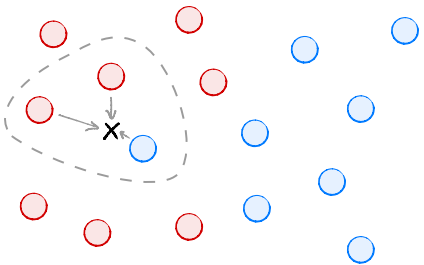


CSE 151A

Intro to Machine Learning

Lecture 02 – Part 01

What is Machine Learning?

Announcements

- First homework will be posted today (April 1).
- Due via Gradescope next Wednesday.
- Covers only today's lecture, basic probability.

What is Machine Learning?

- Computers can do things very quickly.

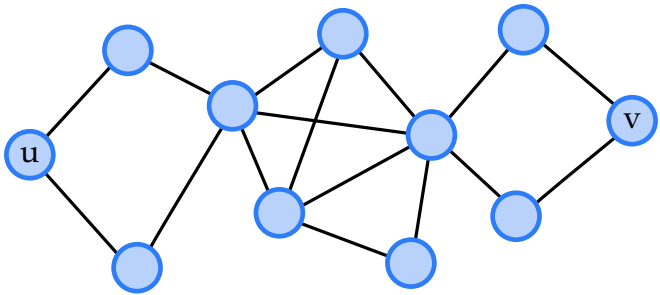
What is Machine Learning?

- Computers can do things very quickly.
- But must be given really specific instructions.

What is Machine Learning?

- Computers can do things very quickly.
- But must be given really specific instructions.
- **Problem**: Not all tasks are easy to dictate.

Example



What is the shortest path between *u* and *v*?

Example

How old is this person?

The Trick: Use Data

age = 28

age = 42

age = 63

age = 24

age = 37

age = 39

age = ?

age = 35

What is Machine Learning?

- Before: Computer is **told** how to do a task.
- Instead: **learn** how to do a task using data.

What is Machine Learning?

- Before: Computer is **told** how to do a task.
- Instead: **learn** how to do a task using data.
- We still have to **tell** the computer how to learn.

A **machine learning algorithm** is a set of precise instructions telling the computer how to learn from data.

A **machine learning algorithm** is a set of precise instructions telling the computer how to learn from data.

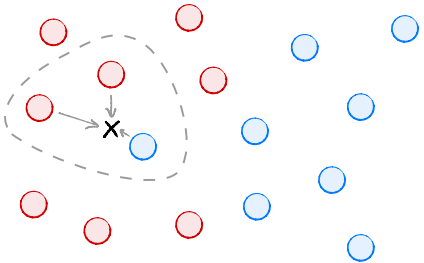
Spoiler: the algorithms are usually pretty simple. It's the **data** that does the real work.

Machine Learning Tasks

- Prediction
- Clustering
- Representation Learning
- Reinforcement Learning
- Anomaly Detection
- ...

Machine Learning Tasks

- **Prediction**
- Clustering
- Representation Learning
- Reinforcement Learning
- Anomaly Detection
- ...



CSE 151A

Intro to Machine Learning

Lecture 02 – Part 02

Prediction

Prediction

How old is this person?

Prediction

- **Given**: a 1024 × 1024 RGB image of a face.
- **Predict**: the age of the person.

Prediction Functions

- **Input Space**, $\mathcal{X}$ = set of all possible 1024 × 1024 images
- **Output Space**, $\mathcal{Y} = \mathbb{R}$
- We want a **prediction function**, $f : \mathcal{X} \to \mathcal{Y}$ which makes **accurate predictions**.

Regression

- **Regression**: when $\mathcal{Y} = \mathbb{R}$
- Example: Predict tomorrow's air quality index.
- Example: Predict someone's life expectancy.

Regression

- **Regression**: when $\mathcal{Y} = \mathbb{R}$
- Example: Predict tomorrow's air quality index.
- Example: Predict someone's life expectancy.
- What are suitable inputs ($\mathcal{X}$) in each case?

Classification

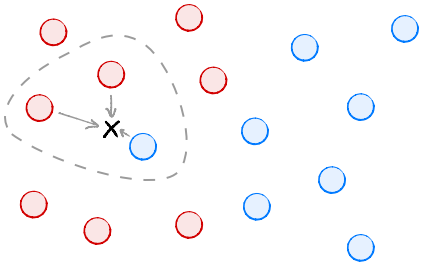
- **Classification**: when $\mathcal{Y}$ is a discrete set.
- **Binary classification**: $|\mathcal{Y}| = 2$
  - Example: Is this picture of a hot dog?
    $\mathcal{Y}$ = {Yes, No}.
- **Multiclass classification**: $|\mathcal{Y}| > 2$
  - Example: What food is in this picture?
    $\mathcal{Y}$ = {hot dog, pizza, sushi, ...}

Probability Estimation

- **Probability Estimation**: when $\mathcal{Y} = [0, 1]$
  - interpret output as a probability
- **Example**: credit card transaction.
- **Given**: details of transaction.
- **Predict**: prob. that the transaction is fraudulent.

Up next...

A simple learning algorithm for prediction.



