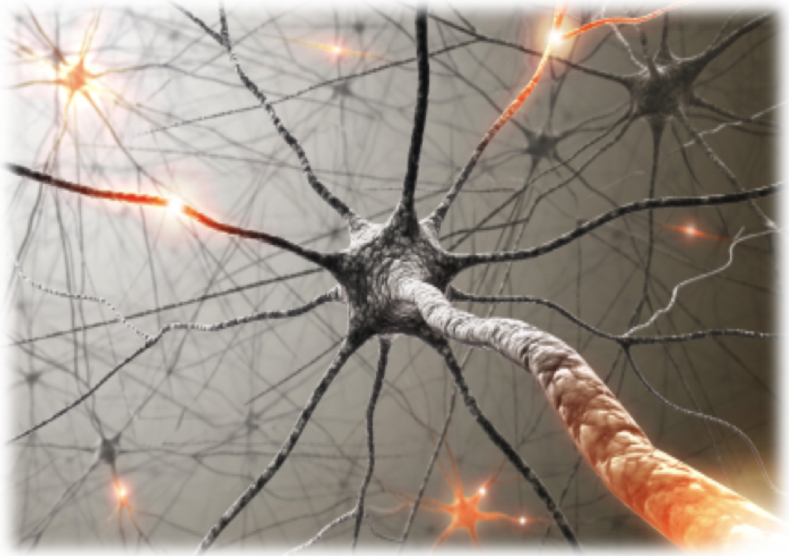
Machine learning: k-means



Word clustering

Input: raw text (100 million words of news articles)...

Output:

Cluster 1: Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends Sundays Saturdays
Cluster 2: June March July April January December October November September August
Cluster 3: water gas coal liquid acid sand carbon steam shale iron
Cluster 4: great big vast sudden mere sheer gigantic lifelong scant colossal
Cluster 5: man woman boy girl lawyer doctor guy farmer teacher citizen
Cluster 6: American Indian European Japanese German African Catholic Israeli Italian Arab
Cluster 7: pressure temperature permeability density porosity stress velocity viscosity gravity tension
Cluster 8: mother wife father son husband brother daughter sister boss uncle
Cluster 9: machine device controller processor CPU printer spindle subsystem compiler plotter
Cluster 10: John George James Bob Robert Paul William Jim David Mike
Cluster 11: anyone someone anybody somebody
Cluster 12: feet miles pounds degrees inches barrels tons acres meters bytes
Cluster 13: director chief professor commissioner commander treasurer founder superintendent dean custodian
Cluster 14: had hadn't hath would've could've should've must've might've
Cluster 15: head body hands eyes voice arm seat eye hair mouth

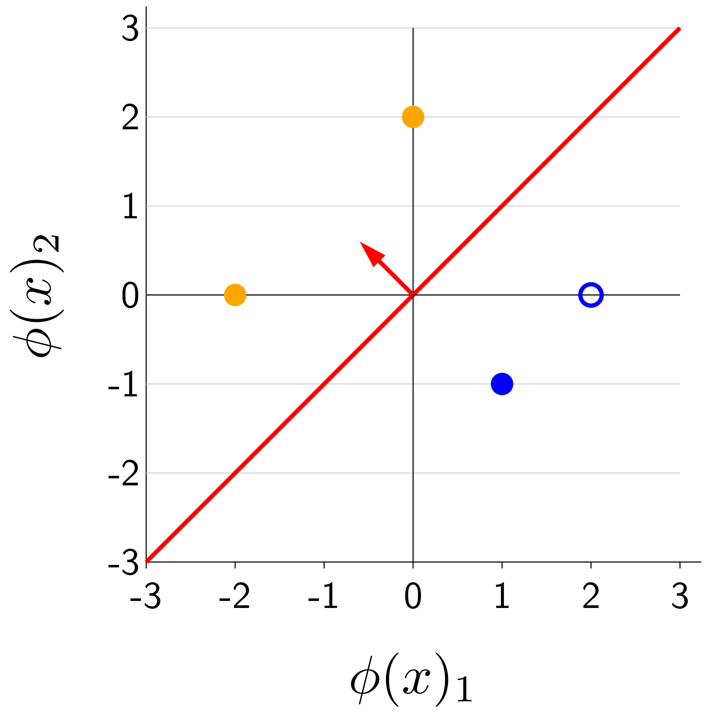
Classification (supervised learning)

