10-607 Computational Foundations for Machine Learning

Perceptron Mistake Bound & Computational Complexity

Instructor: Pat Virtue

Plan

Perceptron Algorithm

Perceptron Mistake Bound Theory

- Background: projections, distances, and margin
- Proof of mistake bound as an example application

Computational Complexity

- How fast is your code/algorithm?
- Counting operations
- Big-O
- Complexity classes

Perceptron Algorithm

Sketch of algorithm

Initialize $\boldsymbol{w} = \boldsymbol{0}$

Loop

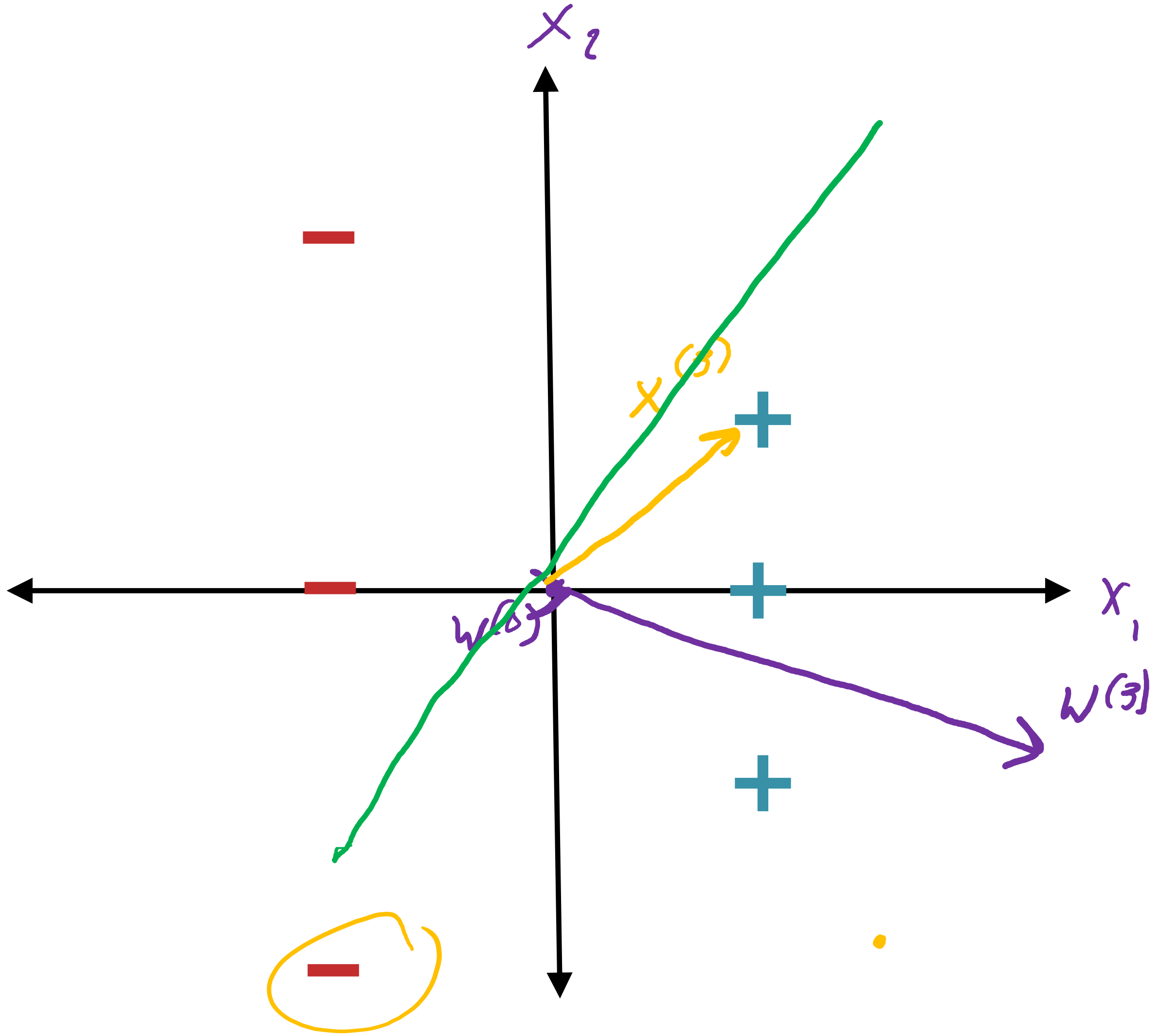
Given a point $\boldsymbol{x}$ predict $\hat{y} = sign(\boldsymbol{w}^T\boldsymbol{x})$

Given actual label y:

If $y \neq \hat{y}$

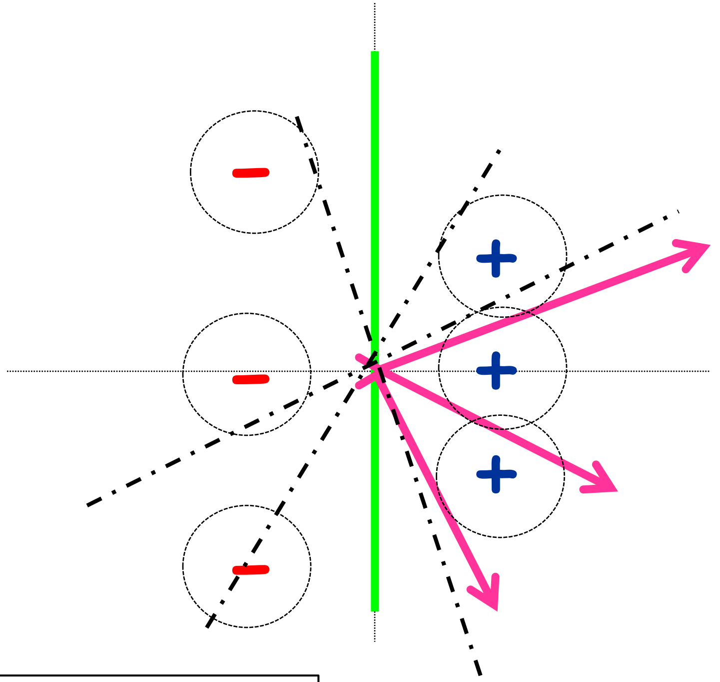
If $y = +1, \quad \boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{x}$