CSE 151A

Intro to Machine Learning

Lecture 02 – Part 03

Nearest Neighbors

Example: NBA

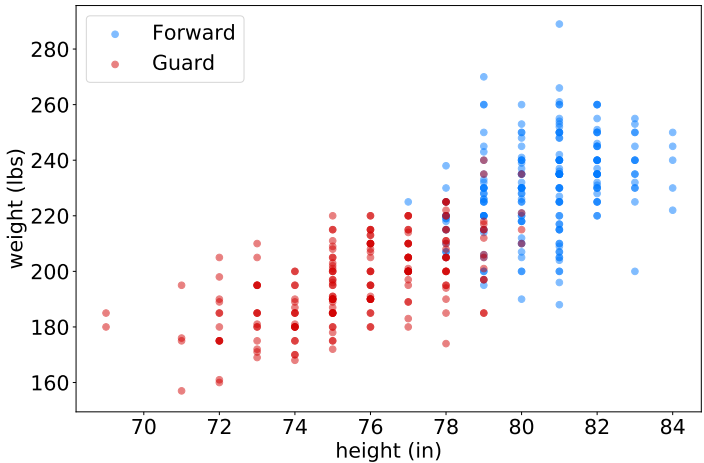
Guards → Forwards → Centers

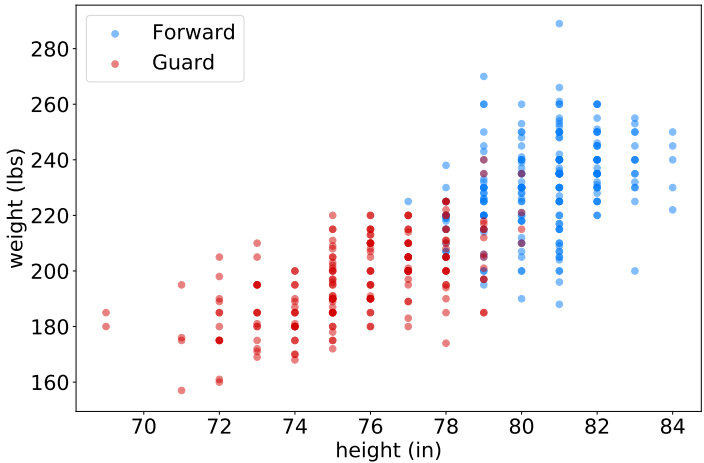
- **Given** a new player's height and weight.
- **Classify** them as a guard, forward, or center.


23
30

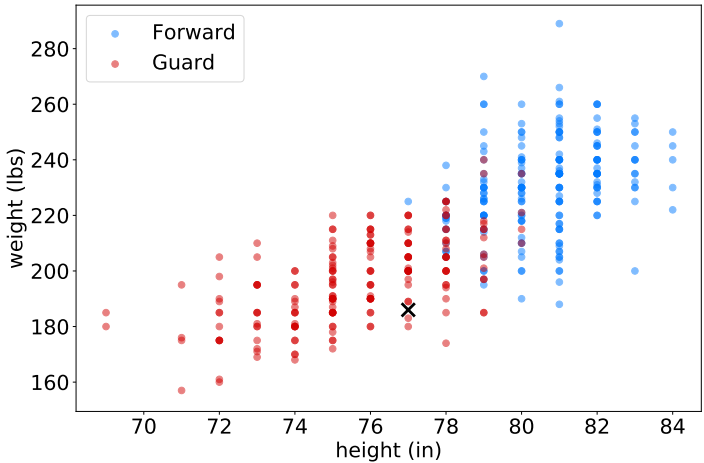
↑
forward

↑
guard

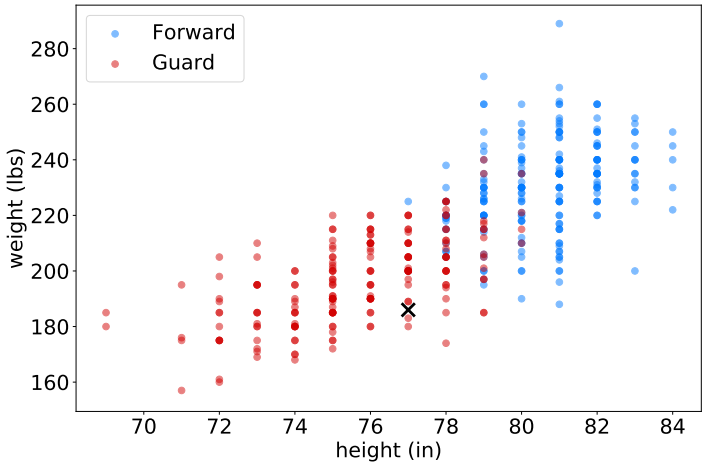

Forward
Guard
weight (lbs)
height (in)
160
180
200
220
240
260
280
70
72
74
76
78
80
82
84



A new player is 77 inches tall and 186 pounds.
What position do they play?

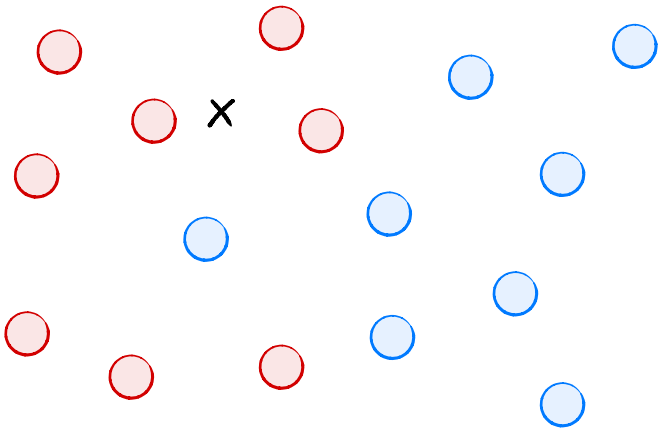


A new player is 75 inches tall and 240 pounds.
What position do they play?

