Introduction to Machine Learning

Lecture 2

Instructor:
Dr. Tom Arodz

Machine Learning

- Machine learning:
  - Study of algorithms that
    - improve their performance P
    - at some task T
    - with experience E
  - We have a well-defined learning task: <P,T,E>

Machine Learning

- Machine learning: study of algorithms that
  - improve their performance P
    - HOW TO MEASURE THE PERFORMANCE?
  - at some task T
    - WHAT IS THE TASK?
  - with experience E
    - HOW IS EXPERIENCE GAINED?

Supervised Machine Learning

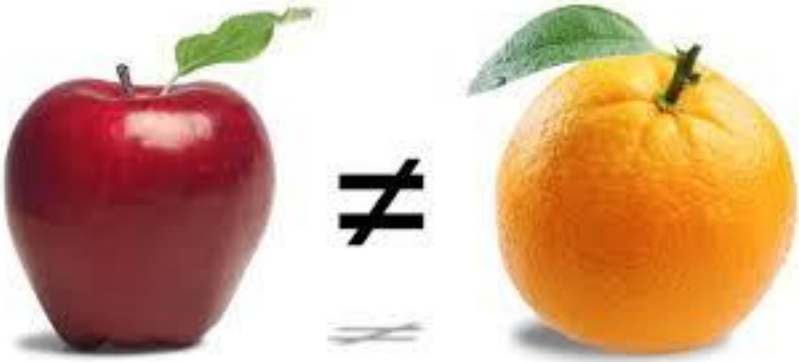
- Algorithms for solving the learning problem:

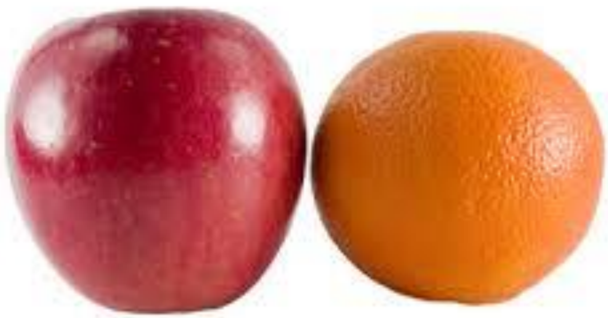
  <Performance, Task, Experience>

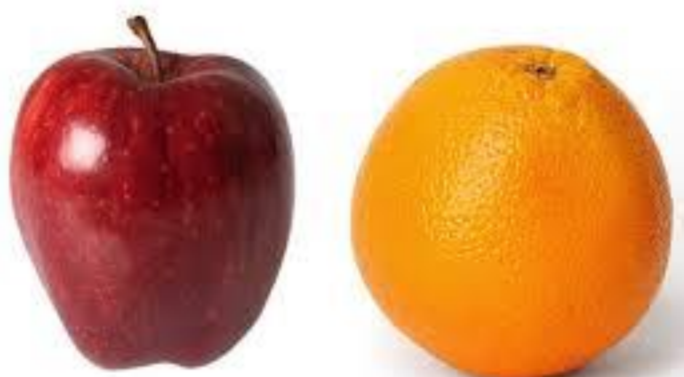
- **Supervised** Machine Learning/Classification:
  - Task:
    - Learn the ability to categorize objects:
      to predict the class of object
      based on some attributes/features of the object
  - Performance:
    - Measured as the accuracy (correctness) of predictions
      for previously unseen objects
  - Experience:
    - A collection of objects, each described by its attributes,
      and labeled with its classes

Supervised ML / Classification

- Apple vs Orange

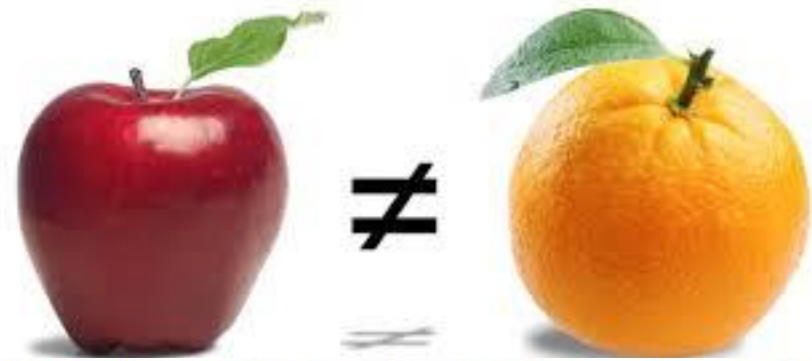


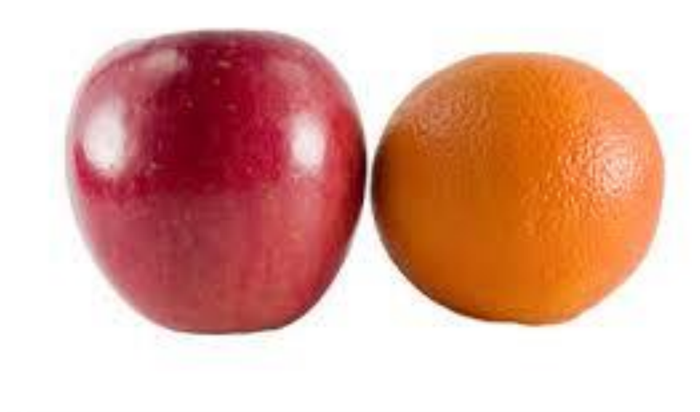




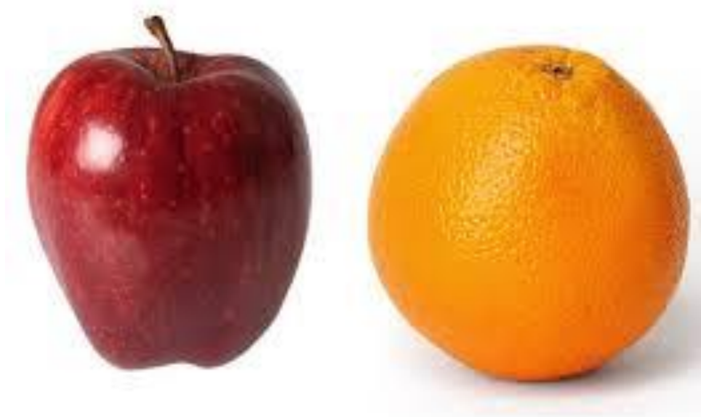
Supervised ML / Classification

- Apple vs Orange



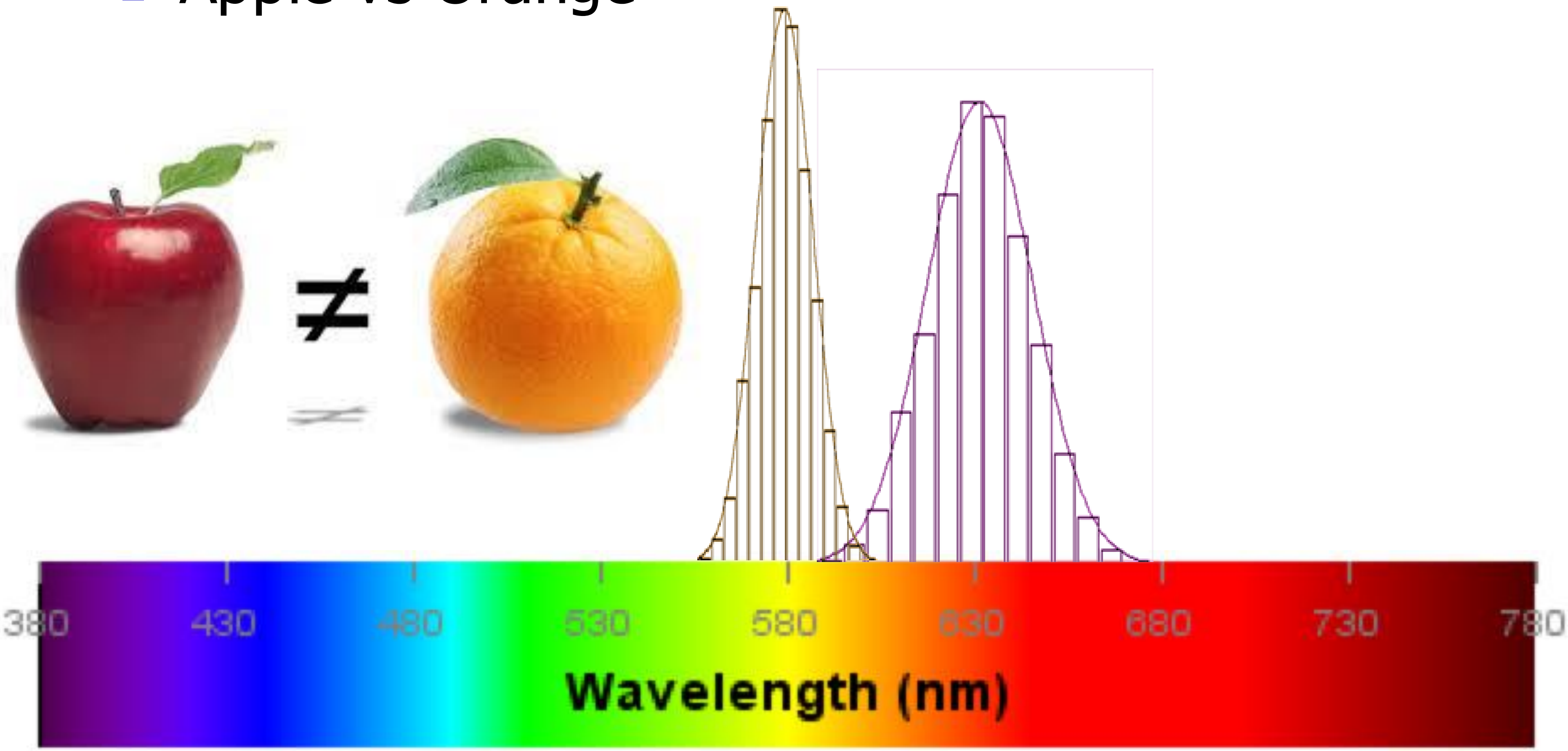


- **Input Feature: object wavelength**
- **If wavelength>600nm then apple otherwise orange**