training data $\mathcal{D}_{\text{train}}$

$\phi(x)_1$	$\phi(x)_2$	y
0	2	1
-2	0	1
1	-1	-1

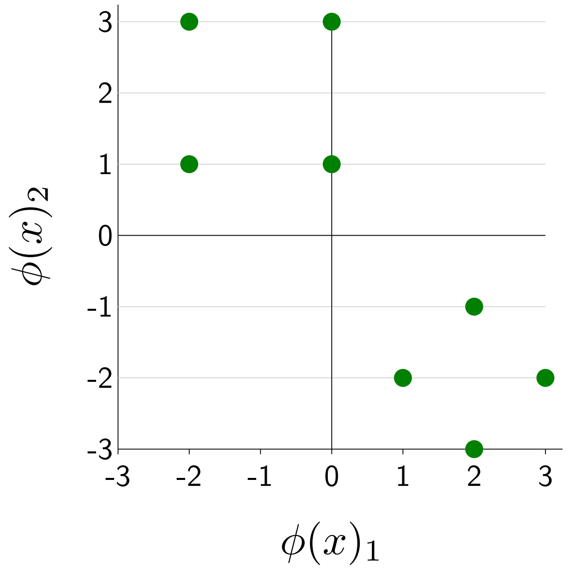
learning algorithm → f **classifier**

[2, 0] → f → -1



Labeled data is expensive to obtain

Clustering (unsupervised learning)



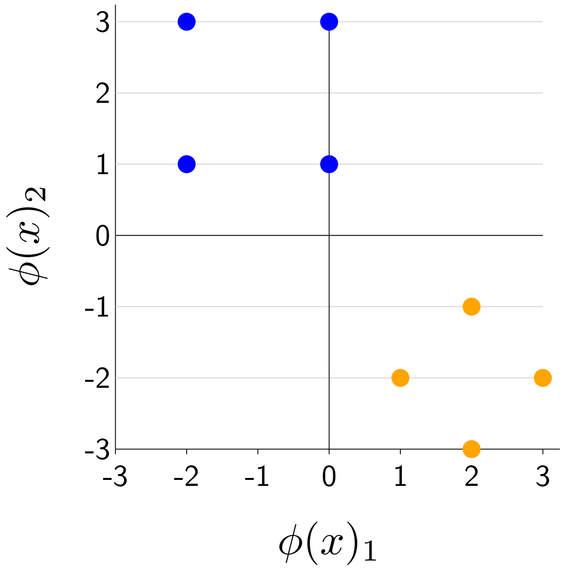
training data $\mathcal{D}_{\text{train}}$

$\phi(x)_1$	$\phi(x)_2$
-2	1
0	1
-2	3
0	3
2	-1
1	-2
2	-3
3	-2

learning algorithm →

assignments

z
1
1
1
1
2
2
2
2



Intuition: Want to assign nearby points to same cluster

Unlabeled data is very cheap to obtain

Clustering task

Definition: clustering

Input: training points

$$\mathcal{D}_{\text{train}} = [x_1, \dots, x_n]$$

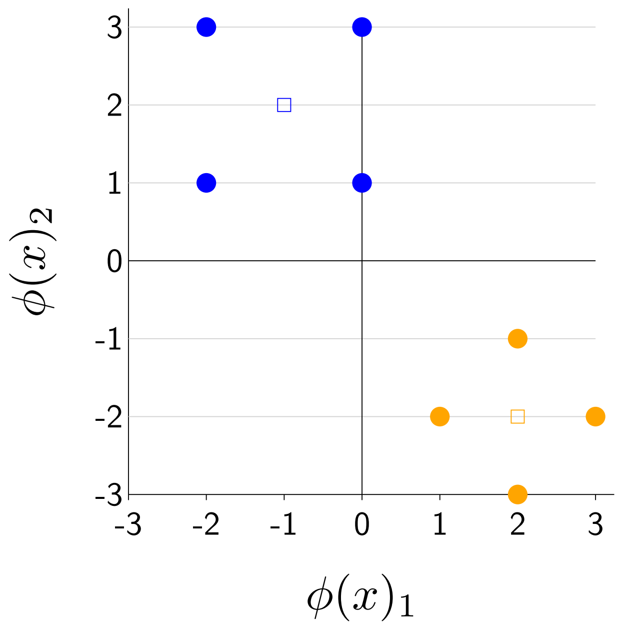
Output: assignment of each point to a cluster