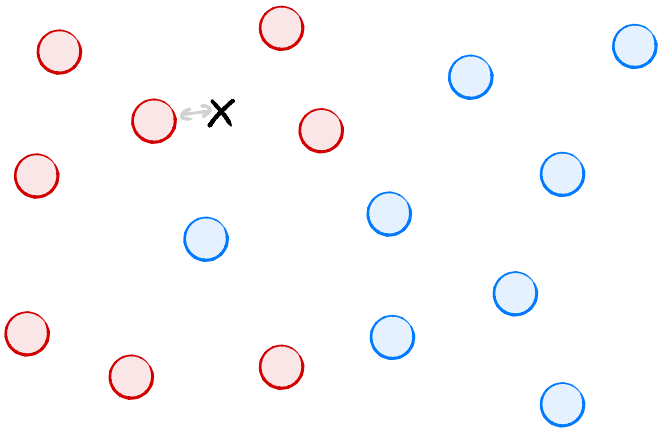


A new player is 75 inches tall and 240 pounds.
What position do they play? **Guard**.

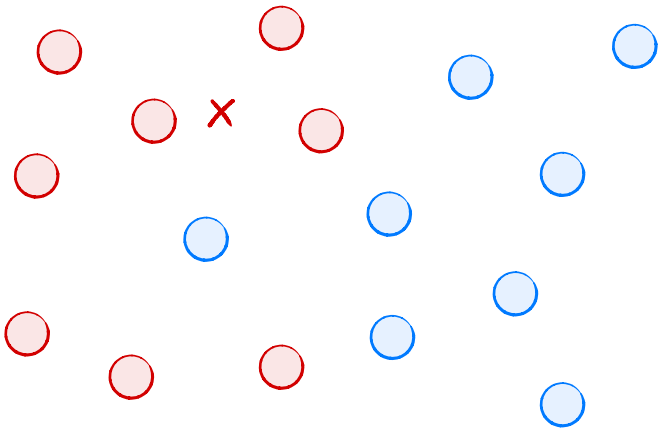
Nearest Neighbor Classification



Nearest Neighbor Classification

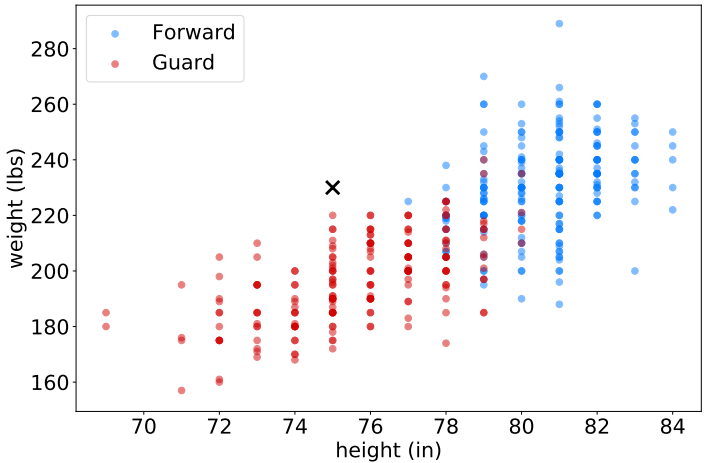


Nearest Neighbor Classification

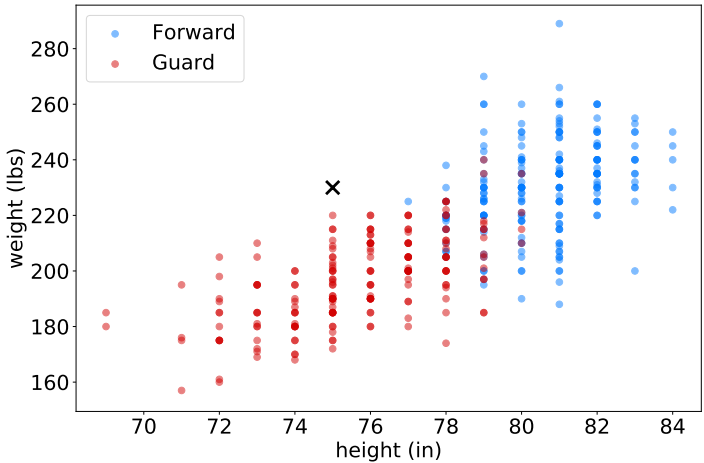


Nearest Neighbor Classification

- **Given**: an input data point.
- **Output**: the class label of the **nearest** point in data set.

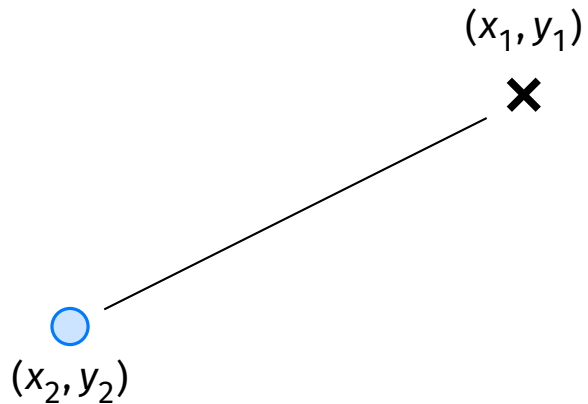


A new player is 75 inches tall and 230 pounds.
What position do they play?

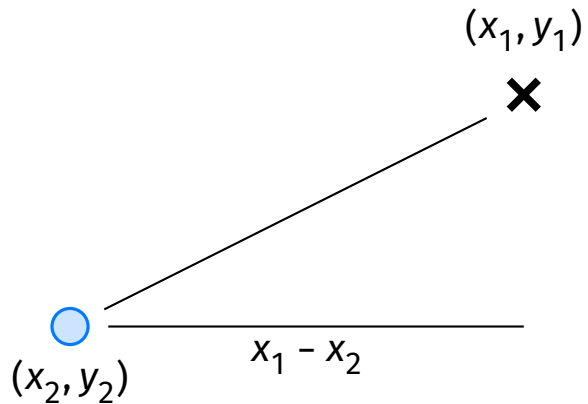


A new player is 75 inches tall and 230 pounds.
What position do they play? **Forward**.