Supervised ML / Classification

- Apple vs Orange

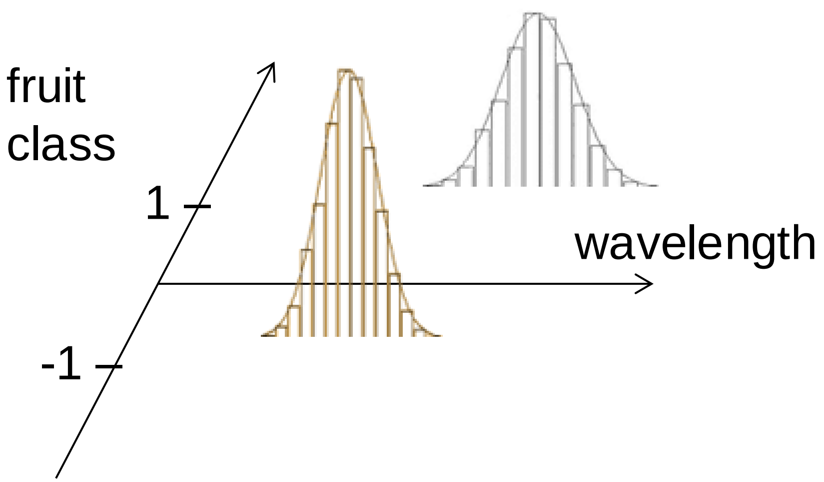


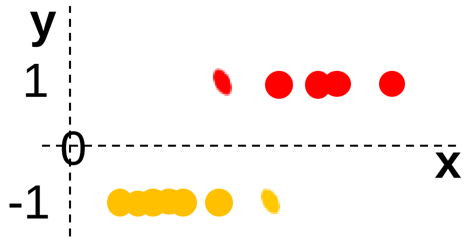
- **Probability distribution of wavelength for Apples**
- **Probability distribution of wavelength for Oranges**

Supervised ML / Classification

- Apple vs Orange
  - **Probability distribution of wavelength for Apples**
  - **Probability distribution of wavelength for Oranges**

    or:
  - **Joint distribution over 2D space**
    (x,y) = (*wavelength, fruit class*)



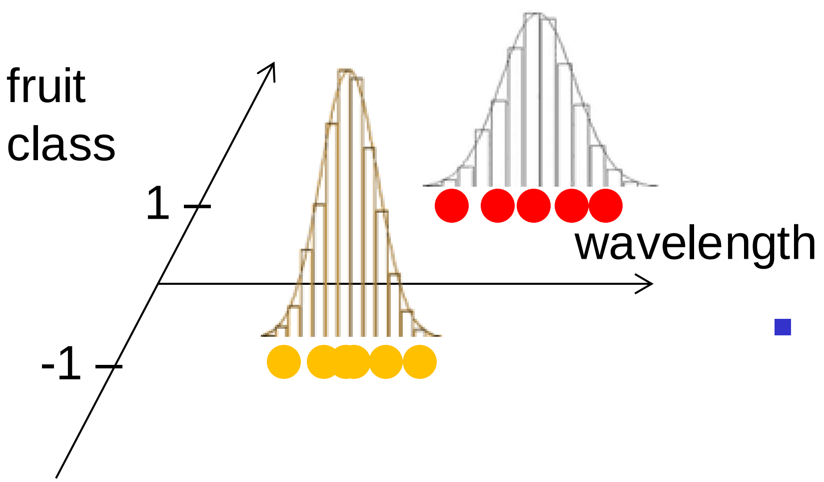


Supervised ML / Classification

- Apple vs Orange
  - Probability distribution of wavelength for Apples
  - Probability distribution of wavelength for Oranges

    or:

  - Joint distribution over 2D space
    (x,y) = (*wavelength*, *fruit class*)



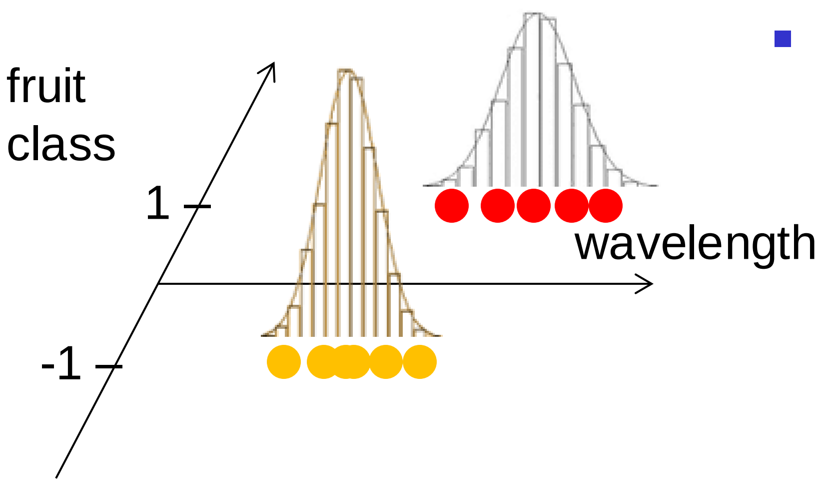
- **We often think in terms of either joint probability distribution $p(x,y)$, or probability distribution within each individual class, i.e., conditional probability of x given class, $p(x|y_i)$**

- **But what we really want for classification is $p(y_i|x)$:**
  what is the probability of class y_i (of apples, or of oranges) for given value of x?

Supervised ML / Classification

- Apple vs Orange
  - Probability distribution of wavelength for Apples
  - Probability distribution of wavelength for Oranges

    or:
  - Joint distribution over 2D space
    (x,y) = (*wavelength, fruit class*)

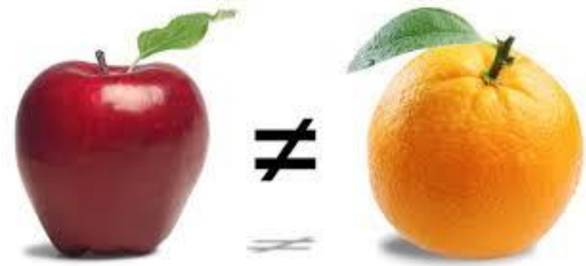


- **For today, let just focus on predicting whether p(apple|x) > 50% (i.e., should we predict apple, or orange)**
  - **That is, we will try to predict class y given wavelength x,**

    not the specific value of probability of that class

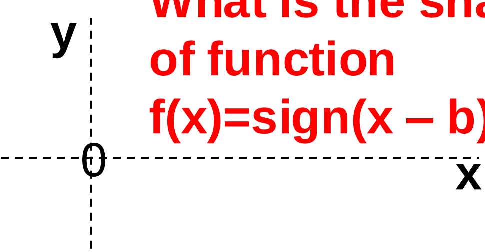
Supervised ML / Classification

- Apple vs Orange

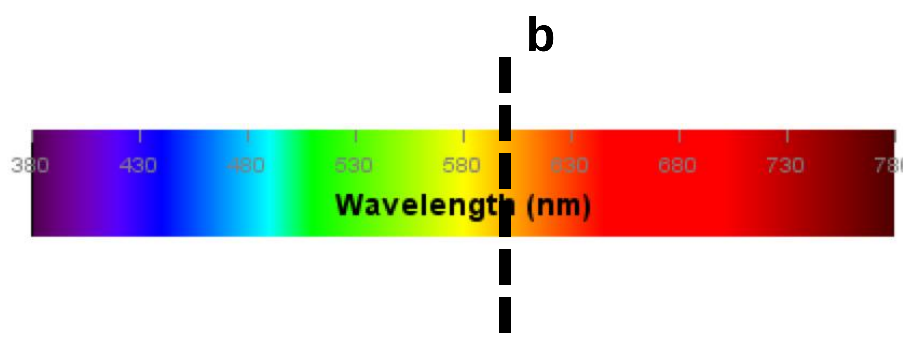


- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)

- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
  - **b is unknown**, need to be learned from examples

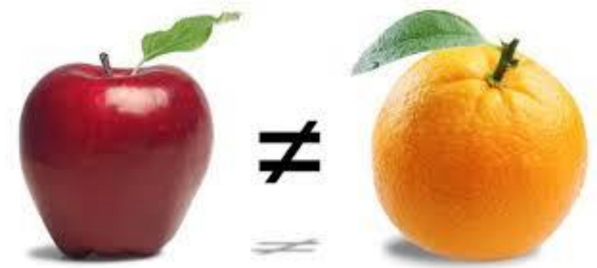


What is the shape/plot of function f(x)=sign(x – b) ?