If $y = -1, \quad \boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{x}$



Perceptron Algorithm: Example

Example: $(-1,2)\ -$ ✗
$(1,0)\ +$ ✓
$(1,1)\ +$ ✗
$(-1,0)\ -$ ✓
$(-1,-2)\ -$ ✗
$(1,-1)\ +$ ✓



$w_1 = (0,0)$

$w_2 = w_1 - (-1,2) = (1,-2)$

$w_3 = w_2 + (1,1) = (2,-1)$

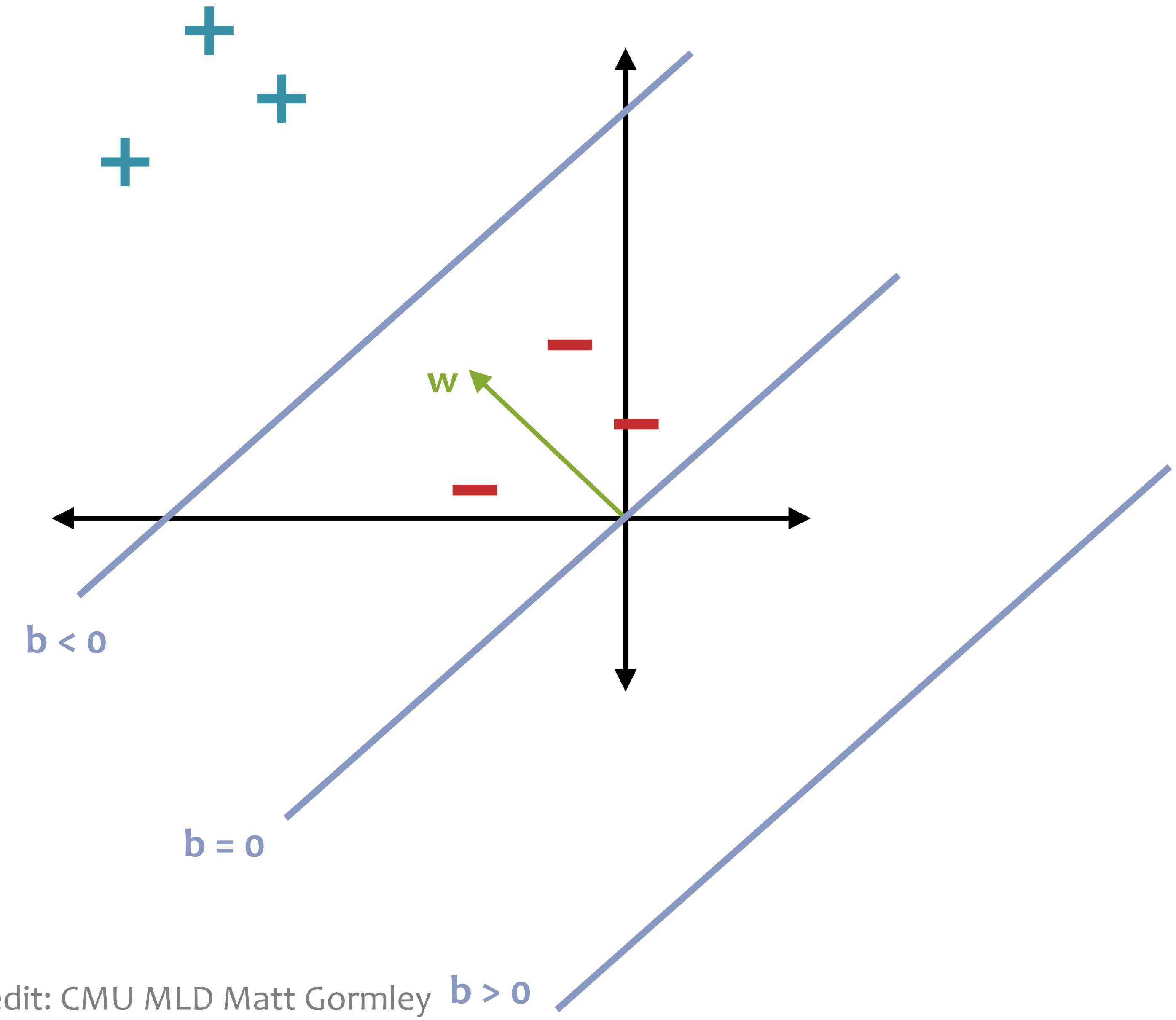
$w_4 = w_3 - (-1,-2) = (3,1)$

Perceptron Algorithm: (without the bias term)

- Set t=1, start with all-zeroes weight vector w_1.
- Given example x, predict positive iff $w_t \cdot x \geq 0$.
- On a mistake, update as follows:
  - Mistake on positive, update $w_{t+1} \leftarrow w_t + x$
  - Mistake on negative, update $w_{t+1} \leftarrow w_t - x$

Slide credit: CMU MLD Nina Balcan

Intercept Term



Q: Why do we need an intercept term?

A: It shifts the decision boundary off the origin

Q: What should happen to b during the perceptron algorithm

A: Two cases

1. Increasing b shifts the decision boundary towards the negative side
2. Decreasing b shifts the decision boundary towards the positive side

Slide credit: CMU MLD Matt Gormley

Perceptron Algorithm

Sketch of algorithm

Initialize $\boldsymbol{w} = \boldsymbol{0}$

Loop

  Given a point $\boldsymbol{x}$ predict $\hat{y} = sign(\boldsymbol{w}^T \boldsymbol{x})$

  Given actual label y:

  If true $y \neq \hat{y}$

    If true $y = +1, \quad \boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{x}$

    If true $y = -1, \quad \boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{x}$

Perceptron Algorithm

Sketch of algorithm

Initialize $\boldsymbol{\theta} = \mathbf{0}$

Loop

Given a point $\boldsymbol{x}$ predict $\hat{y} = sign(\boldsymbol{\theta}^T \boldsymbol{x})$

Given actual label y:

If true $y \neq \hat{y}$

If true $y = +1, \quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{x}$

If true $y = -1, \quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \boldsymbol{x}$

Perceptron Algorithm

Learning for Perceptron if we have a fixed training dataset, D.

Algorithm 1 Perceptron Learning Algorithm

---

procedure PERCEPTRON($\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$)

  $\boldsymbol{\theta} \leftarrow \mathbf{0}$ ▷ I… arameters

  while not converged **do**

    for $i \in \{1, 2, \ldots, N\}$ **do** ▷ For each example

      $\hat{y} \leftarrow \text{sign}(\boldsymbol{\theta}^T \mathbf{x}^{(i)})$ ▷ Predict

      if $\hat{y} \neq y^{(i)}$ **then** ▷ If mistake

        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(i)} \mathbf{x}^{(i)}$ ▷ Update parameters

  return $\boldsymbol{\theta}$

Slide credit: CMU MLD Matt Gormley

Perceptron Algorithm

Learning for Perceptron if we have a fixed training dataset, D.