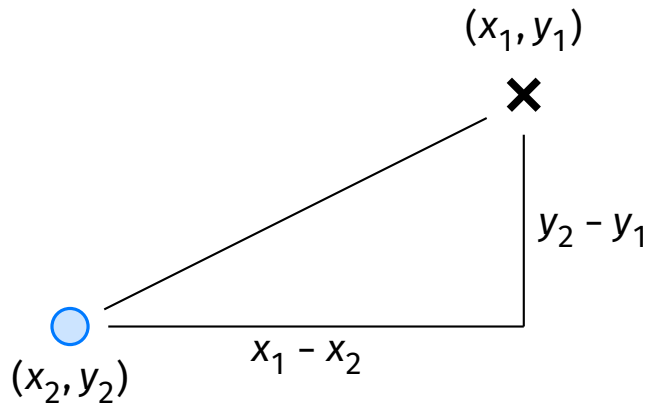
The (Euclidean) Distance



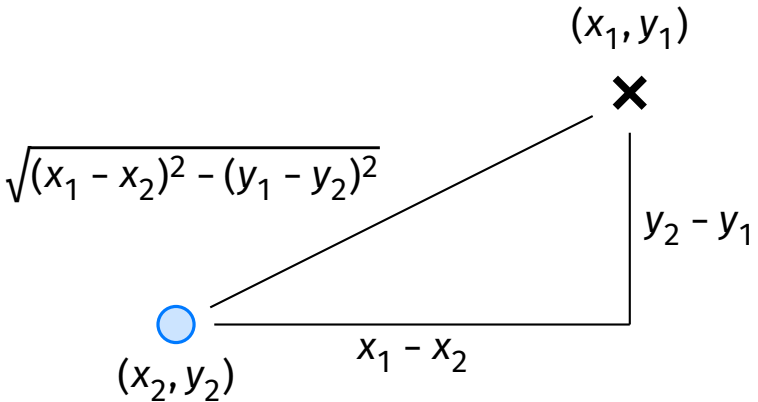
The (Euclidean) Distance

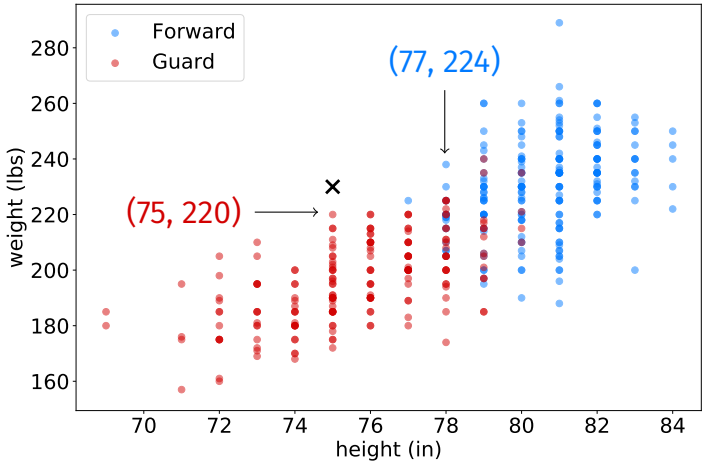


The (Euclidean) Distance

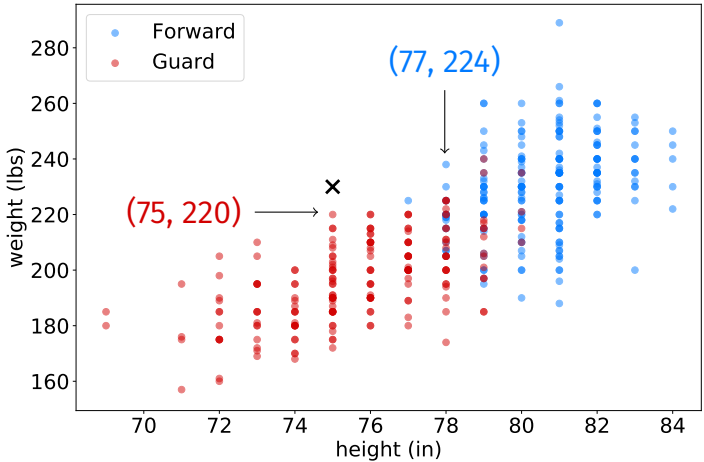


The (Euclidean) Distance





A new player is 75 inches tall and 230 pounds.
What position do they play?



A new player is 75 inches tall and 230 pounds.
What position do they play? **Forward**.

Calculating Distance

- Remember, the input: $(75, 230)$.
- Distance between $(75, 230)$ and $(75, 220)$:

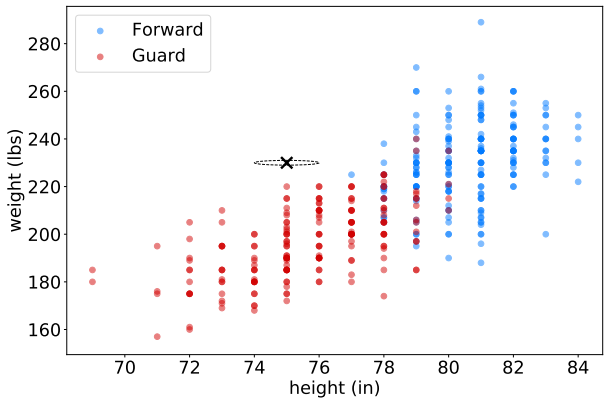
$$\sqrt{(75 - 75)^2 - (230 - 220)^2} = \sqrt{0 - 10^2} = 10$$

- Distance between $(75, 230)$ and $(77, 224)$:

$$\sqrt{(75 - 77)^2 - (230 - 224)^2} = \sqrt{2^2 + 6^2} = \sqrt{40} \approx \mathbf{6.3}$$

Scale Matters!