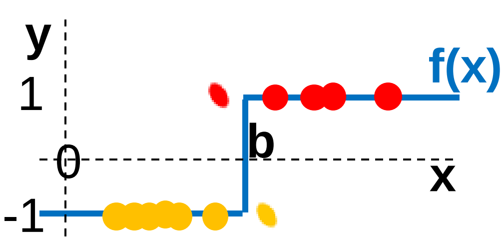
Supervised ML / Classification

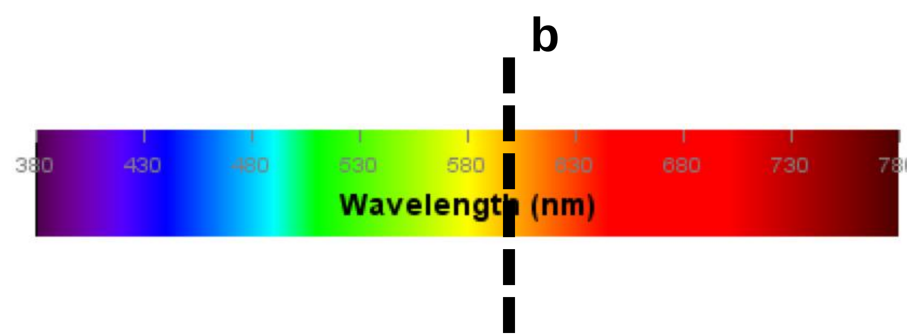
- Apple vs Orange



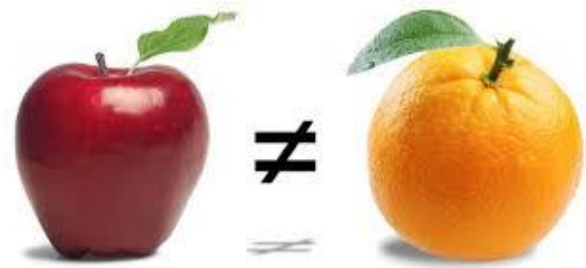
- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)

- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
  - **b is unknown**, need to be learned from examples





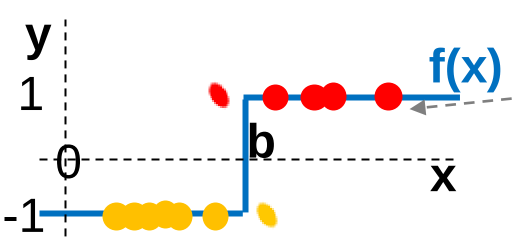
Supervised ML / Classification



- Apple vs Orange

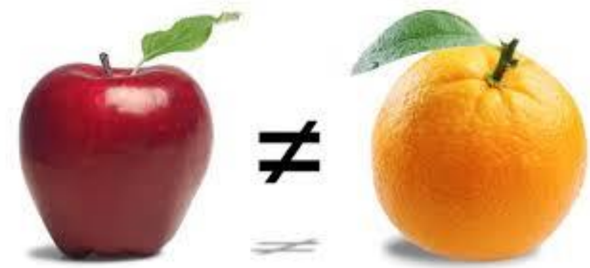
- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)

- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
  - **b is unknown**, need to be learned from examples



An apple: y=1
predicted as
apple: f(x)=1

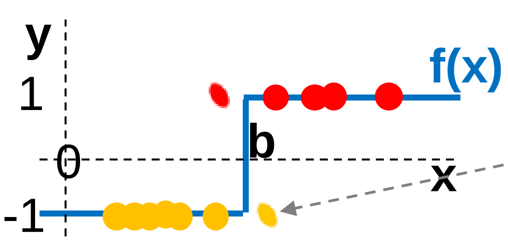
Supervised ML / Classification



- Apple vs Orange

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)

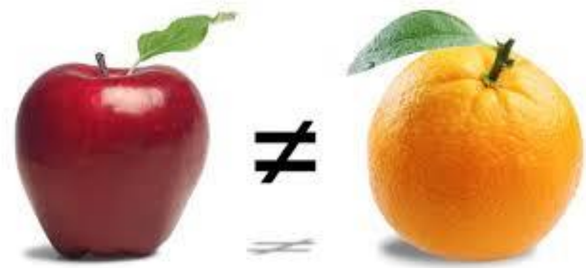
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
  - **b is unknown**, need to be learned from examples



An orange: y=1
predicted as
apple: f(x)=1

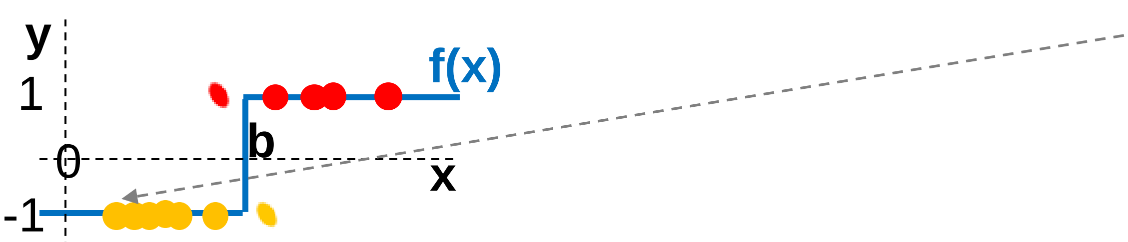
Supervised ML / Classification

- Apple vs Orange



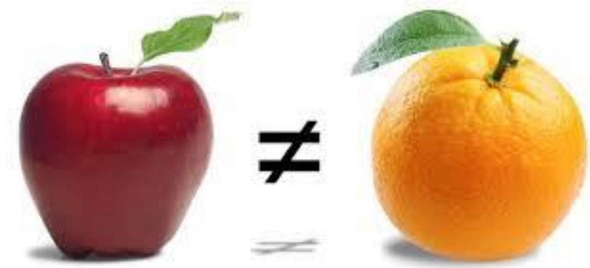
- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)

- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
  - **b is unknown**, need to be learned from examples



An orange: y=-1 predicted as orange: f(x)=-1

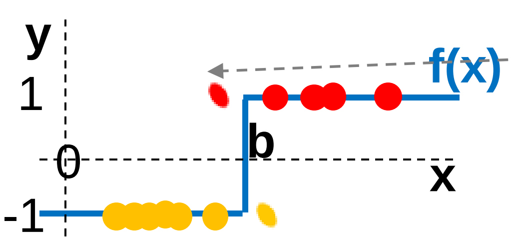
Supervised ML / Classification



- ## Apple vs Orange

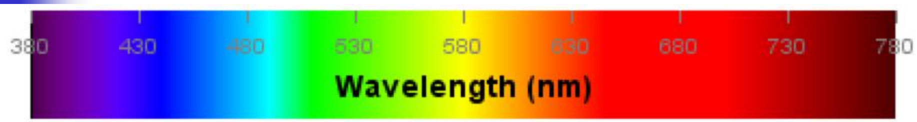
- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)

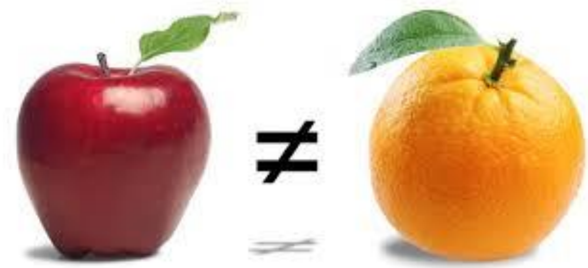
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
  - **b is unknown**, need to be learned from examples



An apple: y=1
predicted as
orange: f(x)=-1

Supervised ML / Classification

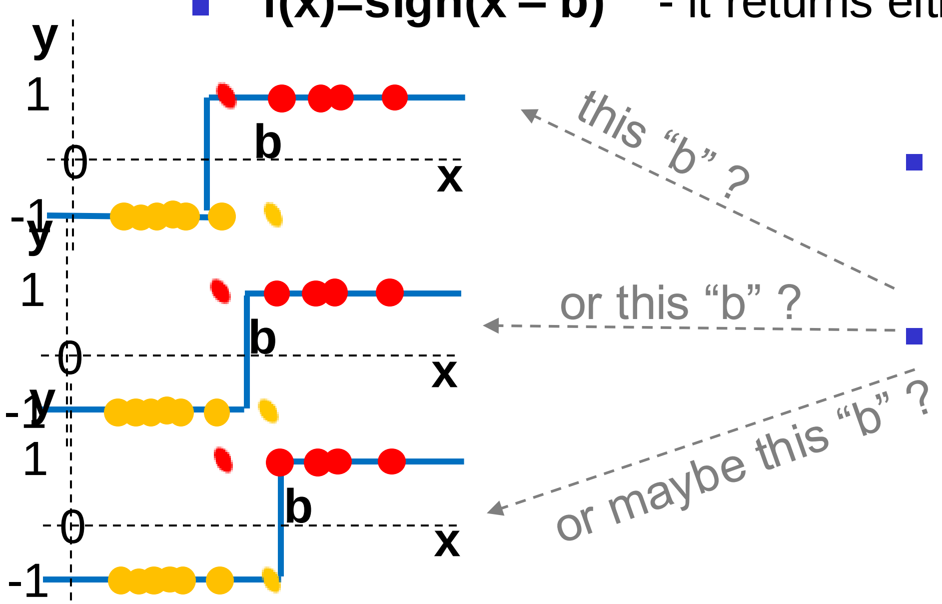




- Apple vs Orange

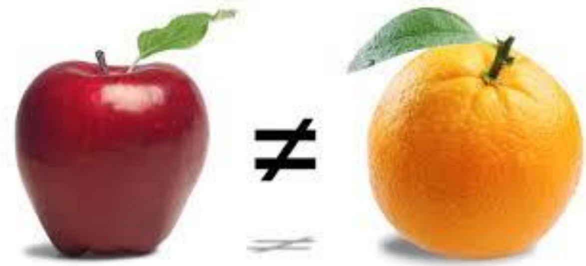
- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)

- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)



- **b is unknown**, need to be learned from **examples**
- different **b** => different **f(x)** => different predictions for the same **x**

