

Table of Contents

Lecture 02 - knn

Lecture 03 - generative - models

Lecture 04 - multivariate - generative - models

Lecture 05 - naive - bayes

Lecture 06 - linear - regression - I

Lecture 07 - linear - regression - II

Lecture 08 - regularization - logistic - regression

Lecture 09 - gradient - descent

Lecture 10 - convexity

Lecture 11 - perceptron

Lecture 12 - svm

Lecture 13 - decision - trees

Lecture 14 - boosting

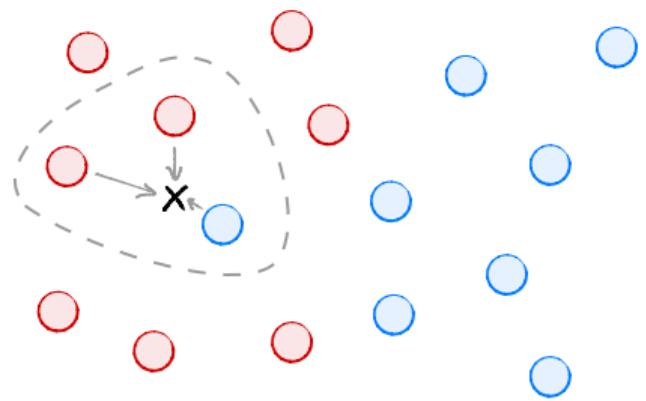
Lecture 15 - clustering

Lecture 16 - em - hierarchical

Lecture 17 - eigen

Lecture 18 - pca

Lecture 19 - bye



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.

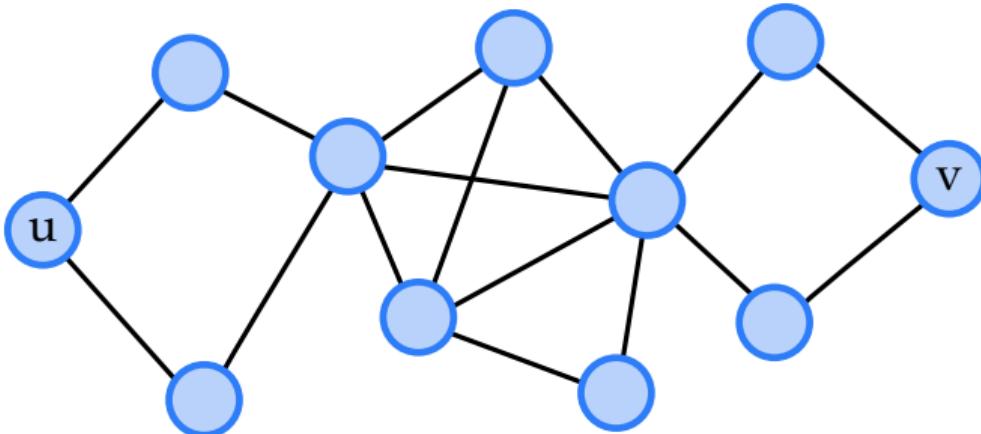
What is Machine Learning?

- ▶ 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.

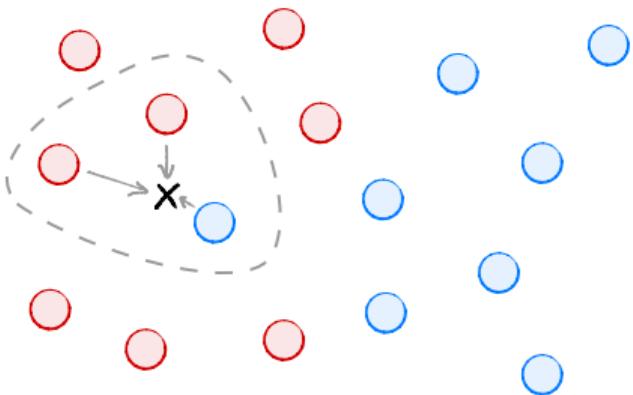
Spoiler: the algorithms are usually pretty simple. It's the **data** that does the real work.

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**, $f : \mathcal{X} \rightarrow \mathcal{Y}$ 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

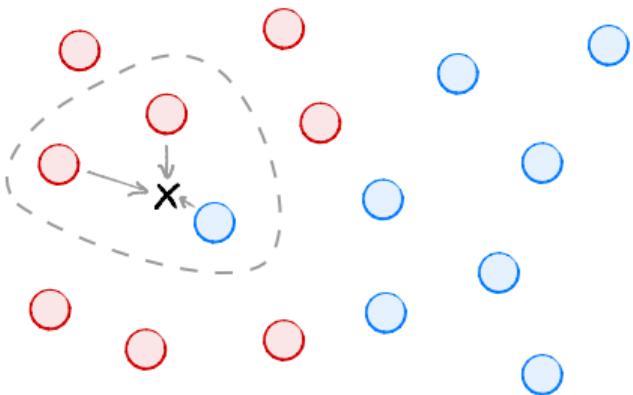
- ▶ **Classification:** when \mathcal{Y} is a discrete set.
- ▶ **Binary classification:** $|\mathcal{Y}| = 2$
 - ▶ Example: Is this picture of a hot dog?
 $\mathcal{Y} = \{\text{Yes, No}\}$.
- ▶ **Multiclass classification:** $|\mathcal{Y}| > 2$
 - ▶ Example: What food is in this picture?
 $\mathcal{Y}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

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.

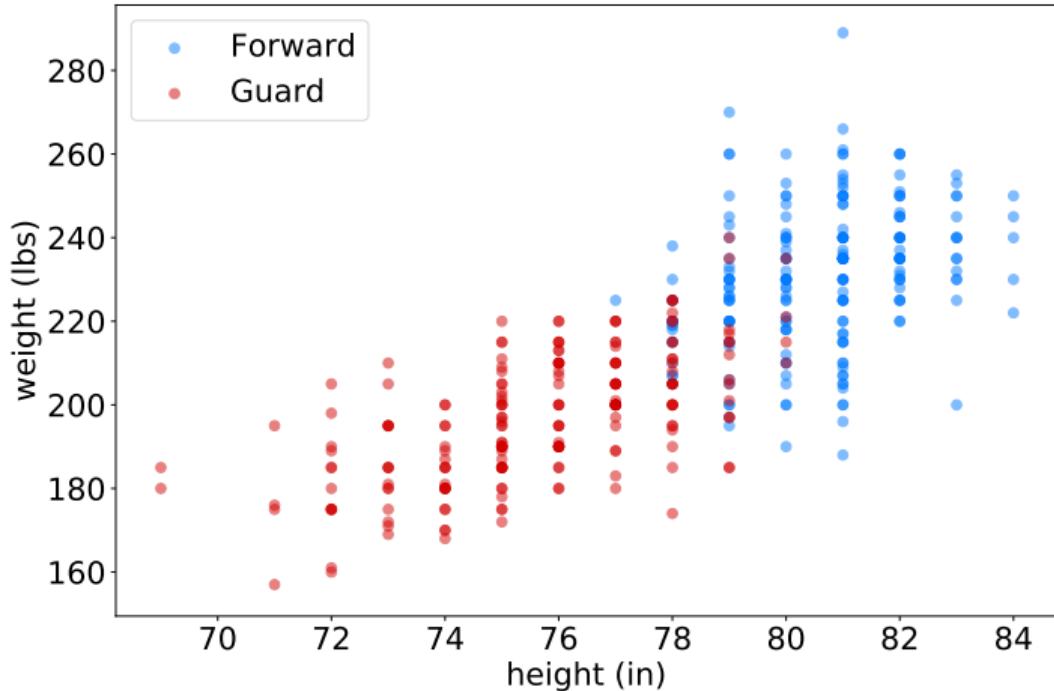


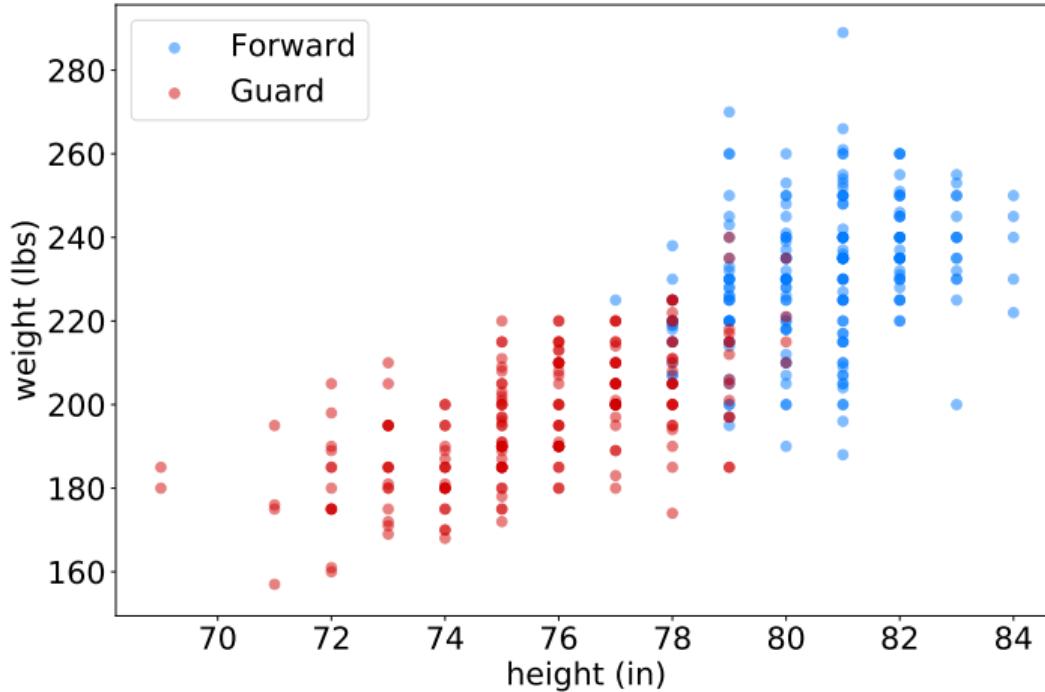


forward

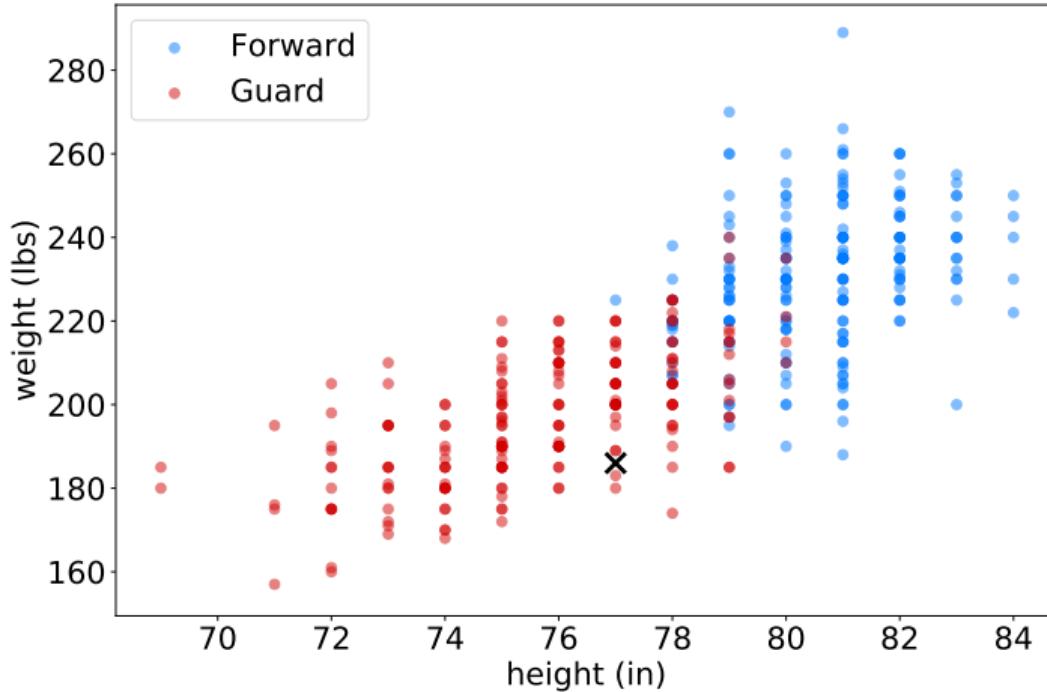


guard

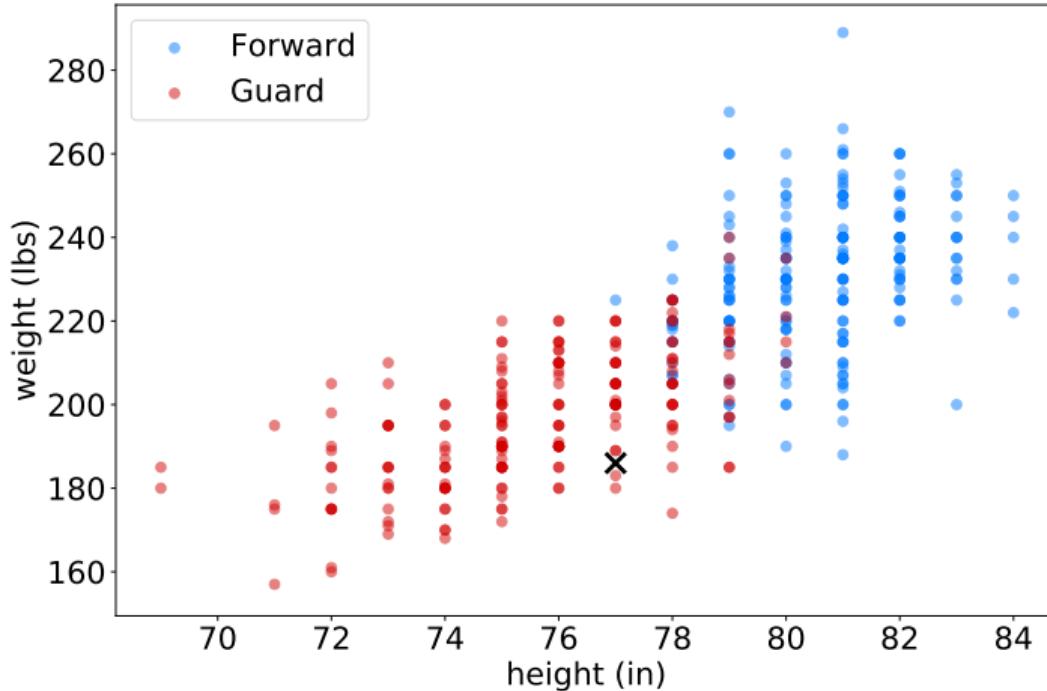




A new player is 77 inches tall and 186 pounds.
What position do they play?

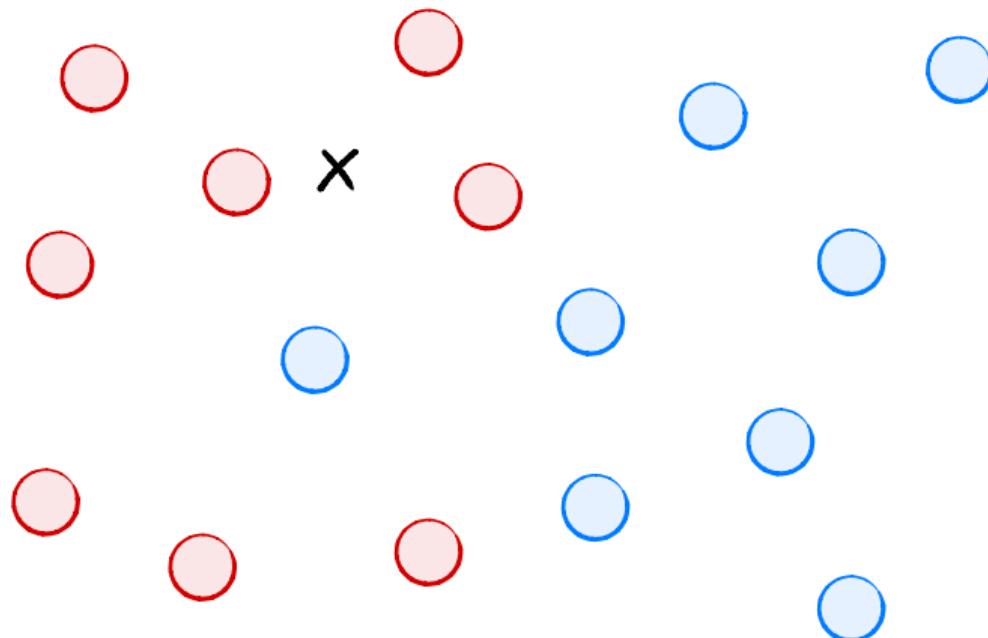


A new player is 75 inches tall and 240 pounds.
What position do they play?

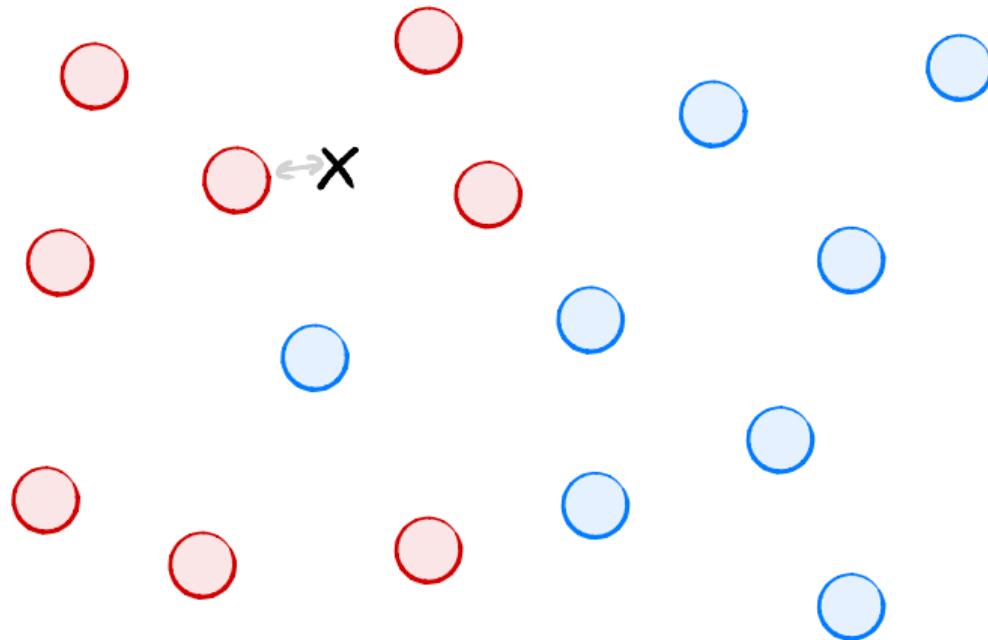


A new player is 75 inches tall and 240 pounds.
What position do they play? **Guard**.

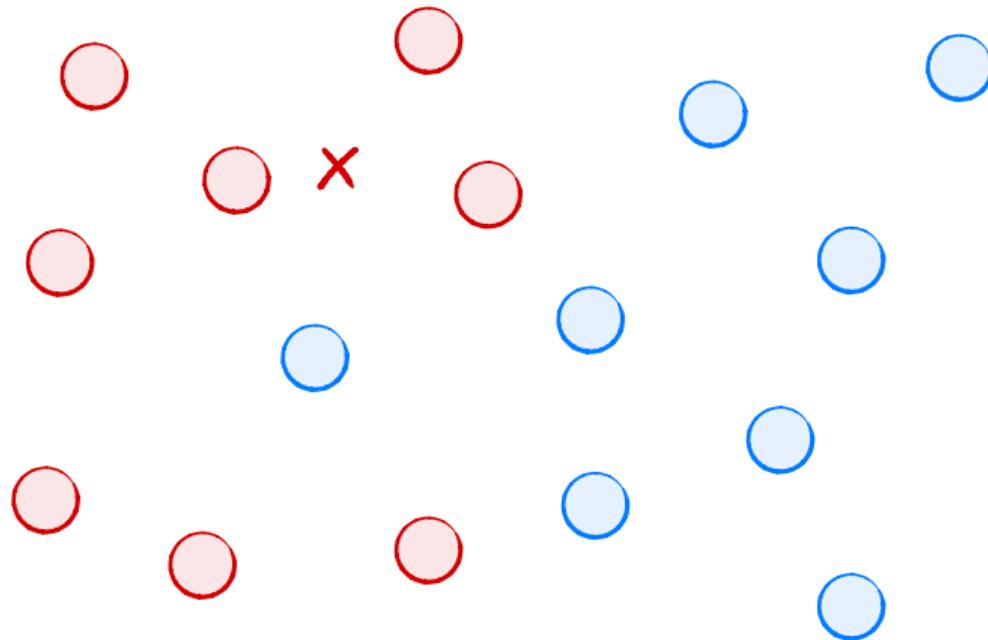
Nearest Neighbor Classification



Nearest Neighbor Classification

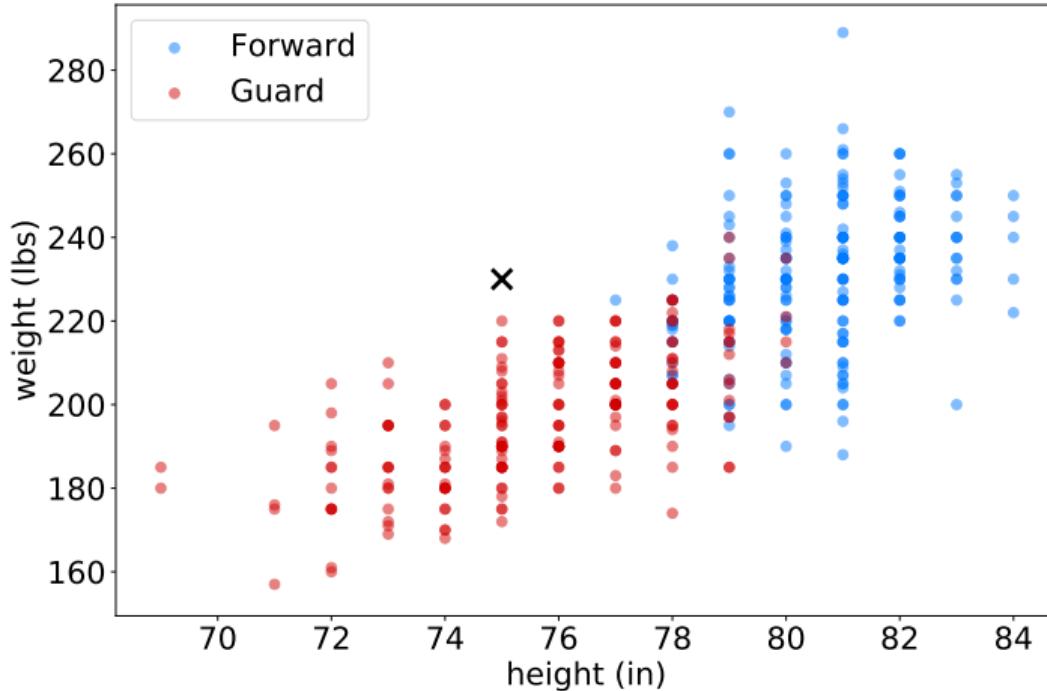


Nearest Neighbor Classification

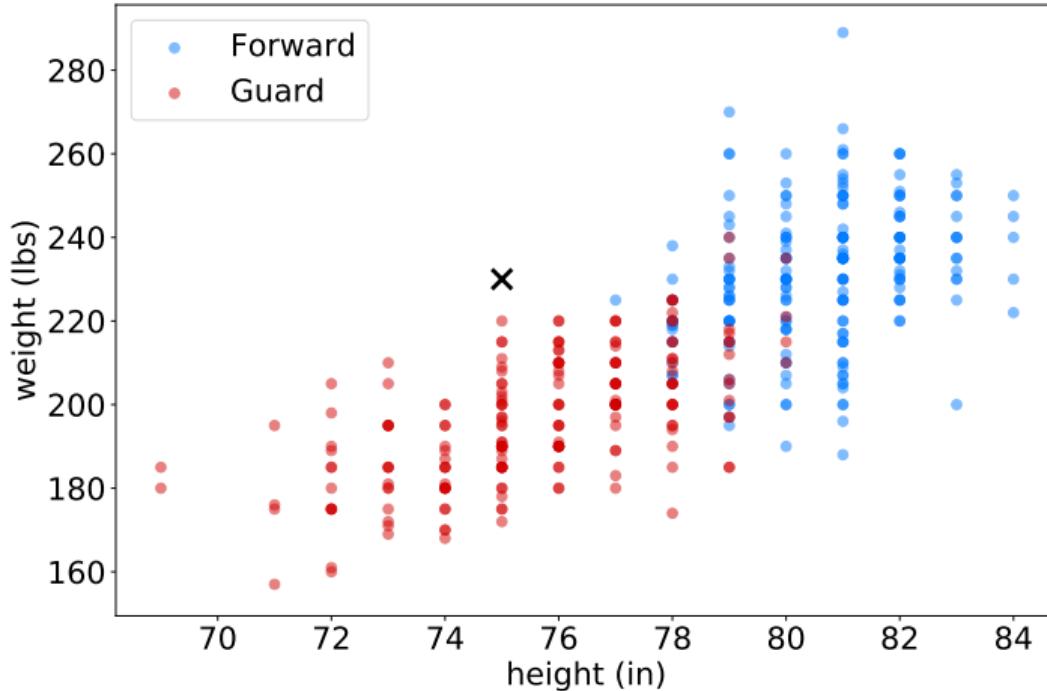


Nearest Neighbor Classification

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

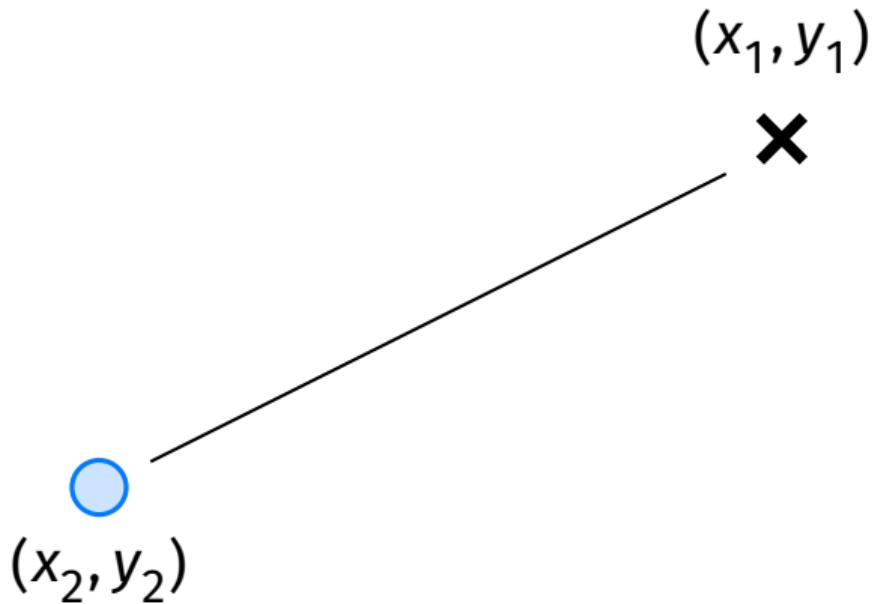


A new player is 75 inches tall and 230 pounds.
What position do they play?

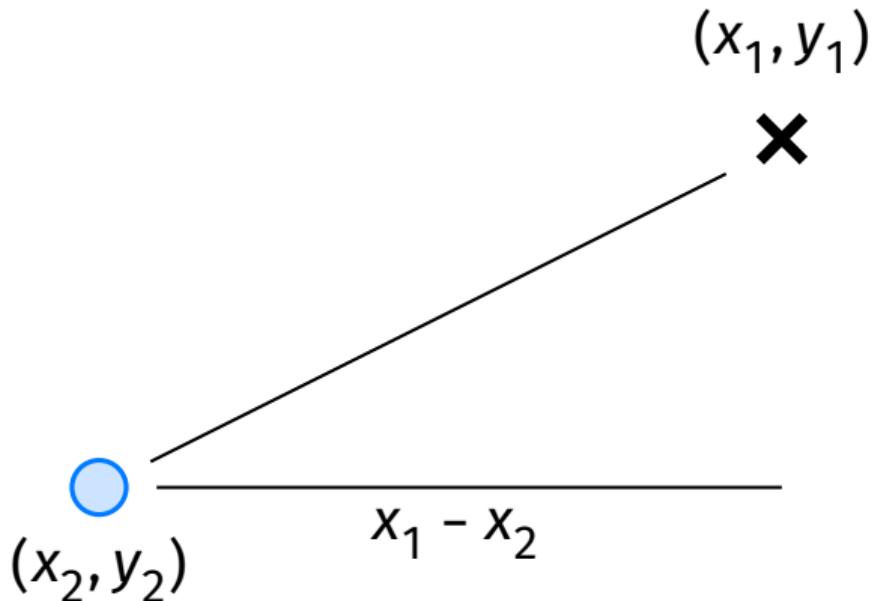


A new player is 75 inches tall and 230 pounds.
What position do they play? **Forward**.

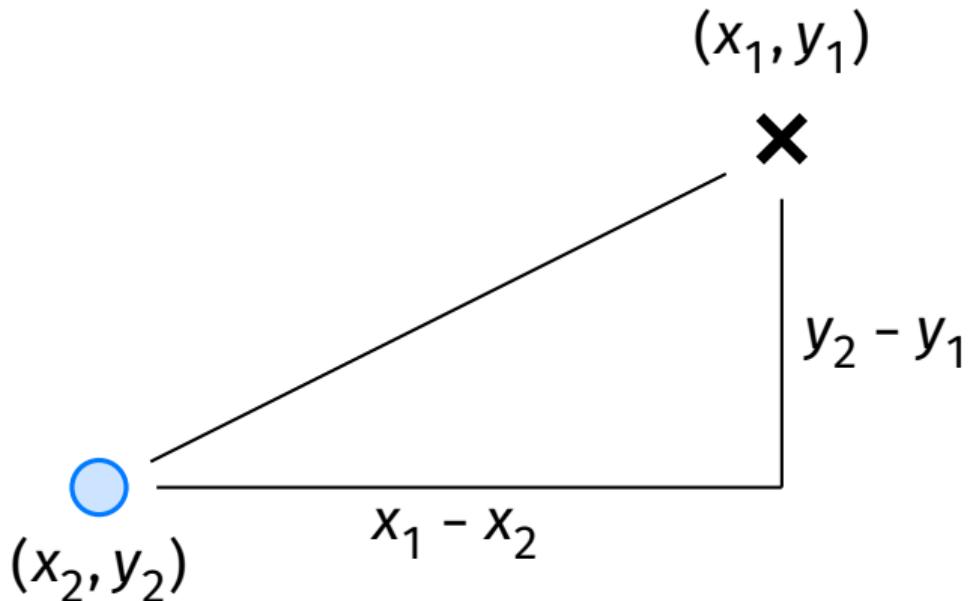
The (Euclidean) Distance



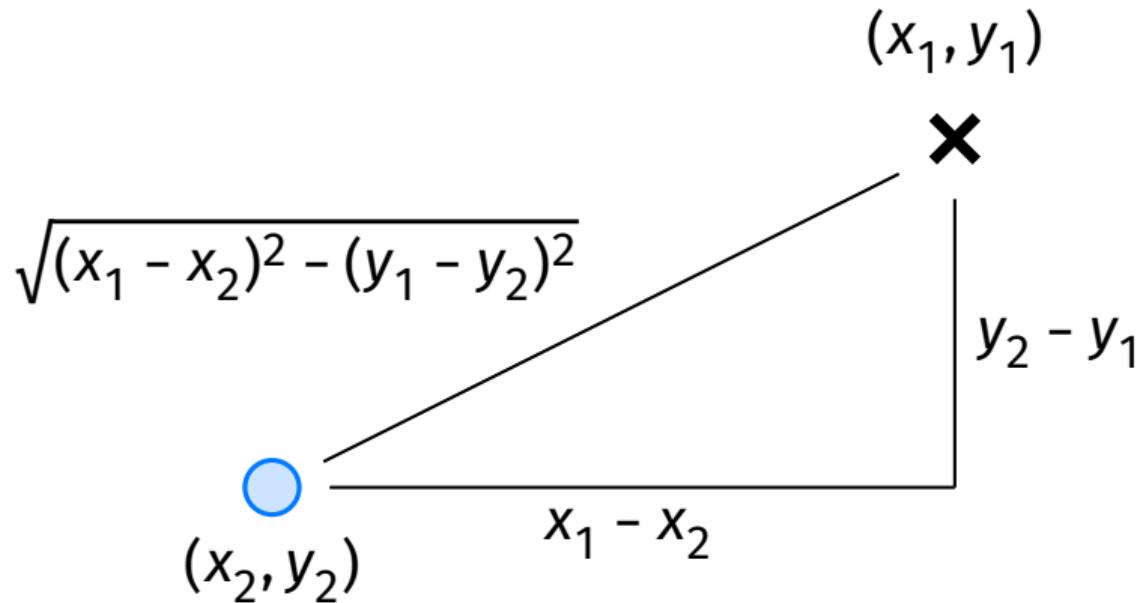
The (Euclidean) Distance

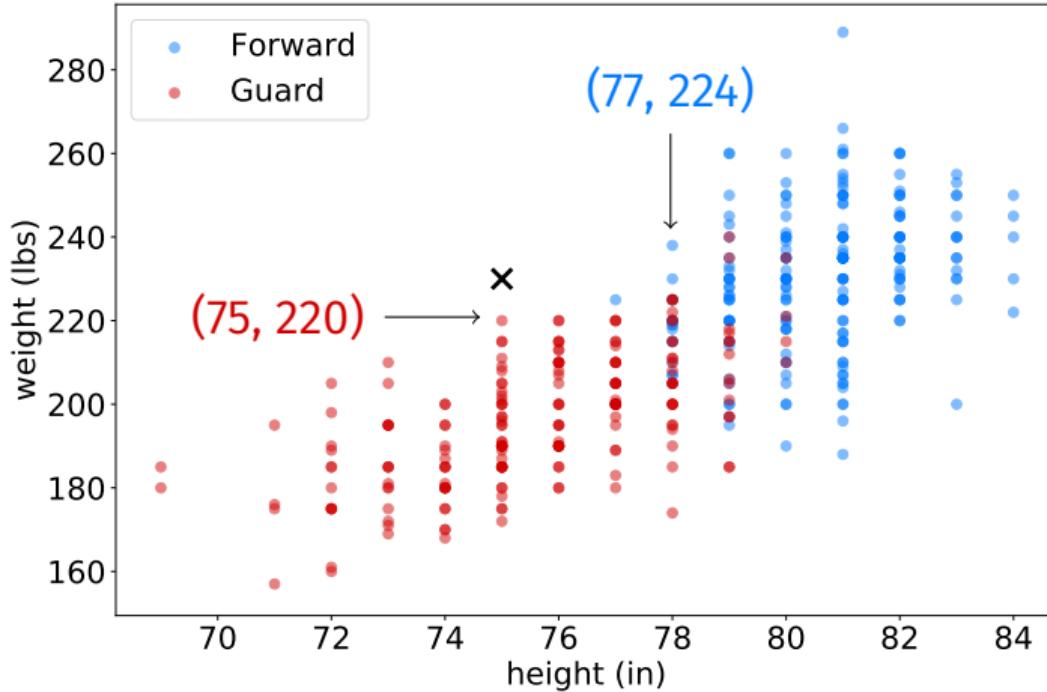


The (Euclidean) Distance

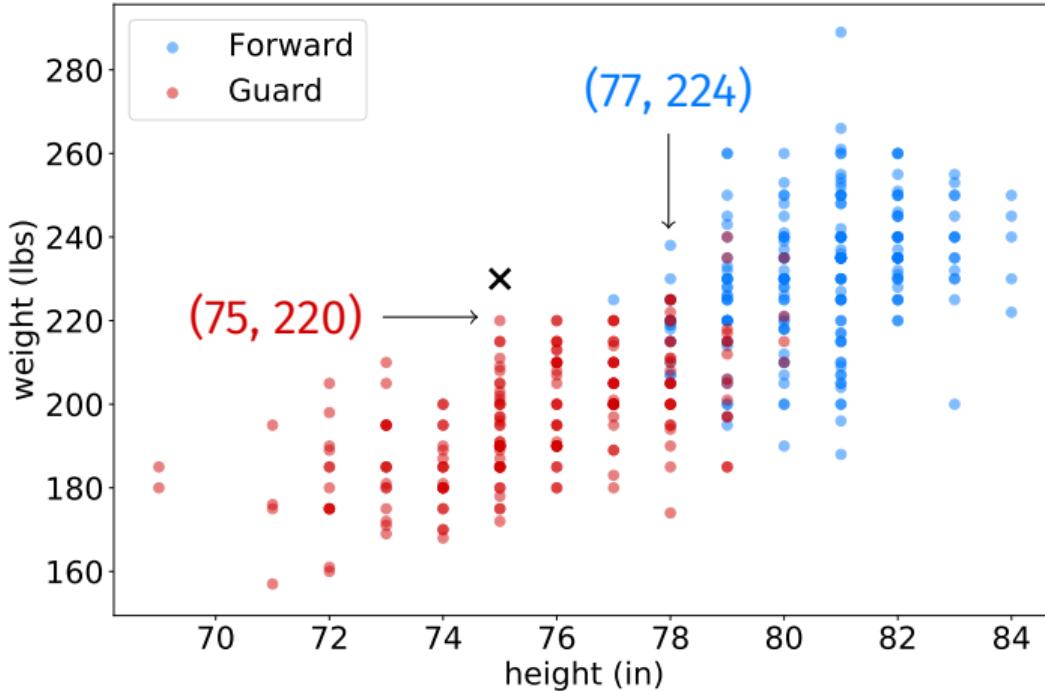


The (Euclidean) Distance





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):

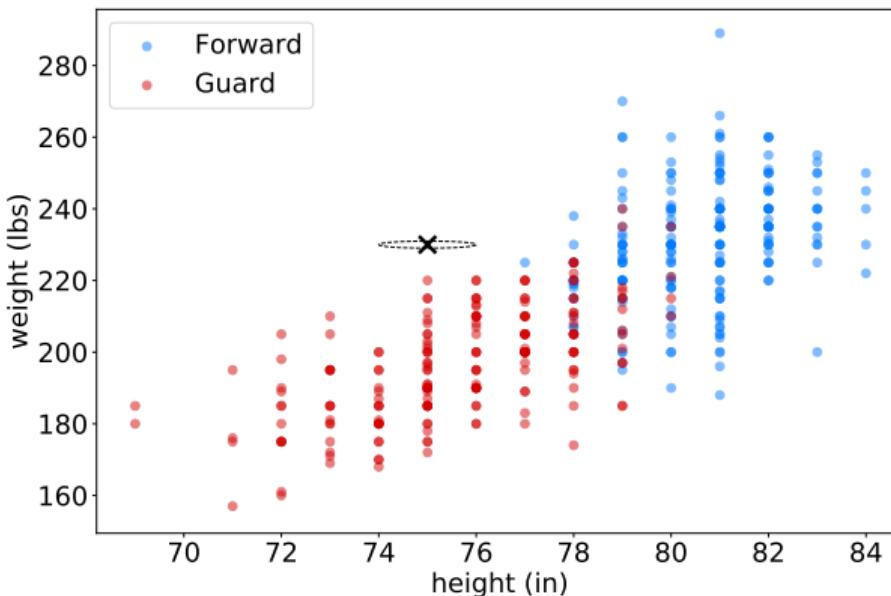
$$\sqrt{(75 - 75)^2 - (230 - 220)^2} = \sqrt{0 - 10^2} = 10$$

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

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

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

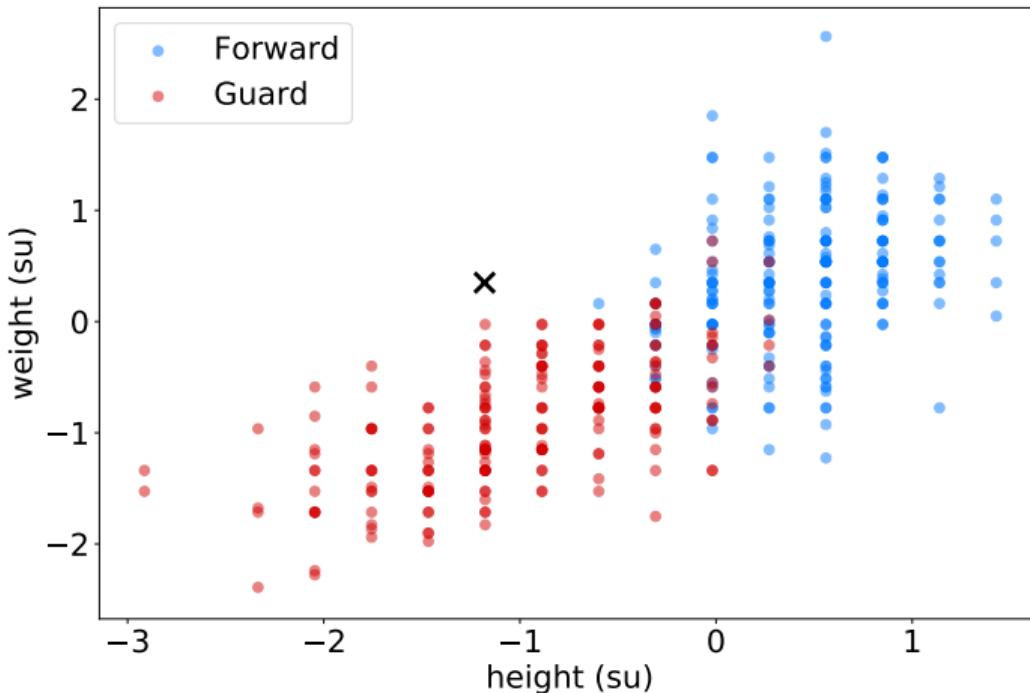
- ▶ What is 230 pounds in standard units?
- ▶ Weights: mean = 220.7, STD = 26.6

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

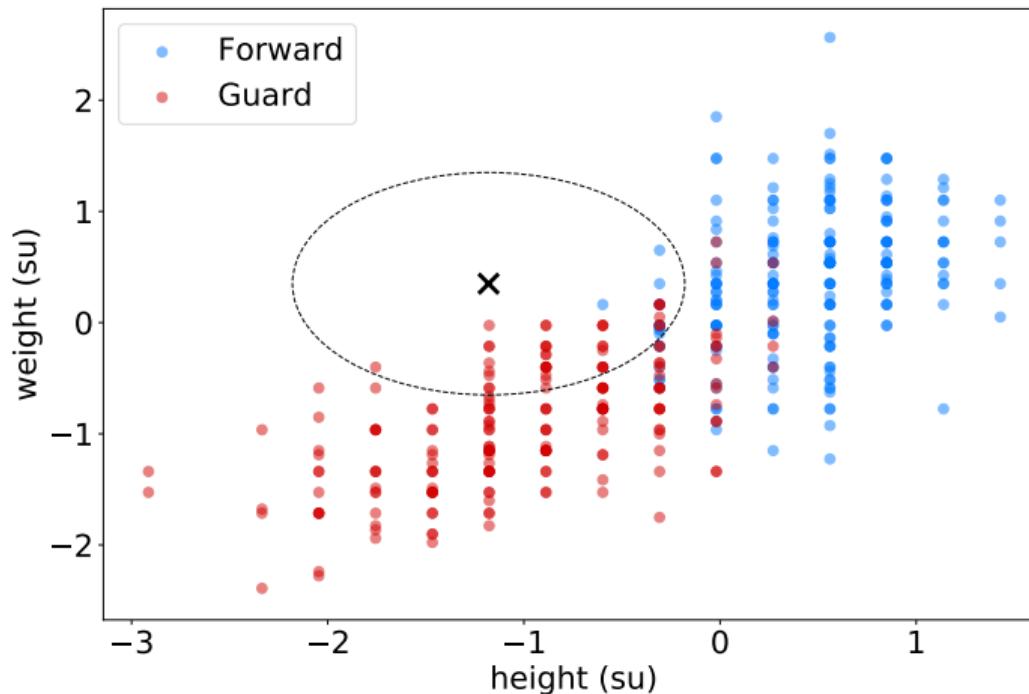
Example

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

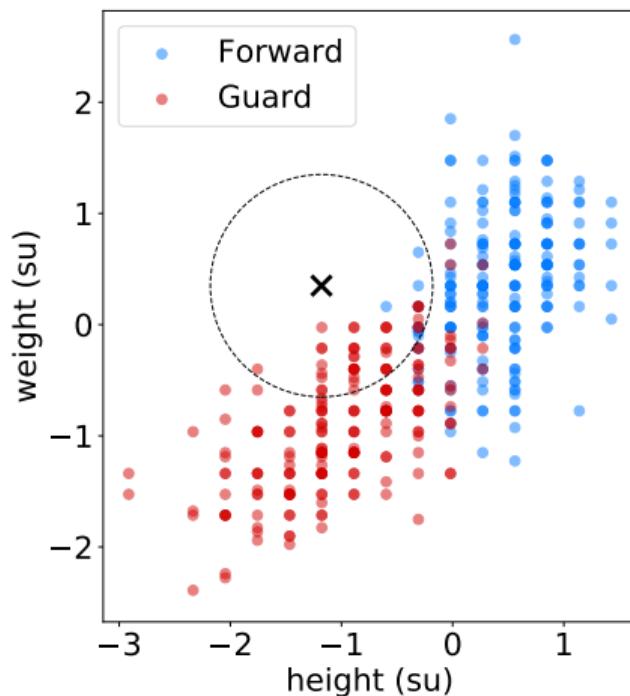
Standard Units



Standard Units



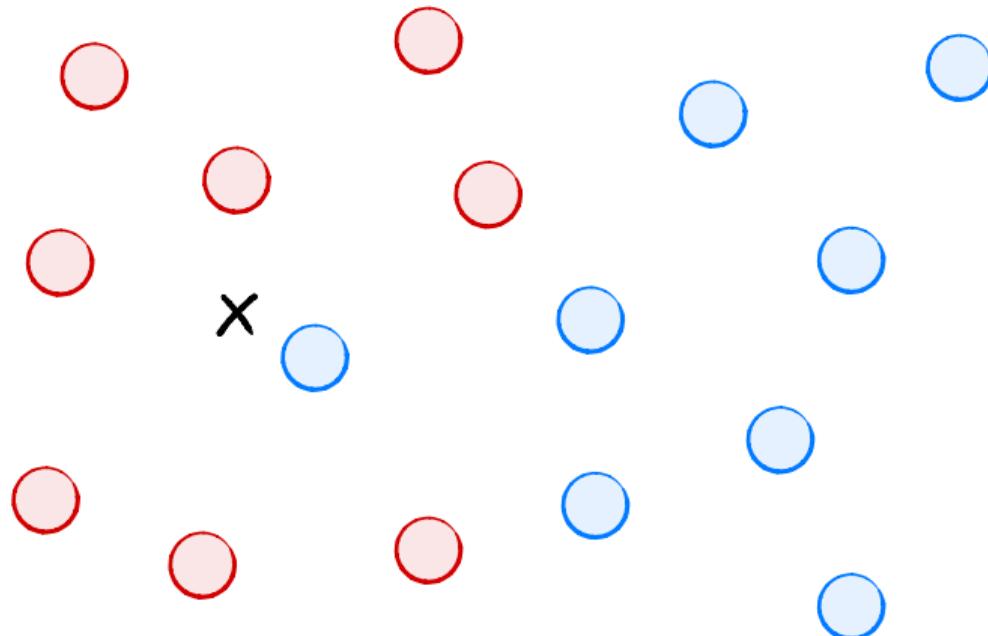
Standard Units



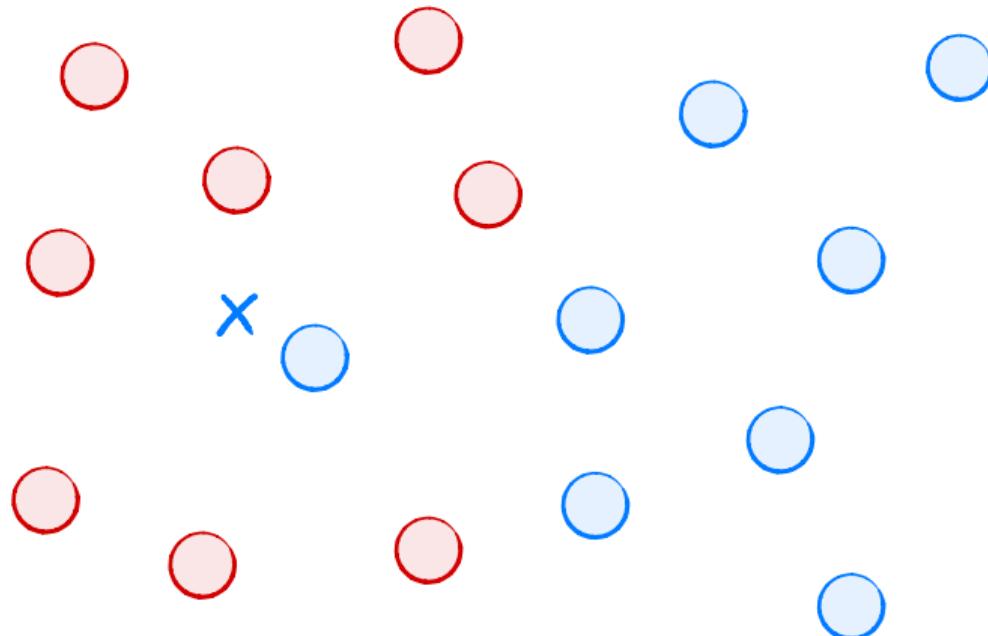
Summary: Nearest Neighbor

- ▶ 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



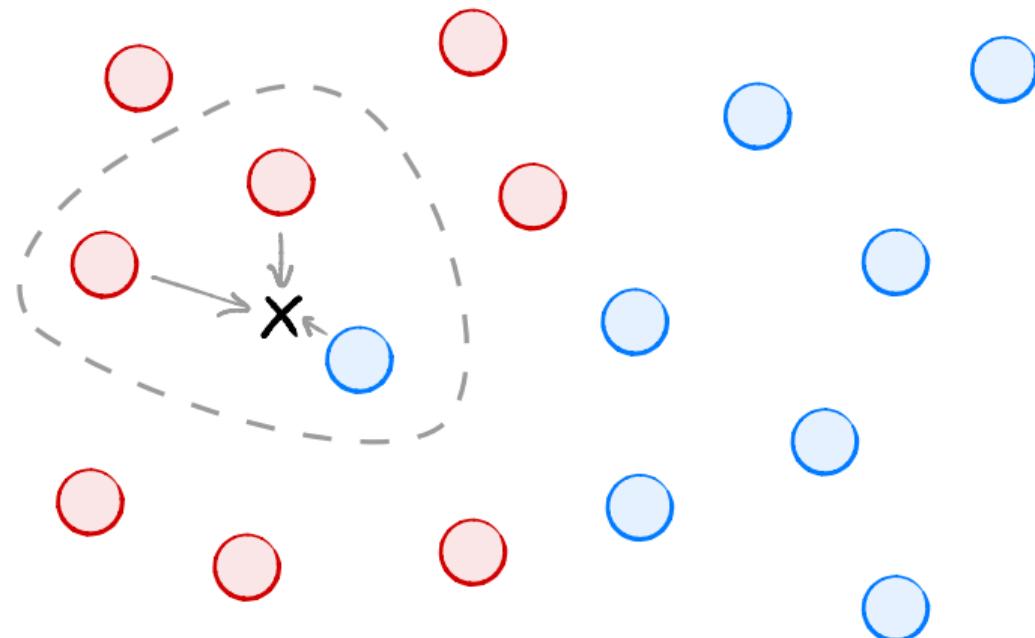
Nearest Neighbor Classification



k-Nearest Neighbors

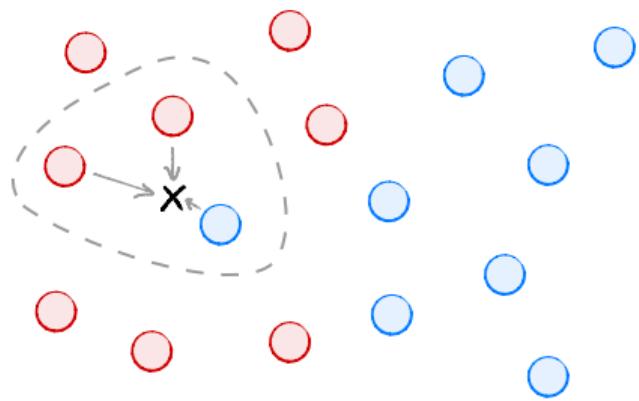
- ▶ Nearest neighbor is sensitive.
- ▶ Solution: have the nearest few neighbors “vote”.
- ▶ **Given:** an input data point, parameter k .
- ▶ **Output:** the most common class label of the k nearest points in data set.

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**:



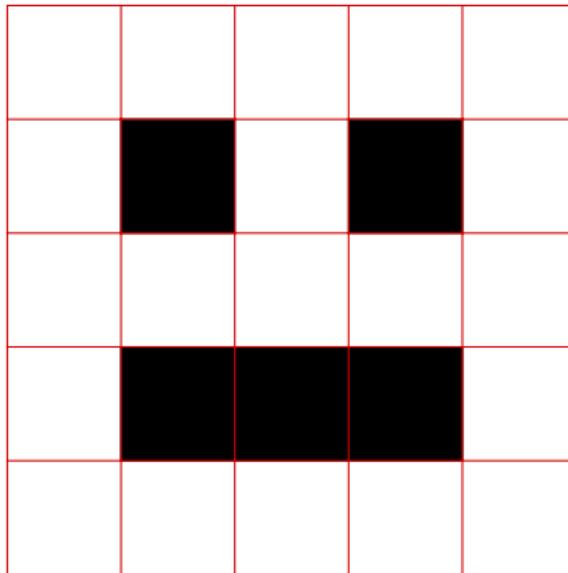
MNIST

- ▶ 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

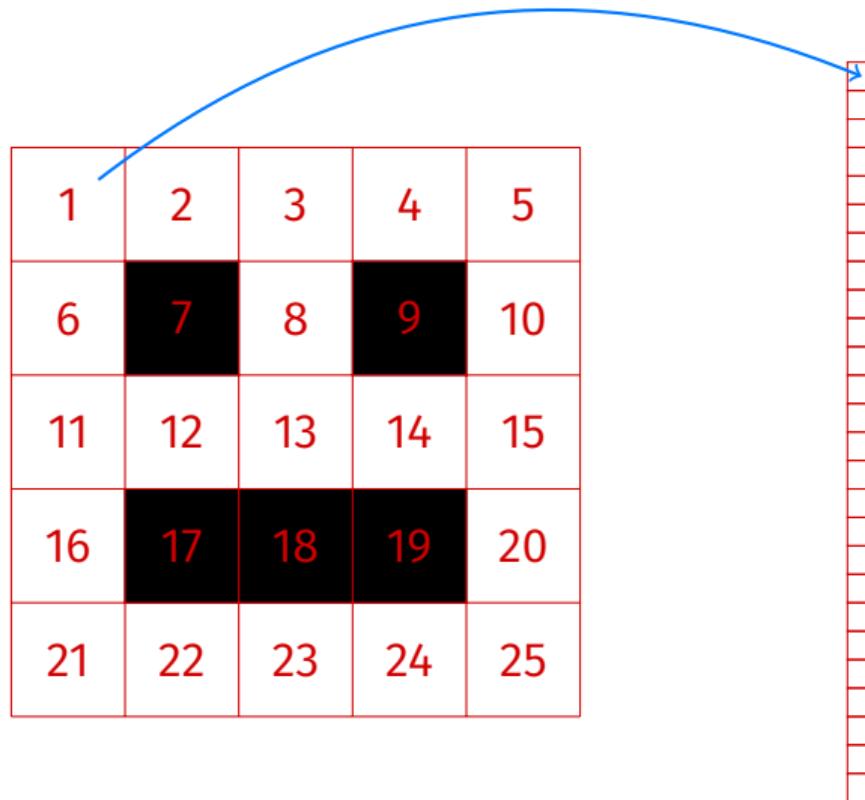


Images as Points in Space

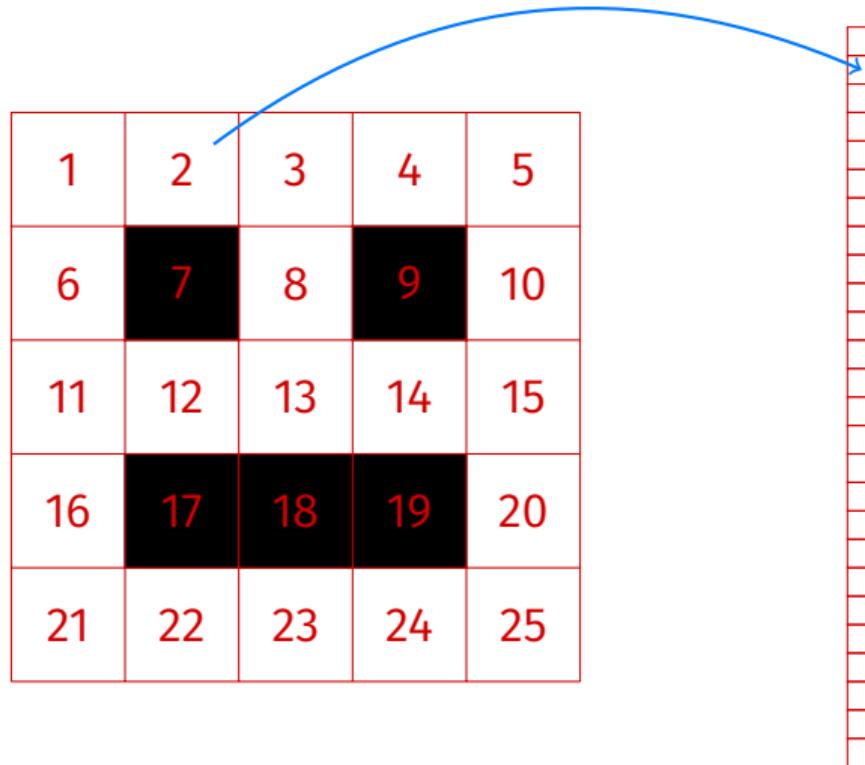
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25



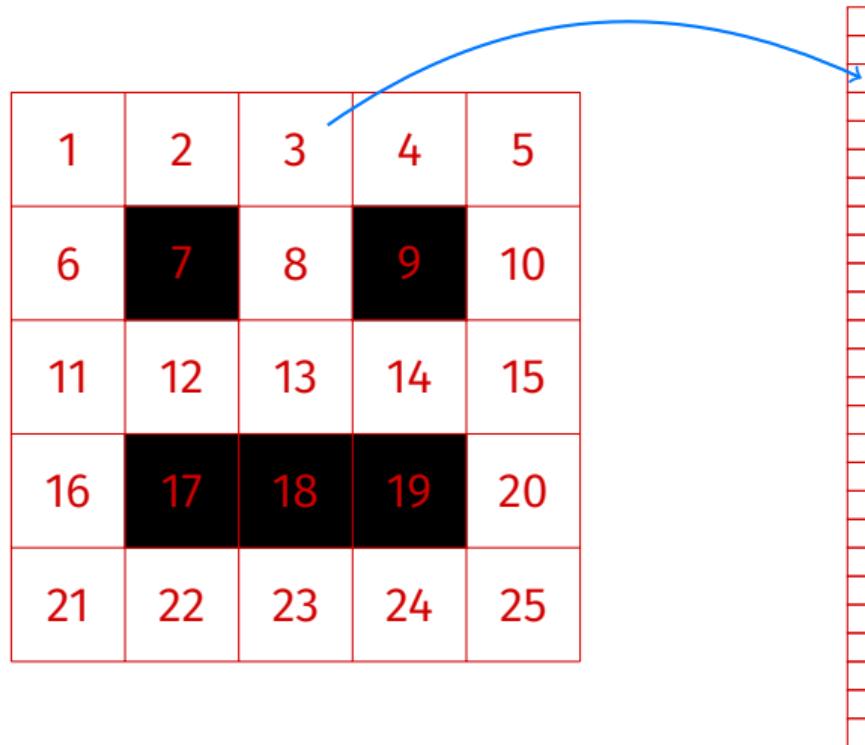
Images as Points in Space



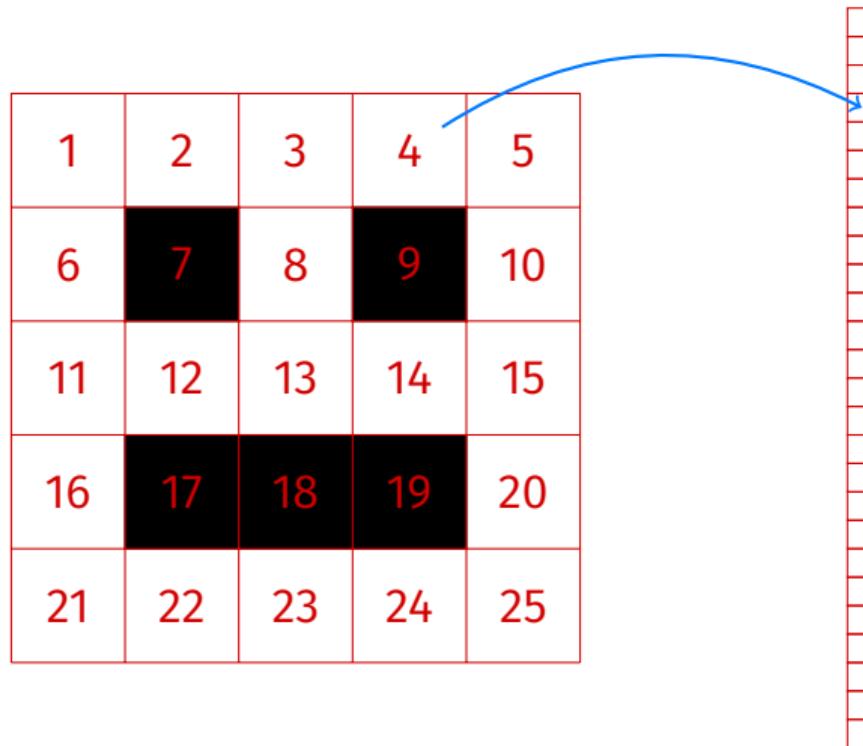
Images as Points in Space



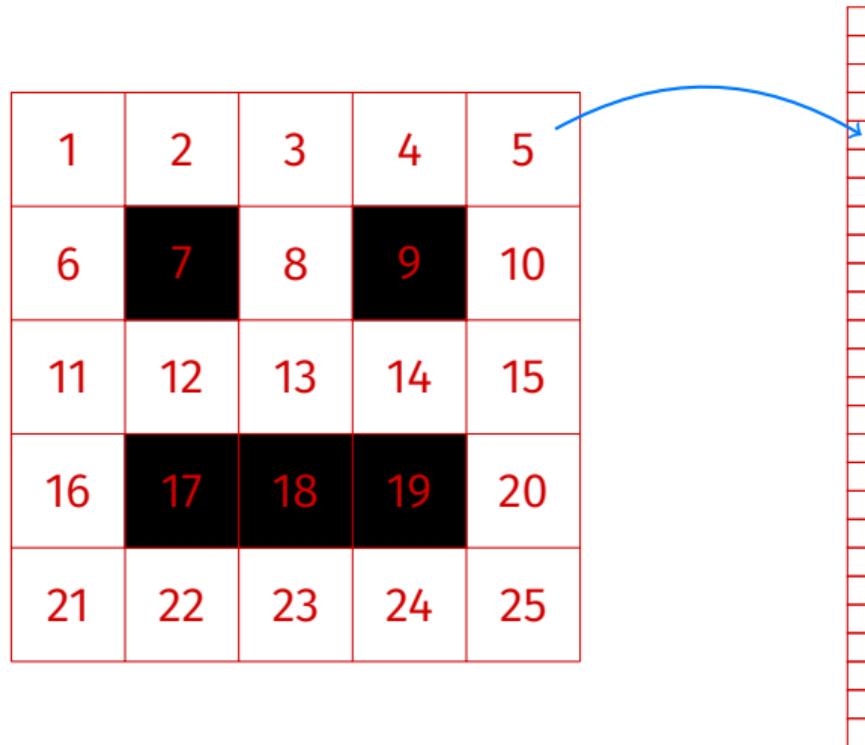
Images as Points in Space



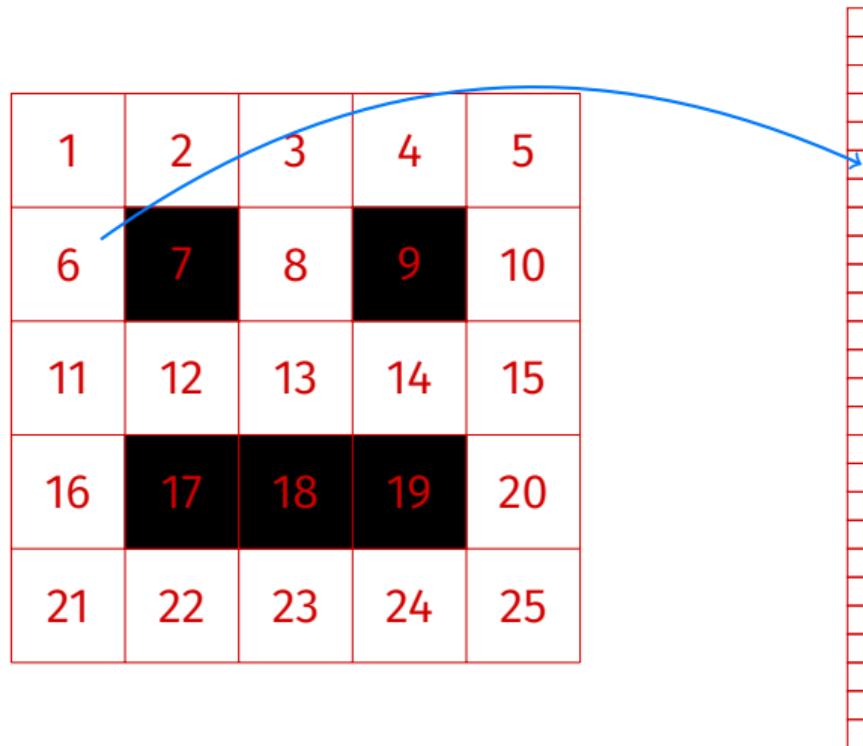
Images as Points in Space



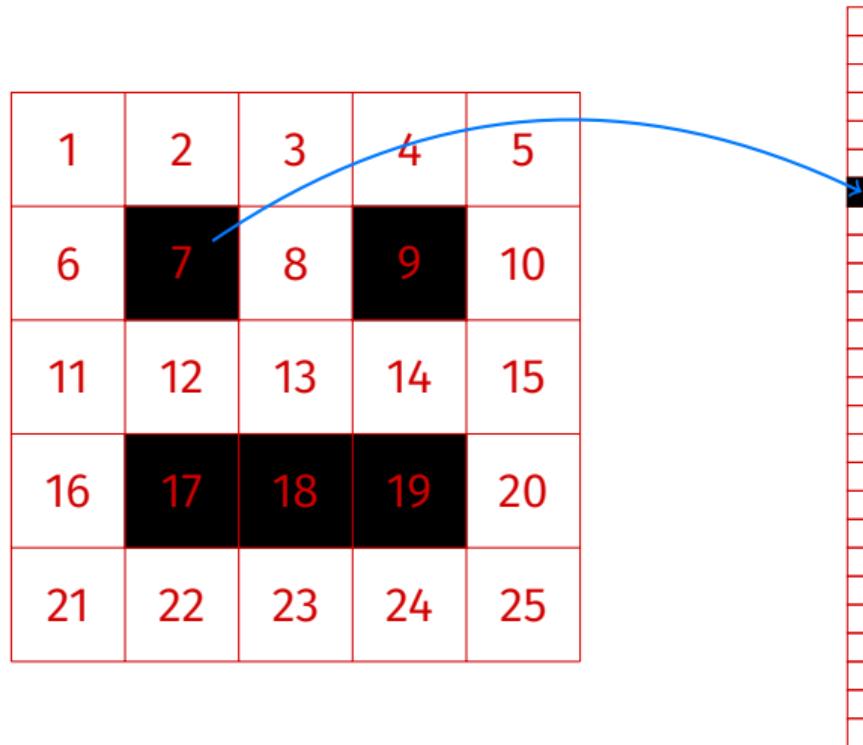
Images as Points in Space



Images as Points in Space



Images as Points in Space



Images as Points in Space

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

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

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} :

$$\begin{aligned}\|\vec{p}^{(1)} - \vec{p}^{(2)}\| &= \sqrt{\left(p_1^{(1)} - p_1^{(2)}\right)^2 + \left(p_2^{(1)} - p_2^{(2)}\right)^2 + \dots + \left(p_{784}^{(1)} - p_{784}^{(2)}\right)^2} \\ &= \sqrt{\sum_{i=1}^{784} \left(p_i^{(1)} - p_i^{(2)}\right)^2}\end{aligned}$$

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?

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

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:

4 0 5 8 7

NN:

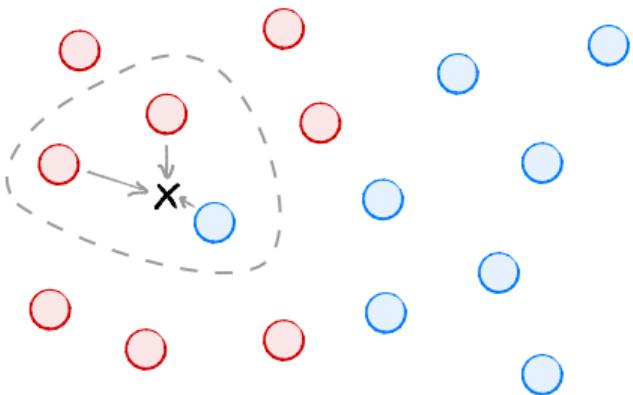
4 0 5 9 9

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

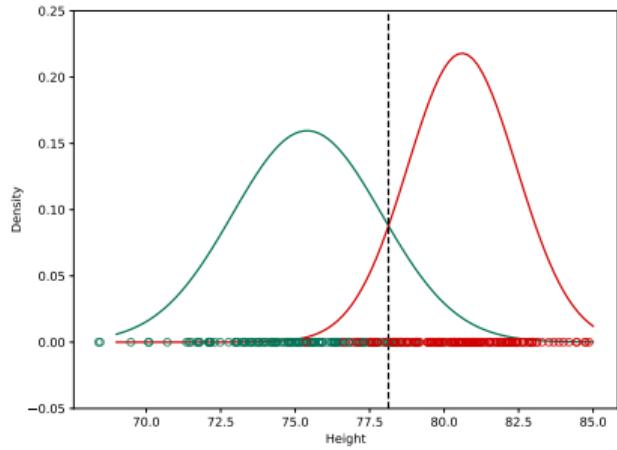
- ▶ Maybe dimensionality can be reduced without losing much info.
 - ▶ Example: “background” pixels in MNIST images.
- ▶ Use PCA, spectral embeddings, etc.