- Height and weight are on **different** scales.



Scale Matters!

- Height and weight are on **different** scales.
- **Solution**: put them on the same scale.
- **Question:** How might you do this?

Standard Units

- To standardize a height, h:

$$\frac{h - (\text{mean height})}{(\text{STD of heights})}$$

- To standardize a weight, w:

$$\frac{w - (\text{mean weight})}{(\text{STD of weights})}$$

Example

- What is $(75, 230)$ in standard units?
- Standardize the height and weight separately.

Example

- What is 75 inches in standard units?
- Heights: mean = 79.1, STD = 3.45

$$\frac{75 - 79.1}{4.35} \approx -1.18$$

Example

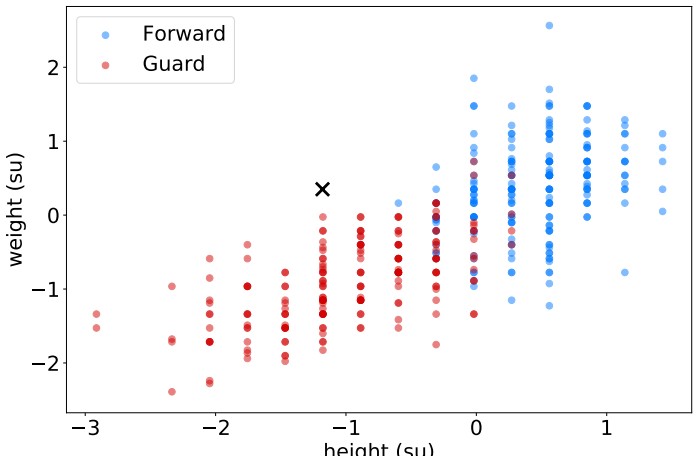
- What is 230 pounds in standard units?
- Weights: mean = 220.7, STD = 26.6