$$\mathbf{z} = [z_1, \dots, z_n] \text{ where } z_i \in \{1, \dots, K\}$$

Centroids

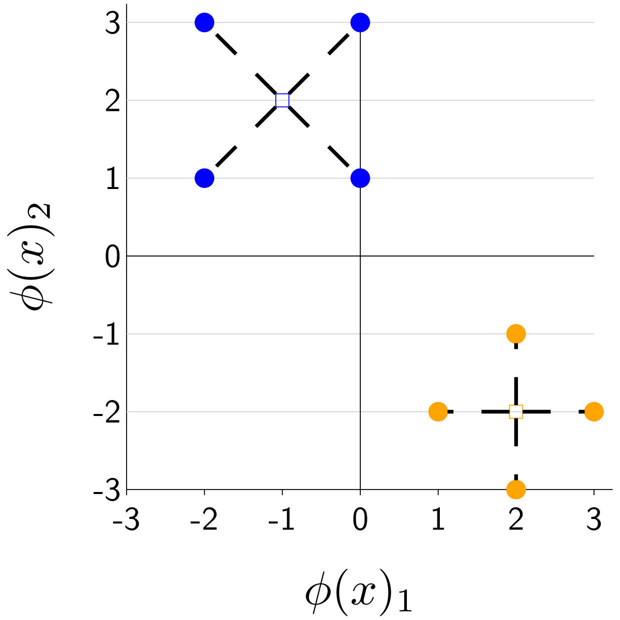
Each cluster $k = 1, \ldots, K$ is represented by a **centroid** $\mu_k \in \mathbb{R}^d$

$$\boldsymbol{\mu} = [\mu_1, \ldots, \mu_K]$$



Intuition: want each point $\phi(x_i)$ to be close to its assigned centroid μ_{z_i}

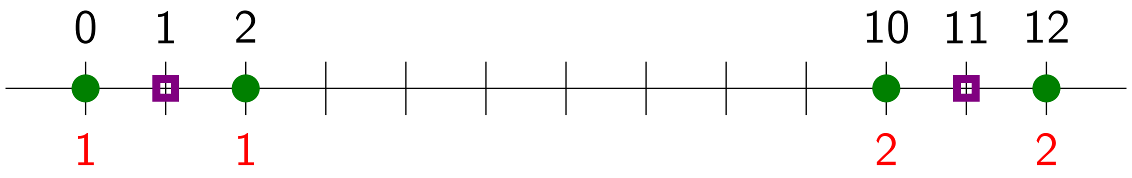
K-means objective



$$\text{Loss}_{\text{kmeans}}(\mathbf{z}, \boldsymbol{\mu}) = \sum_{i=1}^{n} \|\phi(x_i) - \mu_{z_i}\|^2$$

$$\min_{\mathbf{z}} \min_{\boldsymbol{\mu}} \text{Loss}_{\text{kmeans}}(\mathbf{z}, \mu)$$

Alternating minimization from optimum



If know centroids $\mu_1 = 1$, $\mu_2 = 11$:

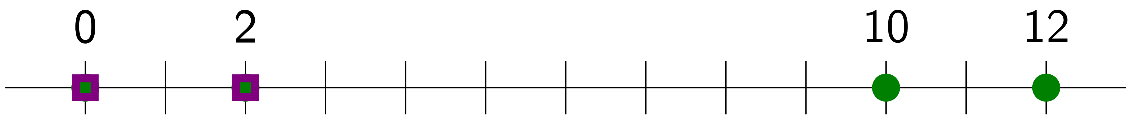
$$z_1 = \arg\min\{(0-1)^2, (0-11)^2\} = 1$$
$$z_2 = \arg\min\{(2-1)^2, (2-11)^2\} = 1$$
$$z_3 = \arg\min\{(10-1)^2, (10-11)^2\} = 2$$
$$z_4 = \arg\min\{(12-1)^2, (12-11)^2\} = 2$$