Supervised ML / Classification

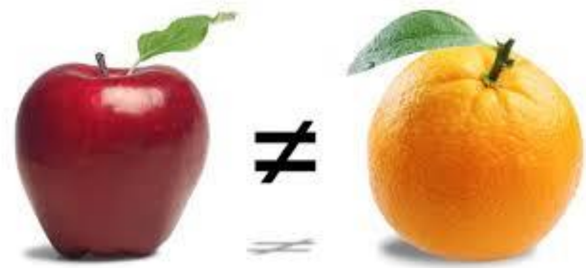


- Apple vs Orange

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
  - The function has one input (x)
    and one trainable parameter (b)

- Algorithm:
  - **Set initial value of trainable parameter b (0, or random, or a guess)**
  - **Loop:**
    - **Present a sample x_i and predict $f(x_i)=\text{sign}(x_i - b)$**
    - **Compare true class y_i with predicted class $f(x_i)$**
    - **If prediction is right, go to next sample (i=i+1)**

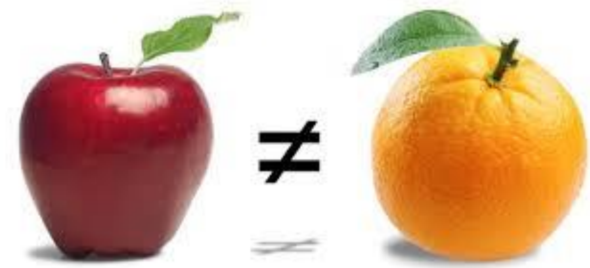
Supervised ML / Classification



- Apple vs Orange

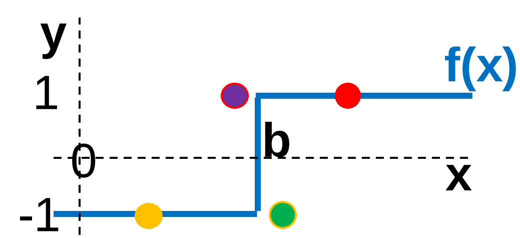
- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - **Set initial value of b (0, or random, or a guess)**
  - **Loop:**
    - **Present a sample x_i and predict $f(x_i)=\mathrm{sign}(x_i - b)$**
    - **Compare true class y_i with predicted class $f(x_i)$**
    - **If prediction is right, go to next sample (i=i+1)**
    - **If prediction is wrong, update trainable parameter b**
      - **How?**

Supervised ML / Classification

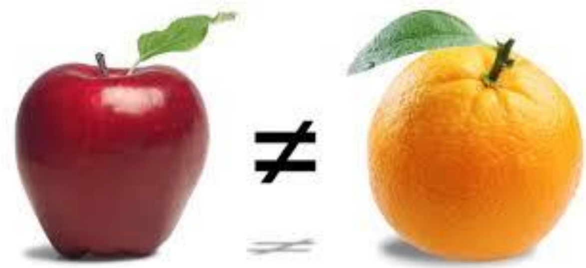


- Apple vs Orange

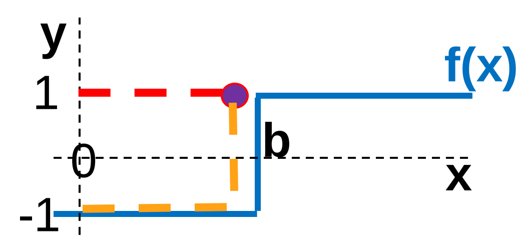
- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - `Set initial value of b (0, or random, or a guess)`
  - `Loop:`
    - `Present a sample` x_i `and predict` $f(x_i)=\text{sign}(x_i - b)$
    - `Compare true class` y_i `with predicted class` $f(x_i)$
    - **`If prediction is wrong, update b`**
      - **Assume $y_i = 1$ (apple), but prediction is $f(x_i) = -1$ (orange)**
      - **Which of the four situations on the plot is it?**



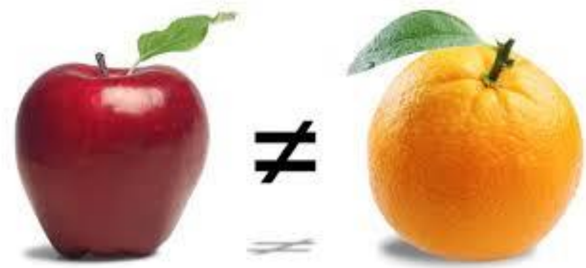
Supervised ML / Classification



- Apple vs Orange
- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **$f(x)=\text{sign}(x - b)$** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - `Set initial value of b (0, or random, or a guess)`
  - `Loop:`
    - `Present a sample` x_i `and predict` $f(x_i)=\text{sign}(x_i - b)$
    - `Compare true class` y_i `with predicted class` $f(x_i)$
    - **`If prediction is wrong, update b`**
      - **if $y_i = 1$ (apple),**
        but prediction is $f(x_i) = -1$ (orange),
        how should we tweak "b"?

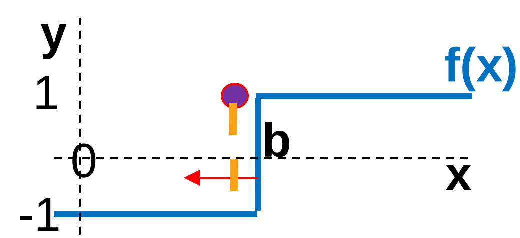


Supervised ML / Classification

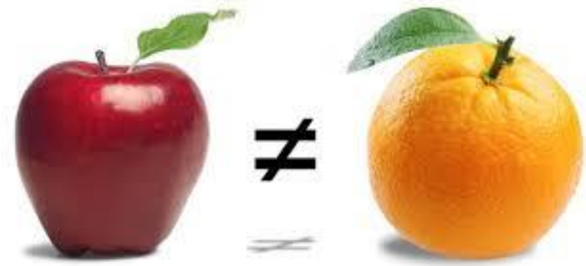


- Apple vs Orange

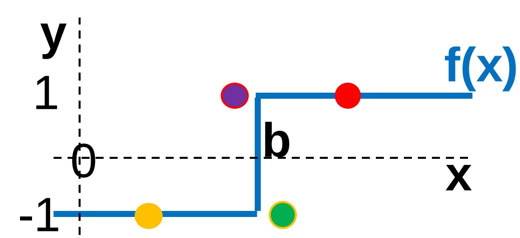
- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - Set initial value of b (0, or random, or a guess)
  - Loop:
    - Present a sample x_i and predict $f(x_i)=\text{sign}(x_i - b)$
    - Compare true class y_i with predicted class $f(x_i)$
    - **If prediction is wrong, update b**
      - **if y_i = 1 (apple), but prediction is $f(x_i)$ = -1 (orange), how should we tweak "b"?**
      - **We need to move the threshold to the left. That is, decrease b. This may increase f() for that sample.**



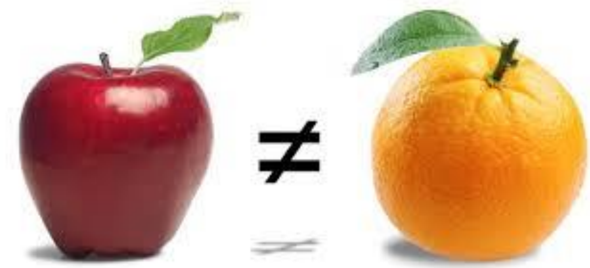
Supervised ML / Classification



- Apple vs Orange
- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - Set initial value of b (0, or random, or a guess)
  - Loop:
    - Present a sample x_i and predict $f(x_i)=\text{sign}(x_i - b)$
    - Compare true class y_i with predicted class $f(x_i)$
    - **If prediction is wrong, update b**
      - **Assume $y_i = -1$ (orange), but prediction is $f(x_i) = +1$ (apple)**
      - **Which of the four situations on the plot is it?**

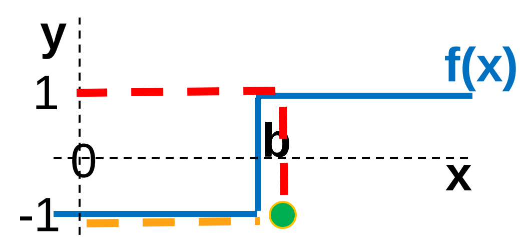


Supervised ML / Classification

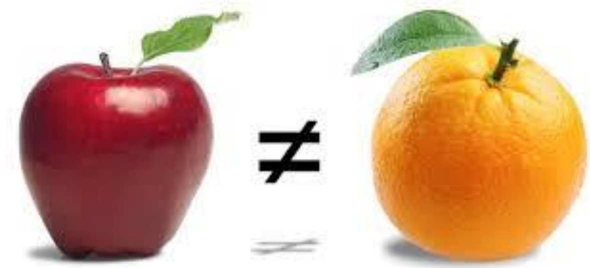


- Apple vs Orange

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - Set initial value of b (0, or random, or a guess)
  - Loop:
    - Present a sample x_i and predict $f(x_i)=sign(x_i – b)$
    - Compare true class y_i with predicted class $f(x_i)$
    - **If prediction is wrong, update b**
      - **Assume y_i = -1 (orange),**
        but prediction is $f(x_i)$ = +1 (apple)
        how should we tweak "b"?