$$\frac{230 - 220.7}{26.6} \approx 0.35$$

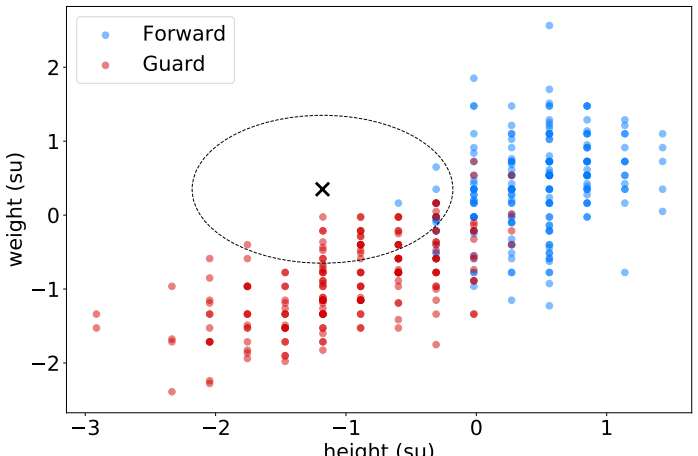
Example

- What is $(75, 230)$ in standard units?
- Answer: $(-1.18, 0.35)$

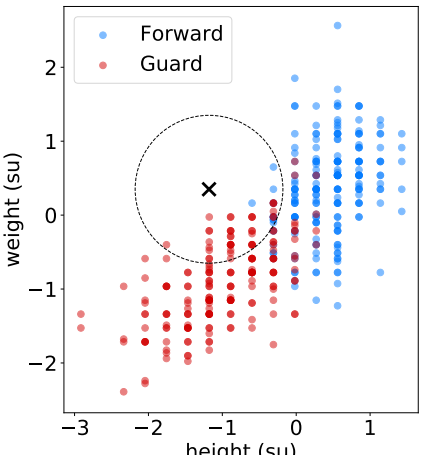
Standard Units



Standard Units



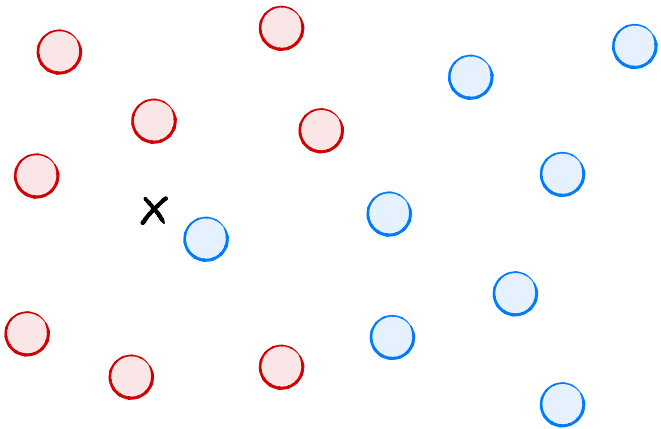
Standard Units



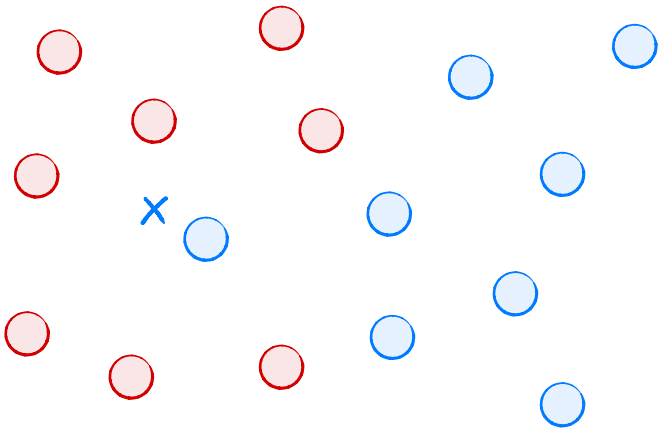
Summary: Nearest Neighbor

- Nearest neighbor is very simple.
- Given a new input, predict the class of closest point in data set.
- Often makes sense to **standardize** data first.
- But nearest neighbor has a problem...

Nearest Neighbor Classification



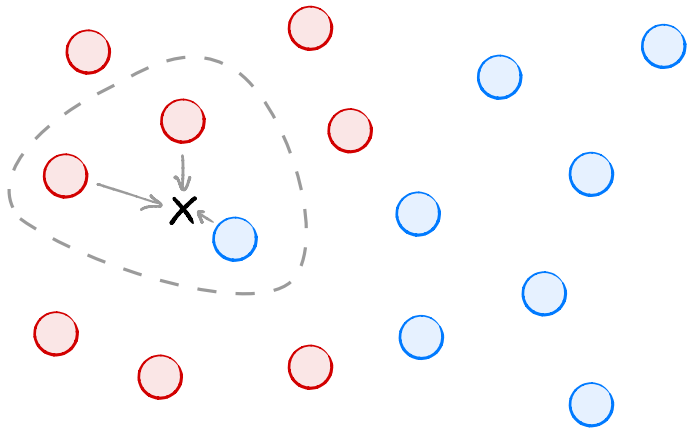
Nearest Neighbor Classification



k-Nearest Neighbors

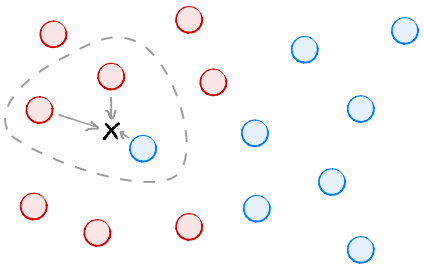
- Nearest neighbor is sensitive.
- Solution: have the nearest few neighbors “vote”.
- **Given**: an input data point, parameter k.
- **Output**: the most common class label of the k nearest points in data set.

Nearest Neighbor Classification



Up next...

- k-NN is a very simple algorithm.
- But often works well in spite of its simplicity...



CSE 151A

Intro to Machine Learning

Lecture 02 – Part 04

Application: Classifying Handwritten Digits

The Problem

- **Given** an image of a handwritten digit.
- **Classify** the image as a one, two, three, etc.

The Machine Learning Approach

- Gather a **training set** of images with **labels**.
- Let the computer **learn** the underlying patterns.
- We'll use a freely available data set, **MNIST**:

MNIST

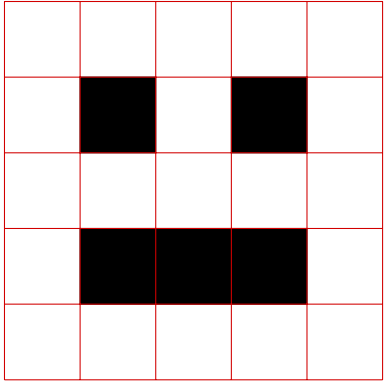
- 60,000 training images and associated labels.
- Each image is 28 × 28 pixels.
- Each pixel is 8 bit grayscale (0 - 255)

kNN on MNIST

- We'll use kNN to do **character recognition**.
  - $\mathcal{X}$ = set of 28 × 28, 8-bit grayscale images
  - $\mathcal{Y} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

- How do we treat an image as a point in space?