If know assignments $z_1 = z_2 = 1$, $z_3 = z_4 = 2$:

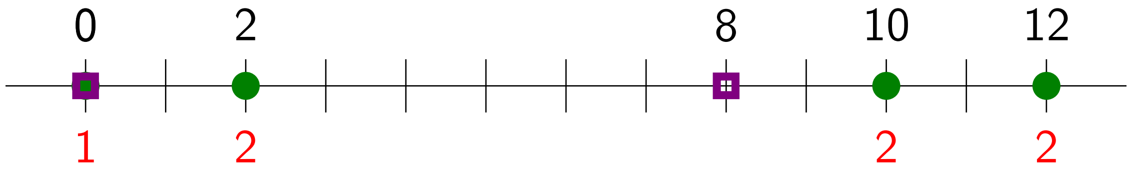
$$\mu_1 = \arg\min_\mu (0-\mu)^2 + (2-\mu)^2 = 1$$
$$\mu_2 = \arg\min_\mu (10-\mu)^2 + (12-\mu)^2 = 11$$

Alternating minimization from random initialization

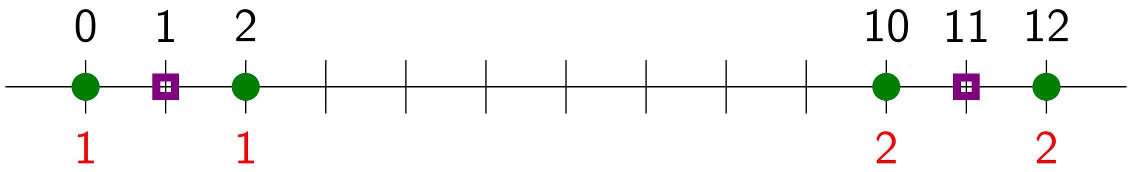
Initialize $\boldsymbol{\mu}$:



Iteration 1:



Iteration 2:



Converged.

K-means algorithm

Algorithm: K-means

Initialize $\boldsymbol{\mu} = [\mu_1, \dots, \mu_K]$ randomly.

For $t = 1, \dots, T$:

Step 1: set assignments $\mathbf{z}$ given $\boldsymbol{\mu}$

For each point $i = 1, \dots, n$:

$$z_i \leftarrow \arg \min_{k=1,\dots,K} \|\phi(x_i) - \mu_k\|^2$$

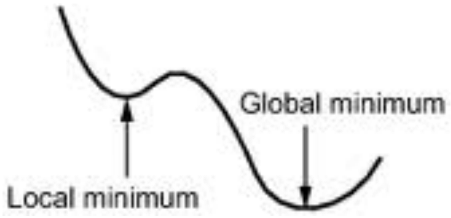
Step 2: set centroids $\boldsymbol{\mu}$ given $\mathbf{z}$

For each cluster $k = 1, \dots, K$:

$$\mu_k \leftarrow \frac{1}{|\{i : z_i = k\}|} \sum_{i : z_i = k} \phi(x_i)$$

Local minima

K-means is guaranteed to converge to a local minimum, but is not guaranteed to find the global minimum.



[demo: getting stuck in local optima, seed = 100]

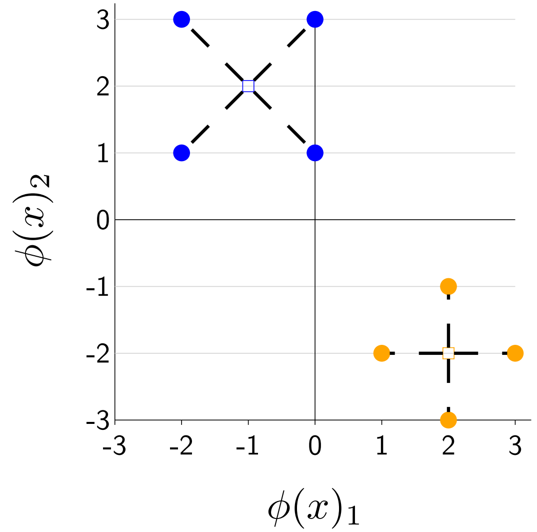
Solutions:

- Run multiple times from different random initializations
- Initialize with a heuristic (K-means++)

Summary

Clustering: discover structure in unlabeled data

K-means objective:



K-means algorithm:

assignments $\mathbf{z}$  centroids $\boldsymbol{\mu}$

Unsupervised learning use cases:

- Data exploration and discovery
- Providing representations to downstream supervised learning

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Cat

Classification

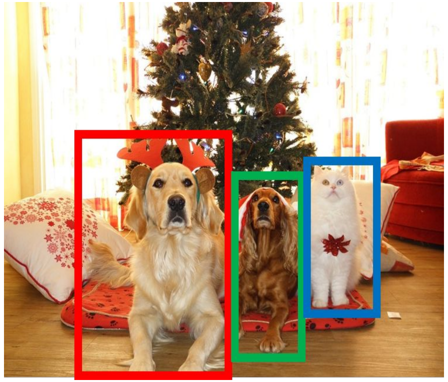
Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.



DOG, DOG, CAT

Object Detection

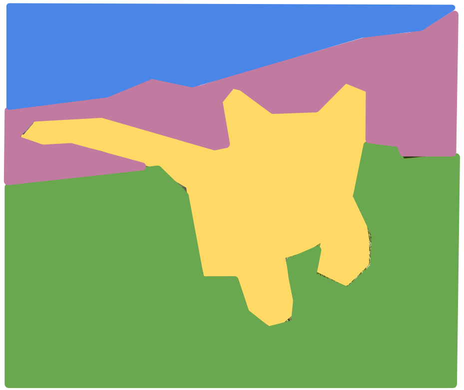
Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.



GRASS, CAT, TREE, SKY

Semantic Segmentation

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

A cat sitting on a suitcase on the floor

Image captioning

Supervised vs Unsupervised Learning

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

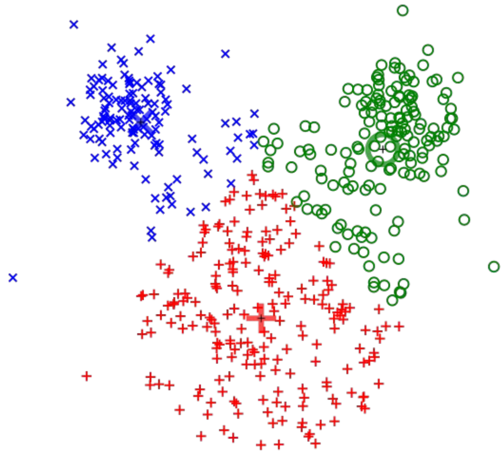
Supervised vs Unsupervised Learning

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



K-means clustering

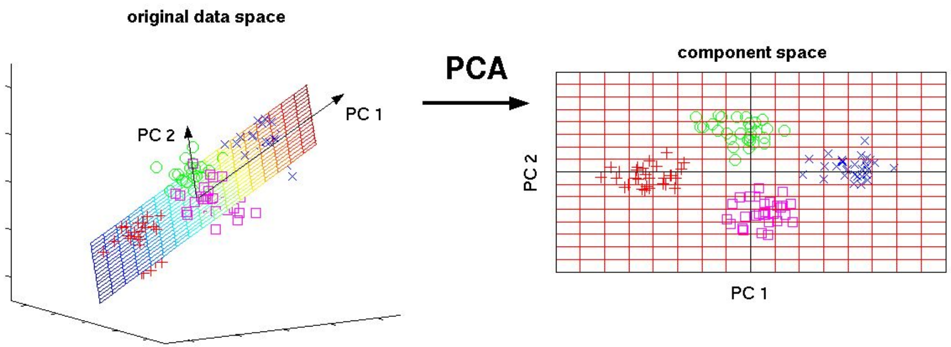
Supervised vs Unsupervised Learning

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



3-d → 2-d

Principal Component Analysis
(Dimensionality reduction)