Algorithm 1 Perceptron Learning Algorithm

1: **procedure** PERCEPTRON($\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$)
2: $\quad \boldsymbol{\theta} \leftarrow \mathbf{0}$ ▷ I… arameters
3: $\quad$ **while** not converged **do**
4: $\quad\quad$ **for** $i \in \{1, 2, \ldots, N\}$ **do** ▷ For each example
5: $\quad\quad\quad \hat{y} \leftarrow \text{sign}(\boldsymbol{\theta}^T \mathbf{x}^{(i)})$ ▷ Predict
6: $\quad\quad\quad$ **if** $\hat{y} \neq y^{(i)}$ **then** ▷ If mistake
7: $\quad\quad\quad\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(i)} \mathbf{x}^{(i)}$
8: $\quad$ **return** $\boldsymbol{\theta}$

Implementation Trick: same behavior as our "*add on positive mistake and subtract on negative mistake*" version, because $y^{(i)}$ takes care of the sign

Slide credit: CMU MLD Matt Gormley

Plan

Perceptron Algorithm

Perceptron Mistake Bound Theory

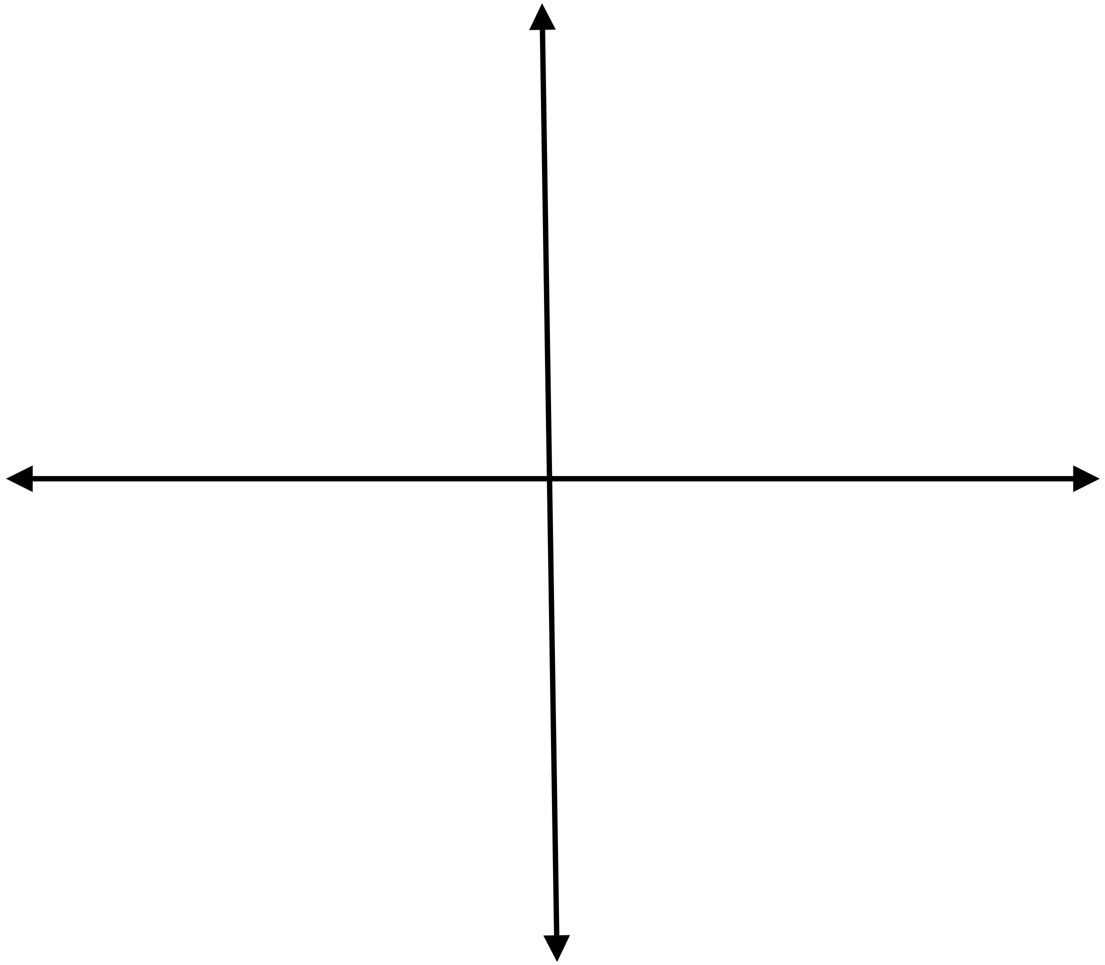
- Background: projections, distances, and margin
- Proof of mistake bound as an example application

Computational Complexity

- How fast is your code/algorithm?
- Counting operations
- Big-O
- Complexity classes

