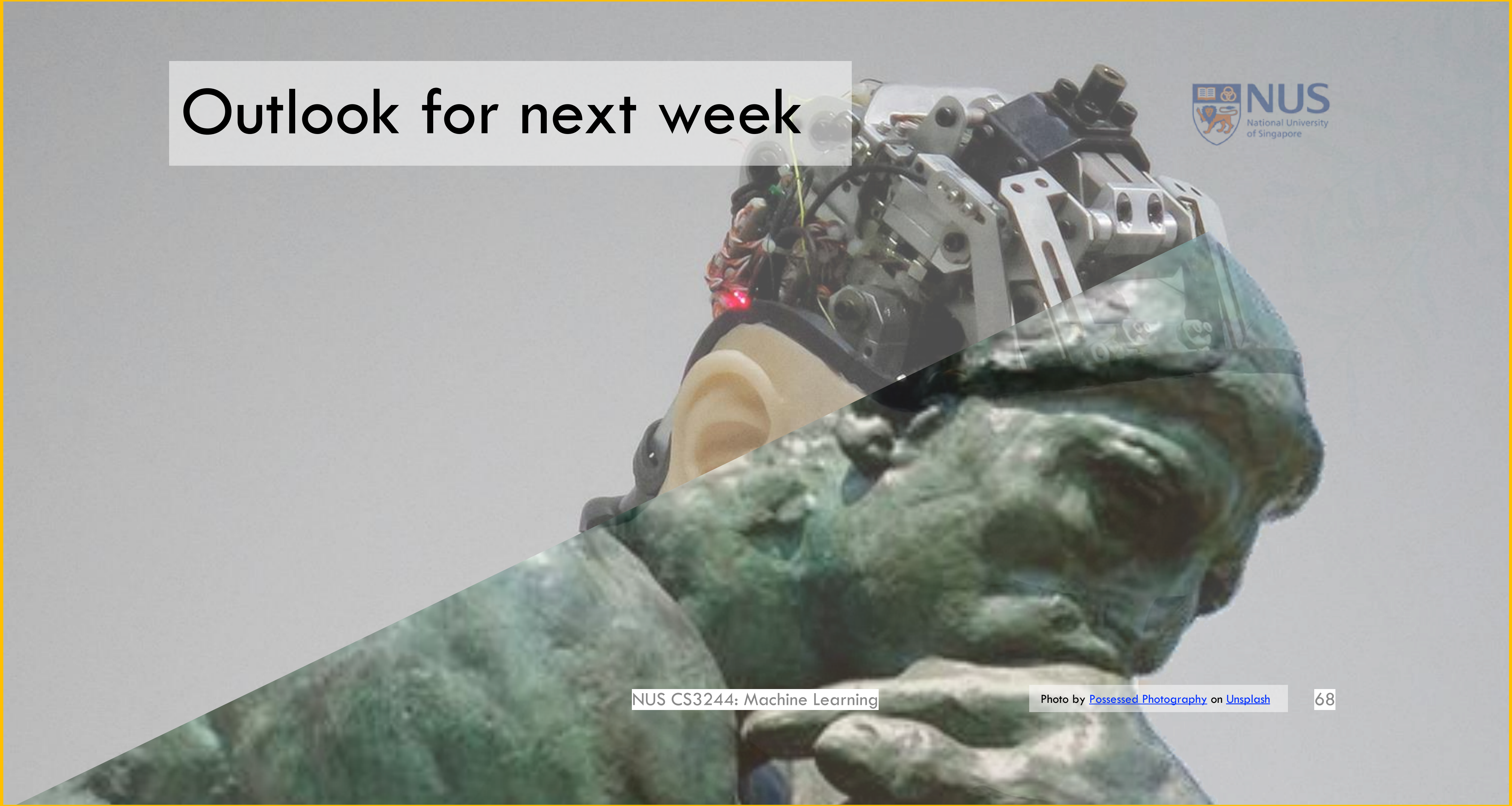


Photo by Possessed Photography on Unsplash

Assigned Task (due before next Mon)

ML Algorithms Behaving Badly. Find an online article about a machine learning system (prototype or fielded) behaving badly and post it to the corresponding thread in your #tg-xx.

In your post, also include 1-2 sentence comment that reacts to the comment below with respect to your post as context.

Machine Learning algorithms are based on purely mathematical constructs and thus cannot be biased.