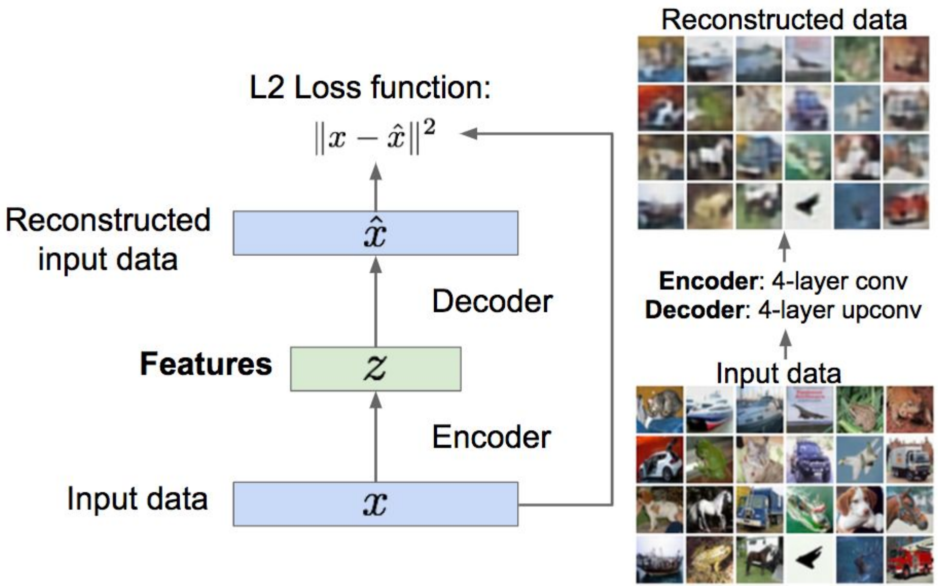
Supervised vs Unsupervised Learning

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



Autoencoders
(Feature learning)

Supervised vs Unsupervised Learning

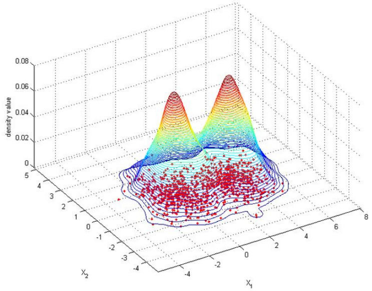
Unsupervised Learning

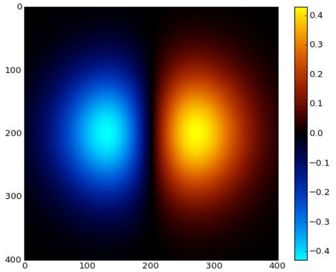
Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

1-d density estimation





2-d density estimation

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Unsupervised Learning

Training data is cheap

Data: x
Just data, no labels!

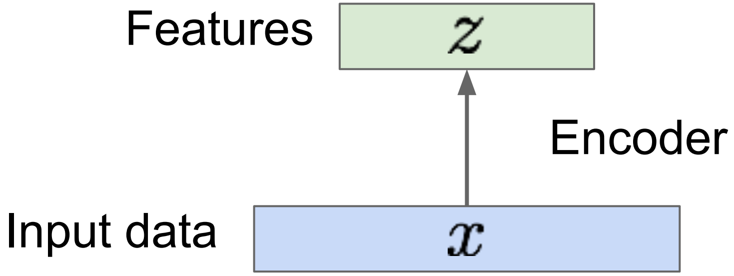
Holy grail: Solve unsupervised learning => understand structure of visual world

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Some background first: Autoencoders