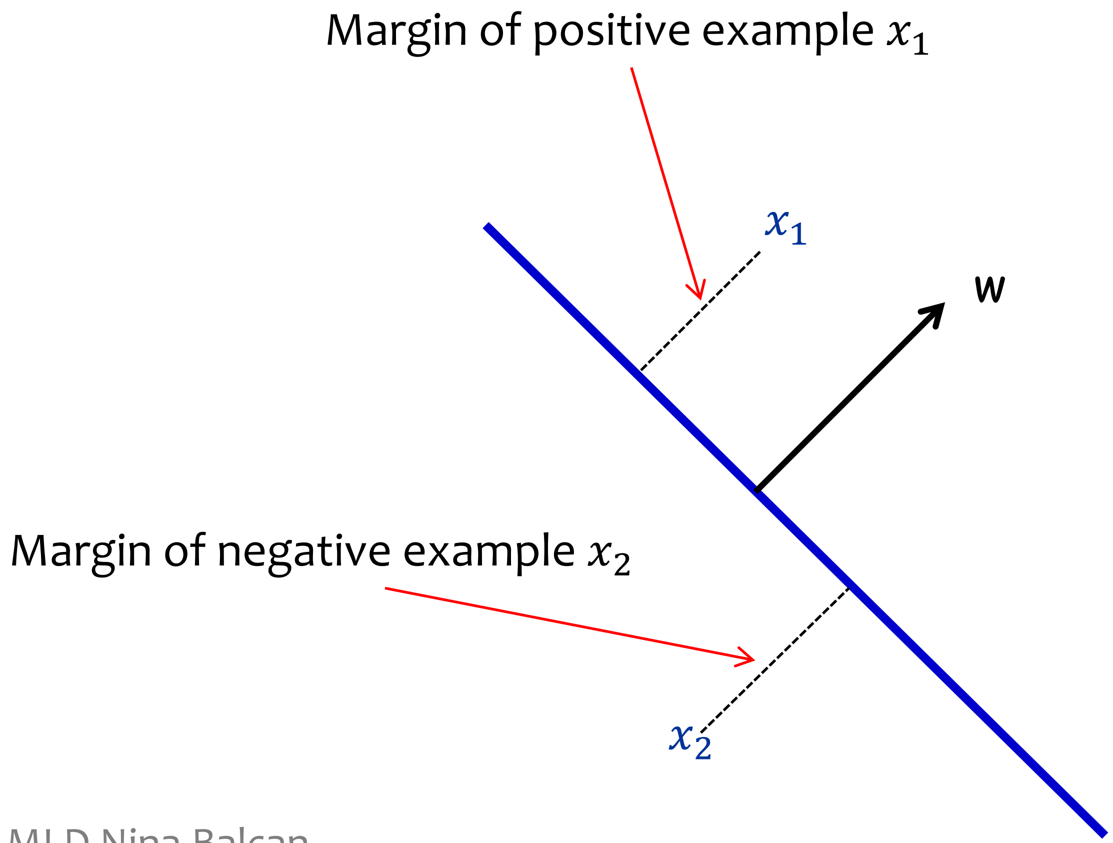
Projection

Projection of $\mathbf{u}$ on to $\mathbf{v}$



Geometric Margin

Definition: The margin of example x w.r.t. a linear sep. w is the distance from x to the plane $w \cdot x = 0$ (or the negative if on wrong side)

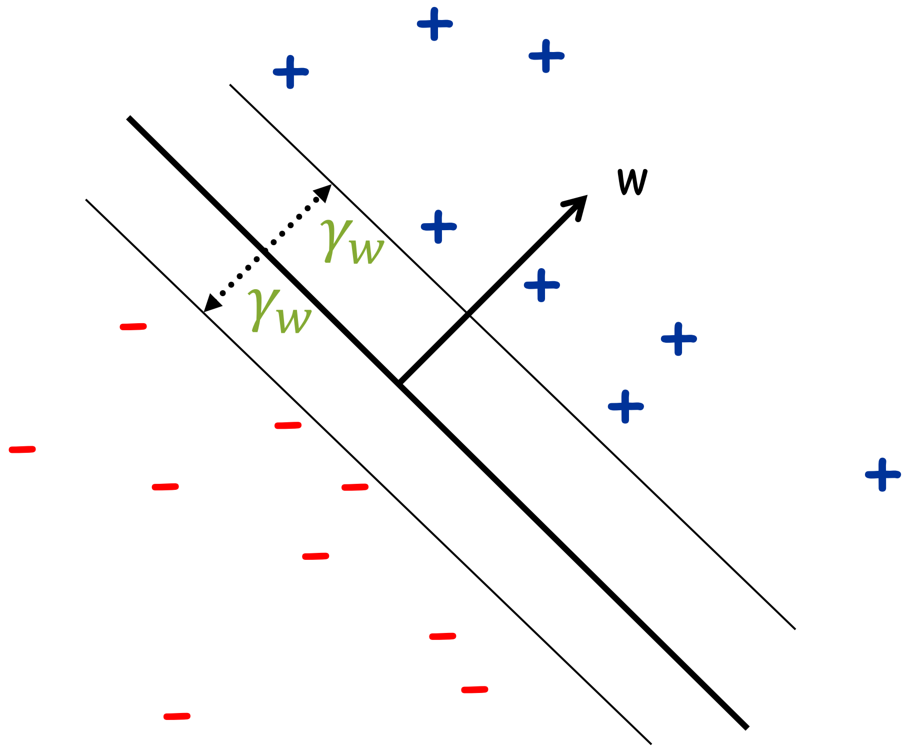


Slide credit: CMU MLD Nina Balcan

Geometric Margin

Definition: The margin of example x w.r.t. a linear sep. w is the distance from x to the plane $w \cdot x = 0$ (or the negative if on wrong side)

Definition: The margin γ_w of a set of examples S wrt a linear separator w is the smallest margin over points $x \in S$.



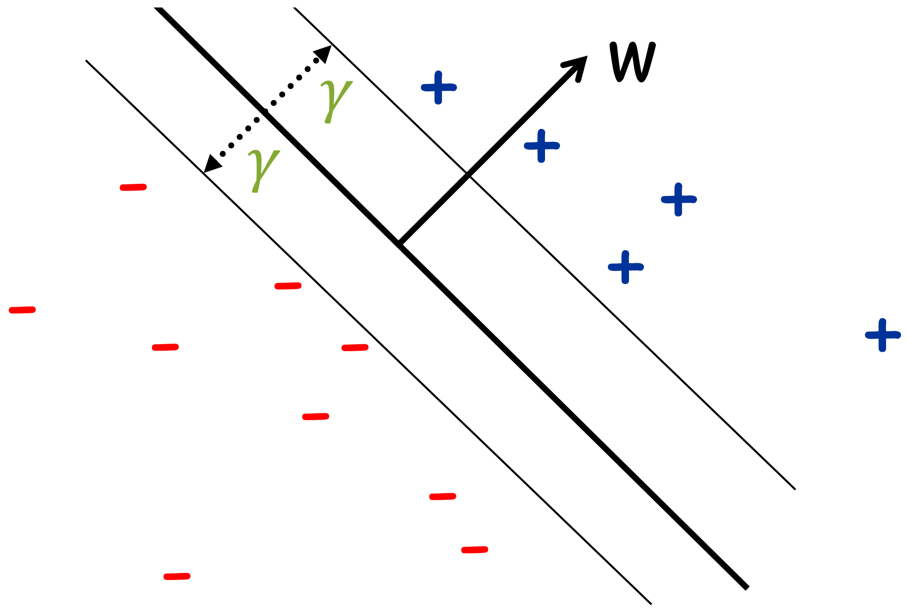
Slide credit: CMU MLD Nina Balcan

Geometric Margin

Definition: The margin of example x w.r.t. a linear sep. w is the distance from x to the plane $w \cdot x = 0$ (or the negative if on wrong side)

Definition: The margin γ_w of a set of examples S wrt a linear separator w is the smallest margin over points $x \in S$.