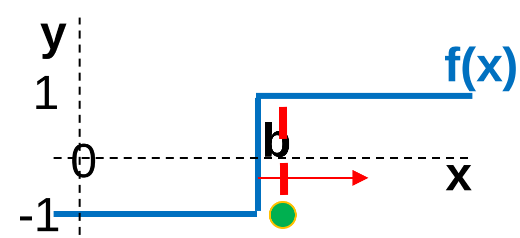


Supervised ML / Classification

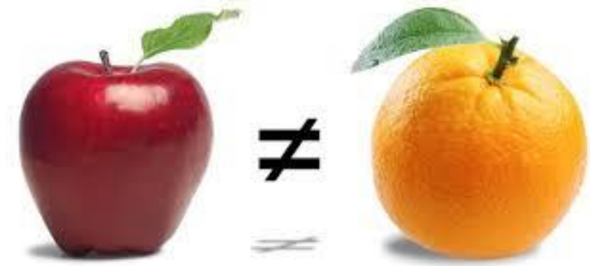


- Apple vs Orange

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - Set initial value of b (0, or random, or a guess)
  - Loop:
    - Present a sample x_i and predict $f(x_i)=\text{sign}(x_i - b)$
    - Compare true class y_i with predicted class $f(x_i)$
    - **If prediction is wrong, update b**
      - **Assume y_i = -1 (orange),**
        but prediction is $f(x_i)$ = +1 (apple)
        how should we tweak "b"?
      - **We need to move the threshold to the right. That is, increase b.**
        This may decrease f() for that sample.



Supervised ML / Classification



- Apple vs Orange
- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - **Set initial value of b (0, or random, or a guess)**
  - **Loop:**
    - **Present a sample x_i and predict $f(x_i)=\text{sign}(x_i - b)$**
    - **Compare true class y_i with predicted class $f(x_i)$**
    - **If prediction is right, go to next sample (i=i+1)**
    - **If prediction is wrong, update b**
      - **In a way that would make it more likely to get correct prediction for the current sample**
      - **if $y_i = 1$, $f(x_i)=-1$, decrease b to increase f**
      - **if $y_i = -1$, $f(x_i)=1$, increase b to decrease f**

Supervised ML / Classification

- Example run
- Start by setting b=560
- Each increase/decrease is by 10

  - y=-1, x=580, f(x)=+1, increase b to b=570

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - **Set initial value of b (0, or random, or a guess)**
  - **Loop:**
    - **Present a sample x_i and predict $f(x_i)=\text{sign}(x_i – b)$**
    - **Compare true class y_i with predicted class $f(x_i)$**
    - **If prediction is right, go to next sample (i=i+1)**
    - **If prediction is wrong, update b**
      - **In a way that would make it more likely to get correct prediction for the current sample**
      - **if y_i = 1, $f(x_i)$=-1, decrease b to increase f**
      - **if y_i = -1, $f(x_i)$=1, increase b to decrease f**

Supervised ML / Classification

- Example run
- Start by setting b=560
- Each increase/decrease is by 10

  - y=-1, x=580, f(x)=+1, increase b to b=570

  - y=+1, x=640, f(x)=+1, do nothing (b=570)

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - **Set initial value of b (0, or random, or a guess)**
  - **Loop:**
    - **Present a sample x_i and predict $f(x_i)=\text{sign}(x_i - b)$**
    - **Compare true class y_i with predicted class $f(x_i)$**
    - **If prediction is right, go to next sample (i=i+1)**
    - **If prediction is wrong, update b**
      - **In a way that would make it more likely to get correct prediction for the current sample**
      - **if $y_i = 1$, $f(x_i)=-1$, decrease b to increase f**
      - **if $y_i = -1$, $f(x_i)=1$, increase b to decrease f**

Supervised ML / Classification

- Example run
- Start by setting b=560
- Each increase/decrease is by 10

  - y=-1, x=580, f(x)=+1, increase b to b=570

  - y=+1, x=640, f(x)=+1, do nothing (b=570)

  - y=-1, x=580, f(x)=+1, increase b to b=580

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - `Set initial value of b (0, or random, or a guess)`
  - `Loop:`
    - Present a sample x_i and predict $f(x_i)=\text{sign}(x_i - b)$
    - Compare true class y_i with predicted class $f(x_i)$
    - If prediction is right, go to next sample (i=i+1)
    - If prediction is wrong, update b
      - In a way that would make it more likely to get correct prediction for the current sample
      - **if y_i = 1, $f(x_i)$=-1, decrease b to increase f**
      - **if y_i = -1, $f(x_i)$=1, increase b to decrease f**

Supervised ML / Classification

- Example run
- Start by setting b=560
- Each increase/decrease is by 10

  - y=-1, x=580, f(x)=+1, increase b to b=570

  - y=+1, x=640, f(x)=+1, do nothing (b=570)

  - y=-1, x=580, f(x)=+1, increase b to b=580

  - y=-1, x=615, f(x)=+1, increase b to b=590

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - **Set initial value of b (0, or random, or a guess)**
  - **Loop:**
    - **Present a sample x_i and predict $f(x_i)=sign(x_i – b)$**
    - **Compare true class y_i with predicted class $f(x_i)$**
    - **If prediction is right, go to next sample (i=i+1)**
    - **If prediction is wrong, update b**
      - **In a way that would make it more likely to get correct prediction for the current sample**
      - **if y_i = 1, $f(x_i)$=-1, decrease b to increase f**
      - **if y_i = -1, $f(x_i)$=1, increase b to decrease f**

Supervised ML / Classification

- Example run
- Start by setting b=560
- Each increase/decrease is by 10
  -  y=-1, x=580, f(x)=+1, increase b to b=570
  -  y=+1, x=640, f(x)=+1, do nothing (b=570)
  -  y=-1, x=580, f(x)=+1, increase b to b=580
  -  y=-1, x=615, f(x)=+1, increase b to b=590
  -  y=-1, x=615, f(x)=+1, increase b to b=600

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - Set initial value of b (0, or random, or a guess)
  - Loop:
    - Present a sample x_i and predict $f(x_i)=sign(x_i – b)$
    - Compare true class y_i with predicted class $f(x_i)$
    - If prediction is right, go to next sample (i=i+1)
    - If prediction is wrong, update b
      - In a way that would make it more likely to get correct prediction for the current sample
      - **if y_i = 1, $f(x_i)$=-1, decrease b to increase f**
      - **if y_i = -1, $f(x_i)$=1, increase b to decrease f**

Supervised ML / Classification

- Example run
- Start by setting b=560
- Each increase/decrease is by 10

  - $y=-1$, $x=580$, $f(x)=+1$, increase b to b=570

  - $y=+1$, $x=640$, $f(x)=+1$, do nothing (b=570)

  - $y=-1$, $x=580$, $f(x)=+1$, increase b to b=580

  - $y=-1$, $x=615$, $f(x)=+1$, increase b to b=590

  - $y=-1$, $x=615$, $f(x)=+1$, increase b to b=600

  - $y=+1$, $x=620$, $f(x)=+1$, do nothing (b=600)

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - Set initial value of b (0, or random, or a guess)
  - Loop:
    - Present a sample x_i and predict $f(x_i)=\text{sign}(x_i - b)$
    - Compare true class y_i with predicted class $f(x_i)$
    - If prediction is right, go to next sample (i=i+1)
    - If prediction is wrong, update b
      - In a way that would make it more likely to get correct prediction for the current sample
      - **if $y_i = 1$, $f(x_i)=-1$, decrease b to increase f**
      - **if $y_i = -1$, $f(x_i)=1$, increase b to decrease f**

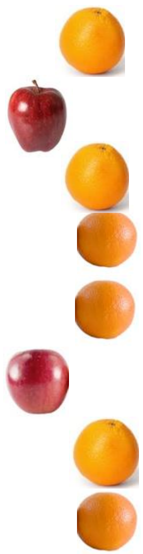
Supervised ML / Classification

- Example run
- Start by setting b=560
- Each increase/decrease is by 10
  -  y=-1, x=580, f(x)=+1, increase b to b=570
  -  y=+1, x=640, f(x)=+1, do nothing (b=570)
  -  y=-1, x=580, f(x)=+1, increase b to b=580
  -  y=-1, x=615, f(x)=+1, increase b to b=590
  -  y=-1, x=615, f(x)=+1, increase b to b=600
  -  y=+1, x=620, f(x)=+1, do nothing (b=600)
  -  y=-1, x=580, f(x)=-1, do nothing (b=600)

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - **Set initial value of b (0, or random, or a guess)**
  - **Loop:**
    - **Present a sample x_i and predict $f(x_i)=\text{sign}(x_i - b)$**
    - **Compare true class y_i with predicted class $f(x_i)$**
    - **If prediction is right, go to next sample (i=i+1)**
    - **If prediction is wrong, update b**
      - **In a way that would make it more likely to get correct prediction for the current sample**
      - **if y_i = 1, $f(x_i)$=-1, decrease b to increase f**
      - **if y_i = -1, $f(x_i)$=1, increase b to decrease f**

Supervised ML / Classification

- Example run
- Start by setting b=560
- Each increase/decrease is by 10



  - y=-1, x=580, f(x)=+1, increase b to b=570
  - y=+1, x=640, f(x)=+1, do nothing (b=570)
  - y=-1, x=580, f(x)=+1, increase b to b=580
  - y=-1, x=615, f(x)=+1, increase b to b=590
  - y=-1, x=615, f(x)=+1, increase b to b=600
  - y=+1, x=620, f(x)=+1, do nothing (b=600)
  - y=-1, x=580, f(x)=-1, do nothing (b=600)
  - y=-1, x=615, f(x)=+1, increase b to b=610

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - Set initial value of b (0, or random, or a guess)
  - Loop:
    - Present a sample x_i and predict $f(x_i)=\text{sign}(x_i - b)$
    - Compare true class y_i with predicted class $f(x_i)$
    - If prediction is right, go to next sample (i=i+1)
    - If prediction is wrong, update b
      - In a way that would make it more likely to get correct prediction for the current sample
      - **if $y_i = 1$, $f(x_i)=-1$, decrease b to increase f**
      - **if $y_i = -1$, $f(x_i)=1$, increase b to decrease f**