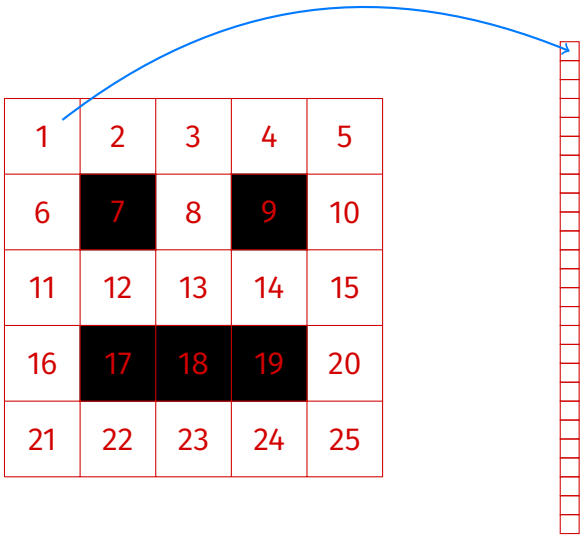
Images as Points in Space



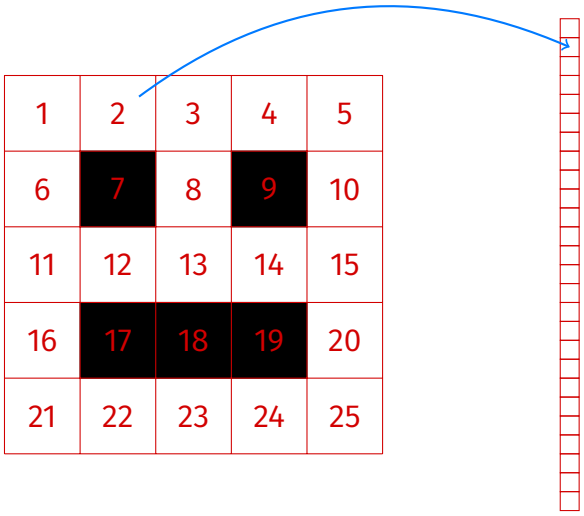
Images as Points in Space

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

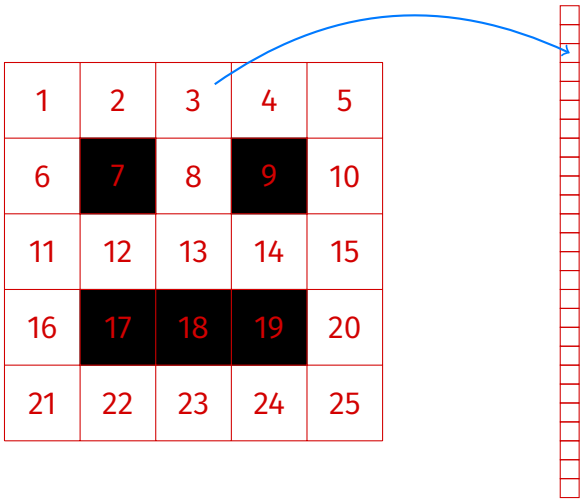
Images as Points in Space



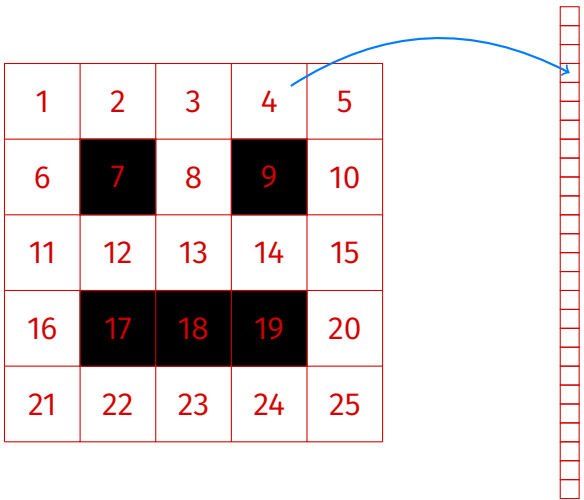
Images as Points in Space



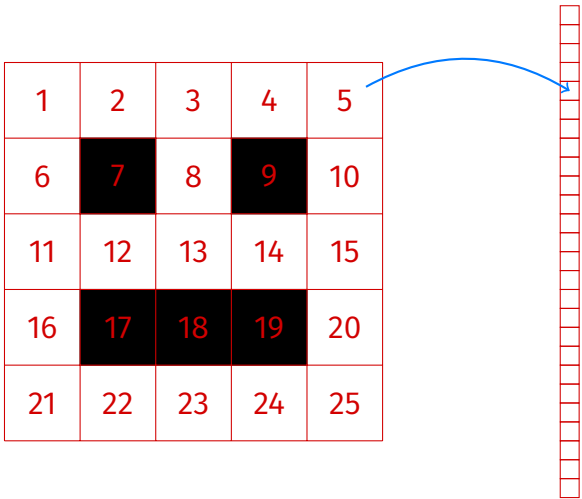
Images as Points in Space



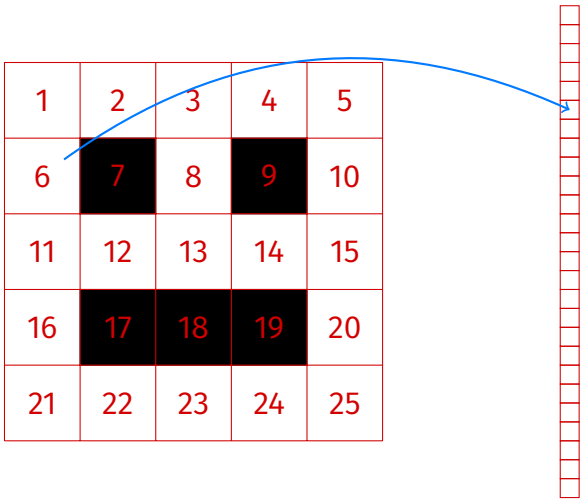
Images as Points in Space



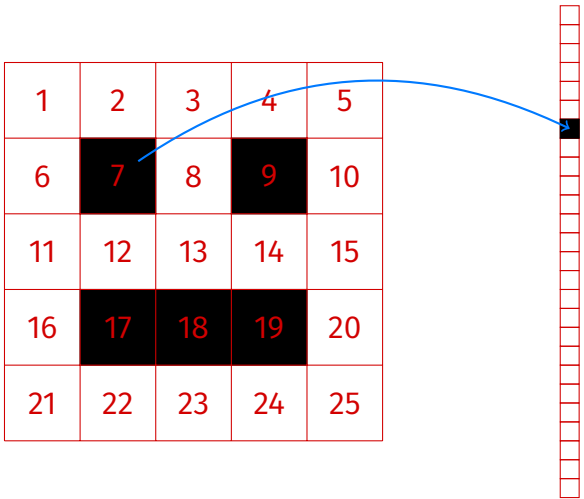
Images as Points in Space



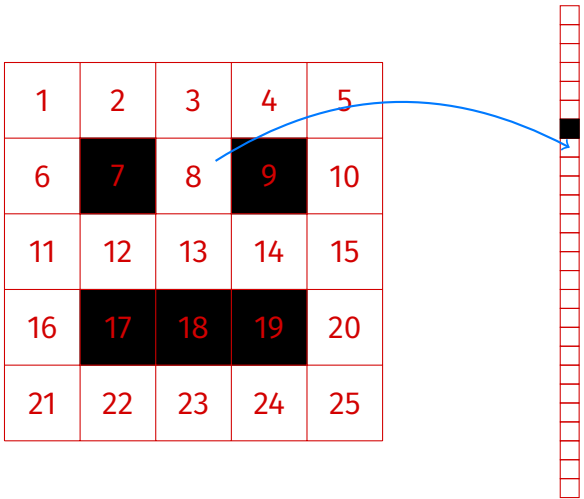
Images as Points in Space



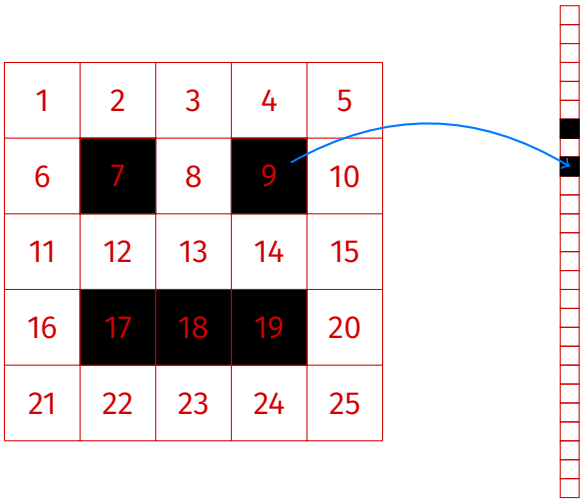
Images as Points in Space



Images as Points in Space

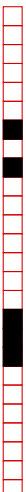


Images as Points in Space



Images as Points in Space

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25



Euclidean Distance Between Images

- We “unroll” a 28×28 image into a vector in $\mathbb{R}^{784}$.
- Euclidean distance between two images $\vec{p}^{(1)}$, $\vec{p}^{(2)}$ in $\mathbb{R}^{784}$:

$$\|\vec{p}^{(1)} - \vec{p}^{(2)}\| = \sqrt{\left(p_1^{(1)} - p_1^{(2)}\right)^2 + \left(p_2^{(1)} - p_2^{(2)}\right)^2 + \ldots + \left(p_{784}^{(1)} - p_{784}^{(2)}\right)^2}$$

$$= \sqrt{\sum_{i=1}^{784} \left(p_i^{(1)} - p_i^{(2)}\right)^2}$$

How well does it work?

- Does this make accurate predictions?
- Use 1-NN to classify each image in **training set**.
- **Question**: What will be the accuracy?

$$\text{error} = \frac{\#\ \textbf{incorrect}\ \text{predictions}}{60{,}000}$$

How well does it work?

- The error will be 0!
- Accuracy on training set is **misleading**, overly-optimistic.
- MNIST also includes a **test set** of 10,000 images and labels. Let's test on this set.

The Test Error

- Test set contains 1,000 images from each digit.
- Suppose our classifier always guesses 7.