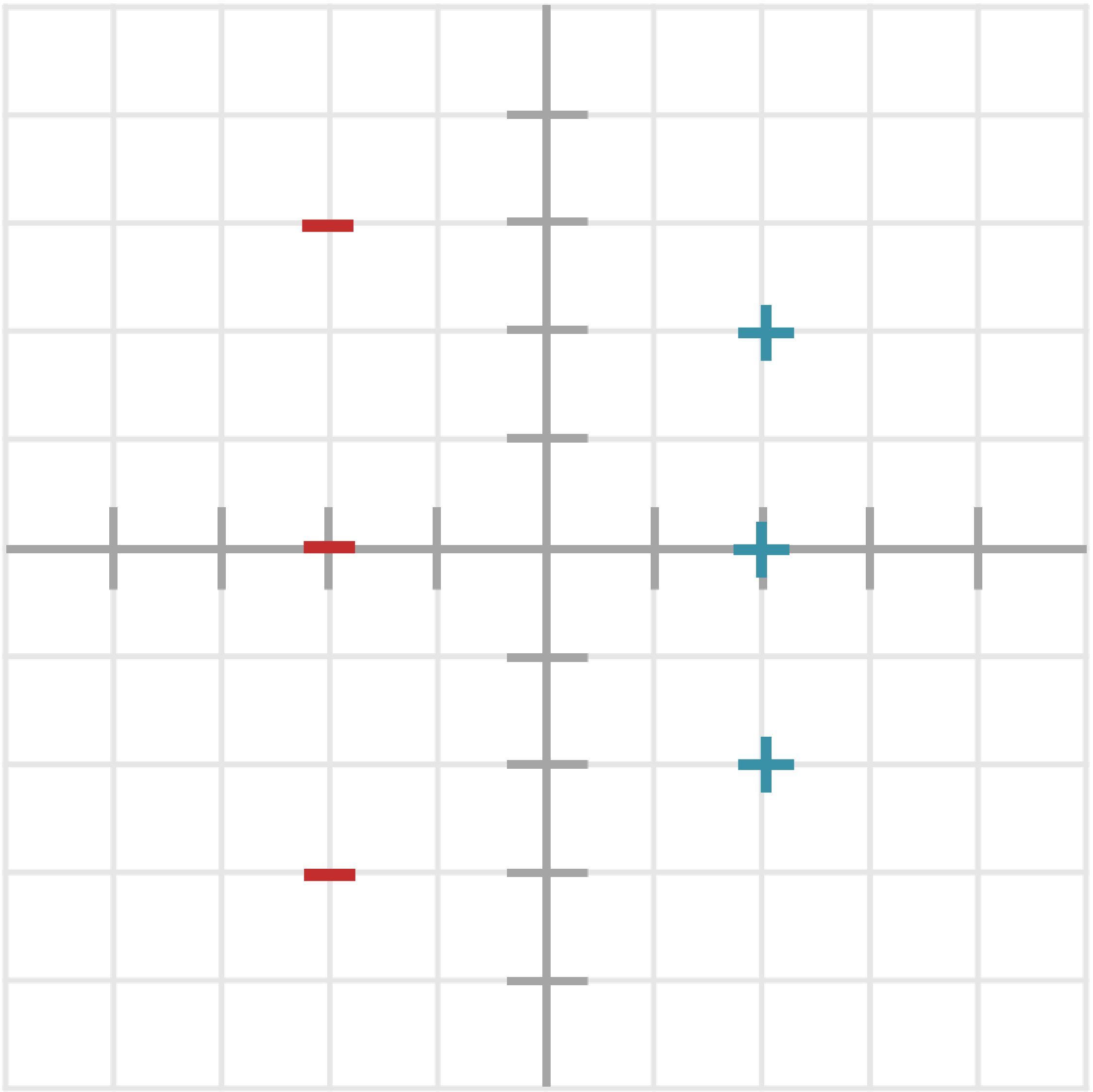
Definition: The margin γ of a set of examples S is the maximum γ_w over all linear separators w.



Slide credit: CMU MLD Nina Balcan

Geometric Margin

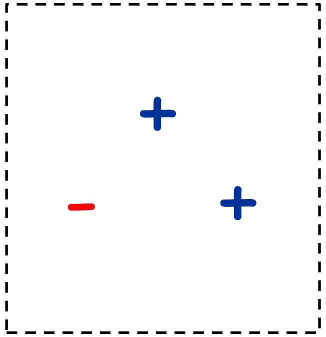
What is the margin for this dataset?



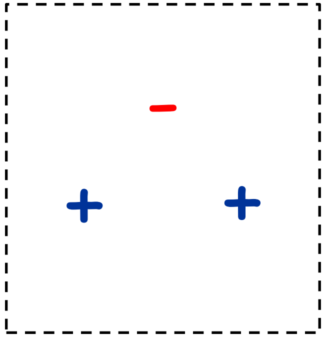
Linear Separability

Def: For a **binary classification** problem, a set of examples S is **linearly separable** if there exists a linear decision boundary that can separate the points

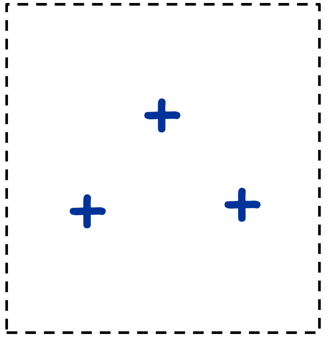
Case 1:



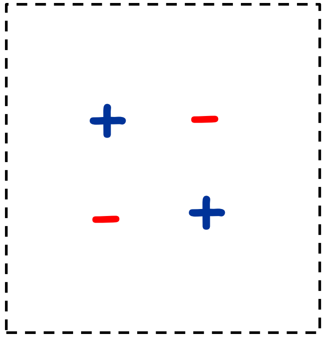
Case 2:



Case 3:



Case 4:



Slide credit: CMU MLD Matt Gormley

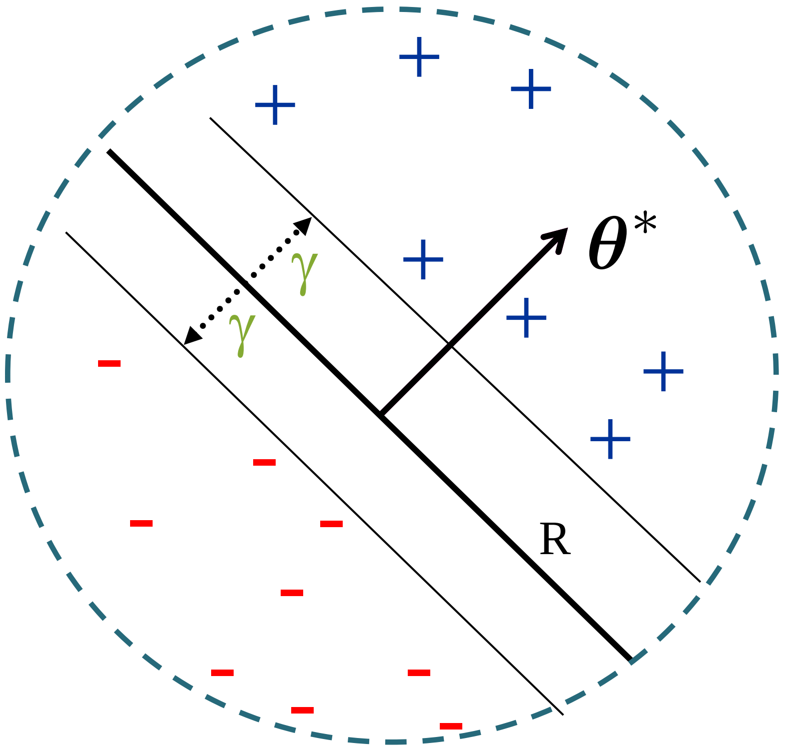
ANALYSIS OF PERCEPTRON

Analysis: Perceptron

Perceptron Mistake Bound

Guarantee: If data has margin γ and all points inside a ball of radius R, then Perceptron makes $\leq (R/\gamma)^2$ mistakes.

(Normalized margin: multiplying all points by 100, or dividing all points by 100, doesn't change the number of mistakes; algo is invariant to scaling.)



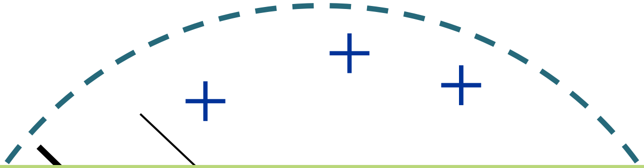
Slide credit: CMU MLD Nina Balcan and Matt Gormley

Analysis: Perceptron

Perceptron Mistake Bound

Guarantee: If data has margin γ and all points inside a ball of radius R, then Perceptron makes $\leq (R/\gamma)^2$ mistakes.

(Normalized margin: multiplying all points by 100, or dividing all points by 100, doesn't change the number of mistakes; algo is invariant to scaling.)



Def: We say that the perceptron algorithm has **converged** if it stops making mistakes on the training data (perfectly classifies the training data).

Main Takeaway: For **linearly separable** data, if the perceptron algorithm cycles repeatedly through the data, it will **converge** in a finite # of steps.

Analysis: Perceptron

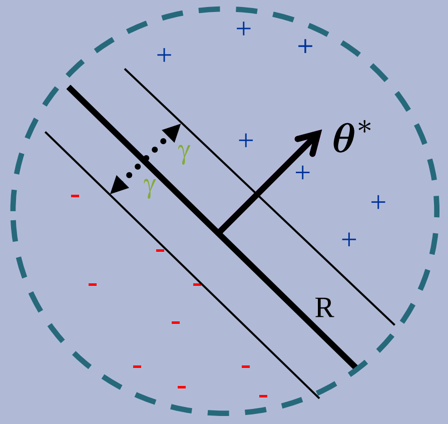
Perceptron Mistake Bound

Theorem 0.1 (Block (1962), Novikoff (1962)).
Given dataset: $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$.
Suppose:

1. *Finite size inputs:* $||x^{(i)}|| \leq R$
2. *Linearly separable data:* $\exists \boldsymbol{\theta}^*$ *s.t.* $||\boldsymbol{\theta}^*|| = 1$ *and* $y^{(i)}(\boldsymbol{\theta}^* \cdot \mathbf{x}^{(i)}) \geq \gamma, \forall i$

Then: The number of mistakes made by the Perceptron algorithm on this dataset is

$$k \leq (R/\gamma)^2$$



Analysis: Percept

Perceptron Mistake Boun

Theorem 0.1 (Block (1962), Novikoff (19

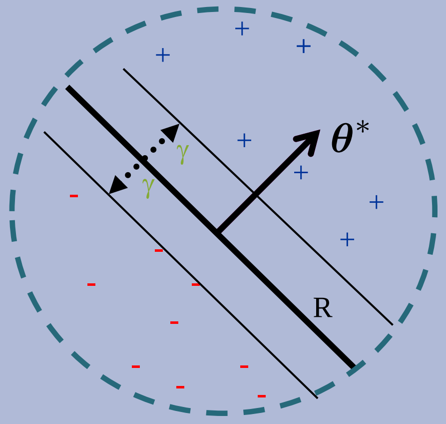
Given dataset: $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$

Suppose:

1. *Finite size inputs:* $||x^{(i)}|| \leq R$
2. *Linearly separable data:* $\exists \boldsymbol{\theta}^*$ *s.t.* $||\boldsymbol{\theta}^*|| = 1$ *and* $y^{(i)}(\boldsymbol{\theta}^* \cdot \mathbf{x}^{(i)}) \geq \gamma, \forall i$

Then: The number of mistakes made by the Perceptron algorithm on this dataset is

$$k \leq (R/\gamma)^2$$



Common Misunderstanding:

The **radius** is **centered at the origin**, not at the center of the points.

