CS 6501: Deep Learning for Computer Graphics

Basics of Machine Learning

Connelly Barnes

Overview

- **Supervised, unsupervised, and reinforcement learning**
- Simple learning models
  - Clustering
  - Linear regression
  - Linear Support Vector Machines (SVM)
  - k-Nearest Neighbors
- Overfitting and generalization
- Training, testing, validation
- Balanced datasets
- Measuring performance of a classifier

Supervised, Unsupervised, Reinforcement

- 3 broad categories:
  - **Supervised learning**: computer presented with example inputs and desired outputs by a “teacher”, goal is to learn general rule that maps inputs to outputs.
  - **Unsupervised learning**: No output labels are given to the algorithm, leaving it on its own to find structure in the inputs.
  - **Reinforcement learning**: An agent determines what actions to best take in an environment to maximize some notion of cumulative reward.

What Kind of Learning is This?

- Learn given input image, whether it is truck or car? Training data:

Label() = Truck

Label() = Car

Label() = Truck

Label() = Car

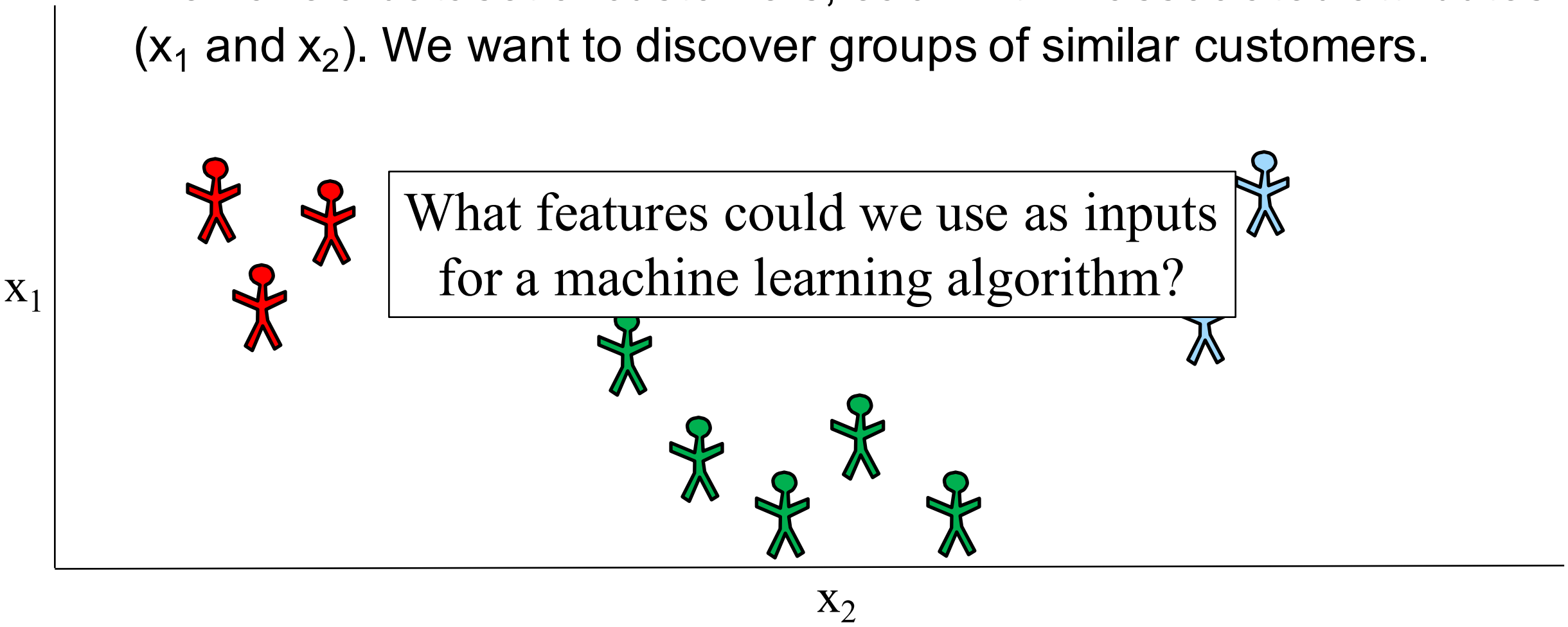
Label() = Truck

Label() = Car

Images are Creative Commons, sources: [1], [2], [3], [4], [5], [6]

What Kind of Learning is This?

- We have a dataset of customers, each with 2 associated attributes (x_1 and x_2). We want to discover groups of similar customers.



What Kind of Learning is This?

[Peng et al., Terrain-Adaptive Locomotion…, SIGGRAPH 2016]

Overview

- Supervised, unsupervised, and reinforcement learning
- Simple learning models
  - **Clustering**
  - Linear regression
  - Linear Support Vector Machines (SVM)
  - k-Nearest Neighbors
- Overfitting and generalization
- Training, testing, validation
- Balanced datasets
- Measuring performance of a classifier

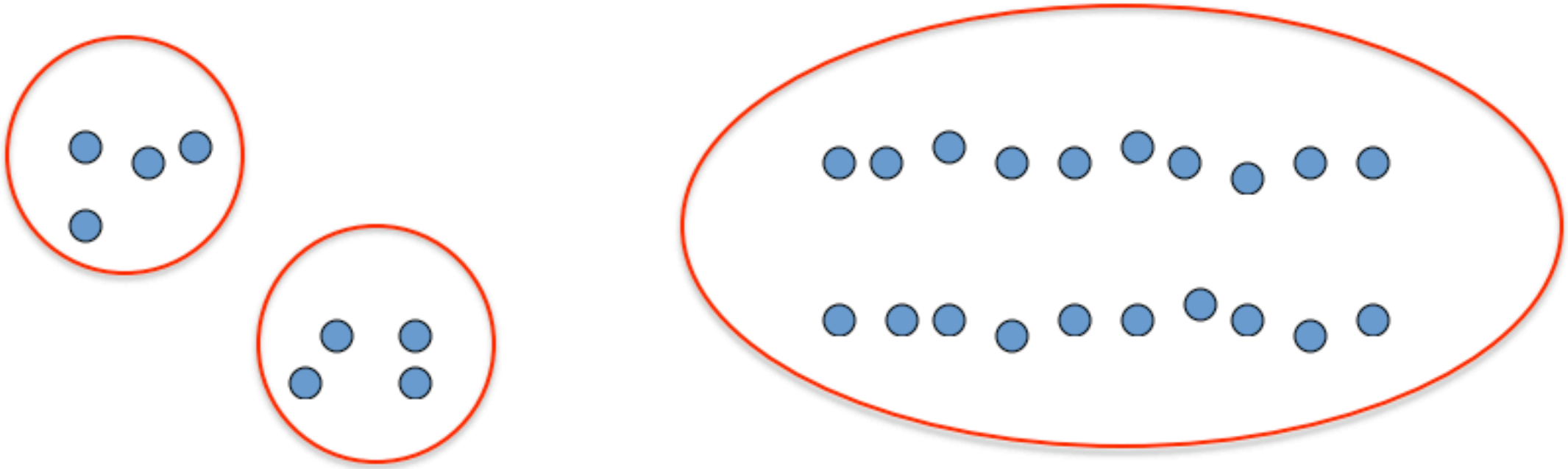
Clustering

- Unsupervised learning
- Requires input data, but no labels
- Detects patterns, e.g.
  - Groups of similar emails, similar web-pages in search results
  - Similar customer shopping patterns
  - Regions of images

Slides adapted from David Sontag, Luke Zettlemoyer, Vibhav, Gogate, Carlos Guestrin, Andrew Moore, Dan Klein

Clustering

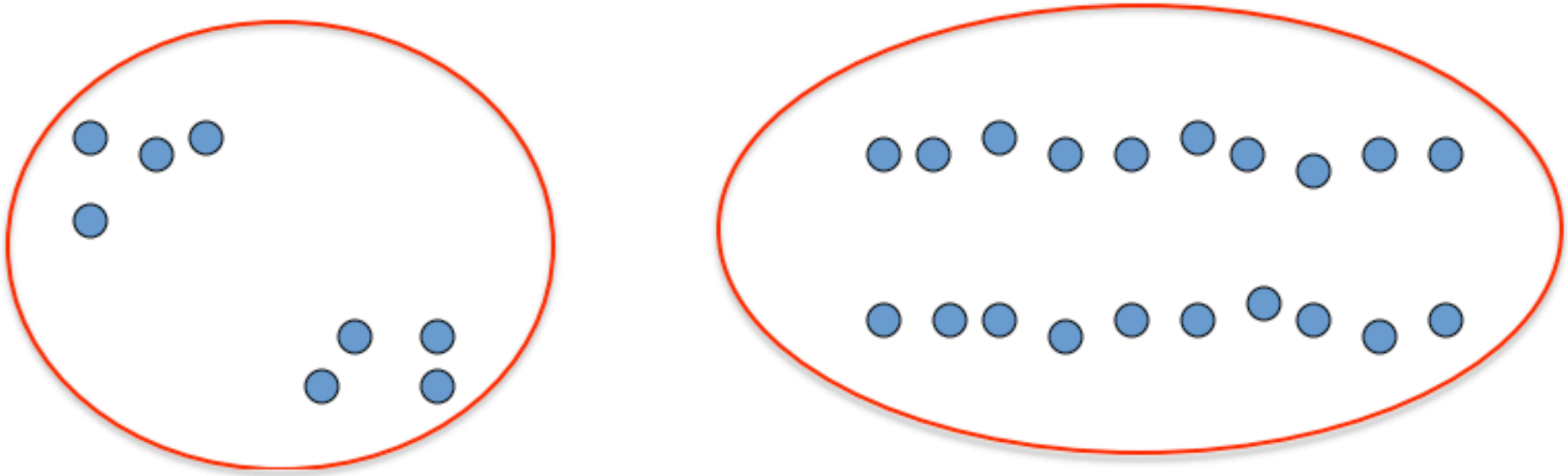
- **Idea**: group together similar instances
- **Example**: 2D point patterns



Slides adapted from David Sontag, Luke Zettlemoyer, Vibhav, Gogate, Carlos Guestrin, Andrew Moore, Dan Klein

Clustering

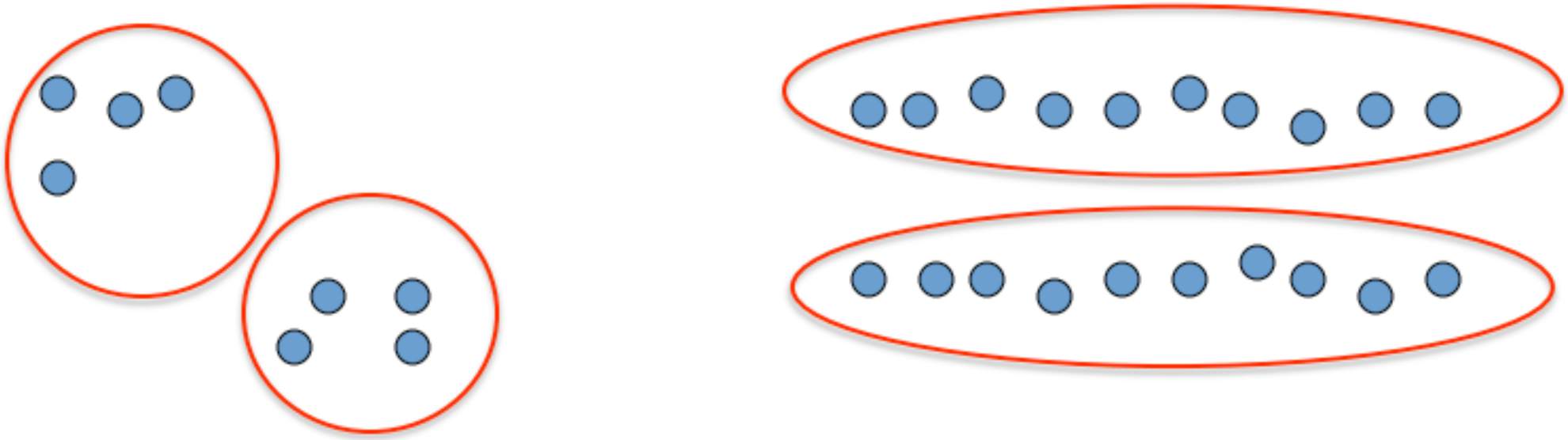
- **Idea**: group together similar instances
- **Example**: 2D point patterns



Slides adapted from David Sontag, Luke Zettlemoyer, Vibhav, Gogate, Carlos Guestrin, Andrew Moore, Dan Klein

Clustering

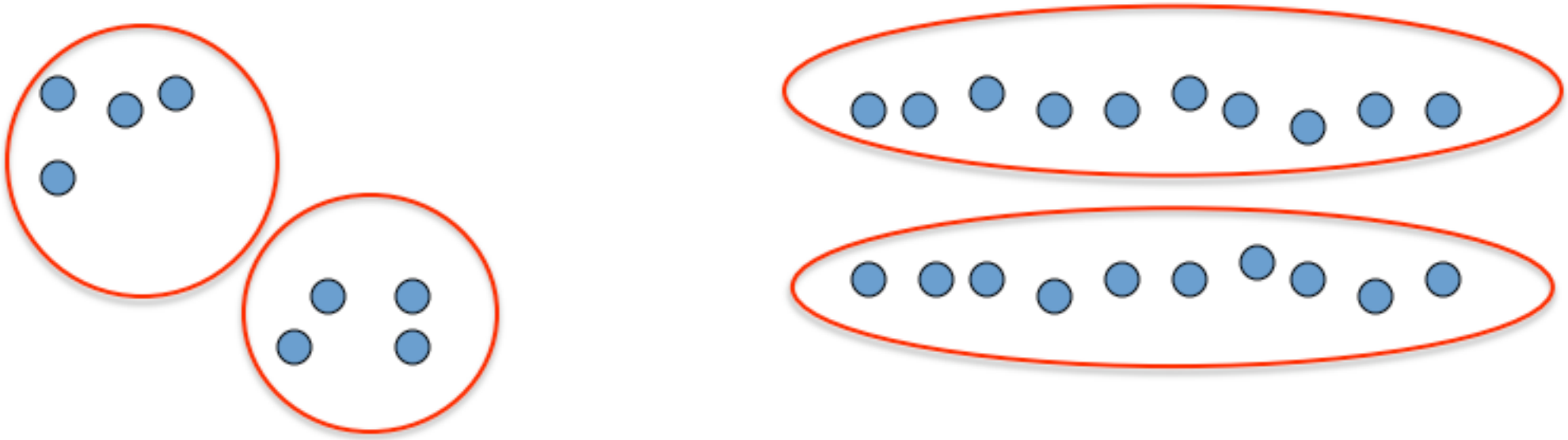
- **Idea**: group together similar instances
- **Example**: 2D point patterns



Slides adapted from David Sontag, Luke Zettlemoyer, Vibhav, Gogate, Carlos Guestrin, Andrew Moore, Dan Klein

Clustering

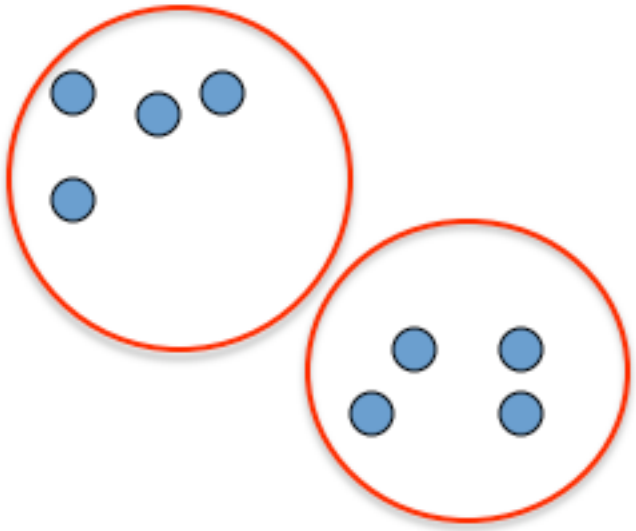
- **Idea**: group together similar instances
- **Problem**: How to define “similar”?
- **Problem**: How many clusters?

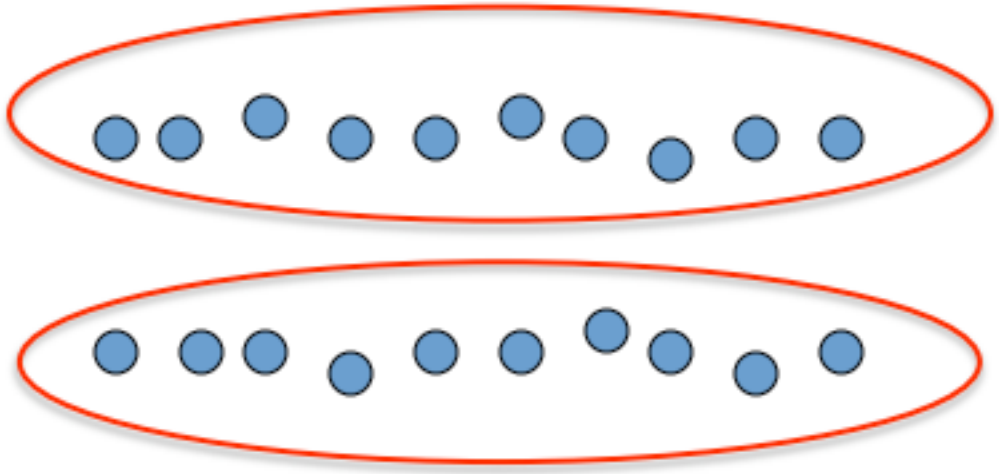


Slides adapted from David Sontag, Luke Zettlemoyer, Vibhav, Gogate, Carlos Guestrin, Andrew Moore, Dan Klein

Clustering

- **Similarity:** in Euclidean space $\mathbf{R}^n$, could be a distance function.
- For example: $D(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^2$
- Clustering results will depend on measure of similarity / dissimilarity

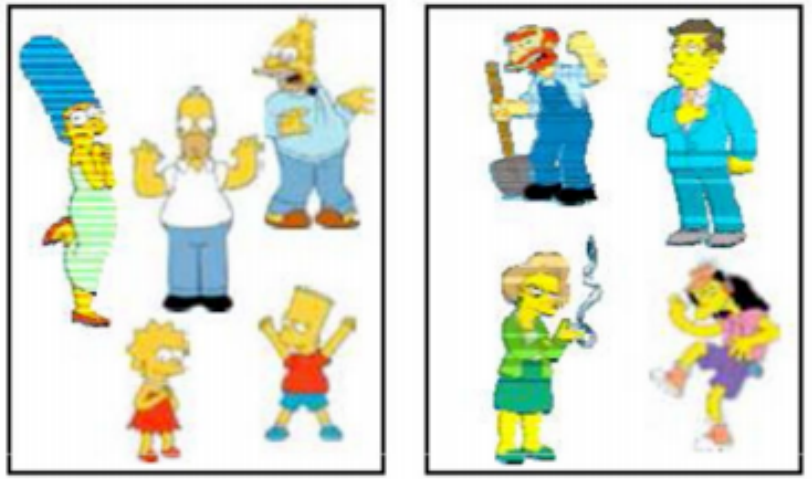




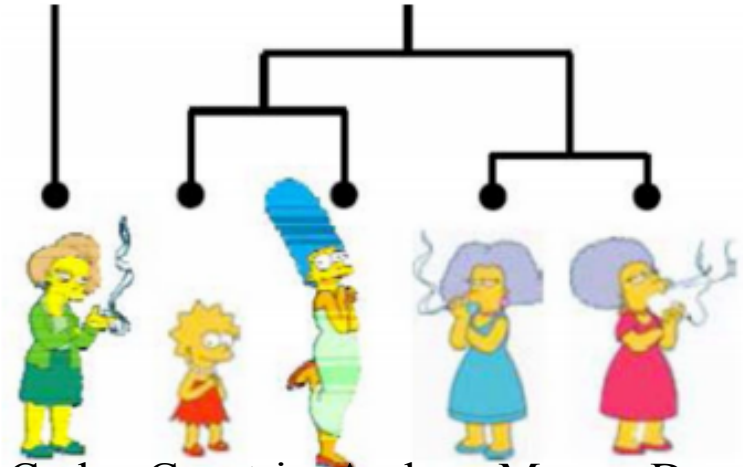
Slides adapted from David Sontag, Luke Zettlemoyer, Vibhav, Gogate, Carlos Guestrin, Andrew Moore, Dan Klein

Clustering Algorithms

- Partitioning algorithms (flat)
  - K-means



- Hierarchical algorithms
  - Bottom-up: agglomerative
  - Top-down: divisive



Slides adapted from David Sontag, Luke Zettlemoyer, Vibhav, Gogate, Carlos Guestrin, Andrew Moore, Dan Klein

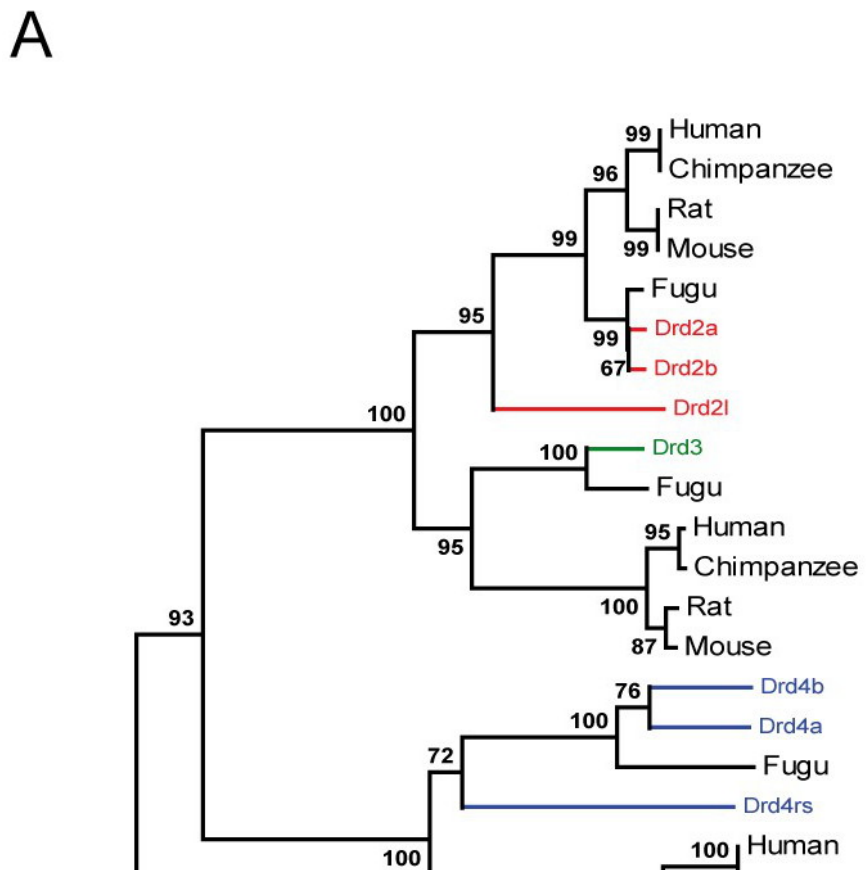
Clustering Examples: Image Segmentation

- Divide an image into regions that are perceptually similar to humans.

Slides adapted from James Hays, David Sontag, Luke Zettlemoyer, Vibhav, Gogate, Carlos Guestrin, Andrew Moore, Dan Klein

Clustering Examples: Biology

- Cluster species based on e.g. genetic or phenotype similarity.

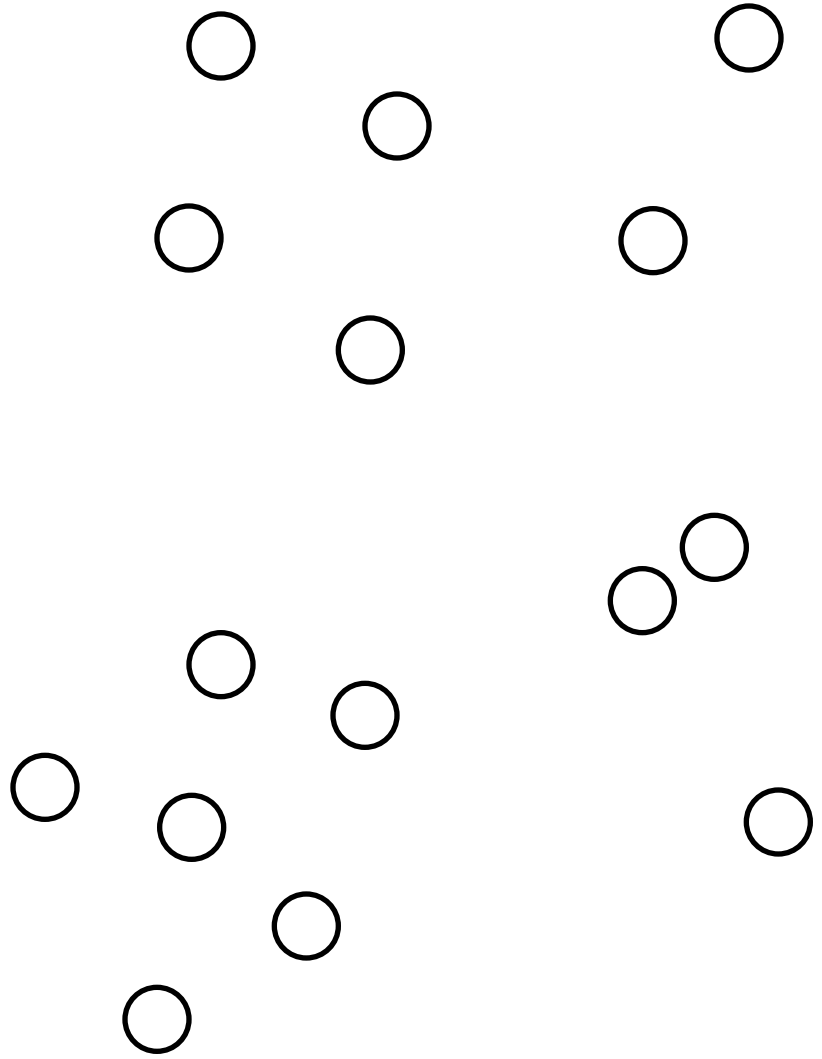


Slides adapted from David Sontag, Luke Zettlemoyer, Vibhav, Gogate, Carlos Guestrin, Andrew Moore, Dan Klein. Image: [1]

Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

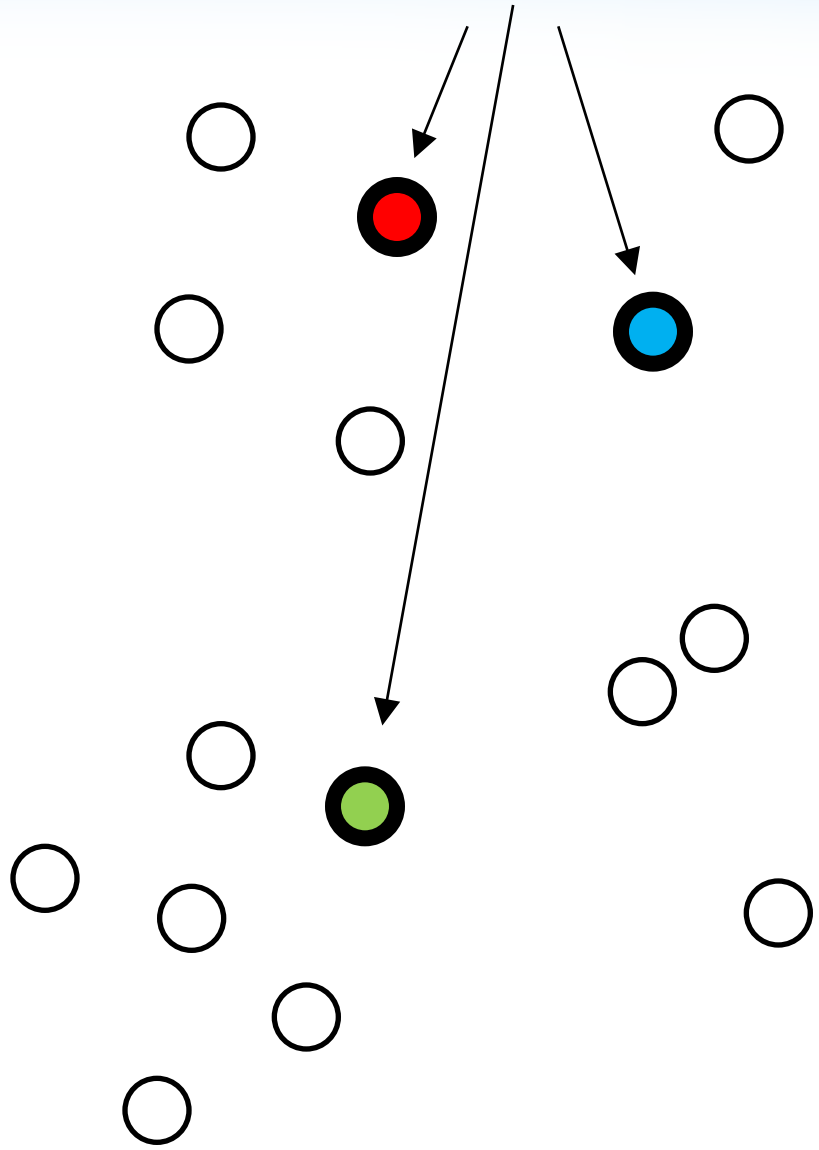
- **Initialize**: Pick *k* random points as cluster centers.
- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.



Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

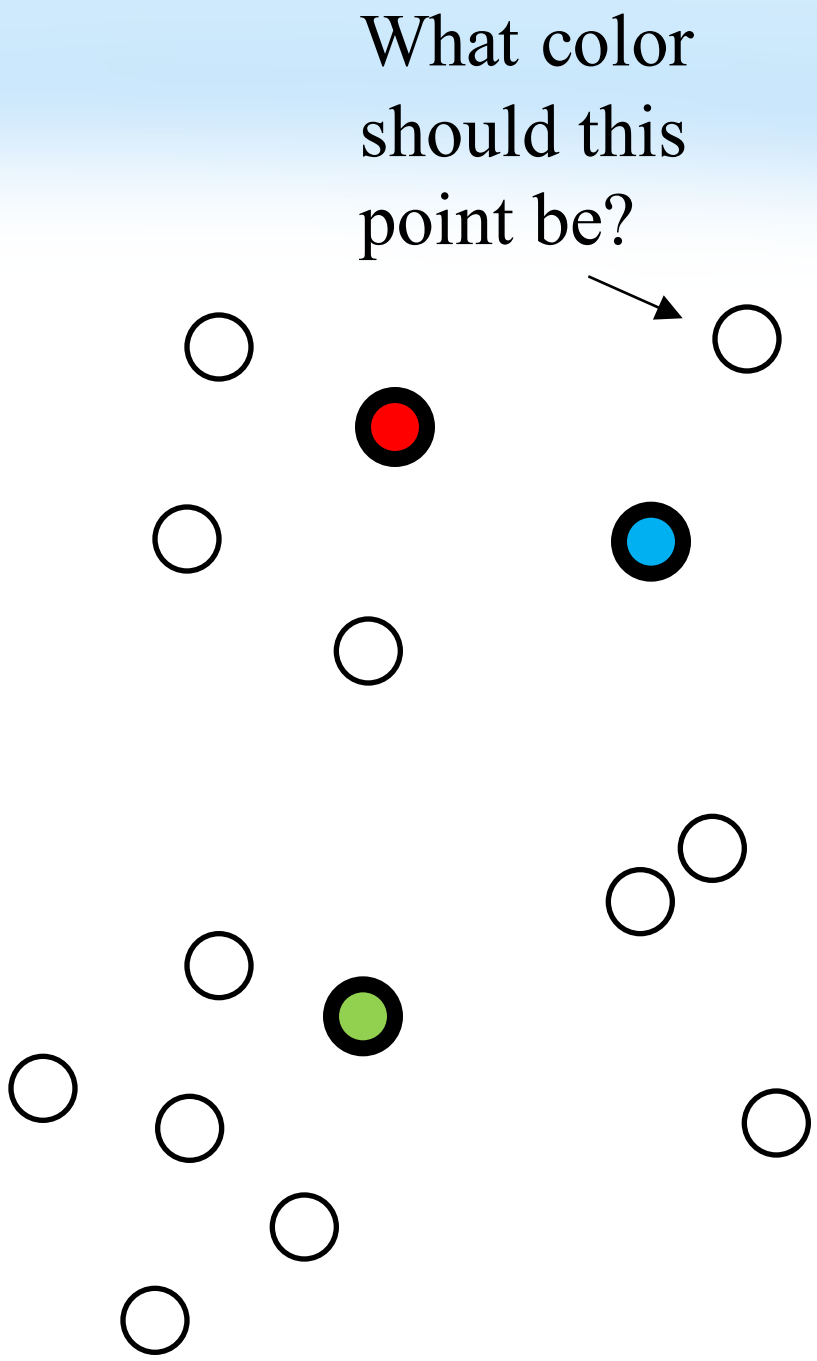
- **Initialize**: Pick *k* random points as cluster centers.
- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.



Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

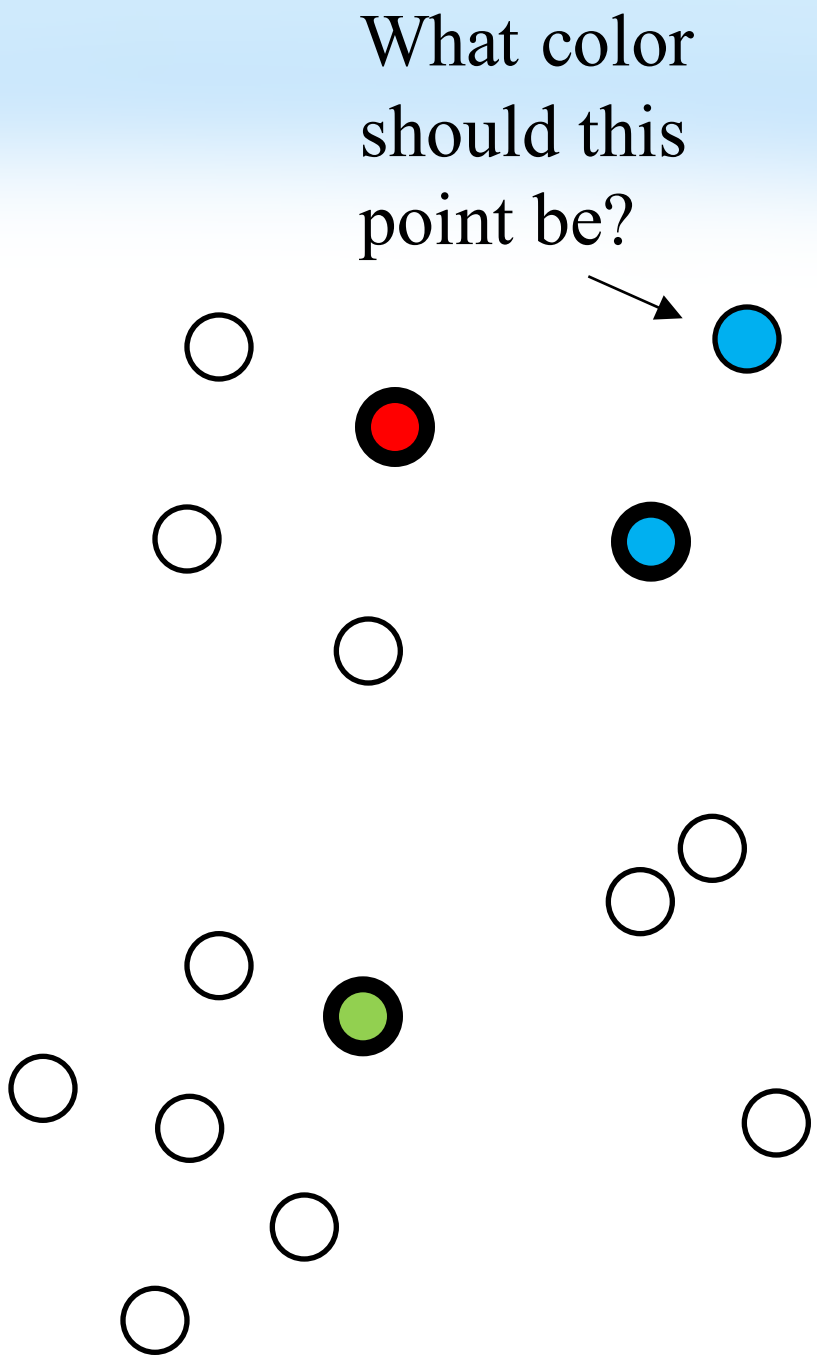
- **Initialize**: Pick *k* random points as cluster centers.
- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.



Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

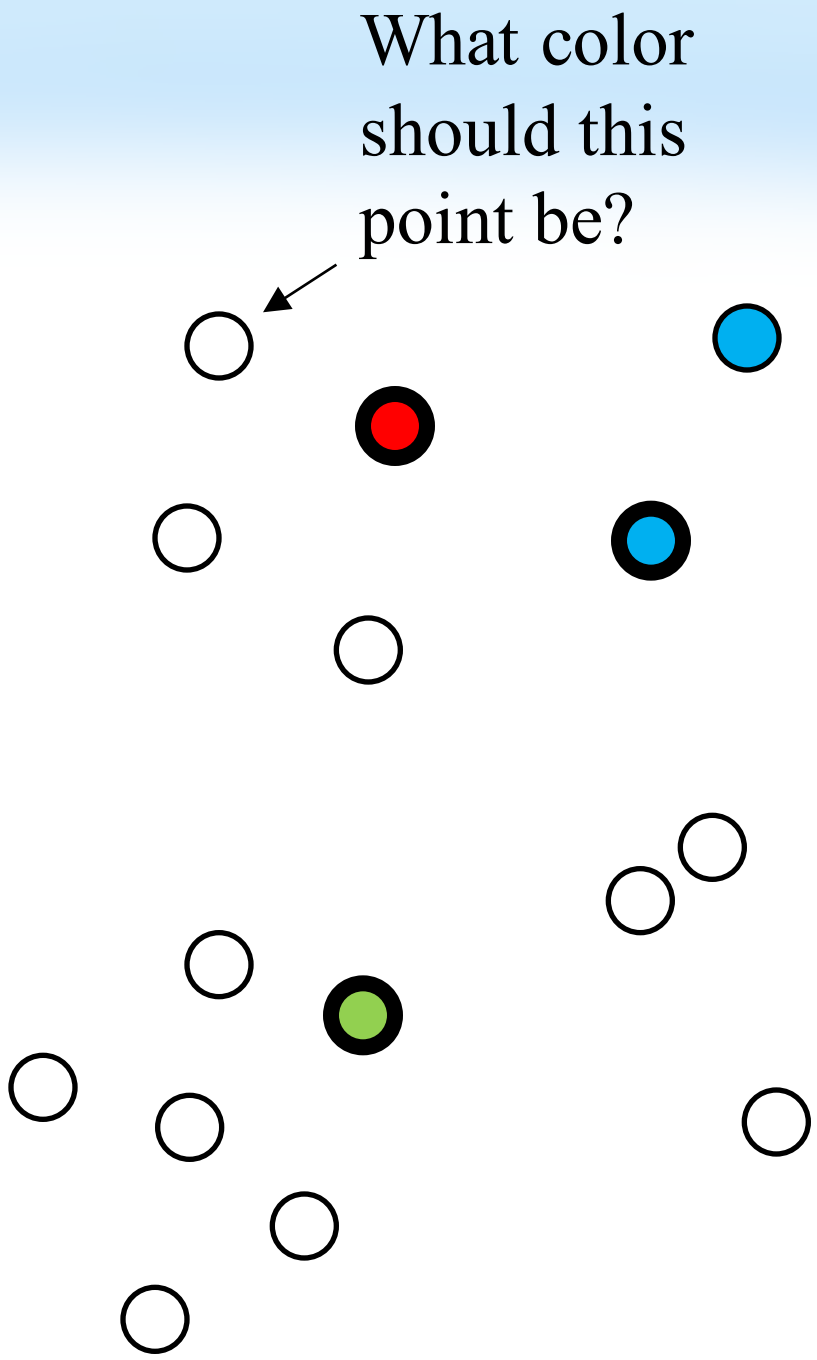
- **Initialize**: Pick *k* random points as cluster centers.
- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.



Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

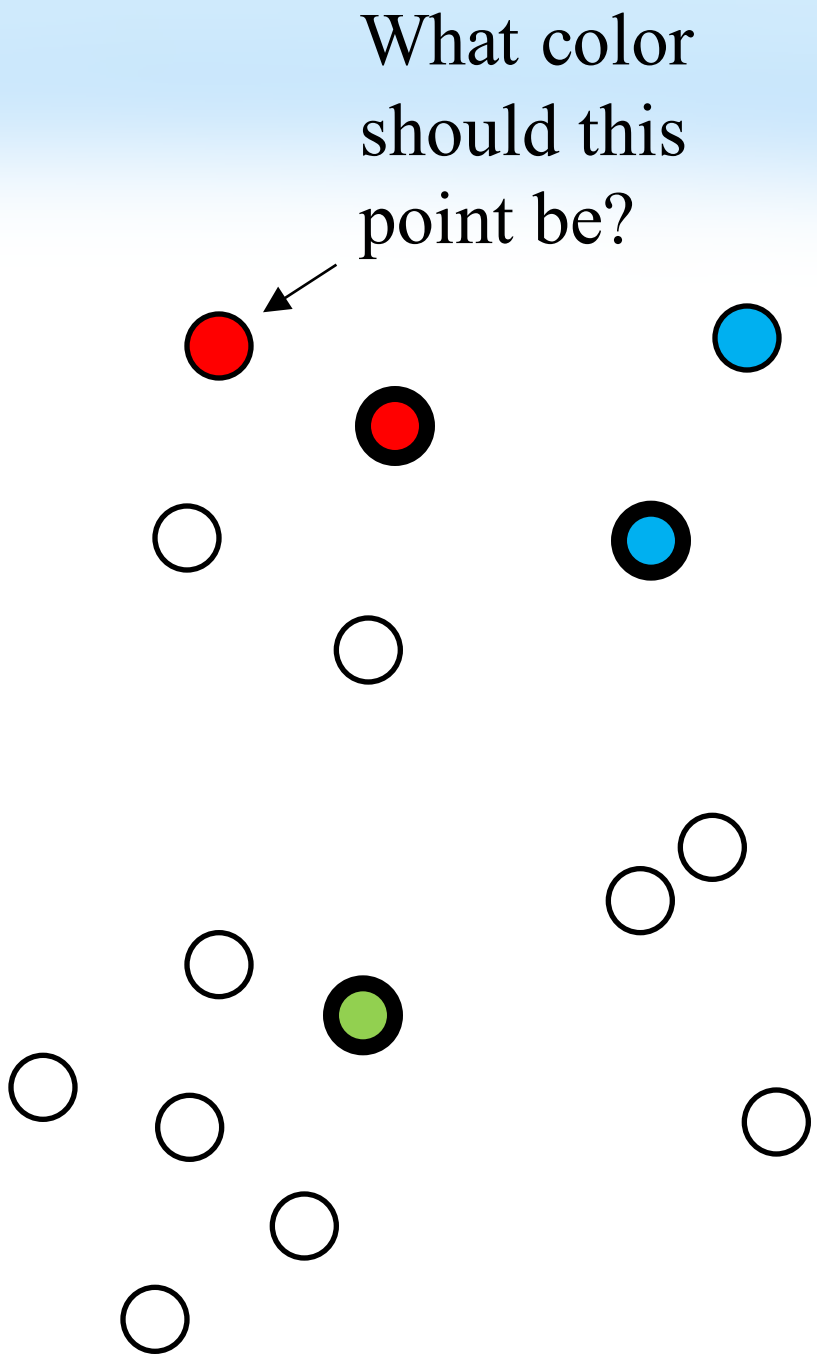
- **Initialize**: Pick *k* random points as cluster centers.
- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.



Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

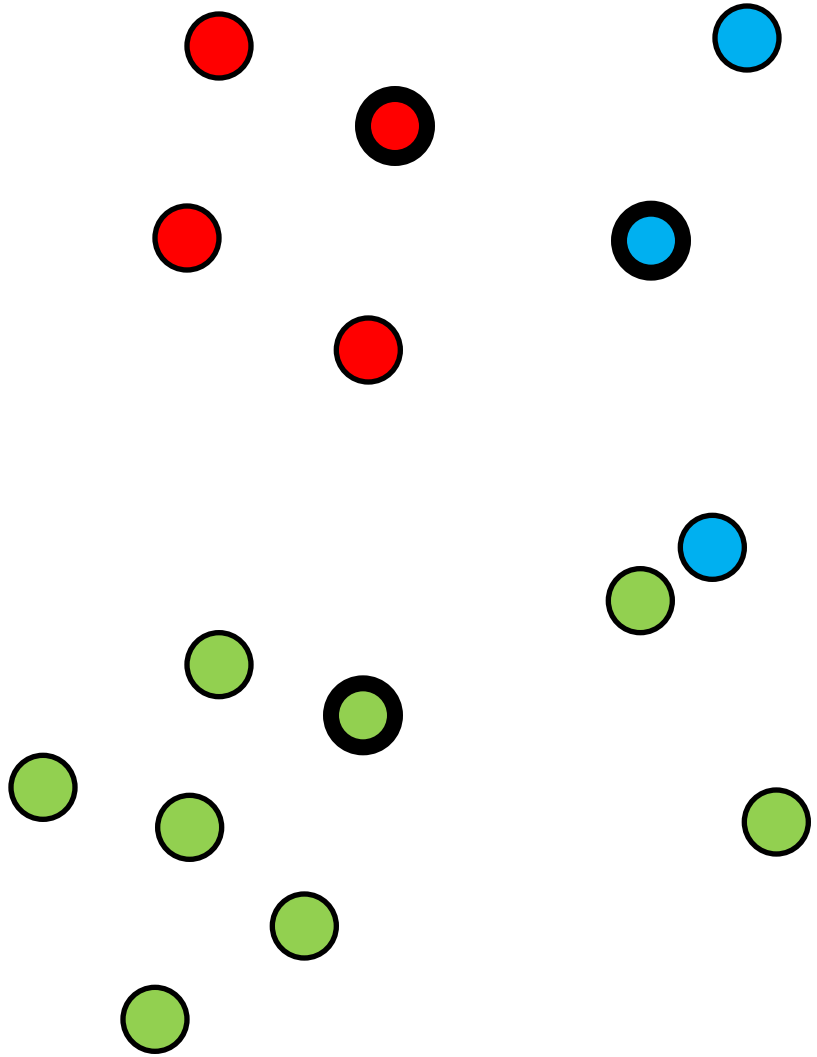
- **Initialize**: Pick *k* random points as cluster centers.
- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.



Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

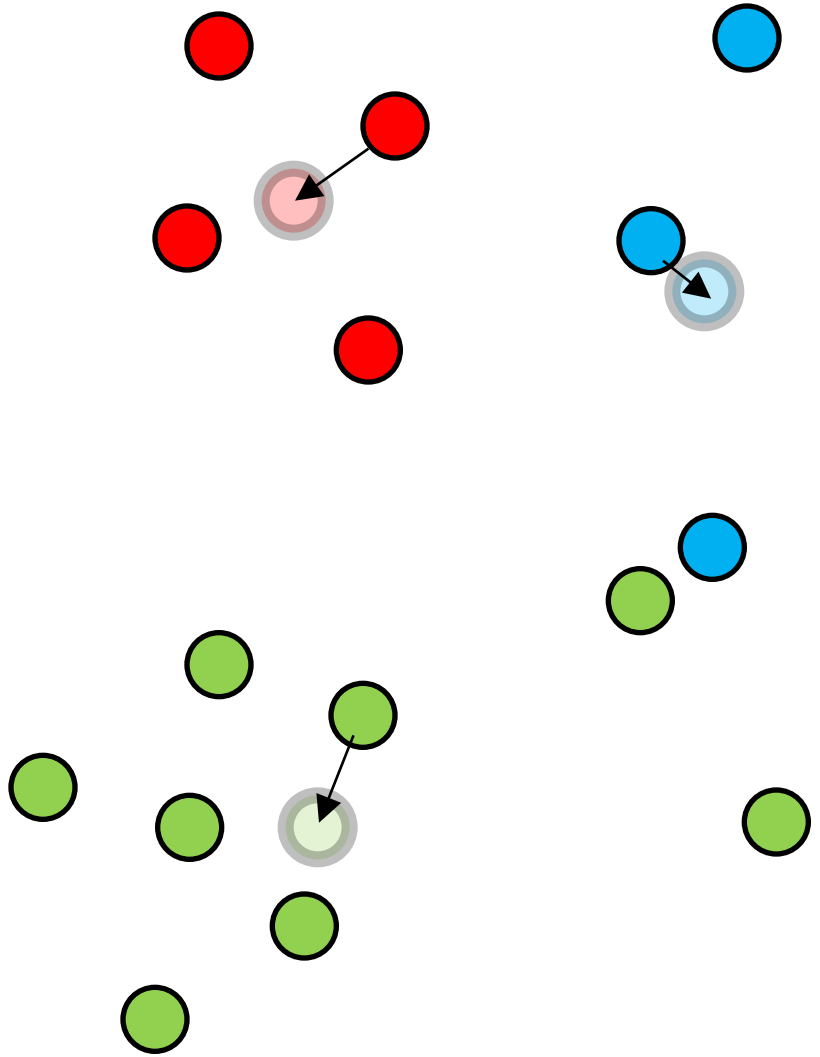
- **Initialize**: Pick *k* random points as cluster centers.
- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.



Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

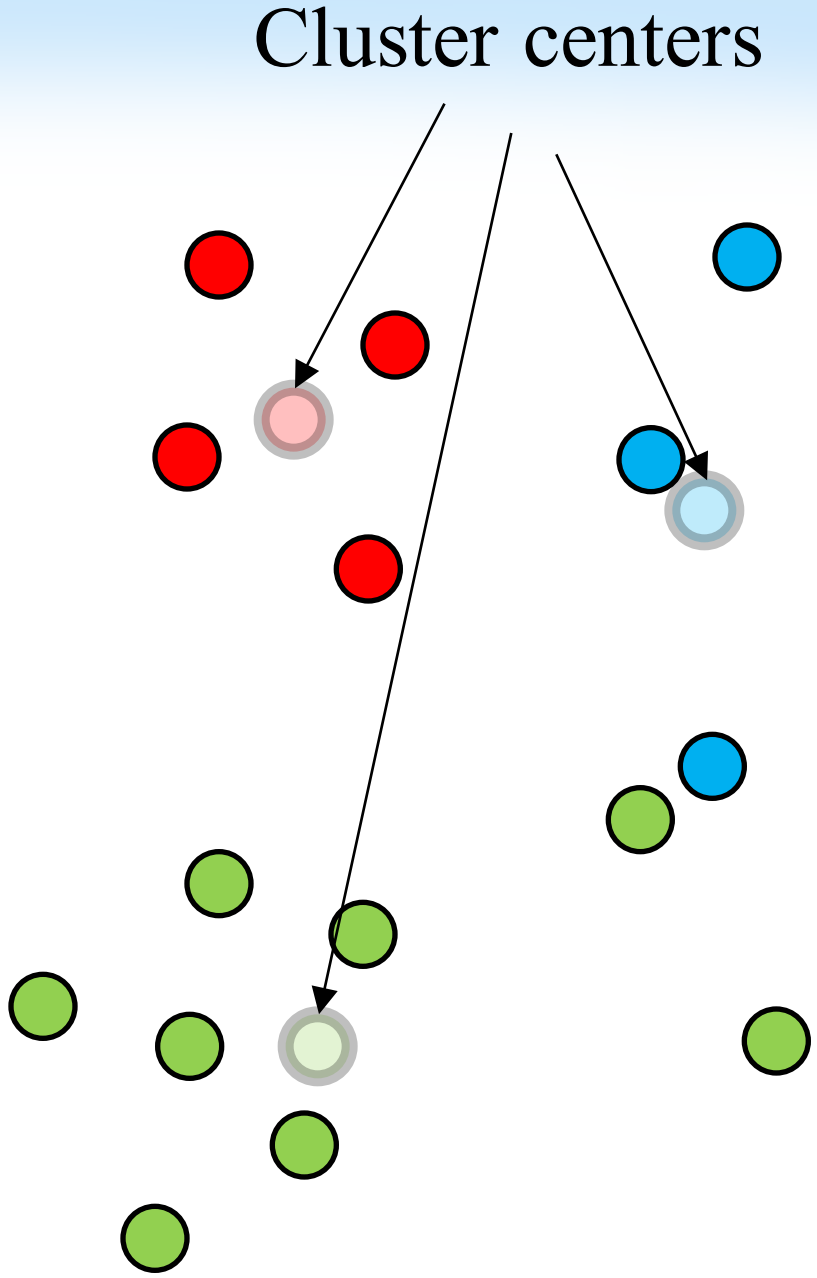
- **Initialize**: Pick *k* random points as cluster centers.
- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.



Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

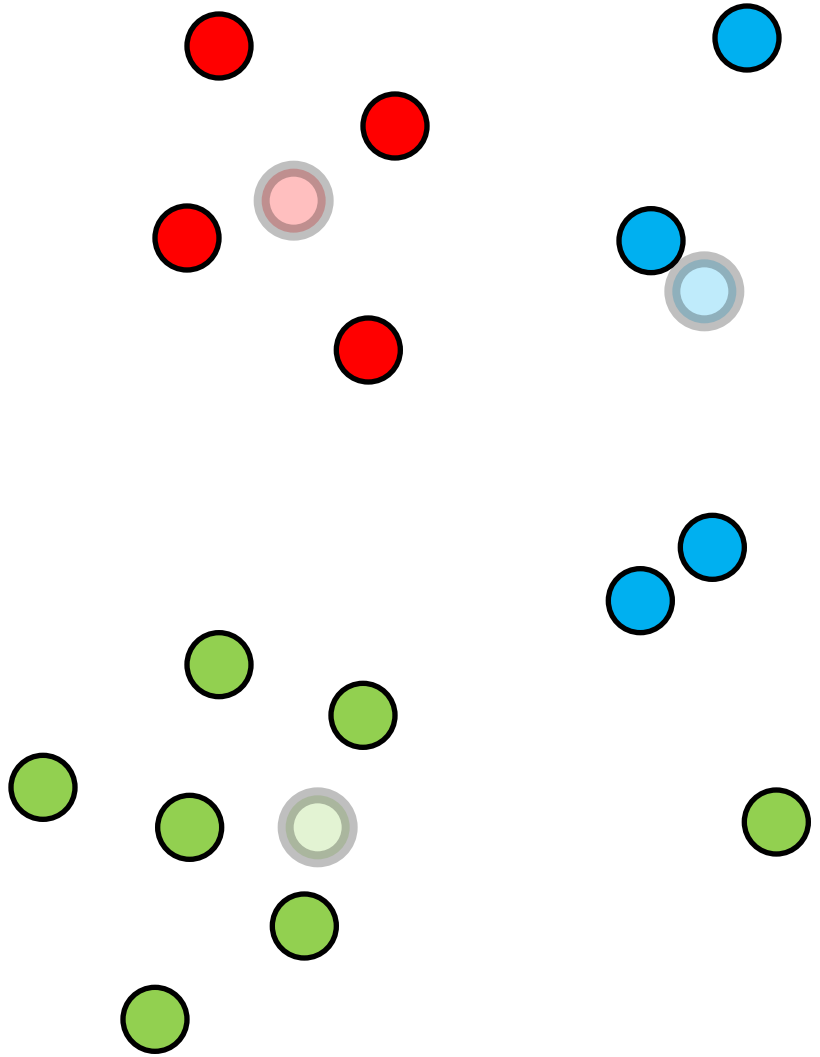
- **Initialize**: Pick *k* random
centers.
- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.



Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

- **Initialize**: Pick *k* random points as cluster centers.
- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.

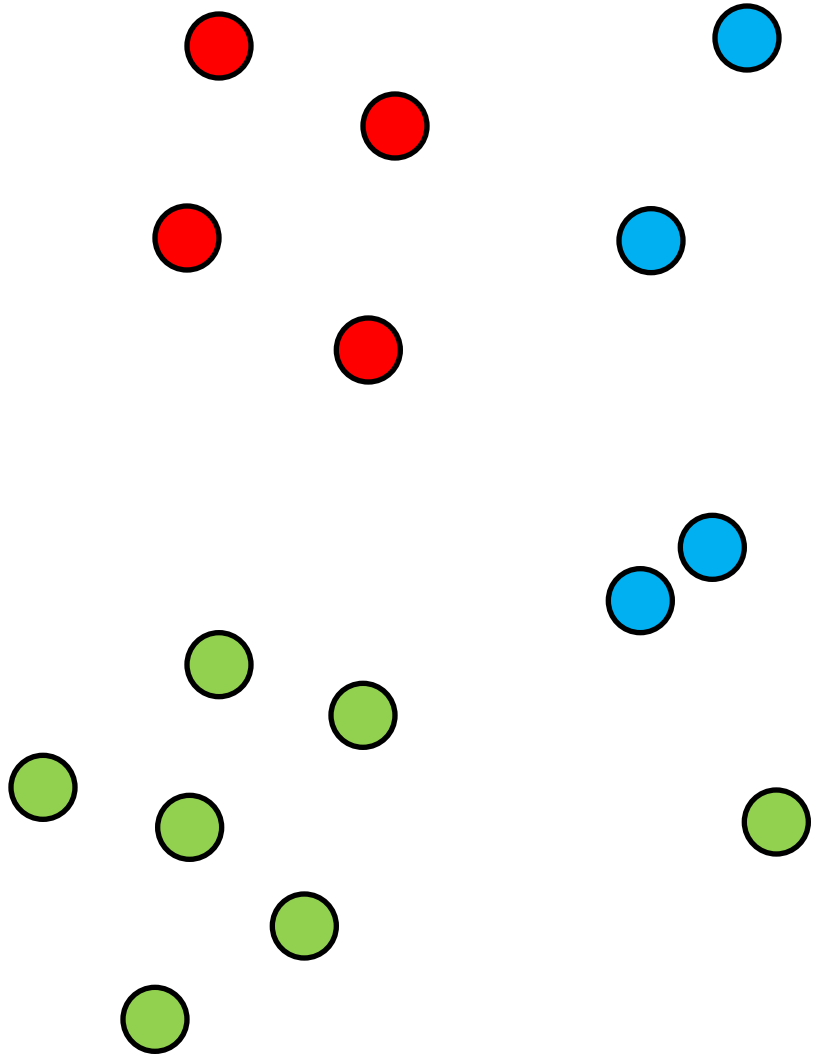


Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

- **Initialize**: Pick *k* random points as cluster centers.
- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.

Result of k-Means:



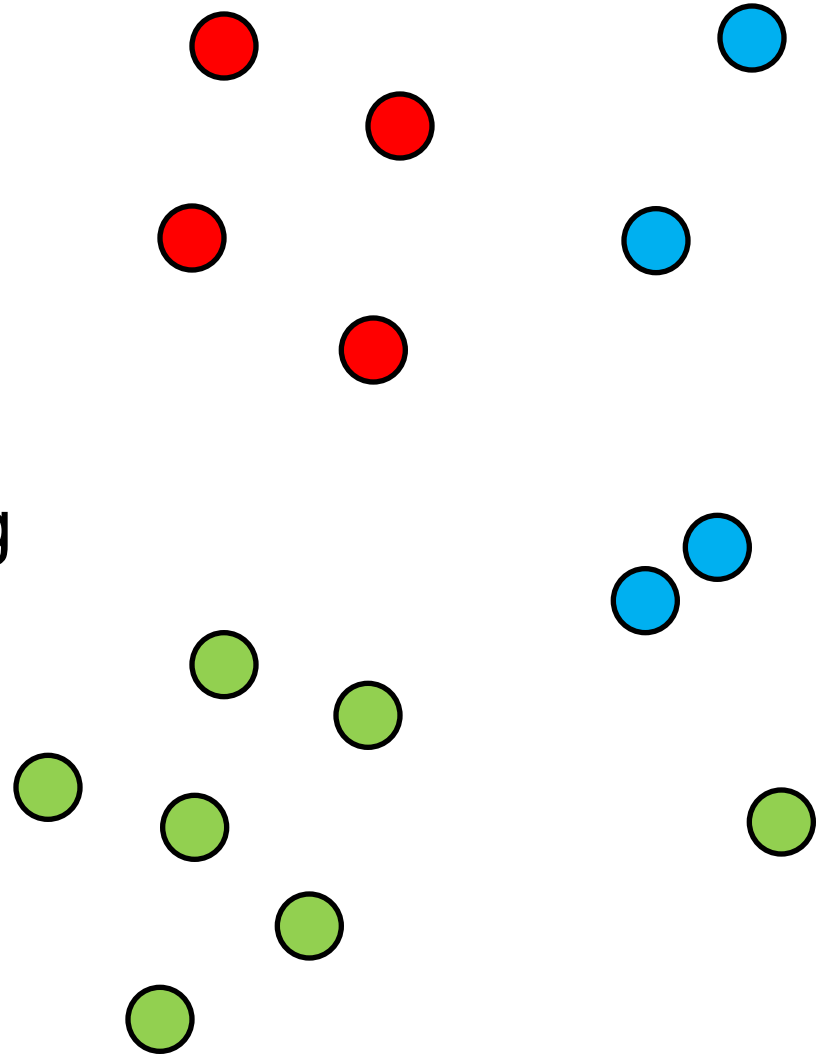
Clustering: k-Means

- Minimizes within-cluster sum of squares distance:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \qquad (1)$$

- Here $\boldsymbol{\mu}_i$ is the mean of the points belonging to cluster S_i.
- No guarantee algorithm will converge to global minimum.
- Can run several times and take best result according to (1).

Result of k-Means:

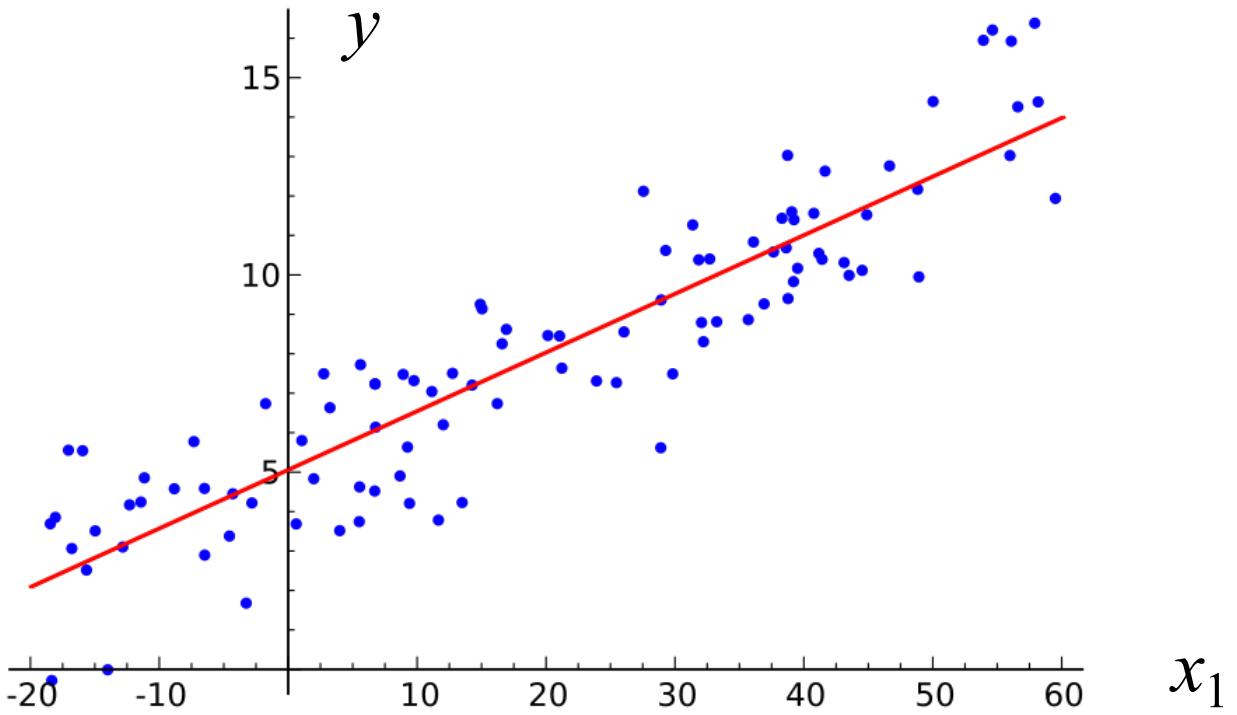


Overview

- Supervised, unsupervised, and reinforcement learning
- Simple learning models
  - Clustering
  - **Linear regression**
  - Linear Support Vector Machines (SVM)
  - k-Nearest Neighbors
- Overfitting and generalization
- Training, testing, validation
- Balanced datasets

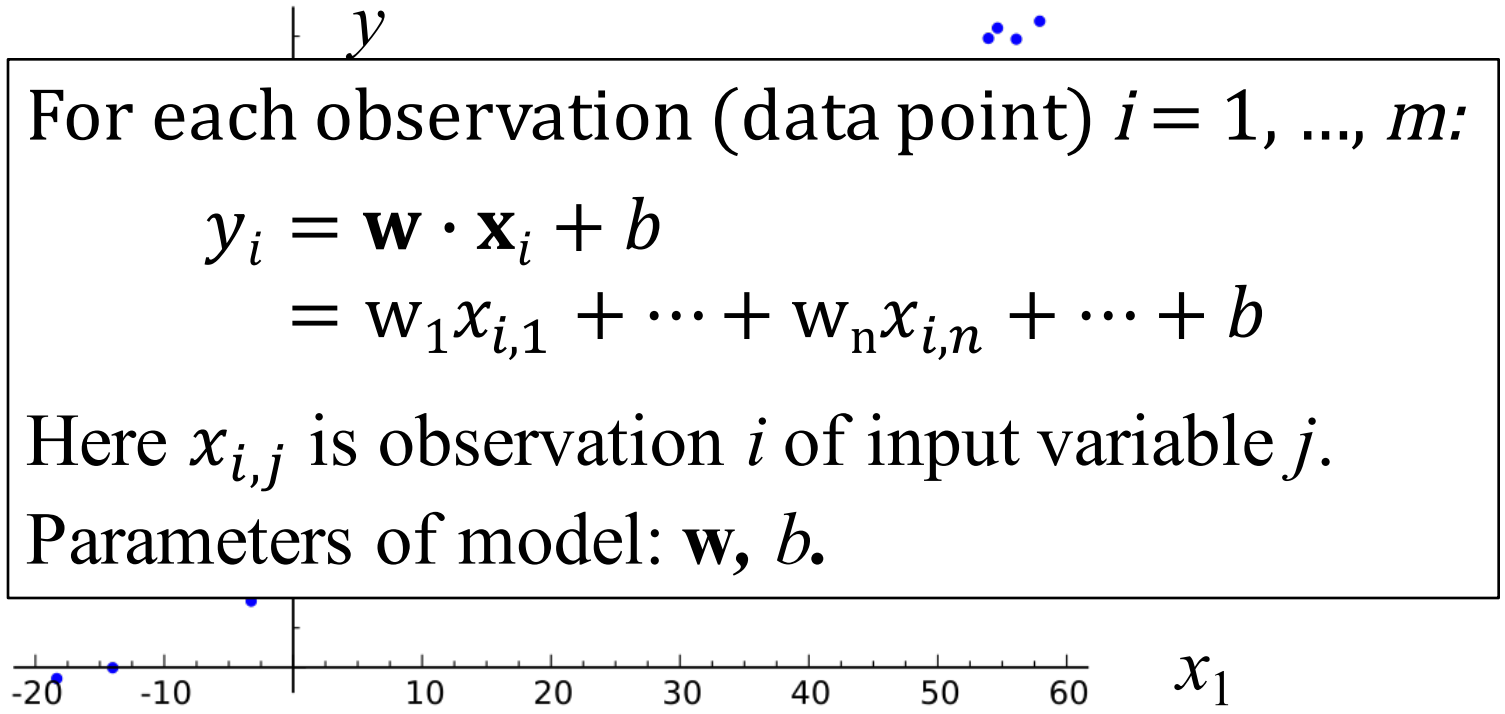
Linear Regression

- Uses a linear model to model relationship between dependent variable $y \in \mathbb{R}$, and input (independent) variables $x_1, \ldots, x_n \in \mathbb{R}^n$
- Is this supervised or unsupervised learning?



Linear Regression

- Uses a linear model to model relationship between dependent variable $y \in \mathbb{R}$, and input (independent) variables $x_1, \ldots, x_n \in \mathbb{R}^n$

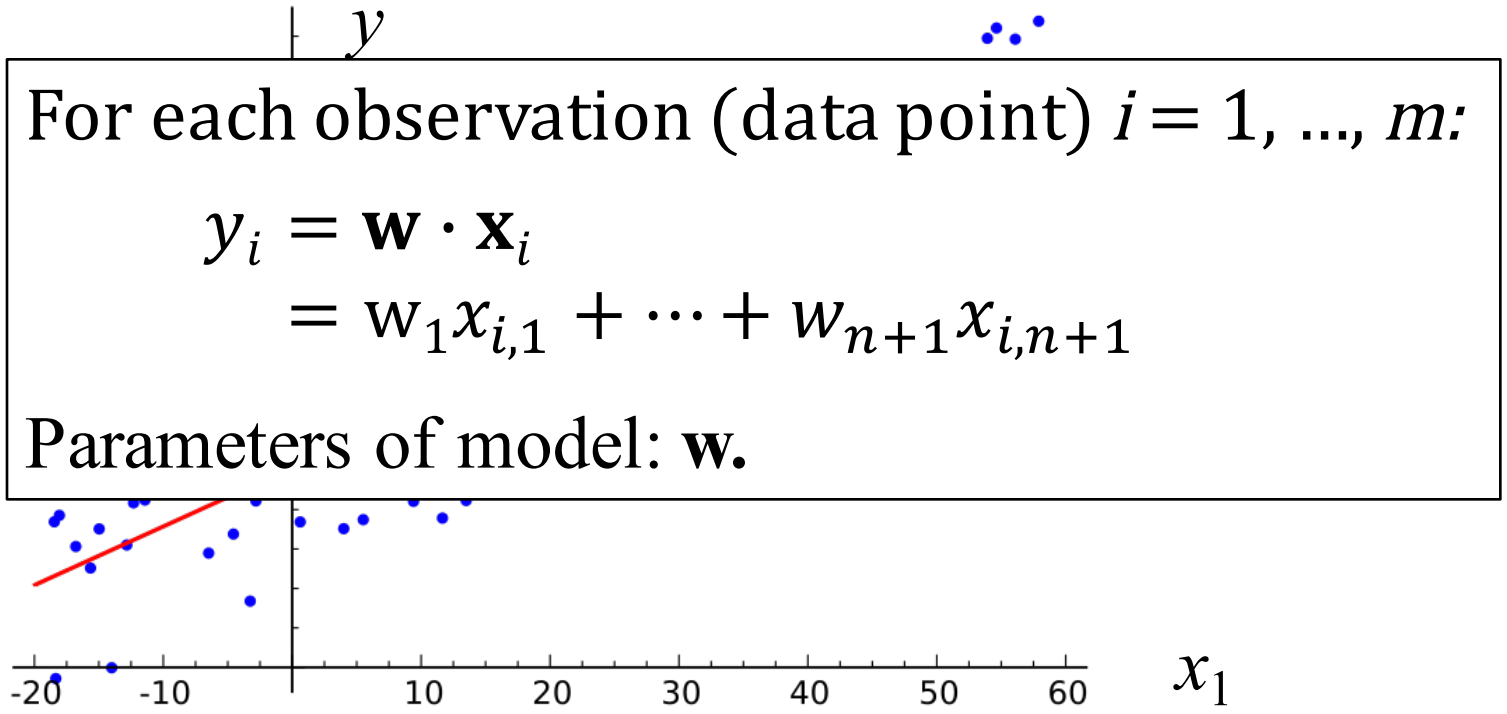


Linear Regression

- Can simply the model by adding additional input that is always one:

  $x_{i,n+1} = 1 \qquad i = 1, \ldots, m$

- The corresponding parameter in w is called the **intercept**.



For each observation (data point) $i = 1, \ldots, m$:

$$y_i = \mathbf{w} \cdot \mathbf{x}_i$$
$$= w_1 x_{i,1} + \cdots + w_{n+1} x_{i,n+1}$$

Parameters of model: **w.**

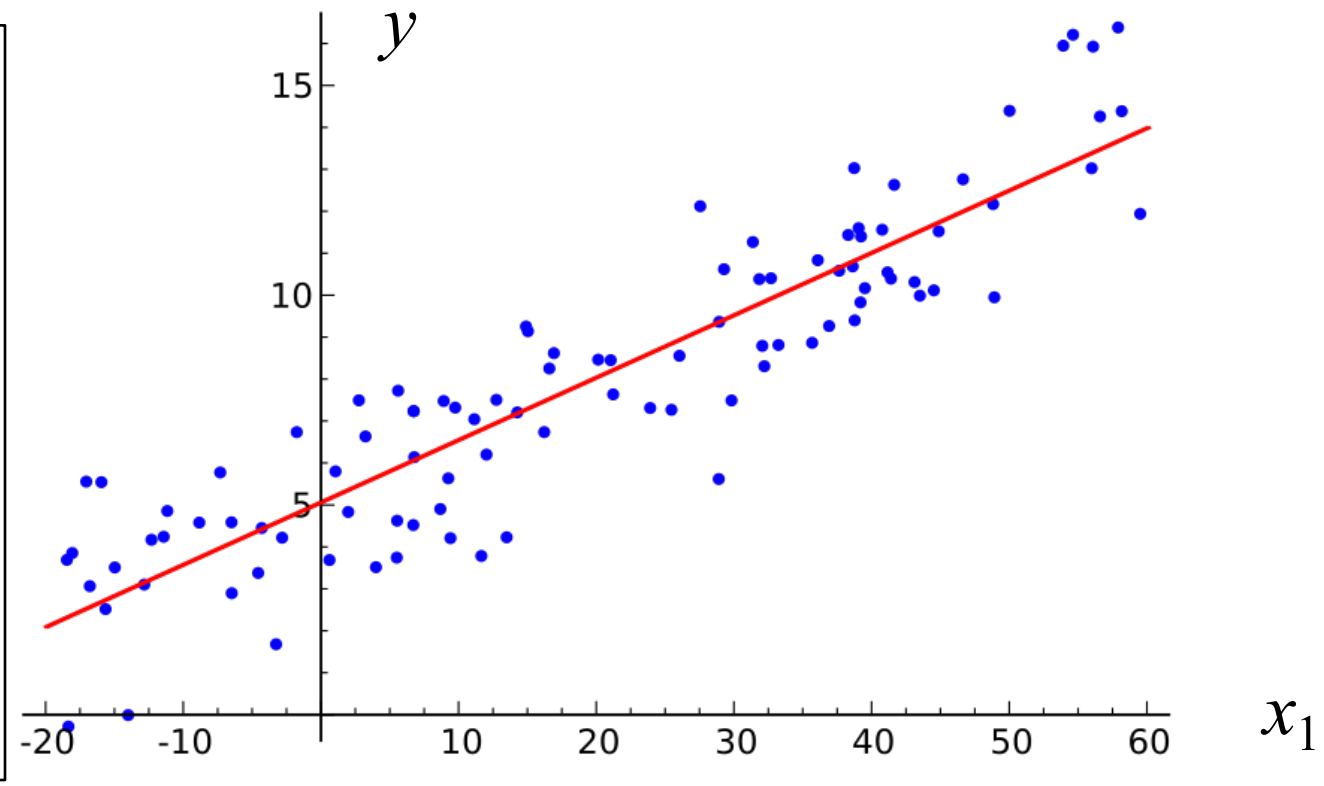
Linear Least Squares Regression

- Define an objective function or loss function to optimize the model
- One loss function: least squares ("least squared error").

$$E = \sum_{i=1}^{m} (\mathbf{w} \cdot \mathbf{x}_i - y_i)^2$$

Parameters of model: **w.**

- What is E for the 2D line fitting case at right? (blackboard)
- How to minimize E?



Linear Least Squares Regression

Set derivatives of objective function with respect to parameters equal to zero.

$$\frac{\partial E}{\partial w_j} = \frac{\partial}{\partial w_j} \sum_{i=1}^{m} (\mathbf{w} \cdot \mathbf{x}_i - y_i)^2 = 0$$

$$2 \sum_{i=1}^{m} (\mathbf{w} \cdot \mathbf{x}_i - y_i) x_{ij} = 0$$

$$\sum_{i=1}^{m} \left(\sum_{k=1}^{n} x_{ik} w_k - y_i \right) x_{ij} = 0$$

Normal equations:

$$\sum_{i=1}^{m} \sum_{k=1}^{n} x_{ik} x_{ij} w_k = \sum_{i=1}^{m} x_{ij} y_i$$

Linear Least Squares Regression

- Normal equations in matrix form:

$$(\mathbf{X}^{\mathrm{T}}\mathbf{X})\mathbf{w} = \mathbf{X}^{\mathrm{T}}\boldsymbol{y}$$

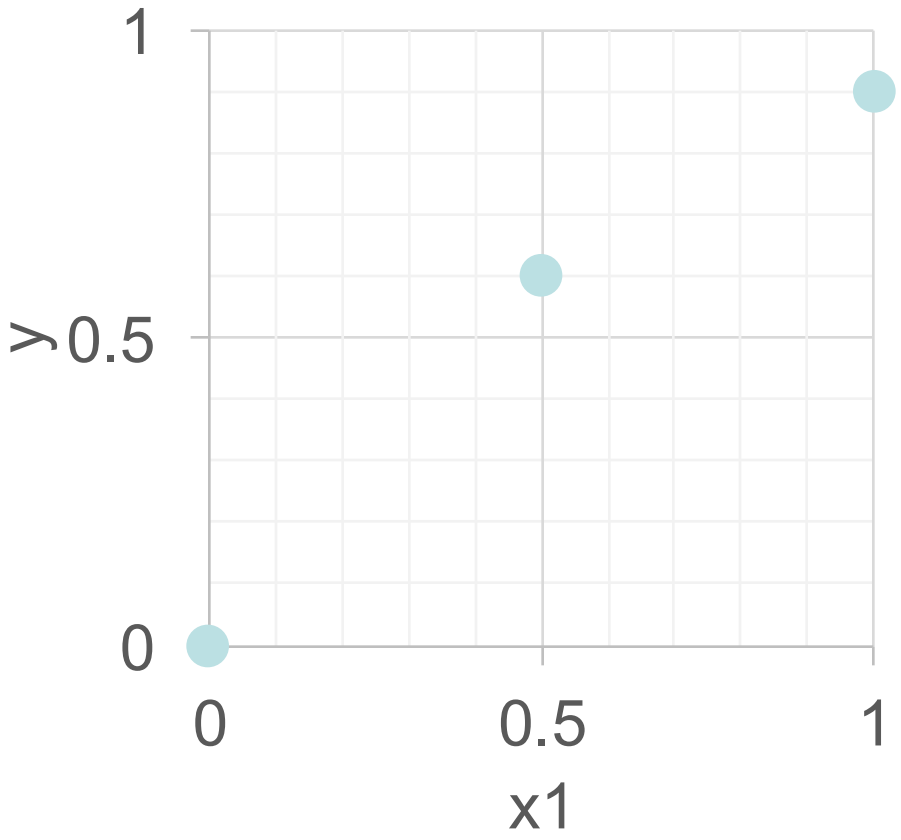
$$\mathbf{w} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}$$

$\mathbf{X}$ is the matrix with x_{ij}
being observation i of input variable j.

$\mathbf{y}$ is the vector of dependent variable (output) observations.

Linear Least Squares Example

- Suppose we have three observations (m=3) of one input variable x_1:



x_1	y
0	0
0.5	0.6
1	0.9

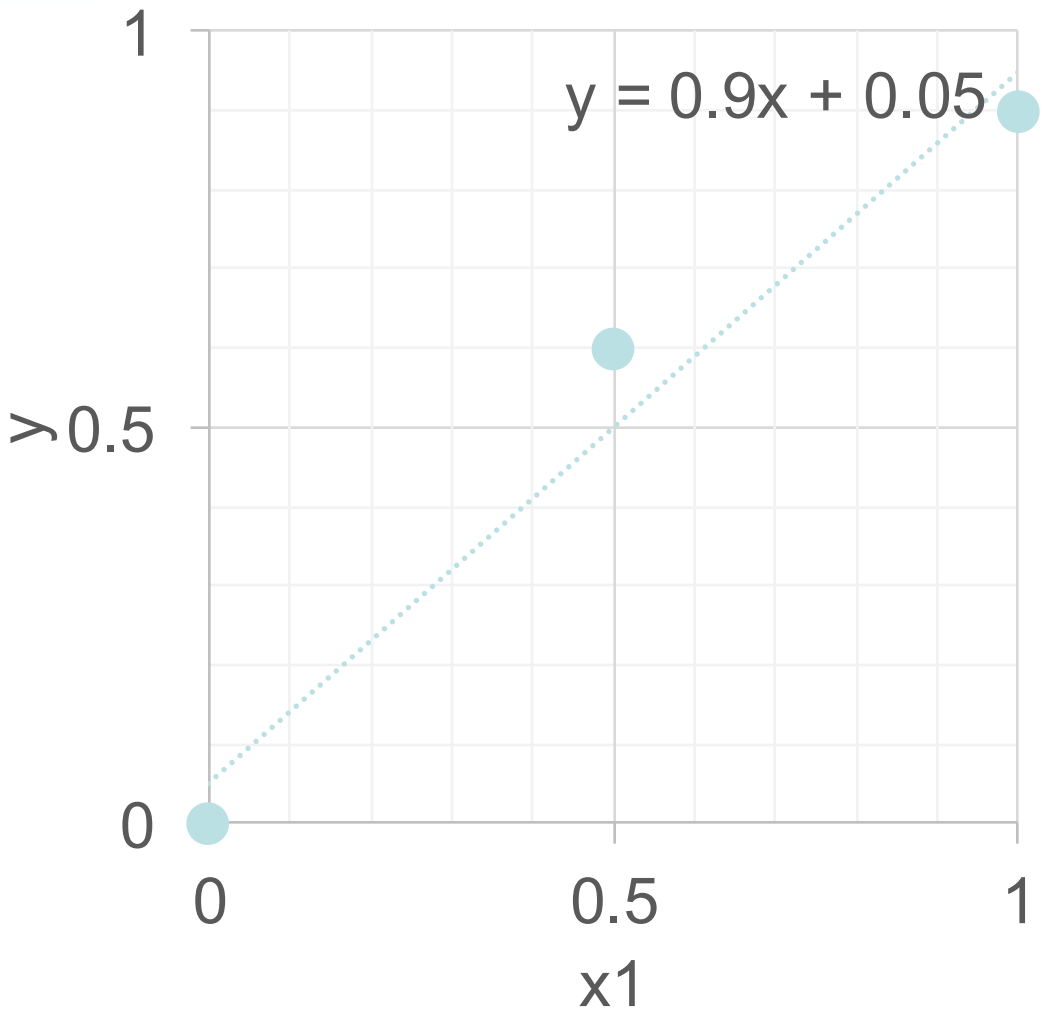
Linear Least Squares Example

- Suppose we have three observations (m=3) of one input variable x_1.
- Add additional constant variable x_2:

x_1	x_2	y
0	1	0
0.5	1	0.6
1	1	0.9

- $\mathbf{X} = \begin{bmatrix} 0 & 1 \\ 0.5 & 1 \\ 1 & 1 \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} 0 \\ 0.6 \\ 0.9 \end{bmatrix}$, so $\mathbf{w} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y} = \begin{bmatrix} 0.9 \\ 0.05 \end{bmatrix}$

Linear Least Squares Example



Overview

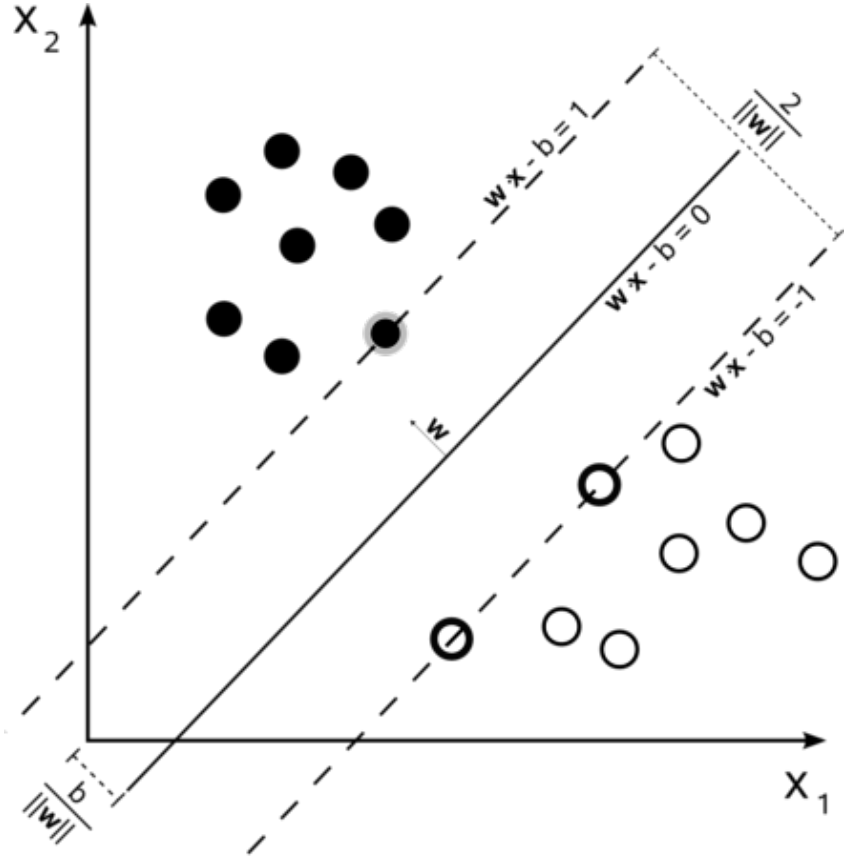
- Supervised, unsupervised, and reinforcement learning
- Simple learning models
  - Clustering
  - Linear regression
  - **Linear Support Vector Machines (SVM)**
  - k-Nearest Neighbors
- Overfitting and generalization
- Training, testing, validation
- Balanced datasets
- Measuring performance of a classifier

Linear Support Vector Machines (SVM)

- In linear regression, we had input variables $x_1, \ldots, x_n$ and we regressed them against a dependent variable $y \in \mathbb{R}$
- But what if we want to make a classifier?
- For example, a binary classifier could predict either $y = -1$, $y = 1$
- One simple option: use linear regression to find a linear model that best fits the data
- But this will not necessarily generalize well to new inputs.

Linear Support Vector Machines (SVM)

- Idea: if data are separable by a linear hyperplane, then maximize separation distance (margin) between points.



From Wikipedia

Linear Support Vector Machines (SVM)

- Two hyperplanes:

$$\vec{w} \cdot \vec{x} + b = 1$$

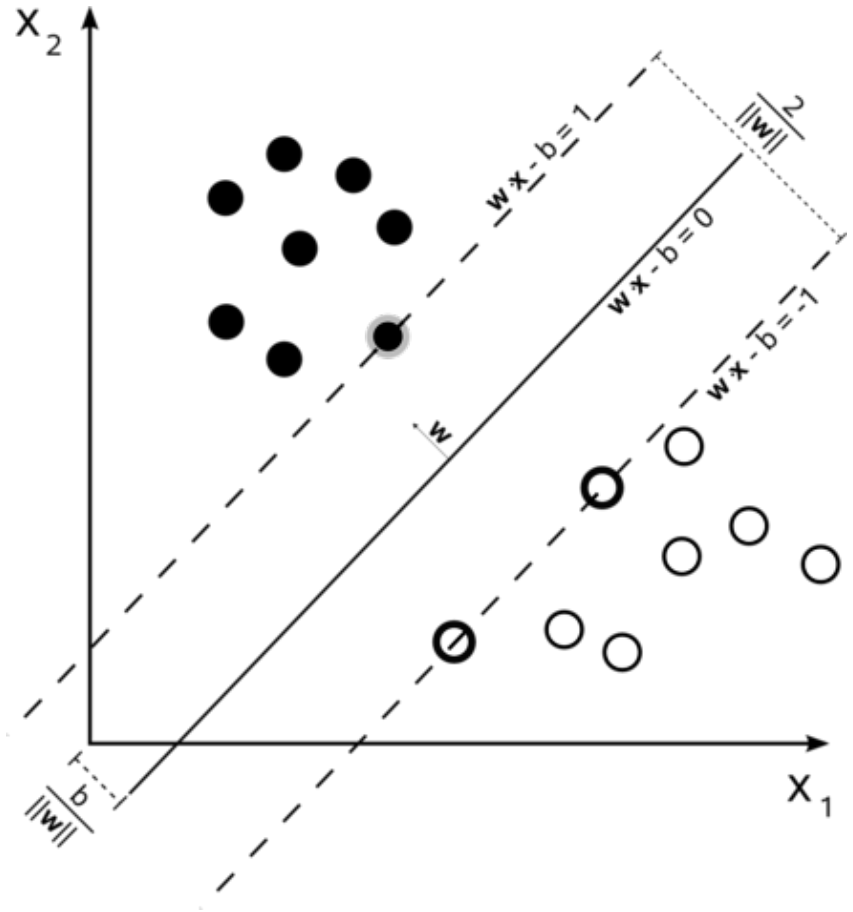
and

$$\vec{w} \cdot \vec{x} + b = -1.$$

- Distance between hyperplanes is:

$$\frac{2}{\|\vec{w}\|}$$

- So to maximize distance, minimize $\|\vec{w}\|$



From Wikipedia

Linear Support Vector Machines (SVM)

- For each point *i*, either:

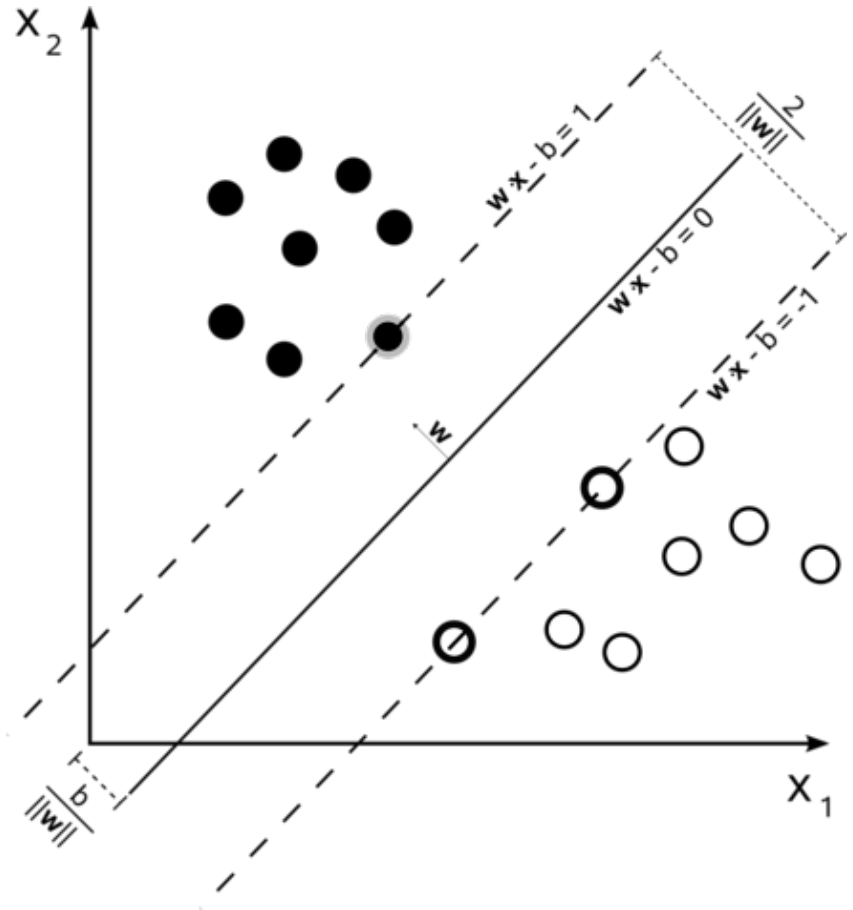
$$\vec{w} \cdot \vec{x}_i + b \geq 1, \text{ if } y_i = 1$$

or

$$\vec{w} \cdot \vec{x}_i + b \leq -1, \text{ if } y_i = -1.$$

- This can be rewritten as, for each *i*:

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1$$



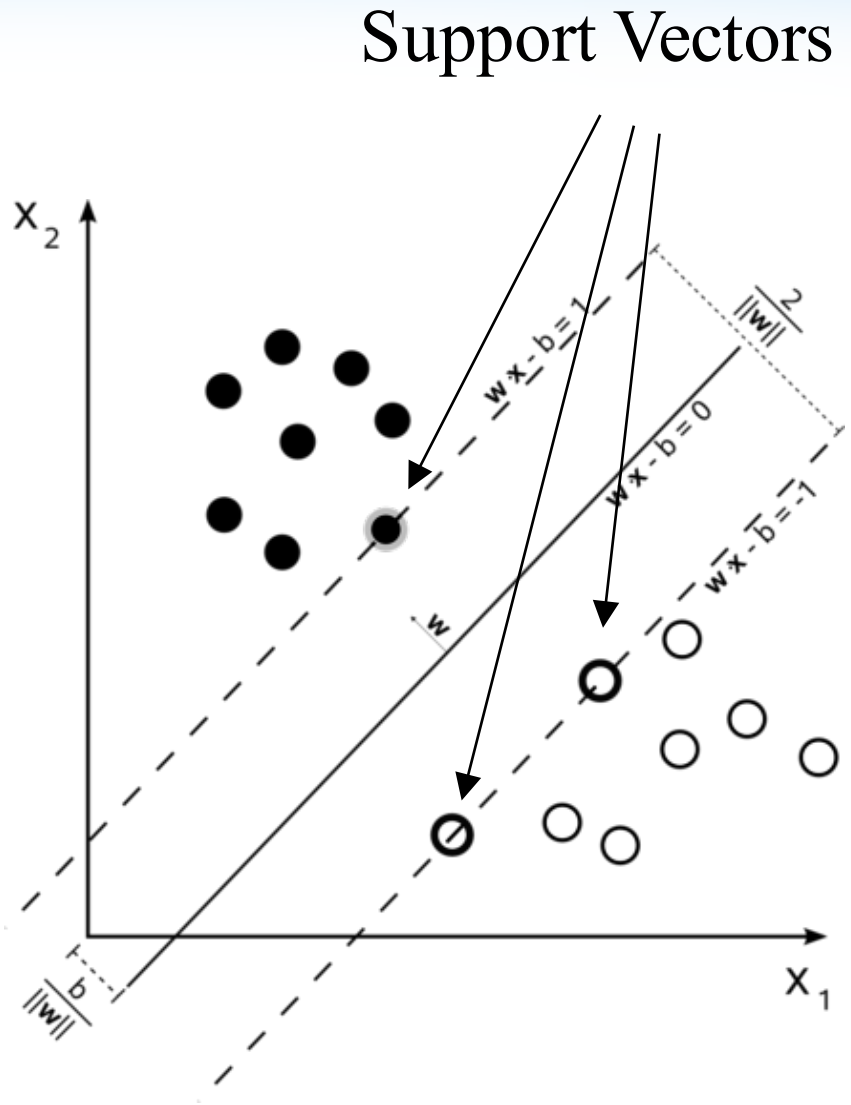
From Wikipedia

Linear Support Vector Machines (SVM)

- So our minimization problem becomes:
- Minimize $\|\vec{w}\|$ subject to the constraint:

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 \qquad i = 1, \ldots, m$$

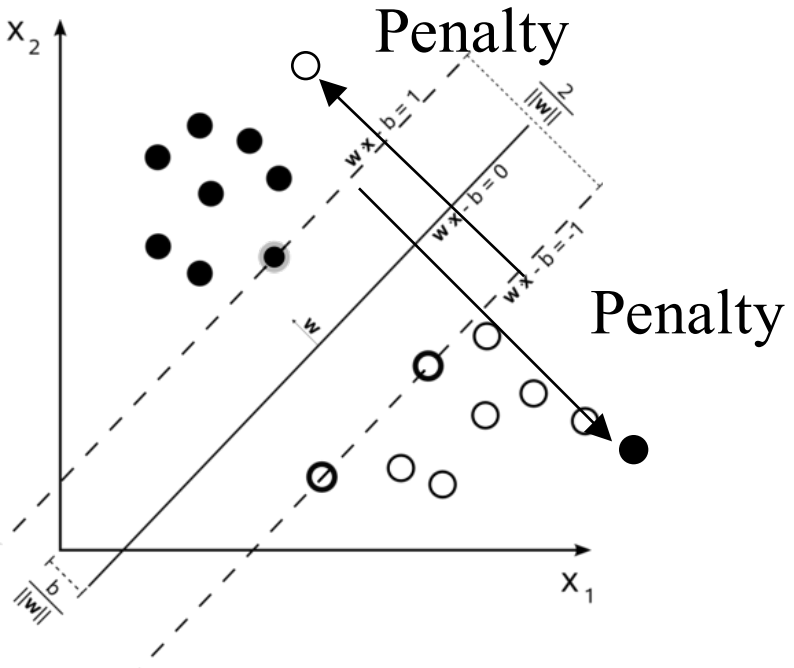
- Can be solved with quadratic programming
- Maximizes distance (margin) between two classes of data



From Wikipedia

Linear Support Vector Machines (SVM)

- If data are not linearly separable, can use a **soft margin classifier**, which has an objective function that sums for all data points *i*, a penalty of zero if the data point is correctly classified, otherwise, the distance to the margin.



How to Use Linear SVMs in Deep Learning?

- Linear SVMs tend to perform well with small amounts of training data
- Deep learning tends to perform well with large amounts of data

- What to do if we have a new problem with only a small dataset?
- One solution: use linear SVM
- Another solution: transfer learning.
  - Use a deep learning model trained on different problem
  - Train a linear SVM using features extracted from neural network.

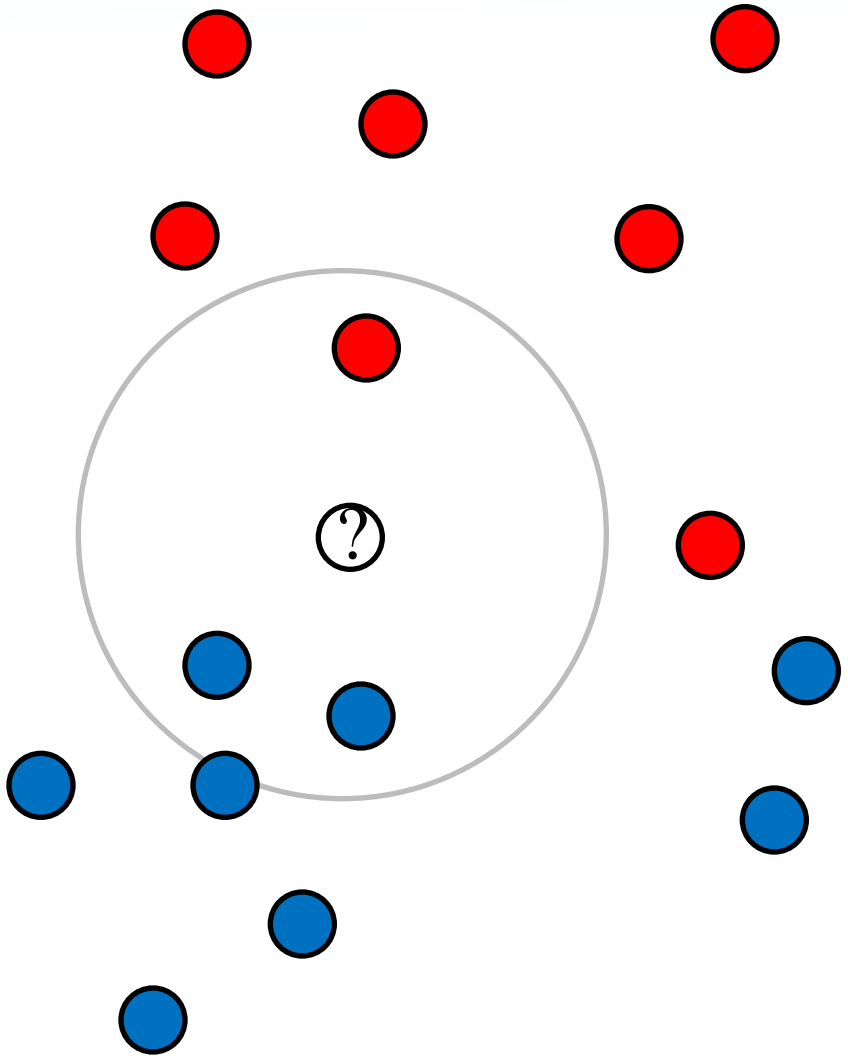
Overview

- Supervised, unsupervised, and reinforcement learning
- Simple learning models
  - Clustering
  - Linear regression
  - Linear Support Vector Machines (SVM)
  - **k-Nearest Neighbors**
- Overfitting and generalization
- Training, testing, validation
- Balanced datasets
- Measuring performance of a classifier

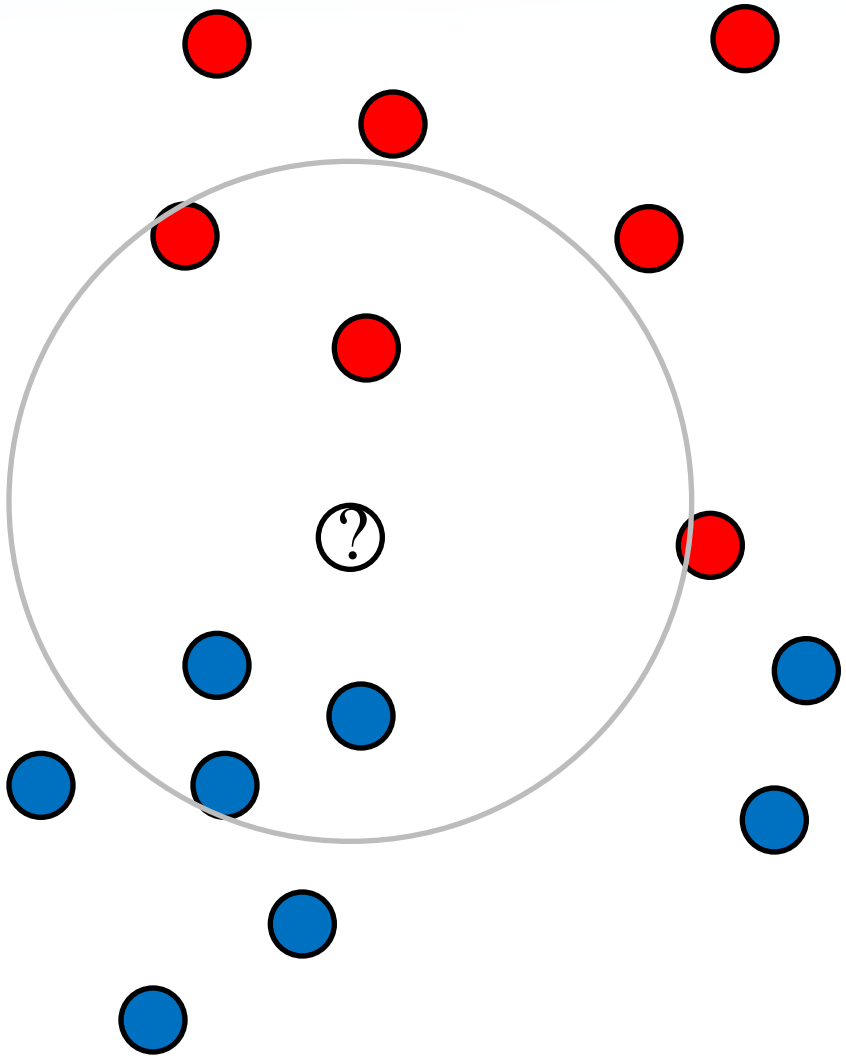
k-Nearest Neighbors

- Suppose we can measure distance between input features.
- For example, Euclidean distance: $D(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^2$
- *k*-Nearest Neighbors simply uses the distance to the nearest *k* points to determine the classification or regression.
  - Classifier: take most common class within the *k* nearest points
  - Regression: take mean of *k* nearest points
- No parameters, so no need to “train” the algorithm

k-Nearest Neighbors Example, k=3



k-Nearest Neighbors Example, k=5

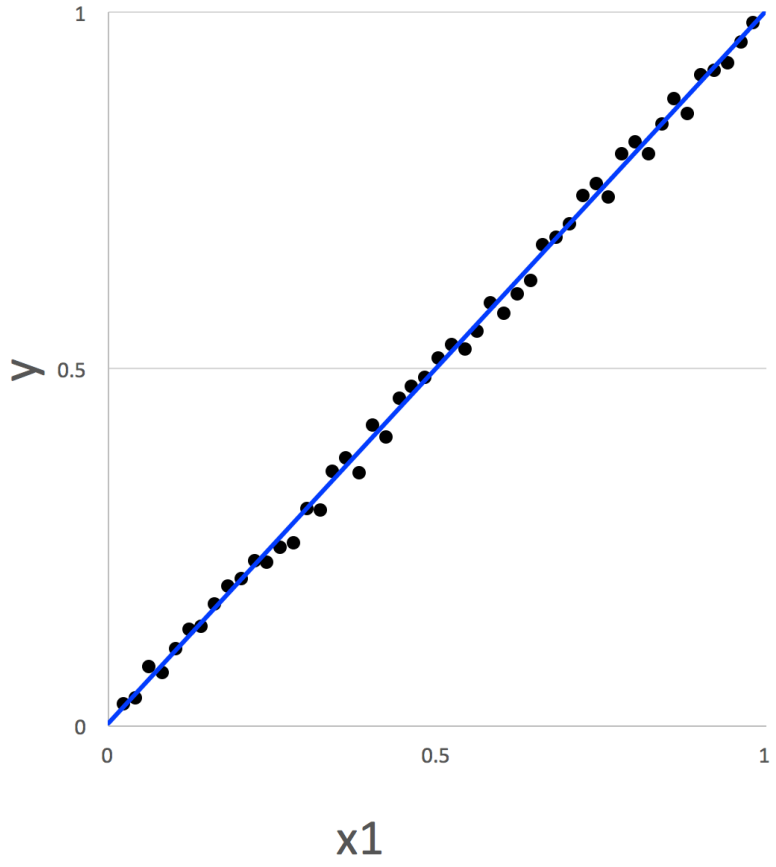


Overview

- Supervised, unsupervised, and reinforcement learning
- Simple learning models
  - Clustering
  - Linear regression
  - Linear Support Vector Machines (SVM)
  - k-Nearest Neighbors
- **Overfitting and generalization**
- Training, testing, validation
- Balanced datasets
- Measuring performance of a classifier

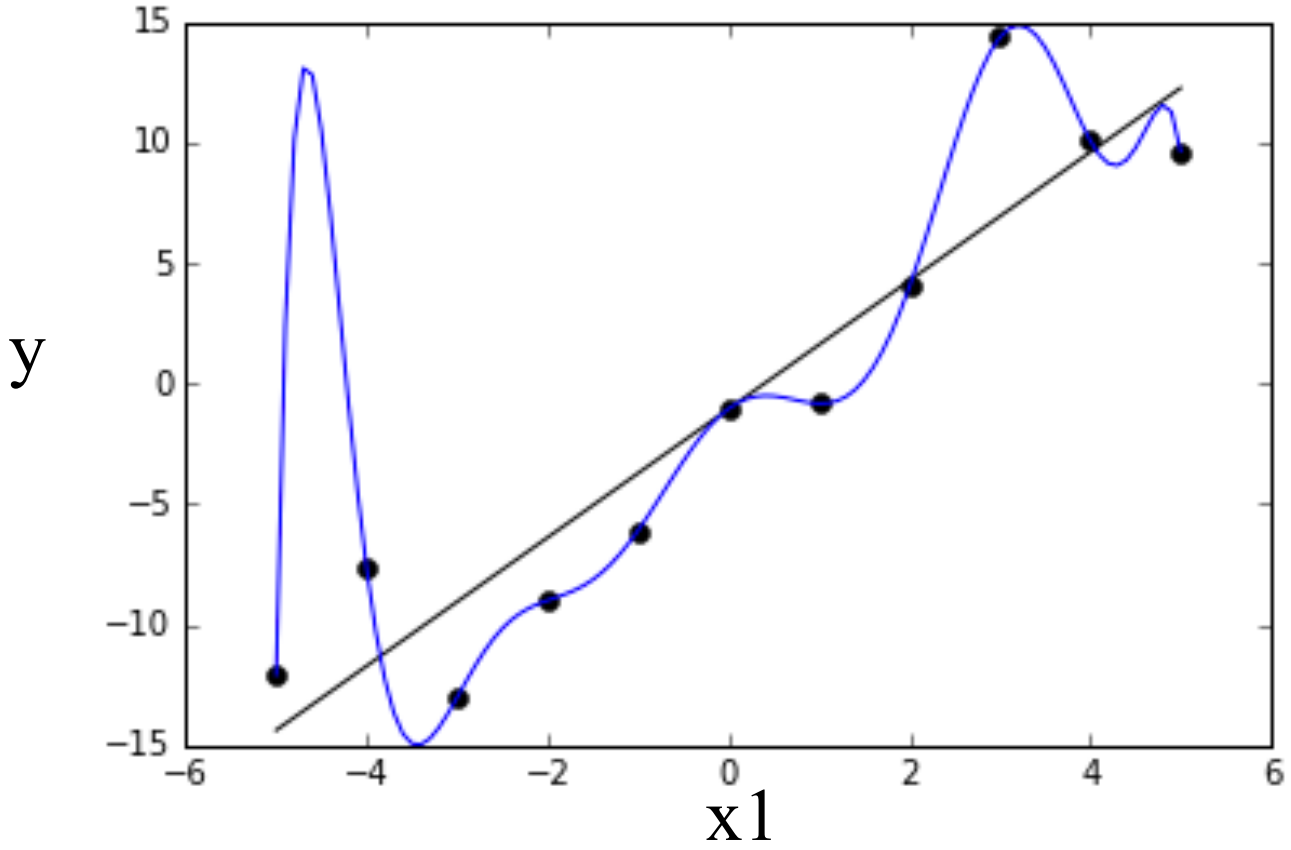
Overfitting and generalization

- Will this model have decent prediction for new inputs? (i.e. inputs similar to the training exemplars in blue)



Overfitting and generalization

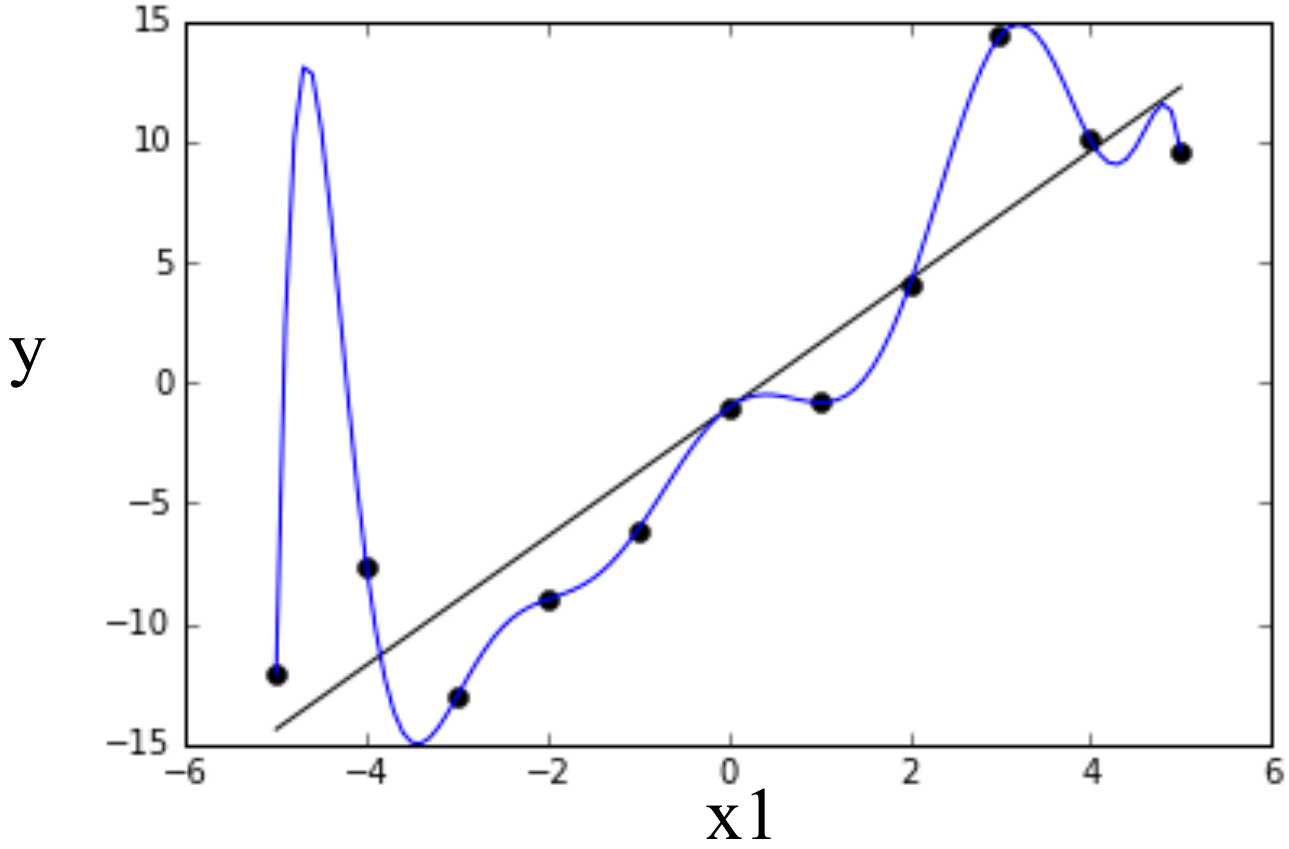
- How about the model here, shown as the blue curve?



From Wikipedia

Overfitting and generalization

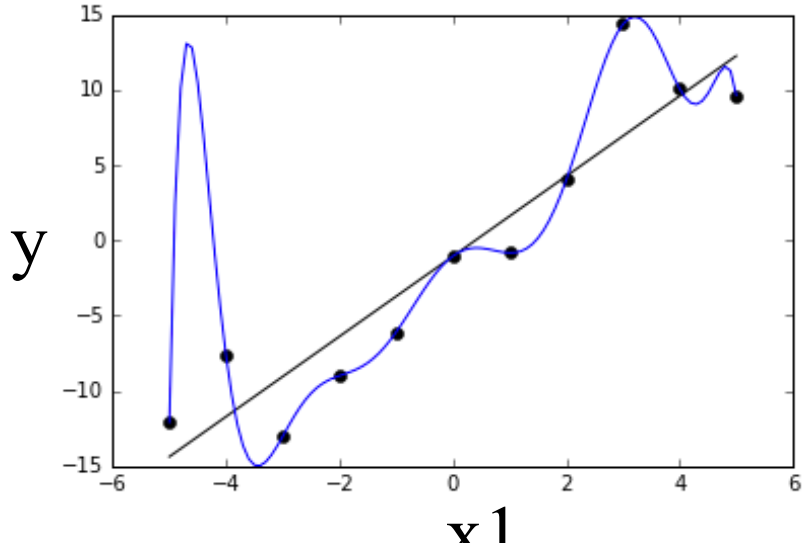
- **Overfitting**: the model describes random noise or errors instead of the underlying relationship.



From Wikipedia

Overfitting and generalization

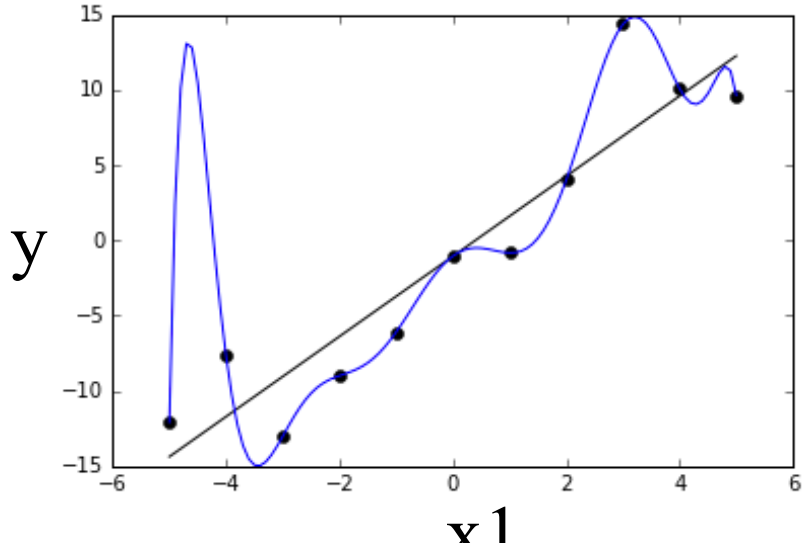
- **Overfitting**: the model describes random noise or errors instead of the underlying relationship.
- Frequently occurs when model is overly complex (e.g. has too many parameters) relative to the number of observations.
- Has poor predictive performance.



From Wikipedia

Overfitting and generalization

- **Overfitting**: the model describes random noise or errors instead of the underlying relationship.
- Frequently occurs when model is overly complex (e.g. has too many parameters) relative to the number of observations.
- Has poor predictive performance.



From Wikipedia

Overfitting and generalization

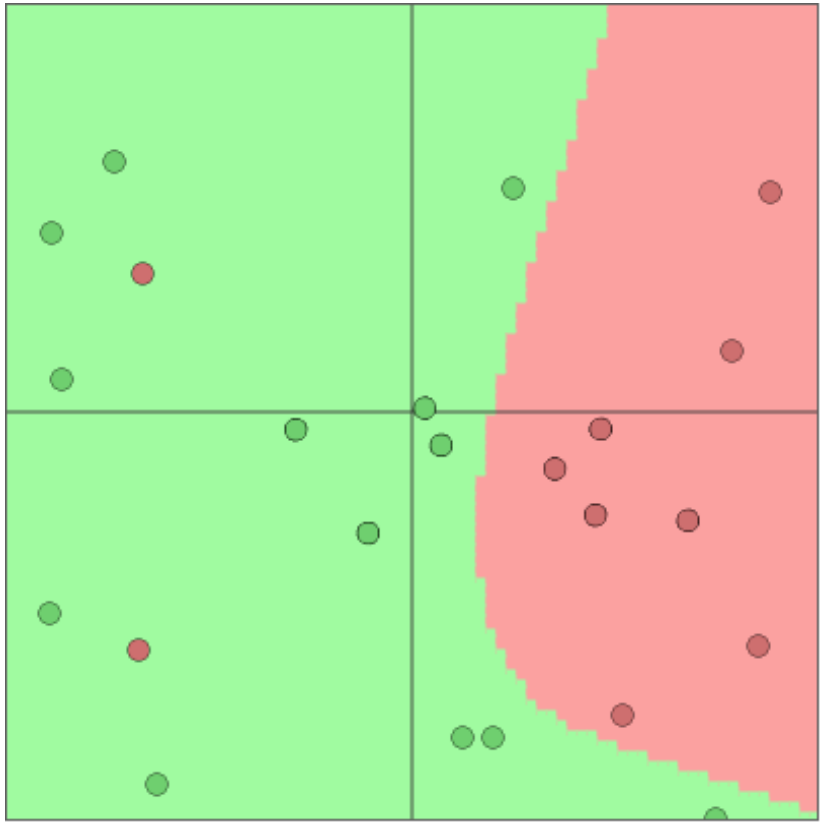
- **A rule of thumb for linear regression: one in ten rule**
- One predictive variable can be studied for every ten events.

- In general, want number of data points >> number of parameters.
- But models with more parameters often perform better!
- One solution: gradually increase number of parameters in model until it starts to overfit, and then stop.

From Wikipedia

Overfitting Example with 2D Classifier

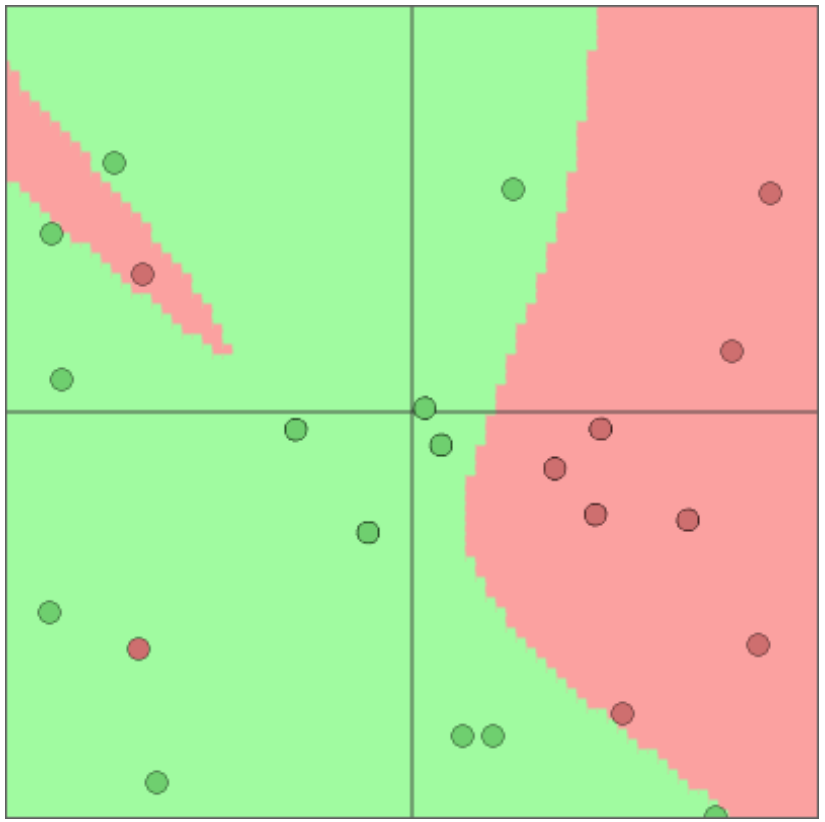
- From [ConvnetJS Demo: 2D Classification with Neural Networks]



23 data points, 17 parameters (5 neurons)

Overfitting Example with 2D Classifier

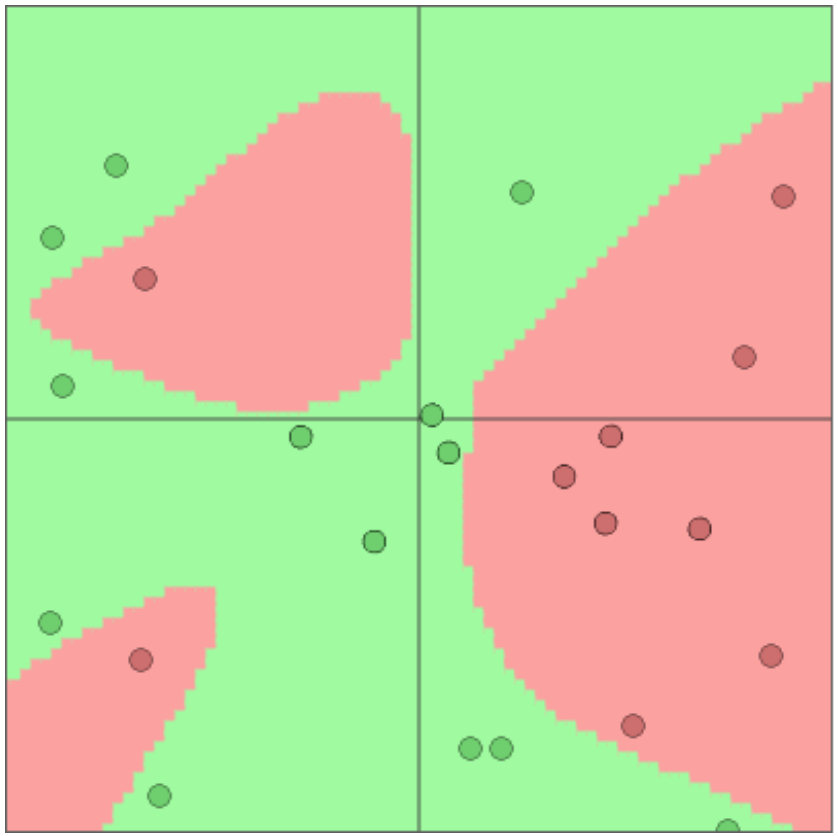
- From [ConvnetJS Demo: 2D Classification with Neural Networks]



23 data points, 32 parameters (10 neurons)

Overfitting Example with 2D Classifier

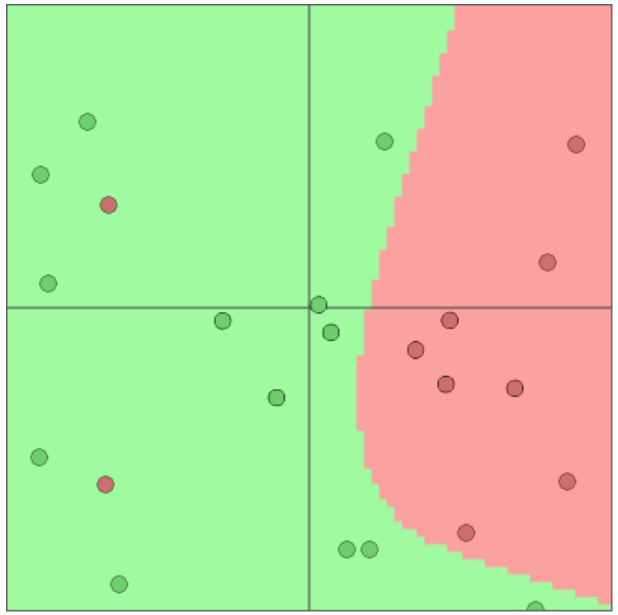
- From [ConvnetJS Demo: 2D Classification with Neural Networks]

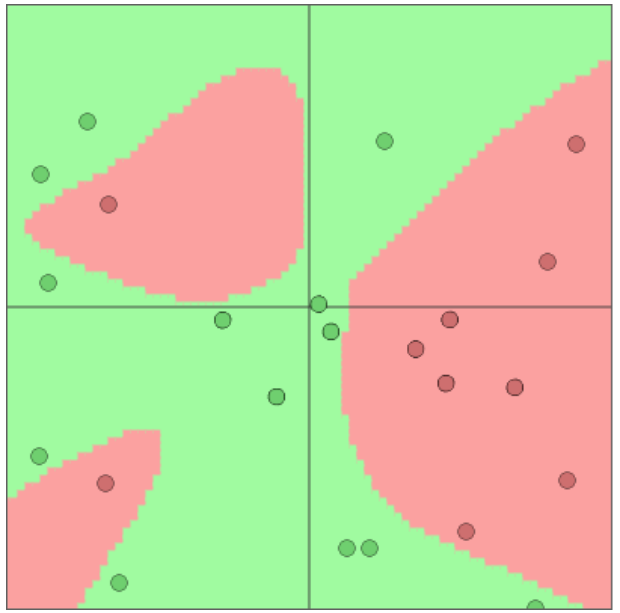


23 data points, 102 parameters (22 neurons total)

Generalization

- **Generalization error** is a measure of how accurately an algorithm is able to predict outcome values for previously unseen data.
- A model that is **overfit** will have poor generalization.





Overview

- Supervised, unsupervised, and reinforcement learning
- Simple learning models
  - Clustering
  - Linear regression
  - Linear Support Vector Machines (SVM)
  - k-Nearest Neighbors
- Overfitting and generalization
- **Training, testing, validation**
- Balanced datasets
- Measuring performance of a classifier

Training, testing, validation

- Break dataset into three parts by random sampling:
  - **Training dataset**: model is fit directly to this data
  - **Testing dataset**: model sees this data only once;
    used to measure the final performance of the classifier.
  - **Validation dataset:** model is repeatedly “tested” on this data during the training process to gain insight into overfitting
- **Common percentages:**
  - Training (80%), testing (15%), validation (5%).

Training, testing, validation

- For neural networks, typically keep running training until validation error increases, then stop.

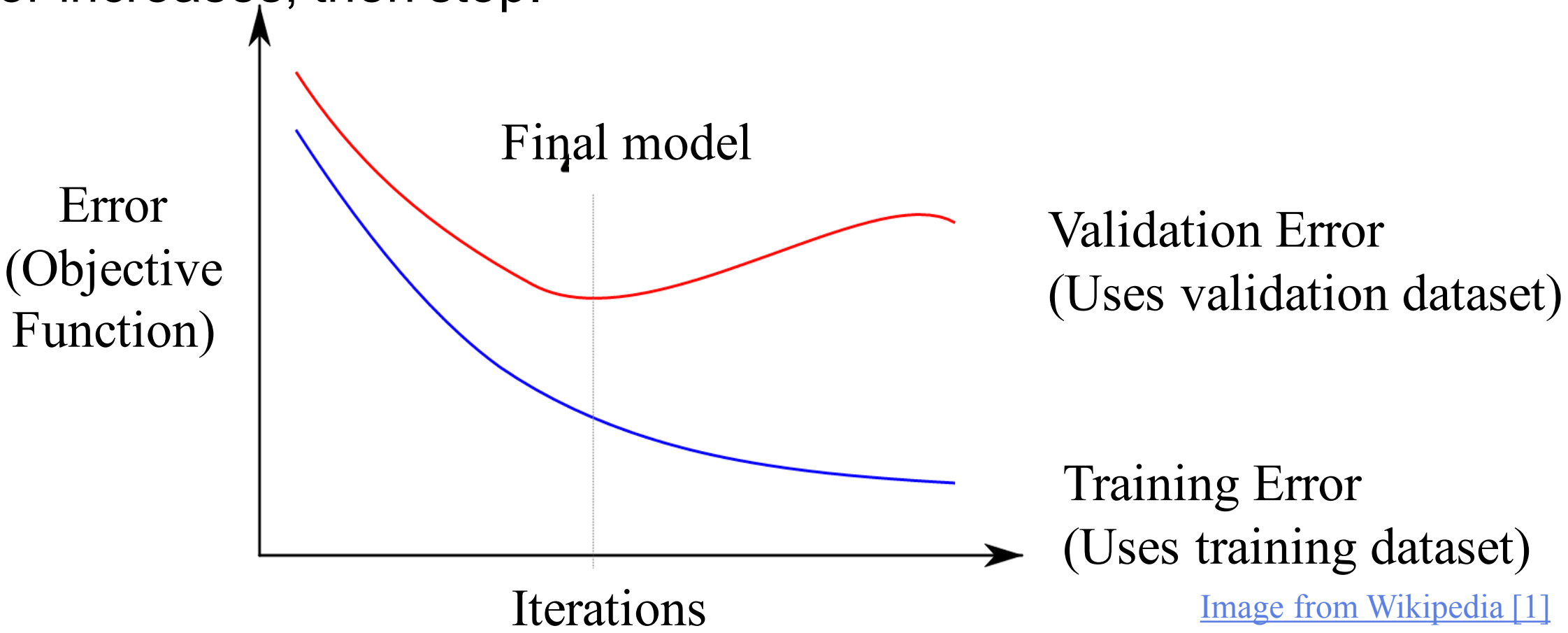


Image from Wikipedia [1]

Cross-validation

- Repeatedly partition data into subsets: training and test.
- Take mean performance of classifier over all such partitions.
- **Leave one out**: train on n-1 samples, test on 1 sample.
  - Requires training n times.
- **k-fold cross-validation**: randomly partition data into k subsets (folds), at each iteration, train on k-1 folds, test on the other fold.
  - Requires training k times.
  - Common: 10-fold cross-validation
- Less common for deep learning (why?)

Overview

- Supervised, unsupervised, and reinforcement learning
- Simple learning models
  - Clustering
  - Linear regression
  - Linear Support Vector Machines (SVM)
  - k-Nearest Neighbors
- Overfitting and generalization
- Training, testing, validation
- **Balanced datasets**
- Measuring performance of a classifier

Balanced datasets

- **Unbalanced dataset**:
  - Suppose we have a binary classification problem (labels: 0, 1)
  - Suppose 99% of our observations are class 0.
  - We might learn the model “everything is zero.”
  - This model would be 99% accurate, but not model class 1 at all.
- **Balanced dataset**:
  - Equal numbers of observations of each class

Overview

- Supervised, unsupervised, and reinforcement learning
- Simple learning models
  - Clustering
  - Linear regression
  - Linear Support Vector Machines (SVM)
  - k-Nearest Neighbors
- Overfitting and generalization
- Training, testing, validation
- Balanced datasets
- **Measuring performance of a classifier**

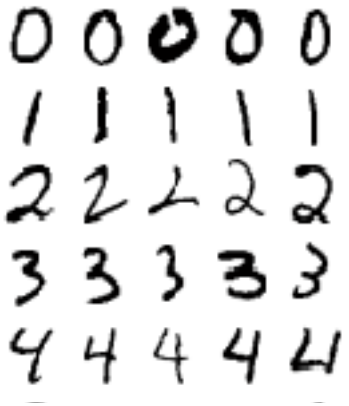
Confusion Matrix

- If n classes, n x n matrix comparing actual versus predicted classes.

Actual Class	Class Predicted by Model: Cat	Class Predicted by Model: Dog
Cat	9	1
Dog	4	6

Example: Handwritten Digit Recognition

Class	1	2	3	4	5	6	7	8	9	0	Error Rate
1	191	0	0	5	1	0	1	0	2	0	4.5
2	0	188	2	0	0	1	3	0	0	6	6.0
3	0	3	191	0	1	0	2	0	3	0	4.5
4	8	0	0	187	4	0	1	0	0	0	6.5
5	0	0	0	0	193	0	0	0	7	0	3.5
6	0	0	0	0	1	196	0	2	0	1	2.0
7	2	2	0	2	0	1	190	0	1	2	5.0
8	0	1	0	0	1	2	2	196	0	0	2.0
9	5	0	2	0	8	0	3	0	179	3	10.5
0	1	4	0	0	0	1	1	0	1	192	4.5
									Overall error rate		4.85%



Slide from Nelson Morgan at ICSI / Berkeley

Example: Handwritten Digit Recognition

- Visualization from MathWorks

Confusion Matrix

- For binary classifier, can call one class positive, the other negative.
- Should we call cats positive or negative?

Photo from [1]

Actual Class	Class Predicted by Model: Cat	Class Predicted by Model: Dog
Cat	9	1
Dog	4	6

Confusion Matrix

- For binary classifier, can call one class positive, the other negative.
- Cats are cuter, so cats = positive.

		Class Predicted by Model	
		Positive	**Negative**
Actual Class	**Positive**	9	1
	Negative	4	6

Confusion Matrix

- For binary classifier, can call one class positive, the other negative.

		Class Predicted by Model	
		Positive	**Negative**
Actual Class	**Positive**	True positive	1
	Negative	4	True negative

Confusion Matrix

- For binary classifier, can call one class positive, the other negative.

		Class Predicted by Model	
		Positive	**Negative**
Actual Class	**Positive**	True positive	**?**
	Negative	?	True negative

Confusion Matrix

- For binary classifier, can call one class positive, the other negative.

		Class Predicted by Model	
		Positive	**Negative**
Actual Class	**Positive**	True positive	False negative
	Negative	False positive	True negative

Classifier Performance

- **Accuracy**: (TP + TN) / (Population Size)
- **Precision**: TP / (Predicted Positives) = (TP + FP)
- **Recall**: TP / (Actual Positives) = TP / (TP + FN)
  (also known as **sensitivity, true positive rate**)
- **Specificity**: TN / (Actual Negatives) = TN / (TN + FP)
  (also known as **true negative rate**)

		Class Predicted by Model	
		Positive	**Negative**
Actual Class	**Positive**	True positive	False negative
	Negative	False positive	True negative

Classifier Performance: Unbalanced Dataset

- **Accuracy**: (TP + TN) / (Population Size)
- **Precision**: TP / (Predicted Positives) = (TP + FP)
- **Recall**: TP / (Actual Positives) = TP / (TP + FN)
  (also known as **sensitivity, true positive rate**)
- **Specificity**: TN / (Actual Negatives) = TN / (TN + FP)
  (also known as **true negative rate**)

		Class Predicted by Model: Positive	Class Predicted by Model: Negative
Actual Class	Positive	TP: 99	FN: 0
Actual Class	Negative	FP: 1	TN: 0

Classifier Performance: Unbalanced Dataset

- **Accuracy**: (TP + TN) / (Population Size) = ?
- **Precision**: TP / (Predicted Positives) = (TP + FP)
- **Recall**: TP / (Actual Positives) = TP / (TP + FN)
  (also known as **sensitivity, true positive rate**)
- **Specificity**: TN / (Actual Negatives) = TN / (TN + FP)
  (also known as **true negative rate**)

		Class Predicted by Model	
		Positive	**Negative**
Actual Class	**Positive**	TP: 99	FN: 0
	Negative	FP: 1	TN: 0

Classifier Performance: Unbalanced Dataset

- **Accuracy**: (TP + TN) / (Population Size) = 99%
- **Precision**: TP / (Predicted Positives) = (TP + FP) = ?
- **Recall**: TP / (Actual Positives) = TP / (TP + FN)
  (also known as **sensitivity, true positive rate**)
- **Specificity**: TN / (Actual Negatives) = TN / (TN + FP)
  (also known as **true negative rate**)

		Class Predicted by Model	
		Positive	**Negative**
Actual Class	**Positive**	TP: 99	FN: 0
	Negative	FP: 1	TN: 0

Classifier Performance: Unbalanced Dataset

- **Accuracy**: (TP + TN) / (Population Size) = 99%
- **Precision**: TP / (Predicted Positives) = (TP + FP) = 99%
- **Recall**: TP / (Actual Positives) = TP / (TP + FN)
  (also known as **sensitivity, true positive rate**)
- **Specificity**: TN / (Actual Negatives) = TN / (TN + FP)
  (also known as **true negative rate**)

		Class Predicted by Model	
		Positive	**Negative**
Actual Class	**Positive**	TP: 99	FN: 0
	Negative	FP: 1	TN: 0

Classifier Performance: Unbalanced Dataset

- **Accuracy**: (TP + TN) / (Population Size) = 99%
- **Precision**: TP / (Predicted Positives) = (TP + FP) = 99%
- **Recall**: TP / (Actual Positives) = TP / (TP + FN) = ?
  (also known as **sensitivity, true positive rate**)
- **Specificity**: TN / (Actual Negatives) = TN / (TN + FP)
  (also known as **true negative rate**)

		Class Predicted by Model	
		Positive	**Negative**
Actual Class	**Positive**	TP: 99	FN: 0
	Negative	FP: 1	TN: 0

Classifier Performance: Unbalanced Dataset

- **Accuracy**: (TP + TN) / (Population Size) = 99%
- **Precision**: TP / (Predicted Positives) = (TP + FP) = 99%
- **Recall**: TP / (Actual Positives) = TP / (TP + FN) = 100%
  (also known as **sensitivity, true positive rate**)
- **Specificity**: TN / (Actual Negatives) = TN / (TN + FP)
  (also known as **true negative rate**)

		Class Predicted by Model	
		Positive	**Negative**
Actual Class	**Positive**	TP: 99	FN: 0
	Negative	FP: 1	TN: 0

Classifier Performance: Unbalanced Dataset

- **Accuracy**: (TP + TN) / (Population Size) = 99%
- **Precision**: TP / (Predicted Positives) = (TP + FP) = 99%
- **Recall**: TP / (Actual Positives) = TP / (TP + FN) = 100%
  (also known as **sensitivity, true positive rate**)
- **Specificity**: TN / (Actual Negatives) = 0%
  (also known as **true negative rate**)

		Class Predicted by Model	
		Positive	**Negative**
Actual Class	**Positive**	TP: 99	FN: 0
	Negative	FP: 1	TN: 0

Summary

- Supervised, unsupervised, and reinforcement learning
- Simple learning models
  - Clustering
  - Linear regression
  - Linear Support Vector Machines (SVM)
  - k-Nearest Neighbors
- Overfitting and generalization
- Training, testing, validation
- Balanced datasets
- Measuring performance of a classifier