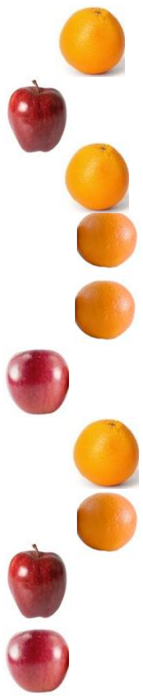
Supervised ML / Classification

- Example run
- Start by setting b=560
- Each increase/decrease is by 10
  - 
    y=-1, x=580, f(x)=+1, increase b to b=570
  - 
    y=+1, x=640, f(x)=+1, do nothing (b=570)
  - 
    y=-1, x=580, f(x)=+1, increase b to b=580
  - 
    y=-1, x=615, f(x)=+1, increase b to b=590
  - 
    y=-1, x=615, f(x)=+1, increase b to b=600
  - 
    y=+1, x=620, f(x)=+1, do nothing (b=600)
  - 
    y=-1, x=580, f(x)=-1, do nothing (b=600)
  - 
    y=-1, x=615, f(x)=+1, increase b to b=610
  - 
    y=+1, x=640, f(x)=+1, do nothing (b=610)

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - **Set initial value of b (0, or random, or a guess)**
  - **Loop:**
    - **Present a sample x_i and predict $f(x_i)=sign(x_i – b)$**
    - **Compare true class y_i with predicted class $f(x_i)$**
    - **If prediction is right, go to next sample (i=i+1)**
    - **If prediction is wrong, update b**
      - **In a way that would make it more likely to get correct prediction for the current sample**
      - **if y_i = 1, $f(x_i)$=-1, decrease b to increase f**
      - **if y_i = -1, $f(x_i)$=1, increase b to decrease f**

Supervised ML / Classification

- Example run
- Start by setting b=560
- Each increase/decrease is by 10

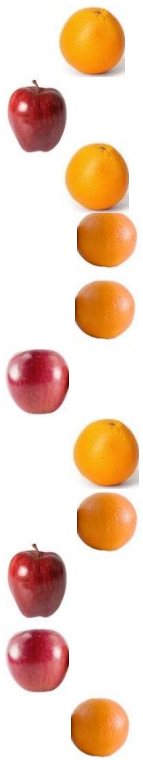


  - y=-1, x=580, f(x)=+1, increase b to b=570
  - y=+1, x=640, f(x)=+1, do nothing (b=570)
  - y=-1, x=580, f(x)=+1, increase b to b=580
  - y=-1, x=615, f(x)=+1, increase b to b=590
  - y=-1, x=615, f(x)=+1, increase b to b=600
  - y=+1, x=620, f(x)=+1, do nothing (b=600)
  - y=-1, x=580, f(x)=-1, do nothing (b=600)
  - y=-1, x=615, f(x)=+1, increase b to b=610
  - y=+1, x=640, f(x)=+1, do nothing (b=610)
  - y=+1, x=618, f(x)=+1, do nothing (b=610)

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - Set initial value of b (0, or random, or a guess)
  - Loop:
    - Present a sample x_i and predict $f(x_i)=\text{sign}(x_i - b)$
    - Compare true class y_i with predicted class $f(x_i)$
    - If prediction is right, go to next sample (i=i+1)
    - If prediction is wrong, update b
      - In a way that would make it more likely to get correct prediction for the current sample
      - **if $y_i = 1$, $f(x_i)=-1$, decrease b to increase f**
      - **if $y_i = -1$, $f(x_i)=1$, increase b to decrease f**

Supervised ML / Classification

- Example run
- Start by setting b=560
- Each increase/decrease is by 10



  - y=-1, x=580, f(x)=+1, increase b to b=570
  - y=+1, x=640, f(x)=+1, do nothing (b=570)
  - y=-1, x=580, f(x)=+1, increase b to b=580
  - y=-1, x=615, f(x)=+1, increase b to b=590
  - y=-1, x=615, f(x)=+1, increase b to b=600
  - y=+1, x=620, f(x)=+1, do nothing (b=600)
  - y=-1, x=580, f(x)=-1, do nothing (b=600)
  - y=-1, x=615, f(x)=+1, increase b to b=610
  - y=+1, x=640, f(x)=+1, do nothing (b=610)
  - y=+1, x=618, f(x)=+1, do nothing (b=610)
  - y=-1, x=615, f(x)=+1, increase b to b=620

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - **Set initial value of b (0, or random, or a guess)**
  - **Loop:**
    - **Present a sample x_i and predict $f(x_i)$=sign(x_i – b)**
    - **Compare true class y_i with predicted class $f(x_i)$**
    - **If prediction is right, go to next sample (i=i+1)**
    - **If prediction is wrong, update b**
      - **In a way that would make it more likely to get correct prediction for the current sample**
      - **if y_i = 1, $f(x_i)$=-1, decrease b to increase f**
      - **if y_i = -1, $f(x_i)$=1, increase b to decrease f**

Supervised ML / Classification

- Example run
- Start by setting b=560
- Each increase/decrease is by 10

  - y=-1, x=580, f(x)=+1, increase b to b=570

  - y=+1, x=640, f(x)=+1, do nothing (b=570)

  - y=-1, x=580, f(x)=+1, increase b to b=580

  - y=-1, x=615, f(x)=+1, increase b to b=590

  - y=-1, x=615, f(x)=+1, increase b to b=600

  - y=+1, x=620, f(x)=+1, do nothing (b=600)

  - y=-1, x=580, f(x)=-1, do nothing (b=600)

  - y=-1, x=615, f(x)=+1, increase b to b=610

  - y=+1, x=640, f(x)=+1, do nothing (b=610)

  - y=+1, x=618, f(x)=+1, do nothing (b=610)

  - y=-1, x=615, f(x)=+1, increase b to b=620

  - y=+1, x=618, f(x)=-1, decrease b to b=610

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - Set initial value of b (0, or random, or a guess)
  - Loop:
    - Present a sample x_i and predict $f(x_i)=\text{sign}(x_i - b)$
    - Compare true class y_i with predicted class $f(x_i)$
    - If prediction is right, go to next sample (i=i+1)
    - If prediction is wrong, update b
      - In a way that would make it more likely to get correct prediction for the current sample
      - **if y_i = 1, $f(x_i)$=-1, decrease b to increase f**
      - **if y_i = -1, $f(x_i)$=1, increase b to decrease f**

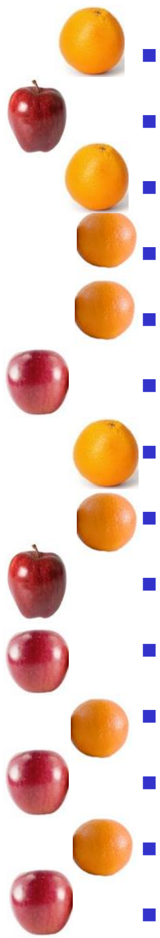
Supervised ML / Classification

- Example run
- Start by setting b=560
- Each increase/decrease is by 10
  - y=-1, x=580, f(x)=+1, increase b to b=570
  - y=+1, x=640, f(x)=+1, do nothing (b=570)
  - y=-1, x=580, f(x)=+1, increase b to b=580
  - y=-1, x=615, f(x)=+1, increase b to b=590
  - y=-1, x=615, f(x)=+1, increase b to b=600
  - y=+1, x=620, f(x)=+1, do nothing (b=600)
  - y=-1, x=580, f(x)=-1, do nothing (b=600)
  - y=-1, x=615, f(x)=+1, increase b to b=610
  - y=+1, x=640, f(x)=+1, do nothing (b=610)
  - y=+1, x=618, f(x)=+1, do nothing (b=610)
  - y=-1, x=615, f(x)=+1, increase b to b=620
  - y=+1, x=618, f(x)=-1, decrease b to b=610
  - y=-1, x=615, f(x)=+1, increase b to b=620

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - **Set initial value of b (0, or random, or a guess)**
  - **Loop:**
    - **Present a sample x_i and predict $f(x_i)=\text{sign}(x_i – b)$**
    - **Compare true class y_i with predicted class $f(x_i)$**
    - **If prediction is right, go to next sample (i=i+1)**
    - **If prediction is wrong, update b**
      - **In a way that would make it more likely to get correct prediction for the current sample**
      - **if y_i = 1, $f(x_i)$=-1, decrease b to increase f**
      - **if y_i = -1, $f(x_i)$=1, increase b to decrease f**

Supervised ML / Classification

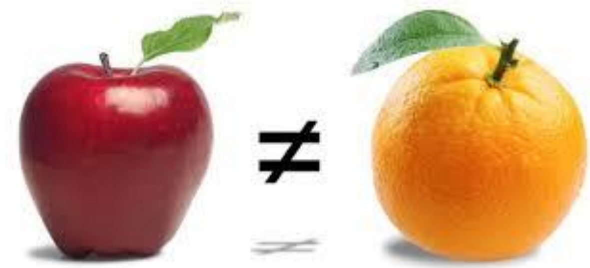
- Example run
- Start by setting b=560
- Each increase/decrease is by 10



  - y=-1, x=580, f(x)=+1, increase b to b=570
  - y=+1, x=640, f(x)=+1, do nothing (b=570)
  - y=-1, x=580, f(x)=+1, increase b to b=580
  - y=-1, x=615, f(x)=+1, increase b to b=590
  - y=-1, x=615, f(x)=+1, increase b to b=600
  - y=+1, x=620, f(x)=+1, do nothing (b=600)
  - y=-1, x=580, f(x)=-1, do nothing (b=600)
  - y=-1, x=615, f(x)=+1, increase b to b=610
  - y=+1, x=640, f(x)=+1, do nothing (b=610)
  - y=+1, x=618, f(x)=+1, do nothing (b=610)
  - y=-1, x=615, f(x)=+1, increase b to b=620
  - y=+1, x=618, f(x)=-1, decrease b to b=610
  - y=-1, x=615, f(x)=+1, increase b to b=620
  - y=+1, x=618, f(x)=-1, decrease b to b=610
  - ...