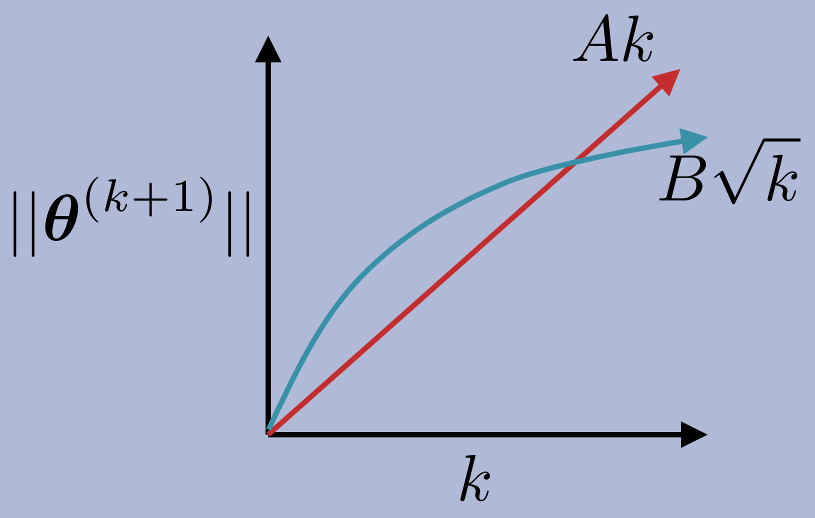
Slide credit: CMU MLD Nina Balcan and Matt Gormley

Analysis: Perceptron

Proof of Perceptron Mistake Bound:

We will show that there exist constants A and B s.t.

$$Ak \leq ||\boldsymbol{\theta}^{(k+1)}|| \leq B\sqrt{k}$$



Slide credit: CMU MLD Nina Balcan and Matt Gormley

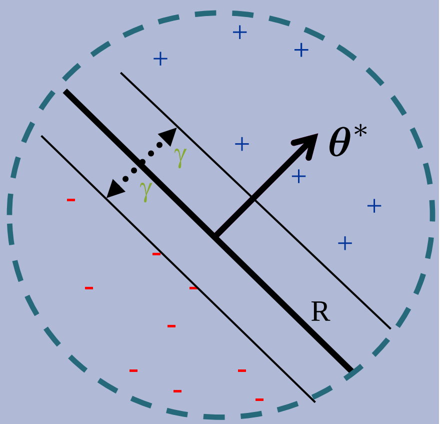
Analysis: Perceptron

Theorem 0.1 (Block (1962), Novikoff (1962)).
Given dataset: $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$.
Suppose:

1. *Finite size inputs:* $||x^{(i)}|| \leq R$
2. *Linearly separable data:* $\exists \boldsymbol{\theta}^*$ s.t. $||\boldsymbol{\theta}^*|| = 1$ *and* $y^{(i)}(\boldsymbol{\theta}^* \cdot \mathbf{x}^{(i)}) \geq \gamma, \forall i$

Then: The number of mistakes made by the Perceptron algorithm on this dataset is

$$k \leq (R/\gamma)^2$$



Algorithm 1 Perceptron Learning Algorithm (Online)

procedure PERCEPTRON($\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots\}$)
  $\boldsymbol{\theta} \leftarrow \mathbf{0}$, $k = 1$ ▷ Initialize parameters
  for $i \in \{1, 2, \ldots\}$ **do** ▷ For each example
    if $y^{(i)}(\boldsymbol{\theta}^{(k)} \cdot \mathbf{x}^{(i)}) \leq 0$ **then** ▷ If mistake
      $\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} + y^{(i)}\mathbf{x}^{(i)}$ ▷ Update parameters
      $k \leftarrow k + 1$
  return $\boldsymbol{\theta}$

Slide credit: CMU MLD Nina Balcan and Matt Gormley

Analysis: Perceptron

Proof of Perceptron Mistake Bound:

Part 1: for some A, $Ak \leq ||\boldsymbol{\theta}^{(k+1)}||$

$$\boldsymbol{\theta}^{(k+1)} \cdot \boldsymbol{\theta}^* = (\boldsymbol{\theta}^{(k)} + y^{(i)}\mathbf{x}^{(i)})\boldsymbol{\theta}^*$$

by Perceptron algorithm update

$$= \boldsymbol{\theta}^{(k)} \cdot \boldsymbol{\theta}^* + y^{(i)}(\boldsymbol{\theta}^* \cdot \mathbf{x}^{(i)})$$

$$\geq \boldsymbol{\theta}^{(k)} \cdot \boldsymbol{\theta}^* + \gamma$$

by assumption

$$\Rightarrow \boldsymbol{\theta}^{(k+1)} \cdot \boldsymbol{\theta}^* \geq k\gamma$$

by induction on k since $\theta^{(1)} = \mathbf{0}$

$$\Rightarrow ||\boldsymbol{\theta}^{(k+1)}|| \geq k\gamma$$

since $||\mathbf{w}|| \times ||\mathbf{u}|| \geq \mathbf{w} \cdot \mathbf{u}$ and $||\theta^*|| = 1$

Cauchy-Schwartz inequality

Slide credit: CMU MLD Matt Gormley

Analysis: Perceptron

Proof of Perceptron Mistake Bound:

Part 2: for some B, $||\boldsymbol{\theta}^{(k+1)}|| \leq B\sqrt{k}$

$$||\boldsymbol{\theta}^{(k+1)}||^2 = ||\boldsymbol{\theta}^{(k)} + y^{(i)}\mathbf{x}^{(i)}||^2$$

by Perceptron algorithm update

$$= ||\boldsymbol{\theta}^{(k)}||^2 + (y^{(i)})^2||\mathbf{x}^{(i)}||^2 + 2y^{(i)}(\boldsymbol{\theta}^{(k)} \cdot \mathbf{x}^{(i)})$$

$$\leq ||\boldsymbol{\theta}^{(k)}||^2 + (y^{(i)})^2||\mathbf{x}^{(i)}||^2$$

since kth mistake $\Rightarrow y^{(i)}(\boldsymbol{\theta}^{(k)} \cdot \mathbf{x}^{(i)}) \leq 0$

$$= ||\boldsymbol{\theta}^{(k)}||^2 + R^2$$

since $(y^{(i)})^2||\mathbf{x}^{(i)}||^2 = ||\mathbf{x}^{(i)}||^2 = R^2$ by assumption and $(y^{(i)})^2 = 1$

$$\Rightarrow ||\boldsymbol{\theta}^{(k+1)}||^2 \leq kR^2$$

by induction on k since $(\theta^{(1)})^2 = 0$

$$\Rightarrow ||\boldsymbol{\theta}^{(k+1)}|| \leq \sqrt{k}R$$

Slide credit: CMU MLD Matt Gormley

Analysis: Perceptron

Proof of Perceptron Mistake Bound:

Part 3: Combining the bounds finishes the proof.