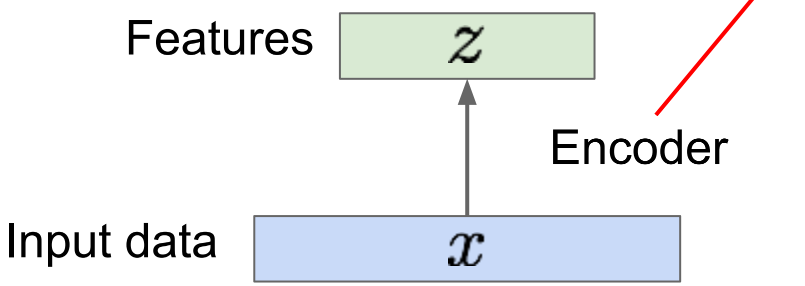
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data



Some background first: Autoencoders

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN



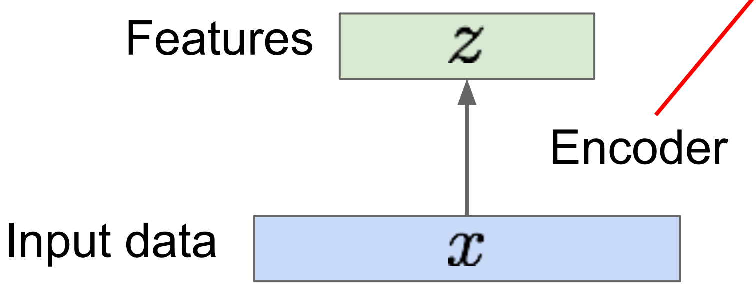
Some background first: Autoencoders

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

z usually smaller than **x**
(dimensionality reduction)

Q: Why dimensionality reduction?

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN



Some background first: Autoencoders

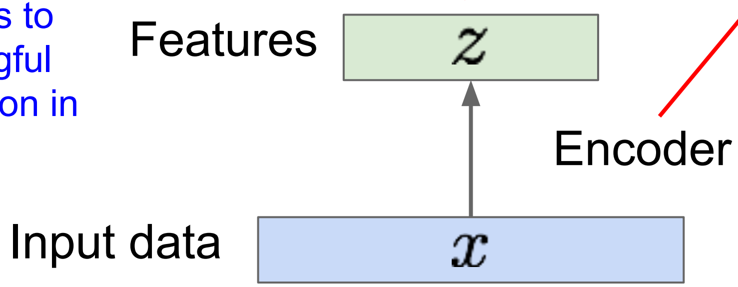
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

z usually smaller than **x** (dimensionality reduction)

Q: Why dimensionality reduction?

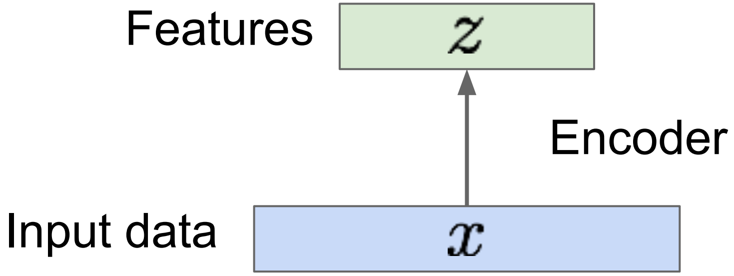
A: Want features to capture meaningful factors of variation in data

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN



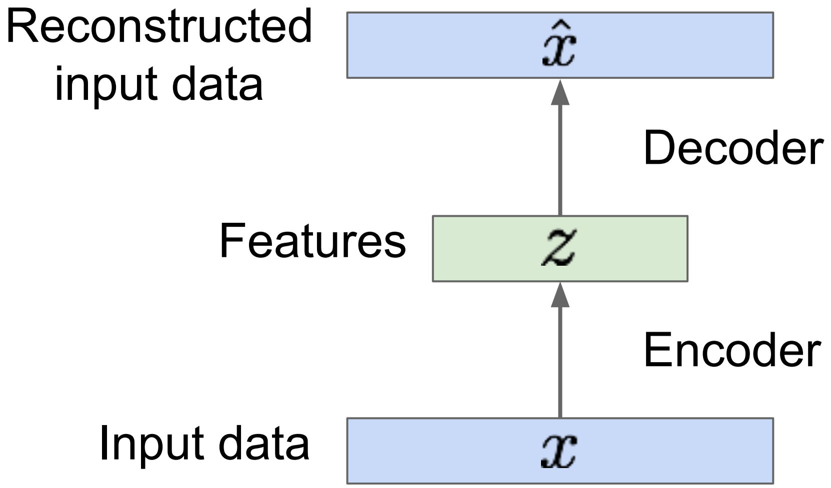
Some background first: Autoencoders

How to learn this feature representation?