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Algorithm:
  - `Set initial value of b (0, or random, or a guess)`
  - `Loop:`
    - `Present a sample` x_i `and predict` $f(x_i)=\text{sign}(x_i - b)$
    - `Compare true class` y_i `with predicted class` $f(x_i)$
    - `If prediction is right, go to next sample (i=i+1)`
    - `If prediction is wrong, update b`
      - `In a way that would make it more likely to get correct prediction for the current sample`
      - **if y_i = 1, $f(x_i)$=-1, decrease b to increase f**
      - **if y_i = -1, $f(x_i)$=1, increase b to decrease f**

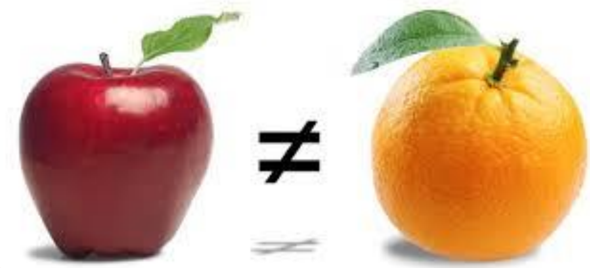
Supervised ML / Classification



- Apple vs Orange

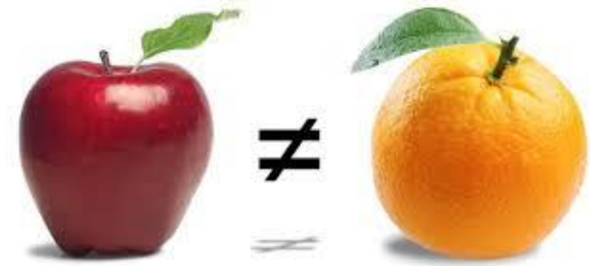
- Algorithm:
  - **Set initial value of b (0, or random, or a guess)**
  - **Loop:**
    - **Present a sample x_i and predict $f(x_i)=\mathrm{sign}(x_i - b)$**
    - **Compare true class y_i with predicted class $f(x_i)$**
    - **If prediction is right, go to next sample (i=i+1)**
    - **If prediction is wrong, update b**
      - if $y_i = 1$, $f(x_i)=-1$, what should we do? Decrease b to increase f
        - $y_i - f(x_i) = 2$
        - $b_{new}=b_{old} - 2c$
      - if $y_i = -1$, $f(x_i)=1$, what should we do? Increase b to decrease f
        - $y_i - f(x_i) = -2$
        - $b_{new}=b_{old} + 2c$
      - **What is "c"? The "learning rate", a small number chosen by the user**
      - **Single formula: $b_{new}=b_{old}-c[y_i - f(x_i)]$**

Supervised ML / Classification



- Apple vs Orange
- Algorithm **(with small tweak, sign in front of b)**:
  - `Set initial value of b (0, or random, or a guess)`
  - `Loop:`
    - Present a sample x_i and predict $f(x_i)=\text{sign}(x_i + b)$
    - Compare true class y_i with predicted class $f(x_i)$
    - If prediction is right, go to next sample (i=i+1)
    - If prediction is wrong, update b
      - if $y_i = 1$, $f(x_i)=-1$, what should we do? **Increase** b to increase f
        - $y_i - f(x_i) = 2$
        - $b_{new}=b_{old} + 2c$
      - if $y_i = -1$, $f(x_i)=1$, what should we do? **Decrease** b to decrease f
        - $y_i - f(x_i) = -2$
        - $b_{new}=b_{old} - 2c$
      -
      - Single formula: $b_{new}=b_{old}+c[y_i - f(x_i)]$

Supervised ML / Classification



- Apple vs Orange

- Algorithm
  unified, w_0 can be set to -1 (original version) or +1 (tweaked version):

  - Set initial value of b (0, or random, or a guess)

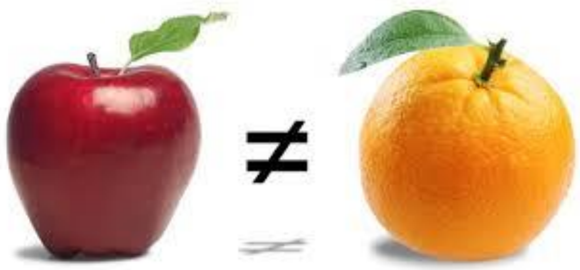
  - Loop:
    - Present a sample x_i and predict $f(x_i)=\text{sign}(x_i + w_0 b)$
    - Compare true class y_i with predicted class $f(x_i)$

    - If prediction is right, go to next sample (i=i+1)

    - If prediction is wrong, update b

      - $b_{new}=b_{old} + c[y_i - f(x_i)]w_0$

Supervised ML / Classification



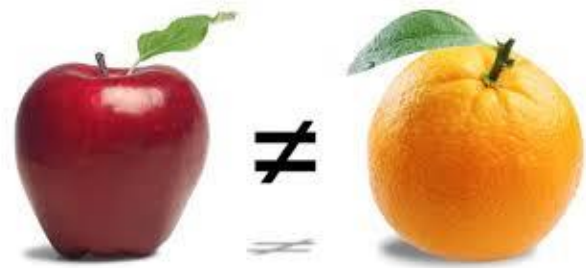
- Apple vs Orange

- We have a simple predictive model that involves one trainable parameter, the threshold value separating apples from oranges

- We have a training algorithm for that model

- **What other ways are there to specify trainable parameters?**

Supervised ML / Classification



- Apple vs Orange

- We have a simple predictive model that involves one trainable parameter, the threshold value separating apples from oranges
- We have a training algorithm for that model

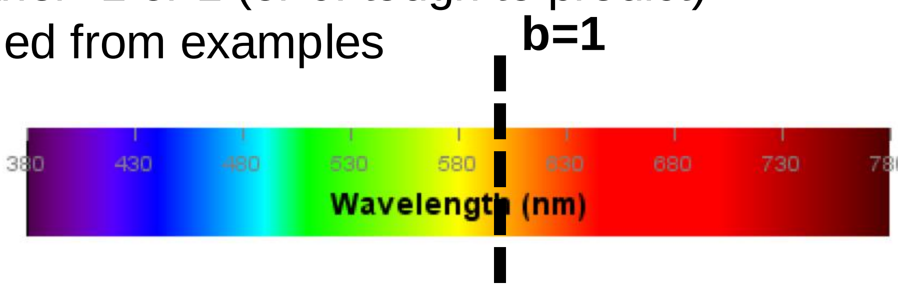
- **What other ways are there to specify trainable parameters?**
  - We can add a parameter that scales the input x

Supervised ML / Classification

- Using a parameter to scale an input attribute

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)

- Prediction will be made by evaluating a function:
  - **f(x)=sign(wx – 1)** - it returns either -1 or 1 (or 0: tough to predict)
  - **w is unknown**, needs to be learned from examples



  - E.g. if wavelength 590 separates applies/oranges, then w=1/590 is a good choice
    - x=600 will lead to wx-1 = 600/590 – 1 = 1.0169 – 1 f>0(apple)
    - x=580 will lead to wx-1 = 580/590 – 1 = 0.9831 – 1 f<0(orange)

Supervised ML / Classification

- Using a parameter to scale an input attribute

- Algorithm
  - Set initial values of w (random guess)
  - Loop:
    - Present a sample x_i and predict $f(x_i)=\text{sign}(wx_i - 1)$
    - Compare true class y_i with predicted class $f(x_i)$
    - If prediction is right, go to next sample (i=i+1)
    - If prediction is wrong, update w
      - $w_{new}=w_{old} + c[y_i - f(x_i)]x_i$
      - Why times x_i?
        - y=+1, f(x)<0, [y-f(x)]>0, we need to increase f(x)
          - x>0: to increase f(x), increase w
          - x<0: to increase f(x), decrease w
        - y=-1, f(x)>0 , [y-f(x)]<0, we need to decrease f(x)
          - x>0: to decrease f(x), decrease w
          - x<0: to decrease f(x), increase w

We could just use: $w_{new}=w_{old} + c[y_i - f(x_i)]\text{sign}(x_i)$
but using x_i also helps us tweak magnitude of the change

Supervised ML / Classification

- Using a parameter to scale an input attribute
- Prediction will be made by evaluating a function:
  - **f(x)=sign(wx – 1)** - it returns either -1 or 1 (or 0: tough to predict)
    - should return the same prediction as
  - **f(x)=sign(x – 1/w)**

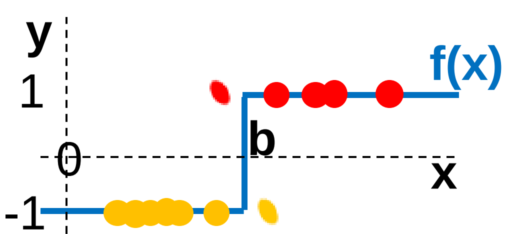
With one attribute (here, wavelength), training the scale w, or training the threshold b is essentially the same*

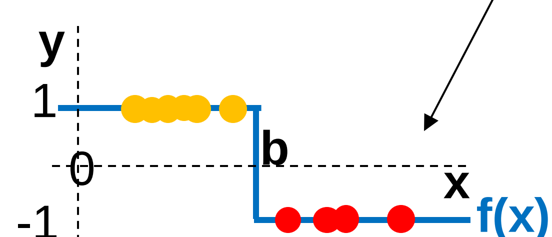
Supervised ML / Classification

- Using a parameter to scale an input attribute
- Prediction will be made by evaluating a function:
  - **f(x)=sign(wx – 1)** - it returns either -1 or 1 (or 0: tough to predict)
    - should return the same prediction as
  - **f(x)=sign(x – 1/w)**