  Question: what will be the test error?

The Test Error

- Test set contains 1,000 images from each digit.
- Suppose our classifier always guesses 7.
  Question: what will be the test error?
- **Answer**: 90%.

The Test Error

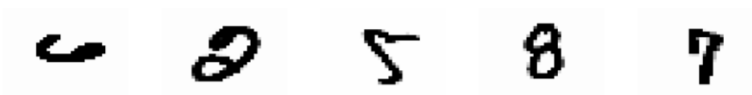
- Test set contains 1,000 images from each digit.
- Suppose our classifier guesses at random.
  Question: what is the expected test error?

The Test Error

- Test set contains 1,000 images from each digit.
- Suppose our classifier guesses at random.
  Question: what is the expected test error?
- **Answer**: 90%.

The Test Error

- The test error of nearest neighbor is only 3.09%.
- Examples of errors:

Input: 

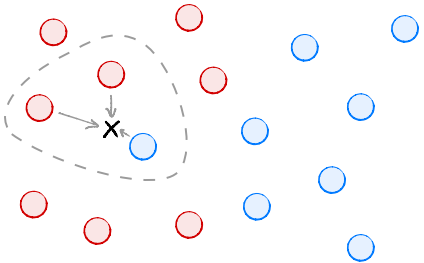
NN: 

Does k-NN do better?

k	1	3	5	7	9	11
Test Error (%):	3.09	2.94	3.13	3.10	3.43	3.34

Up next...

So what's the downside of k-NN?



CSE 151A

Intro to Machine Learning

Lecture 02 – Part 05

Practical Issues

What's right with k-NN?

- k-NN is easy to implement and understand.
- It often works quite well.
- A lot of theory.

What's wrong with k-NN?

- **Memorization**: have to store the whole data set.
- **Slow**: to classify new point, must calculate distance to every data point. Time $\Theta(dn)$.
- **Curse of Dimensionality**: distances are less meaningful in high dimensions.

Practical Improvement #1: Better Data Structures

- Can speed up the NN search by using k-d trees, ball trees, locality sensitive hashing.
- **Problem**: these offer no speed up when d is large.

Practical Improvement #2: Dimensionality Reduction

- Maybe dimensionality can be reduced without losing much info.
  - Example: “background” pixels in MNIST images.
- Use PCA, spectral embeddings, etc.

Practical Improvement #3: Approximate Nearest Neighbor

- We don’t need to find the NNs *exactly*.
- Removes need to search all points.

Next time...

Generative models.