$$k\gamma \leq ||\boldsymbol{\theta}^{(k+1)}|| \leq \sqrt{k}R$$

$$\Rightarrow k \leq (R/\gamma)^2$$

The total number of mistakes must be less than this

Slide credit: CMU MLD Matt Gormley

Plan

Perceptron Algorithm

Perceptron Mistake Bound Theory

- Background: projections, distances, and margin
- Proof of mistake bound as an example application

Computational Complexity

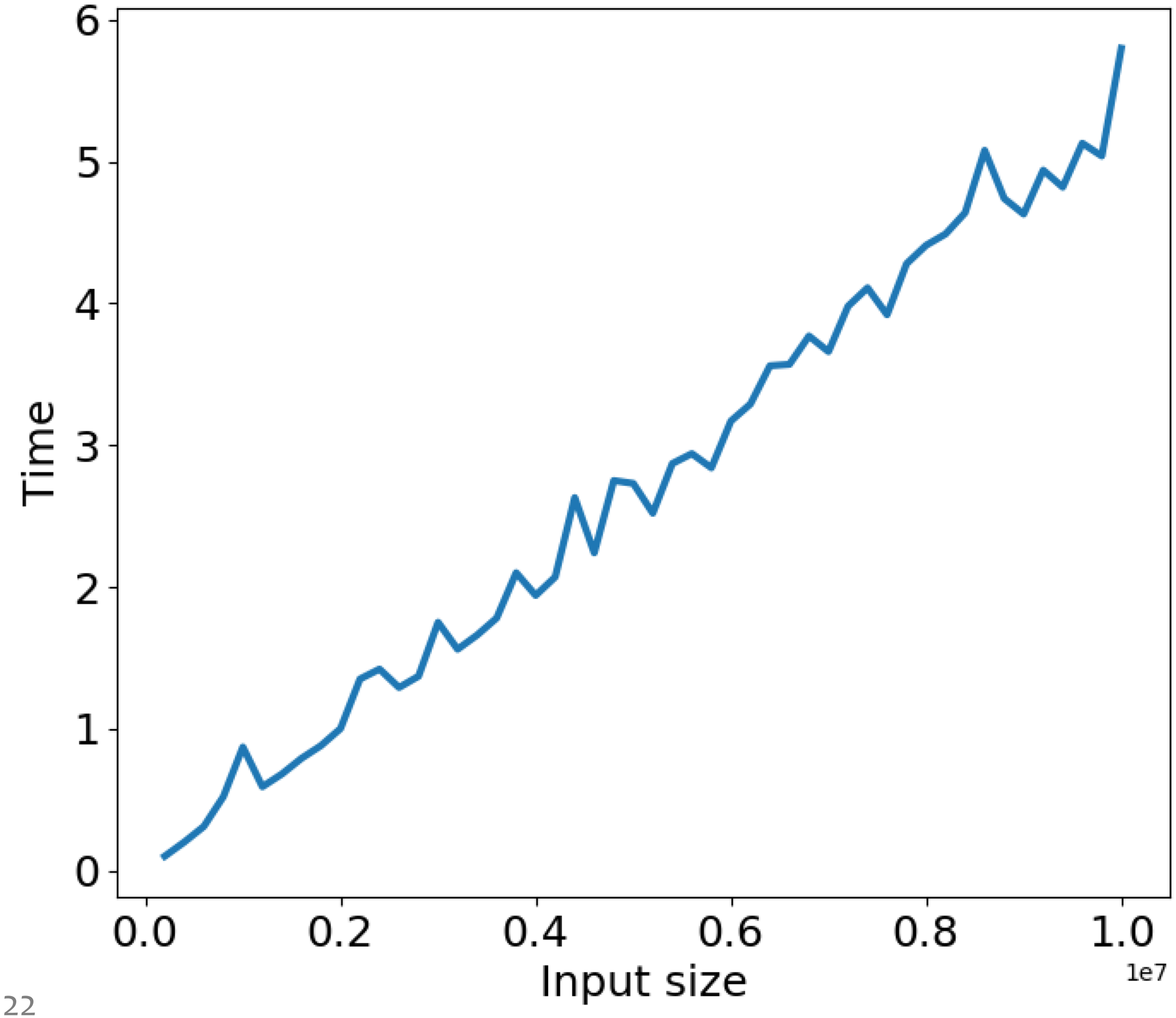
- How fast is your code/algorithm?
- Counting operations
- Big-O
- Complexity classes

How fast is this code?

```
int search(int x, int[] A, int n)
{
  for (int i = 0; i < n; i++)
  {
    if (A[i] == x) {
      return i;
    }
  }
  return -1;
}
```

Slide credit: CMU 15-122

How fast is this code?



Slide credit: CMU 15-122

Need a better way to measure

Permanent

- Independent of hardware or other running processes, etc.

General

- Applicable to a large class of programs/algorithms/problems
- Resources: time, space

Mathematically rigorous

Useful

- Not for actual run time,
- But help us select best algorithm for the task

How many statements are executed?

```
int search(int x, int[] A, int n)
{
  for (int i = 0; i < n; i++)
  {
    if (A[i] == x) {
      return i;
    }
  }
  return -1;
}
```

If x is not in A…
how times are these
statements executed?

	`i = 0`
	`i < n`
	`if (A[i] == x)`
	`i++`
	`return -1`

How many **operations** are executed?

```
int search(int x, int[] A, int n)
{
  for (int i = 0; i < n; i++)
  {
    if (A[i] == x) {
      return i;
    }
  }
  return -1;
}
```

If x is not in A...
how times **operations** are executed?

```
i = 0

i < n

if (A[i] == x)

i++

return -1
```

How many operations are executed?

How many program operations are required to compute:

- L2 norm of vector
- Vector dot product
- Frobenius norm of matrix
- Matrix-vector multiplication
- Matrix-matrix multiplication

```
def norm(a):
    ss=0
    for i in range(len(a)):
        ss = ss + a[i]*a[i]
    norm = np.sqrt(ss)
    return norm
```

Operations:

- Arithmetic operations (e.g. + or **)
- Logical operations (e.g., and)
- Comparison operations (e.g., <=)
- Structure accessing operations (e.g. array indexing like A[i])
- Simple assignment such as copying a value into a variable
- Calls to library functions that don't depend on size of input (e.g., print)
- Control Statements (e.g. if X>5)

Be careful with function calls that scale with the size of the input