With one attribute (here, wavelength), training the scale w, or training the threshold b is essentially the same*

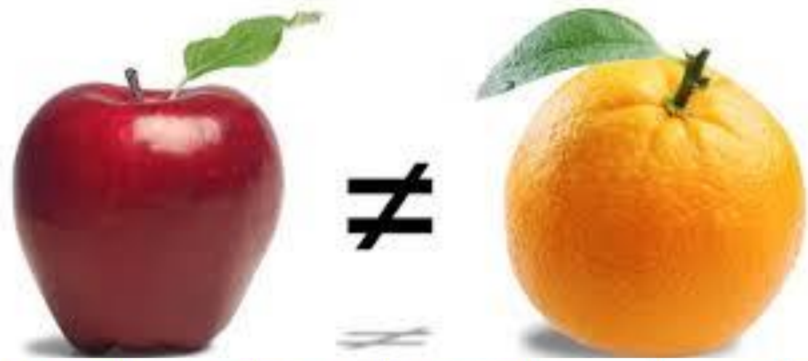
* one small difference: with threshold b, there's no way to achieve inverted predictions, i.e., if we set apples -1, oranges +1, no threshold will work

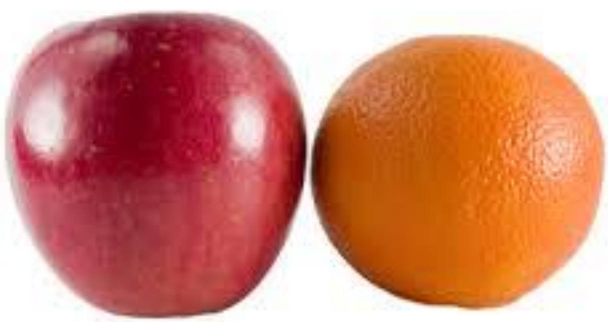




Supervised ML / Classification

- Apple vs Orange
  beyond
  single threshold

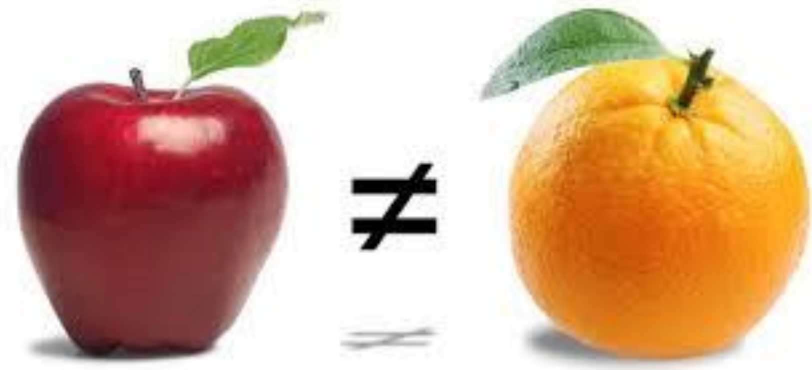


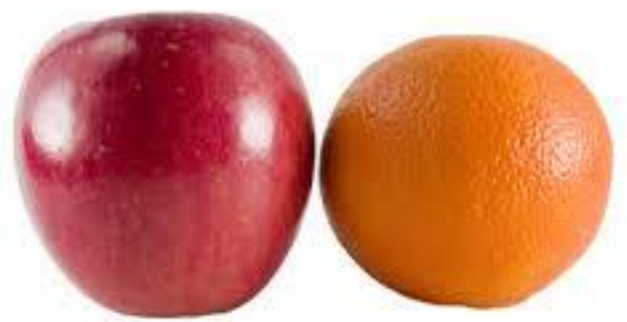


- **Feature A: object wavelength**
- **If A>600 apple**
  otherwise: if A<560 apple
  otherwise orange

Supervised ML / Classification

- Apple vs Orange
  beyond
  single threshold





- **Feature A: object wavelength**
- **If A>600 apple**
  otherwise: if A<560 apple
  otherwise ???

WHAT SHOULD
WE DO?

Wavelength alone
is not enough

Any other attributes/features?

Multiple attributes / features

- **Feature A: color**
  - **Apple: green, red, orange**
  - **Orange: orange**
- **Feature B: color variability**
  - **Apple: uniform or not**
  - **Orange: uniform**
- **Feature C: texture**
  - **Apple: smooth**
  - **Orange: rough**
- **Feature D: reflectance**
  - **Apple: reflects more light**
  - **Orange: reflects less light**
- **Feature E: shape**
  - **Apple: non-convex**
  - **Orange: convex (almost)**

Multiple attributes / features

- **Feature A: color**
  - **Apple: green, red, orange**
  - **Orange: orange**
- **Feature B: color variability**
  - **Apple: uniform or not**
  - **Orange: uniform**
- **Feature C: texture**
  - **Apple: smooth**
  - **Orange: rough**
- **Feature D: reflectance**
  - **Apple: reflects more light**
  - **Orange: reflects less light**
- **Feature E: shape**
  - **Apple: non-convex**
  - **Orange: convex (almost)**
- Using trainable parameters (“w’s”)
  to scale individual features becomes very useful
  when we have >1 feature:
  - we can adjust their importance
  - we can adjust their sign
    (i.e., high means apple or high means orange)

Supervised ML: Perceptron

- One way of making predictions with samples that have 2 features, $x=(x^1, x^2)$
  - Define 2 trainable parameters: weights w_1 and w_2
  - Prediction is: $\mathbf{f(x_i)=sign(w_1 x_i^1 + w_2 x_i^2)}$

Note: here, ^{2} is not 2nd power, just another index: $\mathbf{x_i^2}$=x[i][2]

- Perceptron Algorithm
  - **Set initial values of w_1 and w_2 (random guess)**
  - **Loop:**
  - **Present a sample x_i and predict $f(x_i)=sign(w_1 x_i^1 + w_2 x_i^2)$**
    - **Compare true class y_i with predicted class $f(x_i)$**
    - **If prediction is right, go to next sample (i=i+1)**
    - **If prediction is wrong, update w:**
      - $w_{1,new}=w_{1,old} + c[y_i - f(x_i)]x_i^1$
      - $w_{2,new}=w_{2,old} + c[y_i - f(x_i)]x_i^2$

Same as we did before with one "w", but now it's done separately to each "w_j" using corresponding feature x^j

Same, using vectors

- One way of making predictions with samples that have 2 features, each sample *i* is a 2D vector: $x_i = (x_i^1, x_i^2)$
  - Define a 2D vector $w = (w_1, w_2)$ of 2 trainable parameters

Vector (2x1)

^{T} is transpose op., makes it (1x2)

- Prediction is: $\mathbf{f(x_i) = sign(}\color{red}{\mathbf{w_1 x_i^1 + w_2 x_i^2}}\mathbf{) = sign(w^T x_i)}$

Matrix multiplication: (1x2) times (2x1)

Vector (2x1)

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

2D Vector (2x1, a column vector)

$$\mathbf{x} = \begin{bmatrix} x^1 \\ x^2 \end{bmatrix}$$

2D Vector (2x1, a column vector)

$$\mathbf{w}^T = [w_1, w_2]$$

Transposed vector (1x2)

Matrix multiplication: (1x2) times (2x1) gives a single number (1x1)

$$\mathbf{w}^T\mathbf{x} = [w_1, w_2]\begin{bmatrix} x^1 \\ x^2 \end{bmatrix} = \mathbf{w_1 x^1 + w_2 x^2}$$

Perceptron using vectors

- One way of making predictions with samples that have 2 features, $x=(x^1, x^2)$
  - Define a 2D vector $w=(w_1,w_2)$ of 2 trainable parameters

Vector (2x1)

T is transpose op., makes it (1x2)

  - Prediction is: $\mathbf{f(x_i)=sign(w_1 x_i^1 + w_2 x_i^2) = sign(w^T x)}$

Matrix multiplication: (1x2) times (2x1)

Vector (2x1)

- Perceptron Algorithm
  - **Set initial values of vector w(random guess)**
  - **Loop:**
  - **Present a sample x_i and predict $f(x_i)=sign(w^T x)$**
    - **Compare true class y_i with predicted class $f(x_i)$**
    - **If prediction is right, go to next sample (i=i+1)**
    - **If prediction is wrong (incl. $f(x_i)==0$, no prediction), update w:**
      - $w_{new}=w_{old} + c[y_i - f(x_i)]x_i$

vector vector vector

HW1

HW1 preview

- HW1 will be announced after add/drop, on Wednesday (8/28)
- You will have until 9/10, 5pm to complete it

- It will involve a simple dataset with 2 classes and 2 features
- Your task will be to explore the process of training a perceptron (the vector version) from previous lecture slide
- Python libraries to be used:
  - Pandas (reading in a csv file)
  - Matplotlib (plotting diagrams of training progress)
  - Numpy (storing vectors, doing the math with them)
  - **ML libraries (e.g. sklearn, pytorch, tensorflow, others) not allowed**
- The underlying goal of this simple HW is to bring everyone up to speed with python and its basic libraries for routine ML tasks like reading input, plotting, etc.
- If you are not confident with python, practice it before next Wednesday

Summary

- What we have seen is supervised learning:
  - There is a true class (e.g. type of fruit) that is unknown and should be predicted
- Supervised learning is a major type of machine learning
  - Includes self-supervised learning where the true class comes from the input features
- Other important types of ML are:
  - Unsupervised learning (e.g. cluster objects together)
  - Reinforcement learning (learn to choose good action, with distant reward, e.g. chess)