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.



CSE 151A
Intro to Machine Learning

Lecture 03 – Part 01
The Probabilistic View

Recap

- ▶ Some tasks are not easily dictated to computers.
- ▶ Instead, we give it data and let it learn.
- ▶ But we still have to tell it how to learn.
- ▶ The magic is in the **data**.

Recap: Classification

- ▶ Predict which class the input belongs to.
- ▶ We encode instance as a **feature vector**.
- ▶ We have a **training set** of feature vectors and associated **class labels**.

Recap: Classification

- ▶ Train classifier to do well on the training data.
- ▶ Really want it to do well on data **we haven't seen**.
- ▶ That is, we want the classifier to **generalize**.

Recap: Classification

- ▶ We estimate ability to generalize with a **test set**.
- ▶ Test set contains the “right answers”, too.
- ▶ Training error = error on training set.
- ▶ Test error = error on test set.

Recap: k-Nearest Neighbors

- ▶ kNN: “memorize” the training set.
- ▶ Predict the majority class of the k nearest neighbors in the training set.
- ▶ **Works well, simple, slow, memory-intensive, struggles in high dimensions.**

A New View via Probability



Example

- ▶ The average height of a forward is 80.5 inches.
- ▶ The average height of a guard is 75.4 inches.
- ▶ A new player is 73 inches tall. They're probably a guard / forward.

Example

- ▶ The average height of a forward is 80.5 inches.
- ▶ The average height of a guard is 75.4 inches.
- ▶ A new player is 73 inches tall. They're probably a **guard** / forward.

Example

- ▶ The average height of a forward is 80.5 inches.
- ▶ The average height of a guard is 75.4 inches.
- ▶ Another new player is 81 inches tall. They're probably a guard / forward.

Example

- ▶ The average height of a forward is 80.5 inches.
- ▶ The average height of a guard is 75.4 inches.
- ▶ Another new player is 81 inches tall. They're probably a guard / **forward**.

Example

- ▶ The average height of a forward is 80.5 inches.
- ▶ The average height of a guard is 75.4 inches.
- ▶ Another new player is 77.5 inches tall. They're probably a guard / forward.

Example

- ▶ The average height of a forward is 80.5 inches.
- ▶ The average height of a guard is 75.4 inches.
- ▶ Another new player is 77.5 inches tall. They're probably a ???

Example

- ▶ Each new player has some **probability** of being a **guard** and some **probability** of being a **forward**.
- ▶ These probabilities are influenced by the player's **height**.

Conditional Probabilities

- ▶ Let X be player's height. Let Y be position.
- ▶ The probability that a player is a forward **given** they are 77.5 inches tall is written:

$$P(Y = \text{forward} | X = 77.5)$$

- ▶ This is the **conditional probability** of position given height.

Example

- ▶ $P(Y = \text{forward}|X = 81) \approx 1$ ▶ $P(Y = \text{guard}|X = 81) \approx 0$
- ▶ $P(Y = \text{forward}|X = 70) \approx 0$ ▶ $P(Y = \text{guard}|X = 70) \approx 1$
- ▶ $P(Y = \text{forward}|X = 77.5) \approx \frac{1}{2}?$ ▶ $P(Y = \text{guard}|X = 77.5) \approx \frac{1}{2}?$

Classification

- ▶ A new player is x inches tall. What is their position?
- ▶ If $P(Y = \text{forward}|X = x) > P(Y = \text{guard}|X = x)$, predict **forward**.
- ▶ If $P(Y = \text{guard}|X = x) > P(Y = \text{forward}|X = x)$, predict **guard**.

The Bayes Classifier

- ▶ Given $X = x$, and possible classes $y_1, \dots, y_k \dots$
- ▶ ...predict the class y_i that makes $P(Y = y_i | X = x)$ the largest.

Bayes Error

- ▶ Assume new player with height x is either a guard or a forward (binary classification).
- ▶ Suppose $P(\text{guard}|X = x) = 0.6$.
- ▶ Then $P(\text{forward}|X = x) = 0.4$.
- ▶ We predict **guard**, but 40% chance we're wrong.

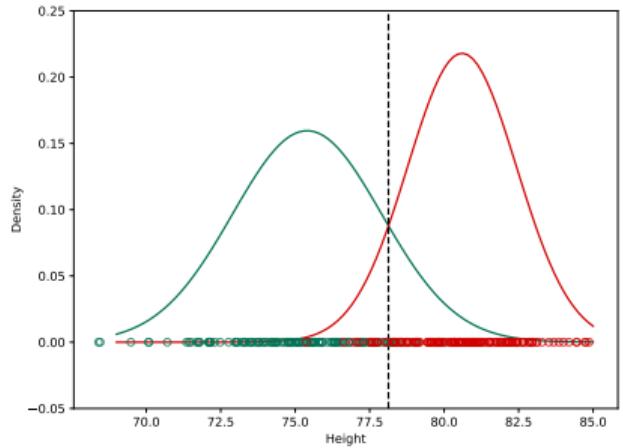
Bayes Error

- ▶ Usually, some error is unavoidable.
- ▶ Out of all possible classifiers, the Bayes Classifier has the smallest expected error.
- ▶ This error is called the **Bayes Error**.

Great! So ML is solved...

Problem: we don't **know** these probabilities.

Solution: gather **data** and **estimate** them.



CSE 151A
Intro to Machine Learning

Lecture 03 – Part 02
**Estimating
Probabilities**

Estimating Probabilities

- ▶ What is the probability that a tablet is defective?
- ▶ There is a “true” probability p ; $P(\text{defective}) = p$.
- ▶ Estimate: sample n tablets, count # defective.

$$P(\text{defective}) \approx \frac{\#\text{defective}}{n}$$

Estimating Probabilities

- ▶ What is the probability that a tablet is defective?
- ▶ There is a “true” probability p ; $P(\text{defective}) = p$.
- ▶ Estimate: sample n tablets, count # defective.

$$P(\text{defective}) \approx \frac{\#\text{defective}}{n}$$

- ▶ Law of large numbers: estimate $\rightarrow p$ as $n \rightarrow \infty$.

Estimating Conditional Probabilities

- ▶ What is the probability that a tablet is defective **given** that it is made by **Apple**?
- ▶ Use above data but discard non-Apple tablets:

$$P(\text{defective} \mid \text{Apple}) \approx \frac{\# \text{ defective Apple}}{\# \text{ Apple}}$$

Estimating Conditional Probabilities

- ▶ We estimate $P(A = a | B = b)$ by gathering data and counting:

$$\frac{\text{\# for which } A = a \text{ and } B = b}{\text{\# for which } B = b}$$

- ▶ **Problem:** what if A or B are **continuous**?

Example: Discrete A, Continuous B

- ▶ Estimate probability that player is a forward, given their height is 77.15 inches.

$$\frac{\text{# forwards with height} = 77.15}{\text{# height} = 77.15}$$

- ▶ **Problem:** no one in data w/ height *exactly* 77.15 in
- ▶ Divide by zero. **Undefined.**

Example: Continuous A, Discrete B

- ▶ Estimate probability that height = 77.15 in, given that player is a forward.

$$\frac{\text{\# forwards with height 77.15 in}}{\text{\# forwards}}$$

- ▶ **Problem:** no one in data w/ height *exactly* 77.15 in
- ▶ **Zero** unless 77.15 in. forward is in data set.

Solution: Smoothing

- ▶ If A is continuous, estimate

$$P(A = \text{close to } a | B = b)$$

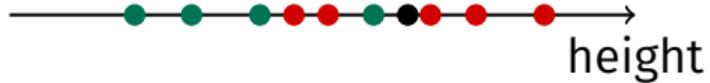
- ▶ If B is continuous, estimate

$$P(A = a | B = \text{close to } b)$$

- ▶ We'll see approaches for each case.

Discrete A, Continuous B

- ▶ Estimate $P(Y = \text{forward} | X = 77.15)$



Discrete A, Continuous B

- ▶ Estimate $P(Y = \text{forward} | X = 77.15)$



- ▶ Use fraction of $k = 3$ nearest neighbors:

$$P(Y = \text{forward} | X = 77.15) \approx \frac{\text{2 red}}{3} = \frac{2}{3}$$

Discrete A, Continuous B

- ▶ Estimate $P(Y = \text{forward} | X = 77.15)$

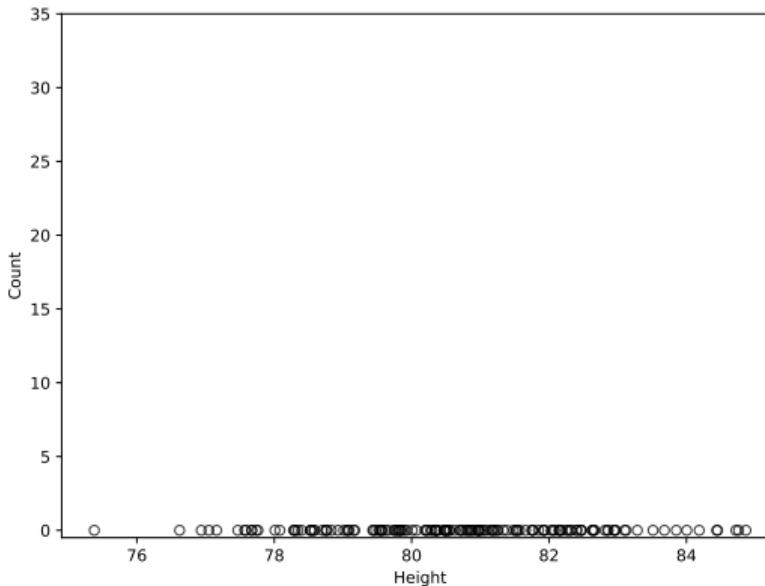


- ▶ Use fraction of $k = 3$ nearest neighbors:

$$P(Y = \text{forward} | X = 77.15) \approx \frac{\text{2 red}}{3} = \frac{2}{3}$$

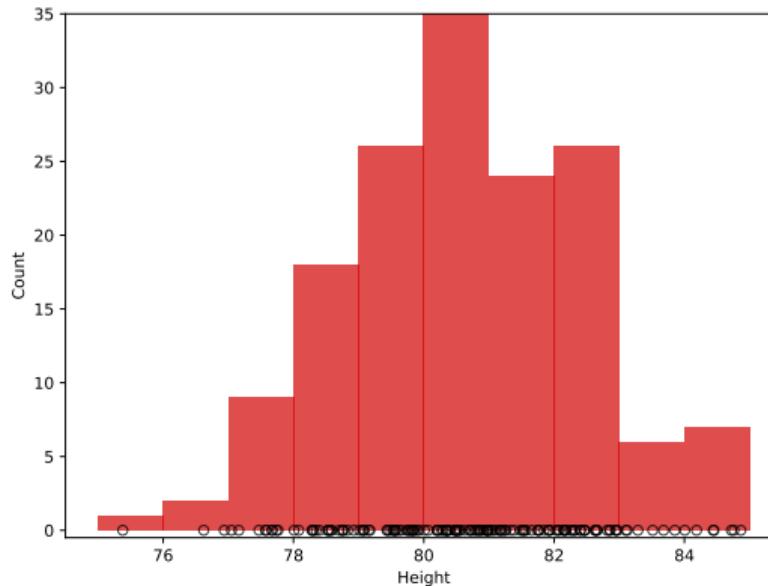
Continuous A, Discrete B

- ▶ Estimate $P(X = 77.15 | Y = \text{forward})$



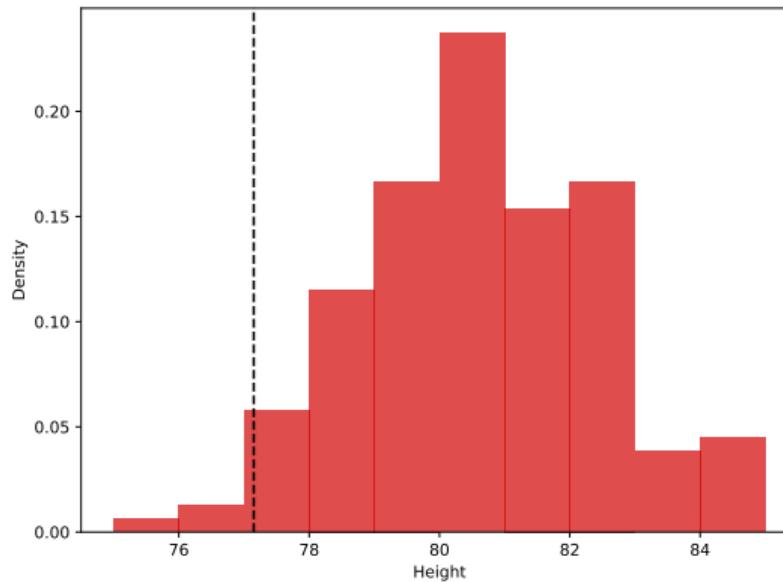
Continuous A, Discrete B

- ▶ Estimate $P(X = 77.15 | Y = \text{forward})$



Continuous A, Discrete B

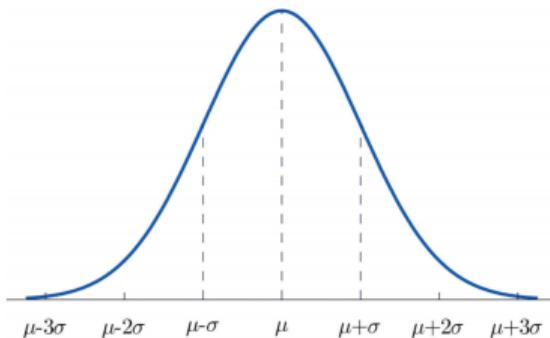
- ▶ Estimate $P(X = 77.15 | Y = \text{forward})$



Histograms

- ▶ We can estimate these probabilities with a **histogram**.
- ▶ Observe: the histogram is bell shaped.
- ▶ Let's try fitting a Normal curve.

The Normal Curve

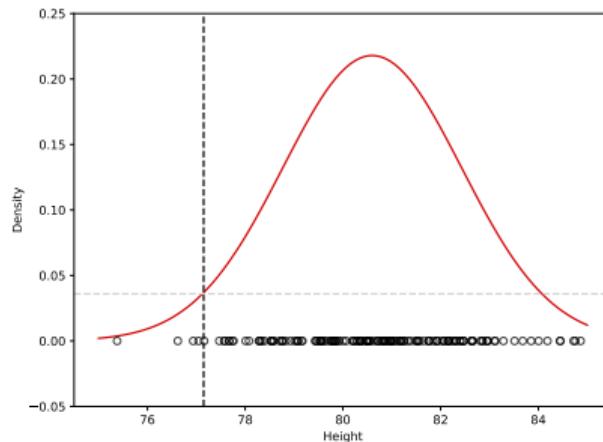


- The equation of the **univariate Gaussian**:

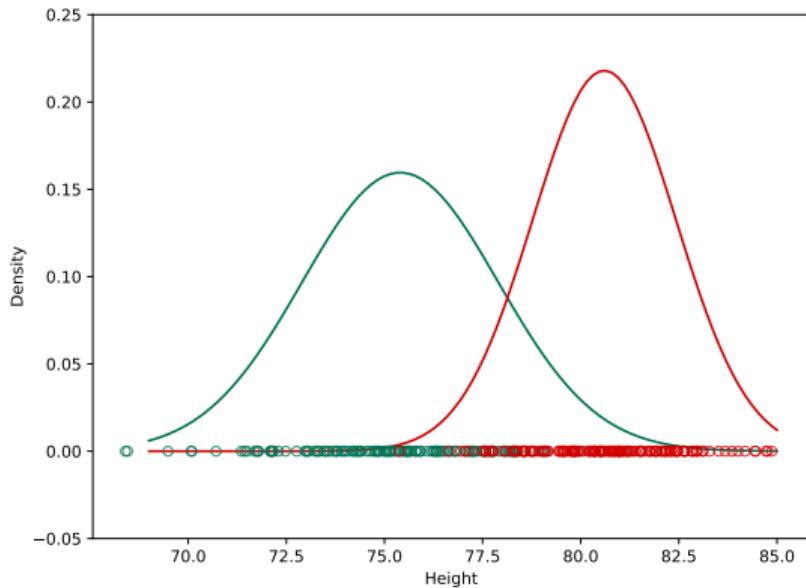
$$f(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$

Fitting a Normal Curve

- ▶ Calculate mean μ and STD σ from data.
- ▶ For forwards: $\mu = 80.5$, $\sigma = 1.84$



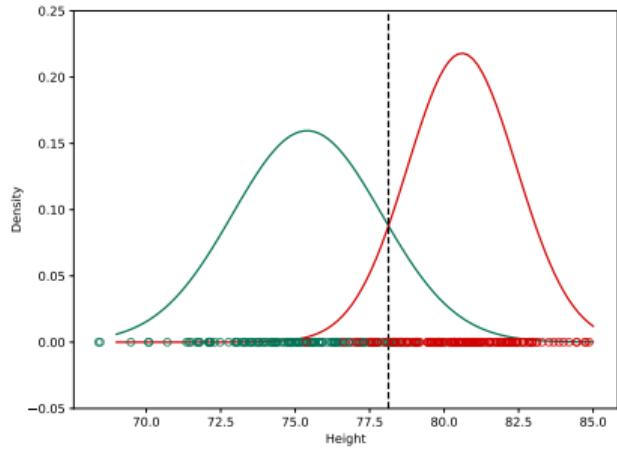
Fitting a Normal Curve to Each Class



Continuous A, Discrete B

- ▶ Two approaches: histograms and fitting a Gaussian.
- ▶ Histograms are **non-parametric**. No assumptions on shape.
- ▶ Fitting a Gaussian is **parametric**. Strong assumption.
- ▶ Other approaches exist.

Up next: using these estimates in the Bayes classifier.



CSE 151A
Intro to Machine Learning

Lecture 03 – Part 03
**Using the Bayes
Classifier**

Remember: The Bayes Classifier

- ▶ Given $X = x$, and possible classes $y_1, \dots, y_k \dots$
- ▶ ...predict the class y_i that makes $P(Y = y_i | X = x)$ the largest.

Bayes Classifier and Estimation

- ▶ The Bayes Classifier is optimal if **true** probabilities are used.
- ▶ If **estimated** probabilities are substituted, the classifier is no longer **optimal**, but still **good**.

Roadmap

We'll see two approaches:

1. Estimating $P(Y|X)$.
2. Estimating $P(X|Y)$ & $P(Y)$ and using Bayes' Rule.

Approach #1: Estimating $P(Y|X)$

- ▶ A new player's height is 77.15 in. What is their position?
- ▶ We need to estimate

$$P(Y = \text{Forward} | X = 77.15)$$

$$P(Y = \text{Guard} | X = 77.15)$$

and choose the largest.

Discrete A, Continuous B

- ▶ Estimate $P(Y = \text{forward} | X = 77.15)$



Discrete A, Continuous B

- ▶ Estimate $P(Y = \text{forward} | X = 77.15)$



- ▶ Use fraction of $k = 3$ nearest neighbors:

$$P(Y = \text{forward} | X = 77.15) \approx \frac{\text{2 red}}{3} = \frac{2}{3}$$

Discrete A, Continuous B

- ▶ Estimate $P(Y = \text{forward} | X = 77.15)$



- ▶ Use fraction of $k = 3$ nearest neighbors:

$$P(Y = \text{forward} | X = 77.15) \approx \frac{\text{2 red}}{3} = \frac{2}{3}$$

Does this seem familiar?

- ▶ We predict **forward** because the majority of the k neighbors are **red**.
- ▶ This is just the **k-Nearest Neighbor** classifier.

In fact...

- ▶ **Theorem:** the 1NN classifier has at most **twice** the Bayes Error as $n \rightarrow \infty$.

Approach #2: Use Bayes' Theorem

- ▶ Remember Bayes' Theorem:

$$P(Y | X) = \frac{P(X | Y)P(Y)}{P(X)}$$

Bayes Classifier after Bayes' Rule

- ▶ Before: predict y_i maximizing $P(Y = y_i | X = x)$.
- ▶ Bayes' Rule says that this is equivalent to picking y_i to maximize:

$$\frac{P(X = x | Y = y_i)P(Y = y_i)}{P(X = x)}$$

A Simplification...

- ▶ Only the numerator will change with different y_i .
- ▶ So pick y_i to maximize:

$$P(X = x | Y = y_i)P(Y = y_i)$$

- ▶ We know how to estimate both terms.

Example: Using Histograms

- ▶ A new player's height is 77.15 in. What is their position? (Assume binary classification.)
- ▶ We need to estimate

$$P(X = 77.15 \mid Y = \text{forward})$$

$$P(X = 77.15 \mid Y = \text{guard})$$

$$P(Y = \text{forward})$$

$$P(Y = \text{guard})$$

Example: Using Histograms

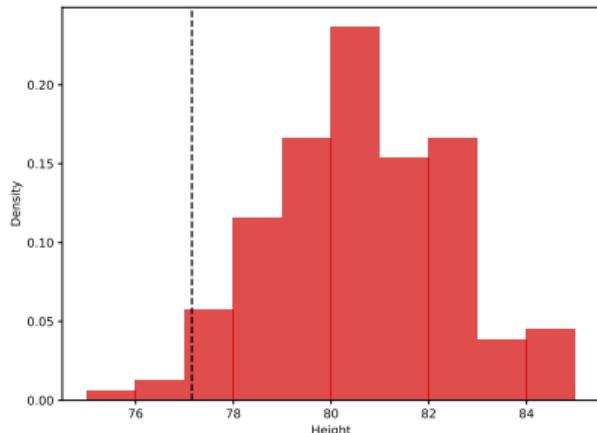
- ▶ Gather a training set of 300 players; 156 forwards, 144 guards.

$$P(Y = \text{forward}) \approx \frac{156}{300} = 0.52$$

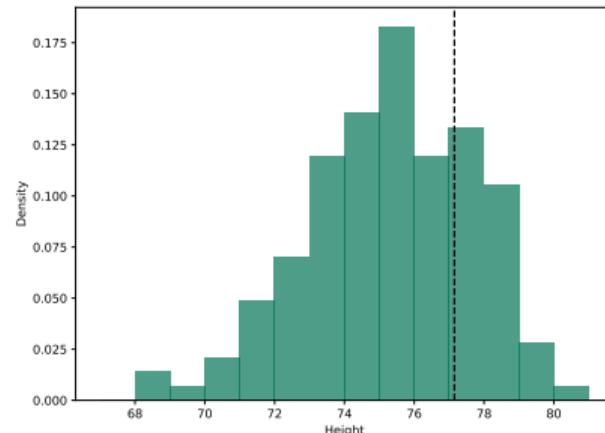
$$P(Y = \text{guard}) \approx \frac{144}{300} = 0.48$$

Example: Using Histograms

- ▶ Estimate conditional probs. with histograms.



$$P(X = 77.15 \mid Y = \text{Forward}) \approx .057$$



$$P(X = 77.15 \mid Y = \text{Guard}) \approx .134$$

Example: Using Histograms

- ▶ Therefore:

$$P(X = 77.15 \mid Y = \text{forward})P(Y = \text{forward}) \approx (.057)(.52) = .03$$

$$P(X = 77.15 \mid Y = \text{guard})P(Y = \text{guard}) \approx (.134)(.48) = .07$$

- ▶ Therefore, we predict **guard**.
- ▶ **Note:** probabilities do not add to one.

Example: Using Gaussians

- ▶ A new player's height is 77.15 in. What is their position? (Assume binary classification.)
- ▶ We need to estimate

$$P(X = 77.15 \mid Y = \text{forward})$$

$$P(X = 77.15 \mid Y = \text{guard})$$

$$P(Y = \text{forward})$$

$$P(Y = \text{guard})$$

Example: Using Gaussians

- ▶ Gather a training set of 300 players; 156 forwards, 144 guards.

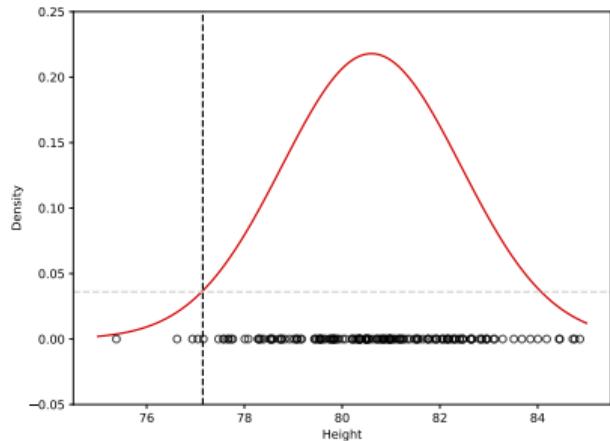
$$P(Y = \text{forward}) \approx \frac{156}{300} = 0.52$$

$$P(Y = \text{guard}) \approx \frac{144}{300} = 0.48$$

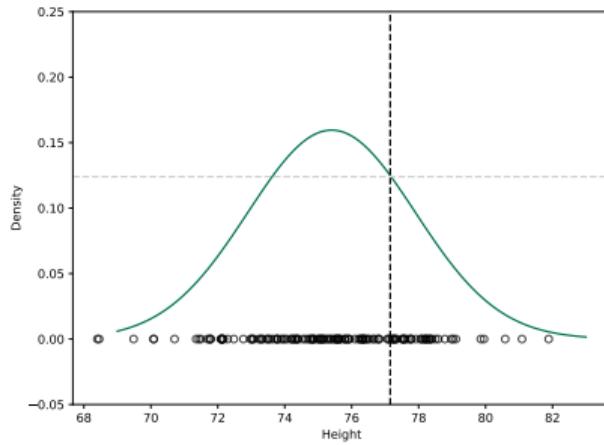
Example: Using Gaussians

- ▶ Estimate conditional probs. with Gaussians.
- ▶ For forwards: $\mu = 80.5, \sigma = 1.84$
- ▶ For guards $\mu = 75.4, \sigma = 2.5$
- ▶ Fit a Gaussian for each.

Example: Using Gaussians



$$P(X = 77.15 \mid Y = \text{Forward}) \approx .037$$



$$P(X = 77.15 \mid Y = \text{Guard}) \approx .125$$

Example: Using Gaussians

- ▶ Therefore:

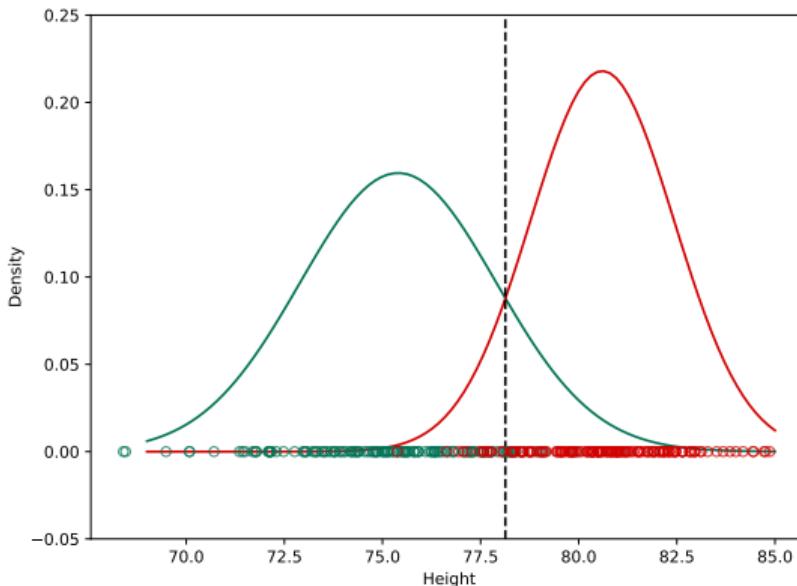
$$P(X = 77.15 \mid Y = \text{forward})P(Y = \text{forward}) \approx (.037)(.52) = .019$$

$$P(X = 77.15 \mid Y = \text{guard})P(Y = \text{guard}) \approx (.125)(.48) = .06$$

- ▶ Therefore, we predict **guard**.
- ▶ **Note:** probabilities do not add to one.

The Decision Boundary

- ▶ Plot $P(X = x | Y = \text{forward})P(Y = \text{forward})$ and $P(X = x | Y = \text{guard})P(Y = \text{guard})$



Results

- ▶ Train Error: 11.3%
- ▶ Test Error: 10%
- ▶ Best possible test error with simple boundary:
9.7%

Summary

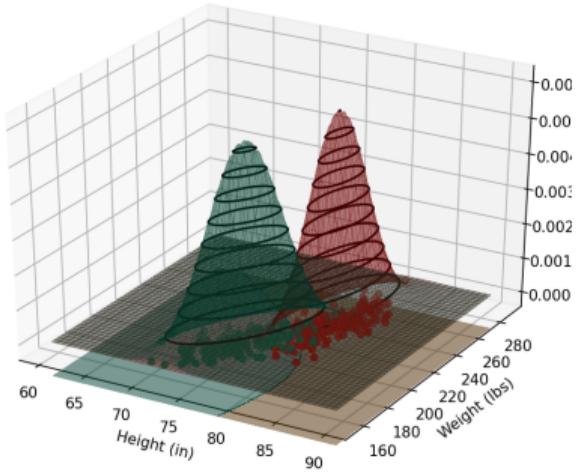
- ▶ Estimate conditional probabilities with histogram estimators or by fitting Gaussians.
- ▶ Pick y_i to maximize:

$$P(X = x | Y = y_i)P(Y = y_i)$$

- ▶ This is called the **generative** approach to classification.

Next time...

- ▶ We have only used one feature so far (height).
- ▶ How do we include more?



CSE 151A

Intro to Machine Learning

Lecture 04 – Part 01

Bayes with Multiple Features

Recap

- ▶ Bayes Classifier: predict y_i that maximizes $P(Y = y_i | X = x)$
- ▶ We have to estimate these probabilities.

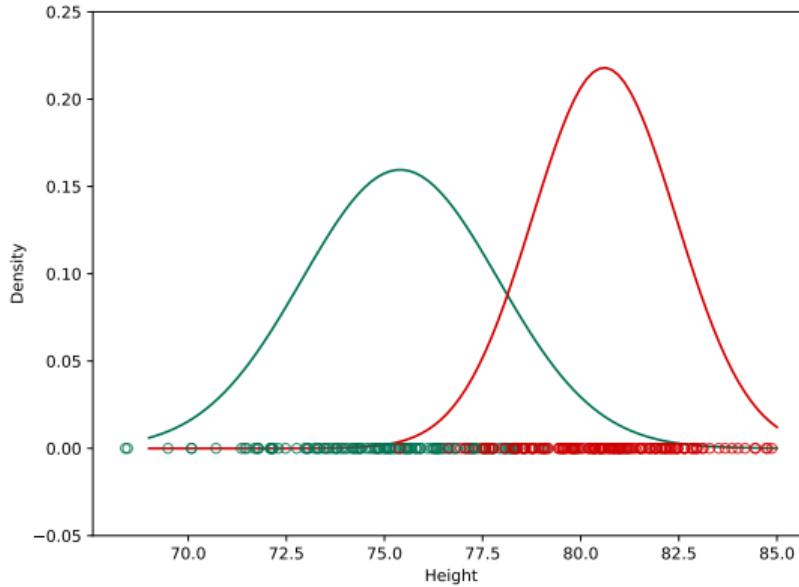
Recap

- ▶ Approach #1: Estimate $P(Y = y_i | X = x)$ using k neighbors.
- ▶ Approach #2: Use Bayes' Rule to write:

$$P(Y = y_i | X = x) = \frac{P(X = x | Y = y_i)P(Y = y_i)}{P(X = x)}$$

Estimate $P(X = x | Y = y_i)$ using histograms or by fitting Gaussians.

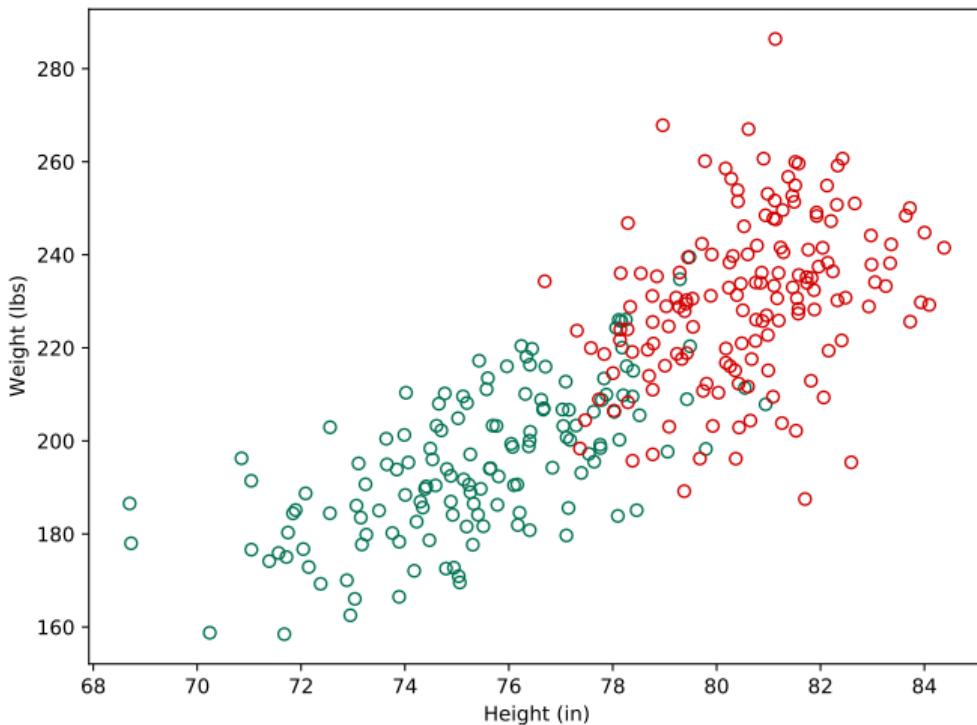
Recap



$P(Y = \text{guard} | X = x)P(Y = \text{guard})$ and $P(Y = \text{forward} | X = x)P(Y = \text{forward})$

Today

- ▶ How do we use more than one feature?
- ▶ Example: predict using height *and* weight.



Bayes in ≥ 2 Dimensions

- ▶ Instead of

$$P(Y = y_i | X = x)$$

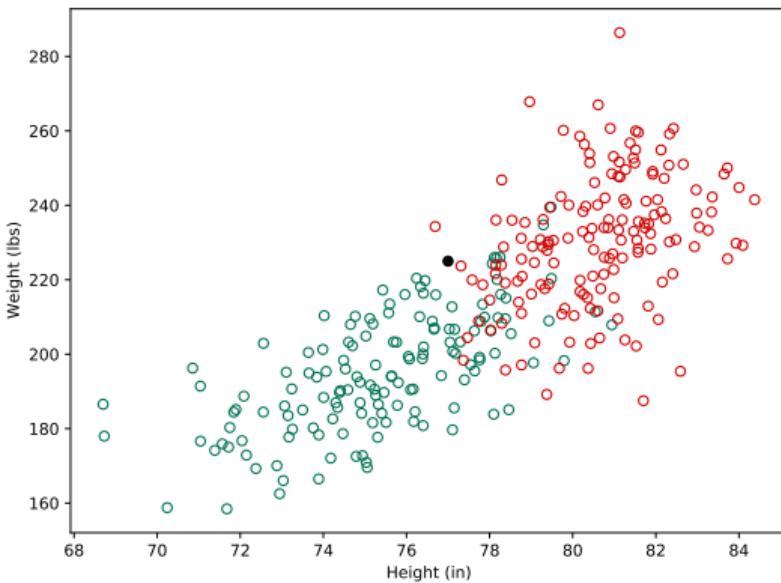
we have

$$P(Y = y_i | \vec{X} = \vec{x})$$

- ▶ \vec{x} is the **feature vector**. Here: $(\text{height}, \text{weight})^T$

Approach #1

- ▶ Can estimate $P(Y = y_i | X = x_i)$ using k neighbors.



Approach #2: Generative Modeling

- ▶ Use Bayes' Rule:

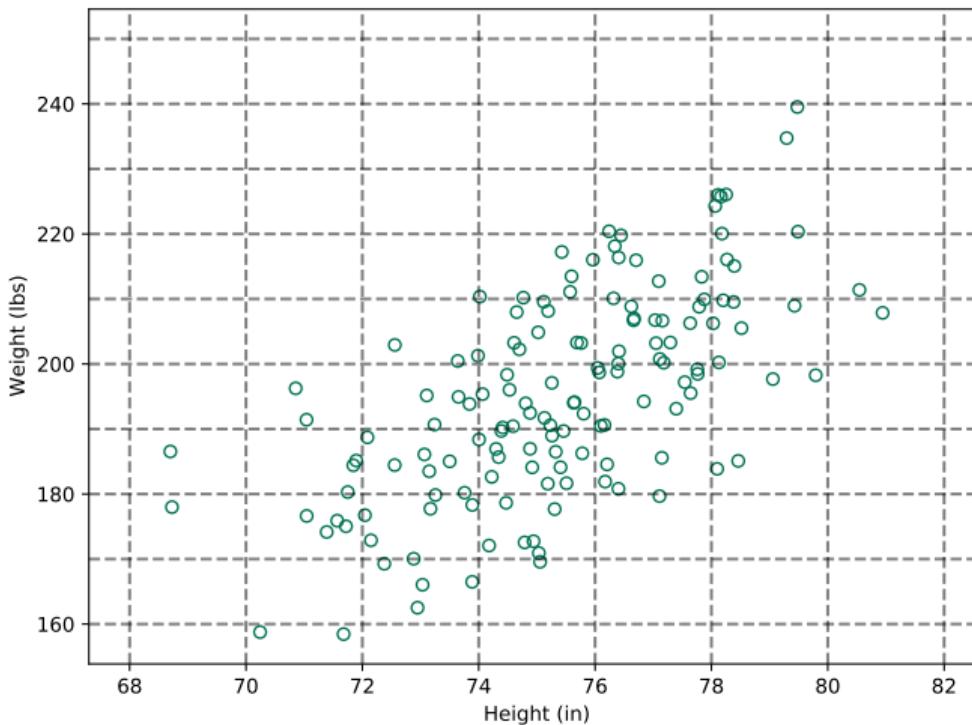
$$P(Y = y_i | \vec{X} = \vec{x}) = \frac{P(\vec{X} = \vec{x} | Y = y_i)P(Y = y_i)}{P(\vec{X} = \vec{x})}$$

- ▶ Estimate $P(\vec{X} = \vec{x} | Y = y_i)$ and $P(Y = y_i)$.

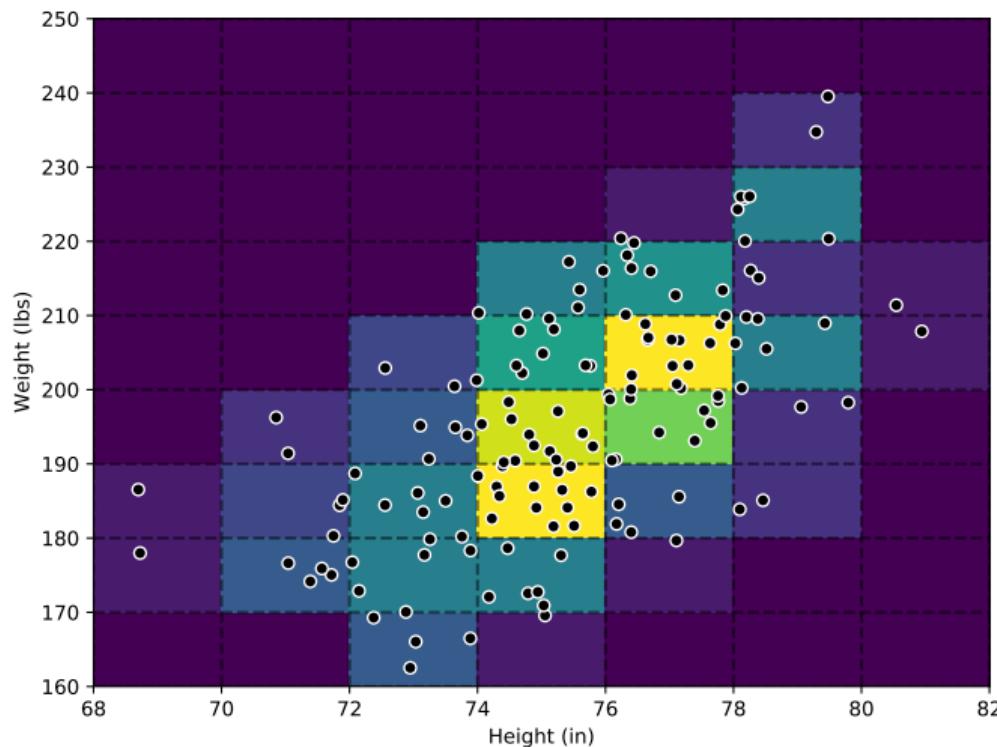
Estimating Density

- ▶ We need to estimate $P(\vec{X} = \vec{x} | Y = y_i)$ for each class y_1, \dots, y_k .
- ▶ See two methods: histograms and Gaussians.

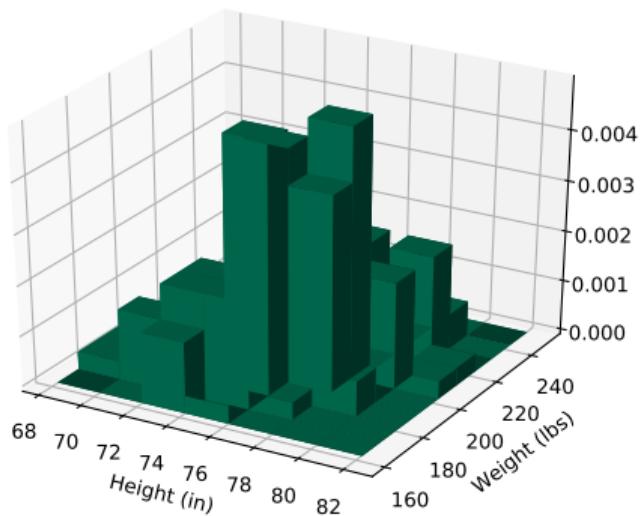
Estimating with Histograms



Estimating with Histograms



Estimating with Histograms



Predicting with Histograms

To predict the class of an input \vec{x} :

1. Use histograms to estimate $P(\vec{X} = \vec{x} | Y = y_i)$ for each class independently.
2. Predict the class y_i maximizing

$$P(\vec{X} = \vec{x} | Y = y_i)P(Y = y_i)$$

Histogram Estimators in $d > 2$

- ▶ With 1 feature, each bin is an interval.
- ▶ With 2 features, each bin is a rectangle.
- ▶ With 3 features, each bin is a cuboid (box).
- ▶ With >4 features, each bin is a **hypercuboid**.

Curse of Dimensionality

- ▶ We need enough bins to “cover” the input space.
- ▶ **Problem:** Number of bins is exponential in d .
- ▶ Example: split each dimension into 10 pieces.

Example

- ▶ In 2-d: $10^2 = 100$ bins.

Example

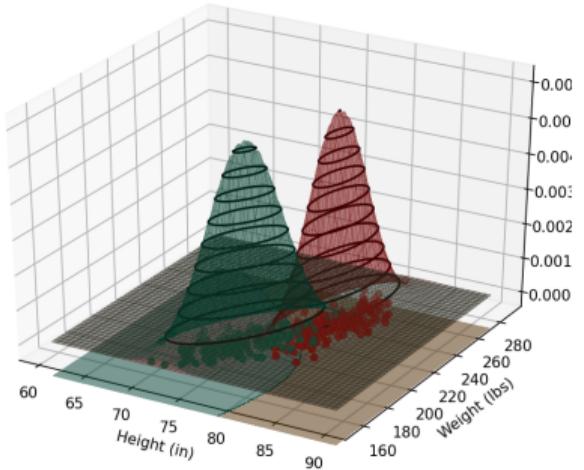
- ▶ In 2-d: $10^2 = 100$ bins.
- ▶ In 100-d: 10^{100} bins.
- ▶ More bins than atoms in universe.
- ▶ Highly likely that no bin has more than a few points.

Histogram Estimators

- ▶ Histogram density estimators are very general.
- ▶ But suffer heavily from **curse of dimensionality**.
- ▶ Then again, so do most things.

Up next...

- ▶ What about fitting Gaussians?



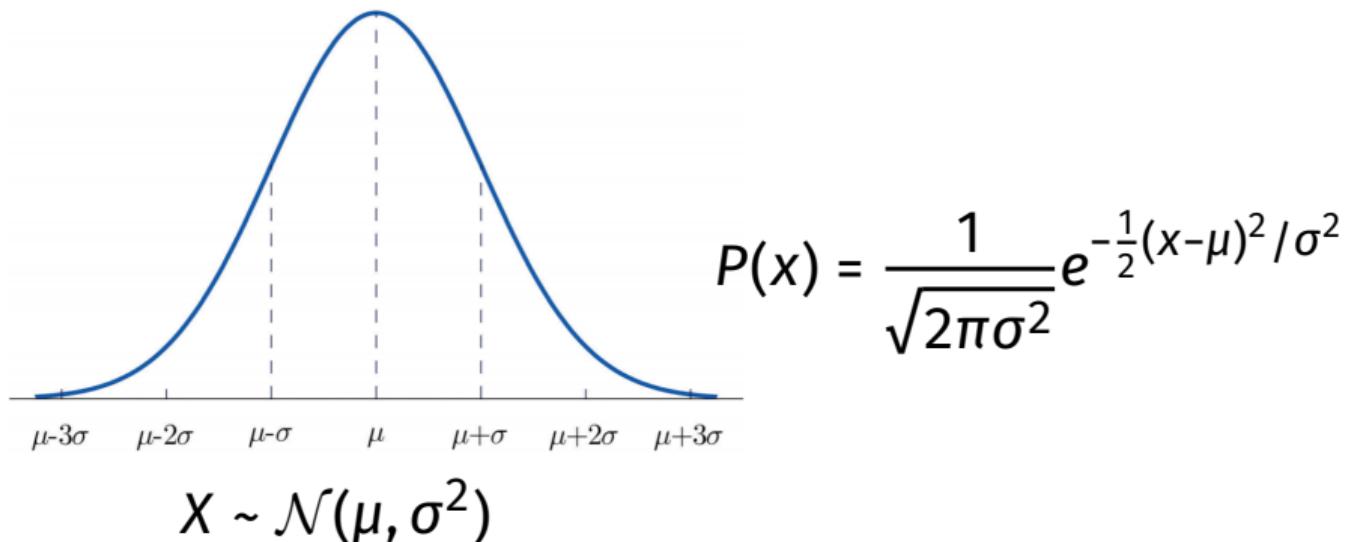
CSE 151A
Intro to Machine Learning

Lecture 04 – Part 02
Multivariate Gaussians

Multivariate Gaussians

- ▶ In 1 dimension, a Gaussian seemed to describe distribution of heights.
- ▶ Does a **multivariate** Gaussian describe distribution of heights and weights?

“Deriving” Multivariate Gaussians



Setting #1

- ▶ Suppose we have d independent random variables X_1, \dots, X_d .
- ▶ Assume that each is Gaussian; different mean, but **same** variance:

$$X_1 \sim \mathcal{N}(\mu_1, \sigma^2), \quad X_2 \sim \mathcal{N}(\mu_2, \sigma^2), \dots, \quad X_d \sim \mathcal{N}(\mu_d, \sigma^2).$$

Setting #1

- ▶ What is $P(x_1, x_2, \dots, x_d)$?
- ▶ Since we assumed X_1, \dots, X_d are independent:

$$\begin{aligned} P(x_1, x_2, \dots, x_d) &= P(x_1)P(x_2) \cdots P(x_d) \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{1}{2}(x-\mu_1)^2/\sigma^2} \right) \cdot \left(\frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{1}{2}(x-\mu_2)^2/\sigma^2} \right) \cdots \left(\frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{1}{2}(x-\mu_d)^2/\sigma^2} \right) \end{aligned}$$

Setting #1

- ▶ What is $P(x_1, x_2, \dots, x_d)$?
- ▶ Since we assumed X_1, \dots, X_d are independent:

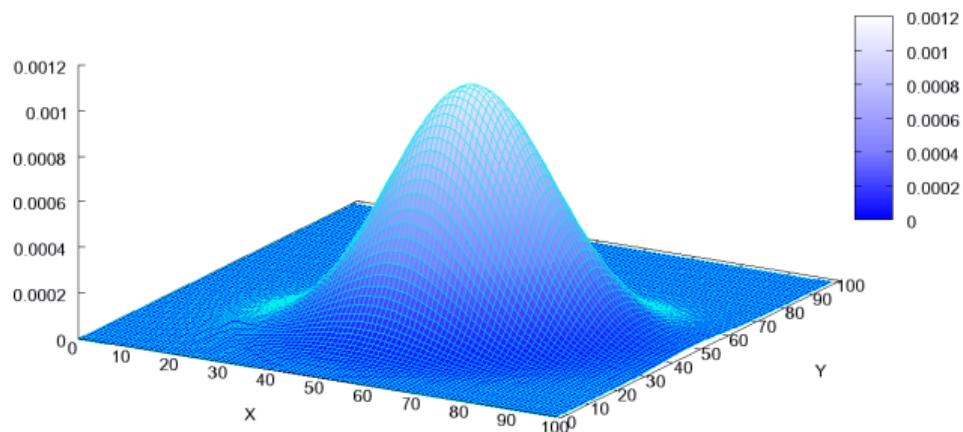
$$\begin{aligned} P(x_1, x_2, \dots, x_d) &= P(x_1)P(x_2) \cdots P(x_d) \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{1}{2}(x-\mu_1)^2/\sigma^2} \right) \cdot \left(\frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{1}{2}(x-\mu_2)^2/\sigma^2} \right) \cdots \left(\frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{1}{2}(x-\mu_d)^2/\sigma^2} \right) \\ &= \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{(x_1 - \mu_1)^2 + (x_2 - \mu_2)^2 + \cdots + (x_d - \mu_d)^2}{2\sigma^2}\right) \end{aligned}$$

Setting #1

- ▶ What is $P(x_1, x_2, \dots, x_d)$?
- ▶ Since we assumed X_1, \dots, X_d are independent:

$$\begin{aligned} P(x_1, x_2, \dots, x_d) &= P(x_1)P(x_2) \cdots P(x_d) \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{1}{2}(x-\mu_1)^2/\sigma^2} \right) \cdot \left(\frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{1}{2}(x-\mu_2)^2/\sigma^2} \right) \cdots \left(\frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{1}{2}(x-\mu_d)^2/\sigma^2} \right) \\ &= \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{(x_1 - \mu_1)^2 + (x_2 - \mu_2)^2 + \cdots + (x_d - \mu_d)^2}{2\sigma^2}\right) \\ &= \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{\|\vec{x} - \vec{\mu}\|^2}{2\sigma^2}\right) \end{aligned}$$

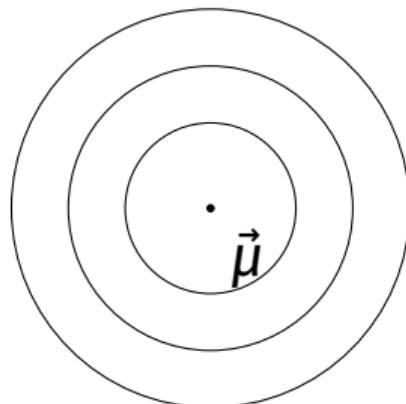
Setting #1



Setting #1: Spherical Gaussians

$$P(\vec{x}) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{1}{2} \frac{\|\vec{x} - \vec{\mu}\|^2}{\sigma^2}\right)$$

- ▶ Contours are (hyper)spheres.
- ▶ Every slice through middle gives same Gaussian.



Setting #2

- ▶ Still assume X_1, \dots, X_d are independent, Normal.
- ▶ But they now have different variances:

$$X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2), \quad X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2), \dots, \quad X_d \sim \mathcal{N}(\mu_d, \sigma_d^2).$$

Setting #2

$$\begin{aligned} P(x_1, x_2, \dots, x_d) &= P(x_1)P(x_2)\cdots P(x_d) \\ &= \left(\frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{1}{2}(x-\mu_1)^2/\sigma_1^2} \right) \cdot \left(\frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{1}{2}(x-\mu_2)^2/\sigma_2^2} \right) \cdots \left(\frac{1}{\sqrt{2\pi\sigma_d^2}} e^{-\frac{1}{2}(x-\mu_d)^2/\sigma_d^2} \right) \end{aligned}$$

Setting #2

$$\begin{aligned} P(x_1, x_2, \dots, x_d) &= P(x_1)P(x_2) \cdots P(x_d) \\ &= \left(\frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{1}{2}(x-\mu_1)^2/\sigma_1^2} \right) \cdot \left(\frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{1}{2}(x-\mu_2)^2/\sigma_2^2} \right) \cdots \left(\frac{1}{\sqrt{2\pi\sigma_d^2}} e^{-\frac{1}{2}(x-\mu_d)^2/\sigma_d^2} \right) \\ &= \frac{1}{(2\pi)^{d/2}\sigma_1 \cdot \sigma_2 \cdots \sigma_d} \exp\left(-\frac{1}{2} \left[\frac{(x_1 - \mu_1)^2}{\sigma_1^2} + \frac{(x_2 - \mu_2)^2}{\sigma_2^2} + \cdots + \frac{(x_d - \mu_d)^2}{\sigma_d^2} \right]\right) \end{aligned}$$

Setting #2

- ▶ Define

$$C = \begin{pmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \sigma_d^2 \end{pmatrix}$$

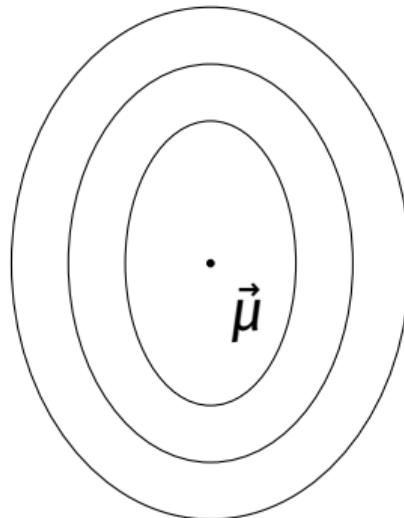
- ▶ Then:

$$P(\vec{x}) = \frac{1}{(2\pi)^{d/2} |C|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu})^T C^{-1}(\vec{x} - \vec{\mu})\right)$$

Setting #2: Diagonal Gaussians

$$P(\vec{x}) = \frac{1}{(2\pi)^{d/2} |C|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu})^T C^{-1}(\vec{x} - \vec{\mu})\right)$$

- ▶ Contours are axis-aligned (hyper)ellipses.
- ▶ C is the **covariance matrix**.
 - ▶ Diagonal.
 - ▶ Entries are variances.



Setting #3: General Gaussians

- ▶ We have assumed that X_1, \dots, X_d are independent.
- ▶ Now assume that they're not. Define **covariance**:

$$\text{Cov}(X_i, X_j) = \mathbb{E}[(X_i - \mu_i)(X_j - \mu_j)]$$

- ▶ **Note:**

$$\text{Var}(X_i) = \text{Cov}(X_i, X_j)$$

Setting #3: General Gaussians

- Now the **covariance matrix** has off-diagonal elements:

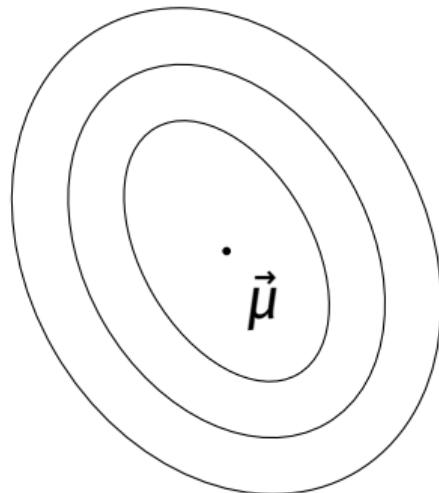
$$C = \begin{pmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) & \cdots & \text{Cov}(X_1, X_d) \\ \text{Cov}(X_2, X_1) & \text{Var}(X_2) & \cdots & \text{Cov}(X_2, X_d) \\ \cdots & \cdots & \cdots & \cdots \\ \text{Cov}(X_d, X_1) & \text{Cov}(X_d, X_2) & \cdots & \text{Var}(X_d) \end{pmatrix}$$

- Since $\text{Cov}(X_i, X_j) = \text{Cov}(X_j, X_i)$, C is symmetric.

Setting #3: General Gaussians

$$P(\vec{x}) = \frac{1}{(2\pi)^{d/2} |C|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu})^T C^{-1}(\vec{x} - \vec{\mu})\right)$$

Contours are general (hyper)ellipses.
 C need not be diagonal.



Fitting Multivariate Gaussians

- ▶ Given vectors $\vec{x}^{(1)}, \dots, \vec{x}^{(n)}$, fit a Gaussian.
- ▶ First, choose assumptions. Spherical? Diagonal? General (no assumptions)?
- ▶ In each case,

$$\vec{\mu} = \frac{1}{n} \sum_{i=1}^n \vec{x}^{(i)}$$

Fitting Spherical Gaussians

- ▶ Only one variance: σ^2 .

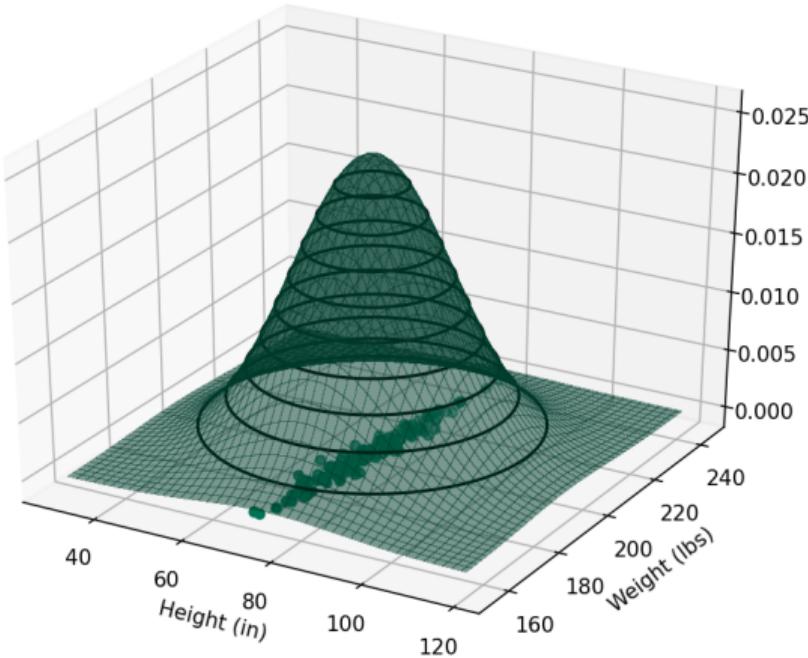
- ▶ In 1 dimension:

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

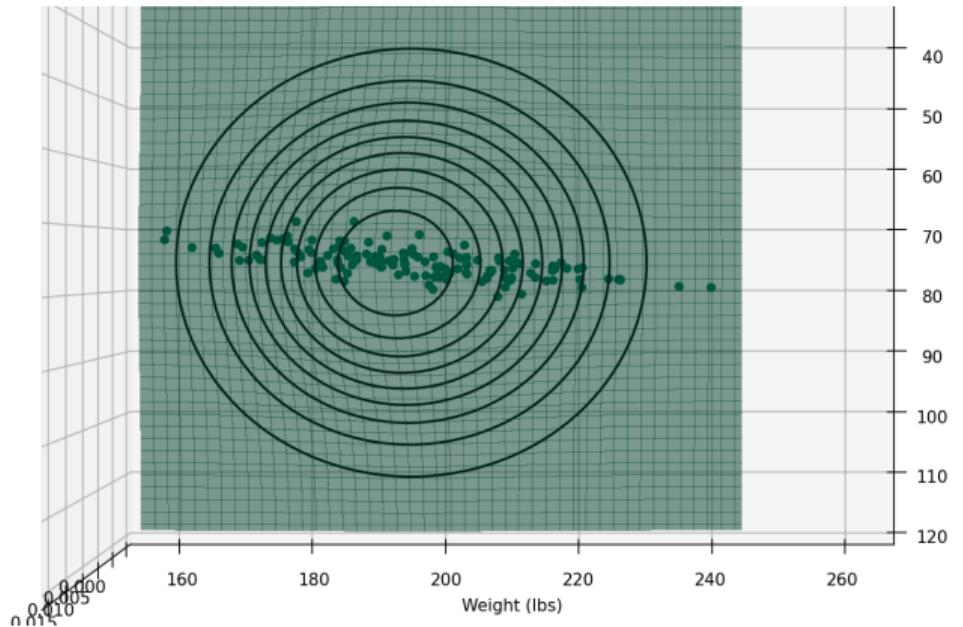
- ▶ In d dimensions:

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n \|\vec{x}^{(i)} - \vec{\mu}\|^2$$

Fitting Spherical Gaussians



Fitting Spherical Gaussians



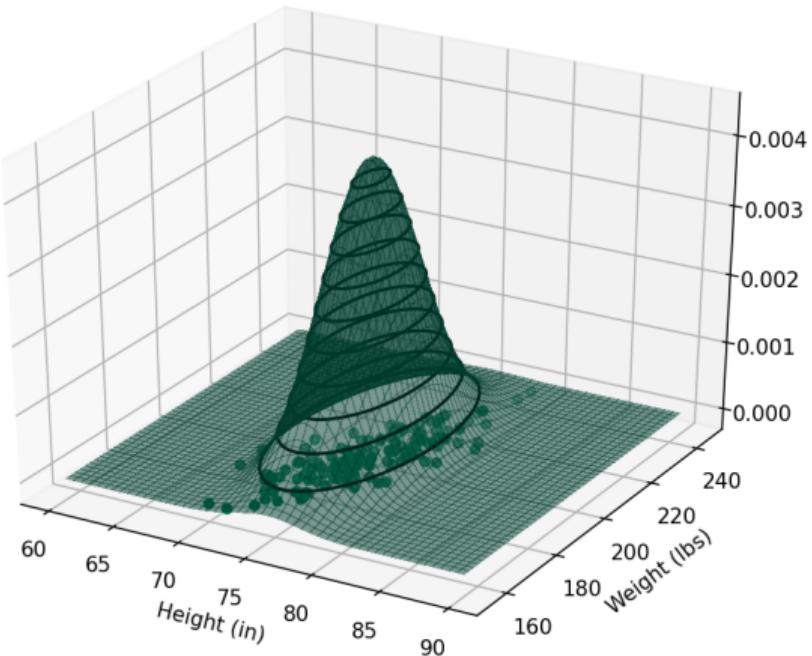
Fitting Diagonal Gaussians

- ▶ Variance for each axis: σ_1^2 and σ_2^2 .
- ▶ Example:

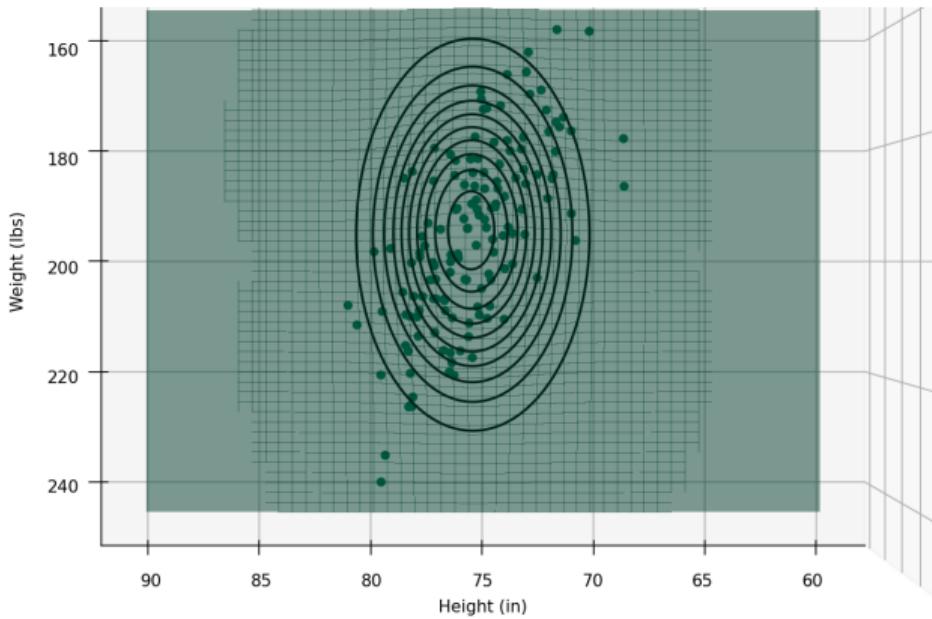
σ_1^2 = variance of heights

σ_2^2 = variance of weights

Fitting Diagonal Gaussians



Fitting Diagonal Gaussians



Fitting General Gaussians

- ▶ Must compute covariance for each pair of dimensions.
- ▶ Empirical covariance:

$$C_{ij} = \left(\frac{1}{n} \sum_{k=1}^n \vec{x}_i^{(k)} \vec{x}_j^{(k)} \right) - \mu_i \mu_j$$

Computing the Covariance Matrix

Step 1. Make matrix with heights in first column, weights in second:

$$\begin{pmatrix} \text{height 1} & \text{weight 1} \\ \text{height 2} & \text{weight 2} \\ \dots & \dots \\ \text{height } n & \text{weight } n \end{pmatrix}$$

Computing the Covariance Matrix

Step 2. Subtract mean height, mean weight from each column. Call this matrix X :

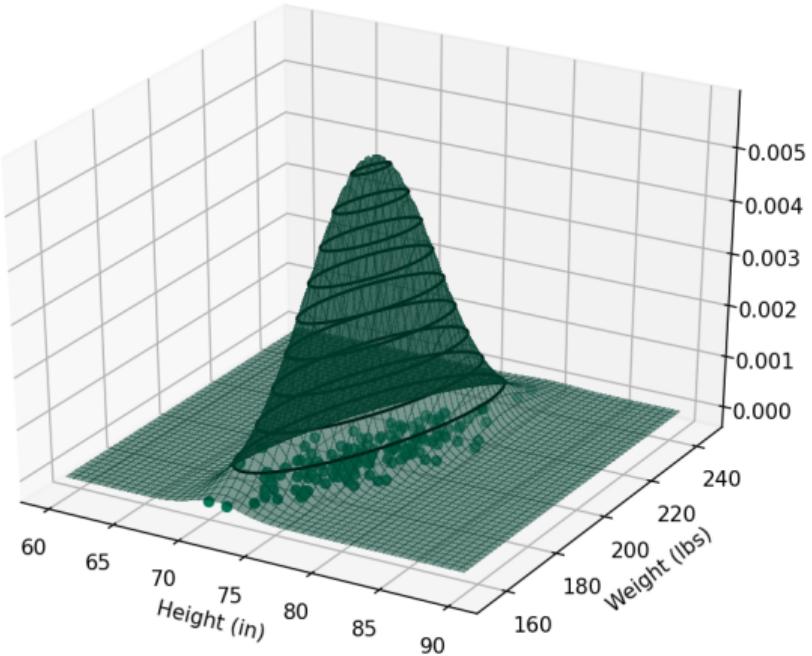
$$X = \begin{pmatrix} \text{height 1 - mean height} & \text{weight 1 - mean weight} \\ \text{height 2 - mean height} & \text{weight 2 - mean weight} \\ \dots & \dots \\ \text{height } n - \text{mean height} & \text{weight } n - \text{mean weight} \end{pmatrix}$$

Computing the Covariance Matrix

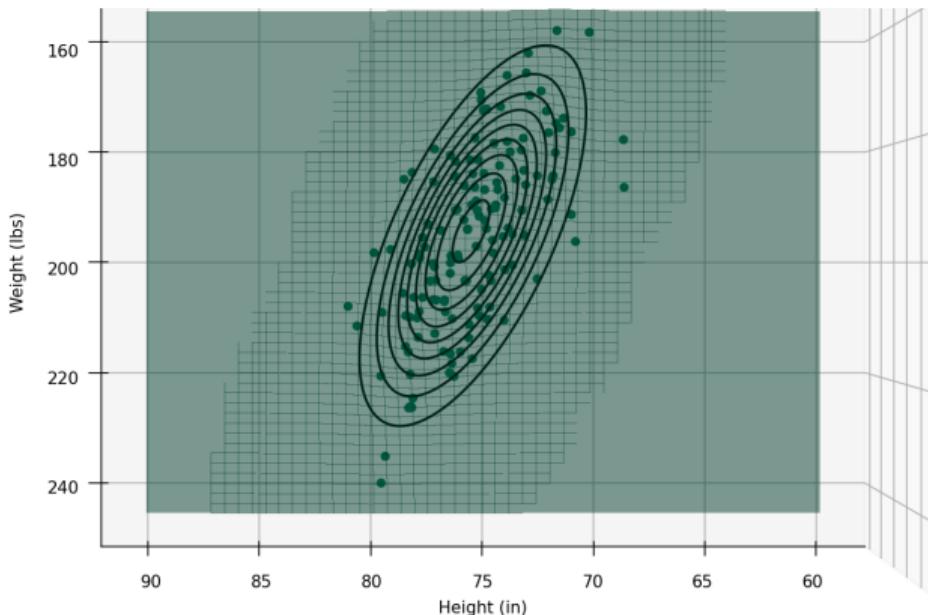
The empirical covariance matrix is then:

$$C = \frac{1}{n} X^T X$$

Fitting General Gaussians

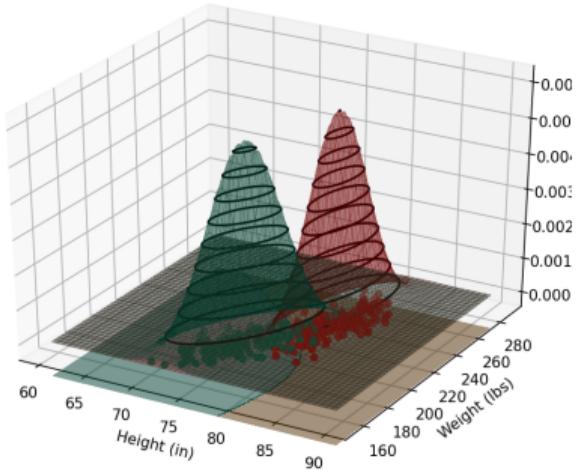


Fitting General Gaussians



Up next...

Making predictions using these fitted Gaussians.



CSE 151A
Intro to Machine Learning

Lecture 04 – Part 03
Discriminant Analysis

Bayes Classifier with MV Gaussians

1. Fit Gaussian for $P(\vec{X} | Y = y_i)$ for each class, y_i .
2. For new point, predict y_i maximizing:

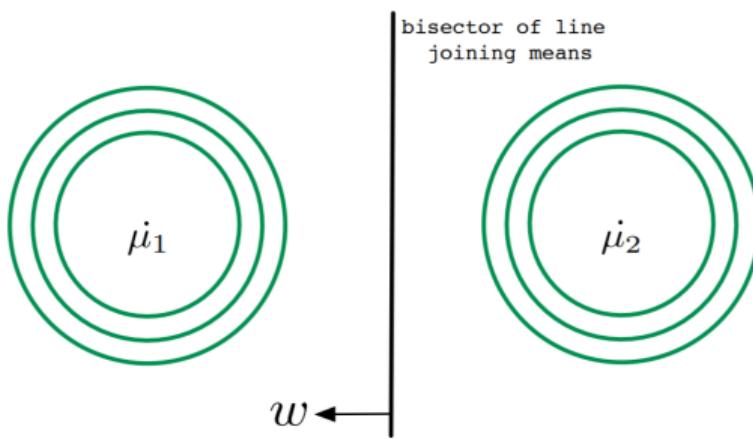
$$P(\vec{X} = \vec{x} | Y = y_i)P(Y = y_i)$$

Decision Boundary

- ▶ For every point in space, we have a classification.
- ▶ The **decision boundary**: surface between different classifications.
 - ▶ On one side, prediction is y_1 ;
 - ▶ on the other, prediction is y_2 .

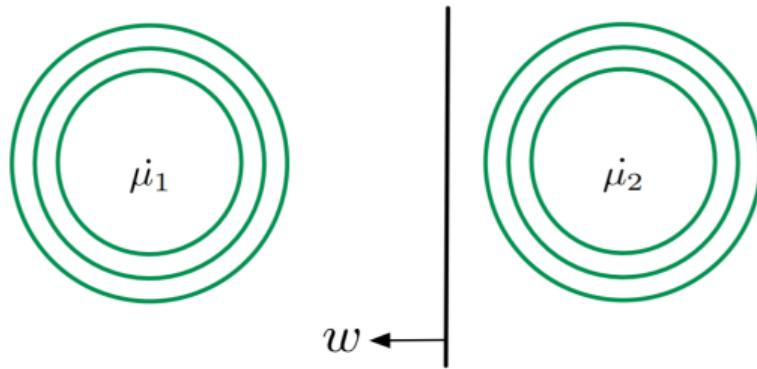
Setting #1

- ▶ Assume:
 - ▶ only two classes (binary classification)
 - ▶ covariance matrices identical, spherical



Setting #1

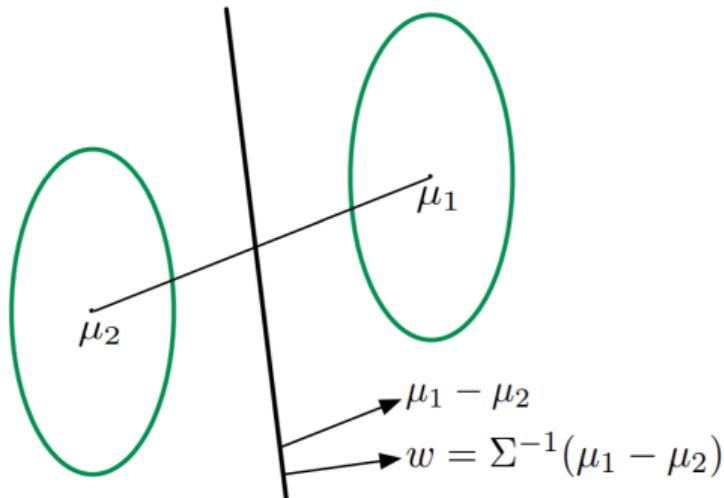
- If $P(Y = y_1) > P(Y = y_2)$:



Choose class 1 if $\vec{w} \cdot \frac{(\vec{\mu}_1 - \vec{\mu}_2)}{\sigma^2} \geq \theta$.

Setting #2

- ▶ Assume:
 - ▶ only two classes (binary classification)
 - ▶ covariance matrices identical, non-diagonal



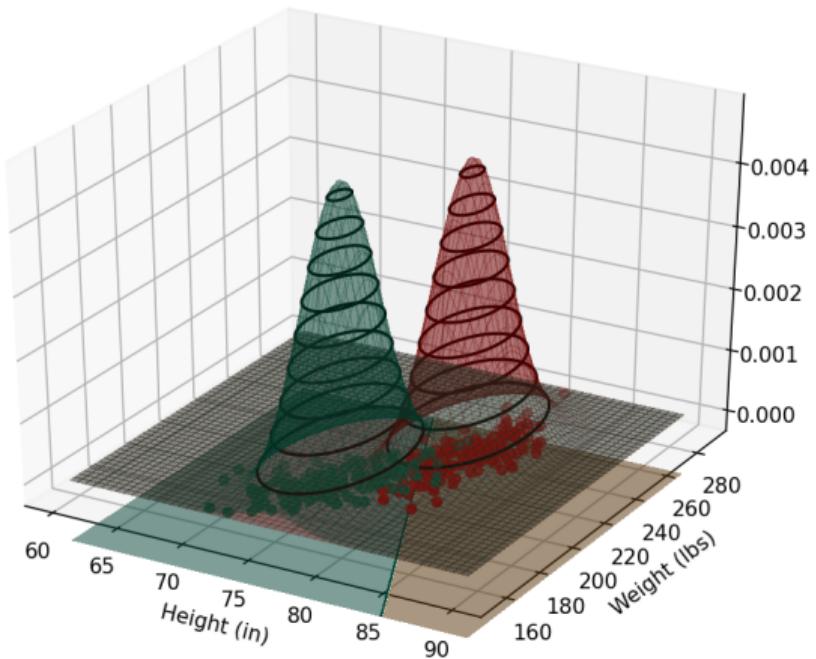
Predict class 1 if
 $\vec{x} \cdot \vec{w} \geq \theta$.

Example

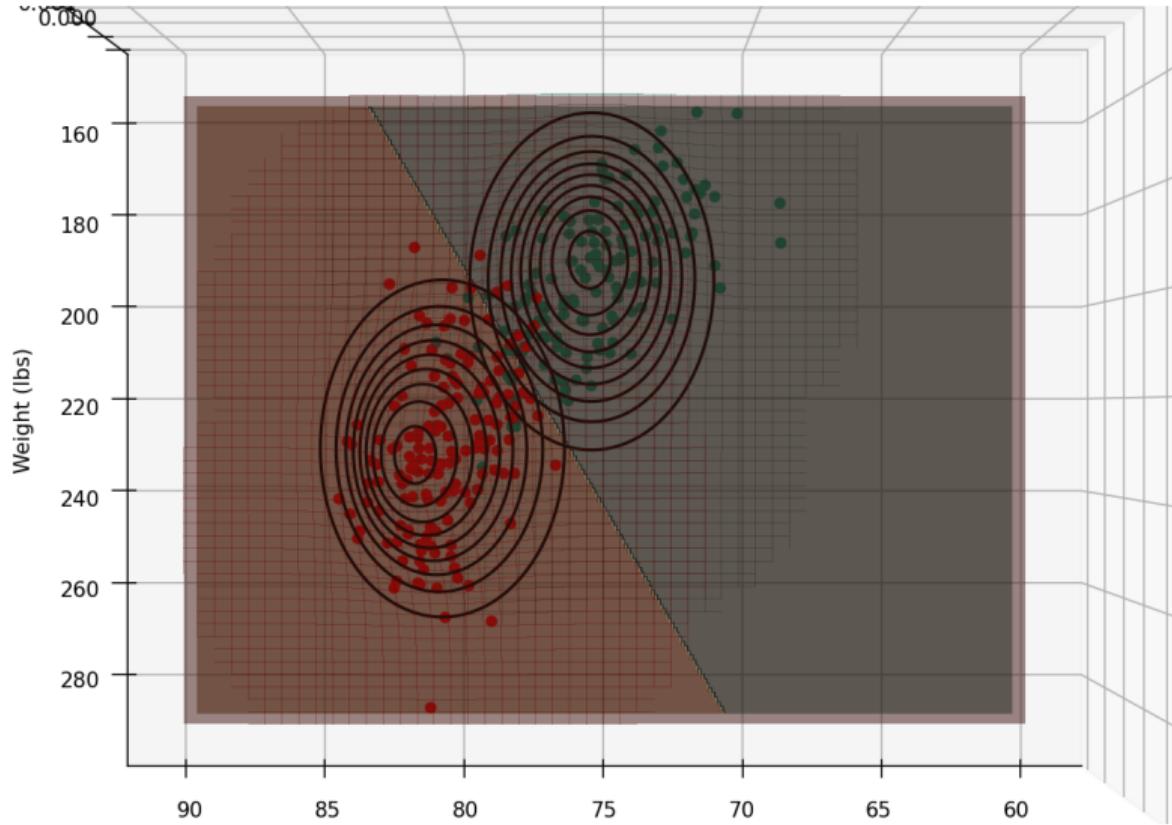
- ▶ Use to predict position given height and weight.
- ▶ How do we get one covariance matrix?
- ▶ Don't lump data together...
- ▶ Instead, compute covariance matrix for each class, perform weighted average:

$$C = \frac{n_1 C_1 + n_2 C_2}{n_1 + n_2}$$

Example



Example

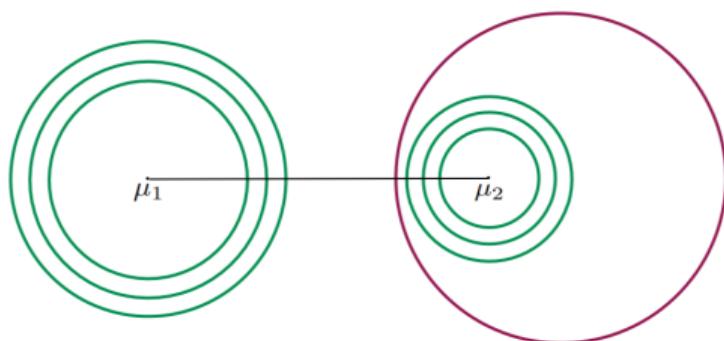


Linear Discriminant Analysis

- ▶ When covariance matrices are equal, decision boundary is linear.
- ▶ This procedure is called **linear discriminant analysis** (LDA).

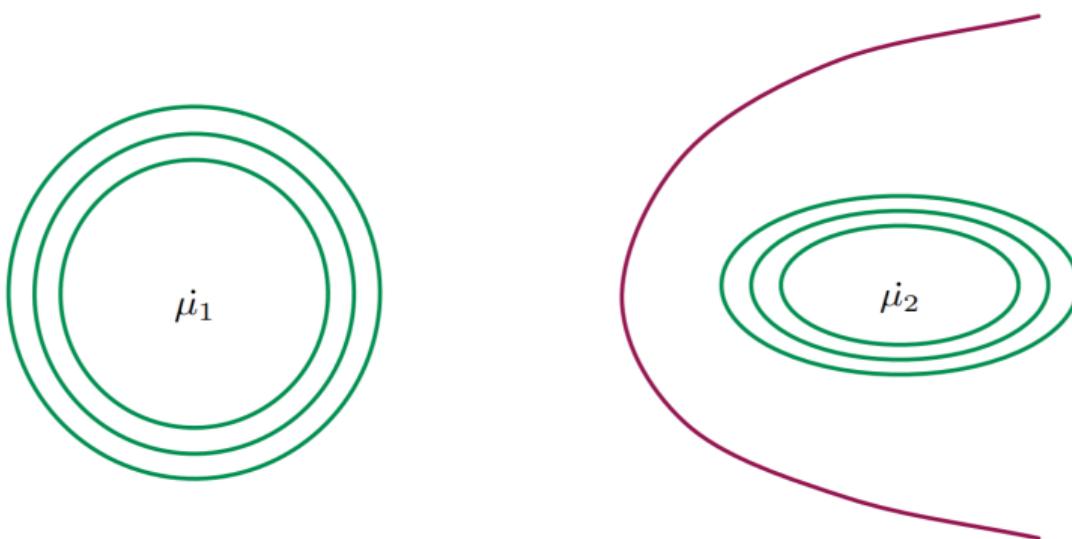
Setting #3

- ▶ Assume:
 - ▶ only two classes (binary classification)
 - ▶ covariance matrices C_1, C_2 different, non-diagonal

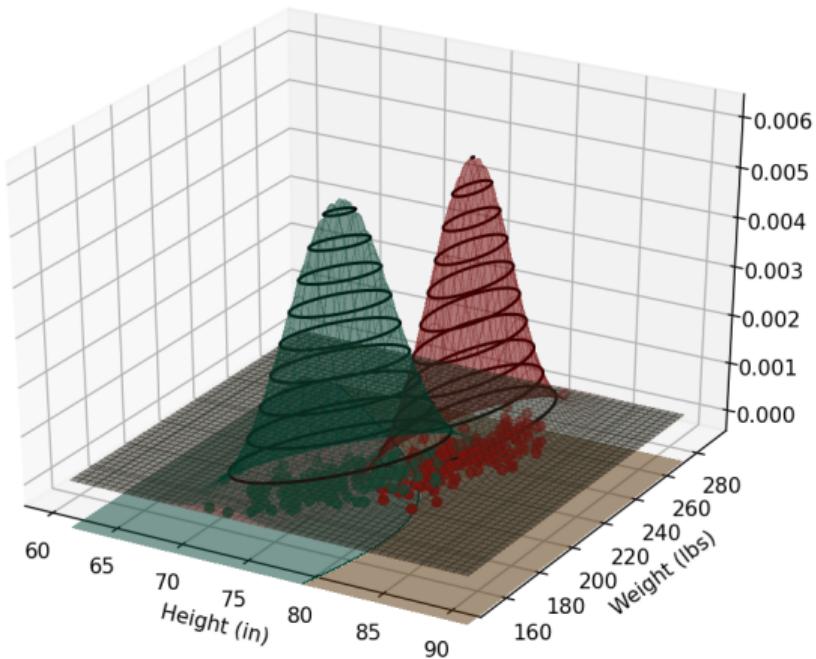


Setting #3

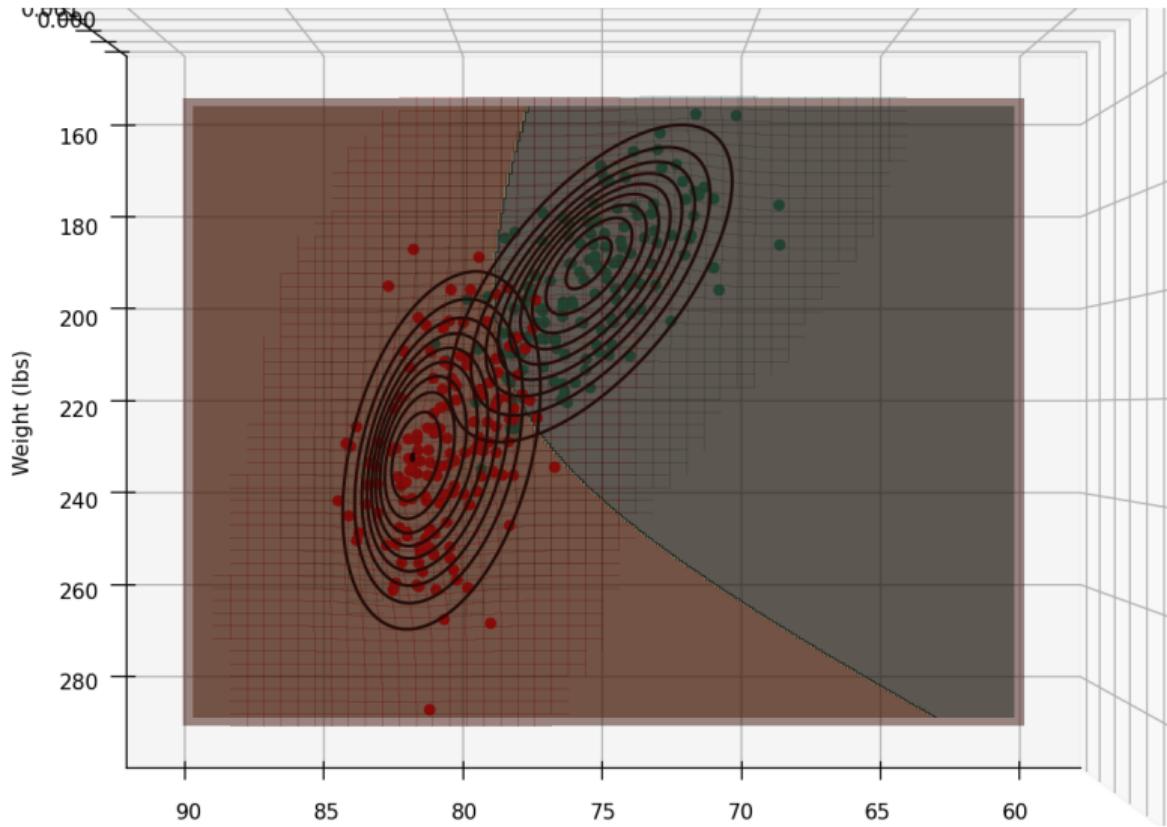
- ▶ Assume:
 - ▶ only two classes (binary classification)
 - ▶ covariance matrices C_1, C_2 different, non-diagonal



Example



Example

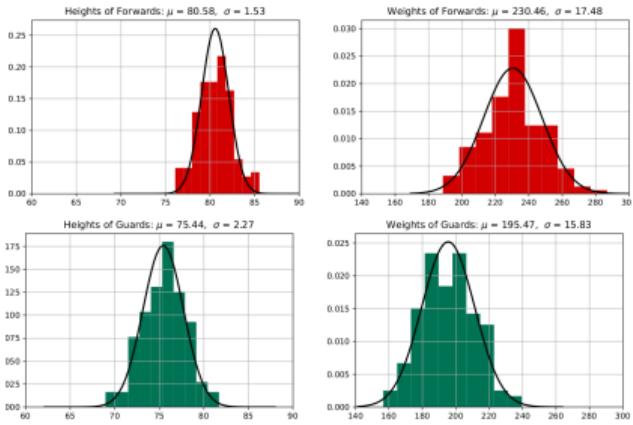


Quadratic Discriminant Analysis

- ▶ When covariance matrices are equal, decision boundary is quadratic (ellipsoidal, paraboloidal, hyperboloidal).
- ▶ This procedure is called **quadratic discriminant analysis** (QDA).

In practice...

- ▶ LDA and QDA can work well.
- ▶ A full covariance requires estimating $\Theta(d^2)$ parameters.
- ▶ Gaussian assumption may be poor.



CSE 151A

Intro to Machine Learning

Lecture 05 – Part 01

What is Conditional Independence?

$$P(A \text{ and } B) = P(A) \cdot P(B)$$

Remember: Independence

- ▶ Events A and B are **independent** if

$$P(A, B) = P(A) \cdot P(B).$$

- ▶ Equivalently, A and B are independent if¹

$$P(A | B) = P(A)$$

¹or $P(B) = 0$

Informally

- ▶ A and B are **independent** if learning B does not influence your belief that A happens.

Example

You draw one card from a deck of 52 cards. A is the event that the card is a heart, B is the event that the card is a face card (J,Q,K,A). Are these independent?

♥: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A

♦: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A

♣: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A

♠: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A

Example

We've lost the King of Clubs! You draw one card from this deck of 51 cards. A is the event that the card is a heart, B is the event that the card is a face card (J,Q,K,A). Are these independent?

♥: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A

♦: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A

♣: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, A

♠: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A

In the Real World...

- ▶ ...true independence is rare.
 - ▶ Example, survivors of the titanic:

In the Real World...

- ▶ $P(\text{Survived} = 1) = .408$
- ▶ $P(\text{Survived} = 1 \mid \text{FavColor} = \text{purple}) = .4$
- ▶ **Not independent...**

In the Real World...

- ▶ $P(\text{Survived} = 1) = .408$
- ▶ $P(\text{Survived} = 1 \mid \text{FavColor} = \text{purple}) = .4$
- ▶ **Not independent... ...but “close”!**

In the Real World...

- ▶ $P(\text{Survived} = 1) = .408$
- ▶ $P(\text{Survived} = 1 \mid \text{Pclass} = 1) =$

In the Real World...

- ▶ $P(\text{Survived} = 1) = .408$
- ▶ $P(\text{Survived} = 1 \mid \text{Pclass} = 1) = .657$

In the Real World...

- ▶ $P(\text{Survived} = 1) = .408$
- ▶ $P(\text{Survived} = 1 \mid \text{Pclass} = 1) = .657$
- ▶ **Strong dependence.**

Remember: Conditional Independence

- ▶ Events A and B are **conditionally independent** given C if

$$P(A, B | C) = P(A | C) \cdot P(B | C)$$

- ▶ Equivalently²:

$$P(A | B, C) = P(A | C)$$

²Or $P(B) = 0$

Informally

- ▶ Suppose you know that C has happened.
- ▶ You have some belief that A happens, given C .
- ▶ A and B are **conditionally independent** given C if learning that B happens in addition to C does not influence your belief that A happens given C .

Very informally

- ▶ A and B are **conditionally independent** given C if learning that B happens in addition to C gives you no more information about A .

Example

We've lost the King of Clubs! You draw one card from this deck of 51 cards. A is the event that the card is a heart, B is the event that the card is a face card (J,Q,K,A). Now suppose you know that the card is red. Are A and B independent **given** this information?

♥: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A

♦: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A

♣: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, A

♠: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A

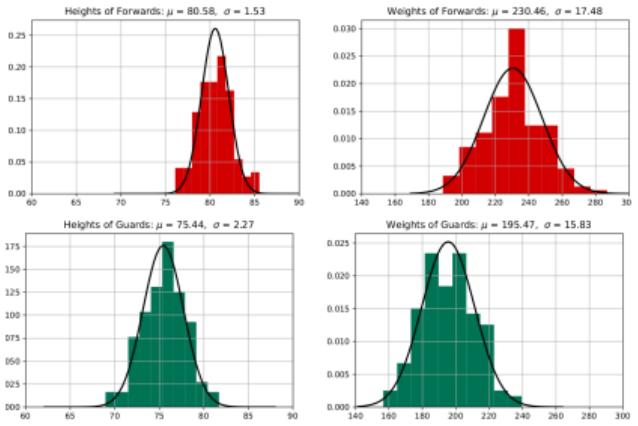
Titanic Example

- ▶ Survival and class are **not** independent.
- ▶ But they're (close) to **conditionally independent** given ticket price:
 - ▶ $P(\text{Survived} = 1 \mid \text{PClass} = 1, \text{Fare} > 50) = .708$
 - ▶ $P(\text{Survived} = 1 \mid \text{Fare} > 50) = .696$

More Variables

- ▶ X_1, X_2, \dots, X_d are **mutually conditionally independent** given Y if

$$P(X_1, X_2, \dots, X_d | Y) = P(X_1 | Y) \cdot P(X_2 | Y) \cdots P(X_d | Y)$$



CSE 151A

Intro to Machine Learning

Lecture 05 – Part 02

How Conditional Independence Helps

Recall: The Bayes Classifier

- ▶ To use the Bayes classifier, we must estimate

$$P(\vec{X} = \vec{x} \mid Y = y_i)$$

for each class y_i , where $\vec{X} = (X_1, X_2, \dots, X_d)$.

- ▶ Written differently, we need to estimate:

$$P(X_1 = x_1, \dots, X_d = x_d \mid Y = y_i)$$

Recall: Histogram Estimators

- ▶ When X_1, \dots, X_d are continuous, we can use **histogram estimators**.
- ▶ **Curse of Dimensionality**: if we discretize each dimension into 10 bins, there are 10^d bins.

Conditional Independence to the Rescue

- ▶ Now suppose X_1, \dots, X_d are mutually conditionally independent given Y . Then:

$$P(X_1 = x_1, \dots, X_d = x_d | Y = y_i) = P(X_1 = x_1 | Y = y_i)P(X_2 = x_2 | Y = y_i) \cdots P(X_d = x_d | Y = y_i)$$

- ▶ Instead of estimating $P(X_1, \dots, X_d | Y)$, estimate $P(X_1 | Y), \dots, P(X_d | Y)$ separately.

Breaking the Curse

- ▶ Lets use histogram estimators.
- ▶ If we discretize each dimension into 10 bins, we need:
 - ▶ 10 bins to estimate $P(X_1|Y)$
 - ▶ 10 bins to estimate $P(X_2|Y)$
 - ▶ ...
 - ▶ 10 bins to estimate $P(X_d|Y)$
- ▶ We therefore need $10d$ bins in total.

Breaking the Curse

- ▶ Conditional independence **drastically reduced** the number of bins needed to cover the input space.
- ▶ From $\Theta(10^d)$ to $\Theta(d)$.

Idea

- ▶ Bayes Classifier needs a lot of data when d is big.
- ▶ But if the features are conditionally independent given the label, we don't need so much data.
- ▶ So let's just **assume** conditional independence.
- ▶ The result: the **Naïve Bayes Classifier**.

Naïve Bayes: The Algorithm

- ▶ **Assume** that X_1, \dots, X_d are mutually independent given the class label.
- ▶ Estimate $P(X_1 = x_1 | Y = y_i), \dots, P(X_d = x_d | Y = y_i)$ however you'd like: histograms, fitting univariate Gaussians, etc.
- ▶ Pick the y_i which maximizes

$$P(X_1 = x_1 | Y = y_i) \cdots P(X_d = x_d | Y = y_i) P(Y = y_i)$$

But wait...

- ▶ ...are we allowed to just assume conditional independence?

But wait...

- ▶ ...are we allowed to just assume conditional independence?
- ▶ **Answer:** who's going to stop us?

But wait...

- ▶ ...isn't the assumption wrong?

But wait...

- ▶ ...isn't the assumption wrong?
- ▶ **Answer:** yeah, usually.

So does it even work?

So does it even work?

- ▶ **Answer:** Yes, surprisingly well.

“All models are wrong, but some are useful.”

- George Box, statistician

Estimating Probabilities

- ▶ You can estimate $P(X_i|Y)$ however makes sense.
- ▶ Often, people assume: **Gaussian Naïve Bayes**.

Example: NBA

- ▶ **Given:** player with height = 75 in, weight = 210 lbs.
- ▶ **Predict:** whether they are a forward or a guard.
- ▶ Let's use Gaussian Naïve Bayes.

Example: NBA

- We need to estimate:

$$P(X_1 = 75 \mid Y = \text{forward})$$

$$P(X_1 = 75 \mid Y = \text{guard})$$

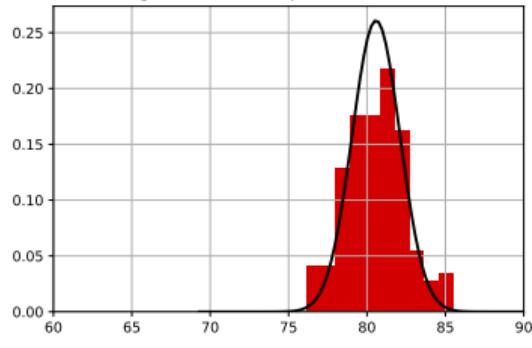
$$P(X_2 = 210 \mid Y = \text{forward})$$

$$P(X_2 = 210 \mid Y = \text{guard})$$

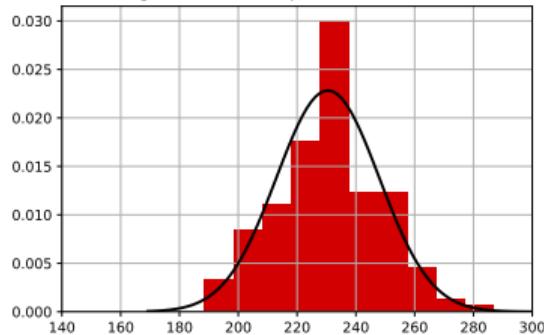
Example: NBA

- ▶ We'll fit 1-d Gaussians to:
 - ▶ heights of forwards.
 - ▶ heights of guards.
 - ▶ weights of forwards.
 - ▶ weights of guards.

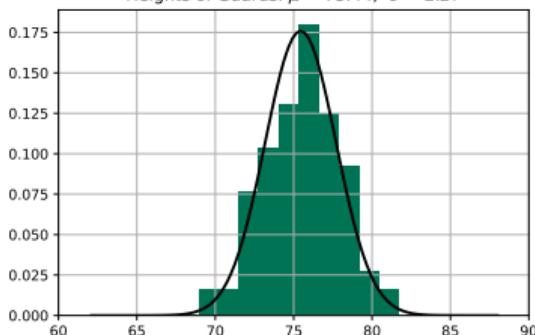
Heights of Forwards: $\mu = 80.58$, $\sigma = 1.53$



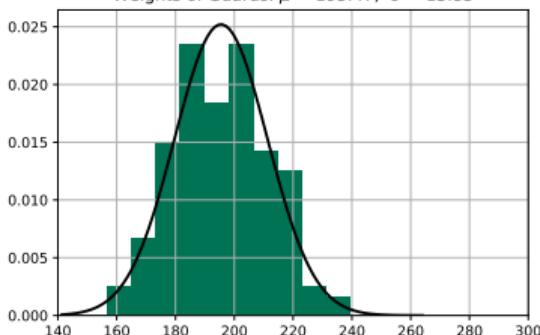
Weights of Forwards: $\mu = 230.46$, $\sigma = 17.48$



Heights of Guards: $\mu = 75.44$, $\sigma = 2.27$



Weights of Guards: $\mu = 195.47$, $\sigma = 15.83$



Example: NBA

$$\begin{aligned} & P(X_1 = 75 \mid Y = \text{forward}) \cdot P(X_2 = 210 \mid Y = \text{forward}) \cdot P(Y = \text{forward}) \\ &= \mathcal{N}(75; 80.58, 1.53^2) \cdot \mathcal{N}(210; 230.46, 17.48^2) \cdot \frac{156}{300} \\ &\approx 6.73 \times 10^{-6} \end{aligned}$$

$$\begin{aligned} & P(X_1 = 75 \mid Y = \text{guard}) \cdot P(X_2 = 210 \mid Y = \text{guard}) \cdot P(Y = \text{guard}) \\ &= \mathcal{N}(75; 75.44, 2.27^2) \cdot \mathcal{N}(210; 195.47, 15.83^2) \cdot \frac{144}{300} \\ &\approx 5.88 \times 10^{-5} \end{aligned}$$

Example: NBA

- ▶ About 85% accurate on test set.
- ▶ But heights and weights are definitely not conditionally independent given position.

Gaussian Naïve Bayes

- ▶ $P(X_1 | Y) \cdots P(X_d | Y)$ is a product of 1-d Gaussians with different means, variances.
- ▶ Remember: result is a d -dimensional Gaussian with diagonal covariance matrix:

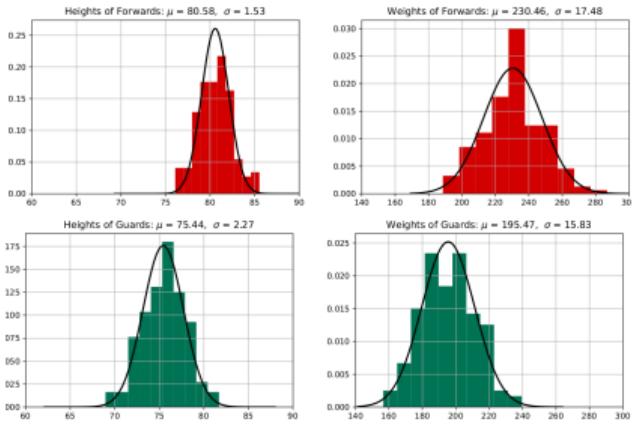
$$C = \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \sigma_d^2 \end{pmatrix}$$

Gaussian Naïve Bayes

- ▶ But in GNB, each class has own diagonal covariance matrix.
- ▶ Therefore: Gaussian Naïve Bayes is **equivalent** to QDA with diagonal covariances.

Up next...

...predicting who survives on the Titanic.



CSE 151A

Intro to Machine Learning

Lecture 05 – Part 03

The Titanic

The Titanic Dataset

PassengerID	Survived	Pclass	Sex	Age	Fare	Embarked	FavColor
0	0	3	female	23.0	7.9250	S	yellow
1	0	1	male	47.0	52.0000	S	purple
2	0	3	male	36.0	7.4958	S	green
3	0	3	male	31.0	7.7500	Q	purple
4	0	3	male	19.0	7.8958	S	purple
...

Goal: predict survival given Age, Sex, Pclass.

Let's use Naïve Bayes

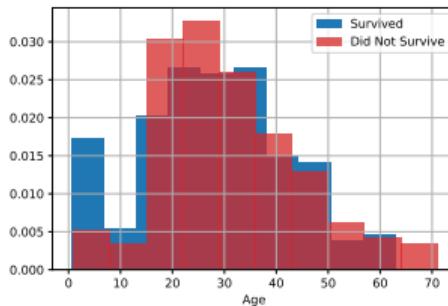
- ▶ We'll pick y_i so as to maximize

$$P(\text{Age} = x_1 | Y = y_i) \cdot P(\text{Sex} = x_2 | Y = y_i) \cdot P(\text{Pclass} = x_3 | Y = y_i) \cdot P(Y = y_i)$$

- ▶ We must choose how to estimate probabilities.
Gaussians?

Estimating Probabilities

- ▶ How do we estimate $P(\text{Age} = x_1 \mid Y = y_i)$?
- ▶ Age is a continuous variable.
- ▶ Looks kind of bell-shaped, we'll fit Gaussians.



Estimating Probabilities

- ▶ How do we estimate $P(\text{Sex} = x_1 \mid Y = y_i)$?
- ▶ Sex is a **categorical** variable: either male or female.
- ▶ Fitting Gaussian makes no sense.
- ▶ But estimating these probabilities is easy.

Estimating Probabilities

$$P(\text{Sex} = \text{male} \mid \text{Survived}) \approx \frac{\# \text{ of survived and male}}{\# \text{ of survived}} \\ = .4$$

$$P(\text{Sex} = \text{male} \mid \text{Did Not Survive}) \approx \frac{\# \text{ of died and male}}{\# \text{ of died}} \\ = .87$$

Estimating Probabilities

- ▶ Pclass, too, is categorical. Estimate in same way.
- ▶ You can estimate $P(X_i | Y)$ however makes sense.
- ▶ **Can use different ways for different features.**
- ▶ Gaussian for age, simple ratio of counts for class, sex.

Example: The Titanic

- ▶ Using just age, sex, ticket class, Naïve Bayes is 70% accurate on test set.
- ▶ Not bad. Not great.
- ▶ To do better, add more features.

In High Dimensions

- ▶ Naïve Bayes excels in high dimensions.
- ▶ Example: document classification.
 - ▶ Document represented by a “bag of words”.
 - ▶ Pick a large number of words; say, 20,000.
 - ▶ Make a d -dimensional vector with i th entry counting number of occurrences of i th word.

Practical Issues

- ▶ We are multiplying lots of small probabilities:

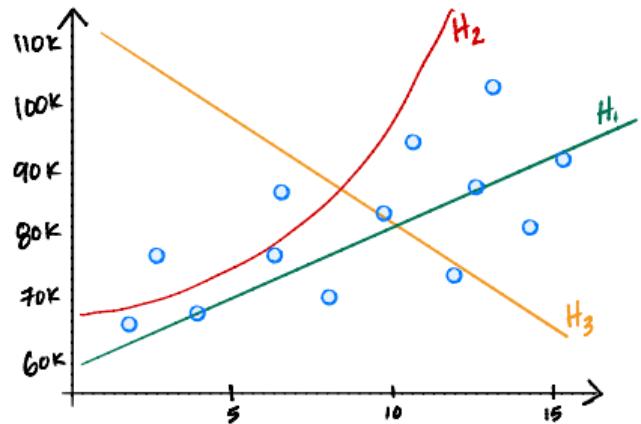
$$P(X_1 | Y) \cdots P(X_d | Y)$$

- ▶ Potential for **underflow**.

Practical Issues

- ▶ “Trick”: work with log-probabilities instead.
- ▶ Pick the y_i which maximizes

$$\begin{aligned} & \log [P(X_1 = x_1 | Y = y_i) \cdots P(X_d = x_d | Y = y_i) P(Y = y_i)] \\ &= \log P(X_1 = x_1 | Y = y_i) + \dots + \log P(X_d = x_d | Y = y_i) + \log P(Y = y_i) \\ &= \left(\sum_{j=1}^d \log P(X_j = x_j | Y = y_i) \right) + \log P(Y = y_i) \end{aligned}$$



CSE 151A

Intro to Machine Learning

Lecture 06 – Part 01

Regression and Loss Functions

What influences salary?

- ▶ **Numerical**: age, height, years of experience
- ▶ **Categorical**: college, city, gender
- ▶ **Boolean**: knows Python?, had internship?

Regression

- ▶ **Given:** someone's age, experience, city, etc.
- ▶ **Predict:** their salary.
- ▶ Since the output space is \mathbb{R} , this is a **regression** problem.

Today

- ▶ **Goal:** Predict salary from years of experience.

Prediction Rules

- ▶ Salary is a function of experience.
- ▶ I.e., there is a function H so that:
 $\text{salary} \approx H(\text{years of experience})$
- ▶ H is a **hypothesis function** or **prediction rule**.
- ▶ **Our goal:** find a good prediction rule, H .

Example Prediction Rules

$$H_1(\text{years of experience}) = \$50,000 + \$2,000 \times (\text{years of experience})$$

$$H_2(\text{years of experience}) = \$60,000 \times 1.05^{(\text{years of experience})}$$

$$H_3(\text{years of experience}) = \$100,000 - \$5,000 \times (\text{years of experience})$$

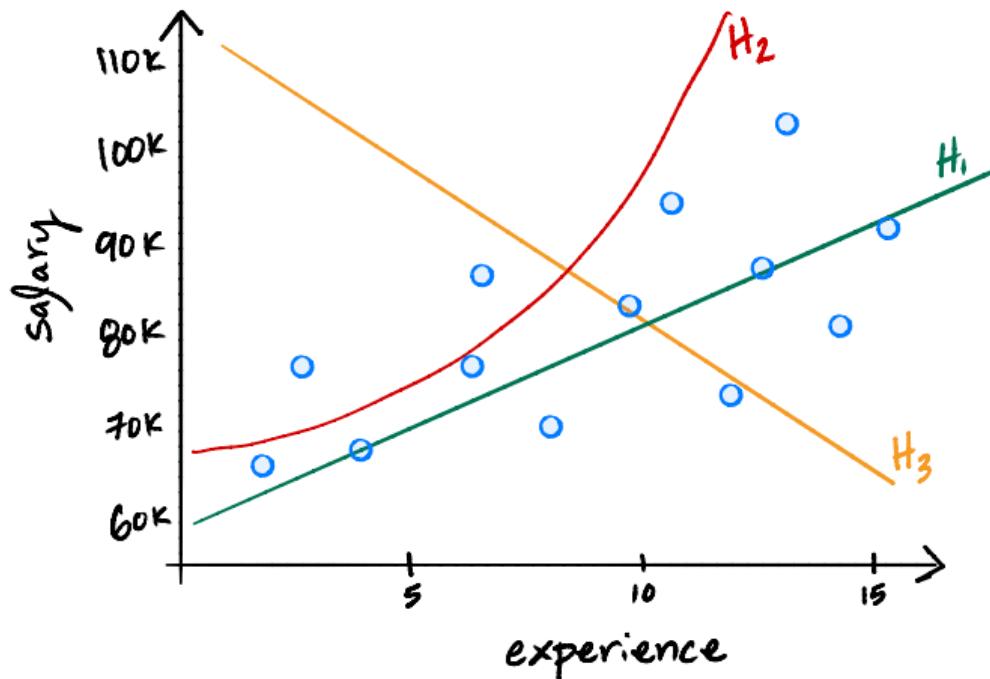
Comparing Predictions

- ▶ How do we know which is best: H_1, H_2, H_3 ?
- ▶ We gather data from n people. Let x_i be experience, y_i be salary:

$$\begin{array}{ll} (\text{Experience}_1, \text{Salary}_1) & (x_1, y_1) \\ (\text{Experience}_2, \text{Salary}_2) & (x_2, y_2) \\ \dots & \dots \\ (\text{Experience}_n, \text{Salary}_n) & (x_n, y_n) \end{array} \rightarrow$$

- ▶ See which rule works better on data.

Example



Quantifying the Error

- ▶ Our prediction for person i 's salary is $H(x_i)$
- ▶ The **absolute error** in this prediction:

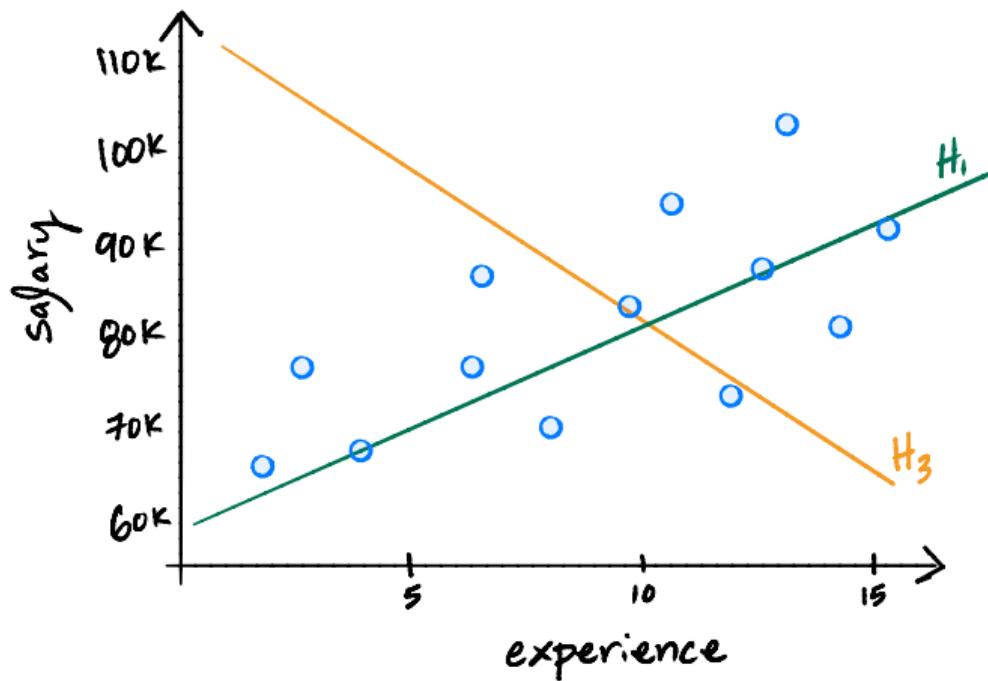
$$|H(x_i) - y_i|$$

- ▶ The **mean absolute error** of H :

$$R_{\text{abs}}(H) = \frac{1}{n} \sum_{i=1}^n |H(x_i) - y_i|$$

- ▶ Smaller the mean absolute error, the **better** the prediction rule.

Mean Absolute Error



Finding the best prediction rule

- ▶ **Goal:** out of all functions $\mathbb{R} \rightarrow \mathbb{R}$, find the function H^* with the smallest mean absolute error.
- ▶ That is, find:

$$H^* = \arg \min_H \frac{1}{n} \sum_{i=1}^n |H(x_i) - y_i|$$

Finding the best prediction rule

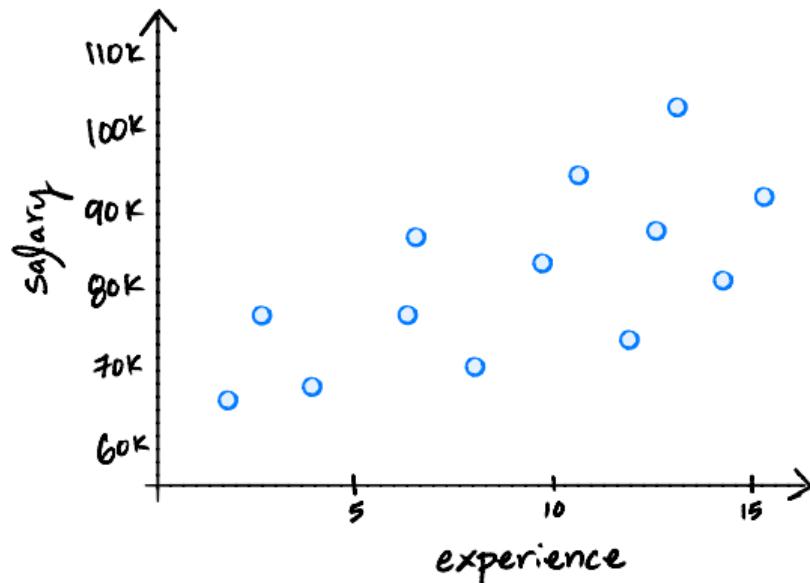
- ▶ **Goal:** out of all functions $\mathbb{R} \rightarrow \mathbb{R}$, find the function H^* with the smallest mean absolute error.
- ▶ That is, find:

$$H^* = \arg \min_H \frac{1}{n} \sum_{i=1}^n |H(x_i) - y_i|$$

- ▶ **There are two problems with this.**

Question

Is there a prediction rule H which has **zero** mean absolute error?



Problem #1

- ▶ We can make mean absolute error very small, even zero!
- ▶ But the function will be weird.
- ▶ This is called **overfitting**.
- ▶ Remember our real goal: make good predictions on data **we haven't seen**.

Solution

- ▶ Don't allow H to be just any function.
- ▶ Require that it has a certain form.
- ▶ Examples:
 - ▶ Linear: $H(x) = w_1x + w_0$
 - ▶ Quadratic: $H(x) = w_2x^2 + w_1x + w_0$
 - ▶ Exponential: $H(x) = w_0e^{w_1x}$
 - ▶ Constant: $H(x) = w_0$

Finding the best linear rule

- ▶ **Goal:** out of all **linear** functions $\mathbb{R} \rightarrow \mathbb{R}$, find the function H^* with the smallest mean absolute error.
- ▶ That is, find:

$$H^* = \arg \min_{\text{linear } H} \frac{1}{n} \sum_{i=1}^n |H(x_i) - y_i|$$

Finding the best linear rule

- ▶ **Goal:** out of all **linear** functions $\mathbb{R} \rightarrow \mathbb{R}$, find the function H^* with the smallest mean absolute error.
- ▶ That is, find:

$$H^* = \arg \min_{\text{linear } H} \frac{1}{n} \sum_{i=1}^n |H(x_i) - y_i|$$

- ▶ **There is still a problem with this.**

Problem #2

- ▶ It is hard to minimize the mean absolute error:¹

$$\frac{1}{n} \sum_{i=1}^n |H(x_i) - y_i|$$

- ▶ Not differentiable!
- ▶ What can we do?

¹Though it can be done with linear programming.

Quantifying the Error

- ▶ Instead of absolute error, use the **squared error** of a prediction:

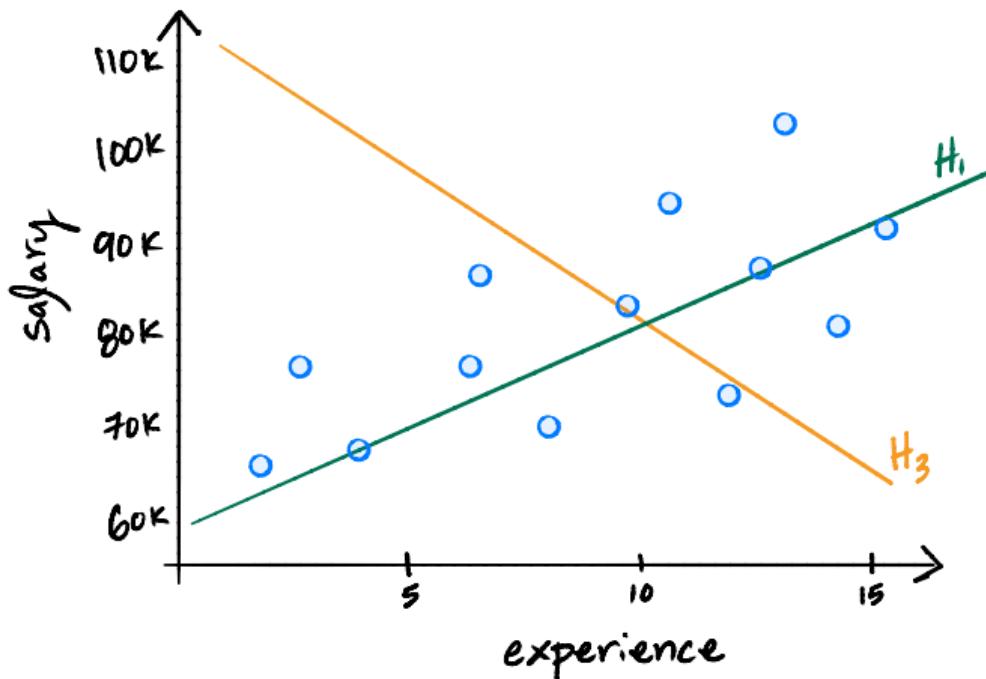
$$(H(x_i) - y_i)^2$$

- ▶ The **mean squared error** (MSE) of H :

$$R_{\text{sq}}(H) = \frac{1}{n} \sum_{i=1}^n (H(x_i) - y_i)^2$$

- ▶ **Is differentiable!**

Mean Squared Error



Our Goal

- ▶ Out of all **linear** functions $\mathbb{R} \rightarrow \mathbb{R}$, find the function H^* with the smallest **mean squared error**.
- ▶ That is, find:

$$H^* = \underset{\text{linear } H}{\arg \min} \frac{1}{n} \sum_{i=1}^n (H(x_i) - y_i)^2$$

- ▶ This problem is called **least squares regression**.

By the way...

- ▶ Prediction functions will play a large role.
- ▶ **Absolute error** and **squared error** are **loss functions**.

$$|H(x) - y_i| \quad (H(x_i) - y_i)^2$$

By the way...

- ▶ The average loss on the training data is called the **empirical risk**
- ▶ Example: the mean squared error is the empirical risk of the square loss:

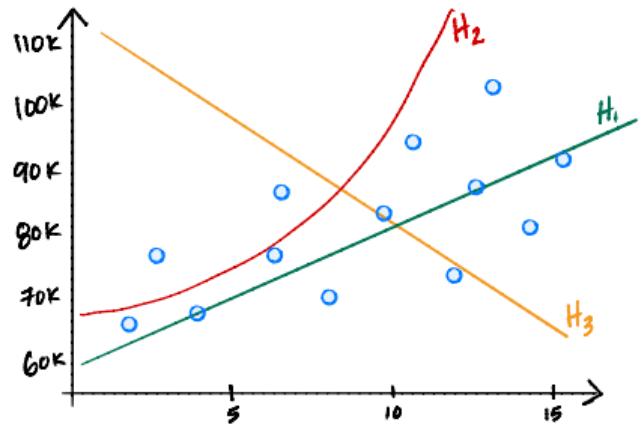
$$R_{\text{sq}}(H) = \frac{1}{n} \sum_{i=1}^n (H(x_i) - y_i)^2$$

By the way...

- ▶ A major paradigm in ML: find the prediction function which minimizes risk.
- ▶ Called **Empirical Risk Minimization**, or **ERM**.

Up next...

...minimizing the MSE.



CSE 151A

Intro to Machine Learning

Lecture 06 – Part 02

Minimizing the MSE

Our Goal

- ▶ Out of all **linear** functions $\mathbb{R} \rightarrow \mathbb{R}$, find the function H^* with the smallest **mean squared error**.
- ▶ That is, find:

$$H^* = \underset{\text{linear } H}{\arg \min} \frac{1}{n} \sum_{i=1}^n (H(x_i) - y_i)^2$$

- ▶ This problem is called **least squares regression**.

Minimizing the MSE

- ▶ The MSE is a function of a function:

$$R_{\text{sq}}(H) = \frac{1}{n} \sum_{i=1}^n (H(x_i) - y_i)^2$$

- ▶ But since H is linear, $H(x) = w_1 x + w_0$.

$$R_{\text{sq}}(w_1, w_0) = \frac{1}{n} \sum_{i=1}^n ((w_1 x + w_0) - y_i)^2$$

- ▶ Now it's a function of w_1, w_0 .

Updated Goal

- ▶ Find slope w_1 and intercept w_0 which minimize the MSE, $R_{\text{sq}}(w_1, w_0)$:

$$R_{\text{sq}}(w_1, w_0) = \frac{1}{n} \sum_{i=1}^n ((w_1 x + w_0) - y_i)^2$$

- ▶ Strategy: multivariate calculus.

Recall: the gradient

- ▶ If $f(x, y)$ is a function of two variables, the **gradient** of f at the point (x_0, y_0) is a **vector** of **partial derivatives**:

$$\nabla f(x_0, y_0) = \begin{pmatrix} \frac{\partial f}{\partial x}(x_0) \\ \frac{\partial f}{\partial y}(y_0) \end{pmatrix}$$

- ▶ **Key Fact:** gradient is zero at critical points.

Strategy

To minimize $R(w_1, w_0)$: compute the gradient, set equal to zero, solve.

$$R_{\text{sq}}(w_1, w_0) = \frac{1}{n} \sum_{i=1}^n ((w_1 x + w_0) - y_i)^2$$

$$\frac{\partial R_{\text{sq}}}{\partial w_1} =$$

$$R_{\text{sq}}(w_1, w_0) = \frac{1}{n} \sum_{i=1}^n ((w_1 x + w_0) - y_i)^2$$

$$\frac{\partial R_{\text{sq}}}{\partial w_0} =$$

Strategy

$$0 = \frac{2}{n} \sum_{i=1}^n ((w_1 x_i + w_0) - y_i) x_i \quad 0 = \frac{2}{n} \sum_{i=1}^n ((w_1 x_i + w_0) - y_i)$$

1. Solve for w_0 in second equation.
2. Plug solution for w_0 into first equation, solve for w_1 .

Solve for w_0

$$0 = \frac{2}{n} \sum_{i=1}^n ((w_1 x_i + w_0) - y_i)$$

Solve for w_0

$$0 = \frac{2}{n} \sum_{i=1}^n ((w_1 x_i + w_0) - y_i)$$

Key Fact

- ▶ Define

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

- ▶ Then

$$\sum_{i=1}^n (x_i - \bar{x}) = 0 \quad \sum_{i=1}^n (y_i - \bar{y}) = 0$$

Solve for w_1

$$0 = \frac{2}{n} \sum_{i=1}^n ((w_1 x_i + w_0) - y_i) x_i \quad w_0 = \bar{y} - w_1 \bar{x}$$

Solve for w_1

$$0 = \frac{2}{n} \sum_{i=1}^n ((w_1 x_i + w_0) - y_i) x_i \quad w_0 = \bar{y} - w_1 \bar{x}$$

Least Squares Solutions

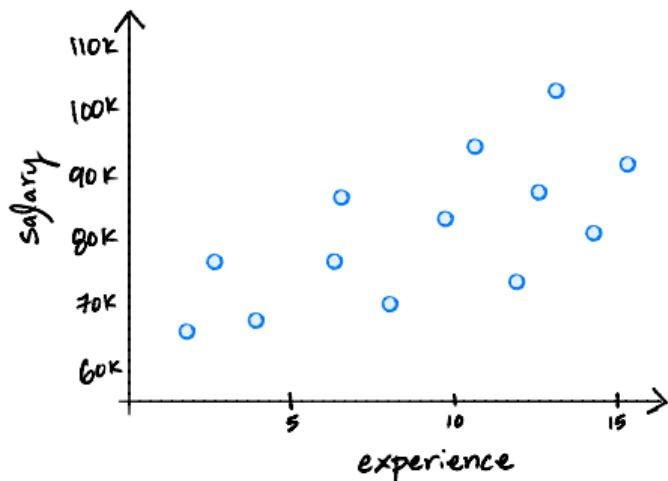
- The **least squares solutions** for the slope w_1 and intercept w_0 are:

$$w_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad w_0 = \bar{y} - w_1 \bar{x}$$

$$\text{where } \bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

Interpretation of Slope

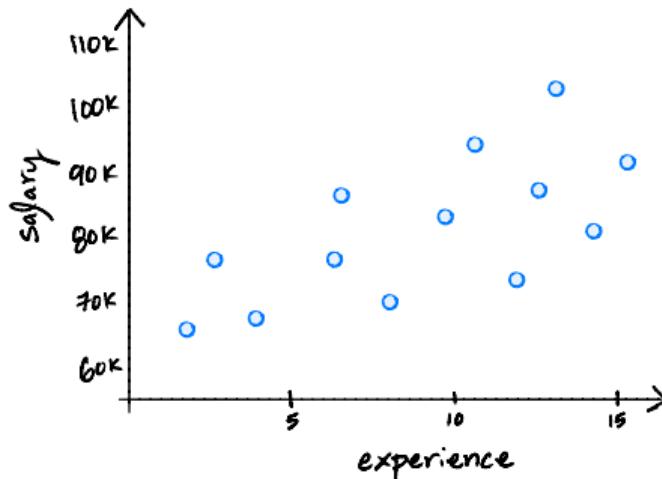
$$w_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$



- ▶ What is the sign of $(x_i - \bar{x})(y_i - \bar{y})$ when:
 - ▶ $x_i > \bar{x}$ and $y_i > \bar{y}$?

Interpretation of Slope

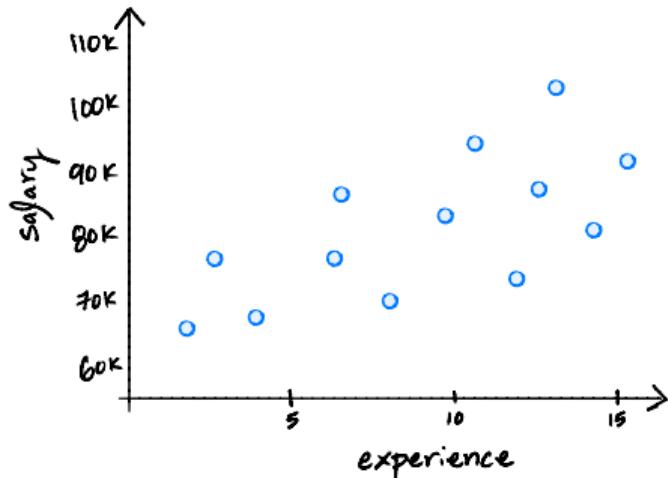
$$w_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$



- ▶ What is the sign of $(x_i - \bar{x})(y_i - \bar{y})$ when:
 - ▶ $x_i < \bar{x}$ and $y_i < \bar{y}$?

Interpretation of Slope

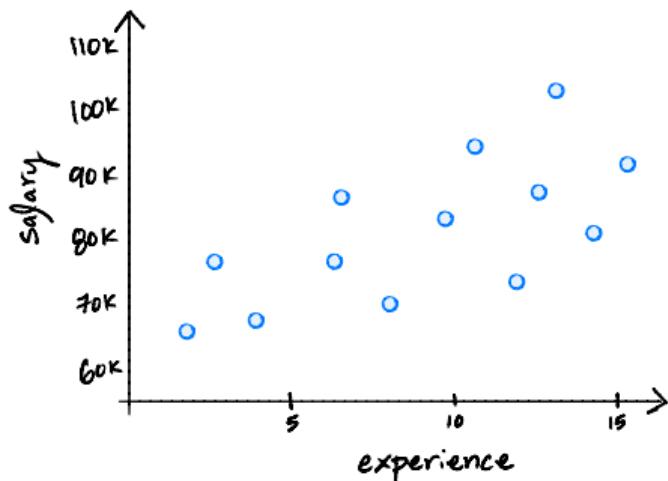
$$w_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$



- ▶ What is the sign of $(x_i - \bar{x})(y_i - \bar{y})$ when:
 - ▶ $x_i > \bar{x}$ and $y_i < \bar{y}$?

Interpretation of Slope

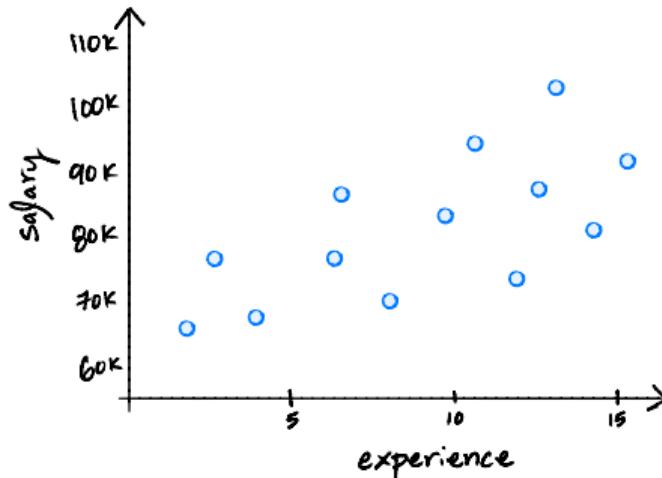
$$w_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$



- ▶ What is the sign of $(x_i - \bar{x})(y_i - \bar{y})$ when:
 - ▶ $x_i < \bar{x}$ and $y_i > \bar{y}$?

Interpretation of Intercept

$$w_0 = \bar{y} - w_1 \bar{x}$$

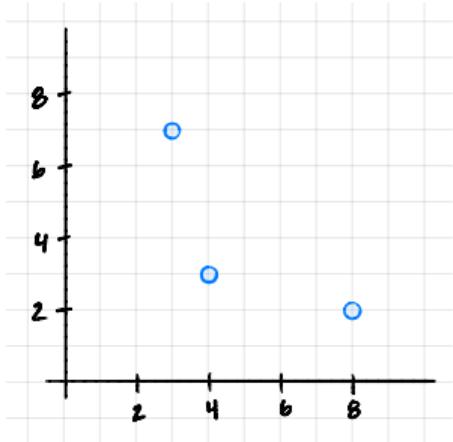


- ▶ What is $H(\bar{x})$?

Question

We fit a linear prediction rule for salary given years of experience. Then everyone gets a \$5,000 raise. What happens to slope/intercept?

Example



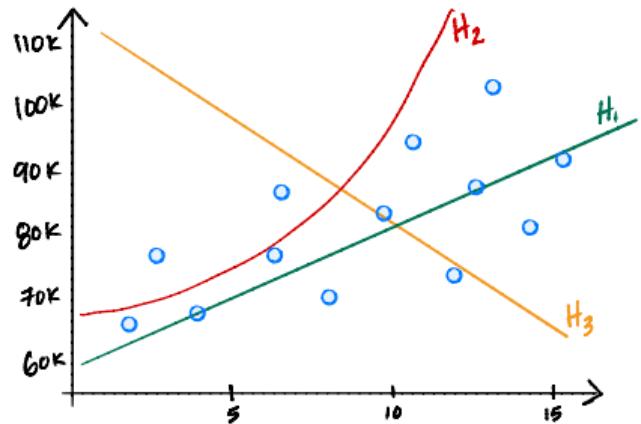
$$\bar{x} =$$

$$\bar{y} =$$

$$w_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} =$$

$$w_0 = \bar{y} - w_1 \bar{x}$$

x_i	y_i	$(x_i - \bar{x})$	$(y_i - \bar{y})$	$(x_i - \bar{x})(y_i - \bar{y})$	$(x_i - \bar{x})^2$
3	7	-1	4	-4	1
4	3	0	-4	0	0
8	2	5	-5	-25	25



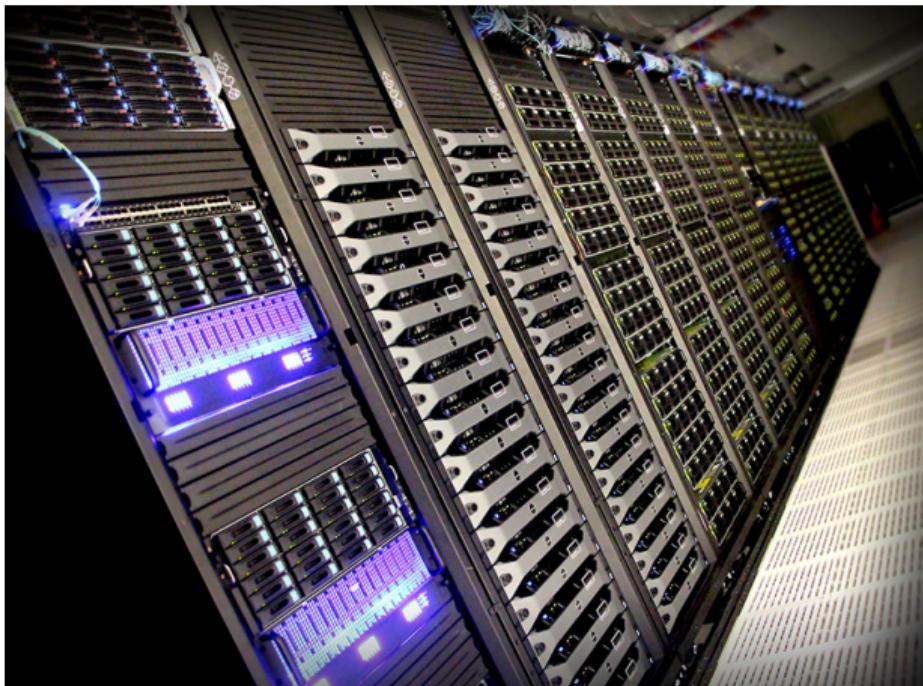
CSE 151A

Intro to Machine Learning

Lecture 06 – Part 03

Fitting Non-Linear Trends

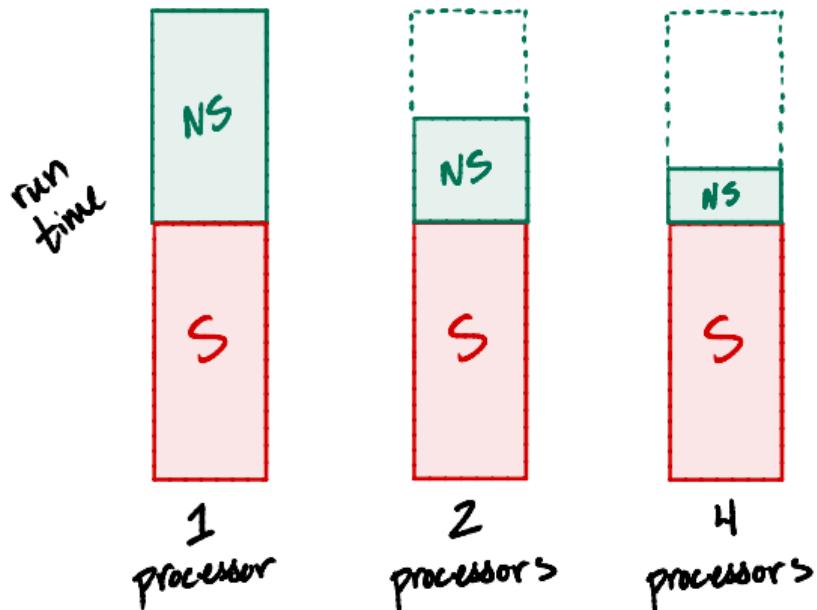
Example: Parallel Processing



Problem

- ▶ Some parts of a program are necessarily **sequential**.
- ▶ E.g., downloading the data must happen before analysis.
- ▶ More processors do not speed up **sequential** code.
- ▶ But they do speed up **non-sequential** code.

Speedup



Amdahl's Law

The time T it takes to run a program on p processors is:

$$T(p) = t_S + \frac{t_{NS}}{p}$$

where t_S and t_{NS} are the time it takes the sequential and non-sequential parts to run on one processor, respectively.

Amdahl's Law

The time T it takes to run a program on p processors is:

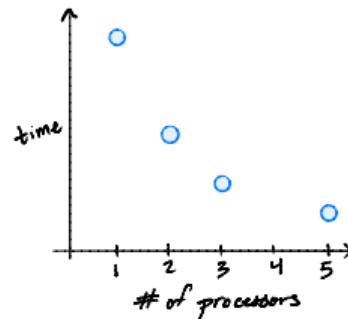
$$T(p) = t_S + \frac{t_{NS}}{p}$$

where t_S and t_{NS} are the time it takes the sequential and non-sequential parts to run on one processor, respectively.

Problem: we don't know t_S and t_{NS} .

Fitting Amdahl's Law

- ▶ **Solution:** we will learn t_S and t_{NS} from data.
- ▶ Run with varying number of processors:



- ▶ Find prediction rule $H(p) = \frac{t_{NS}}{p} + t_S$ by minimizing MSE.

General Problem

- ▶ Given data $(x_1, y_1), \dots, (x_n, y_n)$.
- ▶ Fit a **non-linear** rule $H(x) = w_1 \cdot \frac{1}{x} + w_0$ by minimizing MSE:

$$R_{\text{sq}}(H) = \frac{1}{n} \sum_{i=1}^n (H(x_i) - y_i)^2$$

Using definition of H :

Minimizing MSE

- Take derivatives, you'll find:

$$\frac{\partial R_{\text{sq}}}{\partial w_1}(w_1, w_0) = \frac{2}{n} \sum_{i=1}^n \left[\left(w_1 \cdot \frac{1}{x_i} + w_0 \right) - y_i \right] \frac{1}{x_i}$$

$$\frac{\partial R_{\text{sq}}}{\partial w_0}(w_1, w_0) = \frac{2}{n} \sum_{i=1}^n \left[\left(w_1 \cdot \frac{1}{x_i} + w_0 \right) - y_i \right]$$

Minimizing MSE

- ▶ Set to zero, solve. You'll find:

$$w_1 = \frac{\sum_{i=1}^n \left(\frac{1}{x_i} - \frac{1}{n} \sum_{i=1}^n \frac{1}{x_i} \right) (y_i - \bar{y})}{\sum_{i=1}^n \left(\frac{1}{x_i} - \frac{1}{n} \sum_{i=1}^n \frac{1}{x_i} \right)^2} \quad w_0 = \bar{y} - w_1 \cdot \frac{1}{n} \sum_{i=1}^n \frac{1}{x_i}$$

Minimizing MSE

- ▶ Set to zero, solve. You'll find:

$$w_1 = \frac{\sum_{i=1}^n \left(\frac{1}{x_i} - \frac{1}{n} \sum_{i=1}^n \frac{1}{x_i} \right) (y_i - \bar{y})}{\sum_{i=1}^n \left(\frac{1}{x_i} - \frac{1}{n} \sum_{i=1}^n \frac{1}{x_i} \right)^2} \quad w_0 = \bar{y} - w_1 \cdot \frac{1}{n} \sum_{i=1}^n \frac{1}{x_i}$$

- ▶ Define $z_i = \frac{1}{x_i}$, $\bar{z} = \frac{1}{n} \sum_{i=1}^n z_i = \frac{1}{n} \sum_{i=1}^n \frac{1}{x_i}$. Then:

$$w_1 =$$

$$w_0 =$$

Fitting Non-Linear Trends

To fit a prediction rule of the form $H(x) = w_1 \cdot \frac{1}{x} + w_0$:

1. Create a new data set $(z_1, y_1), \dots, (z_n, y_n)$, where $z_i = \frac{1}{x_i}$.

Fitting Non-Linear Trends

To fit a prediction rule of the form $H(x) = w_1 \cdot \frac{1}{x} + w_0$:

2. Fit $H(z) = w_1 z + w_0$ using familiar least squares solutions:

$$w_1 = \frac{\sum_{i=1}^n (z_i - \bar{z})(y_i - \bar{y})}{\sum_{i=1}^n (z_i - \bar{z})^2} \quad w_0 = \bar{y} - w_1 \cdot \bar{z}$$

Fitting Non-Linear Trends

To fit a prediction rule of the form $H(x) = w_1 \cdot \frac{1}{x} + w_0$:

3. Use w_1 and w_0 in original prediction rule, $H(x)$.

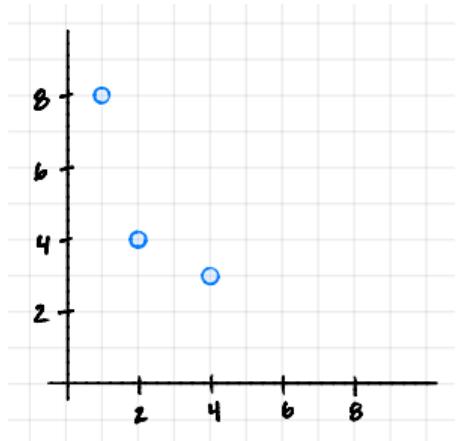
Example: Amdahl's Law

- ▶ We have timed our program:

Processors	Time (Hours)
1	8
2	4
4	3

- ▶ Fit prediction rule: $H(p) = \frac{t_{NS}}{p} + t_S$

Example: fitting $H(x) = w_1 \cdot \frac{1}{x_i} + x_0$



$$\bar{z} =$$

$$\bar{y} =$$

$$w_1 = \frac{\sum_{i=1}^n (z_i - \bar{z})(y_i - \bar{y})}{\sum_{i=1}^n (z_i - \bar{z})^2} =$$

$$w_0 = \bar{y} - w_1 \bar{z}$$

x_i	z_i	y_i	$(z_i - \bar{z})$	$(y_i - \bar{y})$	$(z_i - \bar{z})(y_i - \bar{y})$	$(z_i - \bar{z})^2$
3	7					
4	3					
8	2					

Example: Amdahl's Law

- ▶ We found: $t_{NS} = \frac{48}{7} \approx 6.88$, $t_S = 1$
- ▶ Our prediction rule:

$$H(p) = \frac{t_{NS}}{p} + t_S$$

$$= \frac{6.88}{p} + 1$$

Fitting Non-Linear Trends

- We can fit rules like:

$$w_1x + w_0 \quad w_1 \cdot \frac{1}{x} + w_0 \quad w_1x^2 + w_0 \quad w_1e^x + w_0$$

- We can't fit rules like:

$$w_0e^{w_1x} \quad \sin(w_1x + w_0)$$

- Can fit as long as **linear** function of w_1, w_0 .

What's Left?

- ▶ How do we make predictions with lots of features?
- ▶ E.g., experience, age, GPA, number of internships, etc.

Basic Linear Algebra Review

Matrices

An $m \times n$ **matrix** is a table of numbers with m rows, n columns:

- ▶ Example: 2×3 matrix:

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{pmatrix}$$

- ▶ Example: 3×3 “square” matrix:

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix}$$

Matrix Notation

- ▶ We use upper-case letters for matrices.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

- ▶ Sometimes use subscripts to denote particular elements: $A_{13} = 3, A_{21} = 4$
- ▶ A^T denotes the transpose of A:

$$A^T = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

Matrix Addition and Scalar Multiplication

- ▶ We can add two matrices only if they are the same size.
- ▶ Addition occurs elementwise:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} + \begin{pmatrix} 7 & 8 & 9 \\ -1 & -2 & -3 \end{pmatrix} = \begin{pmatrix} 8 & 10 & 12 \\ 3 & 3 & 3 \end{pmatrix}$$

- ▶ Scalar multiplication occurs elementwise, too:

$$2 \cdot \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} = \begin{pmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \end{pmatrix}$$

Matrix-Matrix Multiplication

- ▶ We can multiply two matrices A and B only if # cols in A is equal to # rows in B
- ▶ If $A = m \times n$ and $B = n \times p$, the result is $m \times p$.
 - ▶ This is **very useful**. Remember it!
- ▶ The low-level definition. the ij entry of the product is:

$$(AB)_{ij} = \sum_{k=1}^n A_{ik}B_{kj}$$

Matrix-Matrix Multiplication Example

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 3 & 4 & 5 \end{pmatrix} \quad B = \begin{pmatrix} 3 & 6 \\ 1 & 3 \\ 4 & 8 \end{pmatrix}$$

- ▶ What is the size of AB ?
- ▶ What is $(AB)_{12}$?

Matrix-Matrix Multiplication Properties

- ▶ Distributive: $A(B + C) = AB + AC$
- ▶ Associative: $(AB)C = A(BC)$
- ▶ **Not commutative in general:** $AB \neq BA$

Identity Matrices

- ▶ The $n \times n$ **identity matrix** I has ones along the diagonal:

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

- ▶ If A is $n \times m$, then $IA = A$.
- ▶ If B is $m \times n$, then $BI = B$.

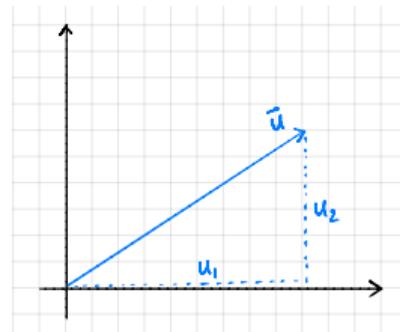
Vectors

- ▶ An d -vector is an $d \times 1$ matrix.
- ▶ Often use arrow, lower-case letters to denote: \vec{x} .
- ▶ Often write $\vec{x} \in \mathbb{R}^d$ to say \vec{x} is a d vector.
- ▶ Example. A 4-vector:

$$\begin{pmatrix} 2 \\ 1 \\ 5 \\ -3 \end{pmatrix}$$

Geometric Meaning of Vectors

- ▶ A vector $\vec{u} = (u_1, \dots, u_d)^T$ is an arrow to the point (u_1, \dots, u_d) :



- ▶ The length, or **norm**, of \vec{u} is
$$\|\vec{u}\| = \sqrt{u_1^2 + u_2^2 + \dots + u_d^2}.$$
- ▶ A **unit vector** is a vector of norm 1.

Dot Products

- ▶ The **dot product** of two d -vectors \vec{u} and \vec{v} is:

$$\vec{u} \cdot \vec{v} = \vec{u}^T \vec{v}$$

- ▶ Using low-level matrix multiplication definition:

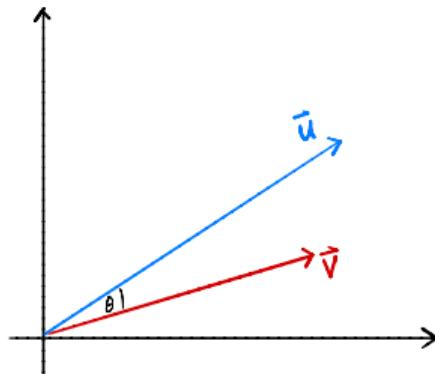
$$\begin{aligned}\vec{u} \cdot \vec{v} &= \sum_{i=1}^n u_i v_i \\ &= u_1 v_1 + u_2 v_2 + \dots + u_n v_n\end{aligned}$$

Dot Product Example

$$\vec{u} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad \vec{v} = \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix} \quad \vec{u} \cdot \vec{v} =$$

Geometric Interpretation of Dot Product

► $\vec{u} \cdot \vec{v} = \|\vec{u}\| \|\vec{v}\| \cos \theta.$



Which of these is another expression for the norm of \vec{u} ?

- a) $\vec{u} \cdot \vec{u}$
- b) $\sqrt{\vec{u}^2}$
- c) $\sqrt{\vec{u} \cdot \vec{u}}$
- d) \vec{u}^2

Properties of the Dot Product

- ▶ Commutative: $\vec{u} \cdot \vec{v} = \vec{v} \cdot \vec{u}$
- ▶ Distributive: $\vec{u} \cdot (\vec{v} + \vec{w}) = \vec{u} \cdot \vec{v} + \vec{u} \cdot \vec{w}$
- ▶ Linear: $\vec{u} \cdot (\alpha\vec{v} + \beta\vec{w}) = \alpha\vec{u} \cdot \vec{v} + \beta\vec{u} \cdot \vec{w}$

Matrix-Vector Multiplication

- ▶ Special case of matrix-matrix multiplication.
- ▶ Result is always a vector with same number of rows as the matrix.
- ▶ One view: a “mixture” of the columns.

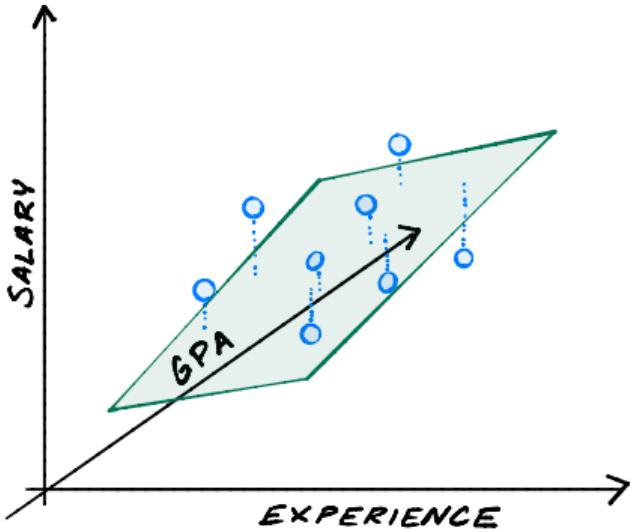
$$\begin{pmatrix} 1 & 2 & 1 \\ 3 & 4 & 5 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = a_1 \begin{pmatrix} 1 \\ 3 \end{pmatrix} + a_2 \begin{pmatrix} 2 \\ 4 \end{pmatrix} + a_3 \begin{pmatrix} 1 \\ 5 \end{pmatrix}$$

Matrices and Functions

- ▶ Matrix-vector multiplication takes in a vector, outputs a vector.
- ▶ An $m \times n$ matrix is an encoding of a function mapping \mathbb{R}^m to \mathbb{R}^n .
- ▶ Matrix multiplication evaluates that function.

Today

- ▶ How do we predict salary given **multiple** features?
 - ▶ years of experience, number of internships, GPA, etc.
- ▶ We'll need to use some linear algebra...



CSE 151A
Intro to Machine Learning

Lecture 07 – Part 01
Setup

Today

- ▶ How do we predict salary given **multiple** features?
 - ▶ years of experience, number of internships, GPA, etc.

Using Multiple Features

- ▶ We believe salary is a function of experience *and* GPA.

- ▶ I.e., there is a function H so that:

$$\text{salary} \approx H(\text{years of experience}, \text{GPA})$$

- ▶ Recall: H is a **prediction rule**.
- ▶ **Our goal:** find a good prediction rule, H .

Example Prediction Rules

$$H_1(\text{experience}, \text{GPA}) = \$40,000 \times \frac{\text{GPA}}{4.0} + \$2,000 \times (\text{experience})$$

$$H_2(\text{experience}, \text{GPA}) = \$60,000 \times 1.05^{(\text{experience}+\text{GPA})}$$

$$H_3(\text{experience}, \text{GPA}) = \sin(\text{GPA}) + \cos(\text{experience})$$

Linear Prediction Rule

- ▶ We'll restrict ourselves to **linear** prediction rules:

$$H(\text{experience}, \text{GPA}) = w_0 + w_1 \times (\text{experience}) + w_2 \times (\text{GPA})$$

- ▶ Can add more features, too¹:

$$\begin{aligned} H(\text{experience}, \text{GPA}, \# \text{ internships}) &= \\ w_0 + w_1 \times (\text{experience}) + w_2 \times (\text{GPA}) + w_3 \times (\# \text{ of internships}) \end{aligned}$$

- ▶ Interpretation of w_i : the **weight** of feature x_i .

¹In practice, might use tens, hundreds, even thousands of features.

Feature Vectors

- ▶ In general, if x_1, \dots, x_d are d features:

$$H(x_1, \dots, x_d) = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_d x_d$$

- ▶ Nicer to pack into a **feature vector** and **parameter vector**:

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \quad \vec{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \\ w_d \end{pmatrix}$$

Augmented Feature Vectors

- The **augmented feature vector** $\text{Aug}(\vec{x})$ is the vector obtained by adding a 1 to the front of \vec{x} :

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \quad \text{Aug}(\vec{x}) = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \quad \vec{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{pmatrix}$$

- Then:

$$\begin{aligned} H(x_1, \dots, x_d) &= w_0 + w_1 x_1 + w_2 x_2 + \dots + w_d x_d \\ &= \text{Aug}(\vec{x}) \cdot \vec{w} \end{aligned}$$

Example

- ▶ Recall the prediction rule:

$$H_1(\text{experience}, \text{GPA}) = \$40,000 \times \frac{\text{GPA}}{4.0} + \$2,000 \times (\text{experience})$$

- ▶ This is linear. If x_1 is experience, x_2 is GPA, then:

$$\vec{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 2,000 \\ 10,000 \end{pmatrix}$$

- ▶ Prediction for 2 years experience, 3.0 GPA:

$$\text{Aug}(\vec{x}) = \begin{pmatrix} \quad \end{pmatrix} \quad H(\vec{x}) = \text{Aug}(\vec{x}) \cdot \vec{w} =$$

The Data

- ▶ For each person, collect 3 features, plus salary:

Person #	Experience	GPA	# Internships	Salary
1	3	3.7	1	85,000
2	6	3.3	2	95,000
3	10	3.1	3	105,000

- ▶ We represent each person with a **data vector**:

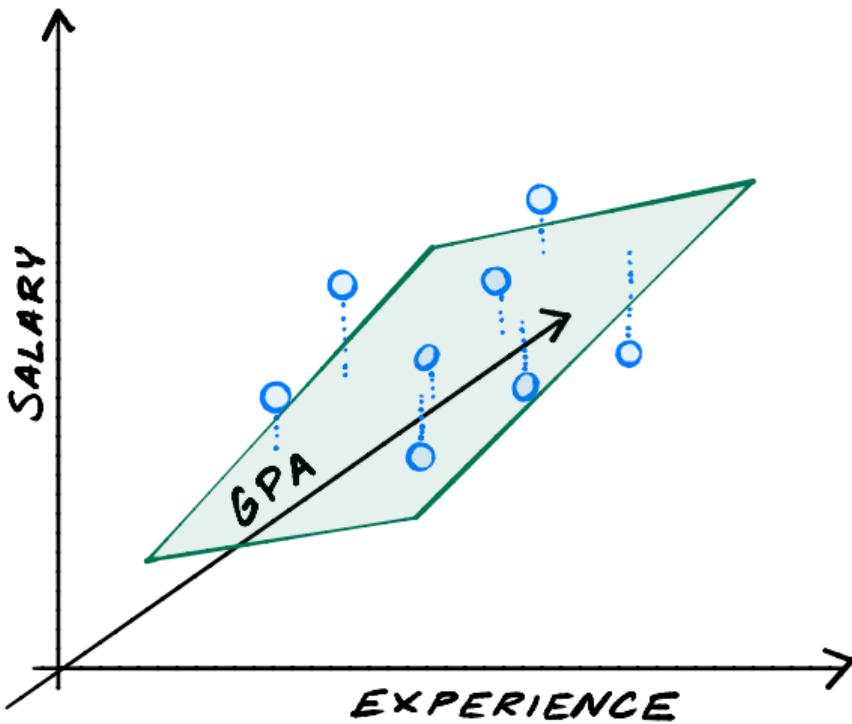
$$\vec{x}^{(1)} = \begin{pmatrix} 3 \\ 3.7 \\ 1 \end{pmatrix}, \quad \vec{x}^{(2)} = \begin{pmatrix} 6 \\ 3.3 \\ 2 \end{pmatrix}, \quad \vec{x}^{(3)} = \begin{pmatrix} 10 \\ 3.1 \\ 3 \end{pmatrix}$$

Notation

- ▶ $\vec{x}^{(i)}$ is the i th data vector.
- ▶ $x_j^{(i)}$ is the j th feature in the i th data vector.
- ▶ If there are d features:

$$\vec{x}^{(i)} = \begin{pmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_d^{(i)} \end{pmatrix}$$

Geometric Interpretation



The General Problem

- ▶ Have n **training examples**: $(\vec{x}^{(1)}, y_1), \dots, (\vec{x}^{(n)}, y_n)$
- ▶ We want to find a good linear prediction rule:

$$H(\vec{x}) = \vec{w} \cdot \text{Aug}(\vec{x})$$

- ▶ To do so, we'll minimize the mean squared error:

$$\begin{aligned} R_{\text{sq}}(\vec{w}) &= \frac{1}{n} \sum_{i=1}^n (H(\vec{x}^{(i)}) - y_i)^2 \\ &= \frac{1}{n} \sum_{i=1}^n ((\vec{w} \cdot \text{Aug}(\vec{x}^{(i)})) - y_i)^2 \end{aligned}$$

The Risk

- ▶ With d features, we have $d + 1$ parameters:
 w_0, w_1, \dots, w_d .
- ▶ The risk $R_{\text{sq}}(\vec{w})$ is a function from \mathbb{R}^{d+1} to \mathbb{R}^1 .
- ▶ It is a $(d + 1)$ -dimensional hypersurface.
- ▶ **No hope of visualizing it directly when $d \geq 2$.**

Rewriting the Mean Squared Error

- ▶ Let \vec{e} be such that e_i is the (signed) error on i th example:

$$e_i = (\vec{w} \cdot \text{Aug}(\vec{x}^{(i)})) - y_i$$

- ▶ Then:

$$R_{\text{sq}}(\vec{w}) = \frac{1}{n} \sum_{i=1}^n [(\vec{w} \cdot \text{Aug}(\vec{x}^{(i)})) - y_i]^2$$

$$= \frac{1}{n} \sum_{i=1}^n e_i^2$$

Rewriting the Mean Squared Error

- ▶ Let \vec{e} be such that e_i is the (signed) error on i th example:

$$e_i = (\vec{w} \cdot \text{Aug}(\vec{x}^{(i)})) - y_i$$

- ▶ Then:

$$R_{\text{sq}}(\vec{w}) = \frac{1}{n} \sum_{i=1}^n [(\vec{w} \cdot \text{Aug}(\vec{x}^{(i)})) - y_i]^2$$

$$= \frac{1}{n} \sum_{i=1}^n e_i^2$$

Rewriting the Mean Squared Error

- ▶ Define $\vec{y} = (y_1, \dots, y_n)^T$. Then:

$$\vec{e} = \begin{pmatrix} (\vec{w} \cdot \text{Aug}(\vec{x}^{(1)})) - y_1 \\ (\vec{w} \cdot \text{Aug}(\vec{x}^{(2)})) - y_2 \\ \vdots \\ (\vec{w} \cdot \text{Aug}(\vec{x}^{(n)})) - y_n \end{pmatrix} =$$

- ▶ \vec{h} is the vector of predictions.

Rewriting the Mean Squared Error

- ▶ So far: $R_{\text{sq}}(\vec{w}) = \frac{1}{n} \|\vec{e}\|^2$, and $\vec{e} = \vec{h} - \vec{y}$.

- ▶ Therefore:

$$R_{\text{sq}}(\vec{w}) = \frac{1}{n} \|\vec{h} - \vec{y}\|^2$$

- ▶ \vec{w} is hidden inside of \vec{h} , let's pull it out.

Rewriting the Mean Squared Error

- ▶ Define the **design matrix** X :

$$X = \begin{pmatrix} \text{Aug}(\vec{x}^{(1)}) & \longrightarrow \\ \text{Aug}(\vec{x}^{(2)}) & \longrightarrow \\ \vdots & \vdots \\ \text{Aug}(\vec{x}^{(n)}) & \longrightarrow \end{pmatrix} = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_d^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_d^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \dots & x_d^{(n)} \end{pmatrix}$$

- ▶ Then $\vec{h} = X\vec{w}$.

Rewriting the Mean Squared Error

- ▶ The mean squared error is:

$$R_{\text{sq}}(\vec{w}) = \frac{1}{n} \|X\vec{w} - \vec{y}\|^2$$

where X is the **design matrix** containing the data, \vec{w} is the **parameter vector**, and \vec{y} is the vector of **observations** (or right answers).

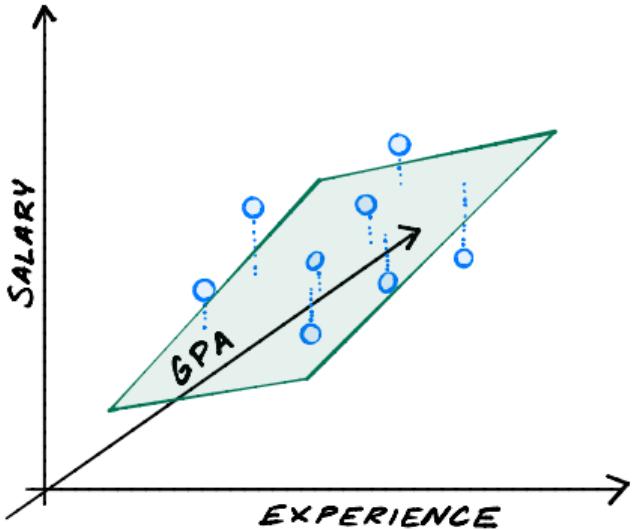
- ▶ To minimize MSE: take derivative (gradient), set to zero, solve.

Minimizing the MSE: Gradient Edition

- ▶ The vector of partial derivatives is called the **gradient**:

$$\left(\frac{\partial R_{\text{sq}}}{\partial w_0}(\vec{w}), \quad \frac{\partial R_{\text{sq}}}{\partial w_1}(\vec{w}), \quad \frac{\partial R_{\text{sq}}}{\partial w_2}(\vec{w}), \quad \dots, \quad \frac{\partial R_{\text{sq}}}{\partial w_d}(\vec{w}) \right)^T$$

- ▶ Written: $\nabla_{\vec{w}} R_{\text{sq}}(\vec{w})$ or $\frac{dR_{\text{sq}}}{d\vec{w}}(\vec{w})$
- ▶ Strategy:
 1. Compute the gradient of $R_{\text{sq}}(\vec{w})$.
 2. Set it to zero and solve for \vec{w} .



CSE 151A
Intro to Machine Learning

Lecture 07 – Part 02
The Gradient

Minimizing the MSE: Gradient Edition

- ▶ The vector of partial derivatives is called the **gradient**:

$$\left(\frac{\partial R_{\text{sq}}}{\partial w_0}(\vec{w}), \quad \frac{\partial R_{\text{sq}}}{\partial w_1}(\vec{w}), \quad \frac{\partial R_{\text{sq}}}{\partial w_2}(\vec{w}), \quad \dots, \quad \frac{\partial R_{\text{sq}}}{\partial w_d}(\vec{w}) \right)^T$$

- ▶ Written: $\nabla_{\vec{w}} R_{\text{sq}}(\vec{w})$ or $\frac{dR_{\text{sq}}}{d\vec{w}}(\vec{w})$
- ▶ Strategy:
 1. Compute the gradient of $R_{\text{sq}}(\vec{w})$.
 2. Set it to zero and solve for \vec{w} .

Minimizing the MSE

- We want to compute:

$$\frac{d}{d\vec{w}} \left[R_{\text{sq}}(\vec{w}) \right] = \frac{d}{d\vec{w}} \left[\|X\vec{w} - \vec{y}\|^2 \right]$$

- Step 1: Rewrite squared norm using dot product.
Recall:

$$(A + B)^T = A^T + B^T$$

$$(AB)^T = B^T A^T$$

$$\vec{u} \cdot \vec{v} = \vec{v} \cdot \vec{u}$$

$$(\vec{u} + \vec{v}) \cdot (\vec{w} + \vec{z}) = \vec{u} \cdot \vec{w} + \vec{u} \cdot \vec{z} + \vec{v} \cdot \vec{w} + \vec{v} \cdot \vec{z}$$

$$\|\vec{u}\|^2 = \vec{u} \cdot \vec{u}$$

Step 1: Rewriting squared norm

$$\|X\vec{w} - \vec{y}\|^2 =$$

=

=

=

Step 2: Take gradients

$$\frac{d}{d\vec{w}} \left[R_{\text{sq}}(\vec{w}) \right] = \frac{d}{d\vec{w}} \left[\vec{w}^T X^T X \vec{w} - 2\vec{y}^T X \vec{w} + \vec{y}^T \vec{y} \right]$$

=

Claim

- ▶ $\frac{d}{d\vec{w}} [\vec{w}^T X^T X \vec{w}] = 2X^T X \vec{w}$
- ▶ $\frac{d}{d\vec{w}} [\vec{y}^T X \vec{w}] = X^T \vec{y}$
- ▶ $\frac{d}{d\vec{w}} [\vec{y}^T \vec{y}] = 0$

Example

Show $\frac{d}{d\vec{w}} [\vec{y}^T X \vec{w}] = X^T \vec{y}$

Step 2: Take gradients

$$\frac{d}{d\vec{w}} \left[R_{\text{sq}}(\vec{w}) \right] = \frac{d}{d\vec{w}} \left[\vec{w}^T X^T X \vec{w} - 2\vec{y}^T X \vec{w} + \vec{y}^T \vec{y} \right]$$

=

The Normal Equations

- ▶ To minimize $R_{\text{sq}}(\vec{w})$, set gradient to zero, solve for \vec{w} :

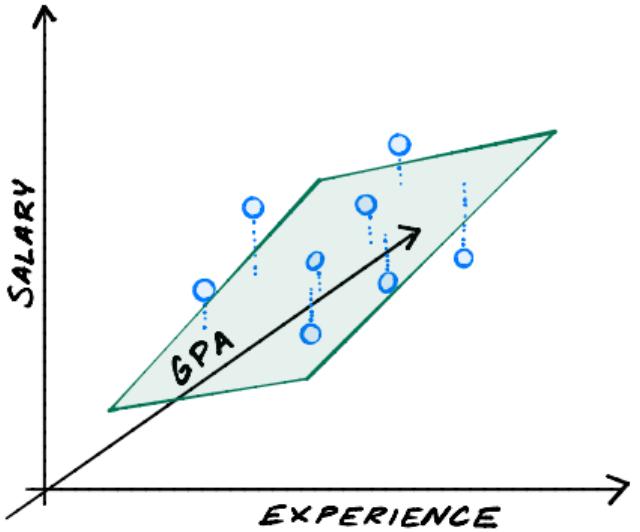
$$2X^T X \vec{w} - 2X^T \vec{y} = 0 \implies X^T X \vec{w} = X^T \vec{y}$$

- ▶ This is a system of equations in matrix form, called the **normal equations**.
- ▶ Solution²: $\vec{w} = (X^T X)^{-1} X^T \vec{y}$.

²Don't actually compute inverse! Use Gaussian elimination.

Regression with Multiple Features

- ▶ We want to find \vec{w} which minimizes $\|X\vec{w} - \vec{y}\|^2$.
- ▶ The answer: $\vec{w} = (X^T X)^{-1} X^T \vec{y}$.



CSE 151A
Intro to Machine Learning

Lecture 07 – Part 03
Interpreting Weights

Interpreting \vec{w}

- ▶ With d features, \vec{w} has $d + 1$ entries.
- ▶ w_0 is the **bias**.
- ▶ w_1, \dots, w_d each give the **weight** of a feature.

$$H(\vec{x}) = w_0 + w_1 x_1 + \dots + w_d x_d$$

- ▶ Sign of w_i tells us about relationship between i th feature and outcome.

Example: Predicting Sales

- ▶ For each of 26 stores, we have:
 - ▶ net sales,
 - ▶ size (sq ft),
 - ▶ inventory,
 - ▶ advertising expenditure,
 - ▶ district size,
 - ▶ number of competing stores.
- ▶ Goal: predict net sales given size, inventory, etc.

To begin...

$$H(\text{size, competitors}) = w_0 + w_1 \times \text{size} + w_2 \times \text{competitors}$$

What will be the sign of w_1 and w_2 ?

$$H(\text{size, competitors}) = w_0 + w_1 \times \text{size} + w_2 \times \text{competitors}$$

(DEMO)

Interpreting Weights

Which has the greatest effect on the outcome?

- A) size: $w_1 = 16.20$
- B) inventory: $w_2 = 0.17$
- C) advertising: $w_3 = 11.53$
- D) district size: $w_4 = 13.58$
- E) competing stores: $w_5 = -5.31$

Which features are most “important”?

- ▶ Not necessarily the feature with largest weight.
- ▶ Features are measured in different units, scales.
- ▶ We should standardize each feature.

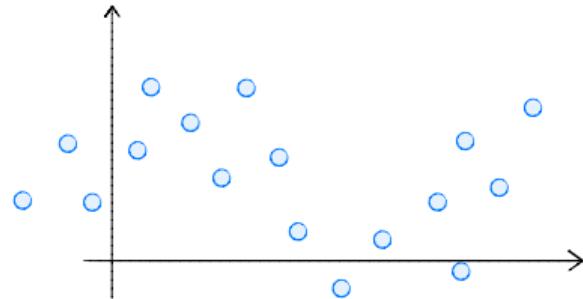
Standard Units

- ▶ Standardize each feature (store size, inventory, etc.) separately.
- ▶ No need to standardize outcome (net sales).
- ▶ Solve normal equations. The resulting w_0, w_1, \dots, w_d are called the **standardized regression coefficients**.
- ▶ They can be directly compared to one another.

(DEMO)

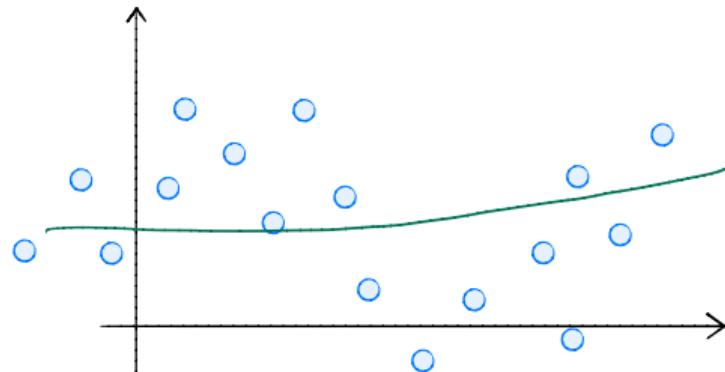
Fitting Non-Linear Patterns

- ▶ Fit a 4th-order polynomial to the data:



- ▶ Fit rule of the form $H(x) = w_1 x^4 + w_0$.
 - ▶ Define $z_i = x_i^4$.
 - ▶ Use $w_1 = \frac{\sum(z_i - \bar{z})(y_i - \bar{y})}{\sum(z_i - \bar{z})^2}$ and $w_0 = \bar{y} - w_1 \bar{z}$.

The Result



- ▶ The rule $H(x) = w_1x^4 + w_0$ **underfits** the data.
- ▶ We need a more complicated rule:

$$H(x) = w_4x^4 + w_3x^3 + w_2x^2 + w_1x + w_0$$

The Trick

- ▶ Treat x, x^2, x^3, x^4 as different features.
- ▶ Create design matrix:

$$X = \begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 & x_1^4 \\ 1 & x_2 & x_2^2 & x_2^3 & x_2^4 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 & x_n^4 \end{pmatrix}$$

- ▶ Solve $X^T X \vec{w} = X^T \vec{y}$ for \vec{w} , as usual.
- ▶ Works for more than just polynomials.

(DEMO)

Polynomial Regression

- ▶ More complicated patterns can be fit with higher-order polynomials.
- ▶ If there are n points, a $n + 1$ degree polynomial can fit them exactly.
- ▶ But for high-order polynomials, it becomes **very hard** to solve the normal equations (numerical accuracy).

Polynomial Regression with Multiple Features

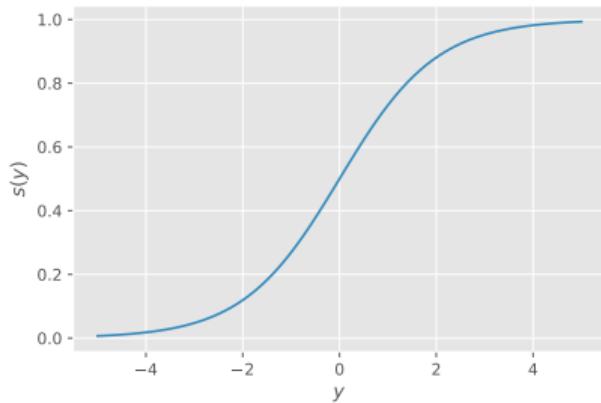
- ▶ Suppose we want to fit a rule of the form:

$$\begin{aligned}H(\text{size, competitors}) &= w_0 + w_1 \text{size} + w_2 \text{size}^2 \\&\quad + w_3 \text{competitors} + w_4 \text{competitors}^2 \\&= w_0 + w_1 s + w_2 s^2 + w_3 c + w_4 c^2\end{aligned}$$

- ▶ Make design matrix:

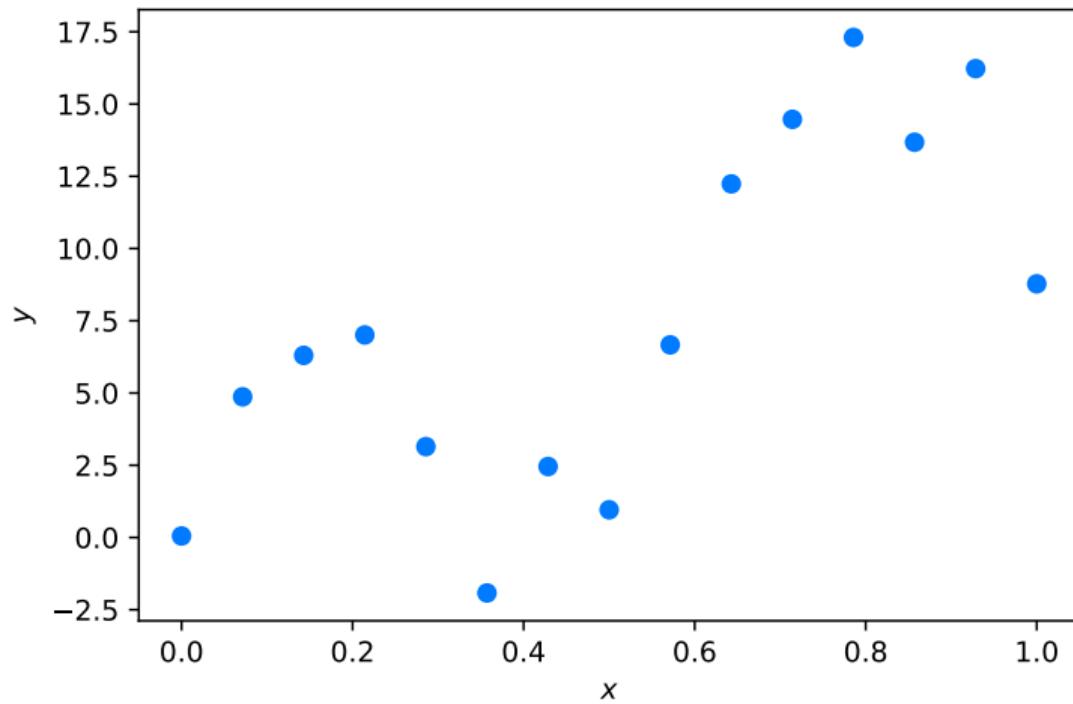
$$X = \begin{pmatrix} 1 & s_1 & s_1^2 & c_1 & c_1^2 \\ 1 & s_2 & s_2^2 & c_2 & c_2^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & s_n & s_n^2 & c_n & c_n^2 \end{pmatrix}$$

Where c_i and s_i are the competitors and size of the i th store.



CSE 151A
Intro to Machine Learning

Lecture 08 – Part 01
Model Complexity



Empirical Risk Minimization

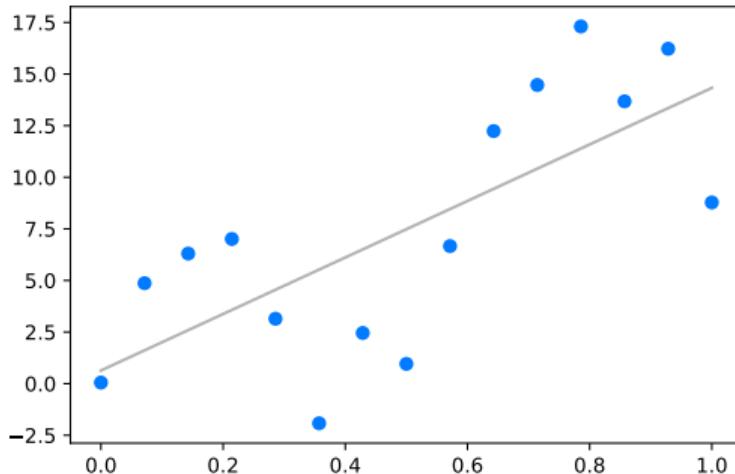
1. Pick a **model**.
 - ▶ E.g., linear prediction rules.
2. Pick a loss.
 - ▶ E.g., mean squared error.
3. Find a prediction rule minimizing the **risk**.

Big Decision

- ▶ Pick a model.
- ▶ Picking the wrong model causes problems.

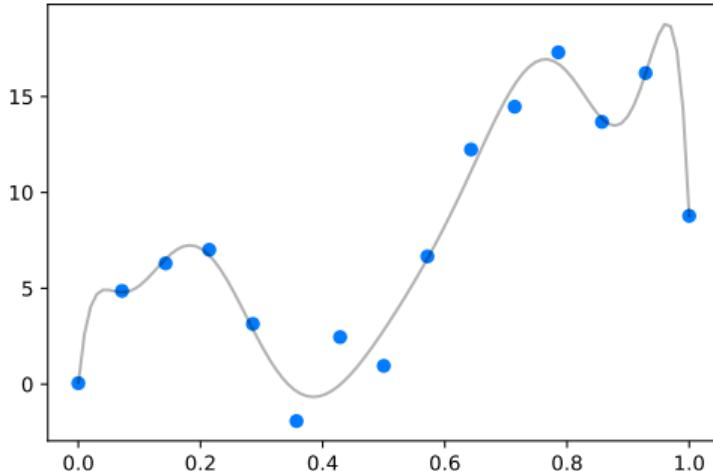
Underfitting

- ▶ Fit $H(x) = w_0 + w_1 x$?
 - ▶ We have **underfit** the data.



Overfitting

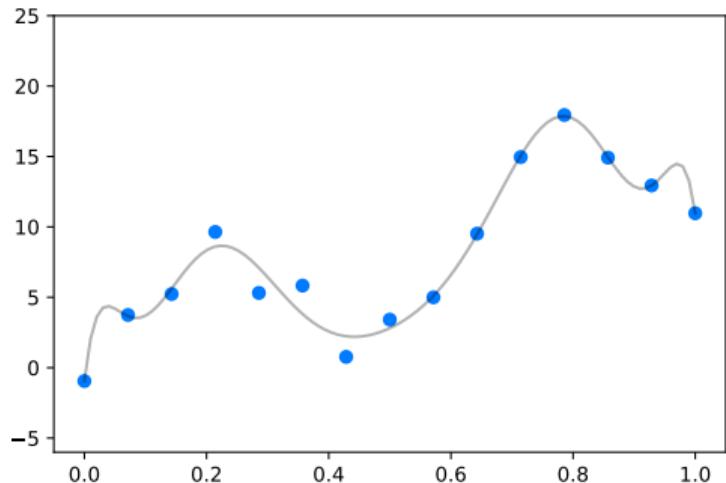
- ▶ Fit $H(x) = w_0 + w_1x + w_2x^2 + \dots + w_{10}x^{10}$?
 - ▶ We have **overfit** the data.



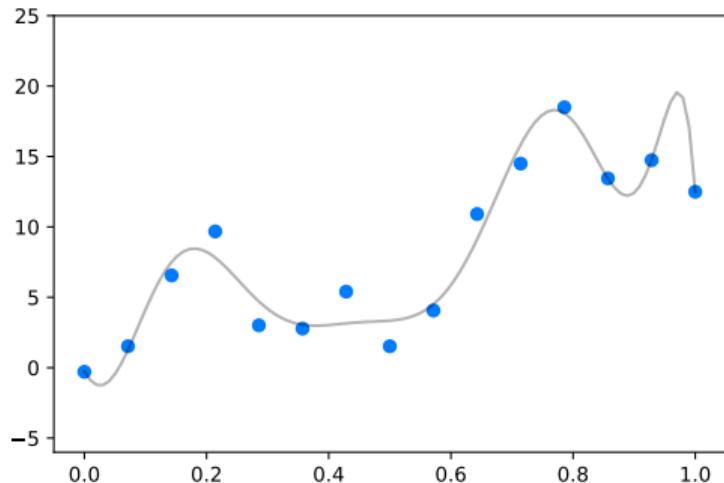
Model Complexity

- ▶ Difference? **Complexity**.
- ▶ Complex models are highly flexible.
 - ▶ They tend to **overfit**.
- ▶ Simple models are stiff.
 - ▶ They tend to **underfit**.

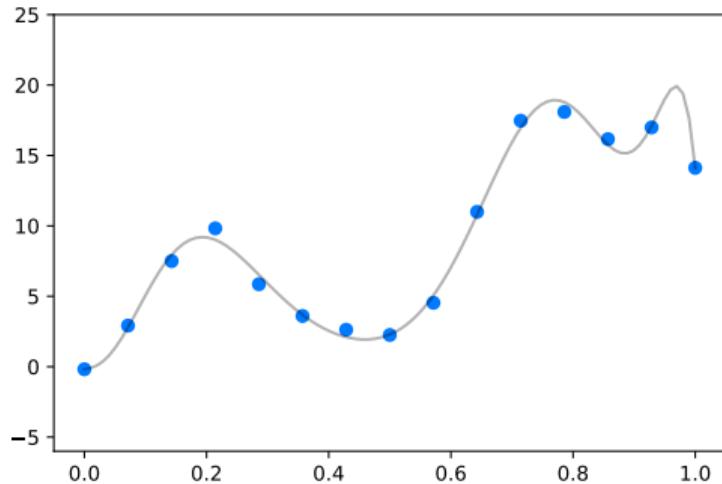
Degree 10 Polynomial



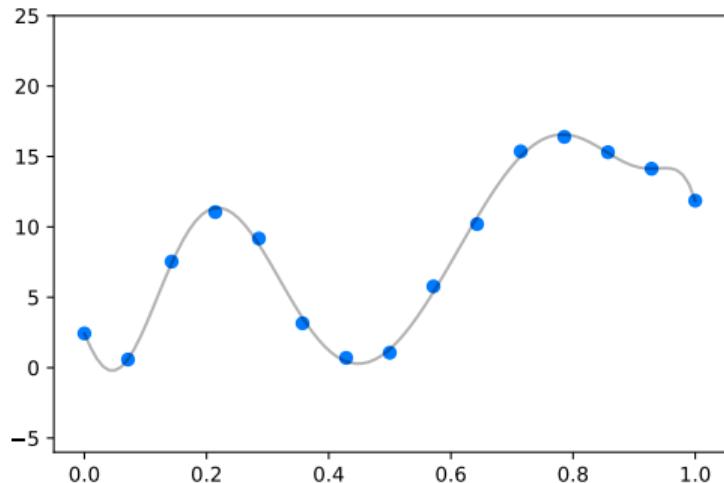
Degree 10 Polynomial



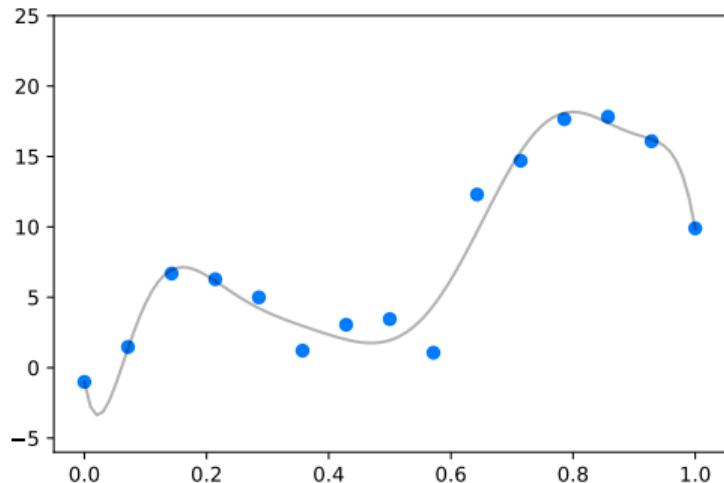
Degree 10 Polynomial



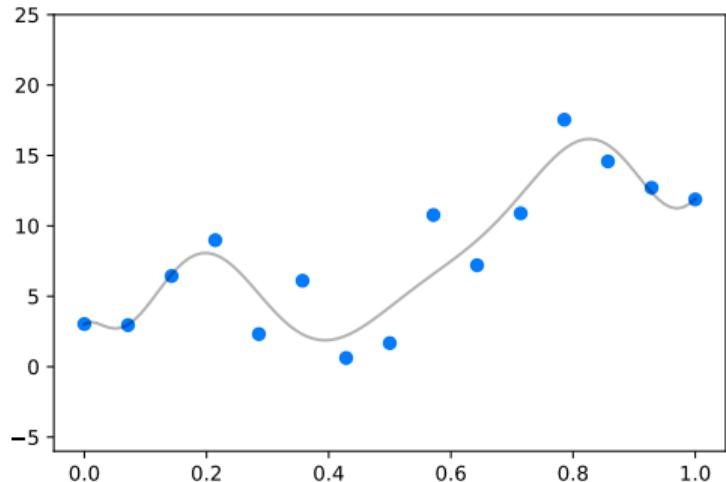
Degree 10 Polynomial



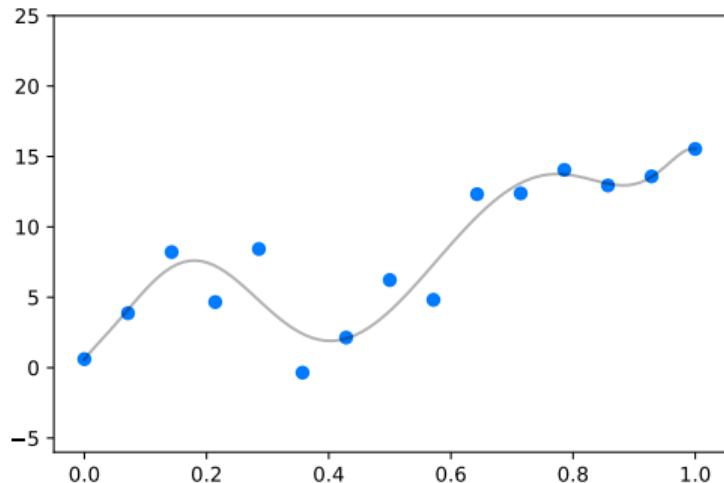
Degree 10 Polynomial



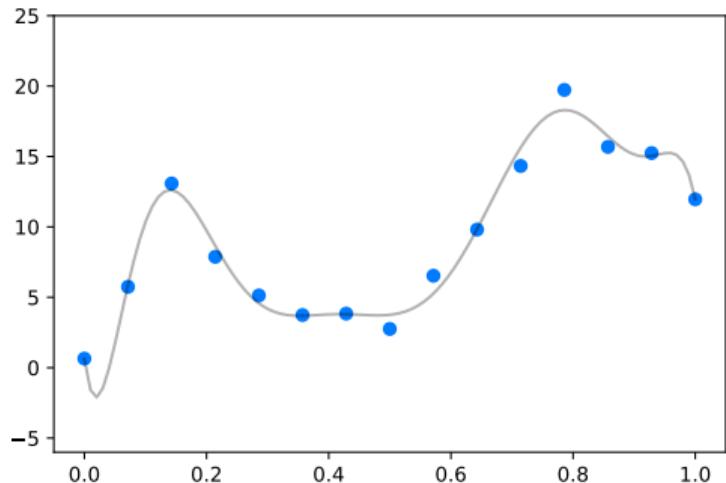
Degree 10 Polynomial



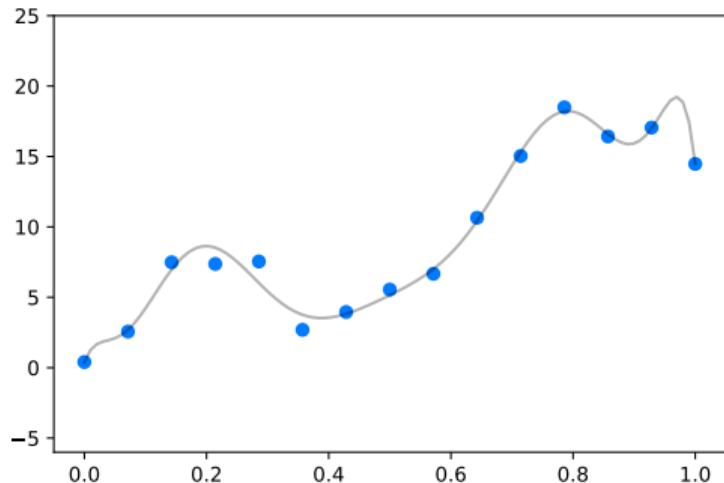
Degree 10 Polynomial



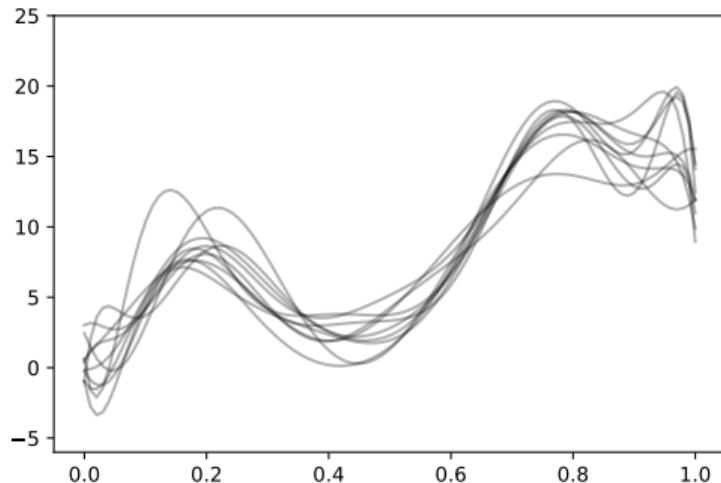
Degree 10 Polynomial



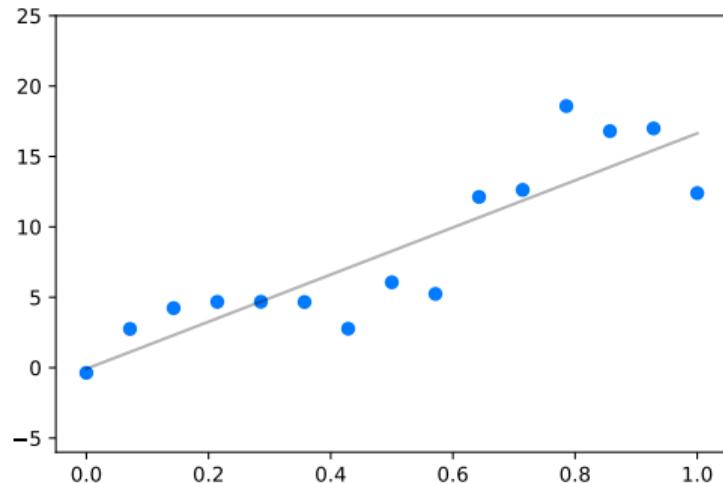
Degree 10 Polynomial



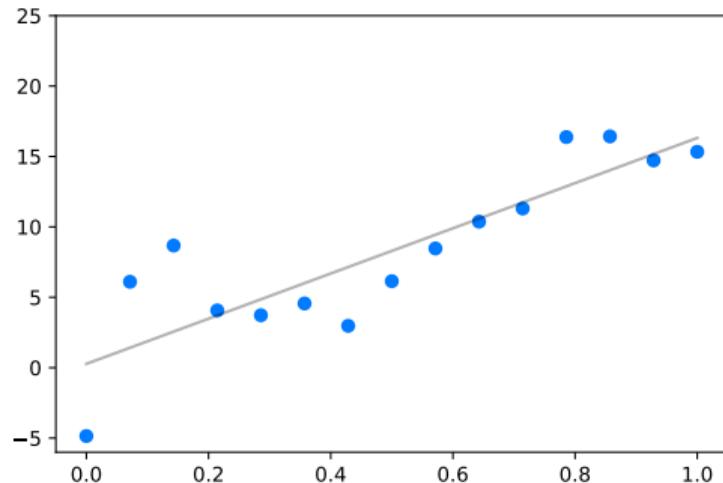
Degree 10 Polynomial



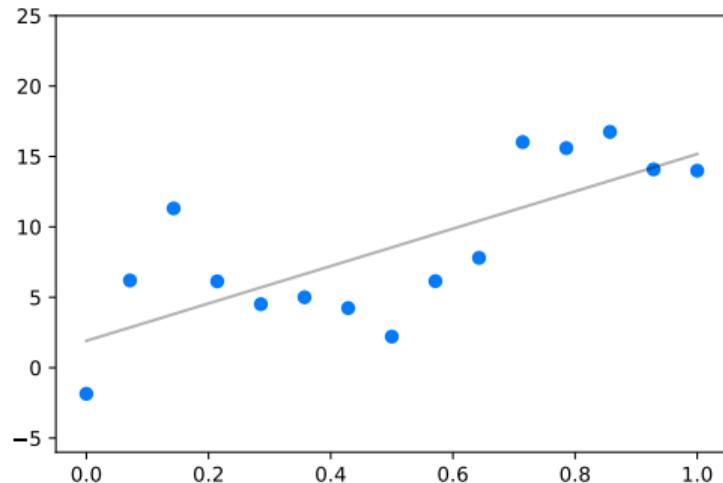
Degree 1 Polynomial



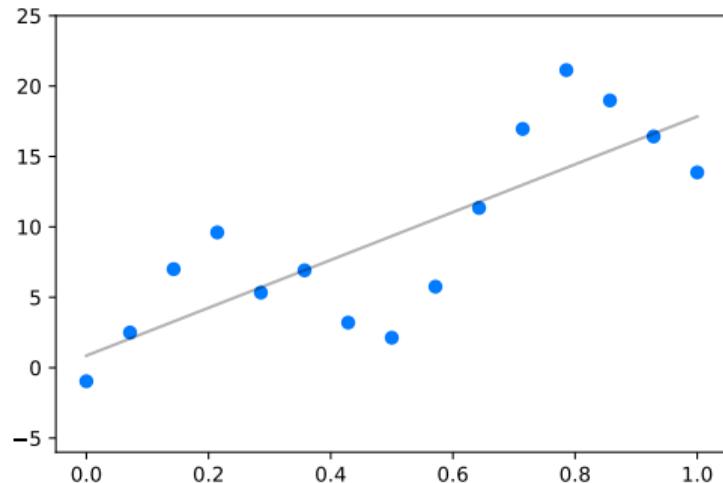
Degree 1 Polynomial



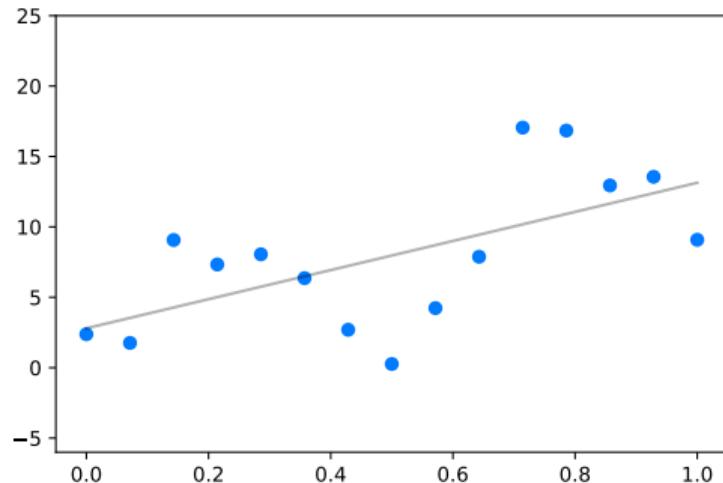
Degree 1 Polynomial



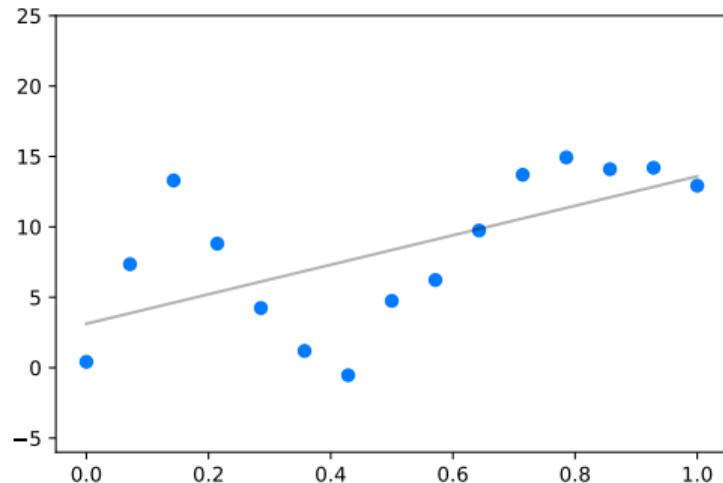
Degree 1 Polynomial



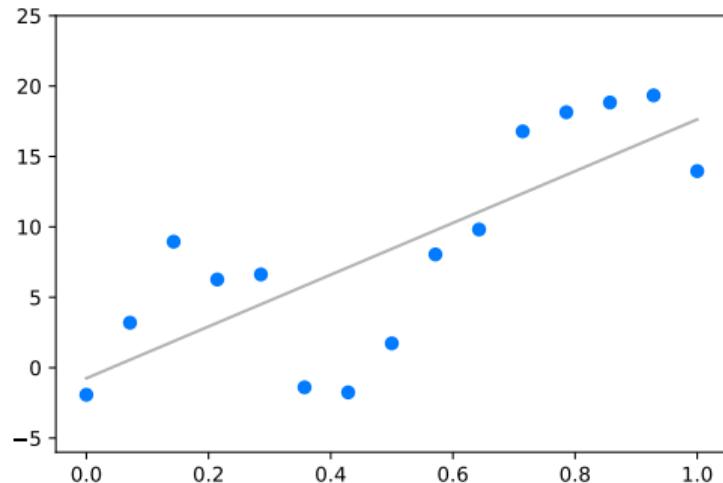
Degree 1 Polynomial



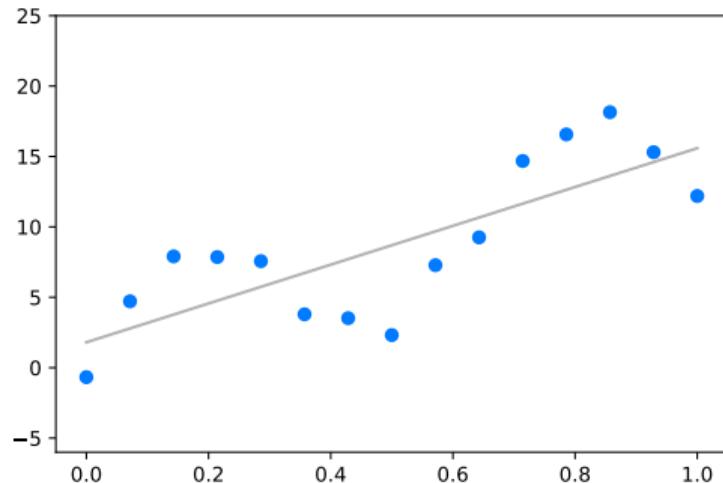
Degree 1 Polynomial



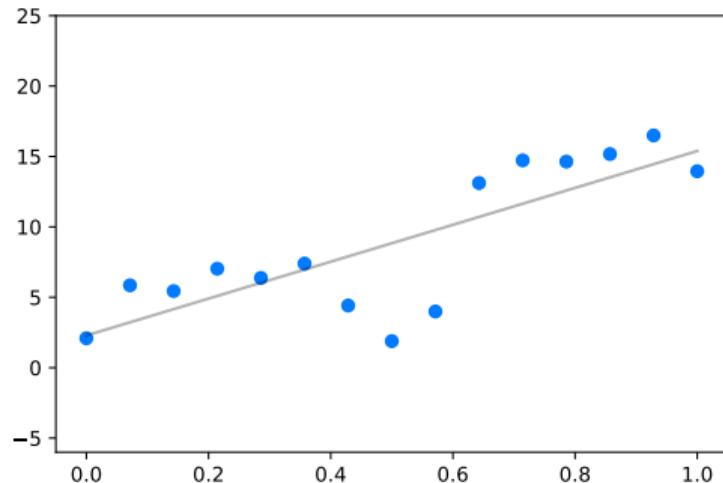
Degree 1 Polynomial



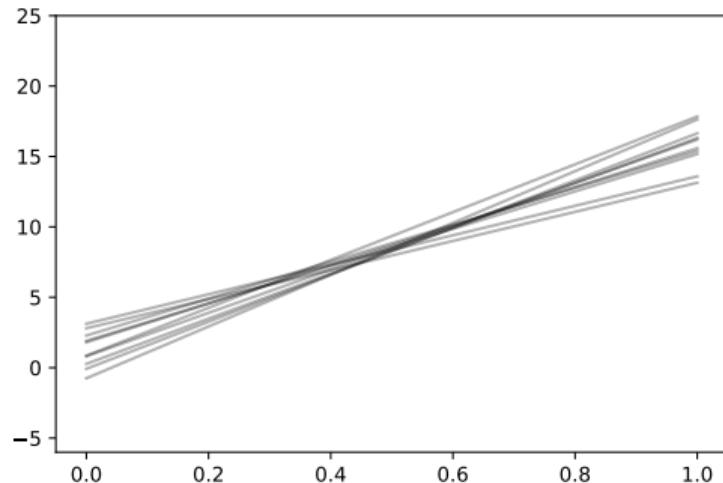
Degree 1 Polynomial



Degree 1 Polynomial



Degree 1 Polynomial



Example: kNN

- ▶ 1NN: complex model, likely to overfit.
- ▶ 20NN: less complex model, likely to underfit.

Choosing Model Complexity

- ▶ How do we choose between two models?
 - ▶ Between degree 10 and degree 1?
 - ▶ Between 1NN and 20NN?
- ▶ Not always obvious.

Bad Idea: Use training MSE

- ▶ Which has smaller MSE on training data?

Bad Idea: Use training MSE

- ▶ Which has smaller MSE on training data?
- ▶ **Problem:** Best 10-degree polynomial will **always** have smaller MSE on training data.

Good Idea: Use validation MSE

- ▶ We care about **generalization**.
- ▶ So keep a small amount of data hidden in a **validation set**.
- ▶ Fit model on training data, compute MSE on **validation set**.
- ▶ Pick whichever model has smaller validation error.

What do you expect?

- ▶ You fit a complex model on training data.
- ▶ Test it on a validation set.
- ▶ Likely: validation MSE > training MSE.

What do you expect?

- ▶ You fit a very simple model on training data.
- ▶ Test it on a validation set.
- ▶ Likely: validation MSE \approx training MSE.

Cross-Validation

- ▶ We want all the training data we can get.
- ▶ Reserving some of it is wasteful.
- ▶ Idea: **split** data into pieces, each takes turn as validation set.

k-Fold Cross Validation

1. Split data set into k pieces, S_1, \dots, S_k .
2. Loop k times; on iteration i :
 - ▶ Use S_i as validation set; rest as training.
 - ▶ Compute validation error ϵ_i
3. Overall error: $\frac{1}{k} \sum \epsilon_i$

Leave-One-Out Cross Validation

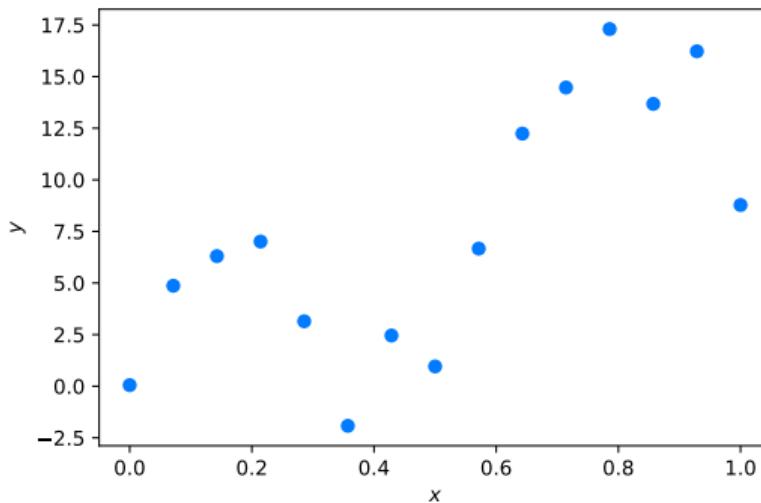
- ▶ Suppose we have n labeled data points.
- ▶ LOOCV: k -fold CV with $k = n$.

Another Approach

- ▶ We can control complexity by choosing model.
- ▶ Also: via **regularization**.

Regularization

- ▶ Let's fit a complex model: $w_0 + w_x x + \dots + w_{10} x^{10}$.
- ▶ But impose a budget on weights, w_0, \dots, w_d .



Budgeting Weights

- ▶ One way to budget: ask that $\|\vec{w}\|^2$ is small.
- ▶ Before: minimize

$$R_{\text{sq}}(\vec{w}) = \frac{1}{n} \sum_{i=1}^n (\vec{w} \cdot \vec{x}^{(i)} - y_i)^2$$

- ▶ Now: minimize

$$\tilde{R}_{\text{sq}}(\vec{w}) = \frac{1}{n} \sum_{i=1}^n (\vec{w} \cdot \vec{x}^{(i)} - y_i)^2 + \lambda \|\vec{w}\|^2$$

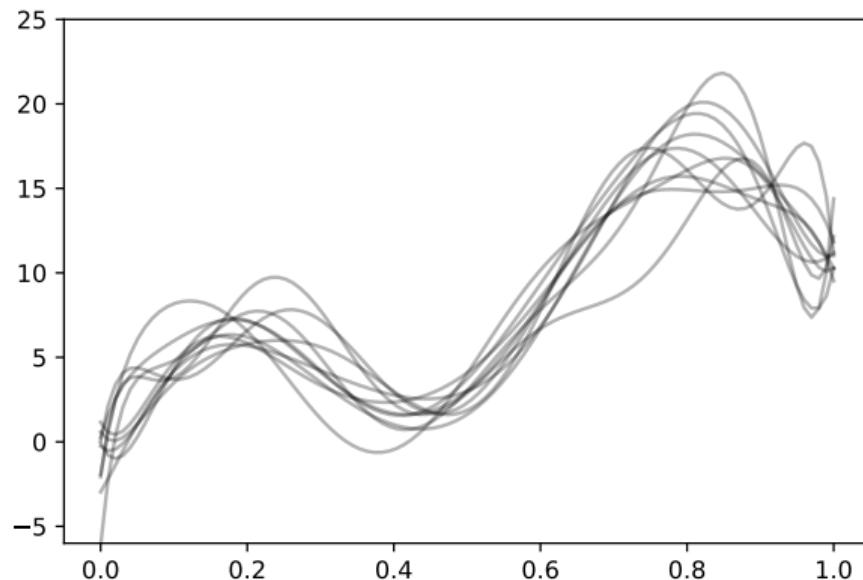
Solution

- The **regularized** Normal Equations:

$$(X^T X + \lambda I) \vec{w} = X^T \vec{y}$$

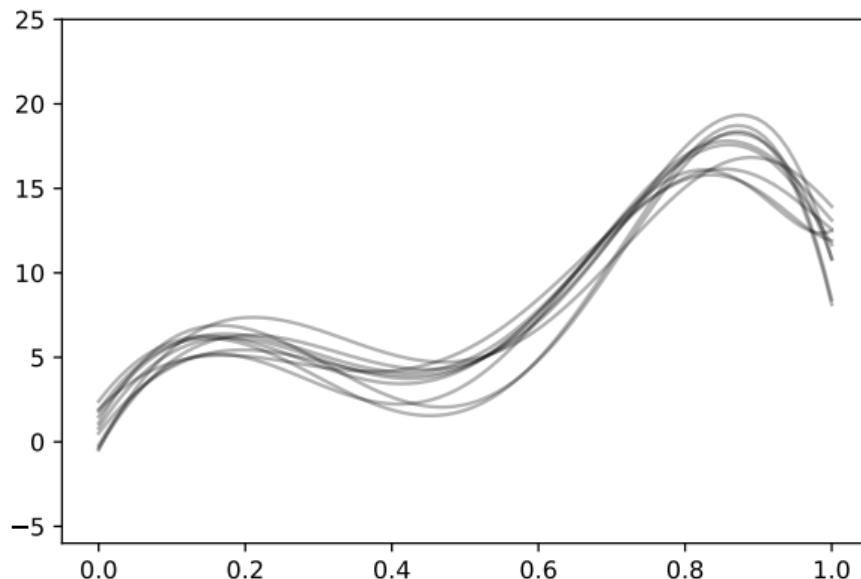
Example

$$\lambda = 0$$



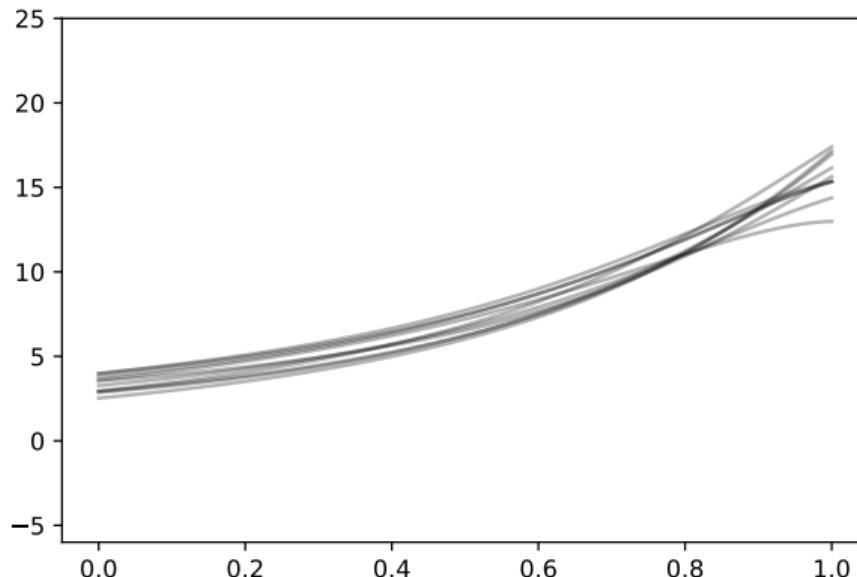
Example

$$\lambda = 1 \times 10^{-4}$$



Example

$$\lambda = 1$$

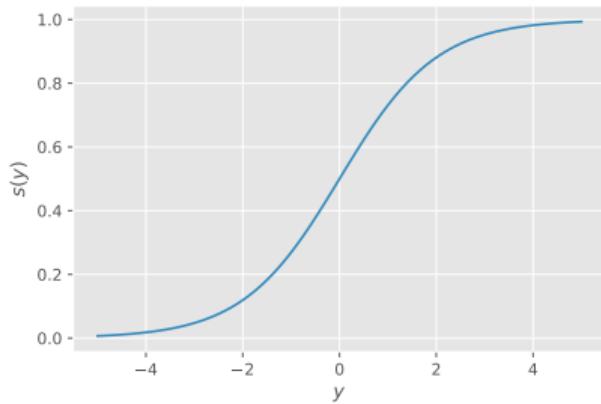


Regularization

- ▶ As λ increases, simpler models preferred.
- ▶ Pick λ using cross-validation.

Other Penalizations

- ▶ $\|\vec{w}\|_2^2$ is ℓ_2 regularization (explicit solution)
 - ▶ a.k.a., **ridge regression**
- ▶ $\|\vec{w}\|_1$ is ℓ_1 regularization (no explicit solution)
 - ▶ a.k.a., **the LASSO**
 - ▶ encourages **sparse** \vec{w}



CSE 151A
Intro to Machine Learning

Lecture 08 – Part 02
Logistic Regression

Note

- ▶ The midterm will cover everything up to right now.
- ▶ Regularization: **yes**.
- ▶ Logistic regression: **no**.

Predicting Heart Disease

- ▶ **Classification problem:** Does a patient have heart disease?
- ▶ **Features:** blood pressure, cholesterol level, exercise amount, maximum heart rate, sex

Better idea...

- ▶ Instead of predicting **yes/no**...
- ▶ Give a **probability** that they have heart disease.
 - ▶ 1 = definitely yes
 - ▶ 0 = definitely no
 - ▶ 0.75 = probably, yes
 - ▶ ...

Associations

- ▶ If cholesterol is high, increased likelihood.
 - ▶ Positive association.
- ▶ If exercise is low, increased likelihood.
 - ▶ Negative association.

The Model

- ▶ Measure cholesterol (x_1), exercise (x_2), etc.
- ▶ **Idea:** weighted¹ “vote” for heart disease:

$$w_1x_1 + w_2x_2 + \dots + w_dx_d$$

- ▶ Convention:
 - ▶ A positive number = vote for yes
 - ▶ A negative number = vote for no

¹We'll learn weights later.

The Model

- ▶ Add a “bias” term:

$$\begin{aligned} w_0 + w_1 x_1 + w_2 x_2 + \dots + w_d x_d \\ = \vec{w} \cdot \text{Aug}(\vec{x}) \end{aligned}$$

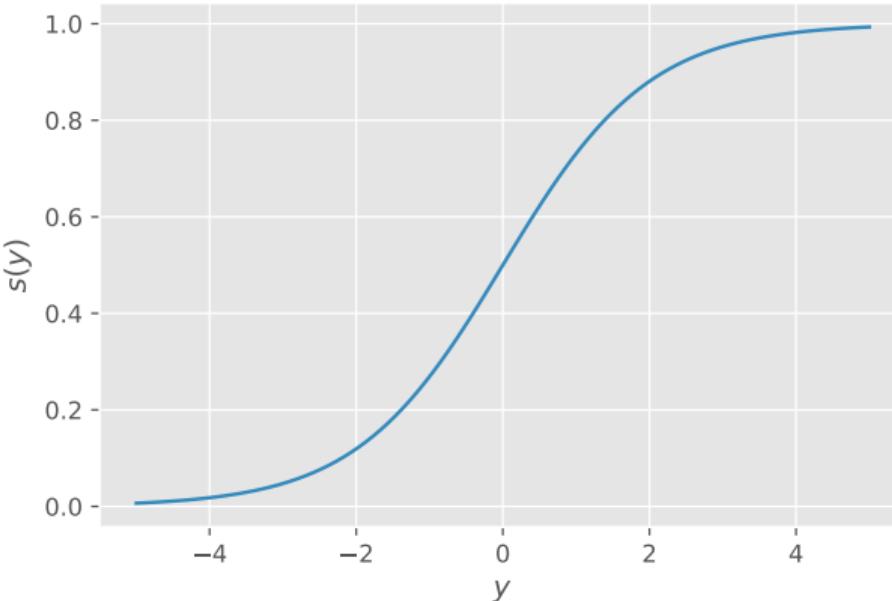
- ▶ The more positive $\vec{w} \cdot \text{Aug}(\vec{x})$, the more likely.
- ▶ The more negative $\vec{w} \cdot \text{Aug}(\vec{x})$, the less likely.

Converting to a Probability

- ▶ Probabilities are between 0 and 1.
- ▶ **Problem:** $\vec{w} \cdot \text{Aug}(\vec{x})$ can be anything in $(-\infty, \infty)$
- ▶ We need to convert it to a probability.

The Logistic Function

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$



The Model

- ▶ Our simplified model for probability of heart disease:

$$H(\vec{x}) = \sigma(\vec{w} \cdot \text{Aug}(\vec{x}))$$

- ▶ What should \vec{w} be?
- ▶ To find \vec{w} , use principle of **maximum likelihood**.

Maximum Likelihood

- ▶ Suppose you have an unfair coin.
- ▶ Probability of heads is p , unknown.
- ▶ Flip 8 times and see: H, H, T, H, H, H, H, T
- ▶ Which is more **likely**: $p = 0.5$ or $p = 0.75$?

Maximum Likelihood

- ▶ Assume coin flips are independent.
- ▶ The **likelihood** of H, H, T, H, H, H, H, T is:

$$\begin{aligned}\mathcal{L}(p) &= p \cdot p \cdot (1 - p) \cdot p \cdot p \cdot p \cdot p \cdot (1 - p) \\ &= p^6(1 - p)^2\end{aligned}$$

- ▶ Idea: find p maximizing $\mathcal{L}(p)$
 - ▶ Equivalently, find p maximizing $\log \mathcal{L}(p)$

Maximum Likelihood

Find p maximizing $\log \mathcal{L}(p) = \log p^6(1 - p)^2$:

Maximum Likelihood

- ▶ In general, given n_1 observed heads, n_2 observed tails, maximize:

$$\begin{aligned} & \log [P(F_1 = f_1) \cdot P(F_2 = f_2) \cdots \cdot P(F_n = f_n)] \\ &= \sum_{i=1}^n \log P(F_i = f_i) \end{aligned}$$

Back to Logistic Regression

- ▶ The probability that person i has heart disease:

$$H(\vec{x}^{(i)}) = \vec{w} \cdot \text{Aug}(\vec{x}^{(i)})$$

- ▶ Gather a data set, $(\vec{x}^{(1)}, y_1), \dots, (\vec{x}^{(n)}, y_n)$.
- ▶ What is the most **likely** \vec{w} ?

Maximum Likelihood

- ▶ Suppose 3 people, (+,-,+).
- ▶ Likelihood:

$$\begin{aligned} & H(\vec{x}^{(1)}) \cdot (1 - H(\vec{x}^{(2)})) \cdot H(\vec{x}^{(3)}) \\ &= \sigma(\vec{w} \cdot \text{Aug}(\vec{x}^{(1)})) \cdot (1 - \sigma(\vec{w} \cdot \text{Aug}(\vec{x}^{(2)}))) \cdot \sigma(\vec{w} \cdot \text{Aug}(\vec{x}^{(3)})) \\ &= \frac{1}{1 + e^{-\vec{w} \cdot \text{Aug}(\vec{x}^{(1)})}} \cdot \left(1 - \frac{1}{1 + e^{-\vec{w} \cdot \text{Aug}(\vec{x}^{(2)})}}\right) \cdot \frac{1}{1 + e^{-\vec{w} \cdot \text{Aug}(\vec{x}^{(3)})}} \end{aligned}$$

Observation

- ▶ Note:

$$1 - \frac{1}{1 + e^{-t}} = \frac{1}{1 + e^t}$$

Maximum Likelihood

- ▶ The likelihood:

$$\frac{1}{1 + e^{-\vec{w} \cdot \text{Aug}(\vec{x}^{(1)})}} \cdot \frac{1}{1 + e^{\vec{w} \cdot \text{Aug}(\vec{x}^{(2)})}} \cdot \frac{1}{1 + e^{-\vec{w} \cdot \text{Aug}(\vec{x}^{(3)})}}$$

- ▶ Suppose $y_i = 1$ if positive, $y_i = -1$ if negative:

$$\frac{1}{1 + e^{-y_1 \vec{w} \cdot \text{Aug}(\vec{x}^{(1)})}} \cdot \frac{1}{1 + e^{-y_2 \vec{w} \cdot \text{Aug}(\vec{x}^{(2)})}} \cdot \frac{1}{1 + e^{-y_3 \vec{w} \cdot \text{Aug}(\vec{x}^{(3)})}}$$

Maximum Likelihood

- ▶ In general, the likelihood is:

$$\mathcal{L}(\vec{w}) = \prod_{i=1}^n \frac{1}{1 + e^{-y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)})}}$$

- ▶ The **log likelihood** is:

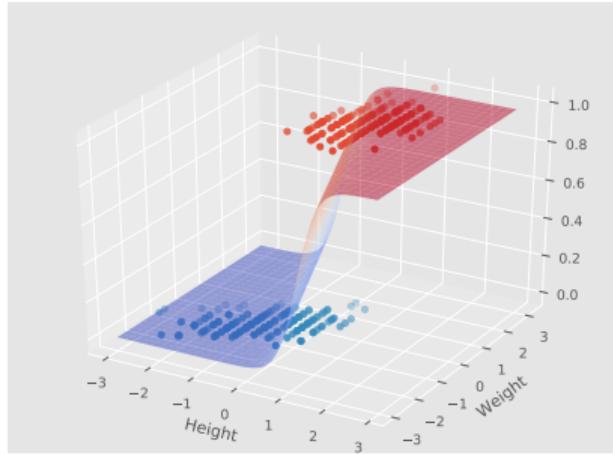
$$\log \mathcal{L}(\vec{w}) = - \sum_{i=1}^n \log \left[1 + e^{-y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)})} \right]$$

Maximizing Likelihood

- ▶ **Goal:** find \vec{w} maximizing $\log \mathcal{L}$
- ▶ Take gradient, set to zero, solve?
- ▶ **Problem:** try it, you'll get stuck.
- ▶ Unlike least-squares regression, there is no explicit solution.

Next Time

How to maximize the log loss with gradient descent.



CSE 151A

Intro to Machine Learning

Lecture 09 – Part 01
About the Midterm

The Midterm

- ▶ First midterm is Friday.
- ▶ Covers everything from Weeks 01 – 04.
 - ▶ excluding logistic regression.
- ▶ Focus is on the **essentials**.

Format

- ▶ Canvas quiz.
 - ▶ Random order/subset, you can change answers.
- ▶ Multiple choice, T/F, short answer.
 - ▶ Result of simple calculation, explanation, etc.
- ▶ Open book, open notes, open Google, etc.
 - ▶ No proctoring software/webcam needed.
- ▶ However, **no collaboration**.

Logistics

- ▶ Exam will be posted on Canvas at 00:00 AM PST.
- ▶ Exam will disappear at 22:30 PM PST.
- ▶ You can start whenever, you'll have 1.5 hours.
- ▶ Open book, open notes, open Google, etc.
 - ▶ Exam designed to take \approx 50 minutes.

Corrections and Clarifications

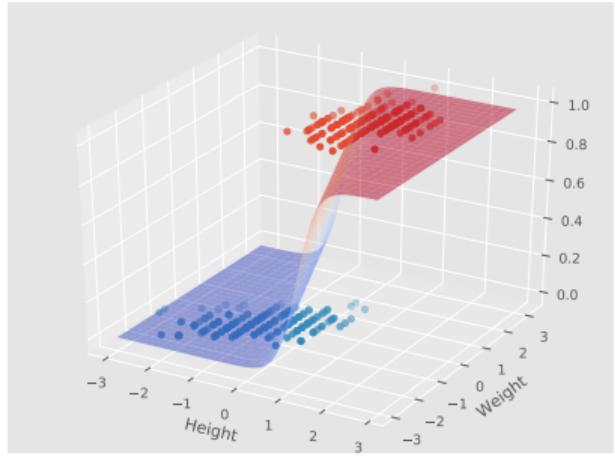
- ▶ This makes clarifications/corrections difficult to do fairly.
- ▶ **Unfortunately, no corrections/clarifications can be made.**
- ▶ Of course, if a question contains an error, it will be thrown out after the fact.

Studying

- ▶ No practice exam.
- ▶ Focus is on **essentials**.
- ▶ Here's a sample question:

A straight line is fit to a data set $\{(x_i, y_i)\}$ using least squares regression; the slope is found to be m . The data is changed by adding 10 to each y to create a new data set $\{(x_i, y_i + 10)\}$, and least squares is used again. The new slope is m' . Which is true?

- a) $m = m'$
- b) $m < m'$
- c) $m > m'$



CSE 151A

Intro to Machine Learning

Lecture 09 – Part 02

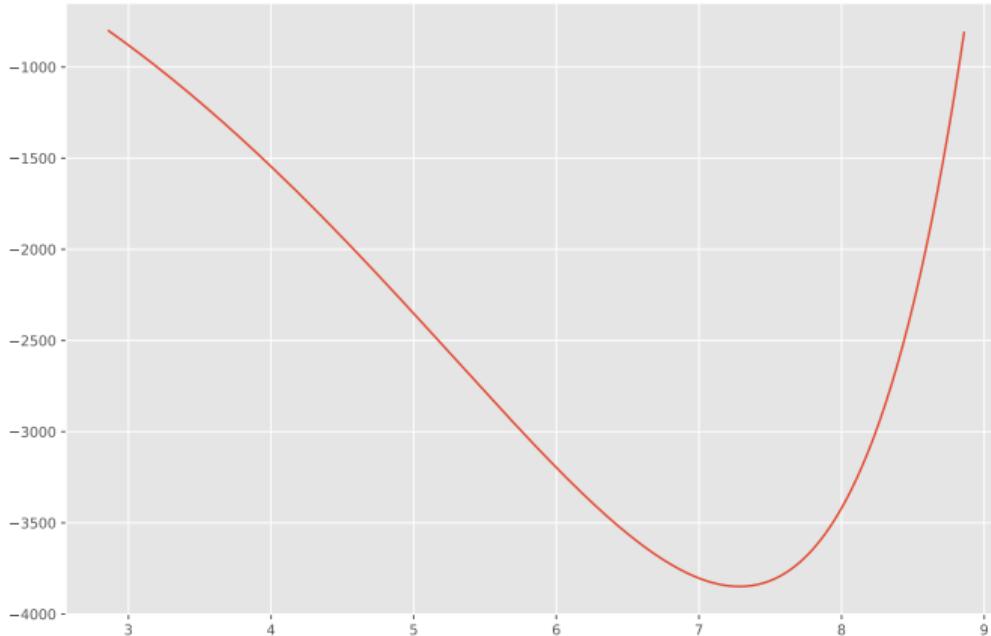
Motivating Gradient Descent

Last time...

- ▶ Set up logistic regression as optimization problem.
- ▶ Claimed: we can't solve it explicitly.
- ▶ Today: solve it using **gradient descent**.

But first...

- ▶ Minimize $f(x) = e^x - 100x^2$



Minimizing via Calculus

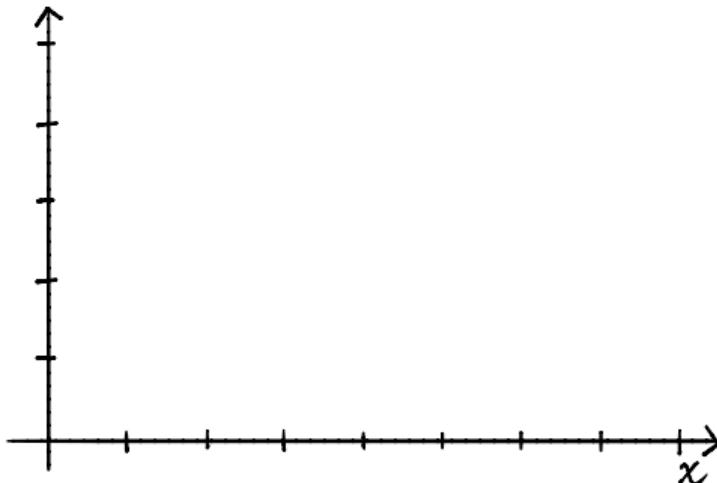
- ▶ Try setting derivative to zero, solving:

Minimizing via Calculus

- ▶ f is **differentiable**.
- ▶ But there is **no explicit solution** for $f'(x) = 0$.
- ▶ Can we use the derivative in some other way?

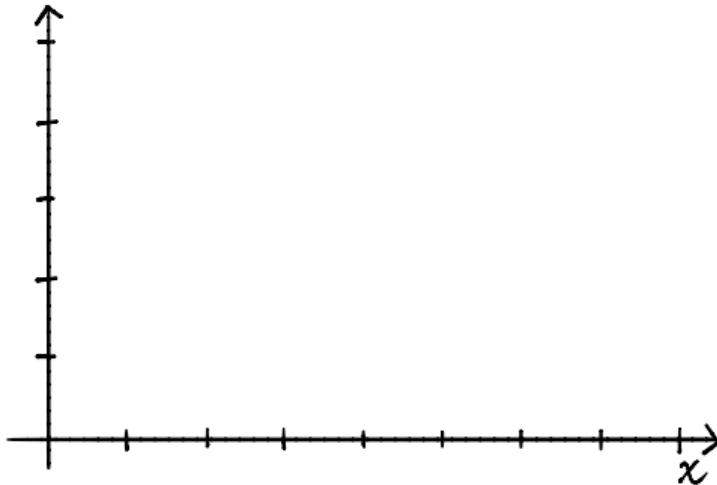
Meaning of the Derivative

- ▶ Meaning of **differentiable**: locally, f looks linear.
- ▶ $f'(x)$ is a function; it gives the **slope** at x .



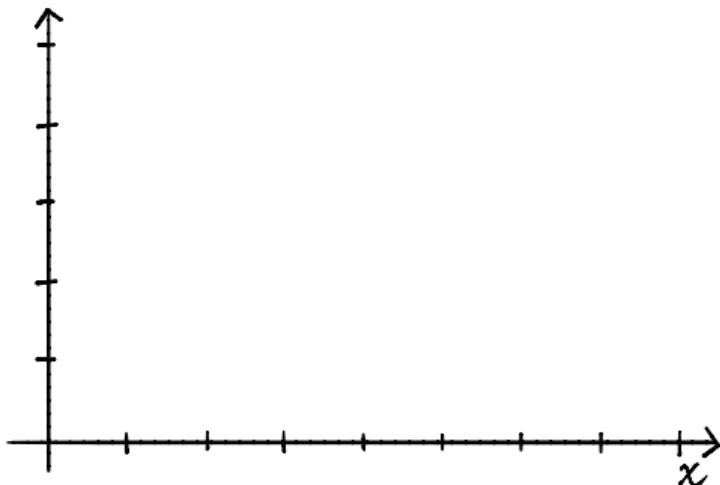
Key Idea Behind Gradient Descent

- ▶ Derivative at x tells us which way to go.
 - ▶ If the slope of f at x is **positive** then moving to the **left** decreases the value of R .



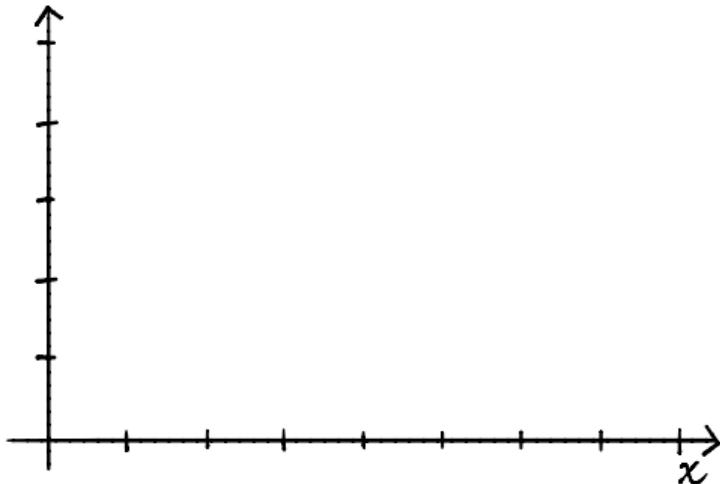
Key Idea Behind Gradient Descent

- ▶ Derivative at x tells us which way to go.
 - ▶ If the slope of f at x is **negative** then moving to the **right** decreases the value of R .



Key Idea Behind Gradient Descent

- ▶ Derivative at x tells us which way to go.
 - ▶ If the slope of f at x is **zero** then we are at a local optimum.



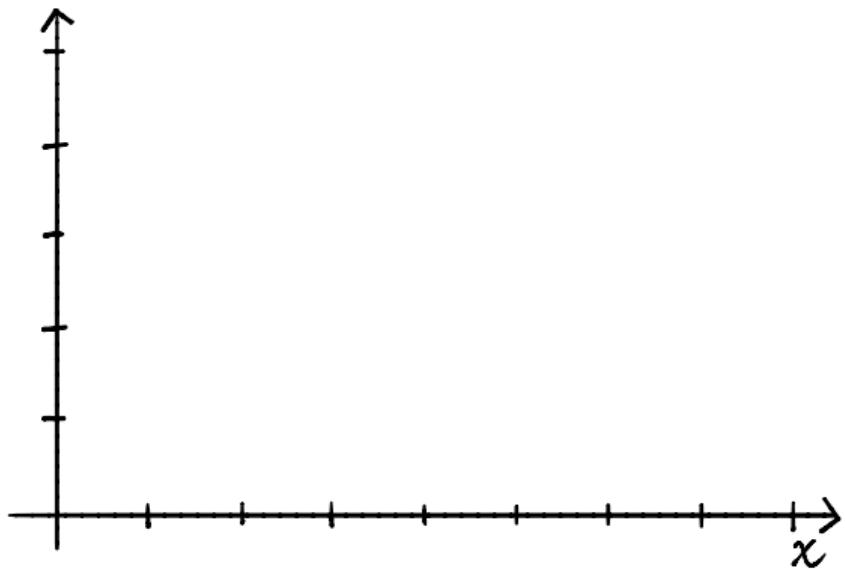
Taking a Step

- ▶ Suppose we are at x_0 . Where do we go next?
- ▶ Slope at x_0 negative? Then **increase** x_0 .
 - ▶ Step **right**.
- ▶ Slope at x_0 positive? Then **decrease** x_0 .
 - ▶ Step **left**.
- ▶ This will work:

$$x_1 = x_0 - f'(x_0)$$

Gradient Descent

- ▶ Pick α to be a positive number.
 - ▶ It is the **learning rate**.
- ▶ Pick a starting guess, x_0 .
- ▶ On step i , perform update $x_i = x_{i-1} - \alpha \cdot f'(x_{i-1})$
- ▶ Repeat until convergence
 - ▶ when x doesn't change much
 - ▶ equivalently, when $f'(x_i)$ is small



```
def gradient_descent(derivative, x, alpha, tol=1e-12):
    """Minimize using gradient descent."""
    while True:
        x_next = x - alpha * derivative(x)
        if abs(x_next - x) < tol:
            break
        x = x_next
    return x
```

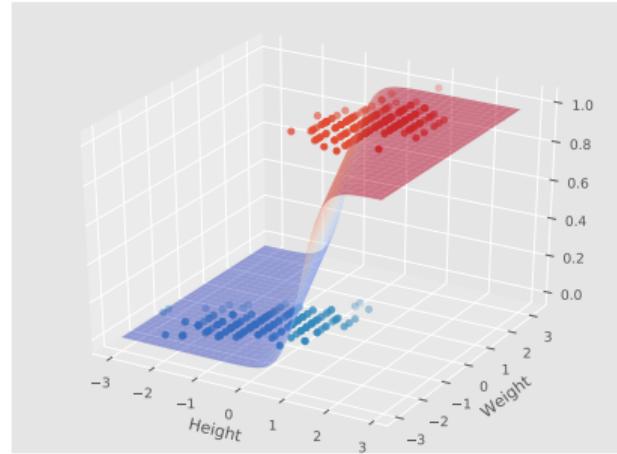
(demo)

Gradient Ascent

- ▶ Pick α to be a positive number.
 - ▶ It is the **learning rate**.
- ▶ Pick a starting guess, x_0 .
- ▶ On step i , perform update $x_i = x_{i-1} + \alpha \cdot f'(x_{i-1})$
- ▶ Repeat until convergence
 - ▶ when h doesn't change much
 - ▶ equivalently, when $f'(x_i)$ is small

Ascent vs. Descent

- ▶ Maximizing f is equivalent to minimizing $-f$.



CSE 151A

Intro to Machine Learning

Lecture 09 – Part 02

Logistic Regression

Recall: Logistic Regression

- ▶ Predict probability that person has heart disease.
- ▶ Prediction rule:

$$H_{\vec{w}}(\vec{x}) = \sigma(\vec{w} \cdot \text{Aug}(\vec{x}))$$

where

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

is the **logistic function**.

Recall: Logistic Regression

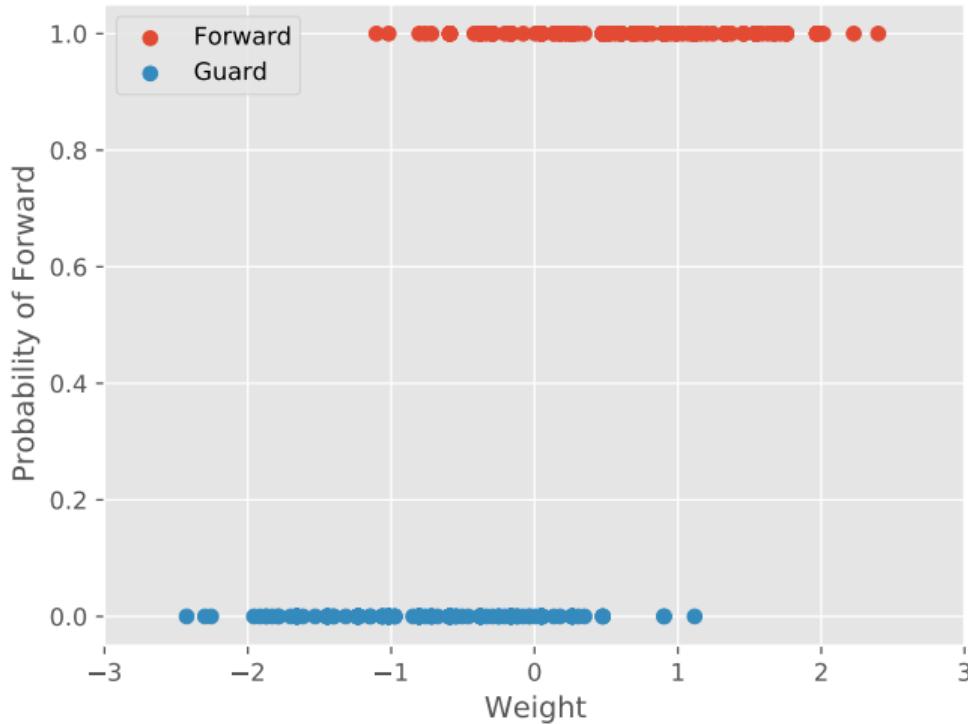
- ▶ Find **most likely** \vec{w} using data.
- ▶ **Goal:** maximize the **log likelihood**,

$$\log \mathcal{L}(\vec{w}) = - \sum_{i=1}^n \log \left[1 + e^{-y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)})} \right]$$

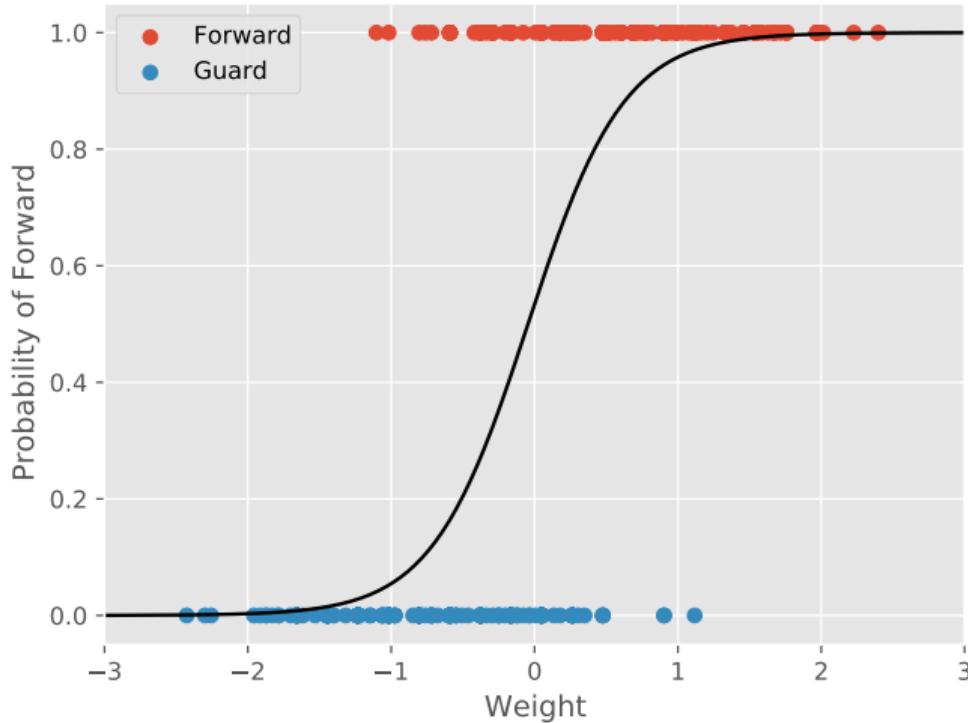
Another Example

- ▶ Given the weight of an NBA player.
- ▶ Predict probability that they are a forward.

Guards vs. Forwards



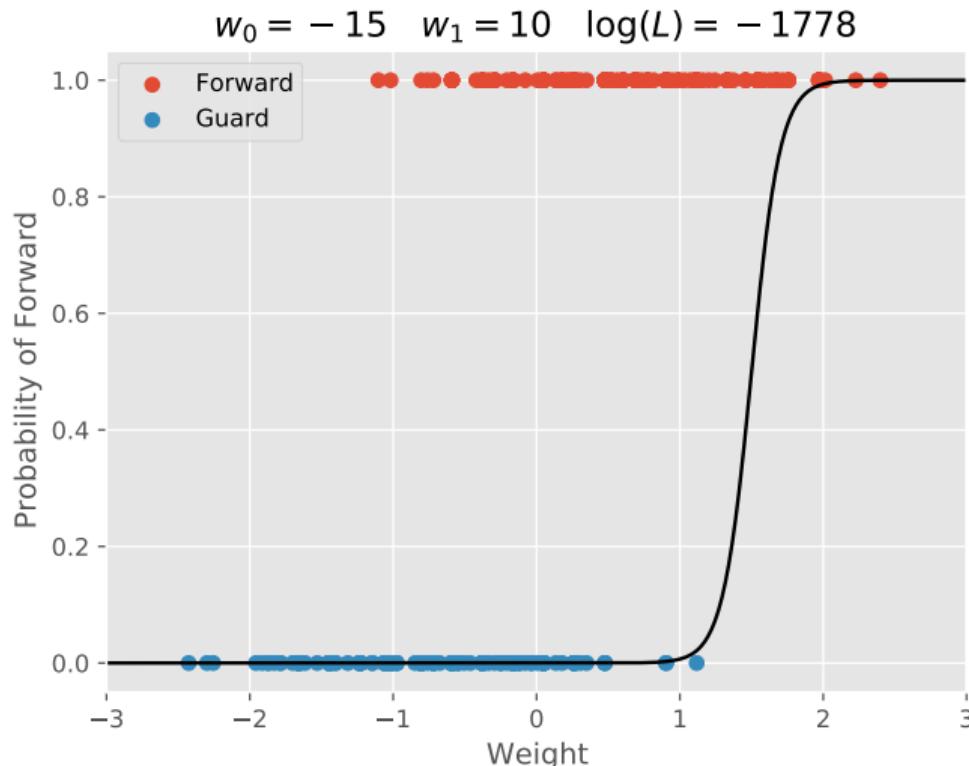
Guards vs. Forwards



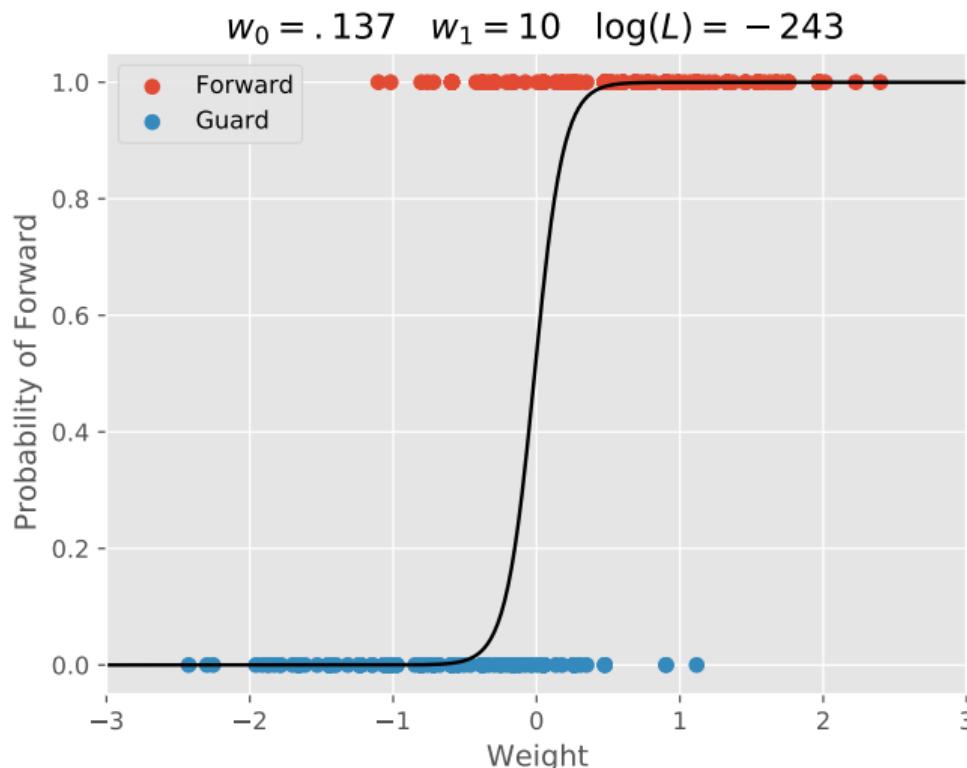
Prediction Rule Parameters

$$\begin{aligned} H_{\vec{w}}(\vec{x}) &= \sigma(\vec{w} \cdot \text{Aug}(\vec{x})) \\ &= \sigma(w_0 + w_1 \times \text{Weight}) \end{aligned}$$

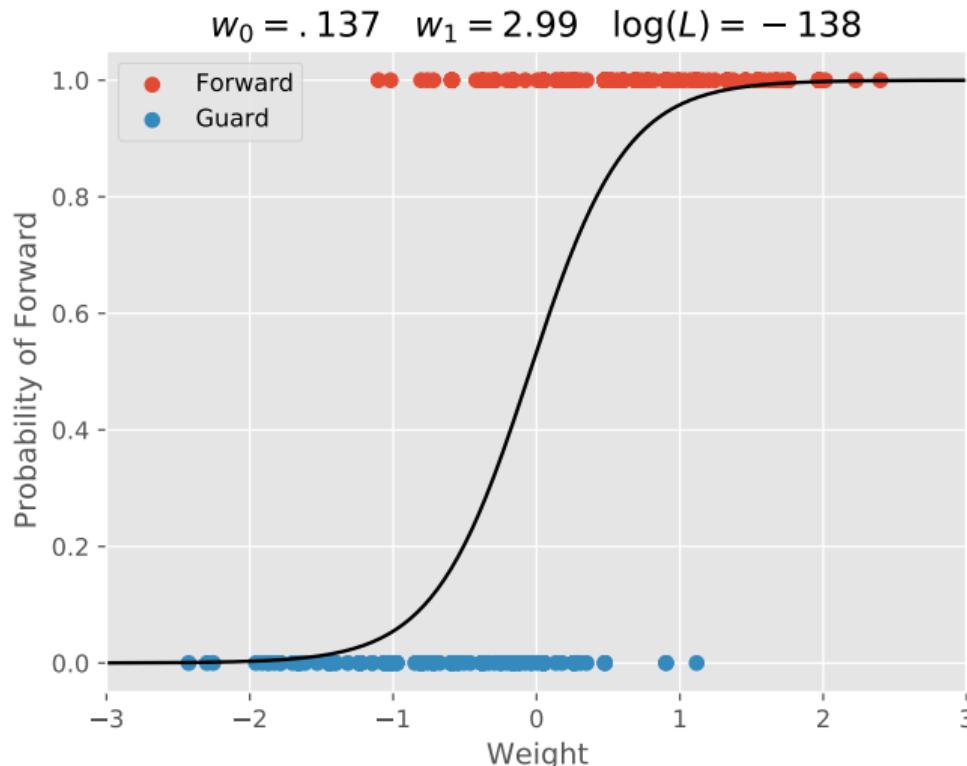
Prediction Rule Parameters



Prediction Rule Parameters



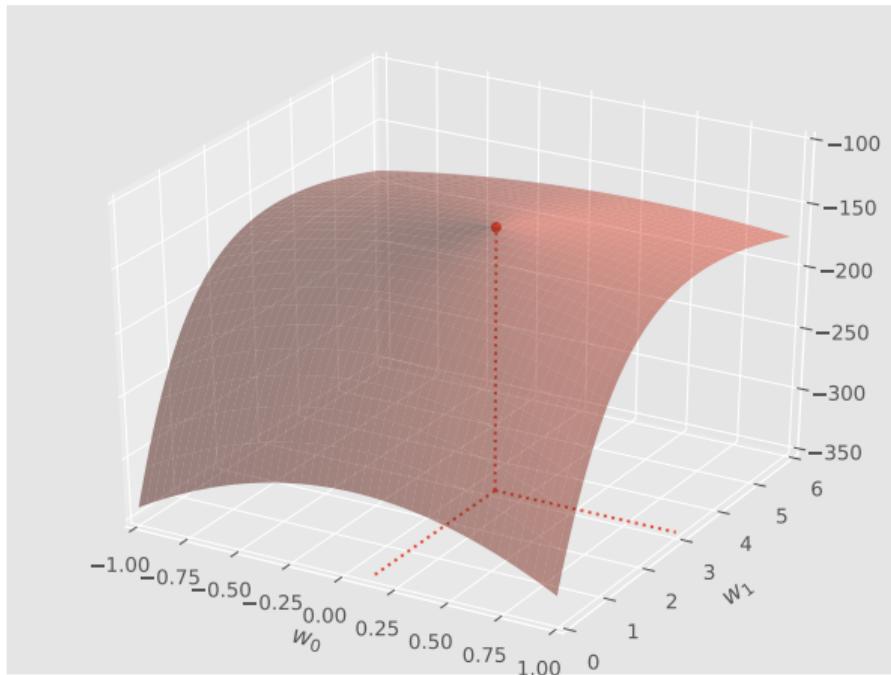
Prediction Rule Parameters



Learning

- ▶ Goal: find \vec{w} maximizing $f(\vec{w}) = \log L(\vec{w})$.

The Log Likelihood

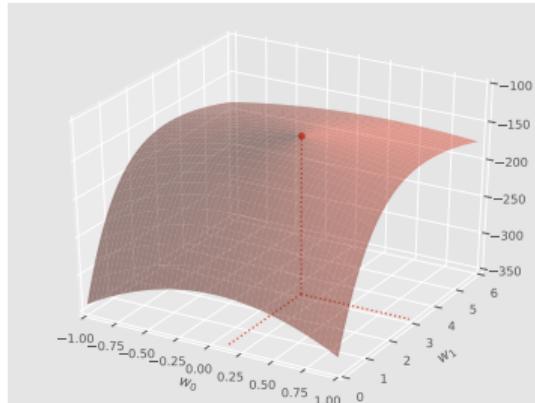


Maximizing

- ▶ Try setting gradient to zero, solving:
 - ▶ $f(\vec{w}) = - \sum_{i=1}^n \log(1 + \exp(-y_i \vec{w} \cdot \vec{x}^{(i)}))$

Meaning of the Gradient

- ▶ Meaning of **differentiable**: locally, f looks linear.
- ▶ $\nabla f(\vec{w})$ is a function; it returns a vector pointing in direction of steepest ascent.



Gradient Ascent

- ▶ Pick α to be a positive number.
 - ▶ It is the **learning rate**.
- ▶ Pick a starting guess, $\vec{w}^{(0)}$.
- ▶ On step i , update $\vec{w}^{(i)} = \vec{w}^{(i-1)} + \alpha \cdot \nabla f(\vec{w}^{(i-1)})$
- ▶ Repeat until convergence
 - ▶ when \vec{w} doesn't change much
 - ▶ equivalently, when $\|\nabla f(\vec{w}^{(i)})\|$ is small

Gradient Ascent for Logistic Regression

- ▶ Recall:

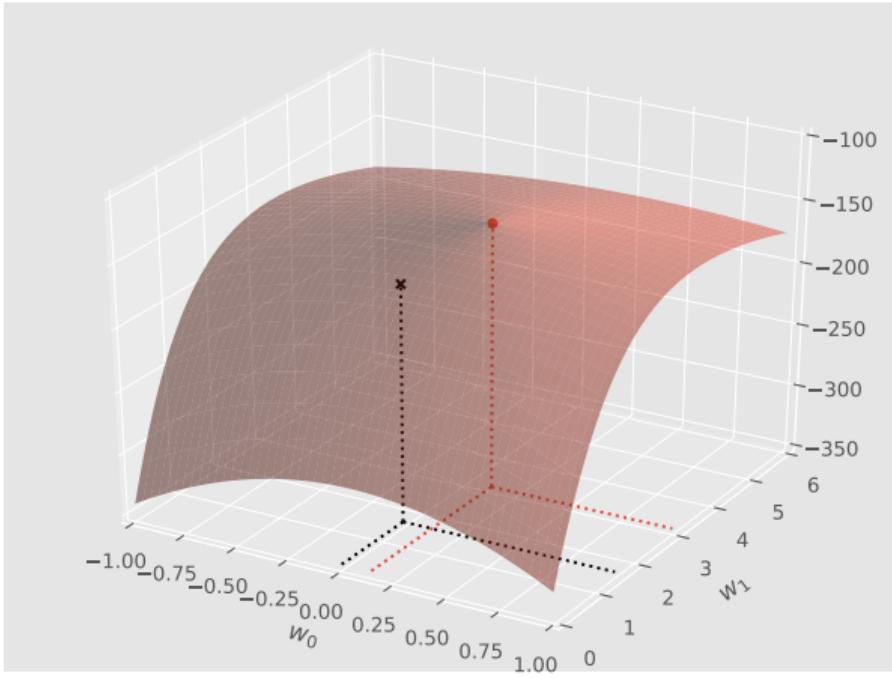
$$\nabla f(\vec{w}) = \sum_{k=1}^n y_k \vec{x}^{(k)} \frac{e^{-y_k \vec{w} \cdot \vec{x}^{(k)}}}{1 + e^{-y_k \vec{w} \cdot \vec{x}^{(k)}}} = \sum_{k=1}^n y_k \vec{x}^{(k)} \frac{1}{1 + e^{y_k \vec{w} \cdot \vec{x}^{(k)}}}$$

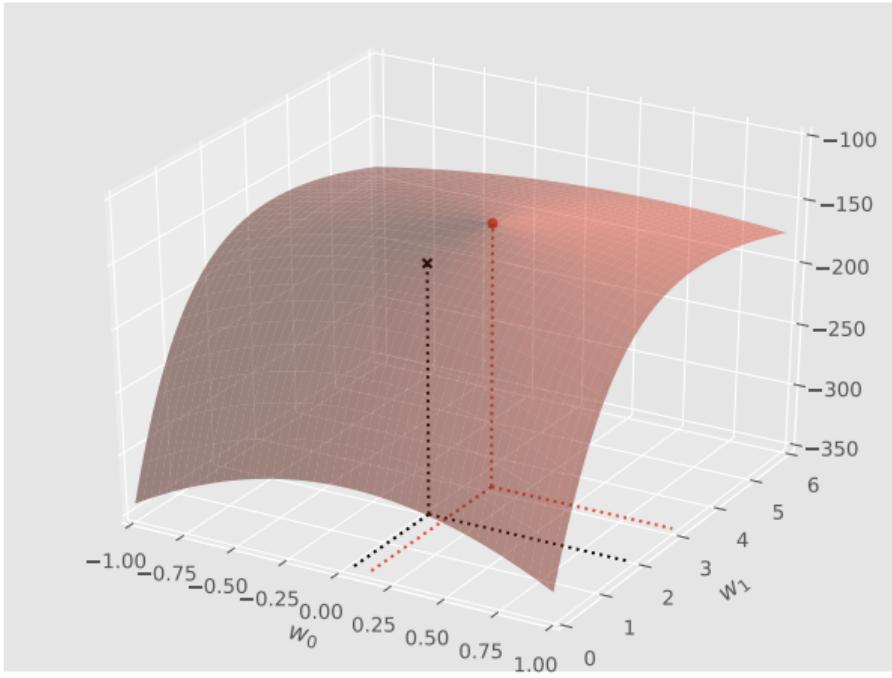
- ▶ Can show: $\nabla f(\vec{w}) = \sum_{k=1}^n y_k \vec{x}^{(k)} H_{\vec{w}}(-y_k \vec{x}^{(k)})$

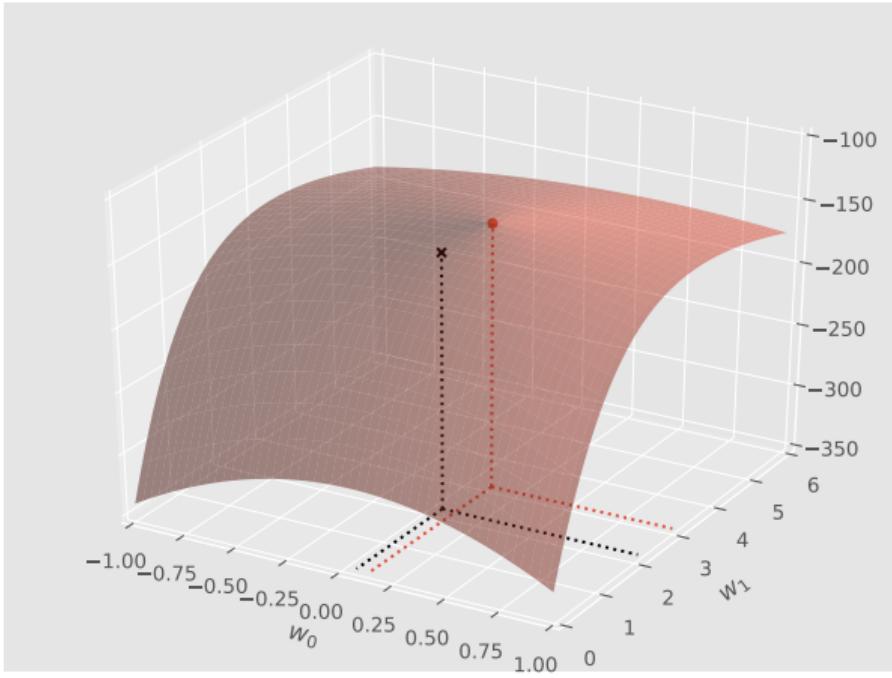
Gradient Ascent for Logistic Regression

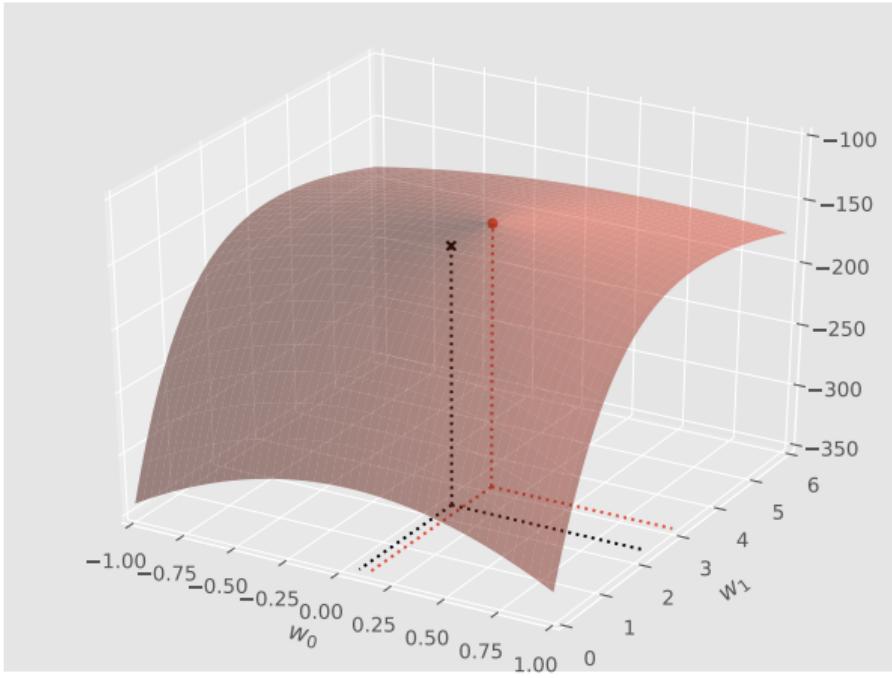
- ▶ On step i , update

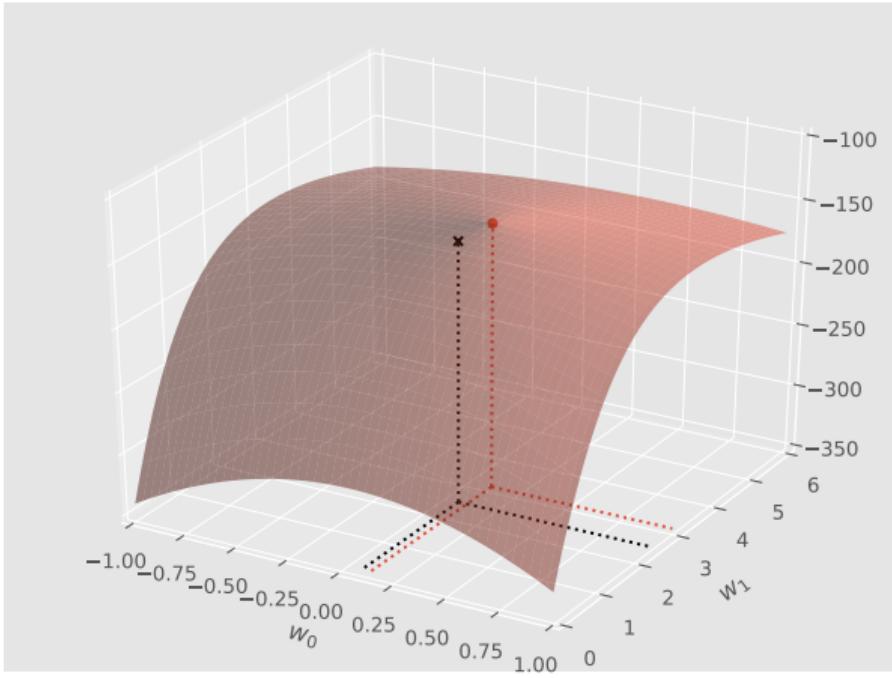
$$\vec{W}^{(i)} = \vec{W}^{(i-1)} + \alpha \cdot \sum_{k=1}^n y_k \vec{x}^{(k)} H_{\vec{W}^{(i-1)}}(-y_k \vec{x}^{(k)})$$

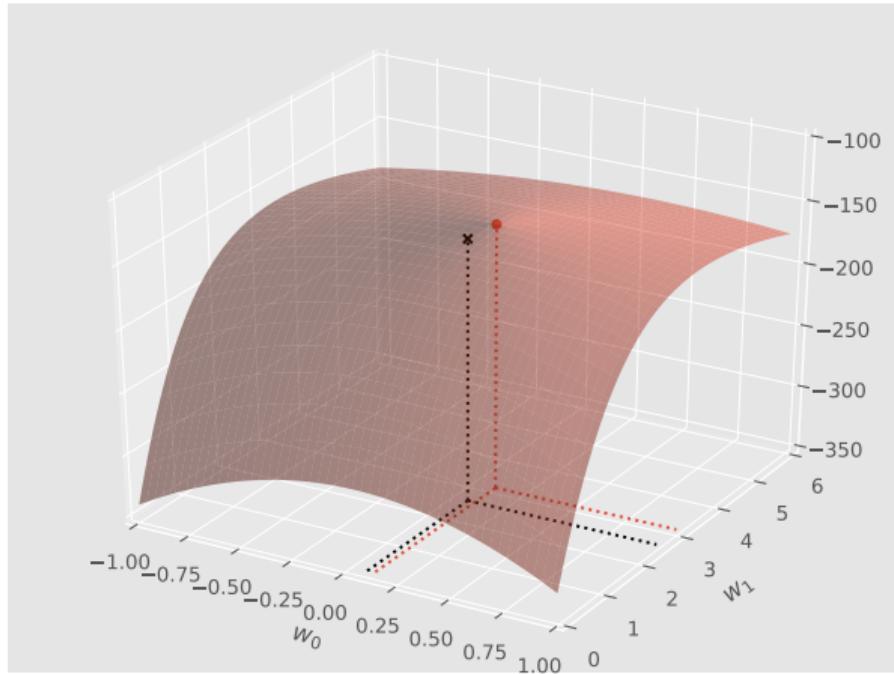


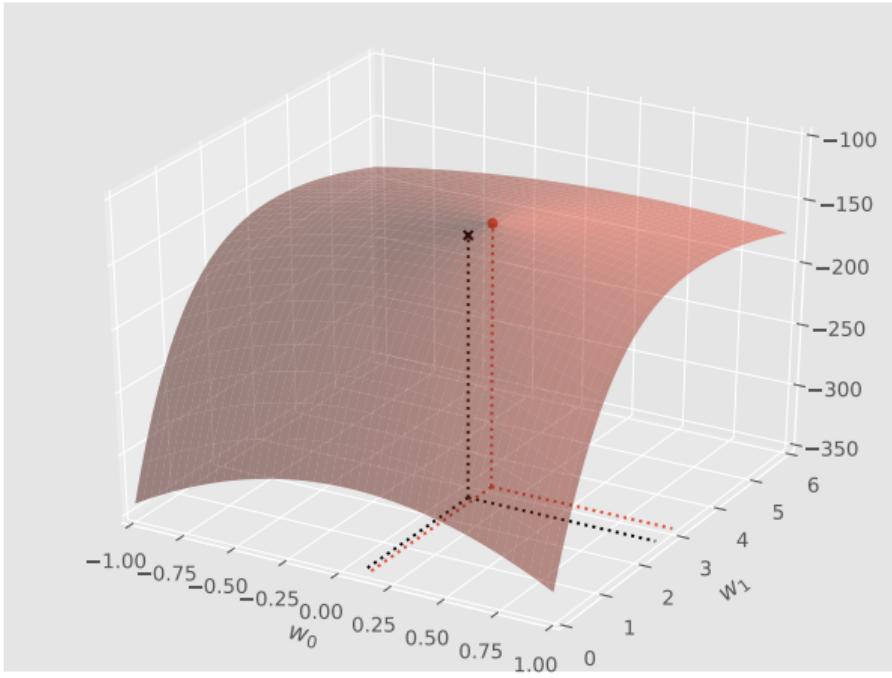


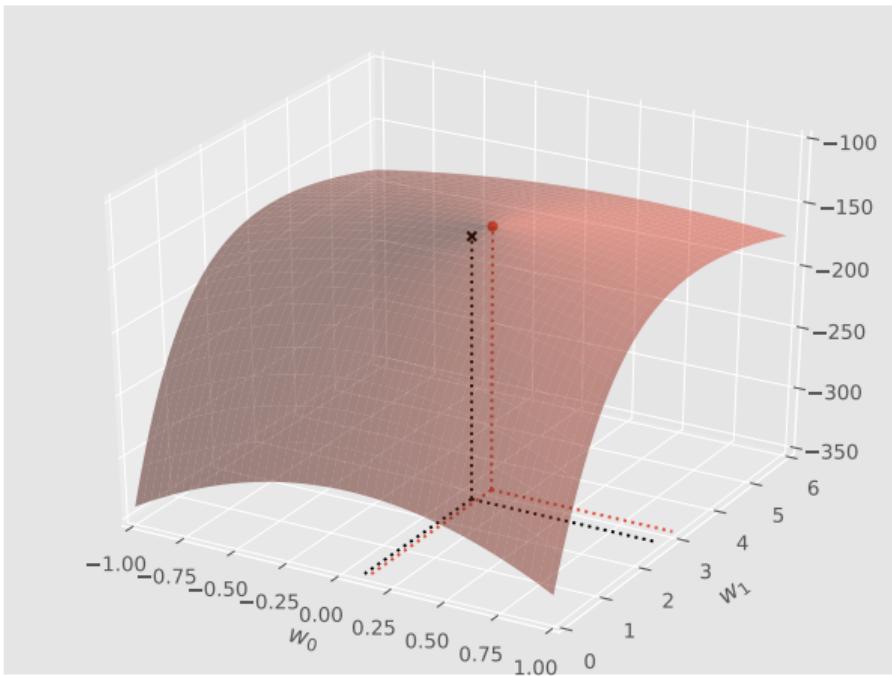


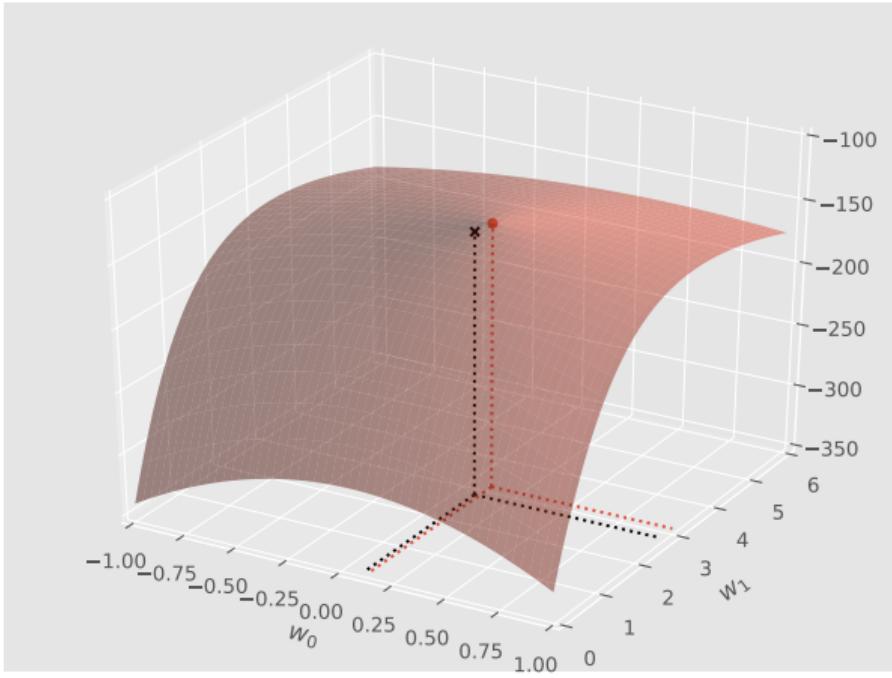


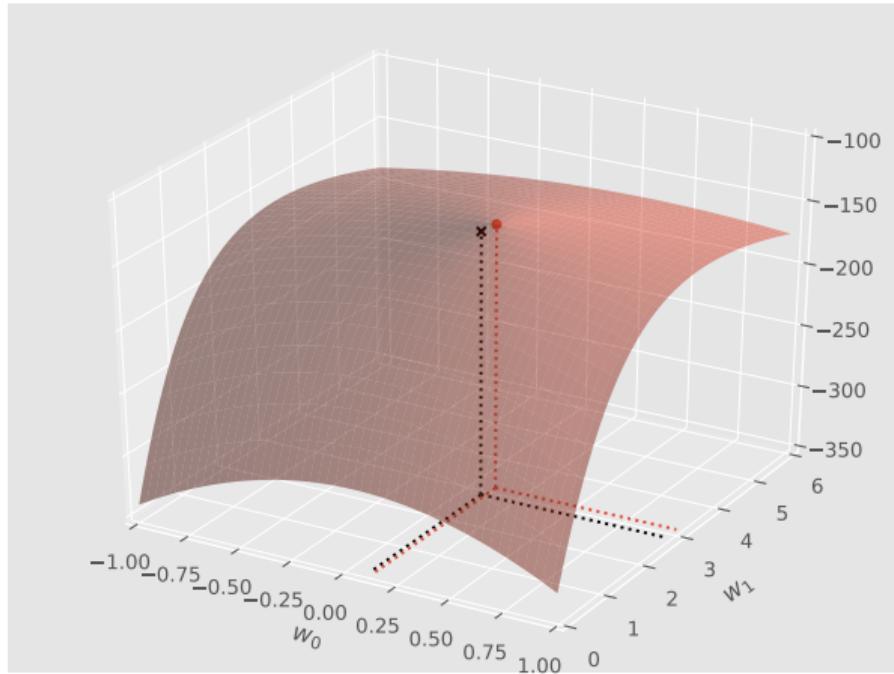


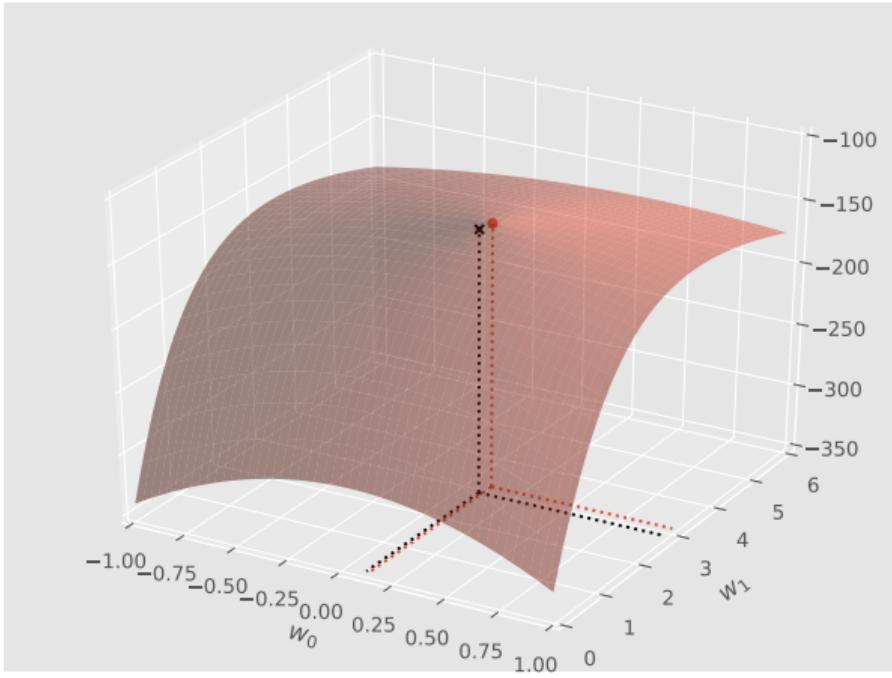


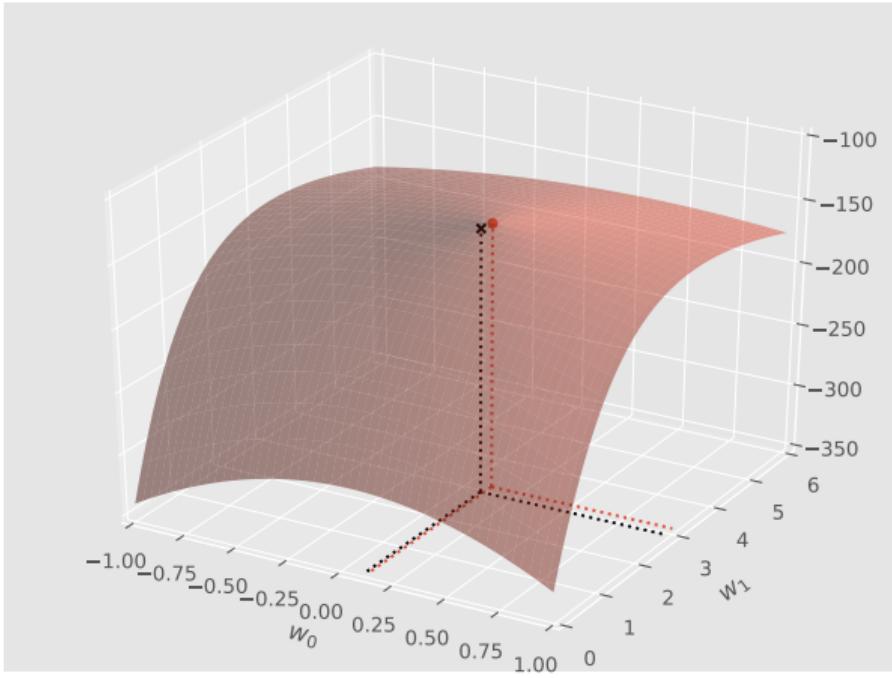


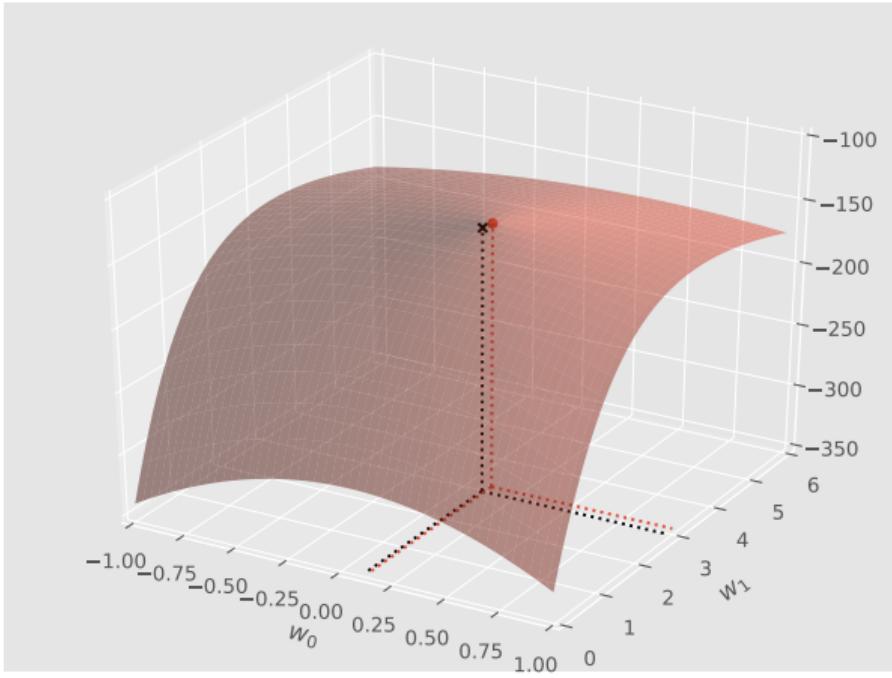


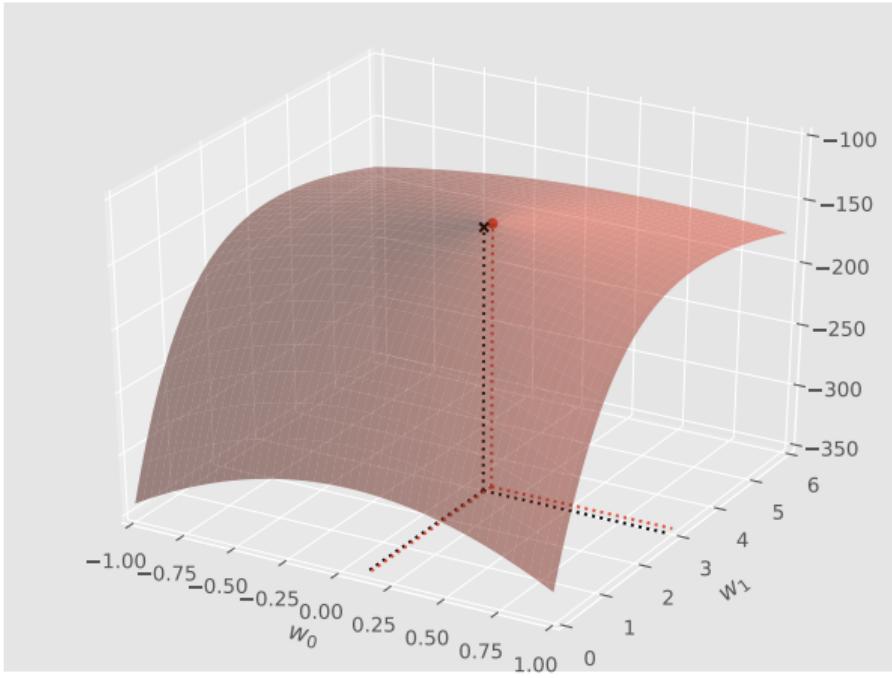


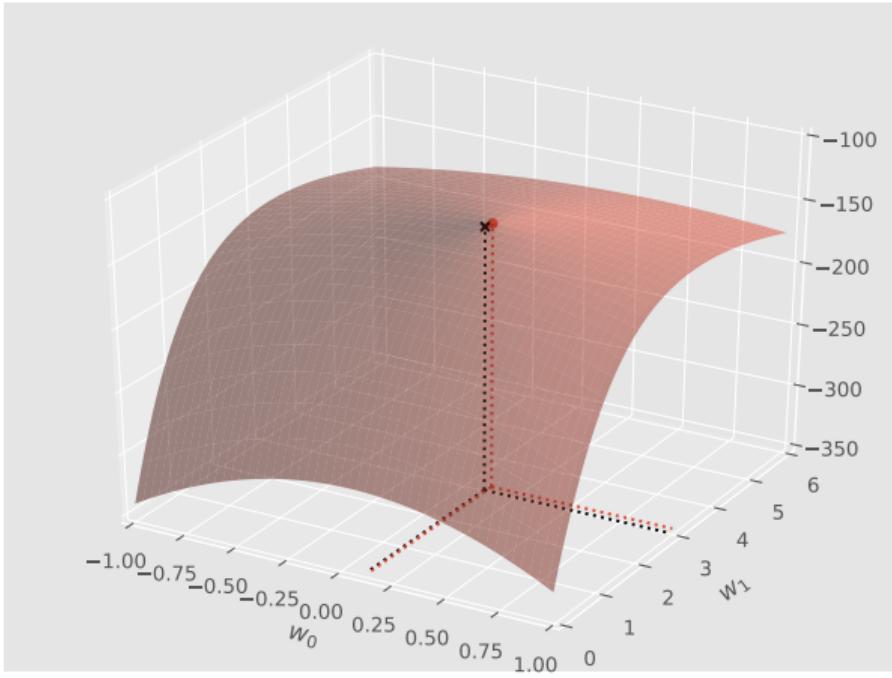


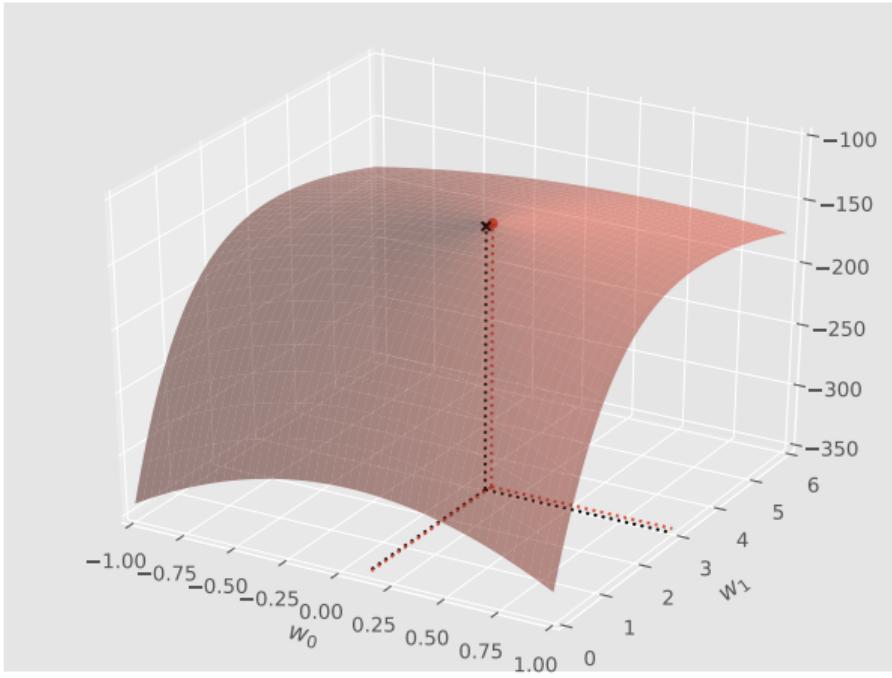


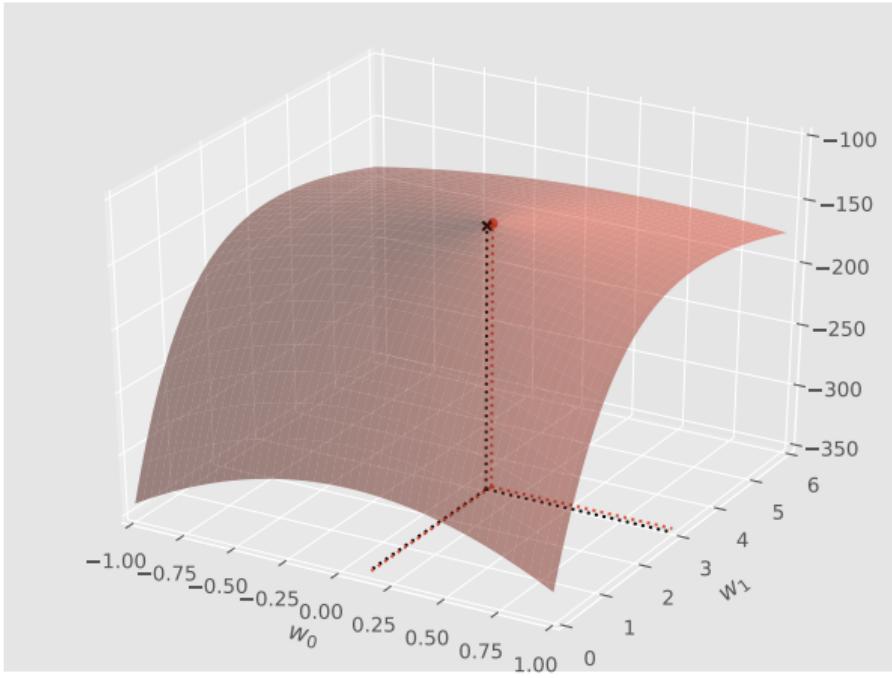


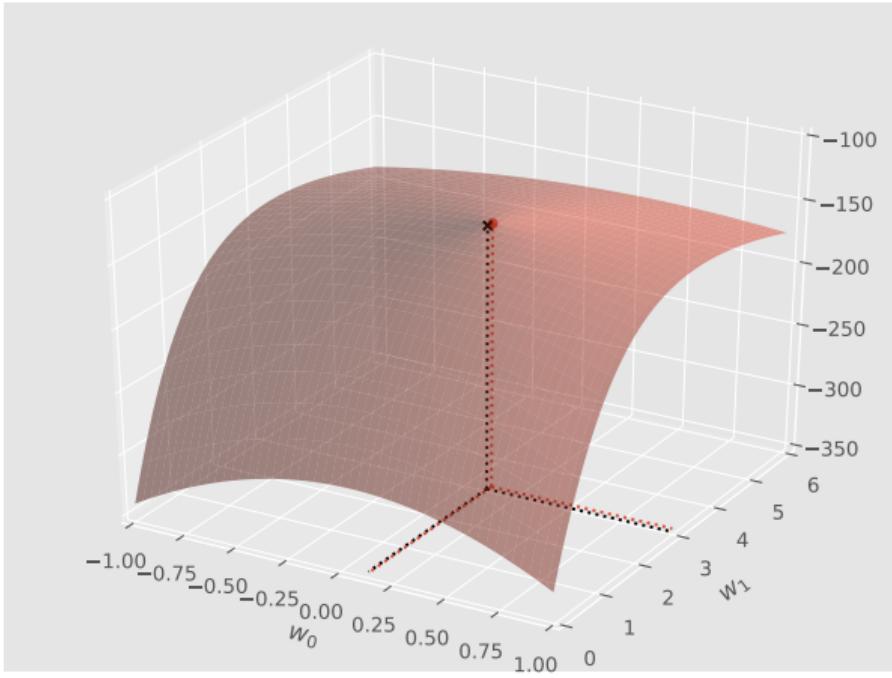


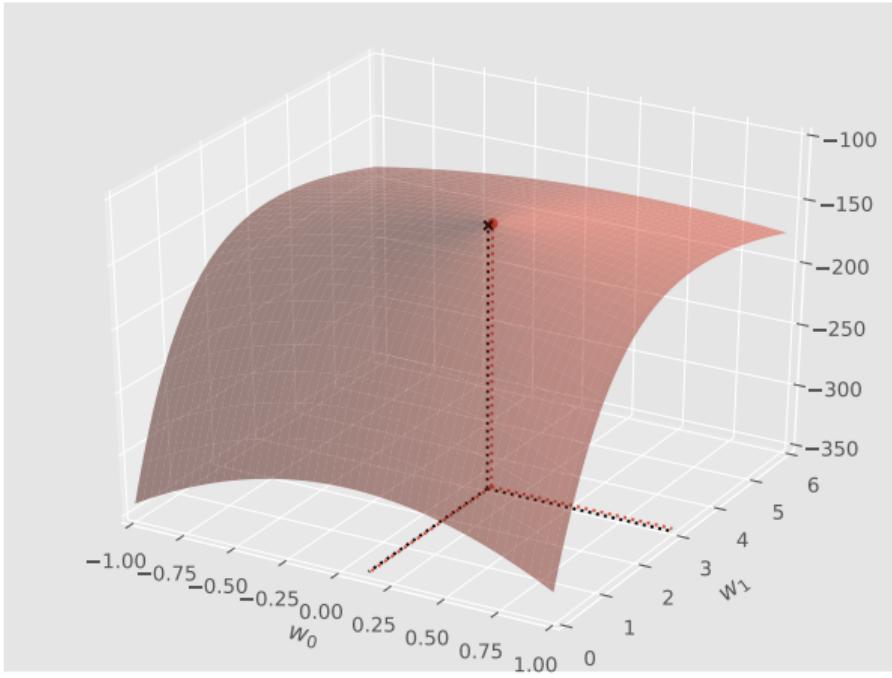


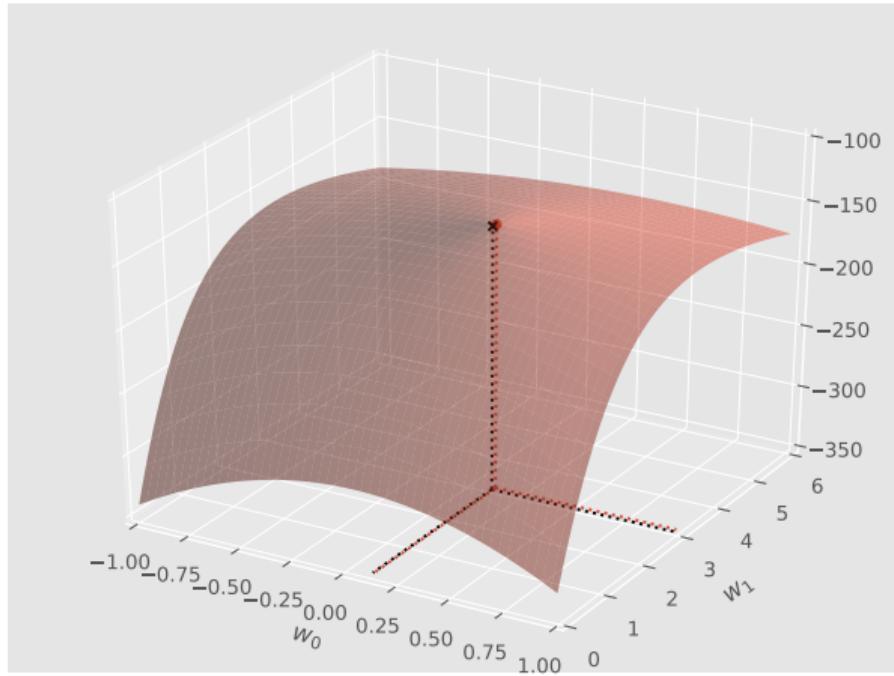












Gradient Descent

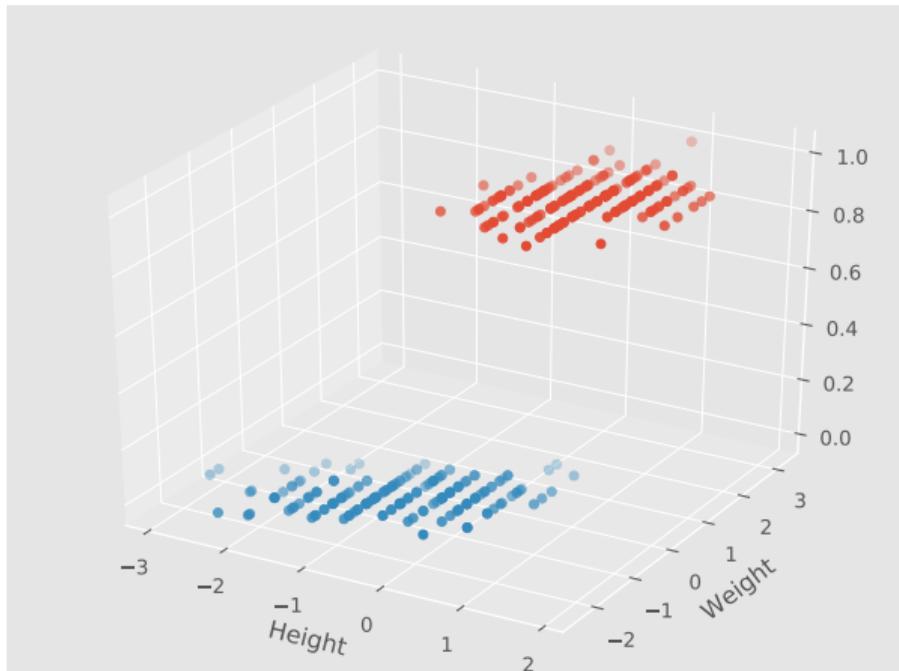
- ▶ Pick α to be a positive number.
 - ▶ It is the **learning rate**.
- ▶ Pick a starting guess, $\vec{w}^{(0)}$.
- ▶ On step i , update $\vec{w}^{(i)} = \vec{w}^{(i-1)} - \alpha \cdot \nabla f(\vec{w}^{(i-1)})$
- ▶ Repeat until convergence
 - ▶ when \vec{w} doesn't change much
 - ▶ equivalently, when $\|\nabla f(\vec{w}^{(i)})\|$ is small

```
def gradient_descent(gradient, w, alpha, tol=1e-12):
    """Minimize using gradient descent."""
    while True:
        w_next = w - alpha * gradient(x)
        if np.linalg.norm(w_next - w) < tol:
            break
        w = w_next
    return w
```

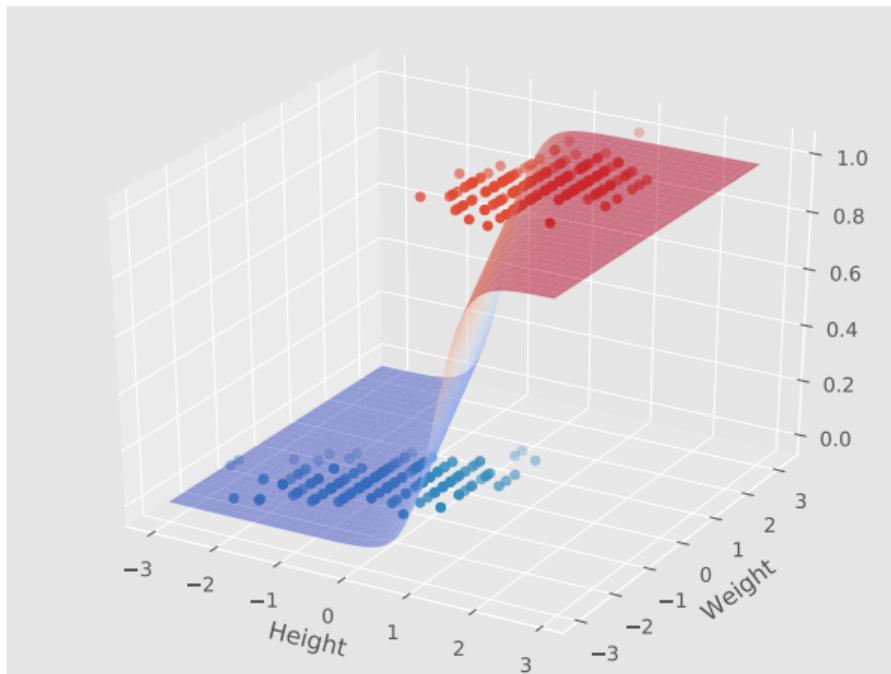
Adding Another Feature

- ▶ Use weight *and* height to predict position.
- ▶ Now $\text{Aug}(\vec{x}) \in \mathbb{R}^3$ and $\vec{w} \in \mathbb{R}^3$.

The Data



After Gradient Ascent



Making Classifications

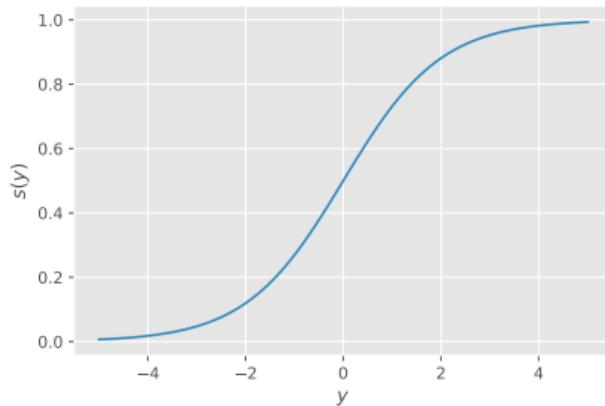
- ▶ Logistic regression predicts a probability:

$$H_{\vec{w}}(\vec{x}) = \sigma(\vec{w} \cdot \vec{x})$$

- ▶ Can turn into classification in two ways.

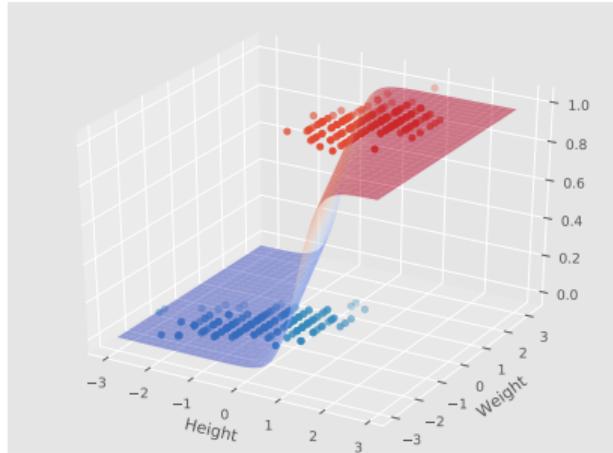
Approach 1

- ▶ If $H_{\vec{w}}(\vec{x}) > 0.5$, predict class 1; else predict class -1.
- ▶ Equivalently, predict class 1 if $\vec{x} \cdot \vec{w} > 0$.



Approach 2

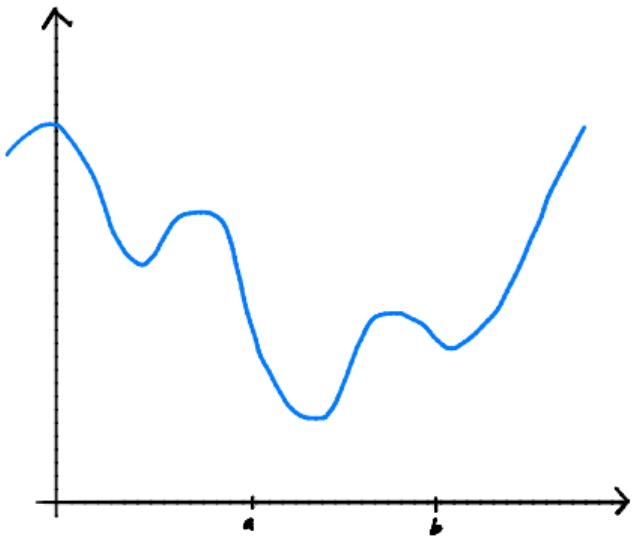
- ▶ More generally, predict class 1 if $H_{\vec{w}}(\vec{x}) > \tau$
- ▶ Equivalently, predict class 1 if $\vec{x} \cdot \vec{w} > t$
- ▶ How to pick τ/t ? Cross-validation!



CSE 151A

Intro to Machine Learning

Lecture 09 – Part 03
Demo: Heart Disease
Dataset



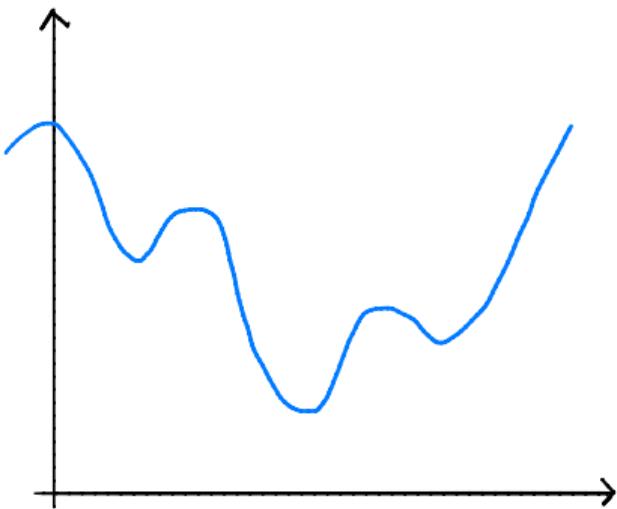
CSE 151A
Intro to Machine Learning

Lecture 10 – Part 01
Convexity in 1-d

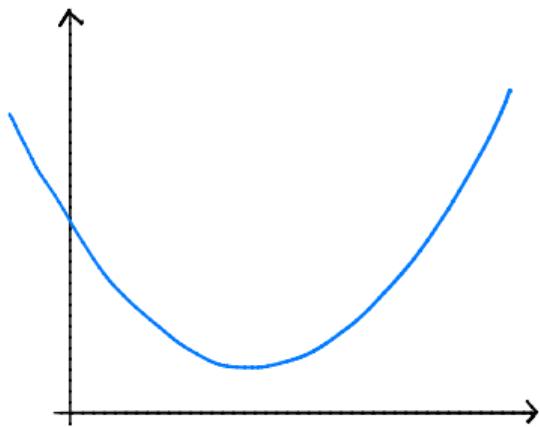
Today

When is gradient descent guaranteed to work?

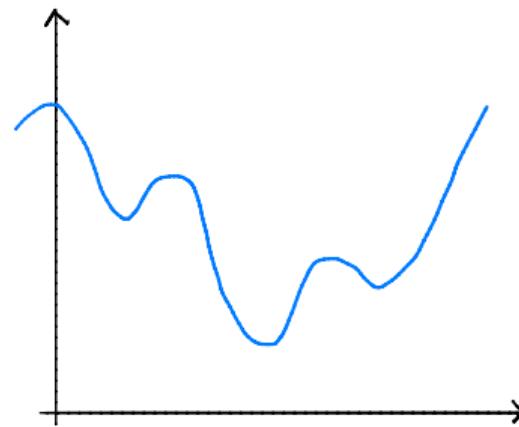
Not here...



Convex Functions



Convex



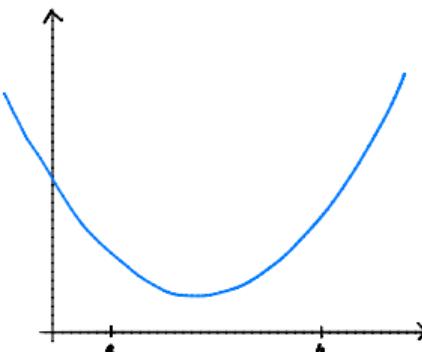
Non-convex

Convexity: Definition

- ▶ f is **convex** if for **every** a, b the line segment between

$$(a, f(a)) \quad \text{and} \quad (b, f(b))$$

does not go below the plot of f .

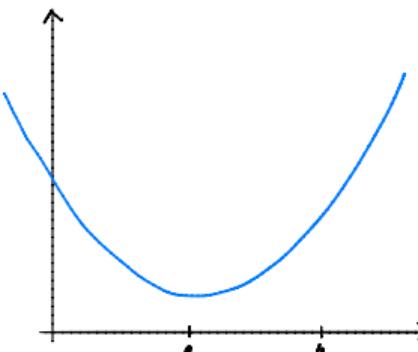


Convexity: Definition

- ▶ f is **convex** if for **every** a, b the line segment between

$$(a, f(a)) \quad \text{and} \quad (b, f(b))$$

does not go below the plot of f .

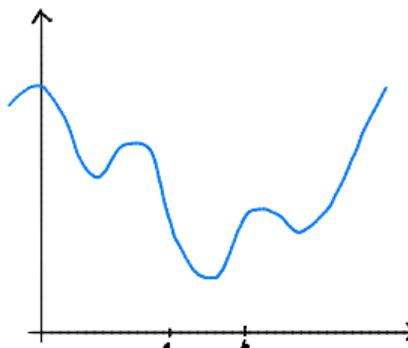


Convexity: Definition

- ▶ f is **convex** if for **every** a, b the line segment between

$$(a, f(a)) \quad \text{and} \quad (b, f(b))$$

does not go below the plot of f .

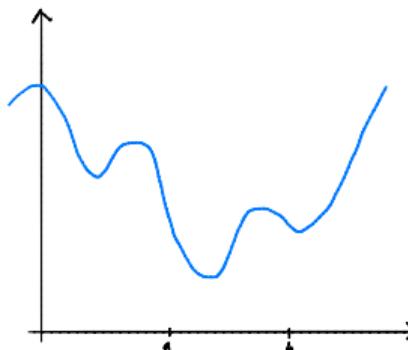


Convexity: Definition

- ▶ f is **convex** if for **every** a, b the line segment between

$$(a, f(a)) \quad \text{and} \quad (b, f(b))$$

does not go below the plot of f .



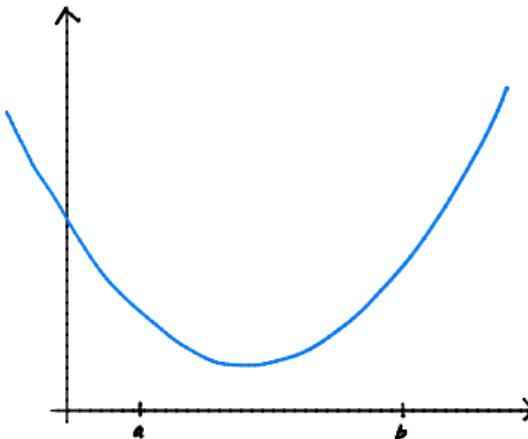
Other Terms

- ▶ If a function is not convex, it is **non-convex**.
- ▶ **Strictly convex**: the line lies strictly above curve.
- ▶ **Concave**: the line lies on or below curve.

Convexity: Formal Definition

- ▶ A function $f : \mathbb{R} \rightarrow \mathbb{R}$ is **convex** if for every choice of $a, b \in \mathbb{R}$ and $t \in [0, 1]$:

$$(1 - t)f(a) + tf(b) \geq f((1 - t)a + tb).$$

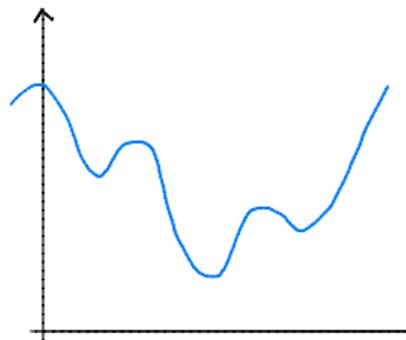
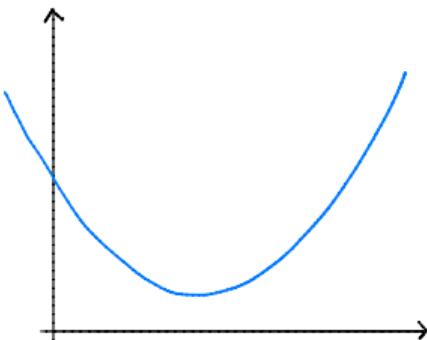


Example

Is $f(x) = |x|$ convex?

Another View: Second Derivatives

- ▶ If $\frac{d^2f}{dx^2}(x) \geq 0$ for all x , then f is convex.
- ▶ Example: $f(x) = x^4$ is convex.
- ▶ **Warning!** Only works if f is twice differentiable!

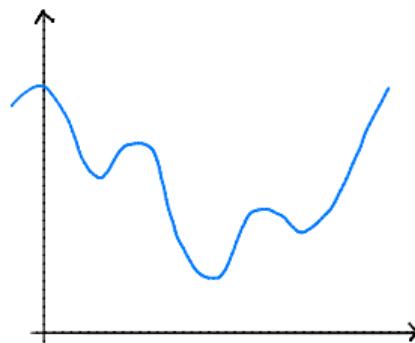
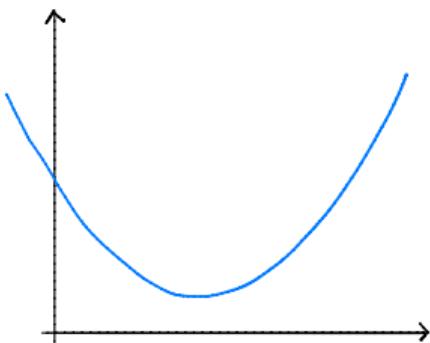


Another View: Second Derivatives

- ▶ “Best” straight line at x_0 :
 - ▶ $h_1(z) = f'(x_0) \cdot z + b$
- ▶ “Best” parabola at x_0 :
 - ▶ At x_0 , f looks like $h_2(z) = \frac{1}{2}f''(x_0) \cdot z^2 + f'(x_0)z + c$
 - ▶ Possibilities: upward-facing, downward-facing.

Convexity and Parabolas

- ▶ Convex if for **every** x_0 , parabola is upward-facing.
 - ▶ That is, $f''(x_0) \geq 0$.



Convexity and Gradient Descent

- ▶ Convex functions are (relatively) easy to optimize.
- ▶ **Theorem:** if $R(x)$ is convex and differentiable¹² then gradient descent converges to a **global optimum** of R provided that the step size is small enough³.

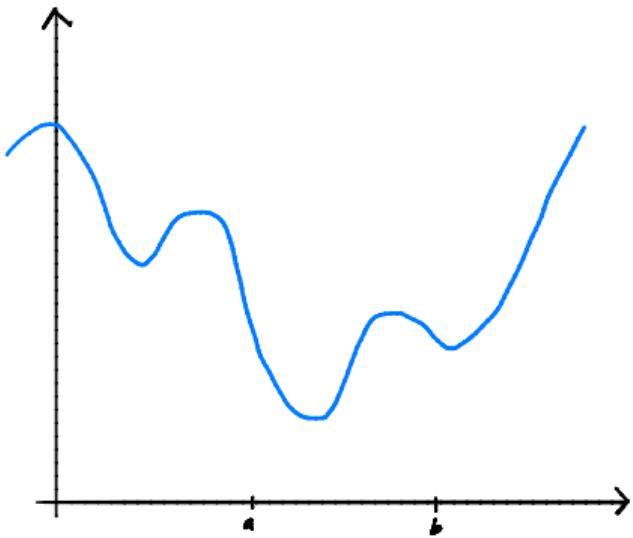
¹and its derivative is not too wild

²actually, a modified GD works on non-differentiable functions

³step size related to steepness.

Nonconvexity and Gradient Descent

- ▶ Nonconvex functions are (relatively) hard to optimize.
- ▶ Gradient descent can still be useful.
- ▶ But not guaranteed to converge to a global minimum.



CSE 151A
Intro to Machine Learning

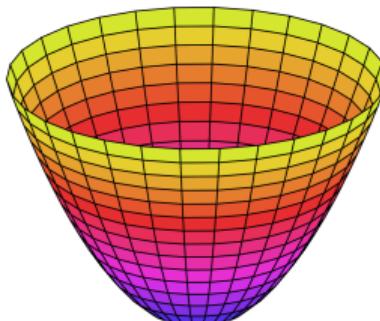
Lecture 10 – Part 02
**Convexity in Many
Dimensions**

Convexity: Definition

- ▶ $f(\vec{x})$ is **convex** if for **every** \vec{a}, \vec{b} the line segment between

$$(\vec{a}, f(\vec{a})) \quad \text{and} \quad (\vec{b}, f(\vec{b}))$$

does not go below the plot of f .



Convexity: Formal Definition

- ▶ A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is **convex** if for every choice of $\vec{a}, \vec{b} \in \mathbb{R}^d$ and $t \in [0, 1]$:

$$(1 - t)f(\vec{a}) + tf(\vec{b}) \geq f((1 - t)\vec{a} + t\vec{b}).$$

The Second Derivative Test

- ▶ For 1-d functions, convex if second derivative ≥ 0 .
- ▶ For 2-d functions, convex if ???

Second Derivatives in 2-d

- ▶ In 2-d, there are 4 second derivatives of $f(\vec{x})$:
 - ▶ $\frac{\partial f^2}{\partial x_1^2}, \frac{\partial f^2}{\partial x_2^2}, \frac{\partial f^2}{\partial x_1 x_2}, \frac{\partial f^2}{\partial x_2 x_1}$

Convexity in 2-d

- ▶ “Best” quadratic function approximating f at \vec{x} :

$$\begin{aligned} h_2(z_1, z_2) &= az_1^2 + bz_2^2 + cz_1z_2 + \dots \\ &= \frac{1}{2} \frac{\partial f^2}{\partial x_1^2}(\vec{x}) \cdot z_1 + \frac{1}{2} \frac{\partial f^2}{\partial x_2^2}(\vec{x}) \cdot z_2 + \frac{\partial f^2}{\partial x_1 x_2}(\vec{x}) \cdot z_1 z_2 + \dots \end{aligned}$$

- ▶ a, b, c determine rough shape. Possibilities:
 - ▶ Upward-facing bowl.
 - ▶ Downward-facing bowl.
 - ▶ “Saddle”

Convexity in 2-d

- Convex if at any \vec{x} , for any z_1, z_2 :

$$\frac{1}{2} \frac{\partial f^2}{\partial x_1^2}(\vec{x}) \cdot z_1 + \frac{1}{2} \frac{\partial f^2}{\partial x_2^2}(\vec{x}) \cdot z_2 + \frac{\partial f^2}{\partial x_1 x_2}(\vec{x}) \cdot z_1 z_2 \geq 0$$

The Hessian Matrix

- ▶ Create the **Hessian** matrix of second derivatives:

$$H(\vec{x}) = \begin{pmatrix} \frac{\partial f^2}{\partial x_1^2}(\vec{x}) & \frac{\partial f^2}{\partial x_1 x_2}(\vec{x}) \\ \frac{\partial f^2}{\partial x_2 x_1}(\vec{x}) & \frac{\partial f^2}{\partial x_2^2}(\vec{x}) \end{pmatrix}$$

In General

- If $f : \mathbb{R}^d \rightarrow \mathbb{R}$, the **Hessian** at \vec{x} is:

$$H(\vec{x}) = \begin{pmatrix} \frac{\partial f^2}{\partial x_1^2}(\vec{x}) & \frac{\partial f^2}{\partial x_1 x_2}(\vec{x}) & \dots & \frac{\partial f^2}{\partial x_1 x_d}(\vec{x}) \\ \frac{\partial f^2}{\partial x_2 x_1}(\vec{x}) & \frac{\partial f^2}{\partial x_2^2}(\vec{x}) & \dots & \frac{\partial f^2}{\partial x_2 x_d}(\vec{x}) \\ \dots & \dots & \dots & \dots \\ \frac{\partial f^2}{\partial x_d x_1}(\vec{x}) & \frac{\partial f^2}{\partial x_d x_2}(\vec{x}) & \dots & \frac{\partial f^2}{\partial x_d^2}(\vec{x}) \end{pmatrix}$$

Observations

- ▶ H is square.
- ▶ H is symmetric.

Convexity in 2-d

- Convex if at any \vec{x} , for any z_1, z_2 :

$$\frac{1}{2} \frac{\partial f^2}{\partial x_1^2}(\vec{x}) \cdot z_1 + \frac{1}{2} \frac{\partial f^2}{\partial x_2^2}(\vec{x}) \cdot z_2 + \frac{\partial f^2}{\partial x_1 x_2}(\vec{x}) \cdot z_1 z_2 \geq 0$$

- Equivalently, convex if for any \vec{x} and any \vec{z} :

$$\vec{z}^T H(\vec{x}) \vec{z} \geq 0$$

Positive Semi-Definite

- ▶ A square, $d \times d$ symmetric matrix X is **positive semi-definite** (PSD) if for any \vec{u} :

$$\vec{u}^T X \vec{u} \geq 0$$

The Second Derivative Test

- ▶ A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is **convex** if for any $\vec{x} \in \mathbb{R}^d$, the Hessian matrix $H(\vec{x})$ is positive semi-definite.

But wait...

- ▶ How can we tell if a matrix is positive semi-definite?

Example

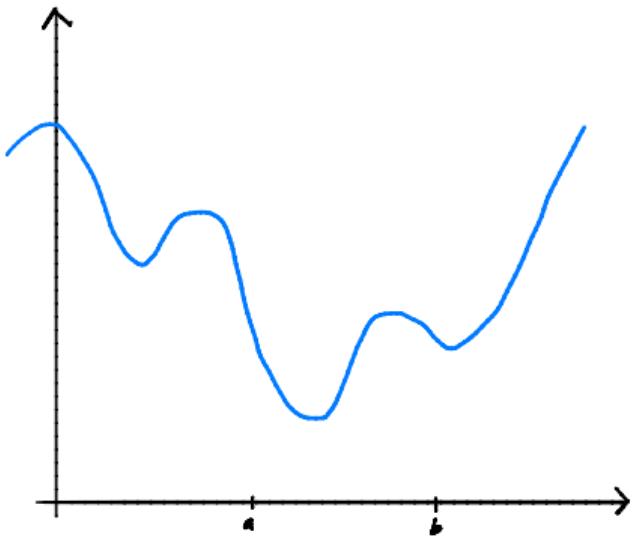
$$M = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Example

$$M = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$$

Example

Is $f(x, y) = x^2 + 4xy + y^2$ convex?



CSE 151A
Intro to Machine Learning

Lecture 10 – Part 03
Convex Loss Functions

Convexity and Gradient Descent

- ▶ Convex functions are (relatively) easy to optimize.
- ▶ **Theorem:** if $R(\vec{w})$ is convex and differentiable⁴⁵ then gradient descent converges to a **global optimum** of R provided that the step size is small enough⁶.

⁴and its gradient is not too wild

⁵actually, a modified GD works on non-differentiable functions

⁶step size related to steepness.

Example

- ▶ Recall the Mean Squared Error:

$$R(\vec{w}) = \frac{1}{n} \sum_{i=1}^n (\vec{x}^{(i)} \cdot \vec{w} - y_i)^2$$

- ▶ Is this convex?

Mean Squared Error

$$R(\vec{w}) = \frac{1}{n} \sum_{i=1}^n (\vec{x}^{(i)} \cdot \vec{w} - y_i)^2$$

A Useful Theorem

- ▶ A square, symmetric matrix M is PSD if and only if M can be written as UU^T for some matrix U .

$$\begin{aligned}\vec{x}^T M \vec{x} &= \vec{x}^T U U^T \vec{x} \\&= (\vec{x}^T U)(U^T \vec{x}) \\&= (U^T \vec{x})^T (U^T \vec{x}) \\&= \|U^T \vec{x}\|^2 \\&\geq 0\end{aligned}$$

Mean Squared Error

- ▶ The MSE is a convex function of \vec{w} .
- ▶ We had an explicit solution for the best \vec{w} :

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

- ▶ But we could also have used gradient descent.

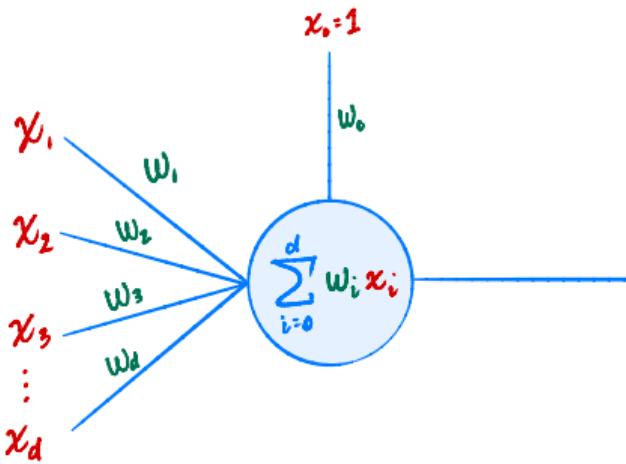
Logistic Regression

- ▶ The log-likelihood is **concave**.

$$\log \mathcal{L}(\vec{w}) = - \sum_{i=1}^n \log \left[1 + e^{-y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)})} \right]$$

Status Update

- ▶ We learned what it means for a function to be **convex**.
- ▶ Convex functions are (relatively) **easy** to optimize with gradient descent.
- ▶ We like **convex loss functions**, like the mean squared error.



CSE 151A

Intro to Machine Learning

Lecture 11 – Part 01

Stochastic Gradient Descent

Midterm 01 Scores Posted

- ▶ Scores were high!
- ▶ Remember: redemption is available on final.
- ▶ Pass v. no pass.
 - ▶ Still a lot of plus points left to earn.

Recall: Gradient Descent

- ▶ Suppose you are minimizing a function $R(\vec{w})$.
- ▶ Gradient descent: on step i , update

$$\vec{w}^{(i)} = \vec{w}^{(i-1)} - \alpha \cdot \nabla R(\vec{w}^{(i-1)})$$

A Small Problem with Big Data

- ▶ In ML, our functions involve data.
 - ▶ MSE:

$$R(\vec{w}) = \sum_{i=1}^n (\text{Aug}(\vec{x}^{(i)}) \cdot \vec{w} - y_i)^2$$

- ▶ Log Likelihood:

$$R(\vec{w}) = - \sum_{i=1}^n \log \left[1 + e^{-y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)})} \right]$$

- ▶ When n is large, gradient is **expensive** to compute.

Decomposability

- ▶ In ML, our functions are often **decomposable**:

$$R(\vec{w}) = \sum_{i=1}^n \ell(\vec{w}; \vec{x}^{(i)}, y_i)$$

- ▶ And so $\nabla R(\vec{w}) = \sum_{i=1}^n \nabla \ell(\vec{w}; \vec{x}^{(i)}, y_i)$

The Idea

- ▶ The **full gradient** is

$$\nabla R(\vec{w}) = \sum_{i=1}^n \nabla \ell(\vec{w}; \vec{x}^{(i)}, y_i)$$

- ▶ Instead, approximate with a **mini-batch**:

$$\nabla R(\vec{w}) \approx \sum_{i \in B} \nabla \ell(\vec{w}; \vec{x}^{(i)}, y_i)$$

Mini-Batch Stochastic Gradient Descent

To minimize a **decomposable** function

$$R(\vec{w}) = \sum_{i=1}^n \ell(\vec{w}; \vec{x}^{(i)}, y_i):$$

- ▶ Pick a starting guess, $\vec{w}^{(0)}$.
- ▶ On step k , randomly select a set B of $n' \leq n$ points, perform update:

$$\vec{w}^{(k)} = \vec{w}^{(k-1)} - \alpha \sum_{i \in B} \nabla \ell(\vec{w}^{(k-1)}; \vec{x}^{(i)}, y_i)$$

- ▶ Repeat until convergence

Stochastic Gradient Descent

To minimize a **decomposable** function

$$R(\vec{w}) = \sum_{i=1}^n \ell(\vec{w}; \vec{x}^{(i)}, y_i):$$

- ▶ Pick a starting guess, $\vec{w}^{(0)}$.
- ▶ Repeat until convergence:
 - ▶ Randomly shuffle the points.
 - ▶ Loop through all n points, one at a time, performing:

$$\vec{w}^{(k)} = \vec{w}^{(k-1)} - \alpha \nabla \ell(\vec{w}^{(k-1)}; \vec{x}^{(i)}, y_i)$$

Example: Logistic Regression

- ▶ **Prediction Rule:**

$$H_{\vec{w}}(\vec{x}) = \sigma(\vec{w} \cdot \vec{x})$$

- ▶ **Goal:** find \vec{w} maximizing log likelihood

$$\log \mathcal{L}(\vec{w}) = - \sum_{i=1}^n \log \left[1 + e^{-y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)})} \right]$$

Observation

- The log likelihood is **decomposable**:

$$\log \mathcal{L}(\vec{w}) = - \sum_{i=1}^n \log \left[1 + e^{-y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)})} \right]$$

$$= \sum_{i=1}^n \ell(\vec{w}; \vec{x}^{(i)}, y_i)$$

where $\ell(\vec{w}; \vec{x}^{(i)}, y_i) = - \log \left[1 + e^{-y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)})} \right]$

Why SGD?

- ▶ SGD is computationally cheaper than GD.
- ▶ One step of GD with **full gradient**: $\Theta(nd)$.
- ▶ One step of SGD: $\Theta(d)$.

The Good and the Bad with SGD

- ▶ **Good:** fast(er) per step, guaranteed to converge if R convex, generalization.
- ▶ **Bad:** single point might result in noisy gradient.

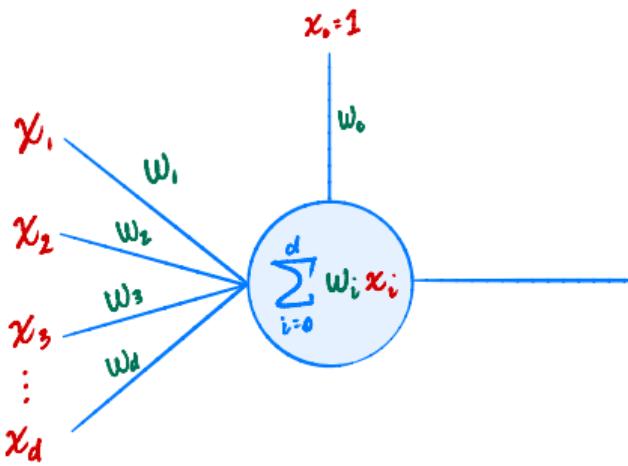
Example

More: Learning Rates

- ▶ How do we pick α ?
- ▶ One strategy: line search.
- ▶ Decay: α_t instead of α .

More: Early Stopping

- ▶ To avoid overfitting, don't run GD/SGD completely.
- ▶ Stop early, when validation error starts to go up.

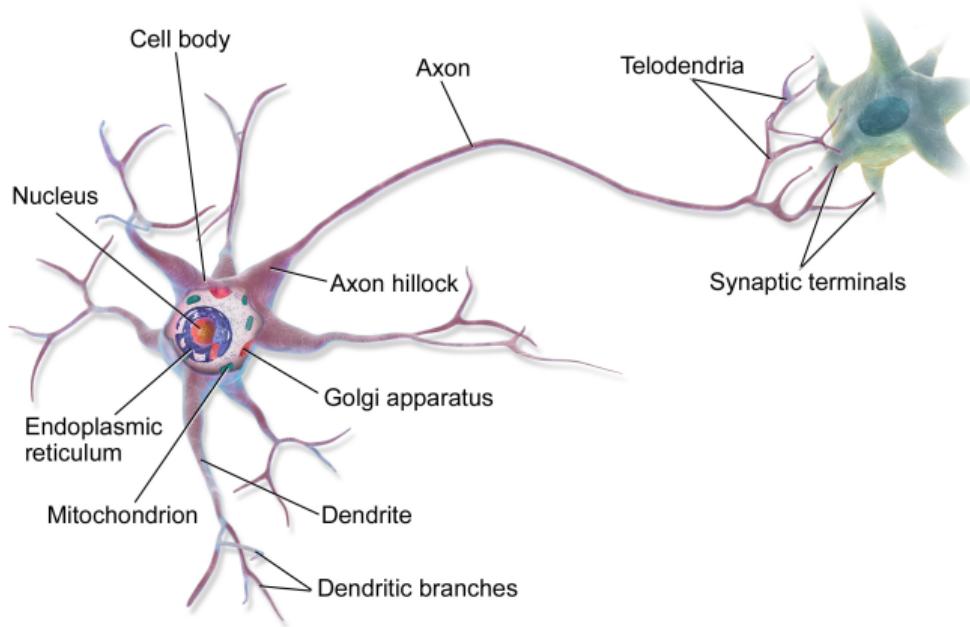


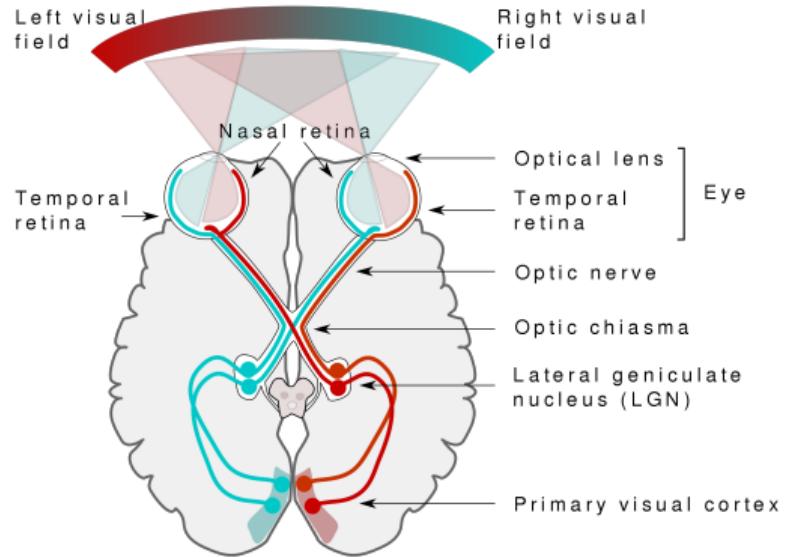
CSE 151A

Intro to Machine Learning

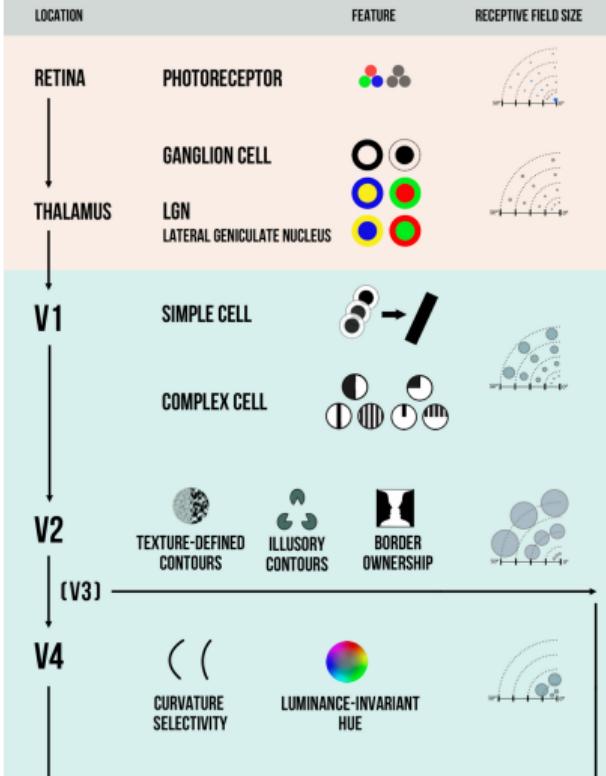
Lecture 11 – Part 02

Perceptrons





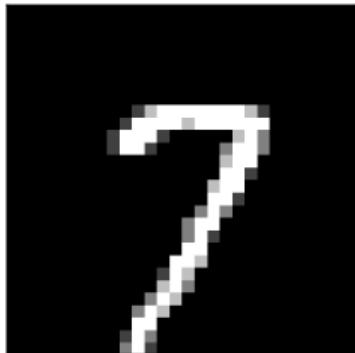
DEEP HIERARCHIES IN THE VISUAL SYSTEM



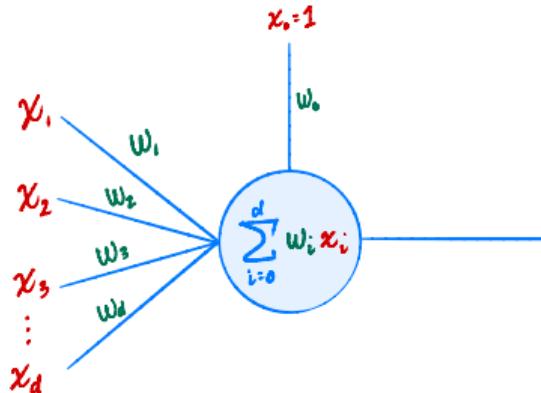


Today

- ▶ Design an artificial “neuron”, a **perceptron**.
- ▶ Train it to recognize images (handwritten digits).

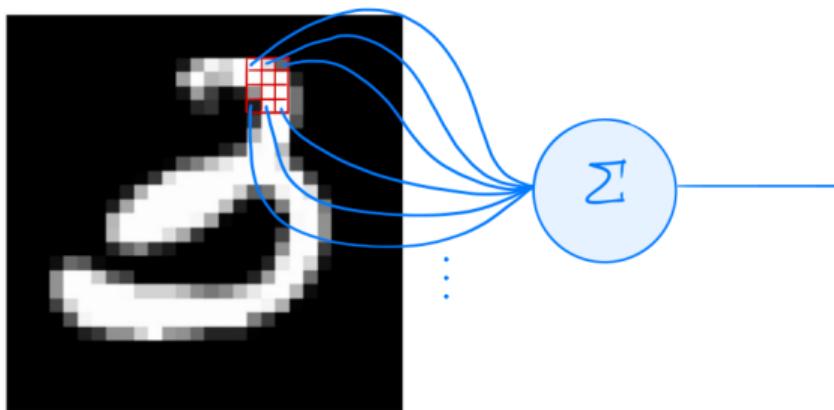


Modeling a Neuron



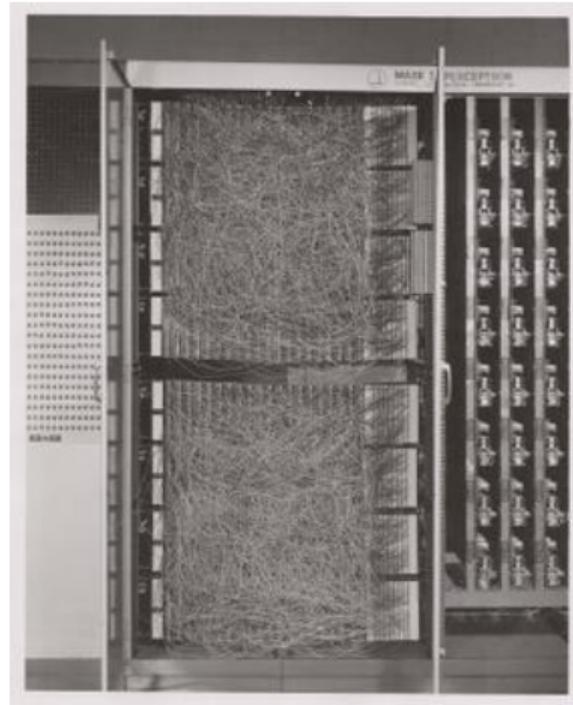
- ▶ Input: $\vec{x} = (x_1, \dots, x_d)^T$
- ▶ **Synapse weights:** $\vec{w} = (w_0, w_1, \dots, w_d)^T$
- ▶ Output: $\sum_{i=0}^d w_i x_i = \text{Aug}(\vec{x}) \cdot \vec{w}$
- ▶ This model is called a **perceptron**.

Example: Image Recognition



- ▶ Binary classifier:
 - ▶ If output > 0 , predicted digit is a three
 - ▶ If output < 0 , predicted digit is not a three

Rosenblatt's Perceptron (1958)



NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo
of Computer Designed to
Read and Grow Wiser

WASHINGTON, July 7 (UPI)—The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's \$2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen..

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of \$100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human be-

ings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

Without Human Controls

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

1958 New York Times...

In today's demonstration, the "704" was fed two cards, one with squares marked on the left side and the other with squares on the right side.

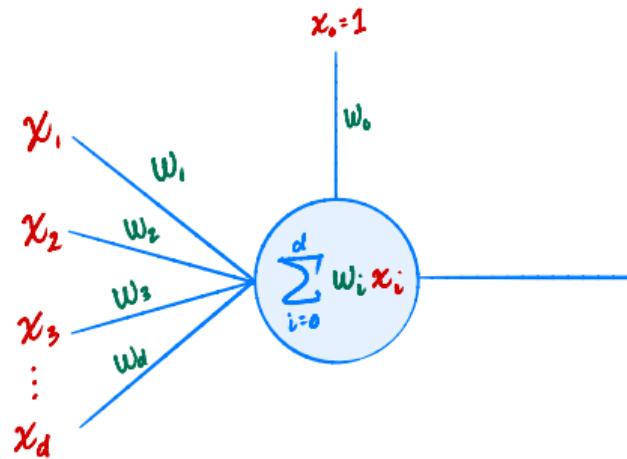
Learns by Doing

In the first fifty trials, the machine made no distinction between them. It then started registering a "Q" for the left squares and "O" for the right squares.

Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the computer had undergone a "self-induced change in the wiring diagram."

The first Perceptron will have about 1,000 electronic "association cells" receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.

Training a Perceptron



- ▶ A perceptron is trained by adjusting its weights, w_0, \dots, w_d .

Example: Predicting Survival

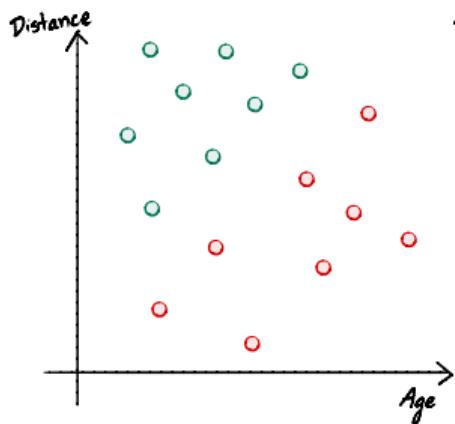
- ▶ We have a data set of hurricane survivors:

Age	Distance From Coast	Survived
21	30	1 (Yes)
84	2	-1 (No)
47	10	-1 (No)
30	15	1 (Yes)
:	:	:

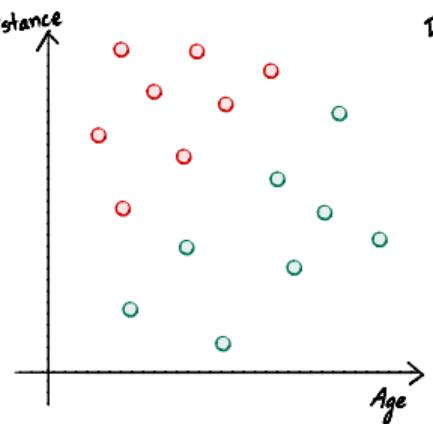
- ▶ Goal: train perceptron to predict if someone will survive.
 - ▶ If output is positive, prediction is “yes”
 - ▶ If output is negative, prediction is “no”

Example

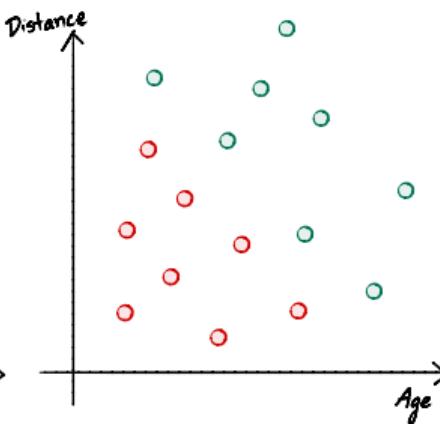
Suppose we plot the data, with green points for people who survived, and red points for those who did not. Which do we see?



(a)

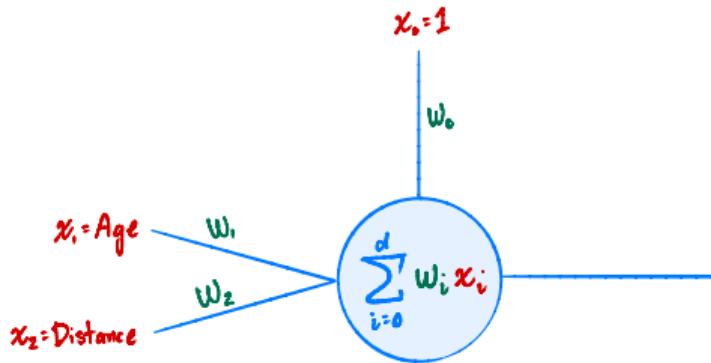


(b)



(c)

Example: Predicting Survival

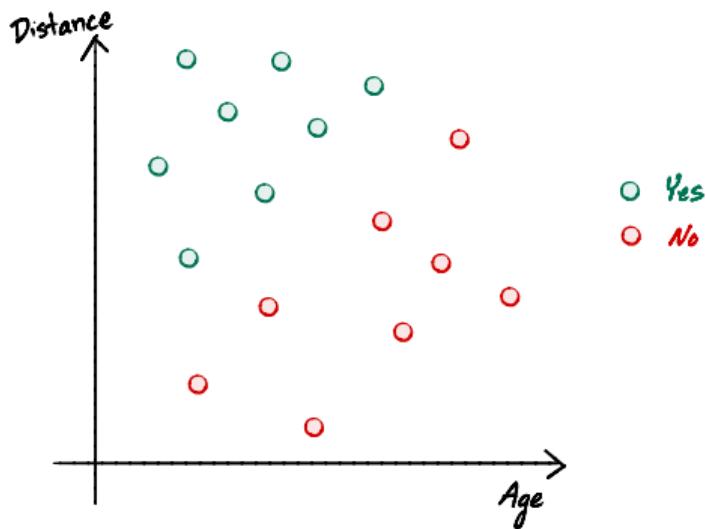


- ▶ Prediction rule:

$$H(\text{Age}, \text{Distance}) = w_0 + w_1 \times \text{Age} + w_2 \times \text{Distance}$$

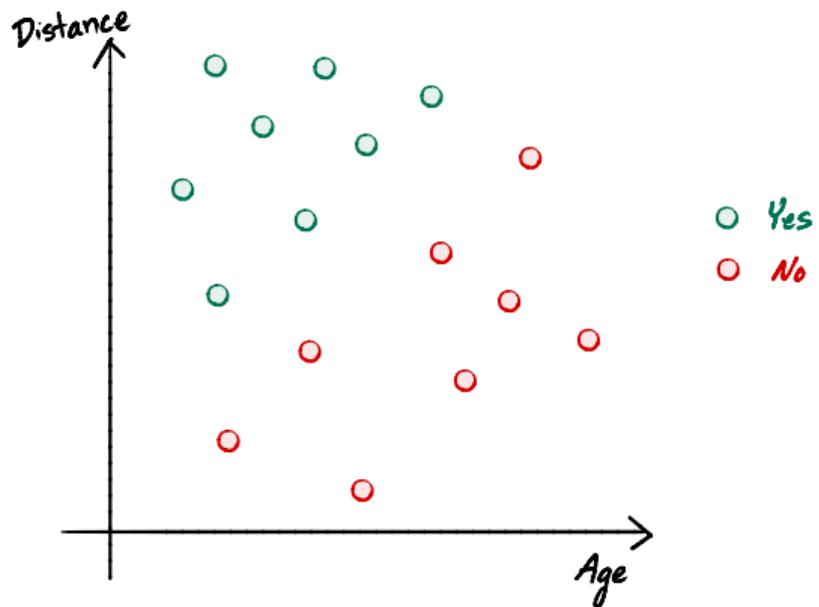
- ▶ If $H(x_1, x_2) > 0$, prediction is “yes”
- ▶ If $H(x_1, x_2) < 0$, prediction is “no”

The Decision Boundary



- ▶ The **decision boundary** is where $H = 0$.
- ▶ On one side of boundary, $H > 0$; prediction is **yes**.
- ▶ On other side, $H < 0$; prediction is **no**.
- ▶ **Important:** $|H(\vec{x})| \propto$ distance from boundary.

A Good Decision Boundary



Learning a Linear Decision Boundary

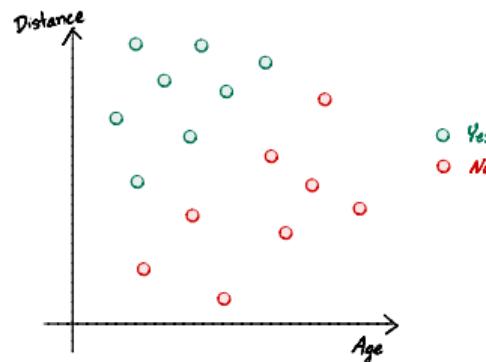
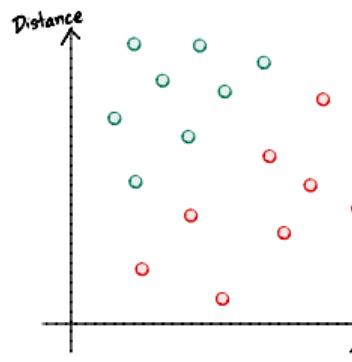
- ▶ Given feature vectors $\vec{x}^{(1)}, \dots, \vec{x}^{(n)}$ and labels y_1, \dots, y_n .
- ▶ Goal: find \vec{w} such that each point is **classified** correctly; i.e., it is on the correct side of boundary.
- ▶ We'll use empirical risk minimization.
- ▶ What is an appropriate objective function?

Risk #1: The 0-1 Loss

- The **0-1 risk** is then:

$$R_{01}(\vec{w}) = \frac{1}{n} \sum_{i=1}^n \begin{cases} 1, & \text{sign}(H(\vec{x})) \neq \text{sign}(y) \\ 0, & \text{sign}(H(\vec{x})) = \text{sign}(y) \end{cases}$$

- Proportion of points which are misclassified.
- Example:



Goal: Minimize the 0-1 Risk

- ▶ Find \vec{w} which results in fewest # of misclassified points.
- ▶ That is, minimize

$$R_{01}(\vec{w}) = \frac{1}{n} \sum_{i=1}^n \begin{cases} 1, & \text{sign}(H(\vec{x}^{(i)})) \neq \text{sign}(y_i) \\ 0, & \text{sign}(H(\vec{x}^{(i)})) = \text{sign}(y_i) \end{cases}$$

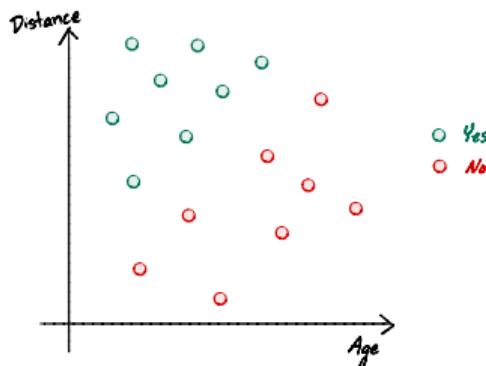
- ▶ Gradient descent?

Analyzing the 0-1 Risk

Is

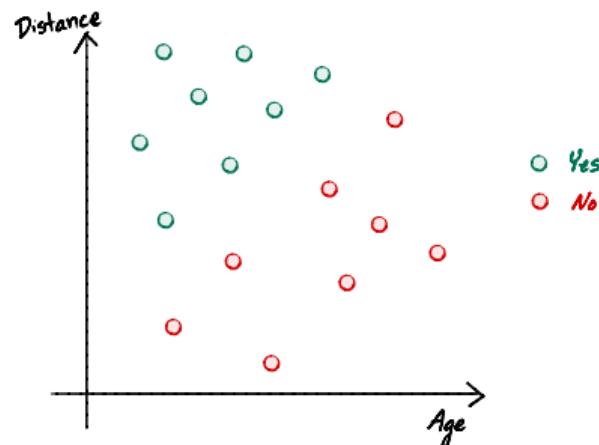
$$R_{01}(\vec{w}) = \frac{1}{n} \sum_{i=1}^n \begin{cases} 1, & \text{sign}(H(\vec{x}^{(i)})) \neq \text{sign}(y_i) \\ 0, & \text{sign}(H(\vec{x}^{(i)})) = \text{sign}(y_i) \end{cases}$$

continuous? Differentiable? Convex?



The Problem

- ▶ R_{01} is **flat**.
- ▶ The gradient gives no information.



Loss #2: The Perceptron Loss

- ▶ We need a loss function that isn't flat.
- ▶ If prediction is wrong, size of $|H(\vec{x})|$ measures how wrong.
- ▶ The **perceptron loss**:

$$L_{\text{tron}}(H(\vec{x}), y) = \begin{cases} 0, & \text{sign}(H(\vec{x})) = \text{sign}(y) \\ |H(\vec{x})|, & \text{sign}(H(\vec{x})) \neq \text{sign}(y) \end{cases}$$

Loss #2: The Perceptron Loss

- ▶ The **perceptron risk**:

$$R_{\text{tron}}(\vec{w}) = \frac{1}{n} \sum_{i=1}^n \begin{cases} 0, & \text{sign}(H(\vec{x}^{(i)})) = \text{sign}(y_i) \\ |H(\vec{x}^{(i)})|, & \text{sign}(H(\vec{x}^{(i)})) \neq \text{sign}(y_i) \end{cases}$$

=

where M is the set of misclassified points.

- ▶ **Continuous, not differentiable, not flat.**

Stochastic Gradient Descent for Perceptron Learning

- ▶ Compute “gradient” of

$$\ell(\vec{w}; \vec{x}^{(i)}, y_i) = \begin{cases} 0, & \text{sign}(H(\vec{x}^{(i)})) = \text{sign}(y_i) \\ |H(\vec{x}^{(i)})|, & \text{sign}(H(\vec{x}^{(i)})) \neq \text{sign}(y_i) \end{cases}$$

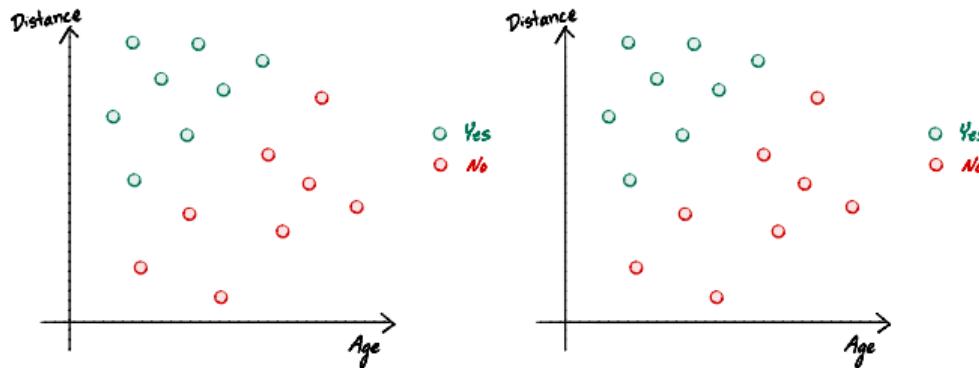
The Perceptron Algorithm

- ▶ Pick an initial $\vec{w}^{(0)}$.
- ▶ Repeat until convergence:
 - ▶ Construct set M of misclassified points using $\vec{w}^{(t-1)}$
 - ▶ If M is empty, break (no points misclassified).
 - ▶ Otherwise, loop over misclassified points $i \in M$, performing update:

$$\vec{w}^{(t)} = \vec{w}^{(t-1)} - \alpha \begin{cases} \text{Aug}(\vec{x}^{(i)}), & \vec{w}^{(t-1)} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 0 \\ -\text{Aug}(\vec{x}^{(i)}), & \vec{w}^{(t-1)} \cdot \text{Aug}(\vec{x}^{(i)}) < 0 \end{cases}$$

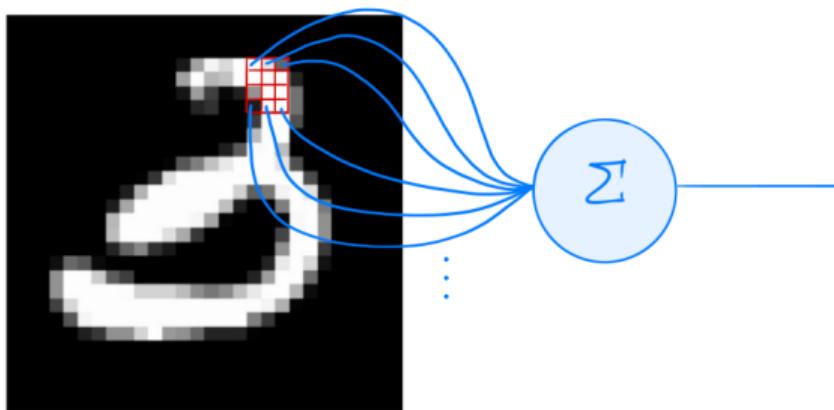
The Perceptron Algorithm

- ▶ Data is **linearly separable** if classes can be separated by a line (plane):



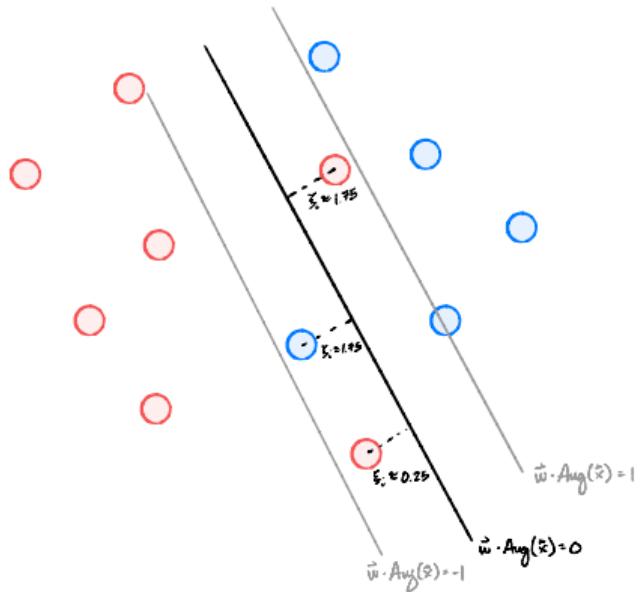
- ▶ If linearly separable, perceptron algorithm will terminate; classify all points correctly.

Example: Image Recognition



- ▶ Binary classifier:
 - ▶ If output > 0 , predicted digit is a three
 - ▶ If output < 0 , predicted digit is not a three

(DEMO)



CSE 151A

Intro to Machine Learning

Lecture 12 – Part 01

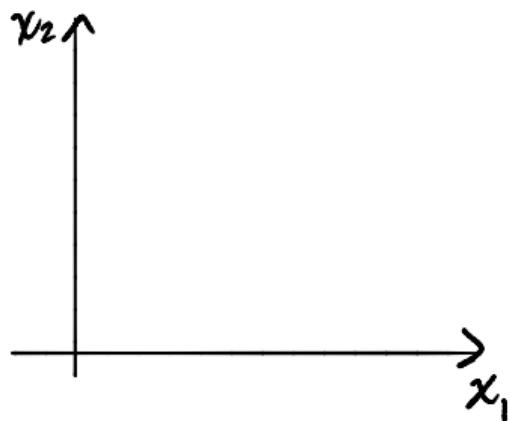
Support Vector Machines

Linear Classifiers

- ▶ **Prediction rule:** $H(\vec{x}) = \vec{w} \cdot \text{Aug}(\vec{x})$
 - ▶ Predict class 1 if $H(\vec{x}) > 0$
 - ▶ Predict class -1 if $H(\vec{x}) < 0$

Decision Boundary

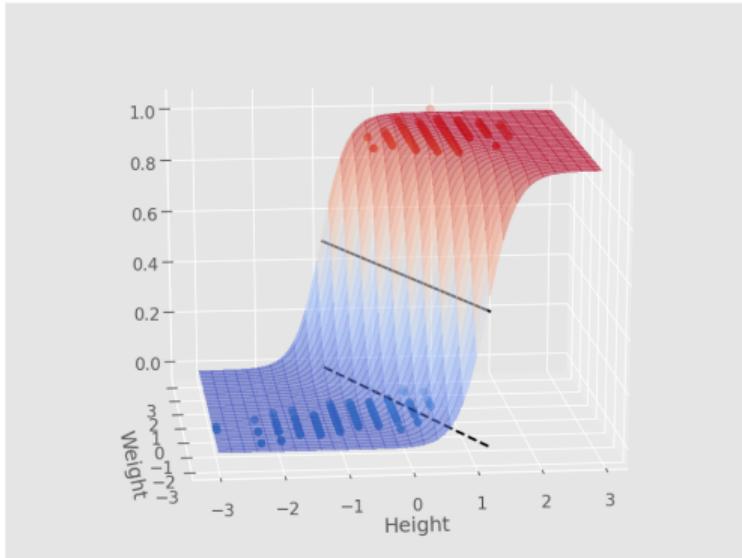
- ▶ $\text{Aug}(\vec{x}) \cdot \vec{w}$ is proportional to distance from boundary.



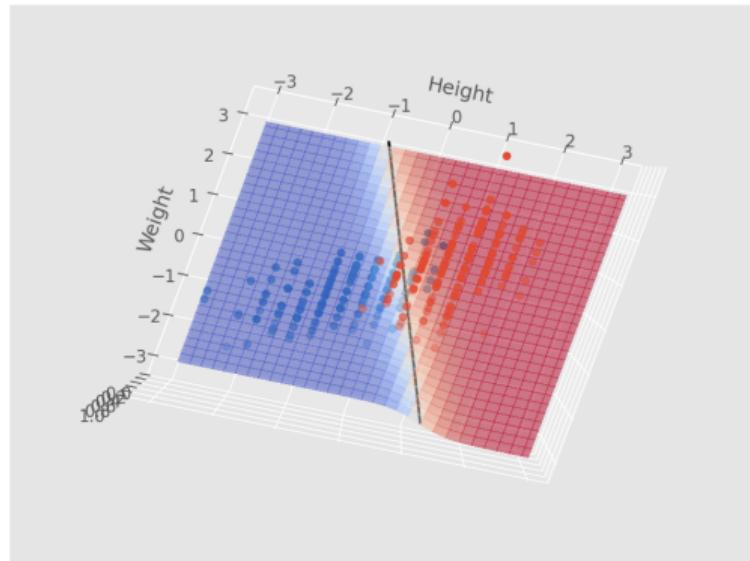
Recall: Logistic Regression

- ▶ **Prediction Rule:** $H(\vec{x}) = \sigma(\vec{w} \cdot \text{Aug}(\vec{x}))$
- ▶ Find \vec{w} by maximizing log likelihood.
- ▶ Predict class 1 if $H(\vec{x}) > 0.5$, class -1 otherwise.
- ▶ But $\sigma(\vec{w} \cdot \text{Aug}(\vec{x})) > 0.5 \iff \vec{w} \cdot \text{Aug}(\vec{x}) > 0$

Recall: Logistic Regression



Recall: Logistic Regression

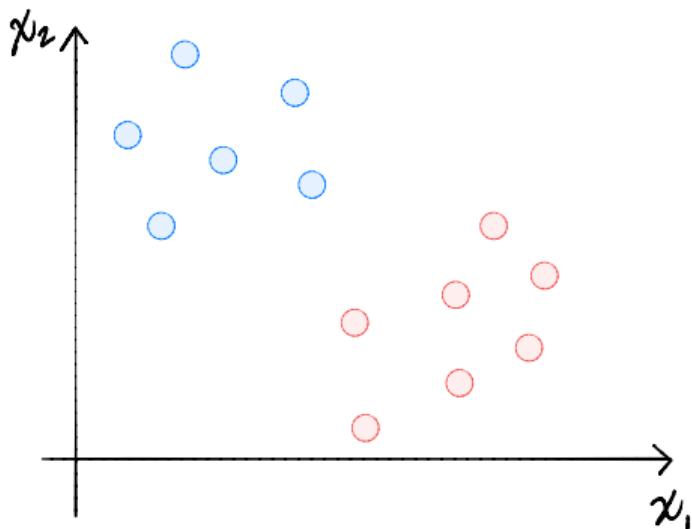


Recall: the Perceptron

- ▶ **Prediction Rule:** $H(\vec{x}) = \vec{w} \cdot \text{Aug}(\vec{x})$
- ▶ Find \vec{w} by minimizing perceptron risk.
- ▶ **Theorem:** if the training data is **linearly separable**, the perceptron algorithm find a dividing hyperplane.

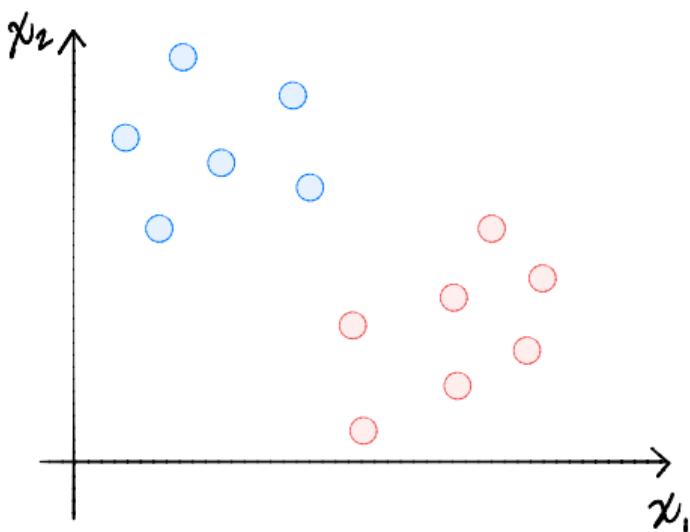
Perceptron Problems

The learned perceptron may have a small **margin**.



Perceptron Problems

The learned perceptron may have a small **margin**.



We prefer **large margins** for generalization.

Maximum Margin Classifiers

- ▶ **Assume:** linear separability (for now).
- ▶ Many possible boundaries with zero error.
- ▶ **Goal:** Find linear boundary with largest margin w.r.t. training data.

Observation

- ▶ Training data: $\{(\vec{x}^{(i)}, y_i)\}$
- ▶ Classification is correct when:
$$\begin{cases} \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) > 0, & \text{if } y_i = 1 \\ \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) < 0, & \text{if } y_i = -1 \end{cases}$$
- ▶ Equivalently, classification is correct if:

$$y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) > 0$$

Recall

- ▶ $y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \propto$ to distance from boundary.
- ▶ Our goal: find \vec{w} that maximizes the smallest distance.

$$\vec{w}_{\text{best}} = \underset{\vec{w} \in \mathbb{R}^{d+1}}{\operatorname{argmax}} \min_{i \in 1, \dots, n} [y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)})]$$

- ▶ This looks **hard**. But there is a **trick**.

Another Observation

- ▶ If linearly separable, then there is a \vec{w} such that

$$y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) > 0$$

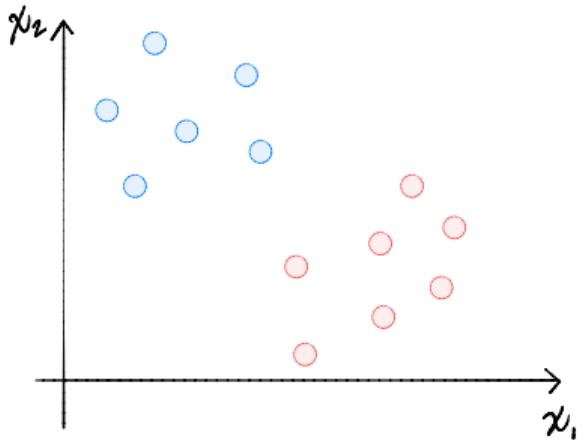
for all $i = 1, \dots, n.$

- ▶ Actually, linearly separable \Rightarrow there is a \vec{w} s.t.

$$y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 1$$

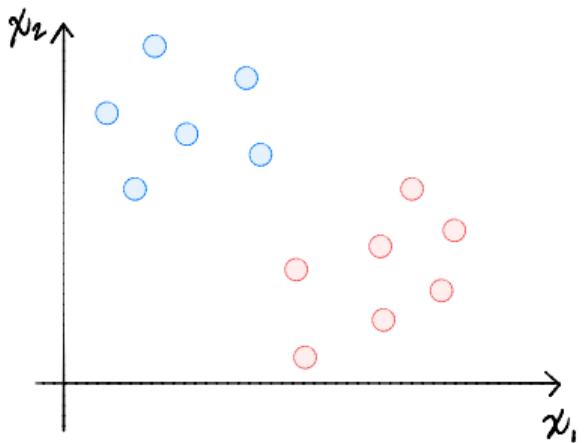
for all $i = 1, \dots, n.$

Why?



- ▶ Suppose \vec{w} separates, but $y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) = 0.01$
- ▶ Define $\vec{\omega} = \frac{1}{0.01} \vec{w} = 100 \vec{w}$.
- ▶ Then $y_i \vec{\omega} \cdot \text{Aug}(\vec{x}^{(i)}) = 1$
- ▶ **Note:** $\|\vec{\omega}\|$ is large!

Why?



- ▶ Suppose \vec{w} separates, but $y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) = 0.5$
- ▶ Define $\vec{\omega} = \frac{1}{0.5} \vec{w} = 2\vec{w}$.
- ▶ Then $y_i \vec{\omega} \cdot \text{Aug}(\vec{x}^{(i)}) = 1$
- ▶ **Note:** $\|\vec{\omega}\|$ is smaller!

The Trick

- ▶ We will demand that

$$y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 1$$

- ▶ The larger $\|\vec{w}\|$, the smaller the margin.
- ▶ **New Goal:** Minimize $\|\vec{w}\|^2$ subject to
 $y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 1$ for all i .

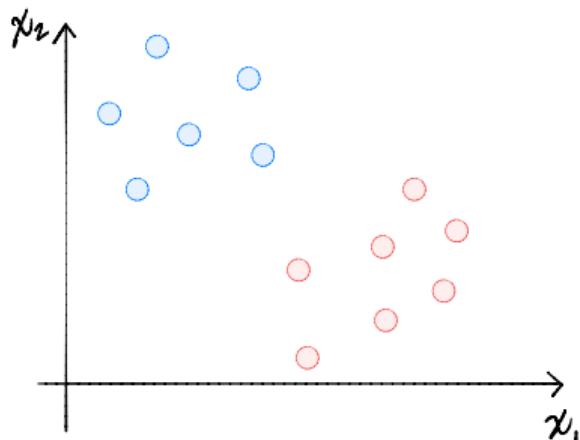
Optimization

- ▶ Minimize $\|\vec{w}\|^2$ subject to $y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 1$ for all i .
- ▶ This is a **convex, quadratic** optimization problem.
- ▶ Can be solved efficiently with **quadratic programming**.
 - ▶ But there is no exact general formula for the solution

Support Vectors

- ▶ A **support vector** is a training point $\vec{x}^{(i)}$ such that

$$y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) = 1$$



Support Vector Machines (SVMs)

- ▶ Then maximum margin solution \vec{w} is a linear combination of the support vectors.
- ▶ Let S be the set of support vectors. Then

$$\vec{w} = \sum_{i \in S} y_i \alpha_i \text{Aug}(\vec{x}^{(i)})$$

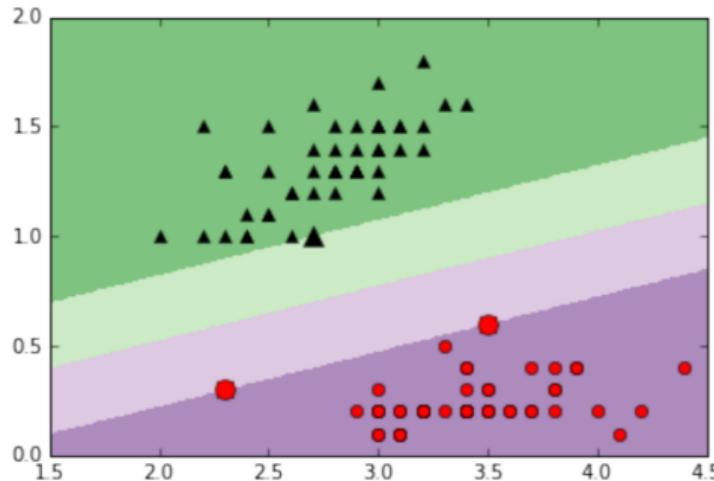
Example: Irises

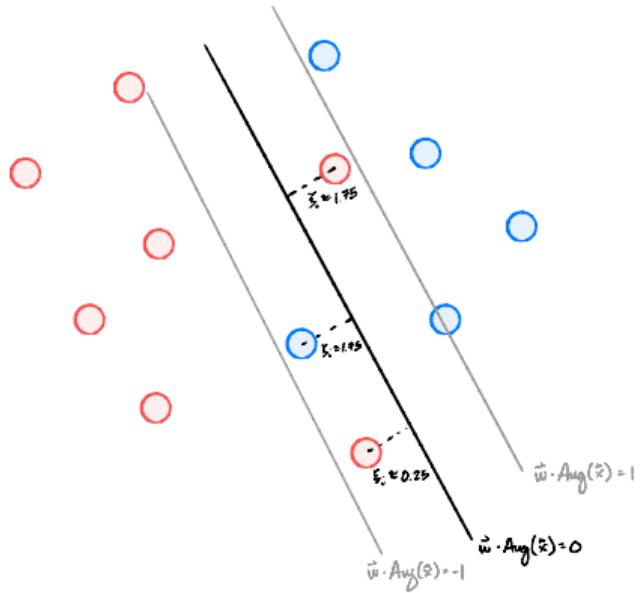


- ▶ 3 classes: *iris setosa*, *iris versicolor*, *iris virginica*
- ▶ 4 measurements: petal width/height, sepal width/height

Example: Irises

- ▶ Using only sepal width/petal width
- ▶ Two classes: versicolor (black), setosa (red)





CSE 151A

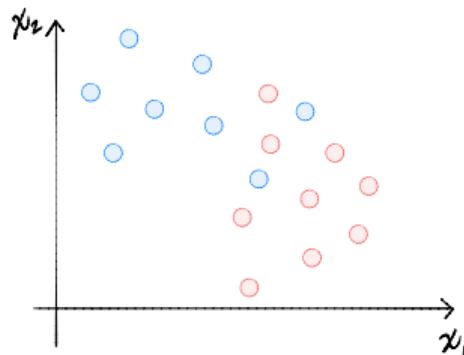
Intro to Machine Learning

Lecture 12 – Part 02

Soft-Margin SVMs

Non-Separability

- ▶ So far we've assumed data is linearly separable.
- ▶ What if it isn't?

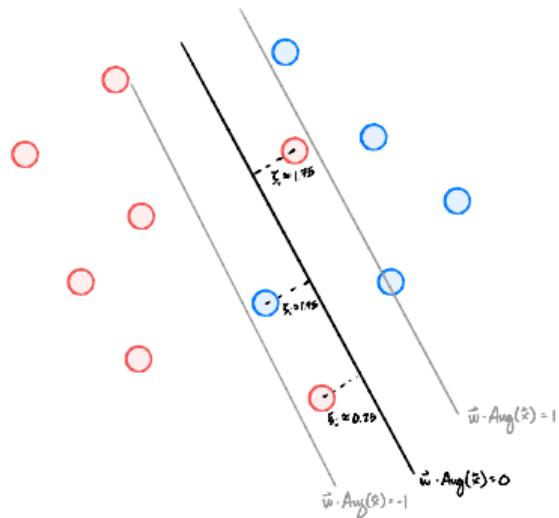


The Problem

- ▶ **Old Goal:** Minimize $\|\vec{w}\|^2$ subject to $y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 1$ for all i .
- ▶ This **no longer makes sense**.

Cut Some Slack

- ▶ **Idea:** allow some classifications to be ξ_i wrong, but not too wrong.



Cut Some Slack

- ▶ **New problem.** Fix some number $C \geq 0$.

$$\min_{\vec{w} \in \mathbb{R}^{d+1}, \vec{\xi} \in \mathbb{R}^n} \|\vec{w}\|^2 + C \sum_{i=1}^n \xi_i$$

subject to $y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 1 - \xi_i$ for all i , $\vec{\xi} \geq 0$.

The Slack Parameter, C

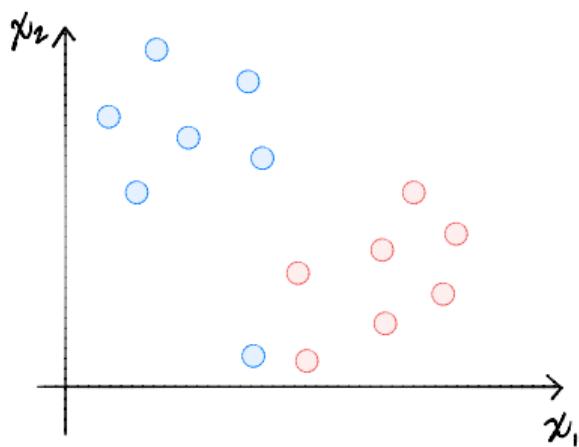
- ▶ C controls how much slack is given.

$$\min_{\vec{w} \in \mathbb{R}^{d+1}, \vec{\xi} \in \mathbb{R}^n} \|\vec{w}\|^2 + C \sum_{i=1}^n \xi_i$$

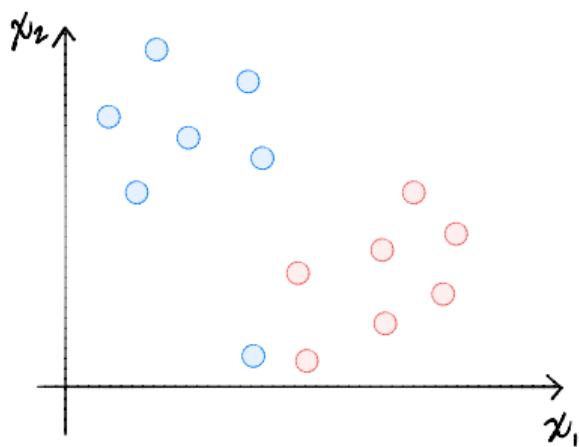
subject to $y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 1 - \xi_i$ for all i , $\vec{\xi} \geq 0$.

- ▶ Large C : don't give much slack. Avoid misclassifications.
- ▶ Small C : allow more slack at the cost of misclassifications.

Example: Small C



Example: Large C



Soft and Hard Margins

- ▶ Max-margin SVM from before has **hard margin**.
- ▶ Now: the **soft margin** SVM.
- ▶ As $C \rightarrow \infty$, the margin hardens.

Another View: Loss Functions

- ▶ Recall our problem:

$$\min_{\vec{w} \in \mathbb{R}^{d+1}, \vec{\xi} \in \mathbb{R}^n} \|\vec{w}\|^2 + C \sum_{i=1}^n \xi_i$$

subject to $y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 1 - \xi_i$ for all i , $\vec{\xi} \geq 0$.

- ▶ **Note:** if $\vec{x}^{(i)}$ is misclassified, then

$$\xi_i = 1 - y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)})$$

Another View: Loss Functions

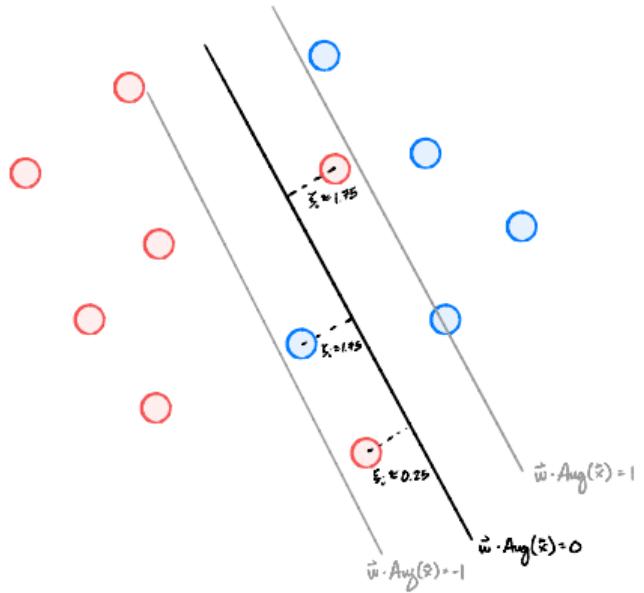
- ▶ New problem:

$$\min_{\vec{w} \in \mathbb{R}^{d+1}, \vec{\xi} \in \mathbb{R}^n} \|\vec{w}\|^2 + C \sum_{i=1}^n \max\{0, 1 - y_i \vec{w} \cdot \vec{x}^{(i)}\}$$

- ▶ $\max\{0, 1 - y_i \vec{w} \cdot \vec{x}^{(i)}\}$ is called the **hinge loss**.

Another Way to Optimize

- ▶ We can use **subgradient descent** to minimize SVM risk.



CSE 151A

Intro to Machine Learning

Lecture 12 – Part 03

Sentiment Analysis

Why use linear predictors?

- ▶ Linear classifiers look to be very simple.
- ▶ That can be both **good** and **bad**.
 - ▶ **Good**: the math is tractable, less likely to overfit
 - ▶ **Bad**: may be too simple, underfit
- ▶ They can work surprisingly well.

Sentiment Analysis

- ▶ **Given:** a piece of text.
- ▶ **Determine:** if it is **positive** or **negative** in tone
- ▶ Example: “Needless to say, I wasted my money.”

The Data

- ▶ Sentences from reviews on Amazon, Yelp, IMDB.
- ▶ Each labeled (by a human) **positive** or **negative**.
- ▶ Examples:
 - ▶ “**Needless to say, I wasted my money.**”
 - ▶ “**I have to jiggle the plug to get it to line up right.**”
 - ▶ “**Will order from them again!**”
 - ▶ “**He was very impressed when going from the original battery to the extended battery.**”

The Plan

- ▶ We'll train a soft-margin SVM.
- ▶ **Problem:** SVMs take **fixed-length vectors** as inputs, not sentences.

Bags of Words

To turn a document into a fixed-length vector:

- ▶ First, choose a **dictionary** of words:
 - ▶ E.g.: ["wasted", "impressed", "great", "bad", "again"]
- ▶ Count number of occurrences of each dictionary word in document.
 - ▶ "It was bad. So bad that I was impressed at how bad it was." → $(0, 1, 0, 3, 0)^T$
- ▶ This is called a **bag of words** representation.

Choosing the Dictionary

- ▶ Many ways of choosing the dictionary.
- ▶ Easiest: take all of the words in the training set.
 - ▶ Perhaps throw out **stop words** like “the”, “a”, etc.
- ▶ Resulting dimensionality of feature vectors: large.

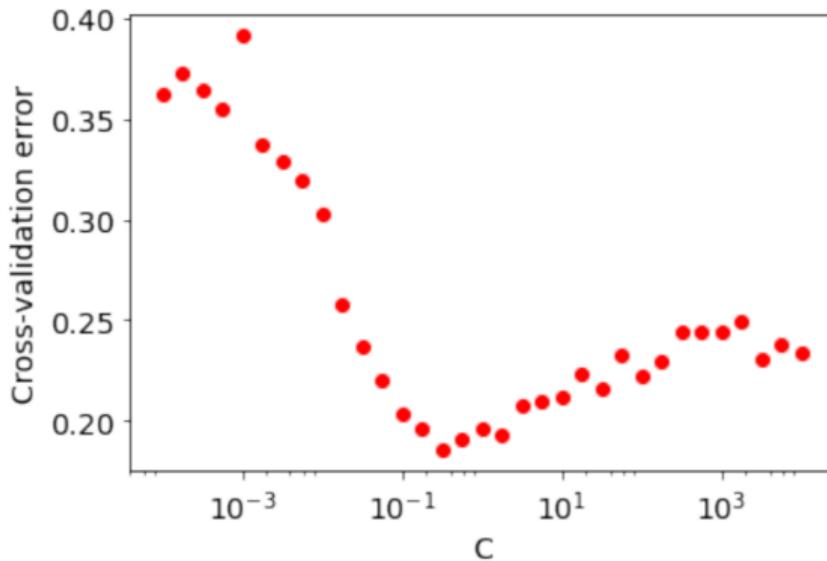
Experiment

- ▶ Bag of words features with 4500 word dictionary.
- ▶ 2500 training sentences, 500 test sentences.
- ▶ Train a soft margin SVM.

Choosing C

- ▶ We have to choose the slack parameter, C .
- ▶ Use **cross validation!**

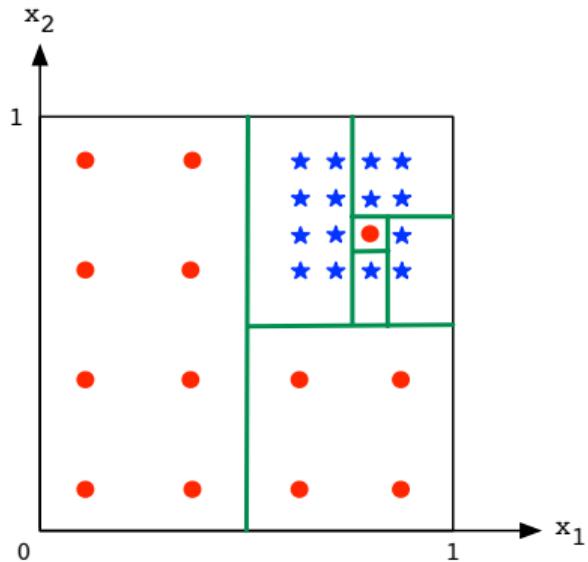
Cross Validation



Results

- ▶ With $C = 0.32$, test error $\approx 15.6\%$.

C	training error (%)	test error (%)	# support vectors
0.01	23.72	28.4	2294
0.1	7.88	18.4	1766
1	1.12	16.8	1306
10	0.16	19.4	1105
100	0.08	19.4	1035
1000	0.08	19.4	950



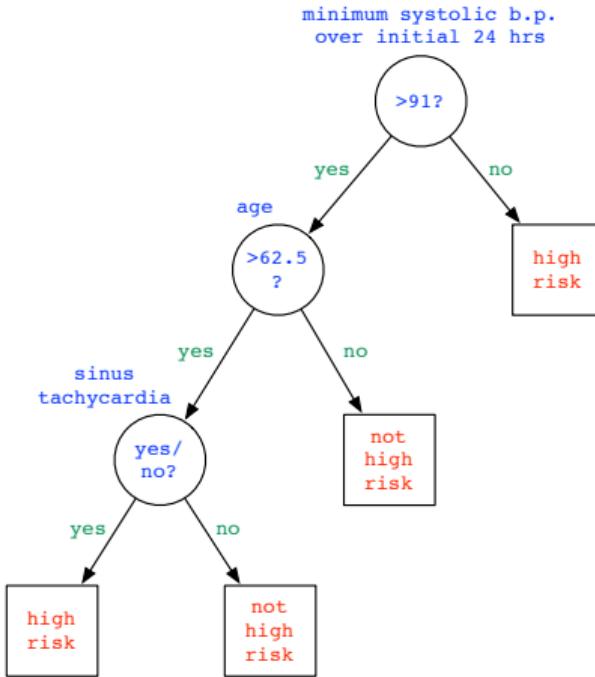
CSE 151A
Intro to Machine Learning

Lecture 13 – Part 01
Decision Trees

The Problem

- ▶ UCSD Medical Center (1970s): identify patients at risk of dying within 30 days after heart attack.

A Decision Tree



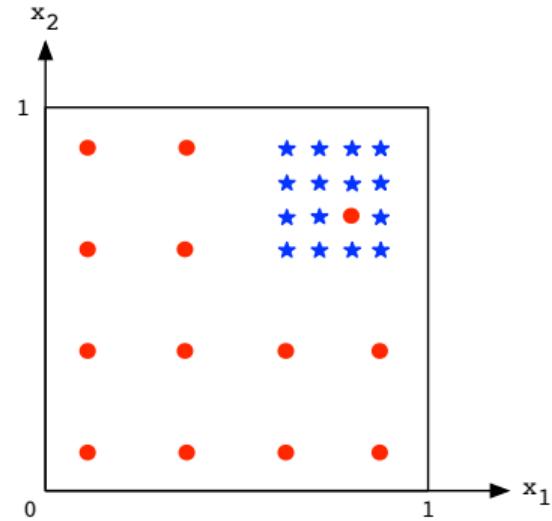
Decision Trees

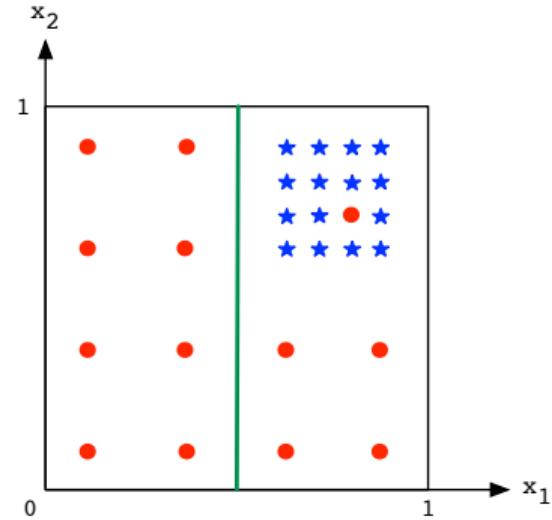
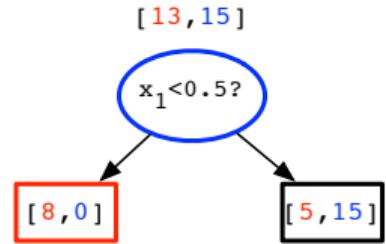
- ▶ A **decision tree** is a rooted tree.
- ▶ Internal nodes ask yes/no questions.
 - ▶ **Categorical:** Is patient a male?
 - ▶ **Numerical:** Is patient's age > 62.5 years?
- ▶ Leaf nodes are decisions (class labels).
- ▶ Path from root is a sequence of “and”s:
 - ▶ Is patient over 62.5 **and** male **and** BP > 100?
Then high risk.

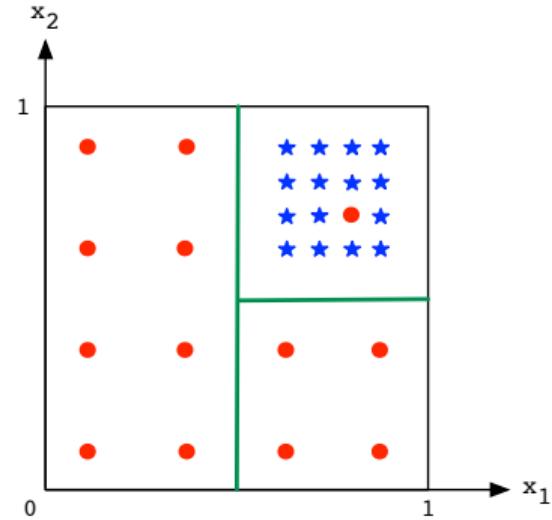
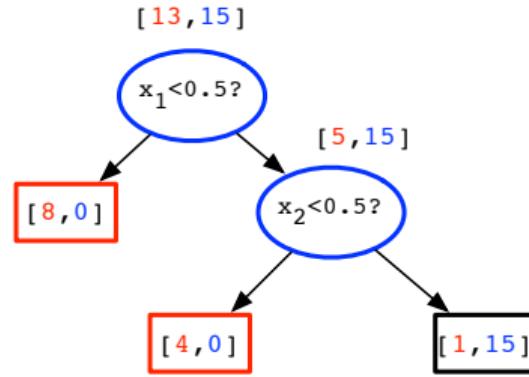
Learning Decision Trees

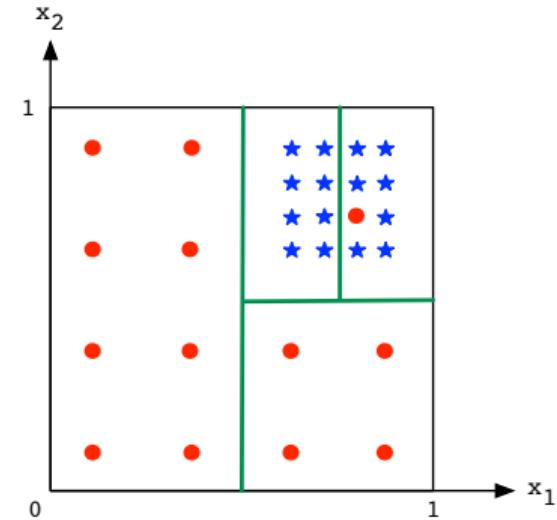
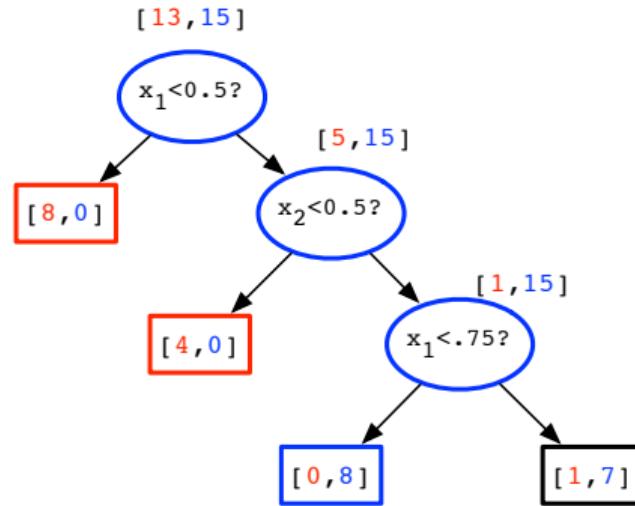
- ▶ How do we **learn** a tree from data?
 - ▶ Find right sequence of questions so that each training point is correctly classified.

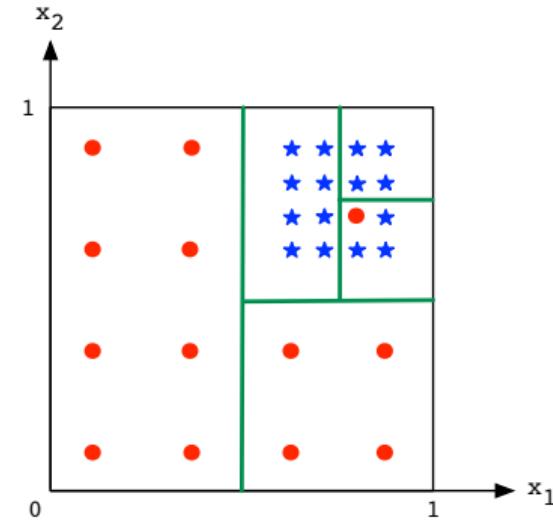
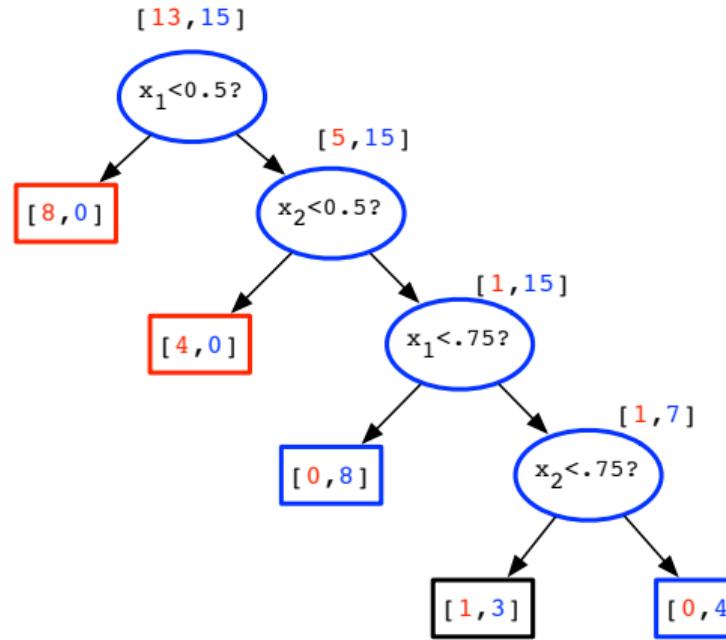
[13 , 15]

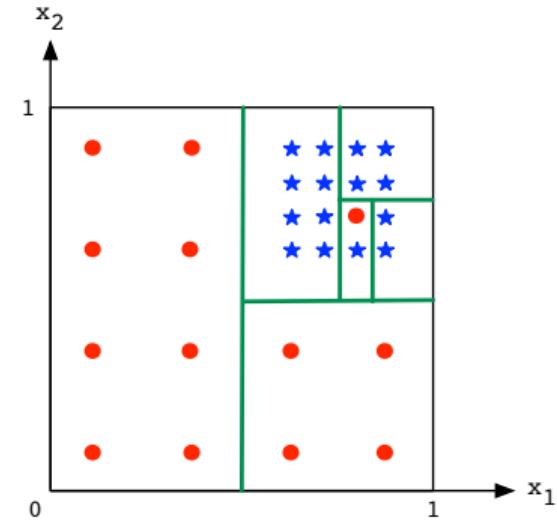
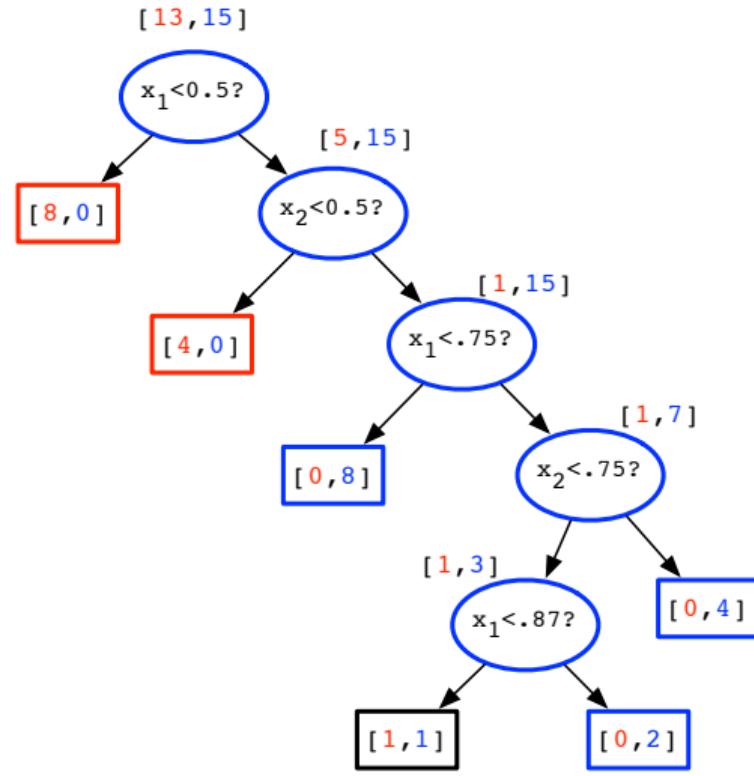


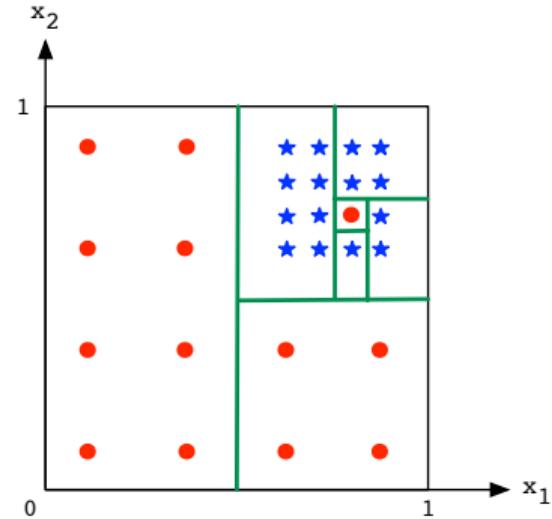
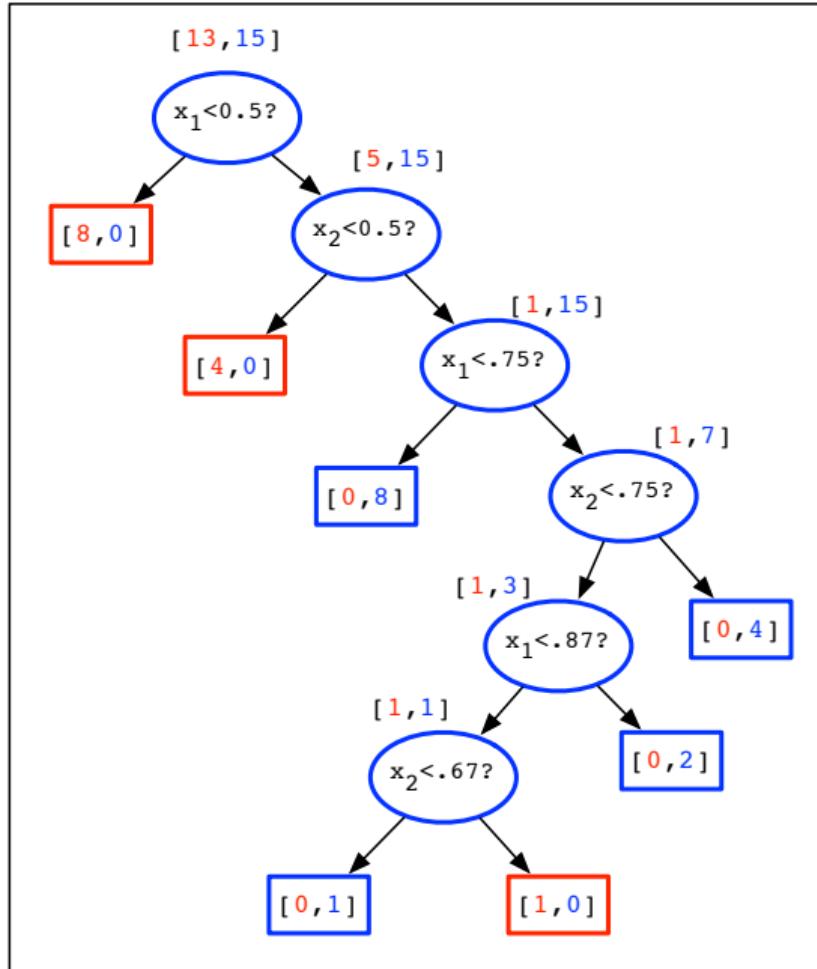












Learning Decision Trees

- ▶ Start with single node containing all data points
- ▶ Repeat greedy procedure:
 - ▶ Look at all possible questions (splits)
 - ▶ Pick the one that most reduces **uncertainty**.
- ▶ Stop when each leaf node is **pure**.

Aside: Generating Possible Questions

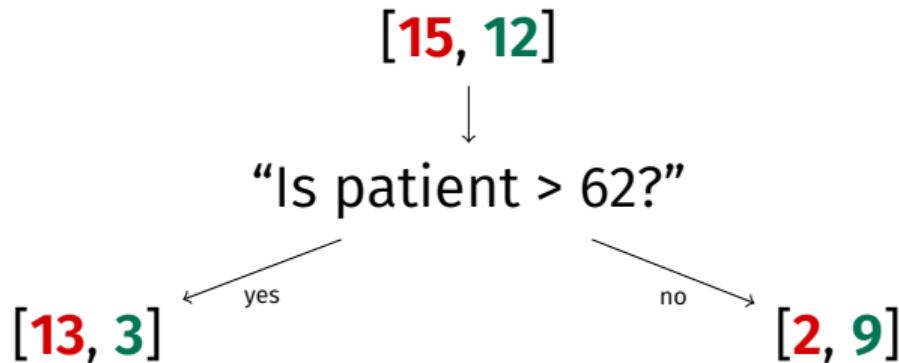
- ▶ **Categorical:** One question per value seen.
- ▶ E.g., county of residence.
 - ▶ Patient is from San Diego County?
 - ▶ Patient is from Riverside County?
 - ▶ Patient is from Orange County?

Aside: Generating Possible Questions

- ▶ **Numerical:** one question between each pair of consecutive values.
- ▶ E.g., ages in data = {42, 43, 55, 57, 61, 75}
 - ▶ Patient is < 42.5?
 - ▶ Patient is < 49?
 - ▶ ...
 - ▶ Patient is < 68?

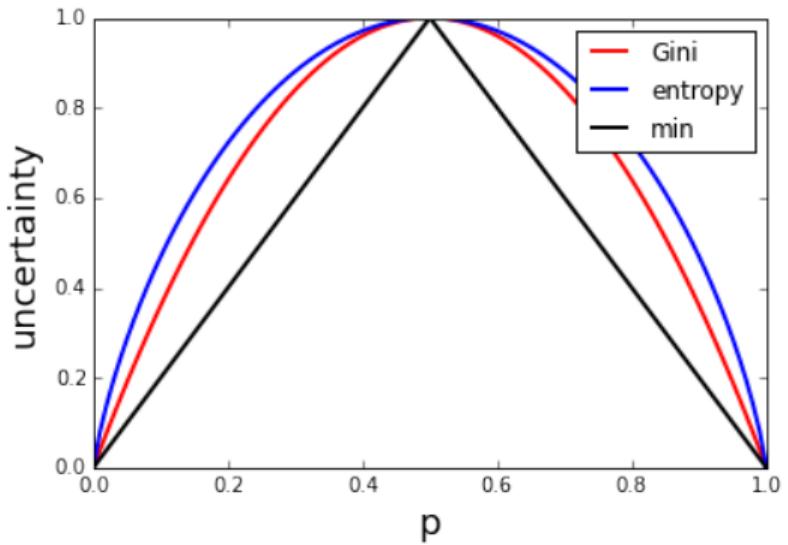
Measuring Uncertainty

- ▶ A good question splits the data by class.



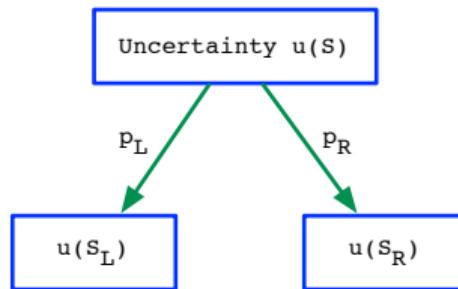
Measuring Uncertainty

- ▶ Suppose our node contains proportions:
 - ▶ p from class +
 - ▶ $(1 - p)$ from class -
- ▶ Common **uncertainty scores**:
 - ▶ **Misclassification rate**: $\min\{p, 1 - p\}$
 - ▶ **Gini index**: $2p(1 - p)$
 - ▶ **Entropy**: $p \log \frac{1}{p} + (1 - p) \log \frac{1}{1-p}$

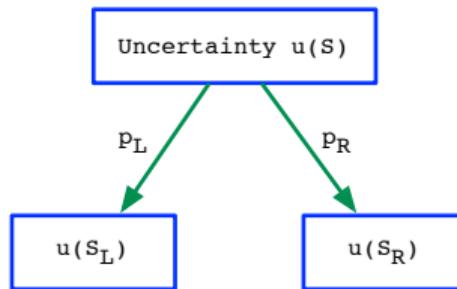


Benefit of a Question

- ▶ Let $u(S)$ be the uncertainty score for a set of labeled points, S .
 - ▶ Consider a particular question (split):



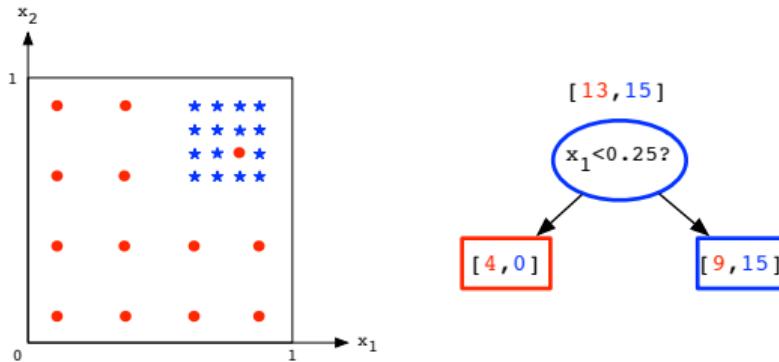
Benefit of a Question



- ▶ Resulting uncertainty:

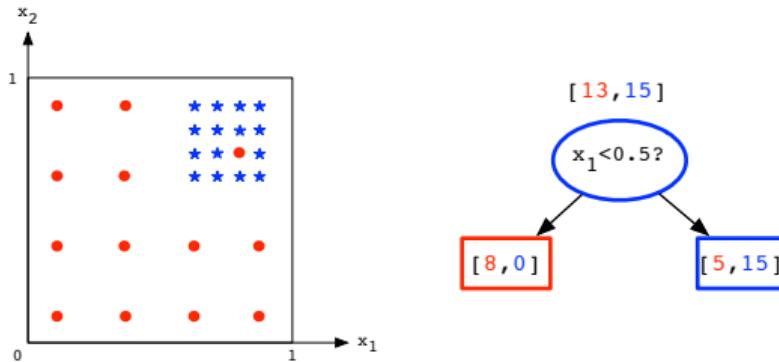
$$p_L u(S_L) + p_R u(S_R)$$

Example



- ▶ Initial Gini uncertainty: $2 \times \frac{13}{28} \times \frac{15}{28}$.
- ▶ $p_L u(S_L) + p_R u(S_R) = \frac{4}{28} \cdot 0 + \frac{24}{28} \cdot 2 \cdot \frac{9}{24} \cdot \frac{15}{24} = \frac{45}{112}$

Example

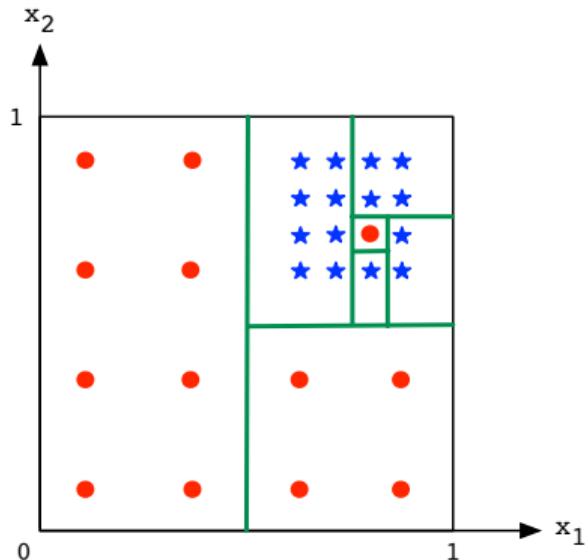


- ▶ Initial Gini uncertainty: $2 \times \frac{13}{28} \times \frac{15}{28}$.
- ▶ $p_L u(S_L) + p_R u(S_R) = \frac{8}{28} \cdot 0 + \frac{20}{28} \cdot 2 \cdot \frac{5}{20} \cdot \frac{15}{20} = \frac{30}{112}$

Summary

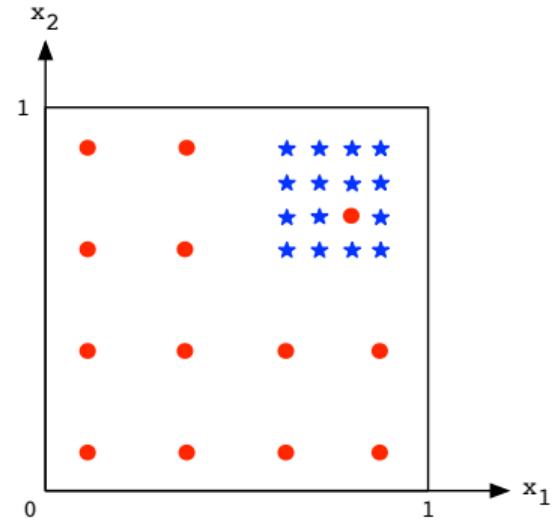
To learn a decision tree:

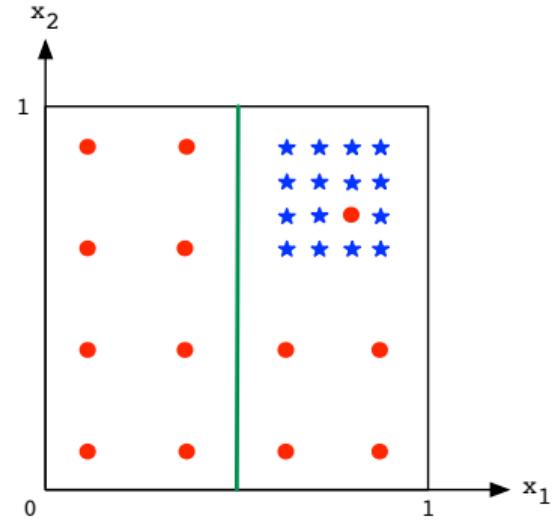
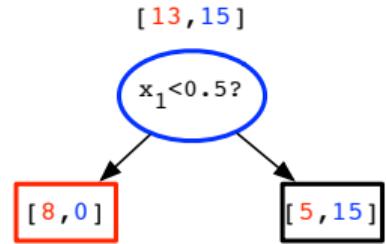
- ▶ Pick a measure of uncertainty (Gini, Entropy, etc.)
- ▶ Recursively ask question minimizing uncertainty.

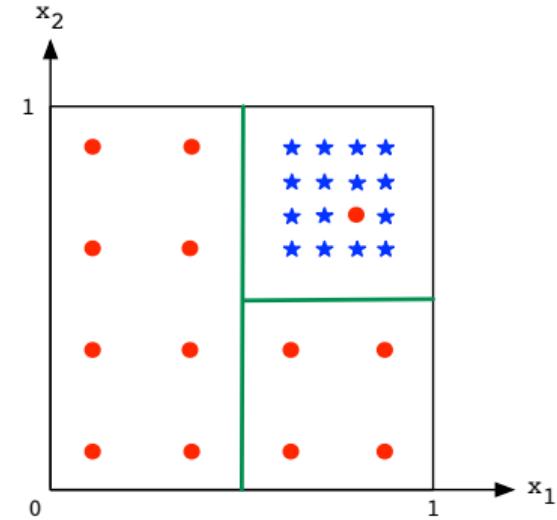
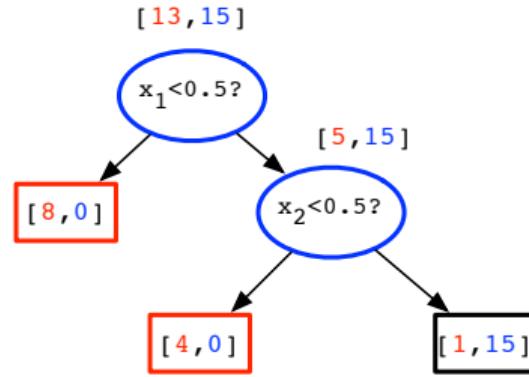


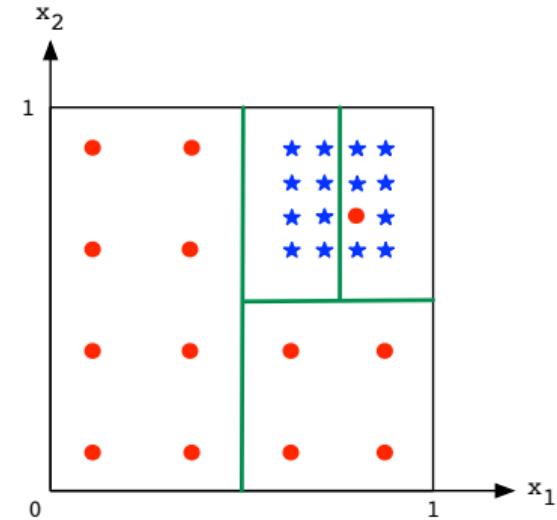
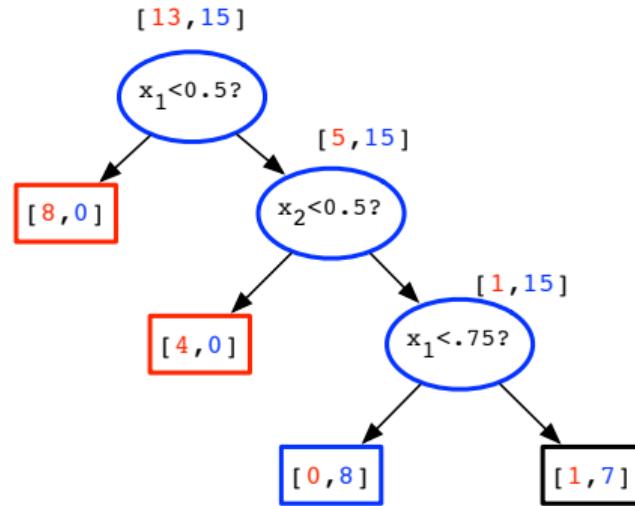
CSE 151A
Intro to Machine Learning
Lecture 13 – Part 02
Overfitting

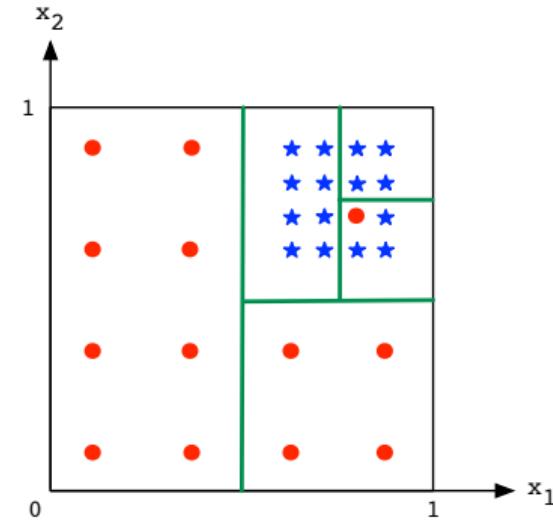
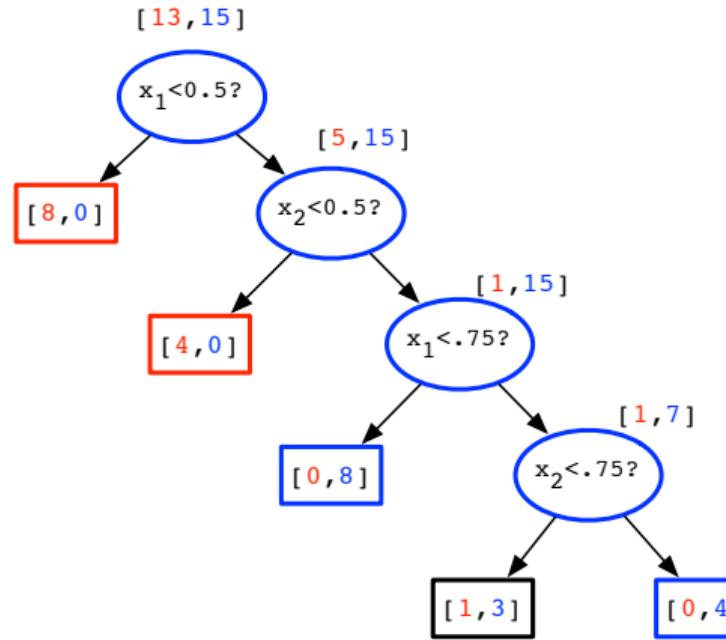
[13 , 15]

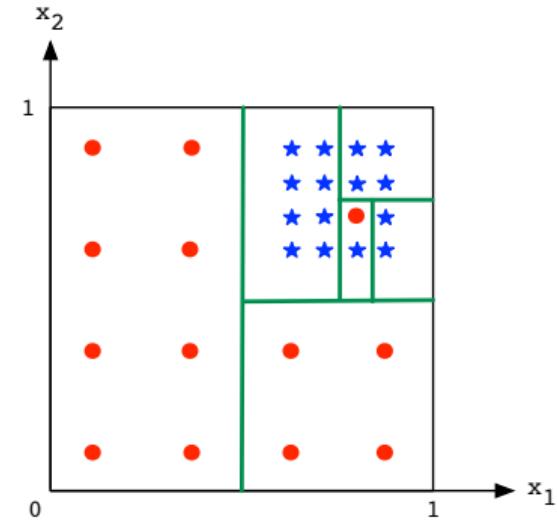
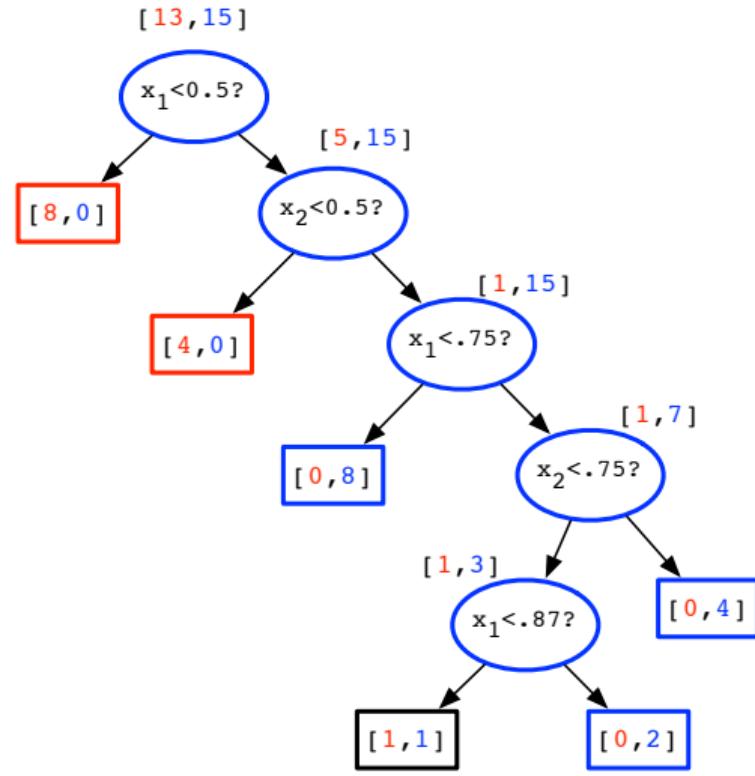


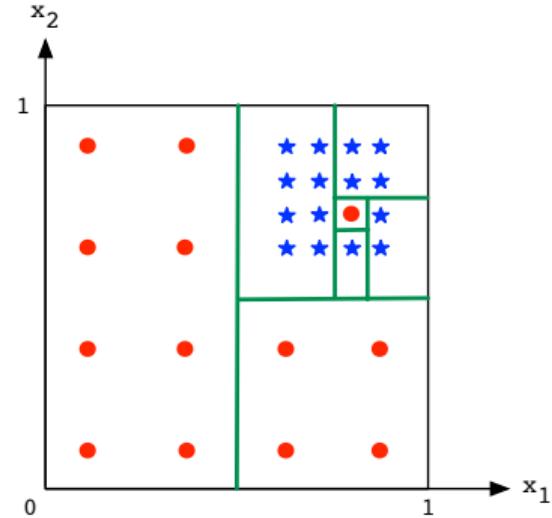
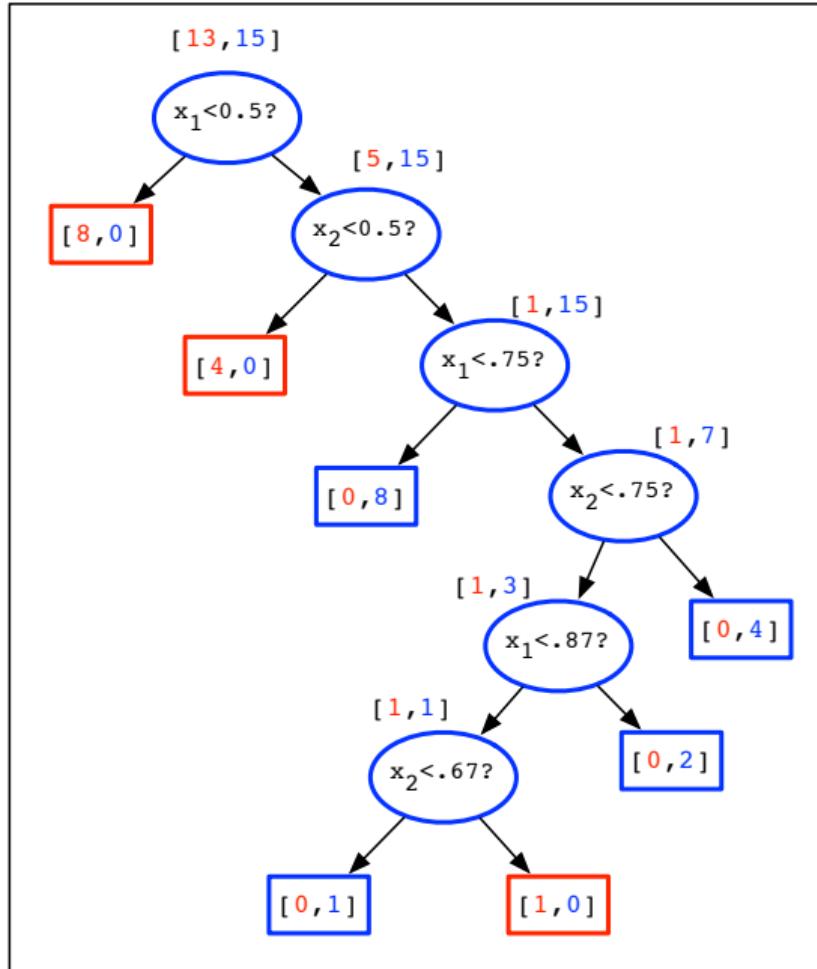