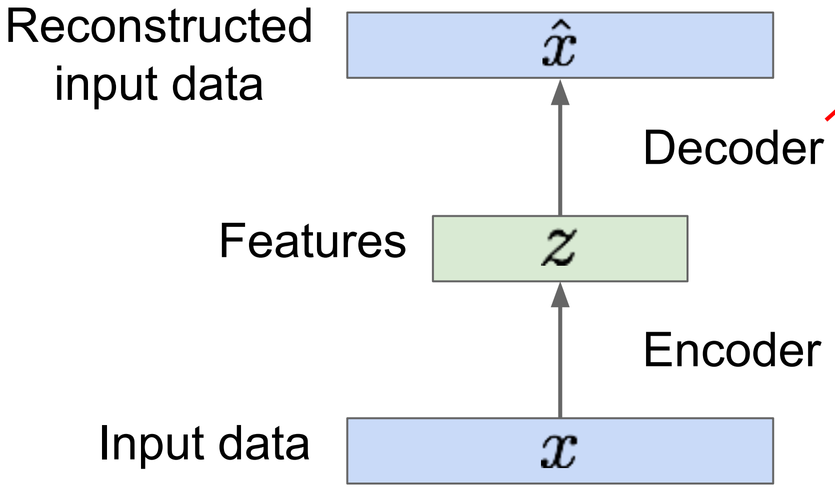
Some background first: Autoencoders

How to learn this feature representation?
Train such that features can be used to reconstruct original data
“Autoencoding” - encoding itself



Some background first: Autoencoders

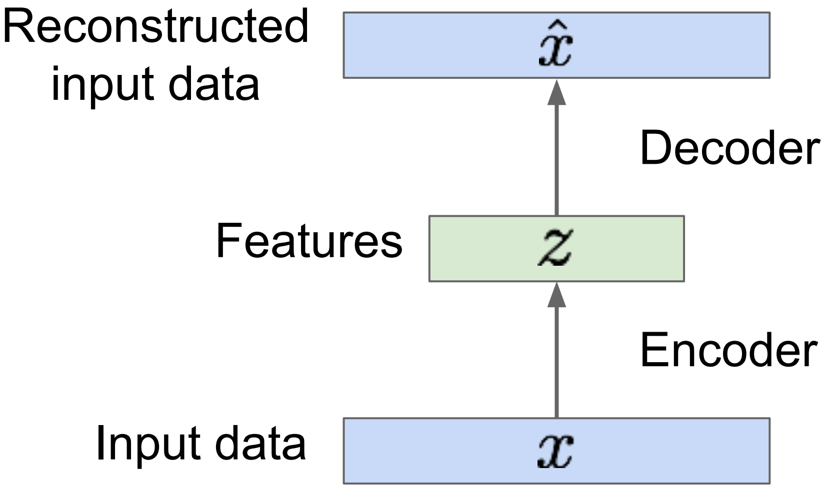
How to learn this feature representation?
Train such that features can be used to reconstruct original data
“Autoencoding” - encoding itself



Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN (upconv)

Some background first: Autoencoders

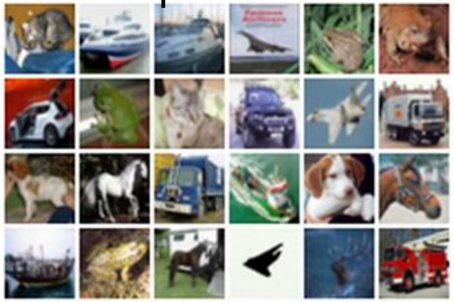
How to learn this feature representation?
Train such that features can be used to reconstruct original data
"Autoencoding" - encoding itself



Reconstructed data

Encoder: 4-layer conv
Decoder: 4-layer upconv

Input data



Some background first: Autoencoders

Train such that features can be used to reconstruct original data

L2 Loss function: