EI5IS102 Traitement de l'Information

Lecture 3: Introduction to Machine Learning

Charles Brazier
Postdoctoral researcher
Université de Bordeaux, CNRS, Bordeaux INP, LaBRI
France

Machine Learning is everywhere

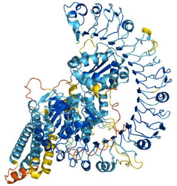
Series Recommenders

Music recommenders

Chatbots

Face recognition

Voice assistants



Drug discovery

Games

Self-driving cars

Image generators

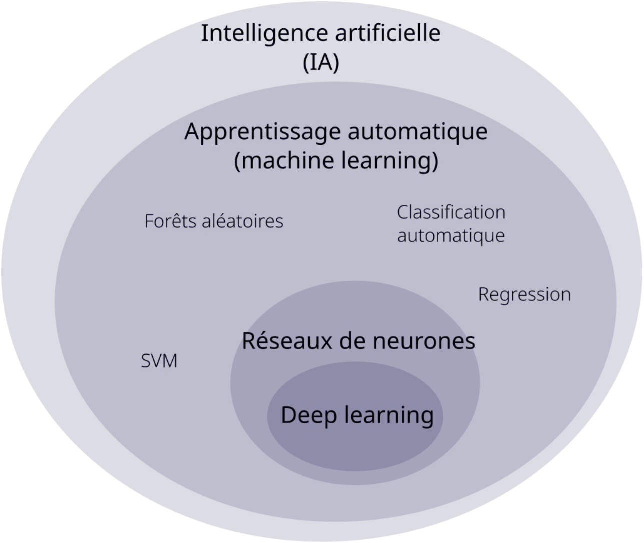
Artificial Intelligence

Artificial Intelligence (AI):

- Inspired by human abilities
- Perception, reasoning, learning

Why "Artificial intelligence"?

- What is **intelligence**?
- Must have something to do with **thinking**
- What are the mechanisms of thinking?
- How to construct a model of thinking?



Wiki

Thinking

William Blake (English poet, 1757–1827) about thinking:

“Man can only desire what he has already perceived”

Noam Chomsky (American linguist, 1928–) about intelligence:

“Take two concepts and create a third one without impairing the two firsts”

Key concepts:

- Creation ex-nihilo cannot really exist*
- Intelligence involves assembling and recognizing patterns based on **prior observations**
- Intelligence is the ability to assemble two ideas that seemed heterogeneous
- But intelligence improves through **learning**

Learning

Past view of AI:

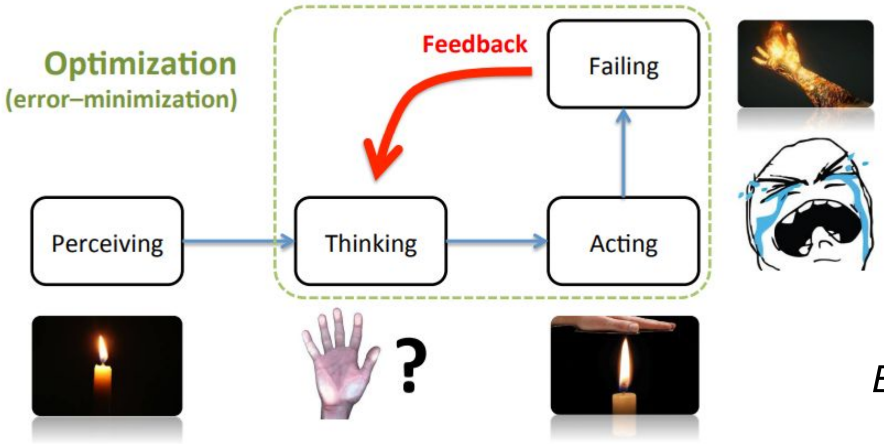
- AI was thought only about **reasoning**
- Programs with predefined rules
- Simulate intelligent decision-making

Current view of AI:

- AI is more about **learning**
- Systems that can improve through experience

Herbert Simon (American psychologist, 1916–2001) about learning:

“Learning is any process by which a system improves performance from experience”



Esling, 2016

Machine Learning

Machine Learning algorithms:

- Programs that can learn from experience to deduce new facts

Early definition of machine learning by **Arthur Samuel** (American pioneer in AI, 1901–1990):

“Field of study that gives computer the ability to learn without being explicitly programmed”

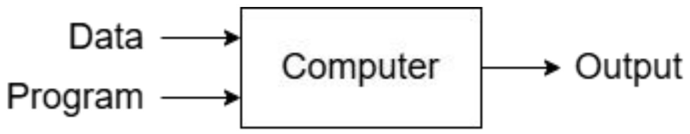
A.Samuel wrote the first self-learning program for checkers.

- Learning to improve its performance by playing against itself
- Invented alpha-beta pruning (decision tree searching)

Machine Learning

How can we have the computer learn without being explicitly programmed?

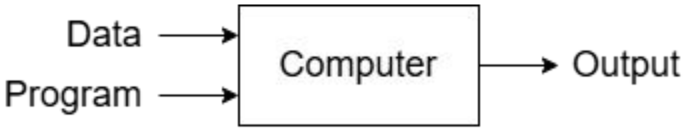
Traditional Programming



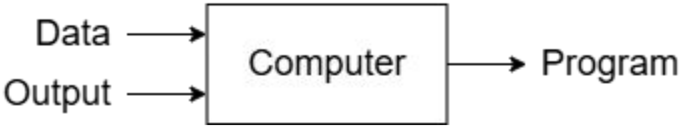
Machine Learning

How can we have the computer learn without being explicitly programmed?

Traditional Programming



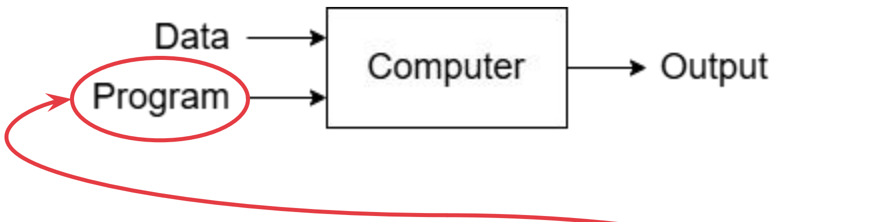
Machine Learning



Machine Learning

How can we have the computer learn without being explicitly programmed?

Traditional Programming



Machine Learning

Machine Learning

How to learn from data?

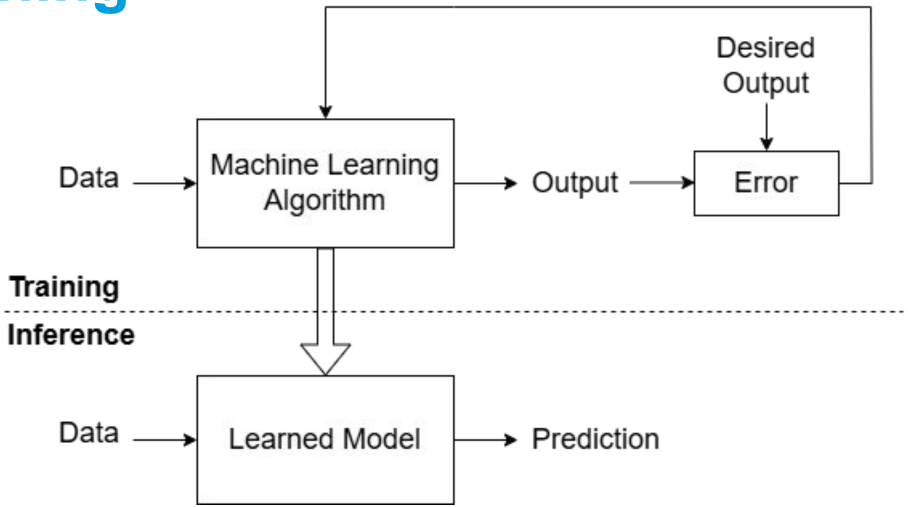
- Memorizing facts: limited by the time of observing facts, by the memory to store facts
- **Generalizing**: deduce new facts from old facts

Learning algorithm:

- A program that can infer useful information from implicit patterns in the data
- Need to figure out what those patterns are
- Then use those patterns to make predictions on new data

Machine Learning

Pipeline:



Training phase:

- Input: a set of observations → **training data**
- Process: **learn patterns** from the training data
- Output: a **model** capable of identifying patterns in the data

Inference phase:

- Input: new observations → unseen data / **test data**
- Process: use the model to make new predictions

Machine Learning

3 major types of learning:

Supervised learning: labeled training data

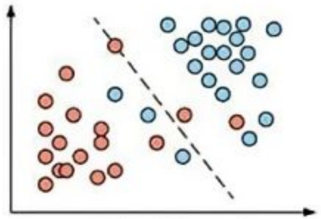
- Training: learn a function to map inputs to their labels
- Inference: predict the label of new unseen inputs

Unsupervised learning: unlabeled training data

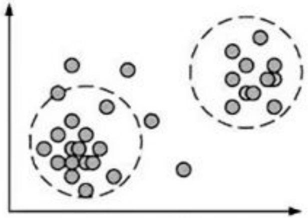
- Training: identify hidden structures in unlabeled data
- Inference: assign new inputs to the most appropriate group

Reinforcement learning: training data with global objective (actions)

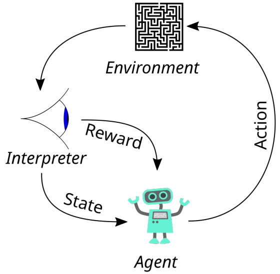
- Training: interact with an environment to maximize a cumulative reward
- Inference: make decisions to guide an agent towards an objective



Supervised Learning



Unsupervised Learning



Reinforcement Learning

Bordeaux INP Enseirb-Matmeca

Example: peaches vs. plums

Let's collect some data on fruits:

- Instances: peaches and plums
- Features: weight, size
- Labels: fruit name

Plums:

- Plum 1: [147.1g, 7.8cm]
- Plum 2: [143.2g, 7.7cm]
- Plum 3: [150.2g, 7.6cm]
- Plum 4: [155.7g, 7.7cm]
- Plum 5: [145.6g, 7.5cm]
- Plum 6: [150.8g, 8.0cm]

Peaches:

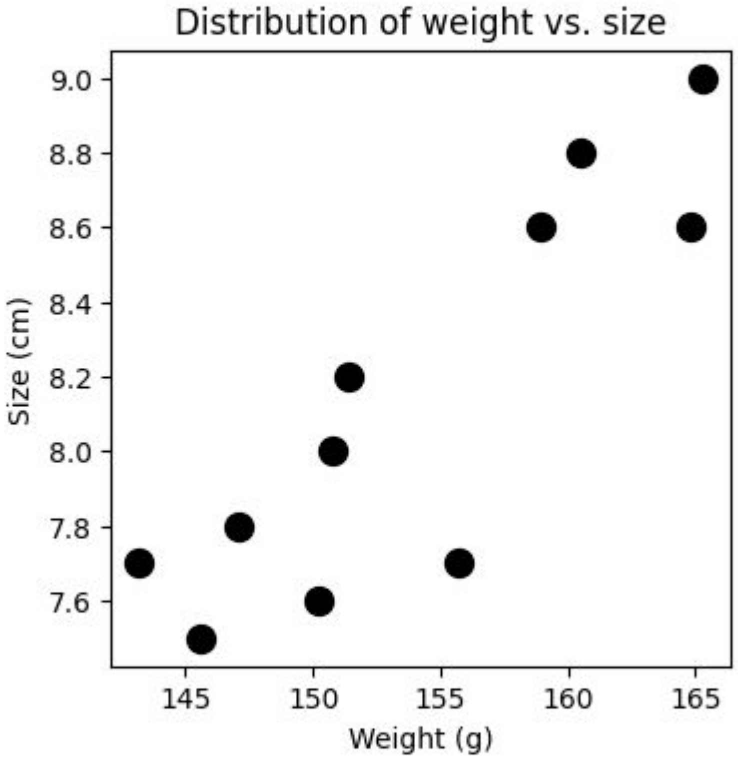
- Peach 1: [151.4g, 8.2cm]
- Peach 2: [160.5g, 8.8cm]
- Peach 3: [165.3g, 9.0cm]
- Peach 4: [164.8g, 8.6cm]
- Peach 5: [158.9g, 8.6cm]

Example: peaches vs. plums

2D plot:
Can we learn patterns from data?

2 natural groups of points
How to separate them?

Clustering



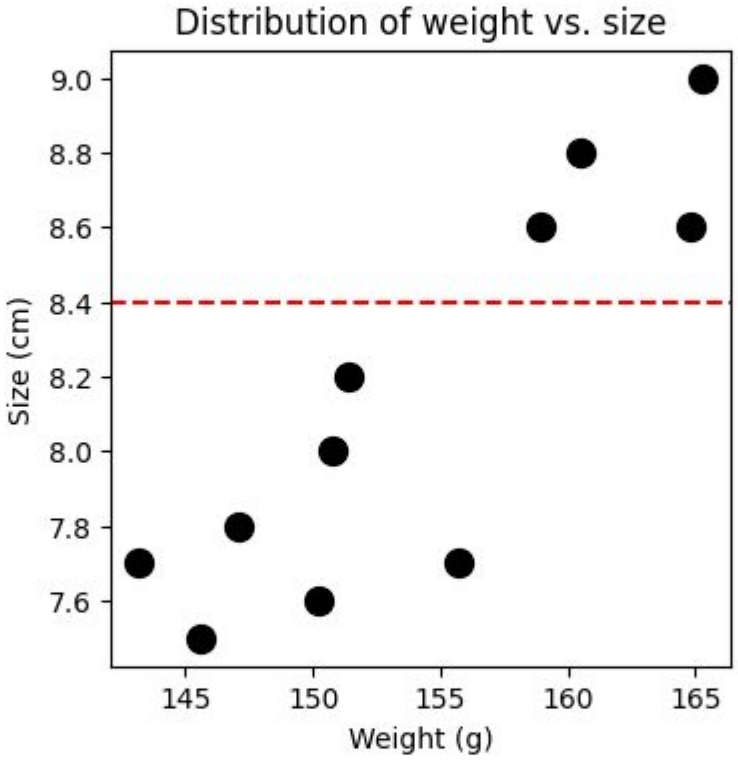
Example: peaches vs. plums

2D plot:
Can we learn patterns from data?

2 natural groups of points
How to separate them?

Clustering

1) Separation on size



Example: peaches vs. plums

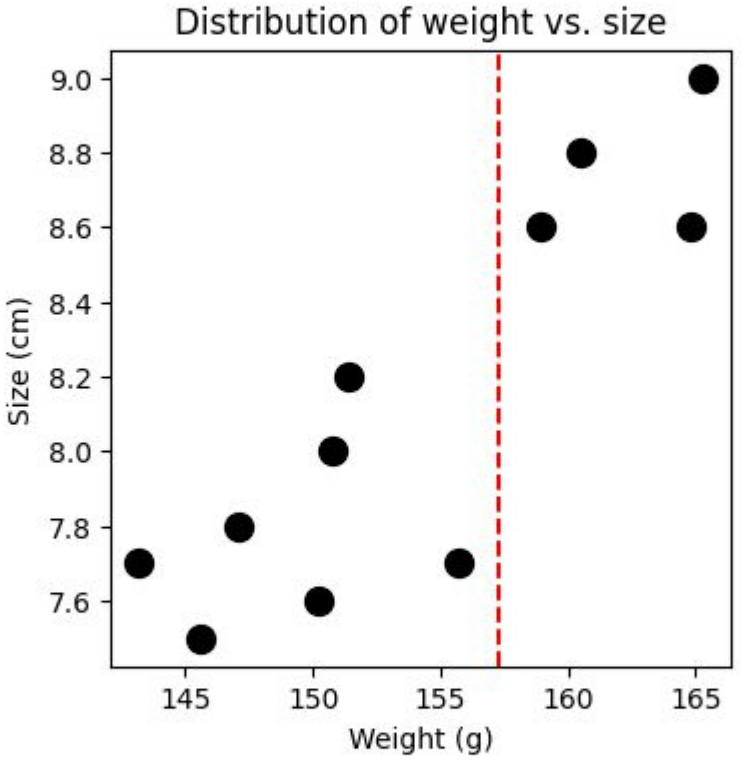
2D plot:
Can we learn patterns from data?

2 natural groups of points
How to separate them?

Clustering

1) Separation on size

2) Separation on weight



Example: peaches vs. plums

2D plot:
Can we learn patterns from data?

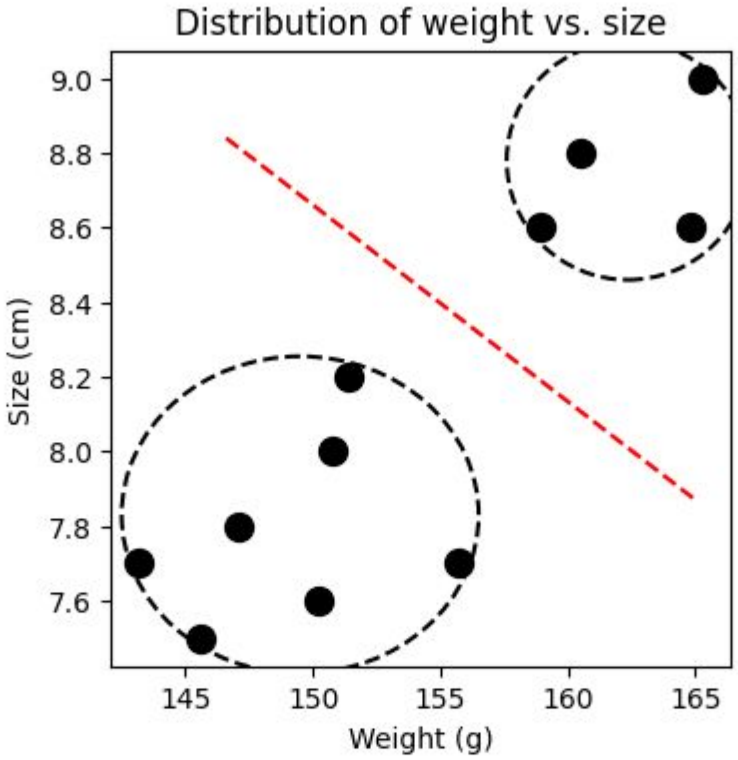
2 natural groups of points
How to separate them?

Clustering

1) Separation on size

2) Separation on weight

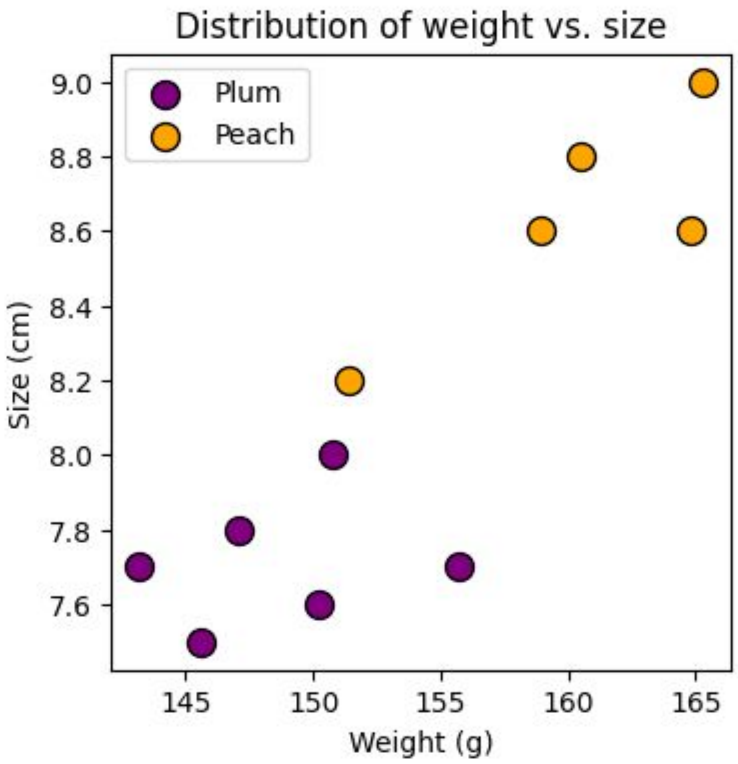
3) Cluster into two groups
using both attributes



Example: peaches vs. plums

2D plot with labels:
How to take advantage of knowing the labels to divide the two groups?

Classification

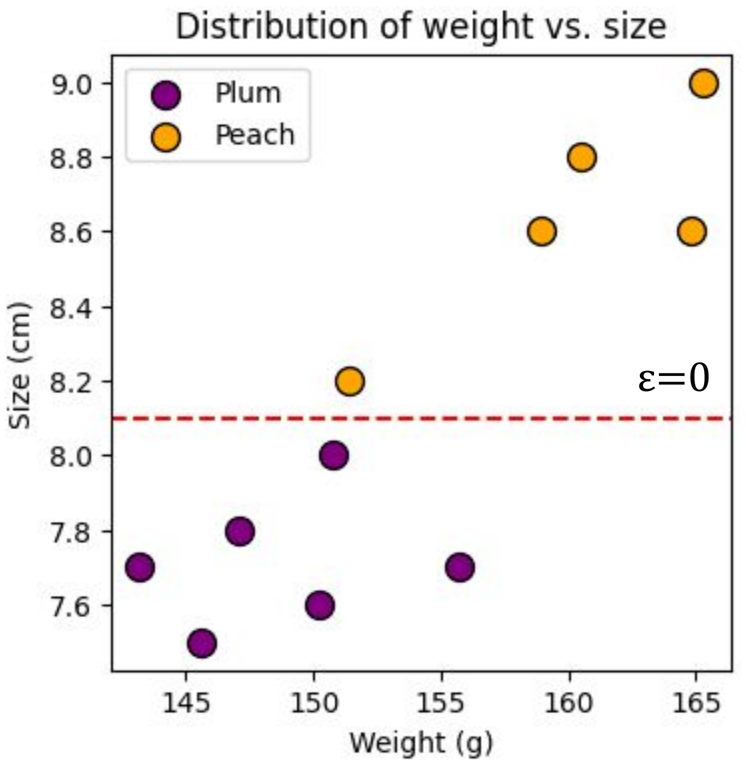


Example: peaches vs. plums

2D plot with labels:
How to take advantage of knowing the labels to divide the two groups?

Classification

1) Separation on size



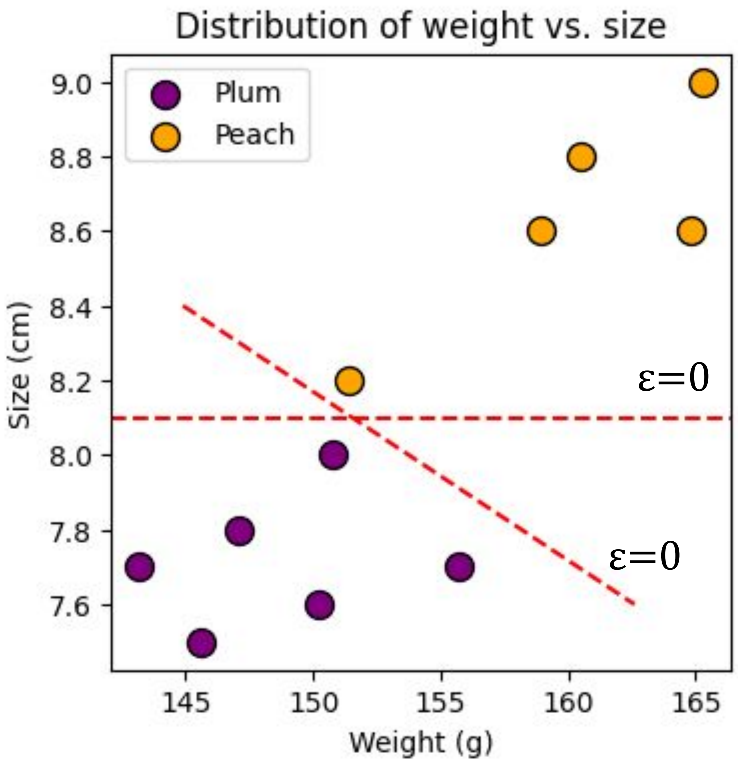
Example: peaches vs. plums

2D plot with labels:
How to take advantage of knowing the labels to divide the two groups?

Classification

1) Separation on size

2) Separation using both attributes



Example: peaches vs. plums

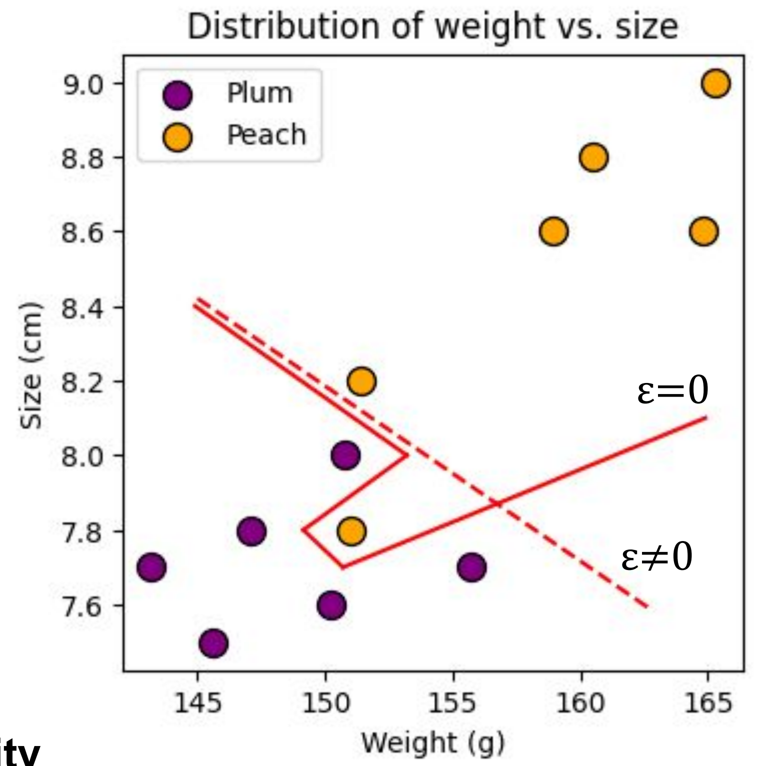
A new data point is added.
Which separator do we choose?

1) Dashed line: simple separator
   with errors

2) Solid line: complex separator
   without error

Option 2 performs well on training data
But it may fail to generalize to new data

Avoid **overfitting:**
Trade-off between **accuracy** and **simplicity**



Machine Learning: learning and generalizing

Machine Learning consists in learning from examples:

- **Training data**: a set of examples (peaches, plums, etc.) **labeled or not**
- **Features**: descriptors that describe each example (weight, size, etc.)
- **Distance** measure: quantifies “similarity” between examples
- **Objective function**: evaluates the quality of any solution

During training, we aim at optimizing the objective function (minimizing or maximizing)

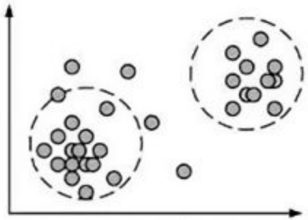
Machine Learning consists in generalizing on unseen data:

- **Test data**: a set of additional examples, unknown during training

During inference, we hope to make accurate predictions on our test data

Unsupervised Learning

Unsupervised learning



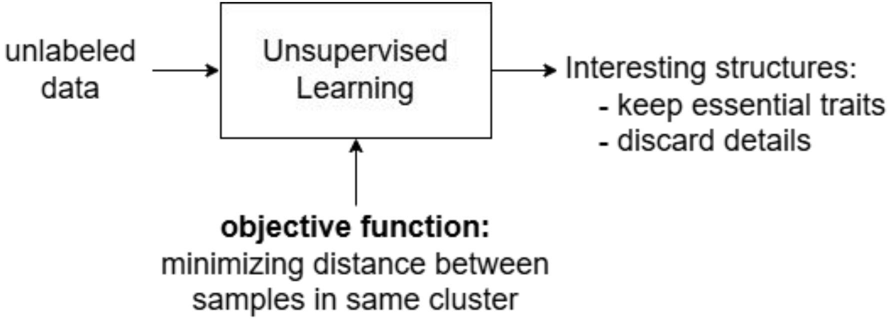
Unsupervised Learning

Unsupervised learning:

- **Unlabeled** training data
- Goal: discover hidden structures in the data

Clustering:

- Group examples that are "close" together
- Assign the same label to all data points in each group
- Relies heavily on the choice of a **distance metric**
- The distance metric often matters more than the clustering algorithm itself.



Distance metric

Distances are used to measure the **similarity** or **dissimilarity** between two data points

Training data:

- n instances
- p features

$$X = \begin{pmatrix} x_1^1 & \dots & x_1^p \\ \vdots & \ddots & \vdots \\ x_n^1 & \dots & x_n^p \end{pmatrix}$$

Popular distance: **Minkowski**

- d=1: Manhattan distance
- d=2: Euclidean distance

$$d(x_i, x_j) = \sqrt[q]{|x_i^1 - x_j^1|^q + \ldots + |x_i^p - x_j^p|^q}$$

Or weighted distance, …

Clustering

K clusters $\{S_k\}$ with associate centroïds $\{C_k\}$.
Goal: find the set S of clusters that minimize the **within-cluster** distance

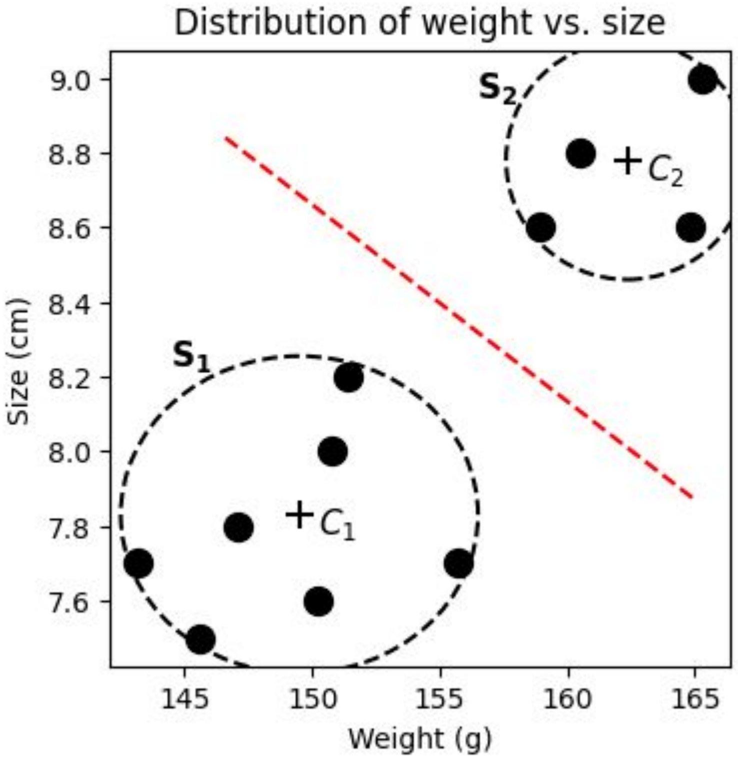
$$\arg\min_S \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in S_k} d(\mathbf{x}_i, C_k)$$

Given centroids $\{C_k\}$, associate each data point to the closest cluster:

$$\forall i, x_i \in S_k \;\; \texttt{where} \;\; k = \arg\min_j d(\mathbf{x}_i, C_j)$$

Given memberships of our data, update centroïds:

$$\forall k, C_k = \frac{1}{|S_k|} \sum_{x_i \in S_k} x_i$$



K-means Clustering

K-means algorithm

1) Start with a random guess of cluster centers

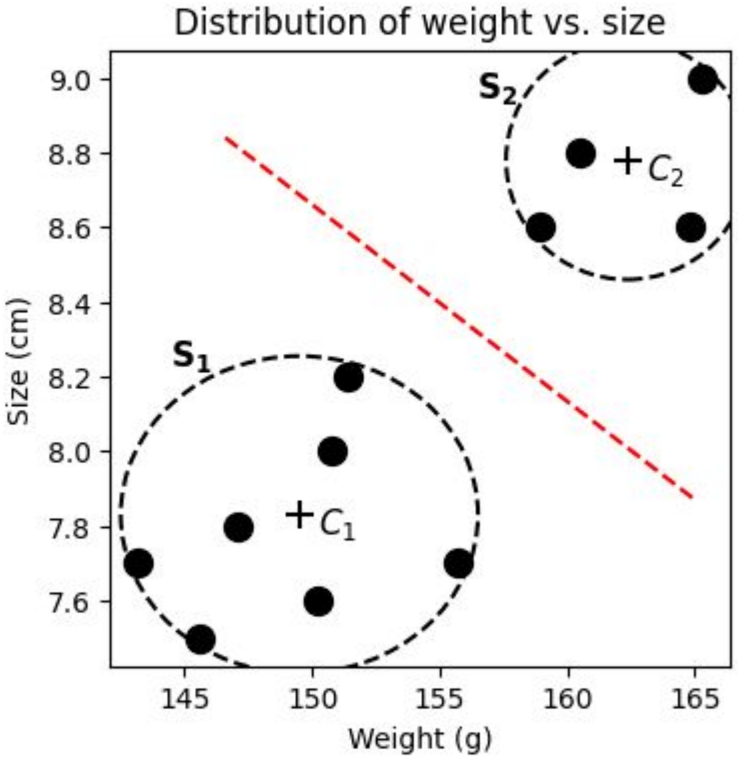
2) Determine the membership of each data point

3) Adjust the cluster centers

Stopping criterion:

- Number of iterations
- Clustering quality criterion $\rightarrow \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in S_k} d(\mathbf{x}_i, C_k)$
- Evolution of the quality

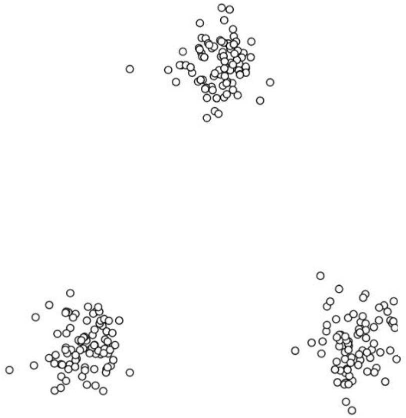
In practice: we stop when $\{C_j\}^{(t+1)} = \{C_j\}^{(t)}$



K-means Clustering

K-means algorithm

1) Set K the number of clusters: here K=3

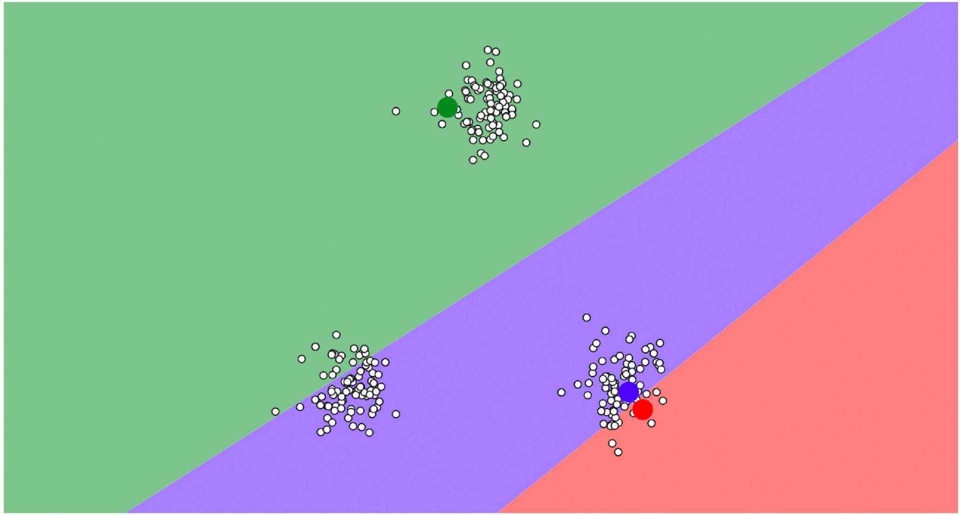


Harris, 2014

K-means Clustering

K-means algorithm

1) Set K the number of clusters: here K=3
2) Start with a random guess of cluster centers: select K random points

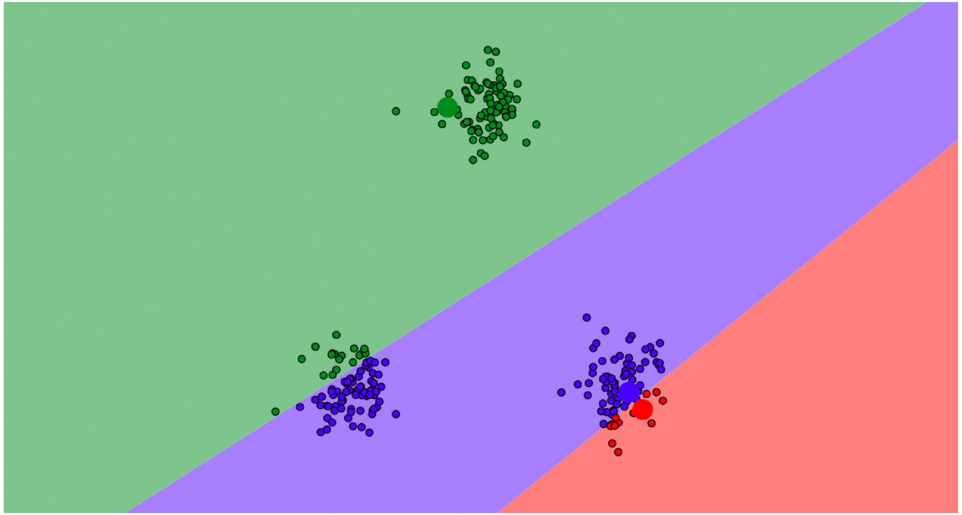


Harris, 2014

K-means Clustering

K-means algorithm

1) Set K the number of clusters: here K=3
2) Start with a random guess of cluster centers: select K random points
3) Associate each data point to its closest cluster

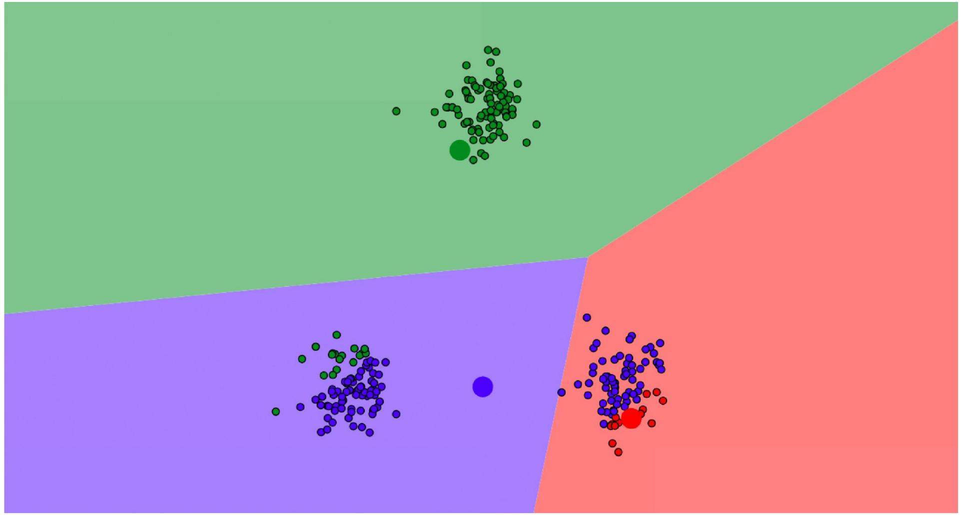


Harris, 2014

K-means Clustering

K-means algorithm

1) Set K the number of clusters: here K=3
2) Start with a random guess of cluster centers: select K random points
3) Associate each data point to its closest cluster
4) Adjust centroïds by computing their new mean

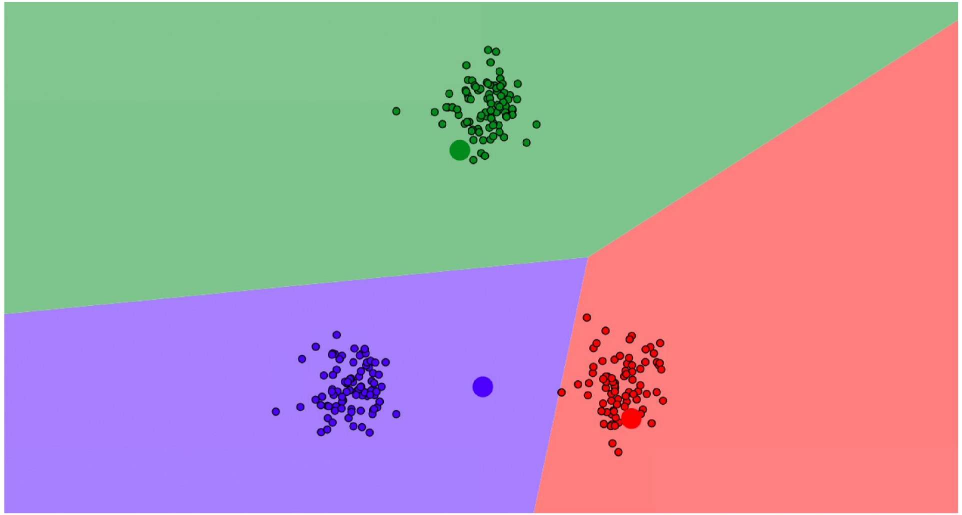


Harris, 2014

K-means Clustering

K-means algorithm

1) Set K the number of clusters: here K=3
2) Start with a random guess of cluster centers: select K random points
3) Associate each data point to its closest cluster
4) Adjust centroïds by computing their new mean

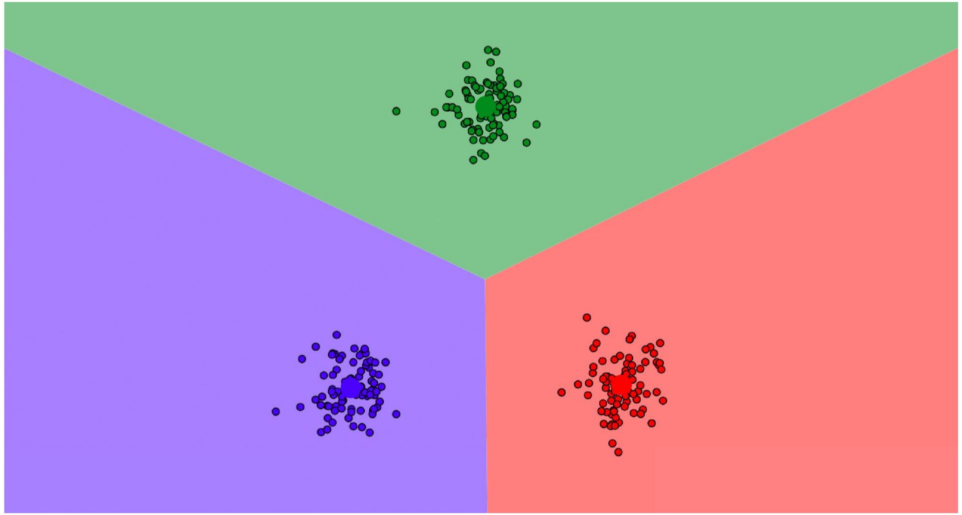


Harris, 2014

K-means Clustering

K-means algorithm

1) Set K the number of clusters: here K=3
2) Start with a random guess of cluster centers: select K random points
3) Associate each data point to its closest cluster
4) Adjust centroïds by computing their new mean



Harris, 2014

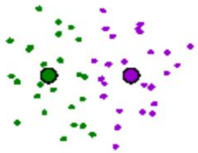
K-means Clustering

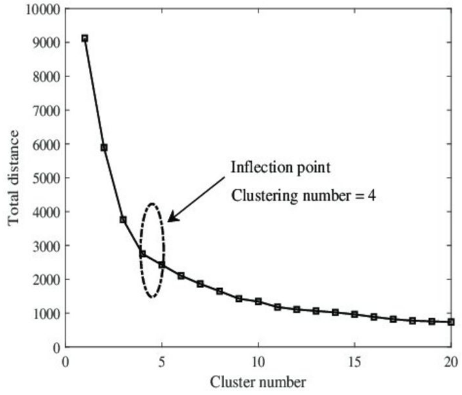
Strengths:

- Relatively efficient algorithm: O(nkt) with n data points, k clusters and t iterations
- Find at least a local optimum

Weaknesses:

- Sensitive to initial values
- Applicable only when the mean is defined (qualitative data?)
- Number of clusters K needs to be fixed in advance: elbow method





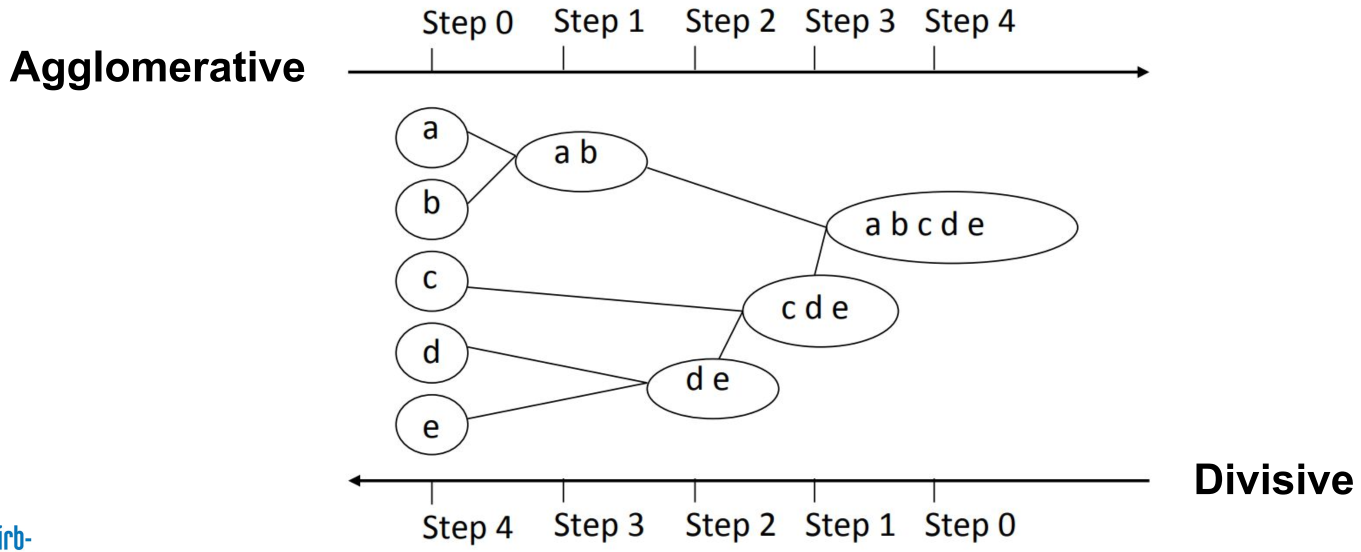
Variants:

- K-medoids, K-modes, ect.

Hierarchical Clustering

Hierarchical Clustering:

- Uses distances between data points as clustering criteria
- Does not require to determine a number of clusters
- Agglomerative vs. divisive



Hierarchical Clustering

Agglomerative Clustering

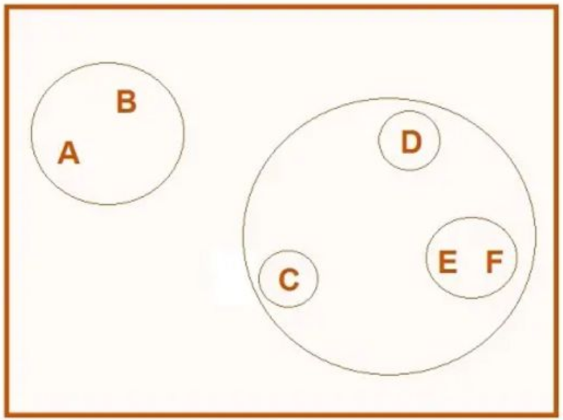
- Start with each point as individual cluster
- At each step, merge the closest pair
- Stop when only one cluster remains

Divisive Clustering

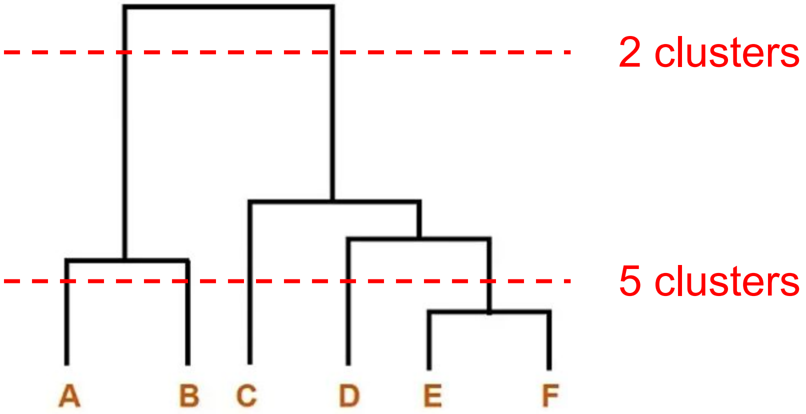
- Start with one cluster
- At each step, split a cluster
- Stop when each cluster contains a single point

Hierarchical Clustering

How many clusters do we want?



Clustering



Dendrogram

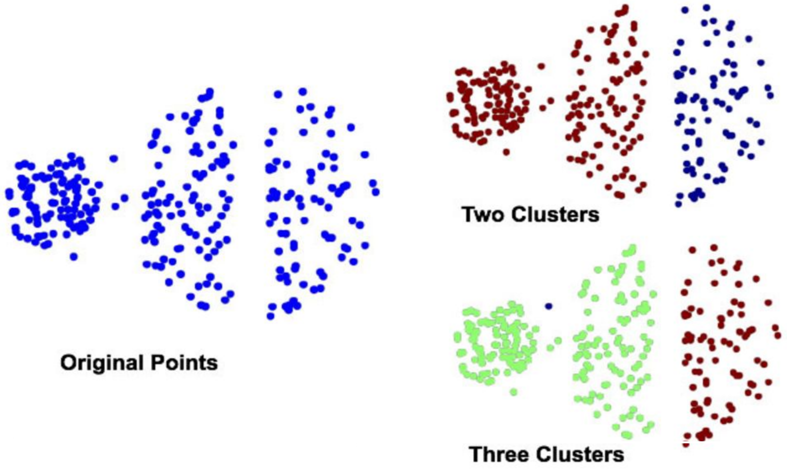
Hierarchical Clustering

Distance between two clusters?

1) **Single-link** distance: minimum distance between any points in S_i and S_j

$$d_{sl}(S_i, S_j) = min_{x,y}\{d(x,y)|x \in S_i, y \in S_j\}$$

Limitations: sensitive to noise and outliers



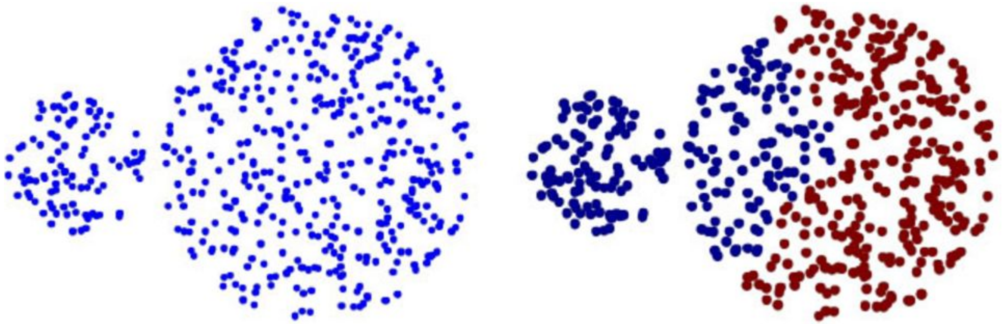
Hierarchical Clustering

Distance between two clusters?

2) **Complete-link** distance: maximum distance between any points in S_i and S_j

$$d_{sl}(S_i, S_j) = max_{x,y}\left\{d(x,y)|x \in S_i, y \in S_j\right\}$$

Limitations: all clusters tend to have same diameters → break large clusters



Original Points   Two Clusters

Hierarchical Clustering

Distance between two clusters?

3) **Group average** distance: average distance between any points in S_i and S_j

$$d_{ga}(S_i, S_j) = \frac{1}{|S_i| \times |S_j|} \sum_{x \in S_i, y \in S_j} d(x, y)$$

Compromise between single and complete links

4) **Centroid** distance

$$d_{centroids}(S_i, S_j) = d(C_i, C_j)$$

5) **Ward's** distance

$$d_{wards}(S_i, S_j) = \sum_{x \in S_i} (x - C_i)^2 + \sum_{x \in S_j} (x - C_j)^2 + \sum_{x \in S_i j} (x - C_i j)^2$$

Hierarchical Clustering

Hierarchical Clustering

- Deterministic: always the same answer
- Greedy algorithm: local optima rather than global optimum
- Very long to compute

Complexity

- Space: $O(n^2)$ to store the distance matrix
- Time: $O(n^3)$
    - n steps
    - at each step, the distance matrix of size n^2 must be updated

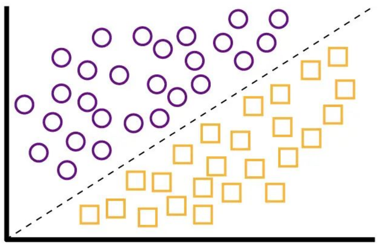
Supervised Learning

Supervised learning

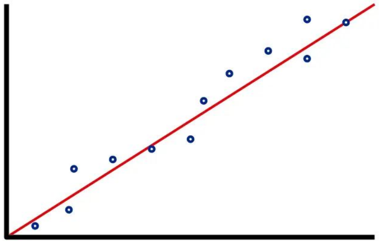
Supervised learning:
- **Labeled** training data
- Goal: learn a function to map inputs to their labels

Classification:
- Labels are **discrete** classes (ex: peach/plum)
- Predict the correct class for each input

Regression:
- Labels are **continuous** values of a dependent variable (ex: ordinate)
- Make the best prediction based on a set of independent variables (ex: abscissa)

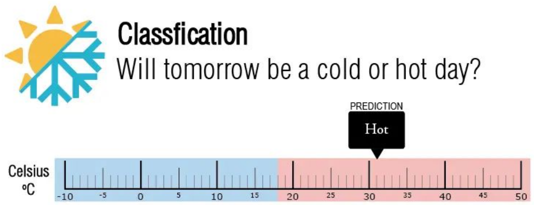


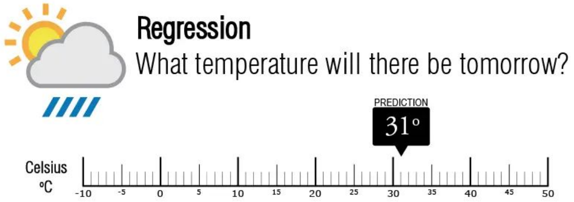
Classification



Regression

Ex:



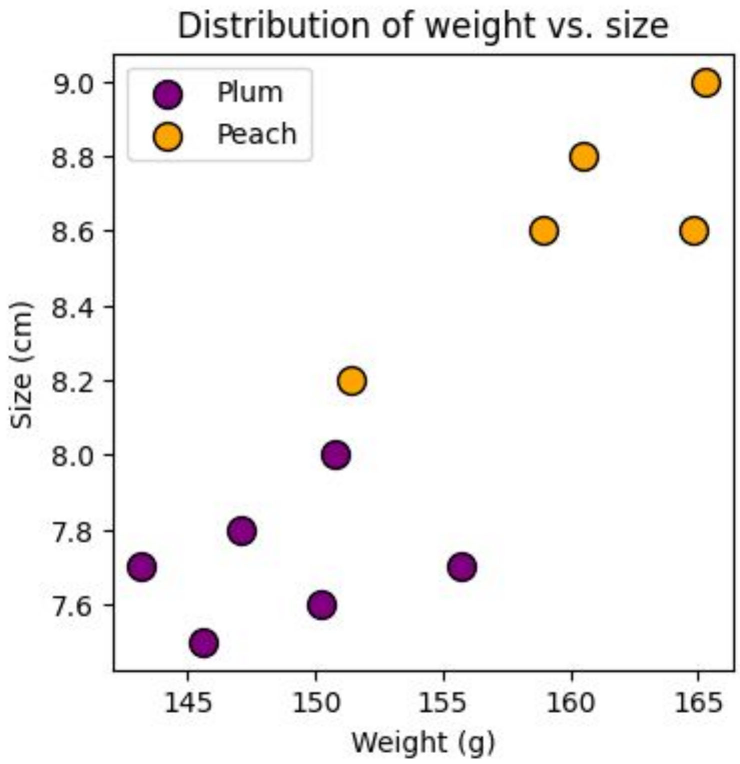


Classification

Our initial example: peaches vs. plums

- Features: weight, size
- Labels: Fruit name

How to divide the space?



Classification

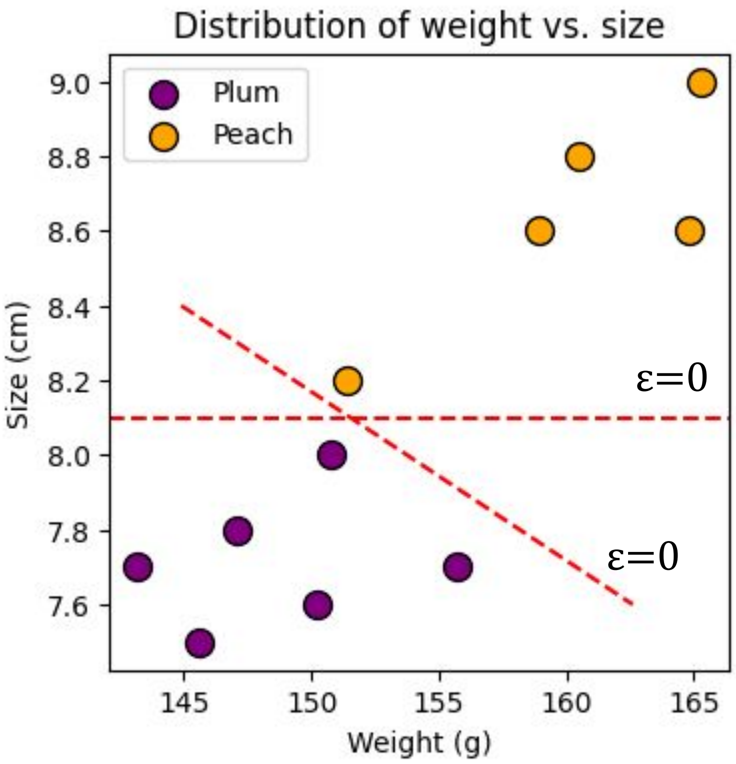
Our initial example: peaches vs. plums

- Features: weight, size
- Labels: Fruit name

How to divide the space?

Many perfect separators exist…
How to find the best separation?

- each separator defines a “street” (margin)
- Can we find the widest street?



Classification

Our initial example: peaches vs. plums

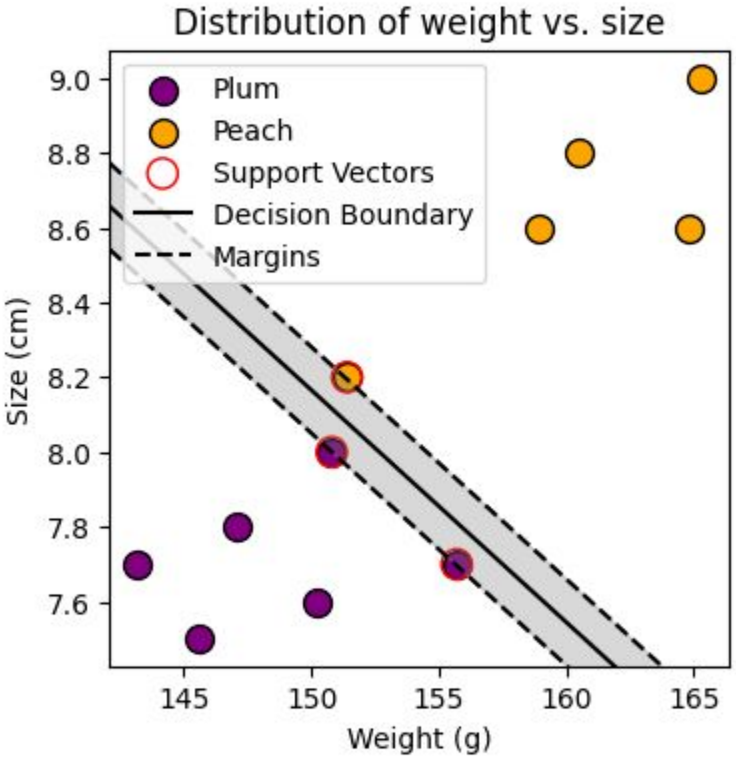
- Features: weight, size
- Labels: Fruit name

How to divide the space?

Many perfect separators exist…
How to find the best separation?

- each separator defines a “street” (margin)
- Can we find the **widest** street?

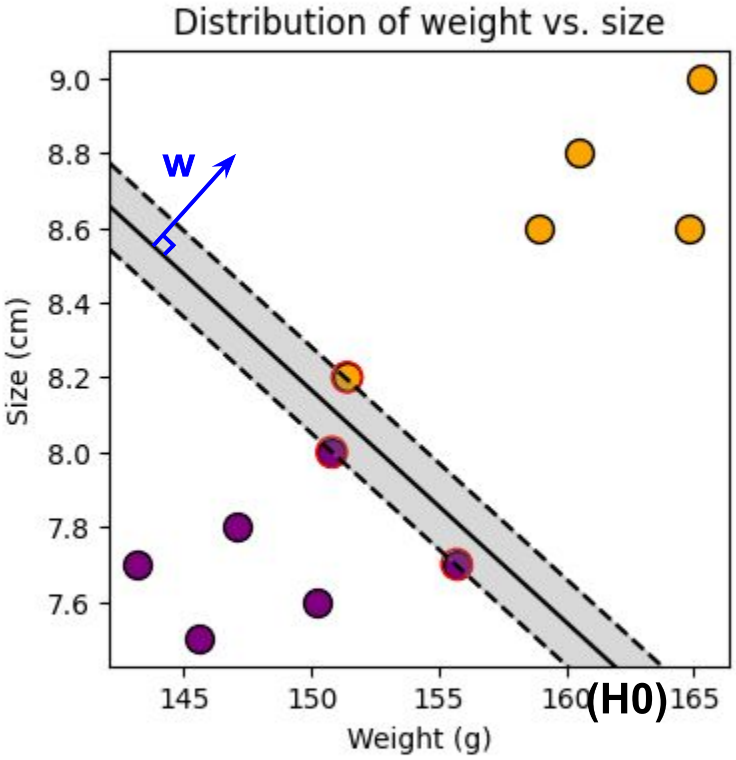
Support Vector Machines



Support Vector Machines

Let's consider **w** ⊥ (H0):

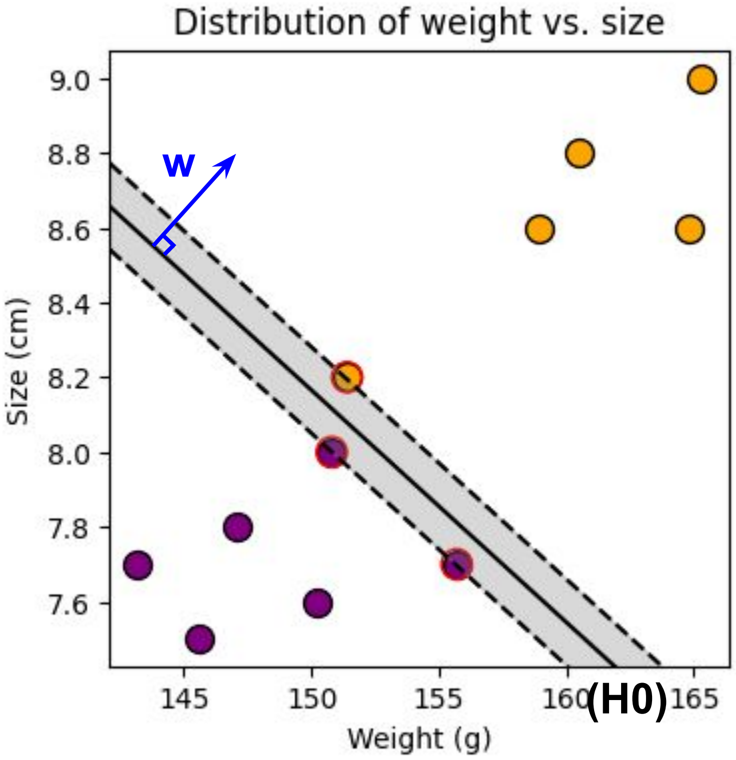
- For each point **x** in (H0): $\vec{w} \cdot \vec{x} = \texttt{cste}$



Support Vector Machines

Let's consider **w**⊥(H0):

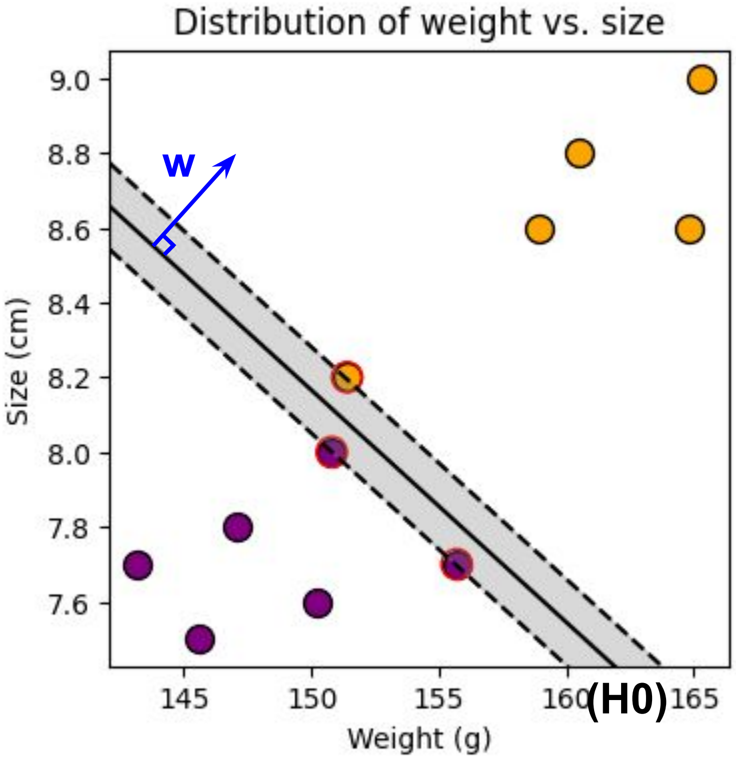
- For each point **x** in (H0): $\vec{w} \cdot \vec{x} = \texttt{cste}$
- So we can find b such as: $\vec{w} \cdot \vec{x} + b = 0$



Support Vector Machines

Let's consider **w** ⊥ (H0):

- For each point **x** in (H0): $\vec{w} \cdot \vec{x} = \texttt{cste}$
- So we can find b such as: $\vec{w} \cdot \vec{x} + b = 0$
- Thus: $\begin{cases} \vec{w} \cdot \vec{x} + b \geq 0 \Rightarrow \texttt{peach} \\ \vec{w} \cdot \vec{x} + b < 0 \Rightarrow \texttt{plum} \end{cases}$

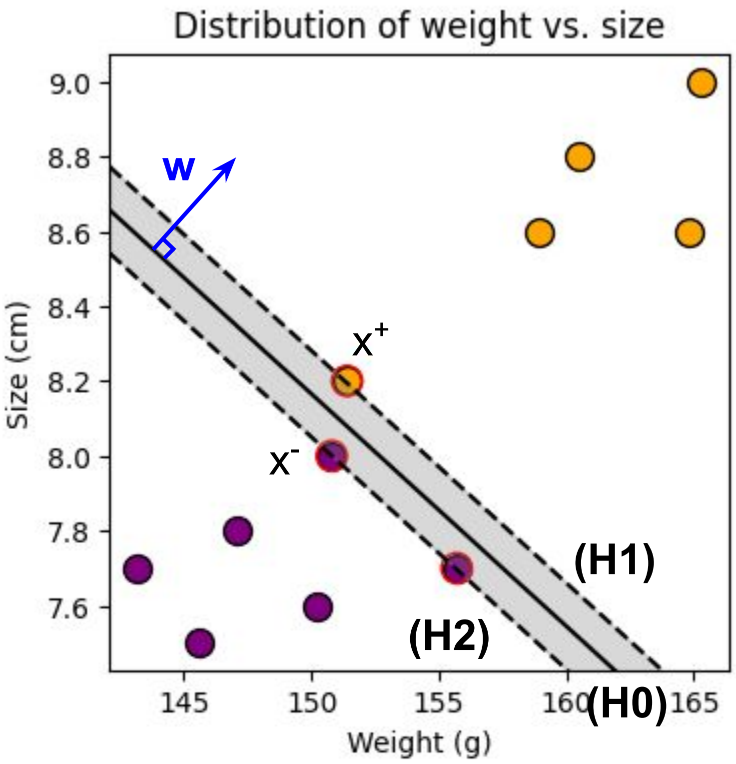


Support Vector Machines

Let's consider **w** ⊥ (H0):

- For each point **x** in (H0): $\vec{w} \cdot \vec{x} = \texttt{cste}$
- So we can find b such as: $\vec{w} \cdot \vec{x} + b = 0$
- Thus: $\begin{cases} \vec{w} \cdot \vec{x} + b \geq 0 \Rightarrow \texttt{peach} \\ \vec{w} \cdot \vec{x} + b < 0 \Rightarrow \texttt{plum} \end{cases}$

Equations of (H1) and (H2): $\begin{cases} \vec{w} \cdot \vec{x^+} + b = +k \\ \vec{w} \cdot \vec{x^-} + b = -k \end{cases}$



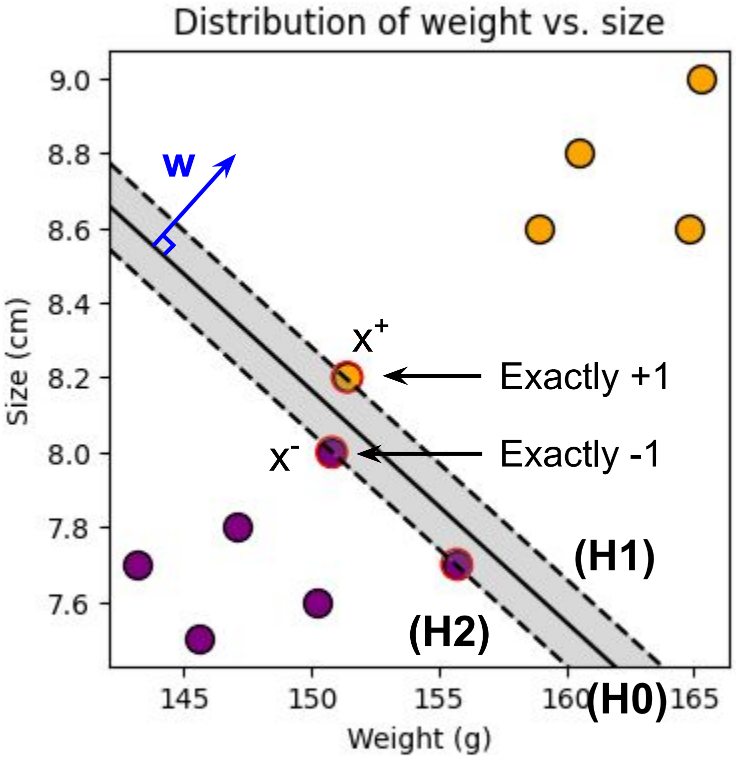
Support Vector Machines

Let's consider **w** ⊥ (H0):

- For each point **x** in (H0): $\vec{w} \cdot \vec{x} = \texttt{cste}$
- So we can find b such as: $\vec{w} \cdot \vec{x} + b = 0$
- Thus: $\begin{cases} \vec{w} \cdot \vec{x} + b \geq 0 \Rightarrow \texttt{peach} \\ \vec{w} \cdot \vec{x} + b < 0 \Rightarrow \texttt{plum} \end{cases}$

Equations of (H1) and (H2): $\begin{cases} \vec{w} \cdot \vec{x^+} + b = +k \\ \vec{w} \cdot \vec{x^-} + b = -k \end{cases}$

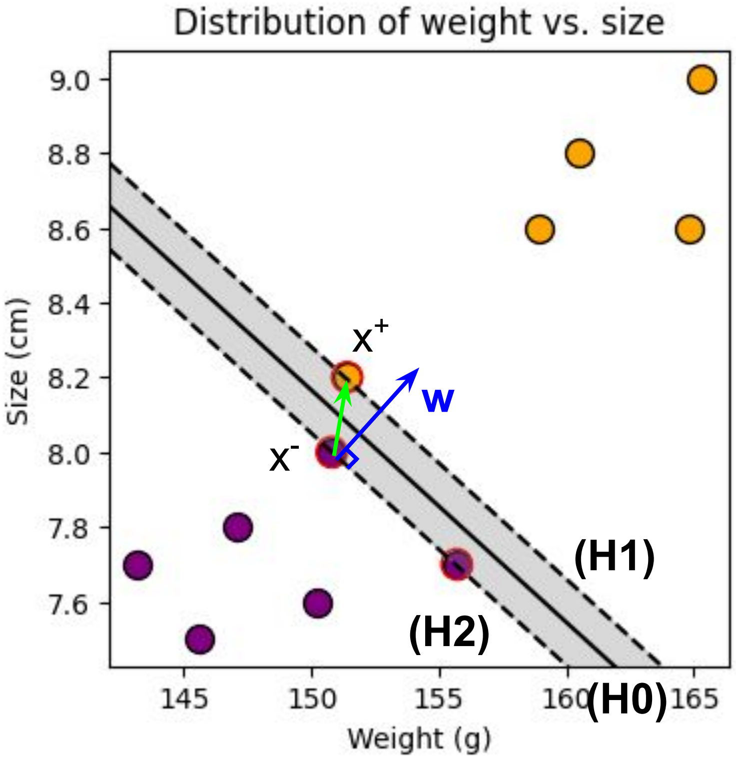
Rescaling **w** and b gives: $\begin{cases} \vec{w} \cdot \vec{x^+} + b = +1 \\ \vec{w} \cdot \vec{x^-} + b = -1 \end{cases}$



Support Vector Machines

SVM maximizes the width of the margin:

$$\texttt{width} = \frac{\vec{w}}{||\vec{w}||} \cdot (\vec{x^+} - \vec{x^-})$$



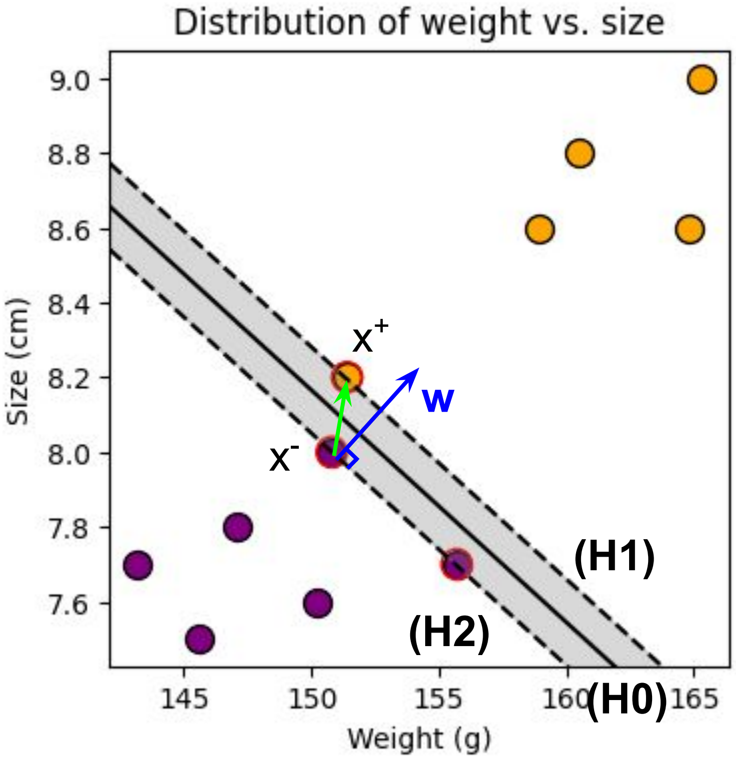
Support Vector Machines

SVM maximizes the width of the margin:

$$\texttt{width} = \frac{\vec{w}}{||\vec{w}||} \cdot (\vec{x^+} - \vec{x^-})$$

Also:

$$\left.\begin{array}{l}\vec{w} \cdot \vec{x^+} + b = +1 \\ \vec{w} \cdot \vec{x^-} + b = -1\end{array}\right\} \Rightarrow \vec{w} \cdot (\vec{x^+} - \vec{x^-}) = 2$$



Support Vector Machines

SVM maximizes the width of the margin:

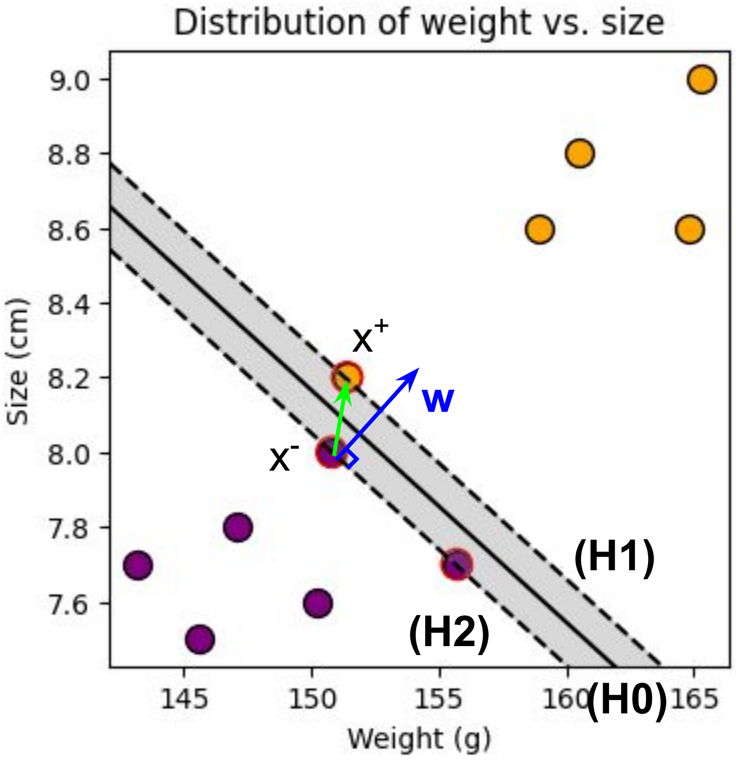
$$\texttt{width} = \frac{\vec{w}}{||\vec{w}||} \cdot (\vec{x^+} - \vec{x^-})$$

Also:

$$\left.\begin{aligned} \vec{w} \cdot \vec{x^+} + b = +1 \\ \vec{w} \cdot \vec{x^-} + b = -1 \end{aligned}\right\} \Rightarrow \vec{w} \cdot (\vec{x^+} - \vec{x^-}) = 2$$

Thus:

maximize → $\texttt{width} = \frac{2}{||\vec{w}||}$ ← minimize



Support Vector Machines

So the goal is to solve $\min ||\mathbf{w}||$ under +1/-1 (y_i) constraints:

- Constrained optimization problem

Support Vector Machines

So the goal is to solve $\min ||\mathbf{w}||$ under +1/-1 (y_i) constraints:

- Constrained optimization problem
- Solved via Lagrange Multiplier:

$$\mathcal{L}(\vec{w}, b) = \frac{\vec{w} \cdot \vec{w}}{2} - \sum_i \alpha_i [y_i(\vec{w} \cdot \vec{x}_i + b) - 1]$$

objective

constraints

Support Vector Machines

So the goal is to solve min ||**w**|| under +1/-1 (y_i) constraints:

- Constrained optimization problem
- Solved via Lagrange Multiplier:

$$\mathcal{L}(\vec{w}, b) = \underbrace{\frac{\vec{w} \cdot \vec{w}}{2}}_{\text{objective}} - \sum_i \underbrace{\alpha_i [y_i(\vec{w} \cdot \vec{x_i} + b) - 1]}_{\text{constraints}}$$

- Get global optimum: $\frac{\partial \mathcal{L}}{\partial \vec{w}} = 0 \Rightarrow \boxed{\vec{w} = \sum_i \alpha_i y_i \vec{x_i}}$ and also $\frac{\partial \mathcal{L}}{\partial \vec{b}} = 0$

Support Vector Machines

So the goal is to solve $\min ||\mathbf{w}||$ under +1/-1 (y_i) constraints:

- Constrained optimization problem
- Solved via Lagrange Multiplier:

$$\mathcal{L}(\vec{w}, b) = \frac{\vec{w} \cdot \vec{w}}{2} - \sum_i \alpha_i [y_i(\vec{w} \cdot \vec{x}_i + b) - 1]$$

objective

constraints

- Get global optimum: $\frac{\partial \mathcal{L}}{\partial \vec{w}} = 0 \Rightarrow \boxed{\vec{w} = \sum_i \alpha_i y_i \vec{x}_i}$ and also $\frac{\partial \mathcal{L}}{\partial \vec{b}} = 0$

Very beautiful result !!

Support Vector Machines

So the goal is to solve $\min ||\mathbf{w}||$ under +1/-1 (y_i) constraints:

- Constrained optimization problem
- Solved via Lagrange Multiplier:

$$\mathcal{L}(\vec{w}, b) = \underbrace{\frac{\vec{w} \cdot \vec{w}}{2}}_{\text{objective}} - \underbrace{\sum_i \alpha_i [y_i(\vec{w} \cdot \vec{x}_i + b) - 1]}_{\text{constraints}}$$

- Get global optimum: $\frac{\partial \mathcal{L}}{\partial \vec{w}} = 0 \Rightarrow \boxed{\vec{w} = \sum_i \alpha_i y_i \vec{x}_i}$ and also $\frac{\partial \mathcal{L}}{\partial \vec{b}} = 0$

Very beautiful result !!

- **w** linear combination of $\mathbf{x}_i$

Support Vector Machines

So the goal is to solve $\min ||\mathbf{w}||$ under +1/-1 (y_i) constraints:

- Constrained optimization problem
- Solved via Lagrange Multiplier:

$$\mathcal{L}(\vec{w}, b) = \underbrace{\frac{\vec{w}\cdot\vec{w}}{2}}_{\text{objective}} - \sum_i \underbrace{\alpha_i[y_i(\vec{w}\cdot\vec{x}_i + b) - 1]}_{\text{constraints}}$$

- Get global optimum: $\frac{\partial \mathcal{L}}{\partial \vec{w}} = 0 \Rightarrow \boxed{\vec{w} = \sum_i \alpha_i y_i \vec{x}_i}$ and also $\frac{\partial \mathcal{L}}{\partial \vec{b}} = 0$

Very beautiful result !!

- **w** linear combination of $\mathbf{x}_i$
- In practice, a lot of $\alpha_i = 0 \rightarrow$ so most points are useless

Support Vector Machines

So the goal is to solve $\min ||\mathbf{w}||$ under +1/-1 (y_i) constraints:

- Constrained optimization problem
- Solved via Lagrange Multiplier:

$$\mathcal{L}(\vec{w}, b) = \frac{\vec{w} \cdot \vec{w}}{2} - \sum_i \alpha_i [y_i(\vec{w} \cdot \vec{x}_i + b) - 1]$$

objective

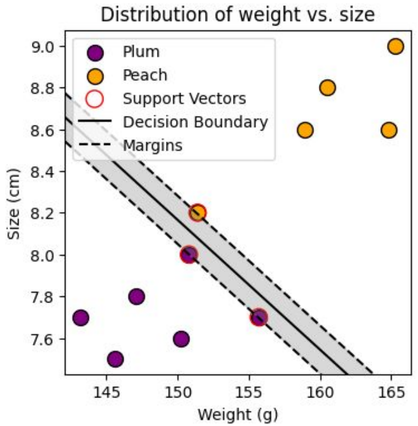
constraints

- Get global optimum: $\frac{\partial \mathcal{L}}{\partial \vec{w}} = 0 \Rightarrow \boxed{\vec{w} = \sum_i \alpha_i y_i \vec{x}_i}$ and also $\frac{\partial \mathcal{L}}{\partial \vec{b}} = 0$

Very beautiful result !!

- **w** linear combination of $\mathbf{x_i}$
- In practice, a lot of $\alpha_i = 0 \rightarrow$ so most points are useless
- Only points on the frontiers dictate the value of **w**

  → **Support Vectors**



Support Vector Machines

The decision rule becomes:

$$\left.\begin{array}{r}\vec{w}\cdot\vec{x}+b\\ \vec{w}=\sum\alpha_i y_i\vec{x_i}\end{array}\right\}\Rightarrow f(\vec{x})=\sum_i \alpha_i y_i \vec{x_i}\cdot\vec{x}+b$$

All we need is:

- Optimize α_i
- Only requires the dot product

Support Vector Machines

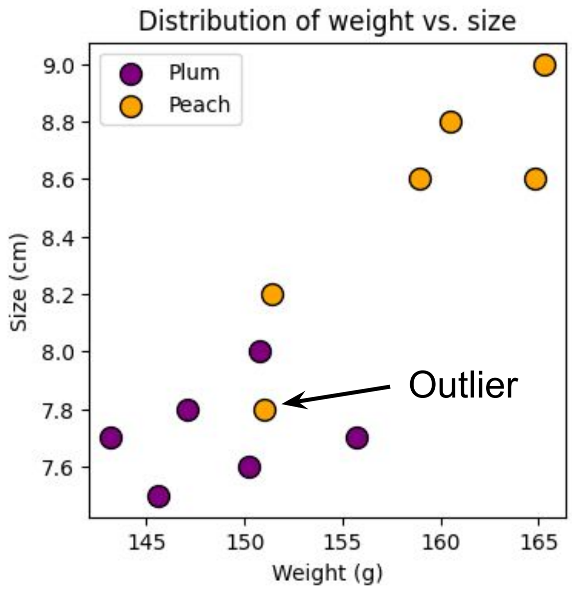
The decision rule becomes:

$$\left.\begin{array}{r}\vec{w}\cdot\vec{x}+b \\ \vec{w}=\sum \alpha_i y_i \vec{x}_i\end{array}\right\} \Rightarrow f(\vec{x}) = \sum_i \alpha_i y_i {\color{red}\vec{x}_i} \cdot {\color{red}\vec{x}} + b$$

All we need is:

- Optimize α_i
- Only requires the dot product

What if there are **outliers** in the data?



Support Vector Machines

The decision rule becomes:

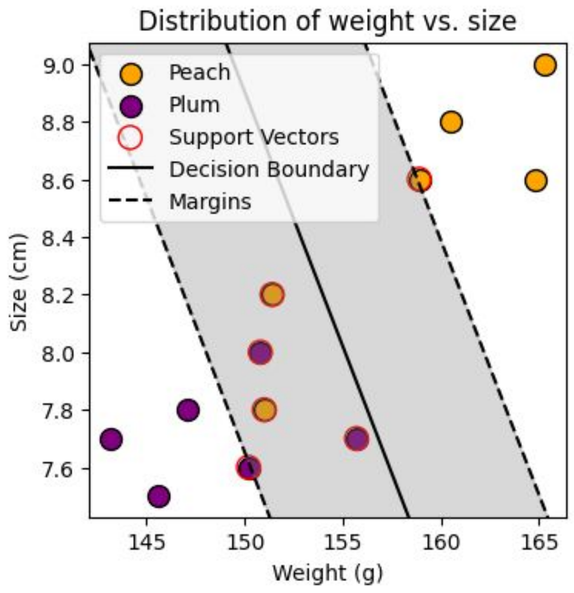
$$\left.\begin{aligned}\vec{w}\cdot\vec{x}+b\\ \vec{w}=\sum\alpha_i y_i\vec{x_i}\end{aligned}\right\}\Rightarrow f(\vec{x})=\sum_i \alpha_i y_i \color{red}{\vec{x_i}}\cdot\color{red}{\vec{x}}\color{black}{}+b$$

All we need is:

- Optimize α_i
- Only requires the dot product

What if there are **outliers** in the data?

- Soft margin SVM
- Constrained equations: $y_i(\vec{w}\cdot\vec{x_i}+b)\geq 1-\zeta_i$
- L1 regularization: penalizing large values of ζ_i
- Optimization function: $\dfrac{\vec{w}\cdot\vec{w}}{2}+\color{red}{C}\color{black}{}\sum_i \zeta_i$

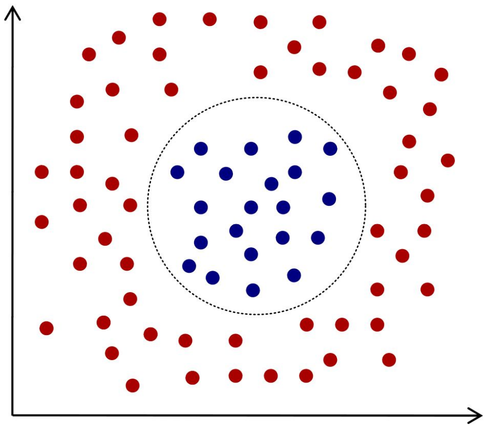


Controlling C → controlling classification error

Support Vector Machines

SVM finds the best **straight** line

What to do when the data is **non-linearly** separable?

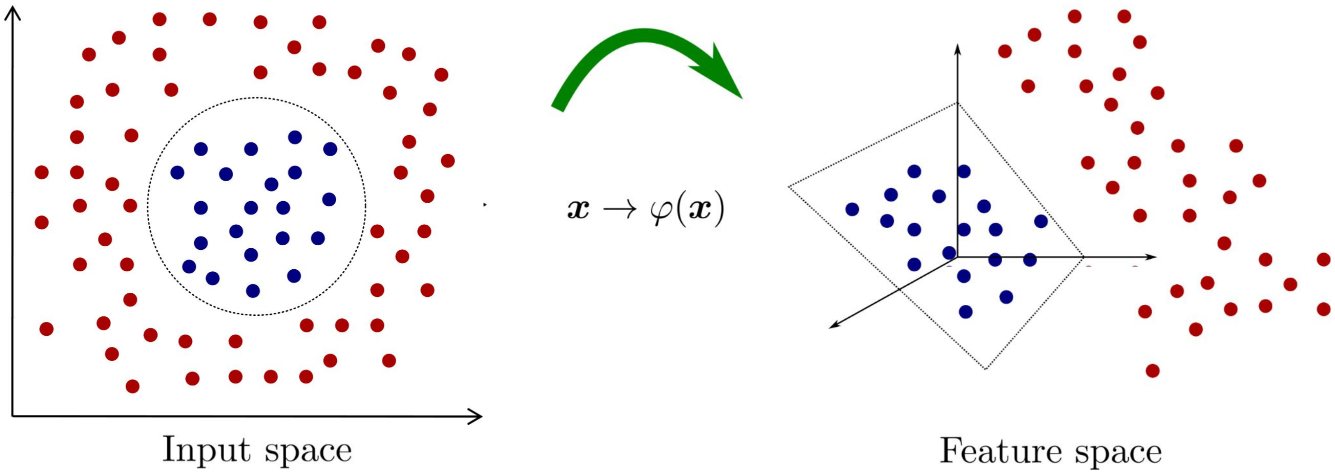


HOW?

Support Vector Machines

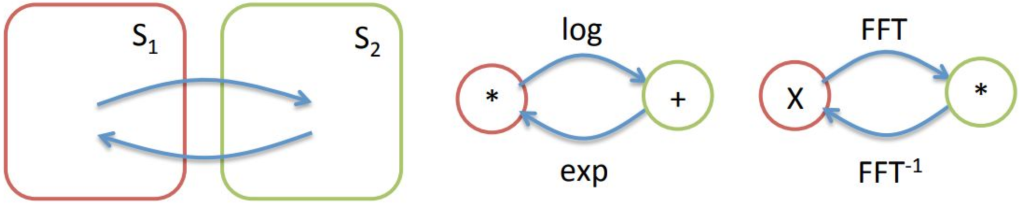
SVM finds the best **straight** line

What to do when the data is **non-linearly** separable?

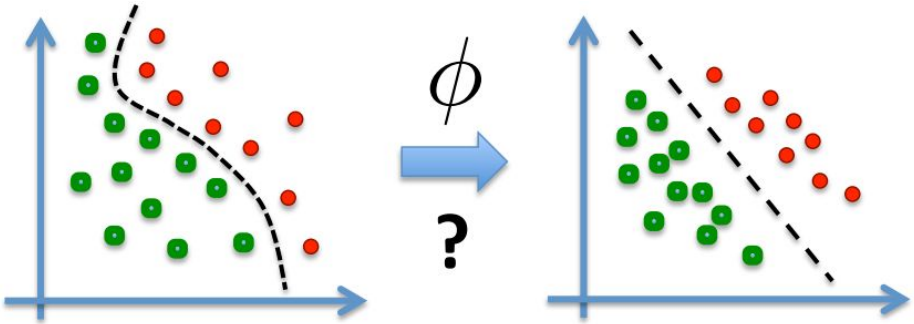


Support Vector Machines

- If the problem is hard in a given space S_1
- It is usual to go to a second space S_2 where it is easier to solve
- And then apply the inverse transformation (from S_2 to S_1)



- Same here: if we don't have a straight line in S_1, maybe we have one in S_2



Support Vector Machines

Limitations:
- Dimensionality of Φ can be very large
- High computational costs

Solution:
- No need to compute the transform Φ
- Only need to compute its dot product!
- **Kernel trick**: $k(x, y) = \phi(x) \cdot \phi(y)$
- New problem formulation:

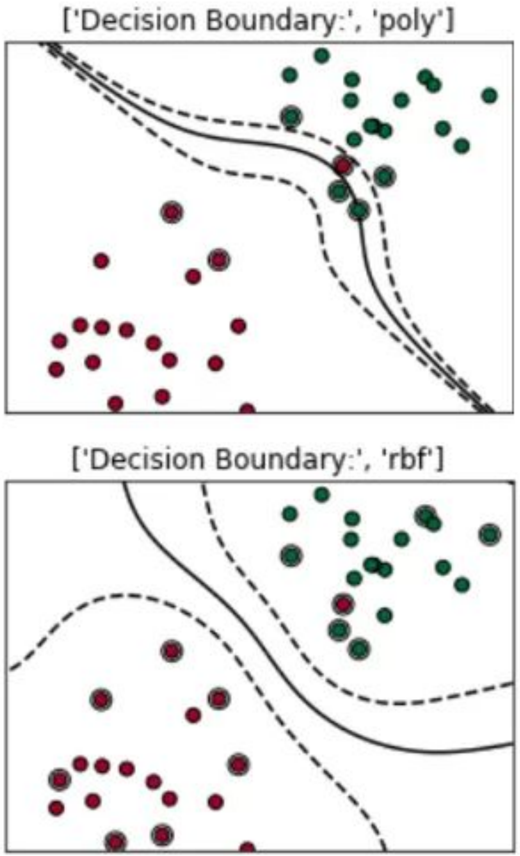
$$f(\vec{x}) = \sum_i \alpha_i y_i \vec{x_i} \cdot \vec{x} + b \Rightarrow f(\vec{x}) = \sum_i \alpha_i y_i \phi(\vec{x_i}) \cdot \phi(\vec{x}) + b$$

- Solved via SMO algorithm (Sequential Minimal Optimization)

Support Vector Machines

Most used kernel functions:

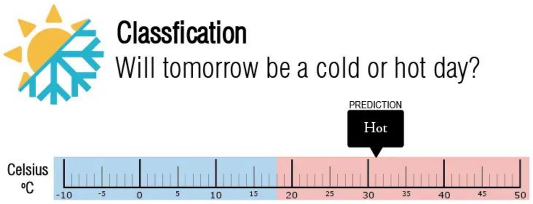
- **Polynomial**: $K(x,y) = (x^T y + 1)^d$
- **Gaussian**: $K(x,y) = \exp\left(-\psi(x-y)^2\right)$
- **Radial Basis**: $K(x,y) = \exp\left(\frac{-||x-y||^2}{2\sigma^2}\right)$
- **Sigmoid**: $K(x,y) = \tanh\left(kx^T y - \Theta\right)$

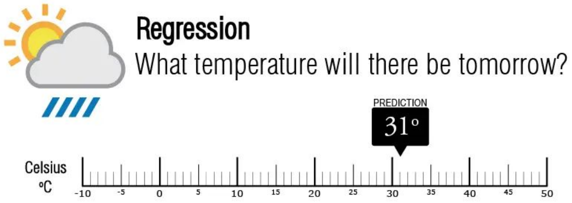


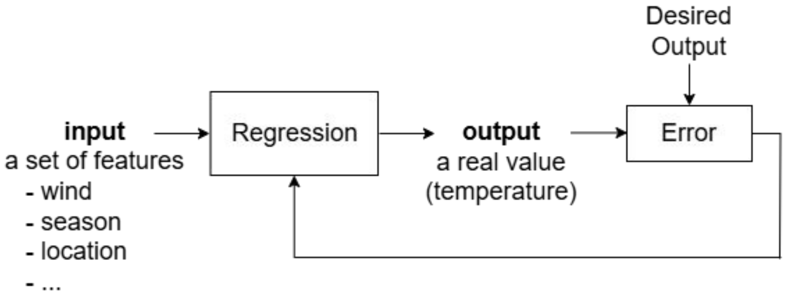
Regression

Regression:

- In classification we aim at predicting a discrete value (label)
- Here we aim at predicting a **real number** from a set of features



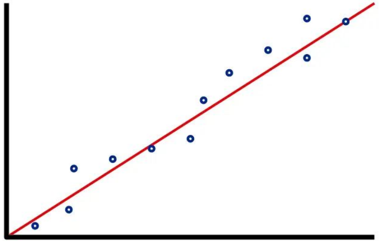




Regression

Objective: Learn a function f that predicts, for each input x, an output f(x,θ) that is closed to the desired value y

$$\underbrace{f}_{\text{regression}}(x, \underbrace{\theta}_{\text{parameters to learn}}) \approx y$$



Regression

Let’s assume a polynomial relationship between input and output:

$$f(x_i, \boldsymbol{w}) = w_0 + w_1 x_i + w_2 x_i^2 + \ldots + w_M x_i^M = \sum_{m=0}^{M} w_m x_i^m$$

Objective: find parameter **w**

Regression

Learning parameters:

- Loss function: quantifies the error between predicted and desired values to minimize

  Example: **Mean Square Error** $$\mathcal{L}(w) = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i, \theta))^2$$

Regression

Learning parameters:

- Loss function: quantifies the error between predicted and desired values to minimize

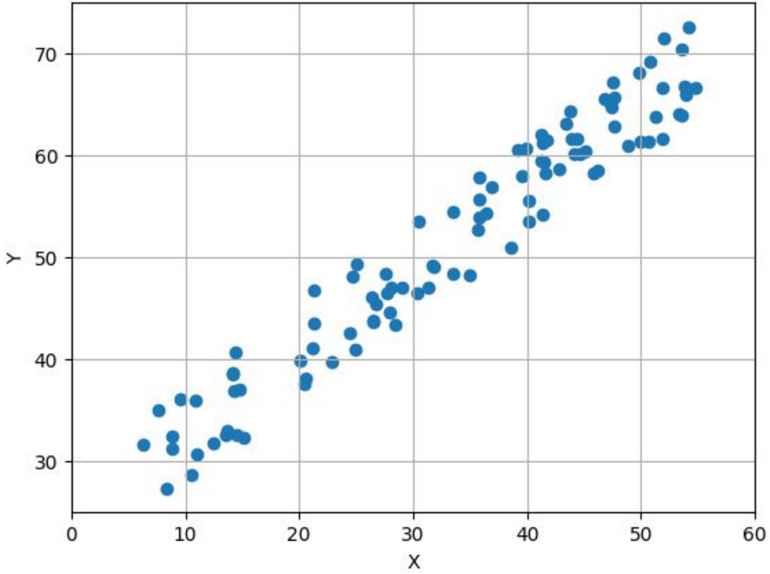
  Example: **Mean Square Error** $$\mathcal{L}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i, \theta))^2$$

- Iterative optimization: gradient descent
  - Compute the gradient of the loss function: $\frac{\partial \mathcal{L}}{\partial w_i}(\boldsymbol{w})$
  - Update parameters: $w_i \leftarrow w_i - \eta \frac{\partial \mathcal{L}}{\partial w_i}$

learning rate

Linear Regression

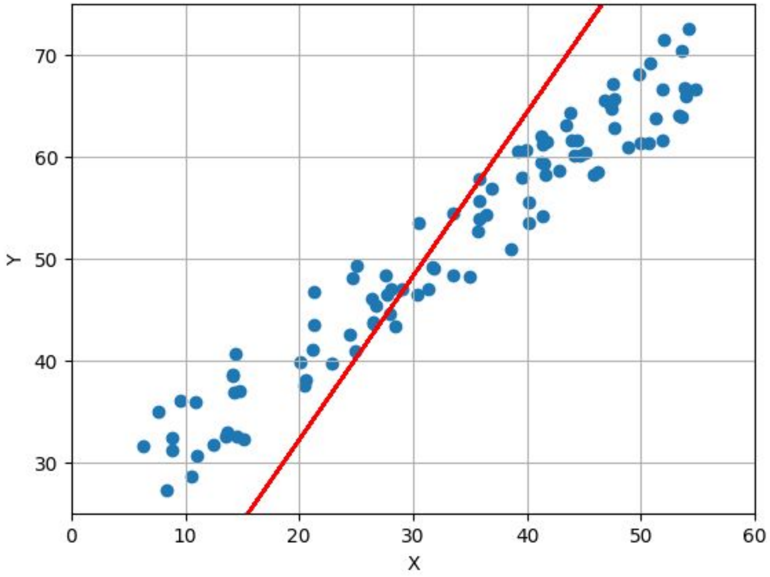
Input data: pairs of coordinates (x,y)
Model: $f(x) = ax + b$



Linear Regression

Input data: pairs of coordinates (x,y)
Model: $f(x) = ax + b$

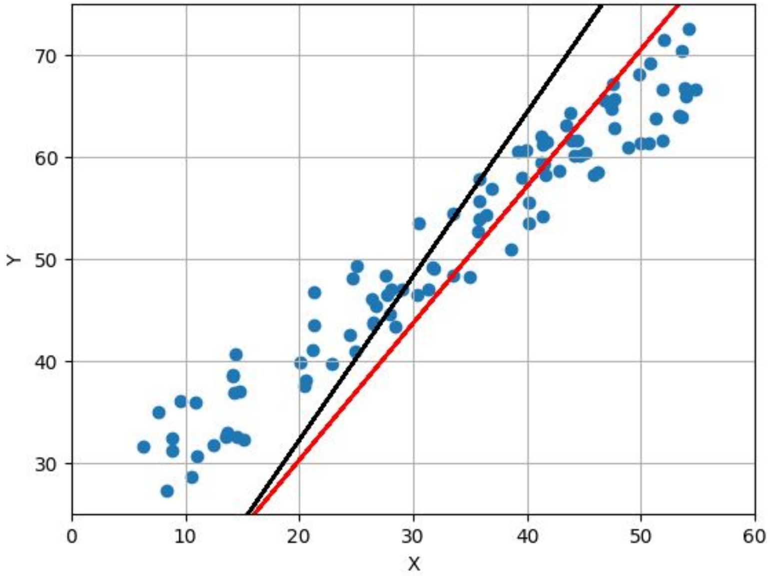
Learning:
epoch 1



Linear Regression

Input data: pairs of coordinates (x,y)
Model: $f(x) = ax + b$

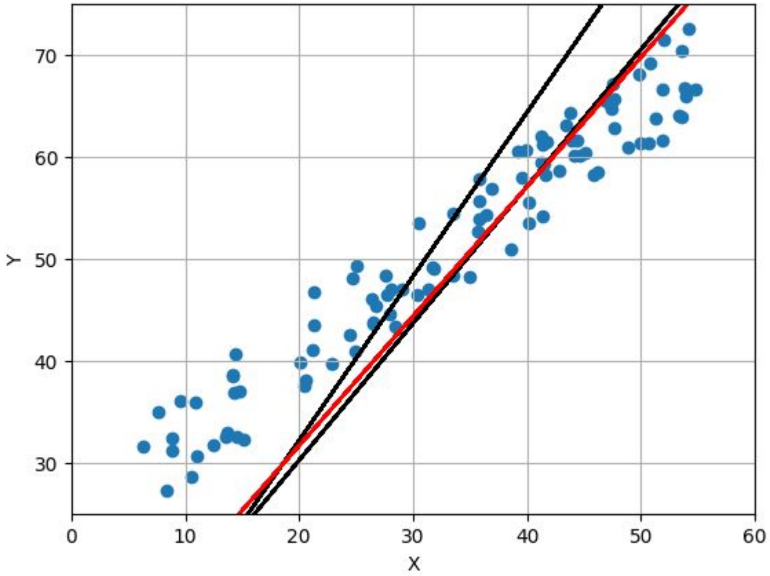
Learning:
epoch 2



Linear Regression

Input data: pairs of coordinates (x,y)
Model: $f(x) = ax + b$

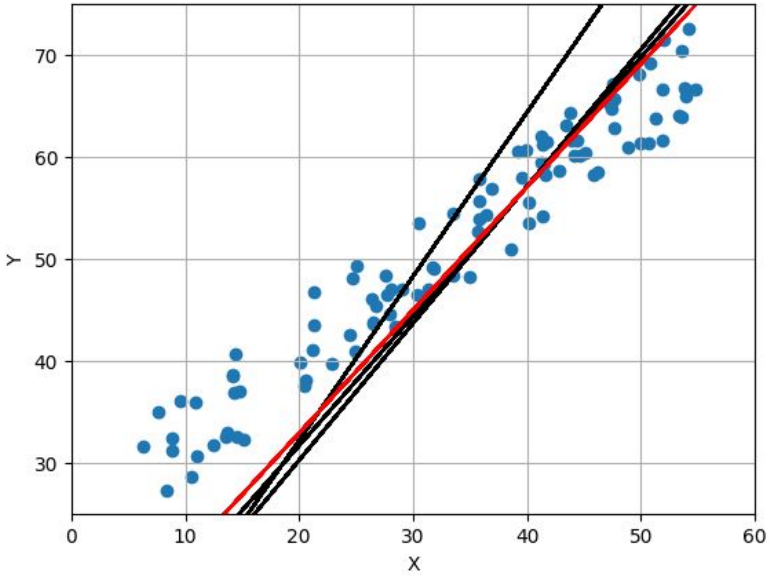
Learning:
epoch 3



Linear Regression

Input data: pairs of coordinates (x,y)
Model: $f(x) = ax + b$

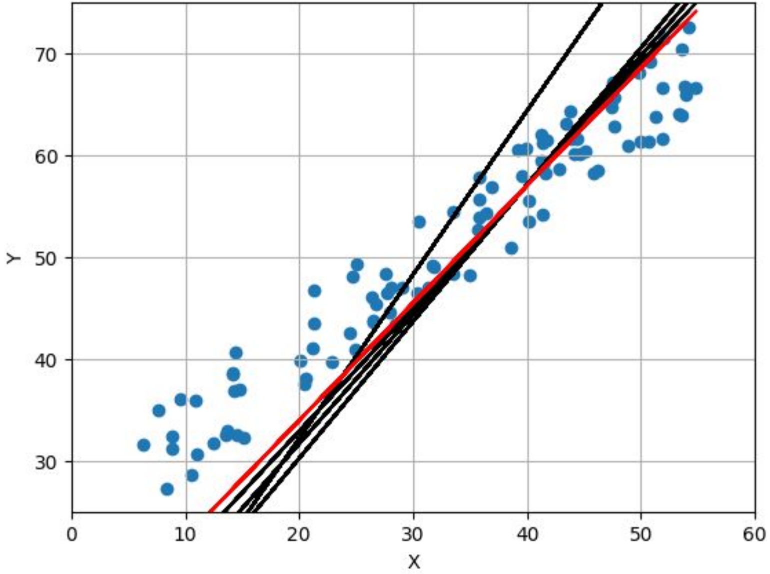
Learning:
epoch 4



Linear Regression

Input data: pairs of coordinates (x,y)
Model: $f(x) = ax + b$

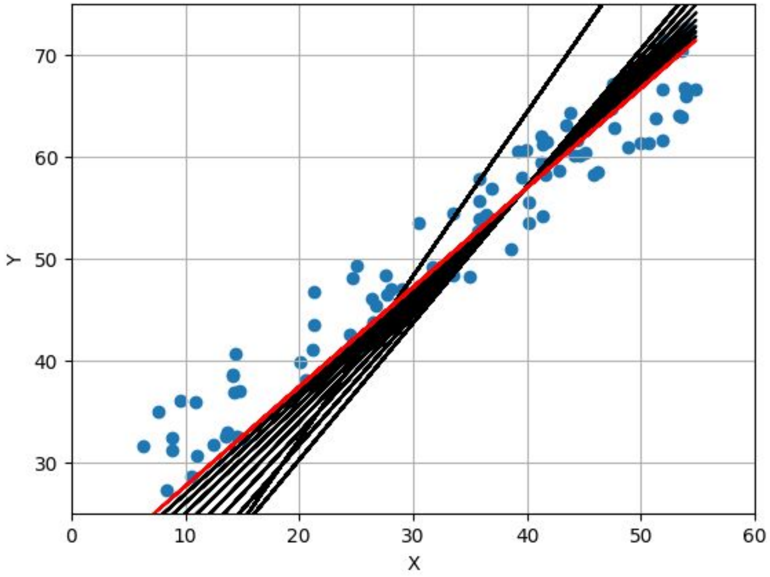
Learning:
epoch 5



Linear Regression

Input data: pairs of coordinates (x,y)
Model: $f(x) = ax + b$

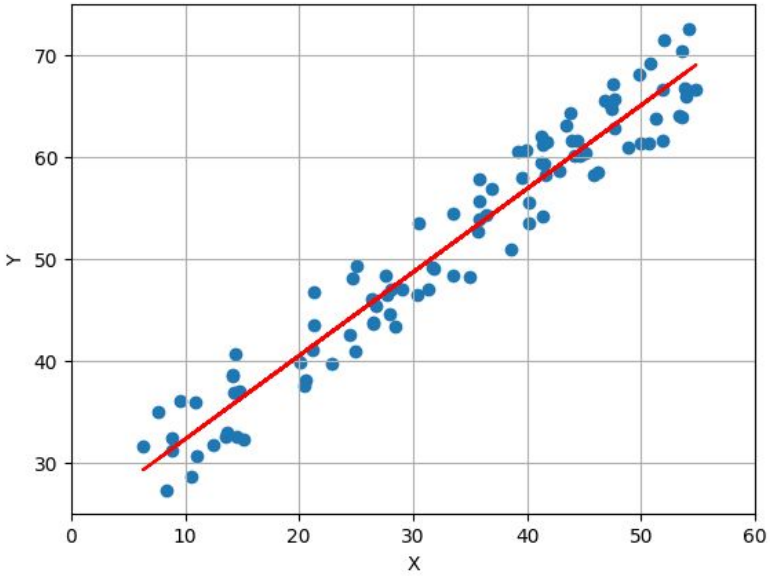
Learning:
epoch 10



Linear Regression

Input data: pairs of coordinates (x,y)
Model: $f(x) = ax + b$

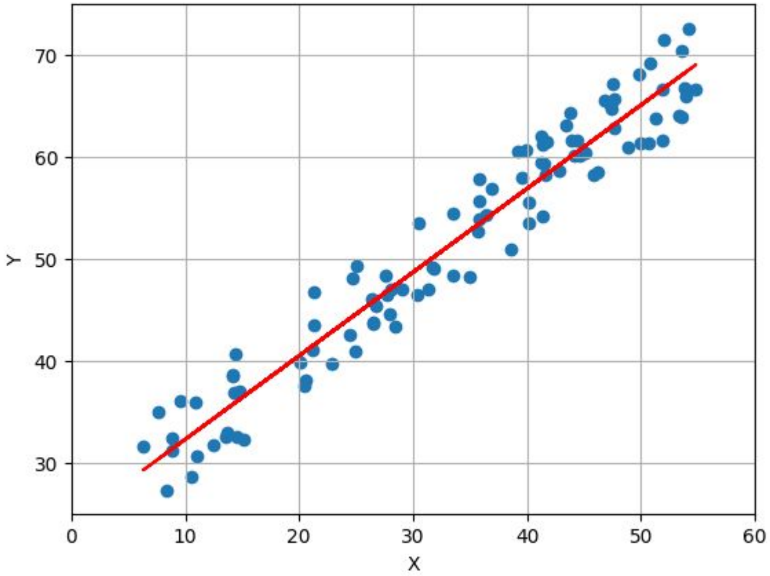
Learning:
epoch 100



Linear Regression

Input data: pairs of coordinates (x,y)
Model: $f(x) = ax + b$

Learning:
epoch 100



Deep Learning reuse the same methodology but with extremely nonlinear models

Conclusion

Conclusion

Methods

- Unsupervised Learning: K-means clustering, hierarchical clustering
- Supervised Learning: SVM, regression

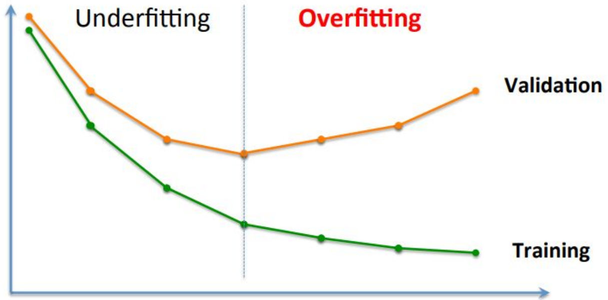
Conclusion

Methods

- Unsupervised Learning: K-means clustering, hierarchical clustering
- Supervised Learning: SVM, regression

Generalization and performance evaluation

- All methods are applied on a training set
- How do they perform on unseen data?
- Split dataset into train/dev/test sets to avoid **overfitting**



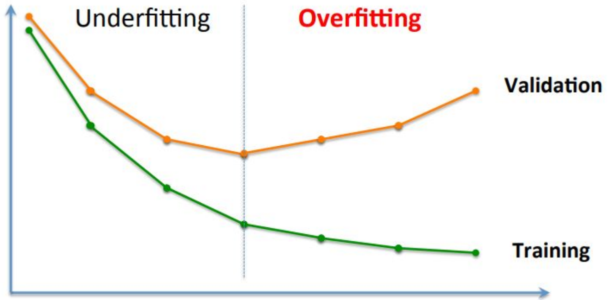
Conclusion

Methods

- Unsupervised Learning: K-means clustering, hierarchical clustering
- Supervised Learning: SVM, regression

Generalization and performance evaluation

- All methods are applied on a training set
- How do they perform on unseen data?
- Split dataset into train/dev/test sets to avoid **overfitting**



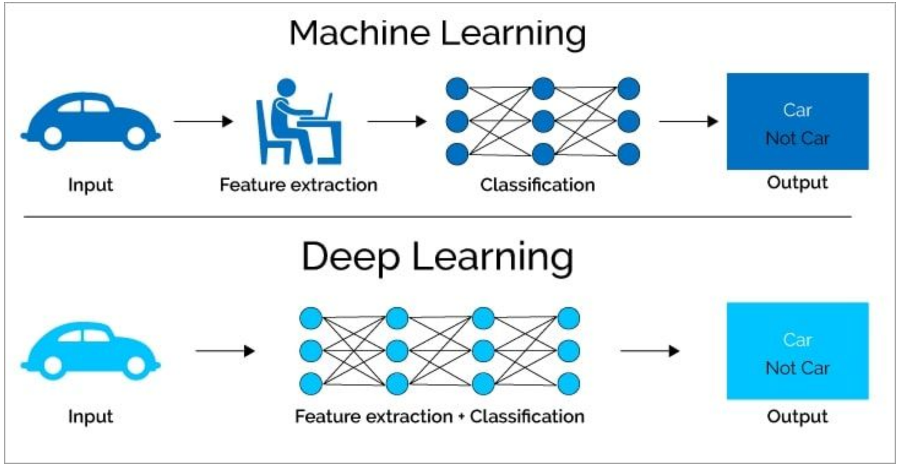
Curse of dimensionality

- We have seen that adding dimensions helps finding a separator
- Yes but overfitting!
- Growing number of features $\Longrightarrow$ amount of data to generalize grows **exponentially**!

Maybe PCA? ☺

Conclusion

From **Machine Learning** to **Deep Learning**



Going deeper:

- Apprentissage et Deep Learning (S7,S8)
- Traitement d’images (S8), Intelligence Artificielle (S8)
- Vision Artificielle (S9).

Questions?

Sources, images courtesy and acknowledgment:
N. Papadakis, J.Guttag, P. Esling

Charles Brazier
charles.brazier@u-bordeaux.fr