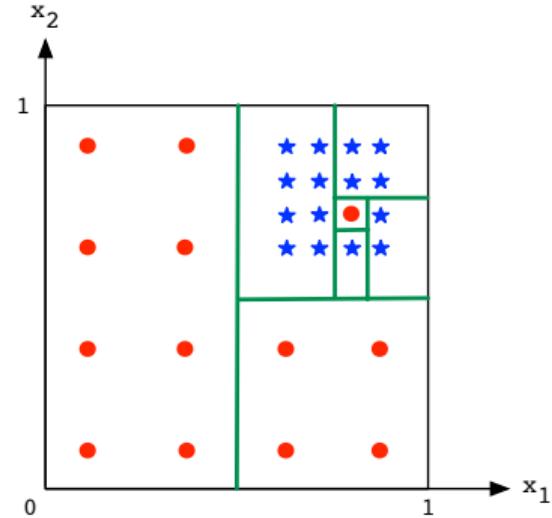
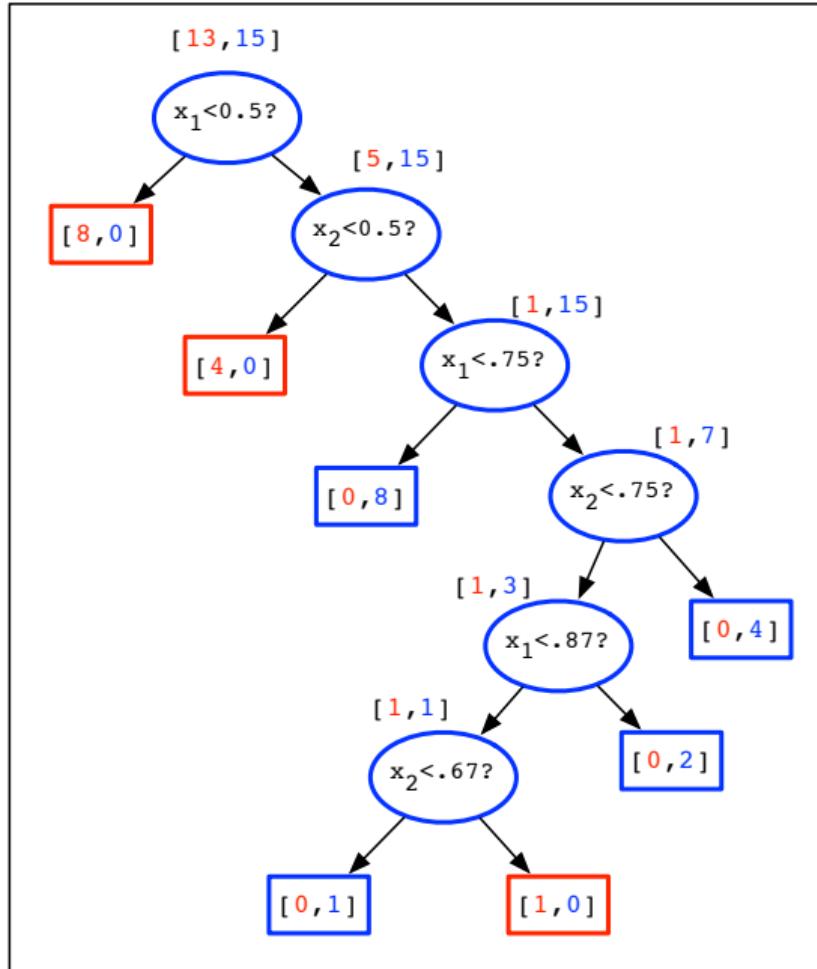


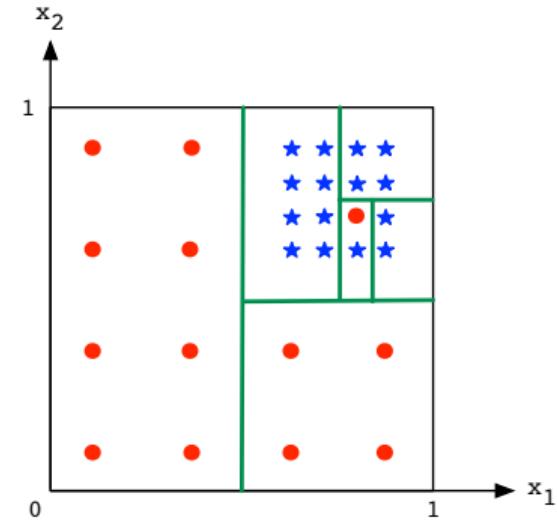
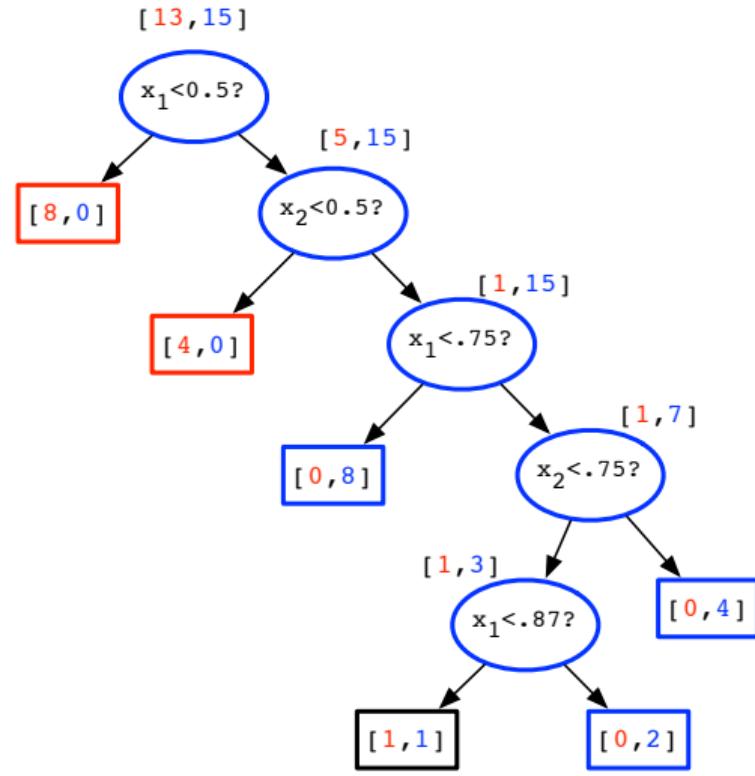


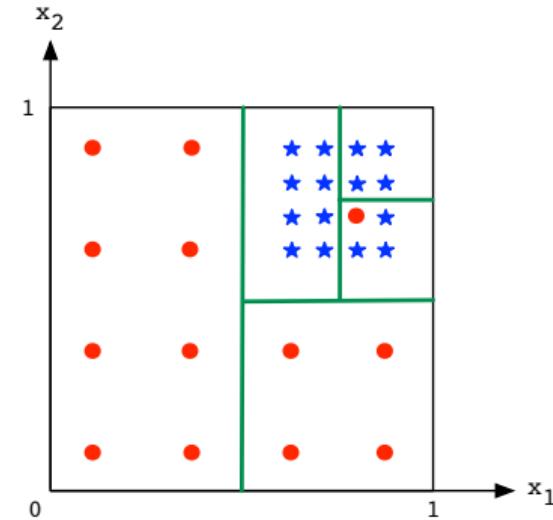
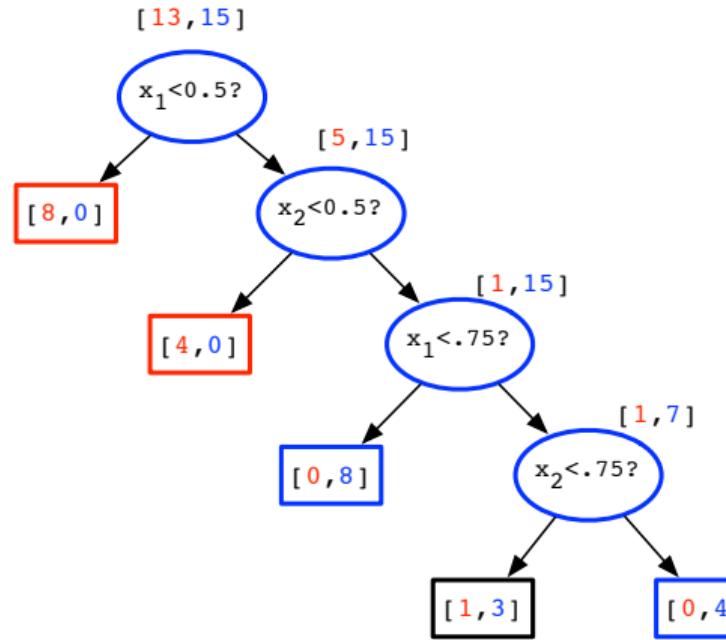


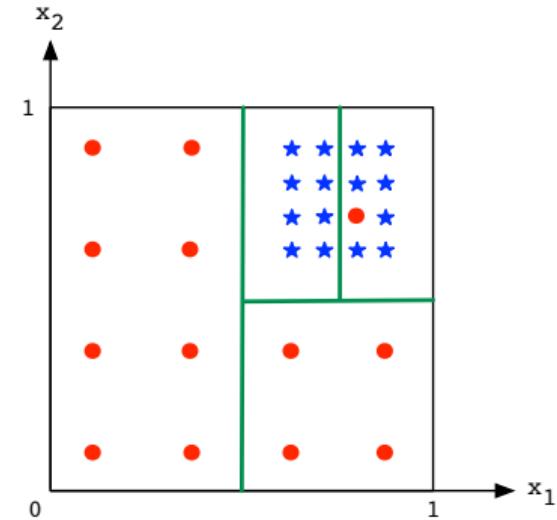
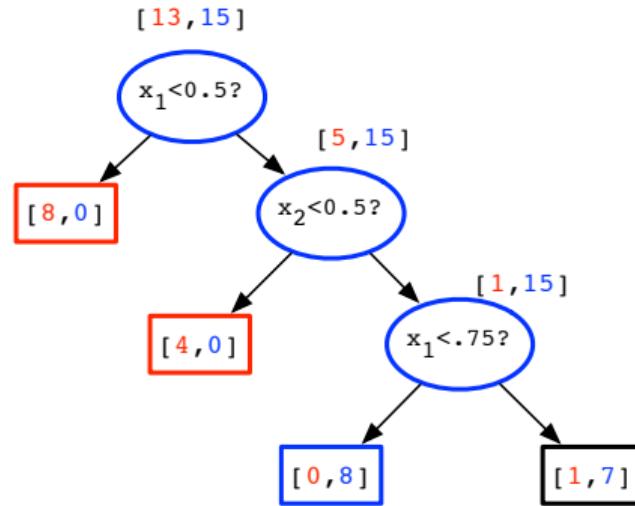
Overfitting

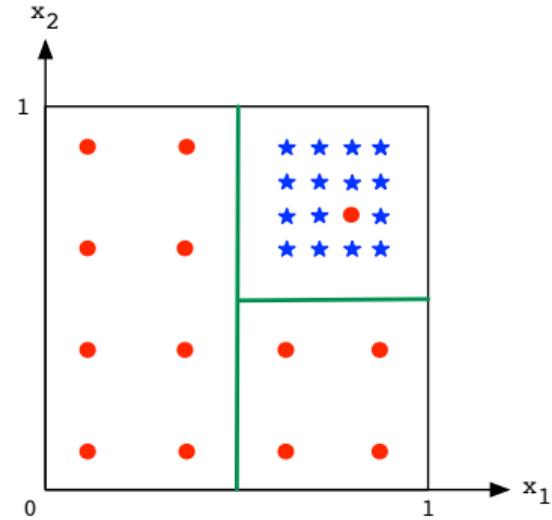
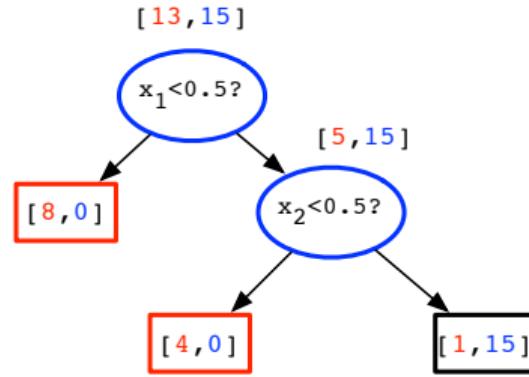
- ▶ The training error is **zero**.
 - ▶ We might be **overfitting**.
- ▶ (One) **solution**: rewind a few steps.



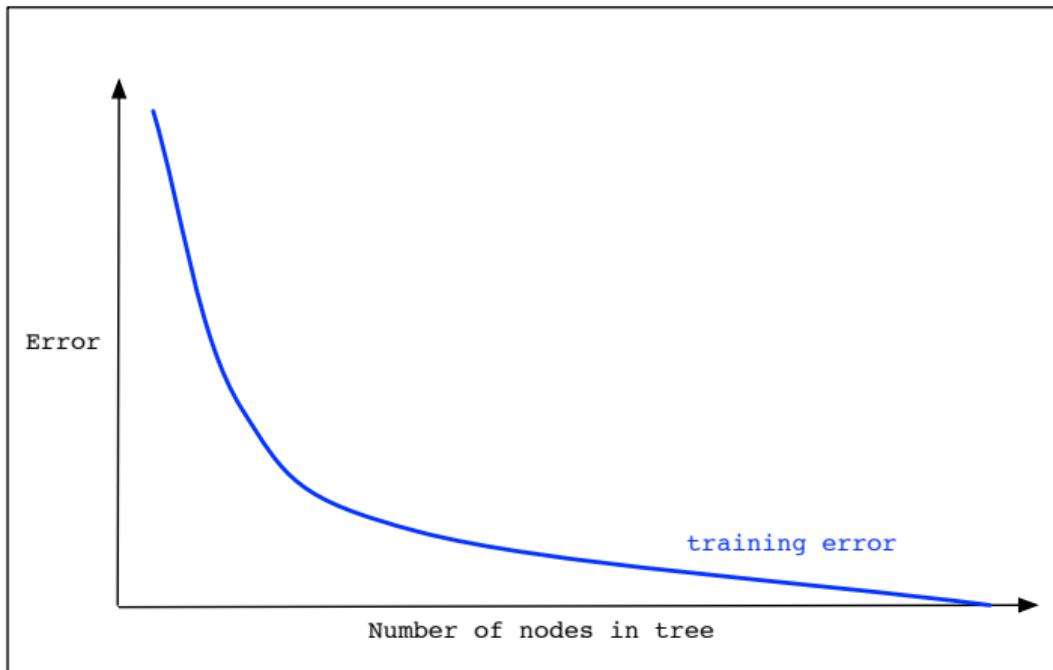




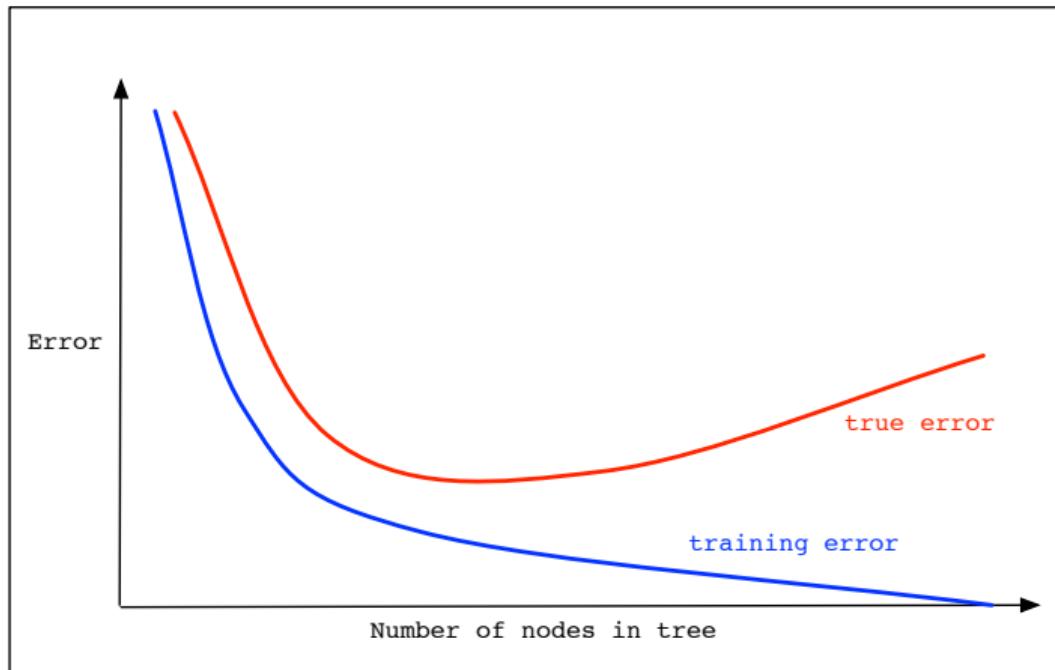




Overfitting



Overfitting



Two Strategies

- ▶ **Pruning**: simplify already-constructed tree.
- ▶ **Early-stopping**: stop early.

Pruning

- ▶ Given a full decision tree.
- ▶ Starting with predecessors of leaf nodes, replace node by most common class.
- ▶ If the change reduces validation error, keep it.
Otherwise reverse it.

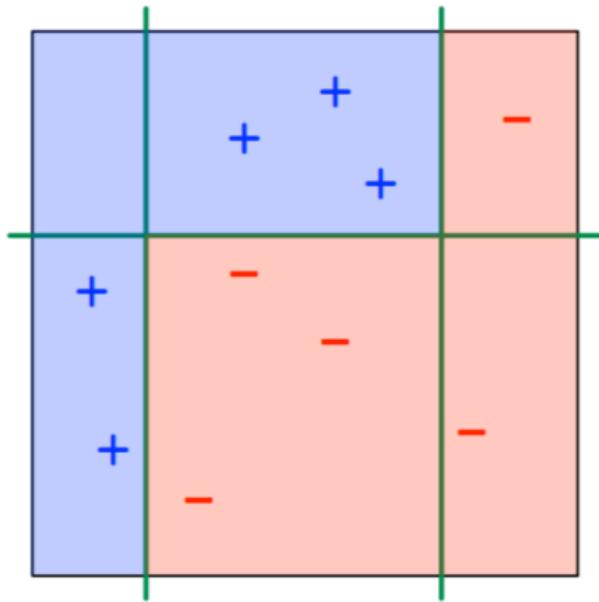
Early-Stopping

- ▶ Stop recursion when:
 - ▶ node is “pure enough” (uncertainty is low).
 - ▶ tree is too deep.

Decision Tree Properties

Very expressive:

- ▶ Can accommodate any type of data
 - ▶ numerical, Boolean, etc.
- ▶ Can accommodate any number of classes
- ▶ Can perfectly fit any data set
 - ▶ If data has no duplicates from different classes.
 - ▶ **Danger!** Overfitting!



CSE 151A
Intro to Machine Learning

Lecture 14 – Part 01
Boosting

So Far in CSE 151A

- ▶ Learn a single (sometimes complex) model:
 - ▶ Logistic Regression
 - ▶ SVMs
 - ▶ LDA/QDA
 - ▶ Decision Trees
 - ▶ ...

Today

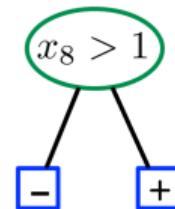
- ▶ Can we **combine** very **simple** models and get good results?
- ▶ **Yes: boosting.**

Weak Learners

- ▶ A **weak classifier** is one which performs only a little better than chance.
- ▶ A learning algorithm capable of consistently producing weak classifiers is called a **weak learner**.
- ▶ Usually very simple, fast.

Example

- ▶ A **decision stump** is a **weak classifier**.



- ▶ **Weak learner:** the strategy discussed last time for picking question.

Example

- ▶ The full decision tree learning algorithm is a **strong learner**.

The Question

- ▶ Can we “boost” the quality of a weak learner?

Boosting: The Idea

- ▶ Train a weak classifier, $H_1 : \mathcal{X} \rightarrow [-1, 1]$.
- ▶ Increase weight (importance) of misclassified points, train another classifier H_2 .
- ▶ Repeat, creating more classifiers, updating weights.
- ▶ Final classifier: a linear combination of H_1, \dots, H_k .

The Details

- ▶ **Q1:** How do we measure the performance of a classifier on a weighted data set?
- ▶ **Q2:** How do we update the point weights?
- ▶ **Q3:** How do we combine the classifiers?

AdaBoost

- ▶ Yoav Freund (UCSD) and Robert Schapire.
- ▶ A theoretically-sound answer to these questions.

Q1: Measuring Performance

- ▶ Suppose weights at step t are in $\vec{w}^{(t)}$.
 - ▶ Assume normalized s.t. weights add to one.
- ▶ We use weights to learn a classifier
 $H_t : \mathcal{X} \rightarrow [-1, 1]$.
- ▶ The “margin”:

$$r_t = \sum_{i=1}^n \omega_i^{(t)} y_i H_t(\vec{x}^{(i)}) \in [-1, 1]$$

Q1: Measuring Performance

- ▶ The **performance** of H_t :

$$\alpha_t = \frac{1}{2} \ln \frac{1 + r_t}{1 - r_t}$$

Q2: Updating Weights

- ▶ We use weights to learn a classifier
 $H_t : \mathcal{X} \rightarrow [-1, 1]$.
- ▶ Weigh misclassified points more heavily.
- ▶ Point is misclassified if $y_i H_t(\vec{x}^{(i)}) < 0$

Q2: Updating Weights

- ▶ This will do the trick:

$$\omega_i^{(t+1)} \propto \omega_i^{(t)} \cdot \exp(-\alpha_t y_i H_t(\vec{x}^{(i)}))$$

- ▶ \propto because we normalize.

Q3: Combining Classifiers

- ▶ The final classifier:

$$H_t(\vec{x}) = \sum_{t=1}^T \alpha_t H_t(\vec{x})$$

AdaBoost

Given data $(\vec{x}^{(1)}, y_1), \dots, (\vec{x}^{(n)}, y_n)$, labels in $\{-1, 1\}$.

- ▶ Initialize weight vector, $\vec{\omega}^{(1)} = (\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n})^T$
- ▶ Repeat:
 - ▶ Give data and weights $\vec{\omega}^{(t)}$ to weak learner. Receive a classifier, $H_t : \mathcal{X} \rightarrow \{-1, 1\}$ back.
 - ▶ Calculate “performance”, $\alpha_t = \frac{1}{2} \ln \frac{1+r_t}{1-r_t}$
 - ▶ Update $\vec{\omega}^{(t+1)} \propto \omega_i^{(t)} \cdot \exp(-\alpha_t y_i H_t(\vec{x}^{(i)}))$
- ▶ Final classifier: $H_t(\vec{x}) = \sum_{t=1}^T \alpha_t H_t(\vec{x})$

Example: Decision Stumps

- ▶ To learn decision stump, given data and $\vec{\omega}^{(t)}$.
- ▶ Try all features, thresholds.
- ▶ Choose that which maximizes the margin:

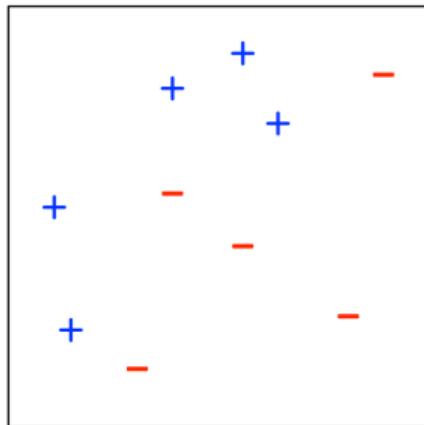
$$r_t = \sum_{i=1}^n \omega_i^{(t)} y_i H_t(\vec{x}^{(i)}) \in [-1, 1]$$

Example: Decision Stumps

- ▶ To learn decision stump, given data and $\vec{\omega}^{(t)}$.
- ▶ Try all features, thresholds.
- ▶ Equivalently, choose that which maximizes the performance:

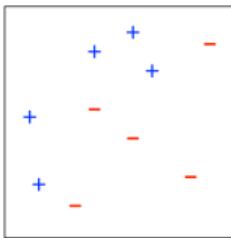
$$\alpha_t = \frac{1}{2} \ln \frac{1 + r_t}{1 - r_t}$$

Example

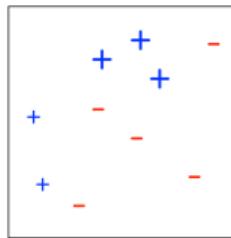


Example

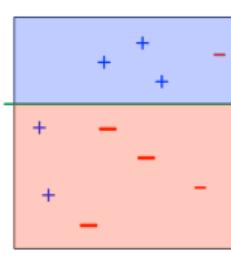
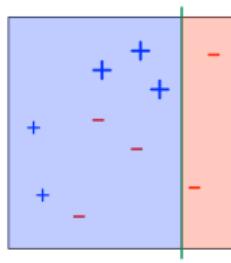
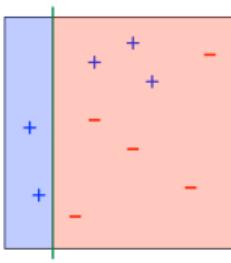
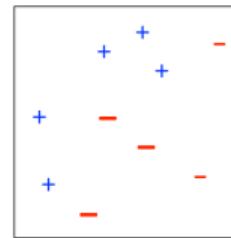
D_1



D_2



D_3



h_1

$$r_1 = 0.40, \alpha_1 = 0.42$$

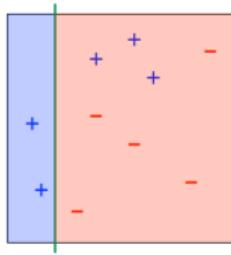
h_2

$$r_2 = 0.58, \alpha_2 = 0.65$$

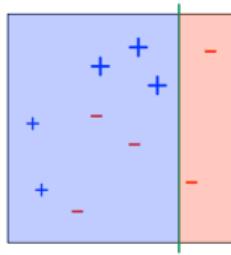
h_3

$$r_3 = 0.72, \alpha_3 = 0.92$$

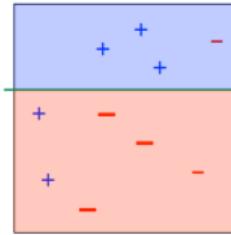
Example



$$h_1 \\ \alpha_1 = 0.42$$



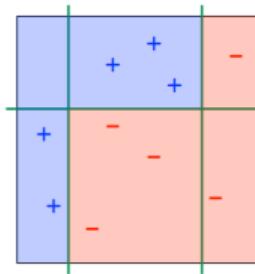
$$h_2 \\ \alpha_2 = 0.65$$



$$h_3 \\ \alpha_3 = 0.92$$

Final classifier:

$$\text{sign}(0.42h_1(x) + 0.65h_2(x) + 0.92h_3(x))$$

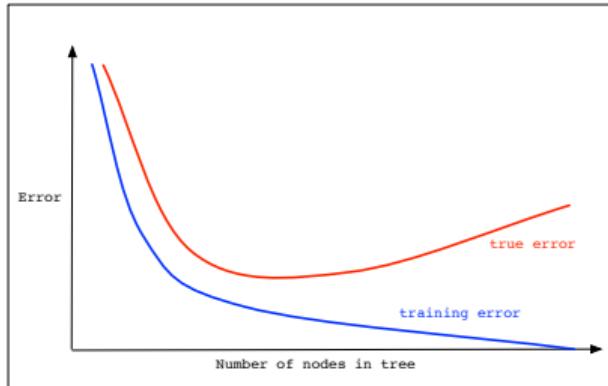


Theory

Suppose that on each round t , the weak learner returns a rule H_t whose error on the step t weighted data is $\leq \frac{1}{2} - \gamma$. Then after T rounds, the training error of the combined rule H is at most $e^{-\gamma^2 T / 2}$.

Generalization

- ▶ Boosted decision stumps are really resistant to overfitting.



Generalization

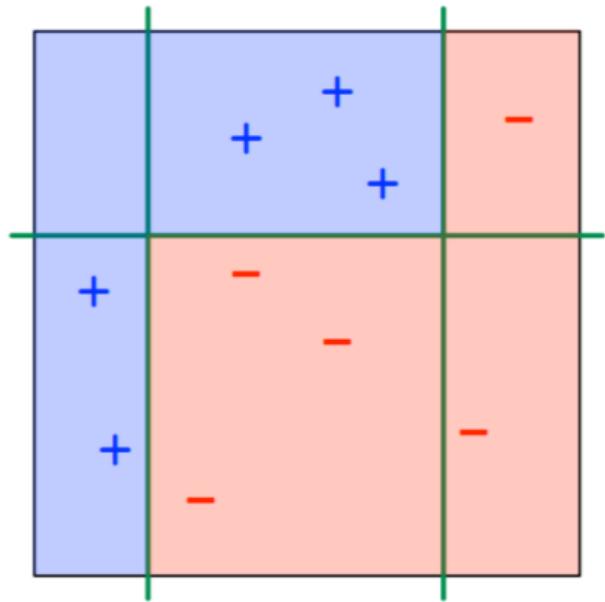
- ▶ Boosted decision stumps are really resistant to overfitting.

Why not?

- ▶ Why use weak learners?
- ▶ What if we replace decision stumps with SVMs or logistic regression?

Why not?

- ▶ Why use weak learners?
- ▶ What if we replace decision stumps with SVMs or logistic regression?
- ▶ You can, but weak learners are **fast** to learn.
- ▶ The point of boosting is that weak learners are “just as good” as strong learners.



CSE 151A
Intro to Machine Learning

Lecture 14 – Part 02
Random Forests

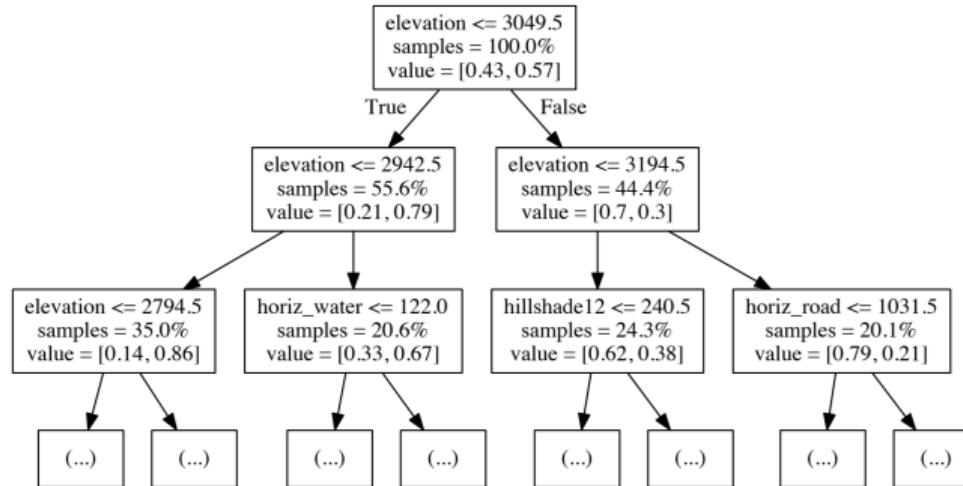
Let's Try

- ▶ Decision trees are susceptible to overfitting.
- ▶ Let's try using boosted decision trees.

Example: Forest Cover Type

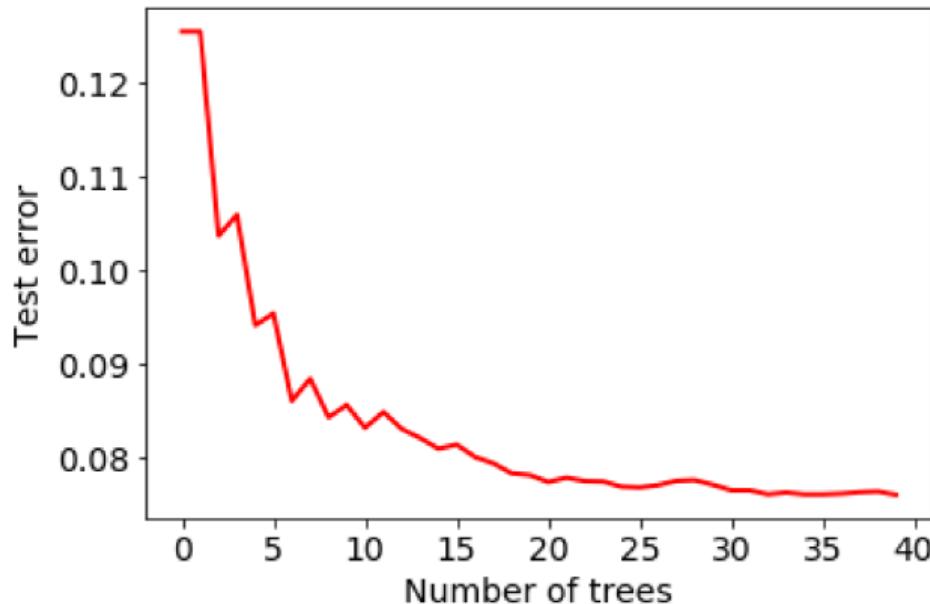
- ▶ **Goal:** predict forest type.
 - ▶ Spruce-fir
 - ▶ Lodgepole pine
 - ▶ etc. 7 classes in total.
- ▶ 54 cartographic/geological features.
 - ▶ Elevation, slope, amount of shade, distance to water, etc.

Decision Tree

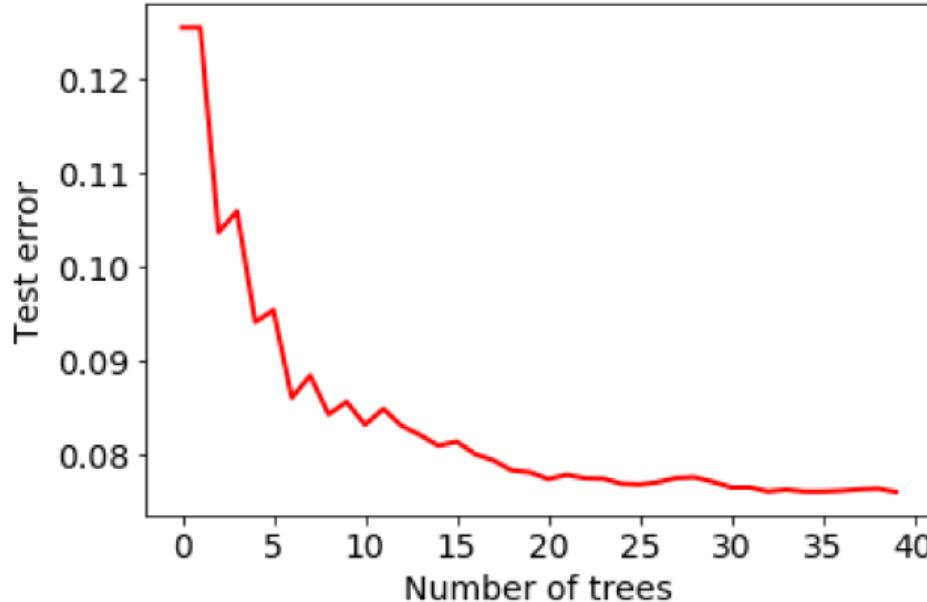


Depth 20. Training error: 1%. Test error: 12.6%.

Boosted Decision Trees



Boosted Decision Trees



Depth 20: Test error: 8.7%. **Slow!**

Another Idea

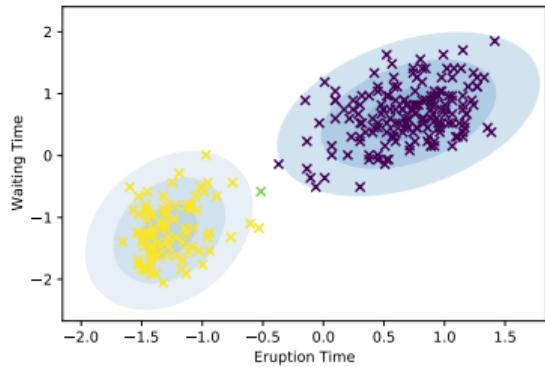
- ▶ Prevent decision trees from overfitting by “hiding data” randomly.
- ▶ Train a bunch of trees, quickly.
- ▶ Average them to make a final prediction.

Random Forests

- ▶ For $t = 1$ to T
 - ▶ Choose n' training points randomly, with replacement.
 - ▶ Fit a decision tree, H_t .
 - ▶ At each node, restrict to one of k features, chosen randomly.
- ▶ Final classifier: majority vote of H_1, \dots, H_T .
- ▶ Common settings: $n' = n$ (bootstrap), $k = \sqrt{d}$.

Forest Cover Type

- ▶ Decision trees: 12.6% error.
- ▶ Boosted decision trees: 8.7% error (but **slow!**)
- ▶ Random forests: 8.8% error.
 - ▶ 50% of features dropped.
 - ▶ Each individual tree H_1, \dots, H_t has test error around 15%.



CSE 151A

Intro to Machine Learning

Lecture 15 – Part 01

Supervised and Unsupervised Learning

Supervised Learning

- ▶ We tell the machine the “right answer”.
 - ▶ There is a **ground truth**.
- ▶ Data set: $\{(\vec{x}^{(i)}, y_i)\}$.
- ▶ **Goal:** learn relationship between features $\vec{x}^{(i)}$ and labels y_i .
- ▶ **Examples:** classification, regression.

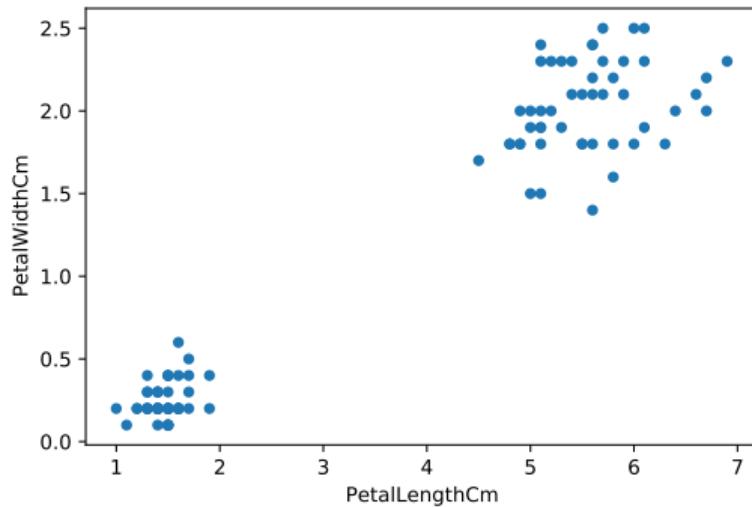
Unsupervised Learning

- ▶ We don't tell the machine the “right answer”.
 - ▶ In fact, there might not be one!
- ▶ Data set: $\vec{x}^{(i)}$ (usually no test set)
- ▶ **Goal:** learn the **structure** of the data itself.
 - ▶ To discover something, for compression, to use as a feature later.
- ▶ **Example:** **clustering**

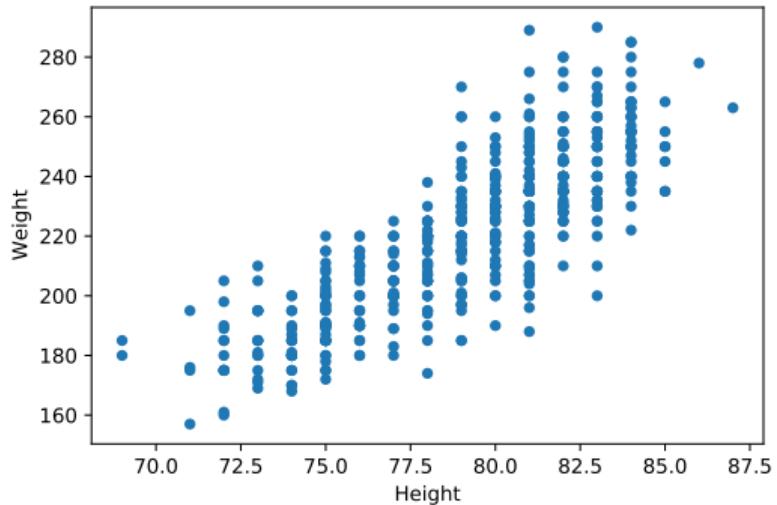
Example

- ▶ We gather measurements $\vec{x}^{(i)}$ of a bunch of flowers.
- ▶ **Question:** how many species are there?
- ▶ **Goal:** **cluster** the similar flowers into groups.

Example



Example

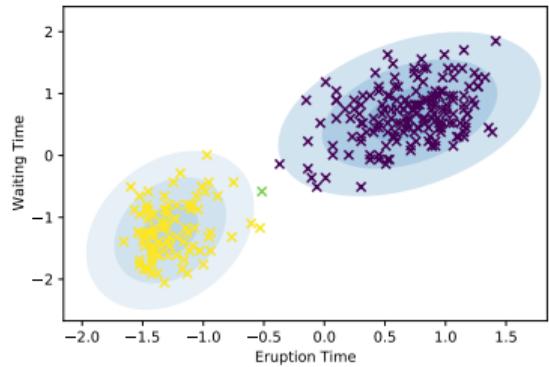


Clustering and Dimensionality

- ▶ Groups emerge with more features.
- ▶ But too many features, and groups disappear.
 - ▶ **Curse of dimensionality.**
- ▶ **Also:** We can't see in $d > 3$.

Ground Truth

- ▶ If we don't have labels, we can't measure accuracy.
- ▶ Sometimes, labels don't exist.
- ▶ Example: cluster customers into types by previous purchases.

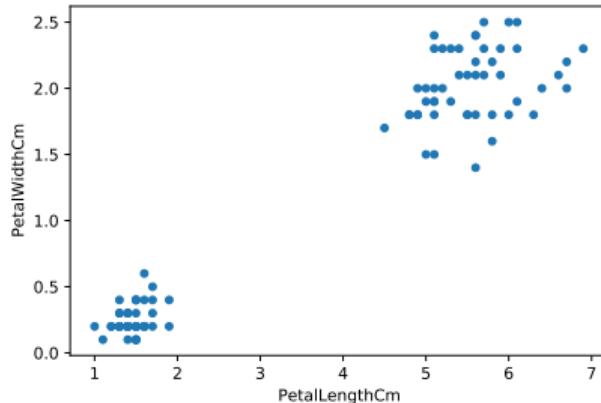


CSE 151A
Intro to Machine Learning

Lecture 15 – Part 02
K-Means Clustering

Learning

- ▶ **Goal:** turn clustering into optimization problem.
- ▶ **Idea:** clustering is like **compression**



K-Means Objective

- ▶ **Given:** data, $\{\vec{x}^{(i)}\} \in \mathbb{R}^d$ and a parameter k .
- ▶ **Find:** k cluster centers $\vec{\mu}^{(1)}, \dots, \vec{\mu}^{(k)}$ so that the average squared distance from a data point to nearest cluster center is small.
- ▶ The **k-means objective function**:

$$\text{Cost}(\vec{\mu}^{(1)}, \dots, \vec{\mu}^{(k)}) = \frac{1}{n} \sum_{i=1}^n \min_{j \in \{1, \dots, k\}} \|\vec{x}^{(i)} - \vec{\mu}^{(j)}\|^2$$

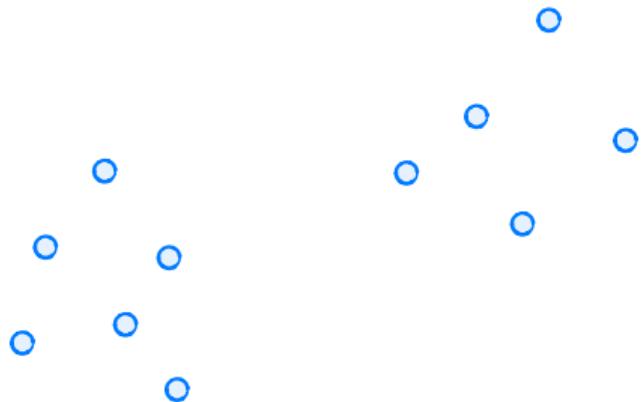
Optimization

- ▶ **Goal:** find $\vec{\mu}^{(1)}, \dots, \vec{\mu}^{(k)}$ minimizing k -means objective function.
- ▶ **Problem:** this is NP-Hard.
- ▶ We use a heuristic instead of solving exactly.

Lloyd's Algorithm for K-Means

- ▶ Initialize centers, $\vec{\mu}^{(1)}, \dots, \vec{\mu}^{(k)}$ somehow.
- ▶ Repeat until convergence:
 - ▶ Assign each point $\vec{x}^{(i)}$ to closest center
 - ▶ Update each $\vec{\mu}^{(i)}$ to be mean of points assigned to it

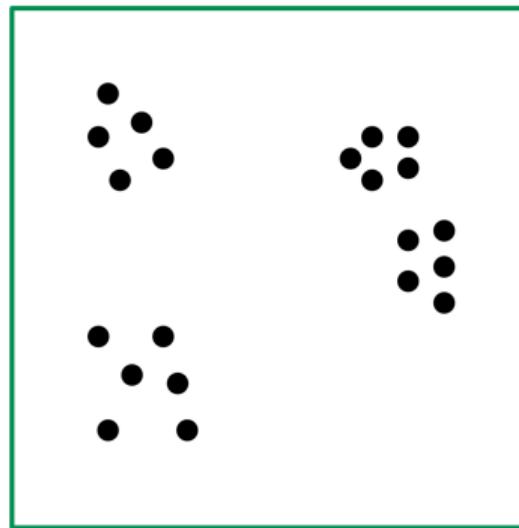
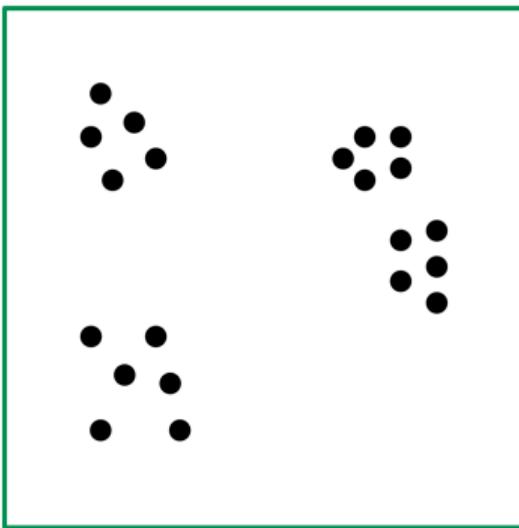
Example



Theory

- ▶ Each iteration reduces cost.
- ▶ This guarantees convergence to a **local** min.
- ▶ Initialization is very important.

Example



Initialization Strategies

- ▶ Basic Approach: Pick k data points at random.
- ▶ Better Approach: **k-means++**:
 - ▶ Pick first center at random from data.
 - ▶ Let $C = \{\vec{\mu}^{(1)}\}$ (centers chosen so far)
 - ▶ Repeat $k - 1$ more times:
 - ▶ Pick random data point \vec{x} according to distribution

$$\mathbb{P}(\vec{x}) \propto \min_{\vec{\mu} \in C} \|\vec{x} - \mu\|^2$$

- ▶ Add \vec{x} to C

Picking k

- ▶ How do we know how many clusters the data contains?

Plot of K-Means Objective

Applications of K-Means

- ▶ Discovery
- ▶ Vector Quantization
 - ▶ Find a finite set of representatives of a large (possibly infinite) set.

Example #1

- ▶ Cluster animal descriptions.
- ▶ 50 animals: grizzly bear, dalmatian, rabbit, pig, ...
- ▶ 85 attributes: long neck, tail, walks, swims, ...
- ▶ 50 data points in \mathbb{R}^{85} . Run k -means with $k = 10$

Results

- | | |
|--|--|
| <ul style="list-style-type: none">❶ zebra❷ spider monkey, gorilla, chimpanzee❸ tiger, leopard, wolf, bobcat, lion❹ hippopotamus, elephant, rhinoceros❺ killer whale, blue whale, humpback whale, seal, walrus, dolphin❻ giant panda❼ skunk, mole, hamster, squirrel, rabbit, bat, rat, weasel, mouse, raccoon❽ antelope, horse, moose, ox, sheep, giraffe, buffalo, deer, pig, cow❾ beaver, otter❿ grizzly bear, dalmatian, persian cat, german shepherd, siamese cat, fox, chihuahua, polar bear, collie | <ul style="list-style-type: none">❶ zebra❷ spider monkey, gorilla, chimpanzee❸ tiger, leopard, fox, wolf, bobcat, lion❹ hippopotamus, elephant, rhinoceros, buffalo, pig❺ killer whale, blue whale, humpback whale, seal, otter, walrus, dolphin❻ dalmatian, persian cat, german shepherd, siamese cat, chihuahua, giant panda, collie❼ beaver, skunk, mole, squirrel, bat, rat, weasel, mouse, raccoon❽ antelope, horse, moose, ox, sheep, giraffe, deer, cow❾ hamster, rabbit❿ grizzly bear, polar bear |
|--|--|

Example #2

- ▶ How do we represent images of different sizes as fixed length feature vectors for use in classification tasks?



Visual Bags-of-Words

- ▶ **Idea:** build a “dictionary” of image patches.
- ▶ Extract all $\ell \times \ell$ image patches from all training images.
- ▶ Cluster them with k -means.
 - ▶ Each cluster center is now a dictionary “word”
- ▶ Represent an image as a histogram over $\{1, 2, \dots, k\}$ by associating each patch with nearest center.

Online Learning

- ▶ What if the dataset is huge?
 - ▶ It doesn't even fit in memory.
- ▶ What if we're continuously getting new data?
 - ▶ Don't want to retrain with every new point.
- ▶ We can update the model **online**.

Sequential k-Means

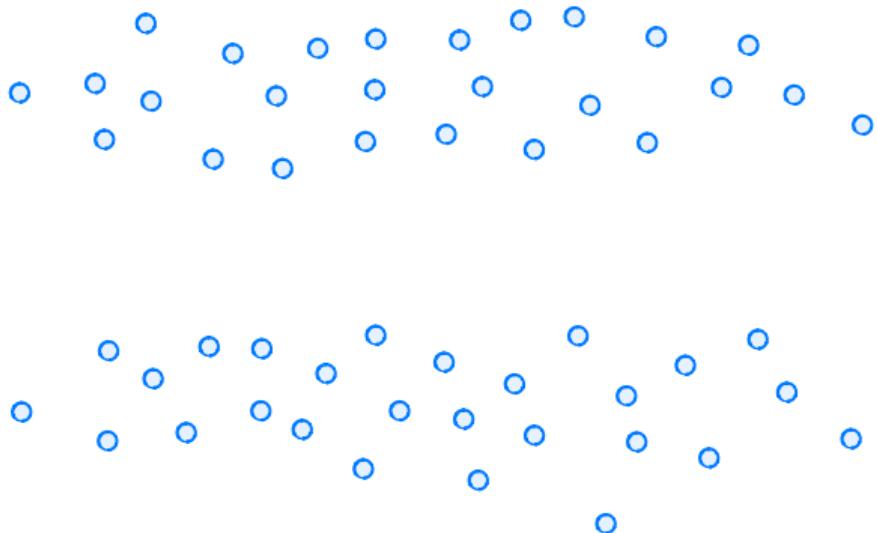
- ▶ Set the centers $\vec{\mu}^{(1)}, \dots, \vec{\mu}^{(k)}$ to be first k points
- ▶ Set counts to be $n_1 = n_2 = \dots = n_k = 1$.
- ▶ Repeat:
 - ▶ Get next data point, \vec{x}
 - ▶ Let $\vec{\mu}^{(j)}$ be closest center
 - ▶ Update $\vec{\mu}^{(j)}$ and n_j :

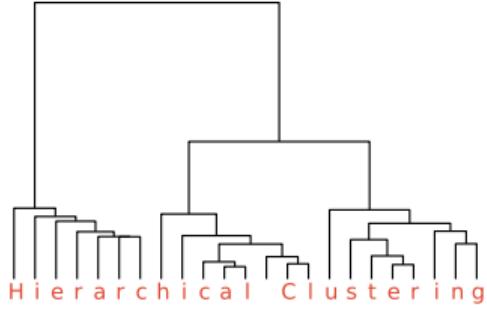
$$\vec{\mu}^{(j)} = \frac{n_j \vec{\mu}^{(j)} + \vec{x}}{n_j + 1} \quad n_j = n_j + 1$$

K-Means

- ▶ Perhaps the most popular clustering algorithm.
- ▶ **Fast, easy to understand.**
- ▶ **Assumes spherical clusters.**

Example





CSE 151A
Intro to Machine Learning

Lecture 16 – Part 01
Gaussian Mixtures

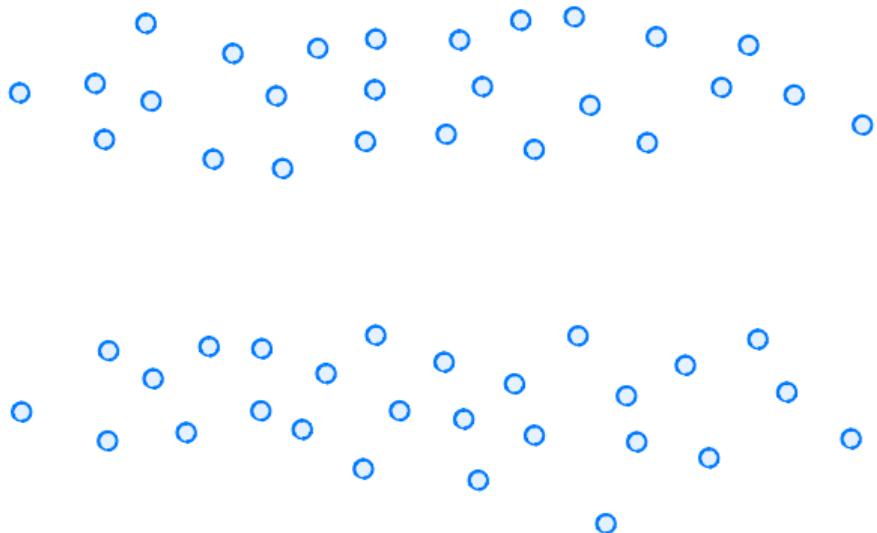
Announcements

- ▶ Please submit regrade requests for yellows!
- ▶ No class on Monday.

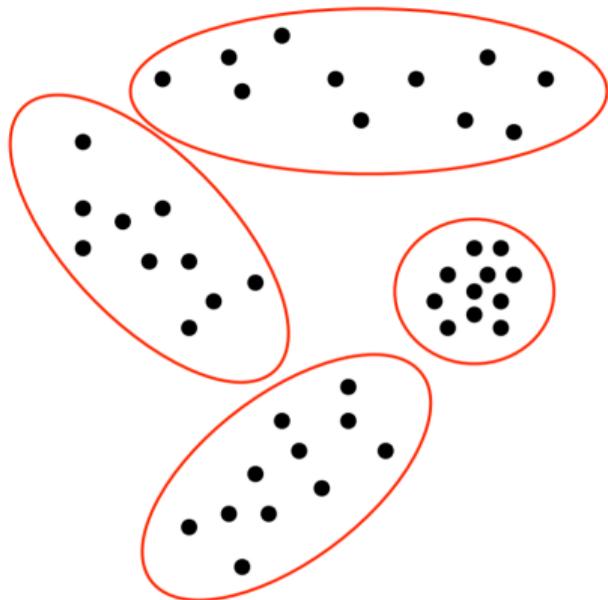
K-Means

- ▶ Perhaps the most popular clustering algorithm.
- ▶ **Fast, easy to understand.**
- ▶ **Assumes spherical clusters.**

Example



Mixtures of Gaussians



Each cluster is specified by:

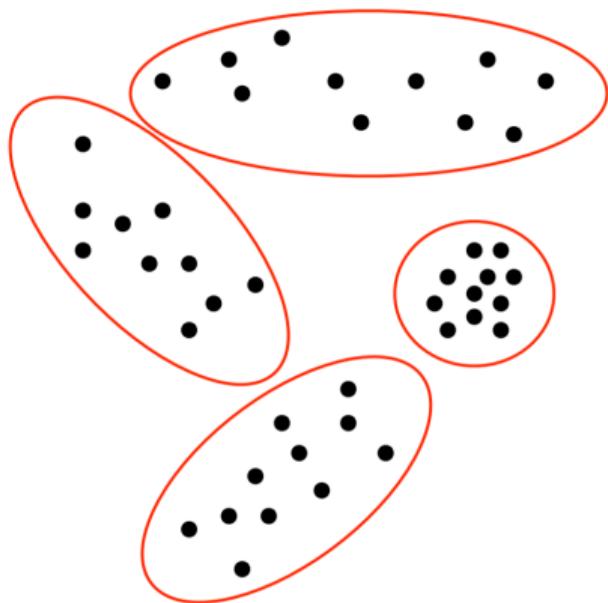
- ▶ a Gaussian $P_i = \mathcal{N}(\vec{\mu}^{(i)}, C_i)$
- ▶ a mixing weight π_i

Mixture distribution:

$$\mathbb{P}(\vec{x}) = \sum_{i=1}^k \pi_i P_i(\vec{x})$$

Interpretation

- ▶ **Soft-assignment:** each point belongs to multiple Gaussians



Responsibility of cluster j for point i :

$$w_{ij} = \mathbb{P}(\text{cluster } j | \vec{x}^{(i)})$$
$$= \frac{\pi_j \mathbb{P}_j(\vec{x}^{(i)})}{\sum_l \pi_l \mathbb{P}_l(\vec{x}^{(i)})}$$

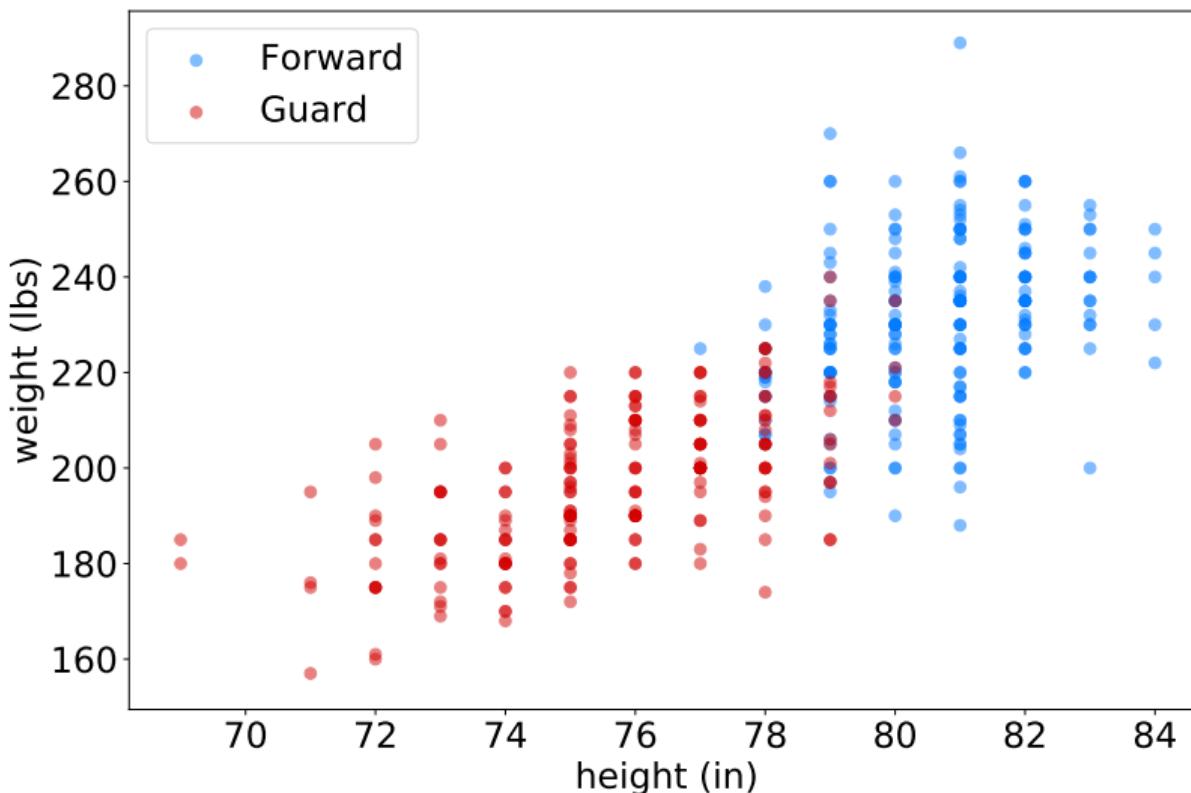
Fitting

- ▶ Recall how we fit a multivariate Gaussian.

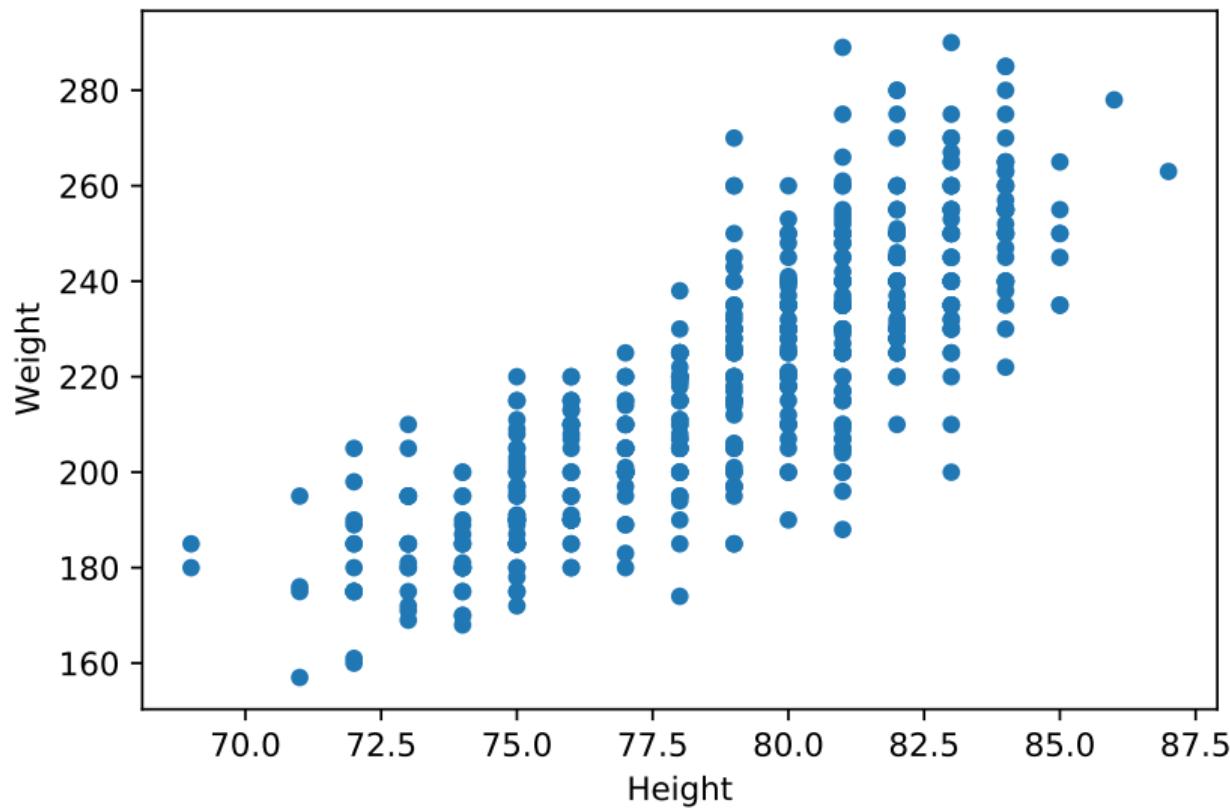
$$\vec{\mu} = \frac{1}{n} \sum_{i=1}^n \vec{x}^{(i)}$$

$$C = \frac{1}{n} \sum_{i=1}^n (\vec{x}^{(i)} - \vec{\mu})(\vec{x}^{(i)} - \vec{\mu})^T$$

Fitting



Fitting



Fitting a Mixture

- ▶ Now to fit j th Gaussian with responsibilities w_{ij} :

$$\vec{\mu}^{(j)} = \frac{1}{\sum_{i=1}^n w_{ij}} \sum_{i=1}^n w_{ij} \vec{x}^{(i)}$$

$$C_j = \frac{1}{\sum_{i=1}^n w_{ij}} \sum_{i=1}^n w_{ij} (\vec{x}^{(i)} - \vec{\mu}^{(j)}) (\vec{x}^{(i)} - \vec{\mu}^{(j)})^T$$

$$\pi_j = \frac{1}{n} \sum_{i=1}^n w_{ij}$$

Problem

- ▶ To calculate $\vec{\mu}^{(j)}$, C_j , π_j we need responsibilities w_{ij} .
- ▶ But to calculate responsibilities, we need $\vec{\mu}^{(j)}$, π_j , C_j .

Idea

- ▶ Guess $\vec{\mu}^{(j)}$, π_j , and C_j
- ▶ Use these guesses to calculate responsibilities
(i.e., make a **soft assignment**):

$$w_{ij} = \frac{\pi_j \mathbb{P}_j(\vec{x}^{(i)})}{\sum_\ell \pi_\ell \mathbb{P}_\ell(\vec{x}^{(i)})}$$

- ▶ Then update $\vec{\mu}^{(j)}$, π_j , C_j using w_{ij} . Repeat.

The EM Algorithm

- ▶ Initialize $\pi_1, \dots, \pi_j, \vec{\mu}^{(1)}, \dots, \vec{\mu}^{(k)}, C_1, \dots, C_k$
- ▶ Repeat until convergence:
 - ▶ Make soft assignment (update responsibilities):

$$w_{ij} = \frac{\pi_j \mathbb{P}_j(\vec{x}^{(i)})}{\sum_\ell \pi_\ell \mathbb{P}_\ell(\vec{x}^{(i)})}$$

- ▶ Update mixing weights, means, covariances:

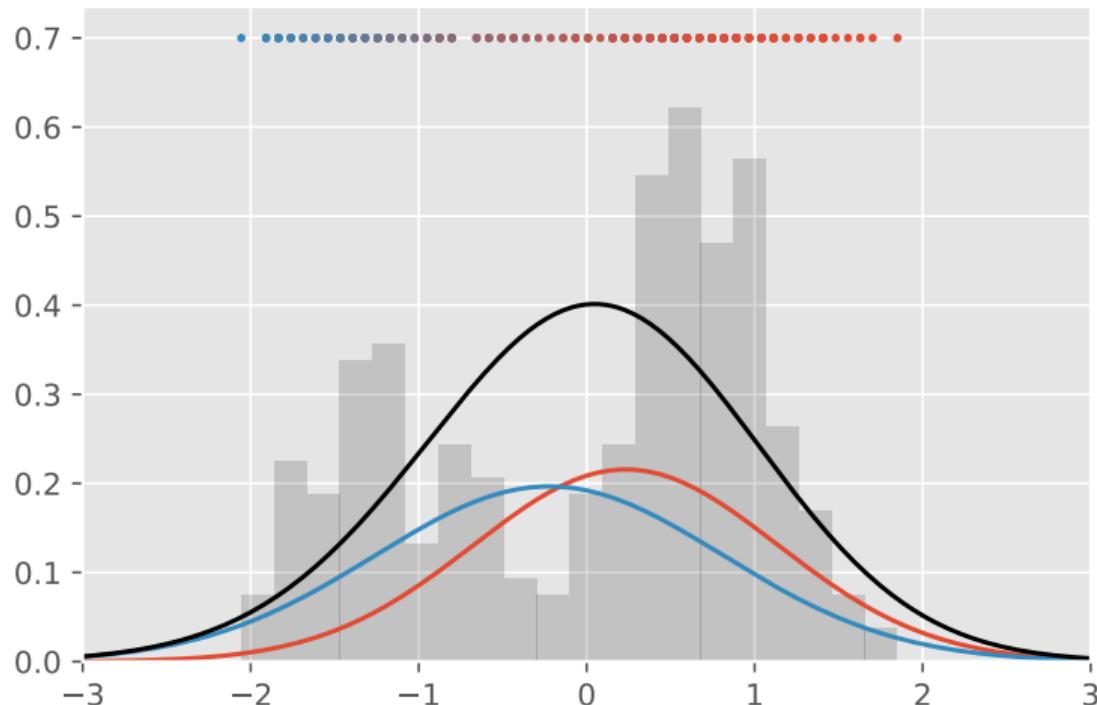
$$\vec{\mu}^{(j)} = \frac{1}{\sum_{i=1}^n w_{ij}} \sum_{i=1}^n w_{ij} \vec{x}^{(i)}$$

$$C_j = \frac{1}{\sum_{i=1}^n w_{ij}} \sum_{i=1}^n w_{ij} (\vec{x}^{(i)} - \vec{\mu}^{(j)}) (\vec{x}^{(i)} - \vec{\mu}^{(j)})^T$$

$$\pi_j = \frac{1}{n} \sum_{i=1}^n w_{ij}$$

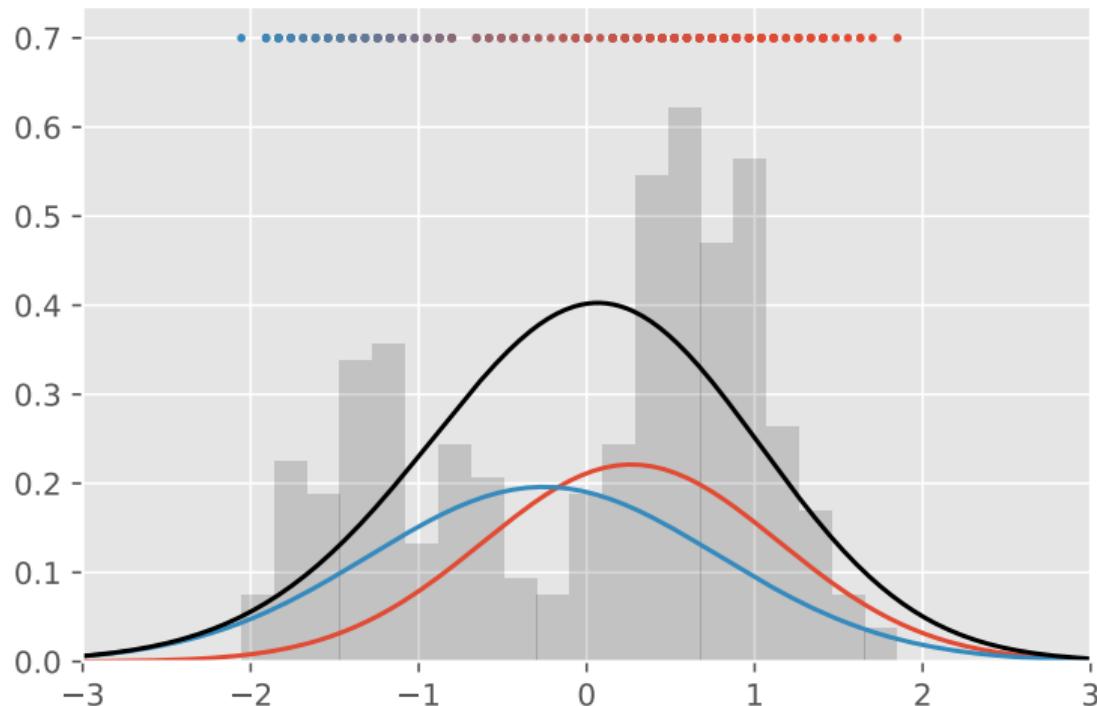
E-M Algorithm Demo

Iteration #1



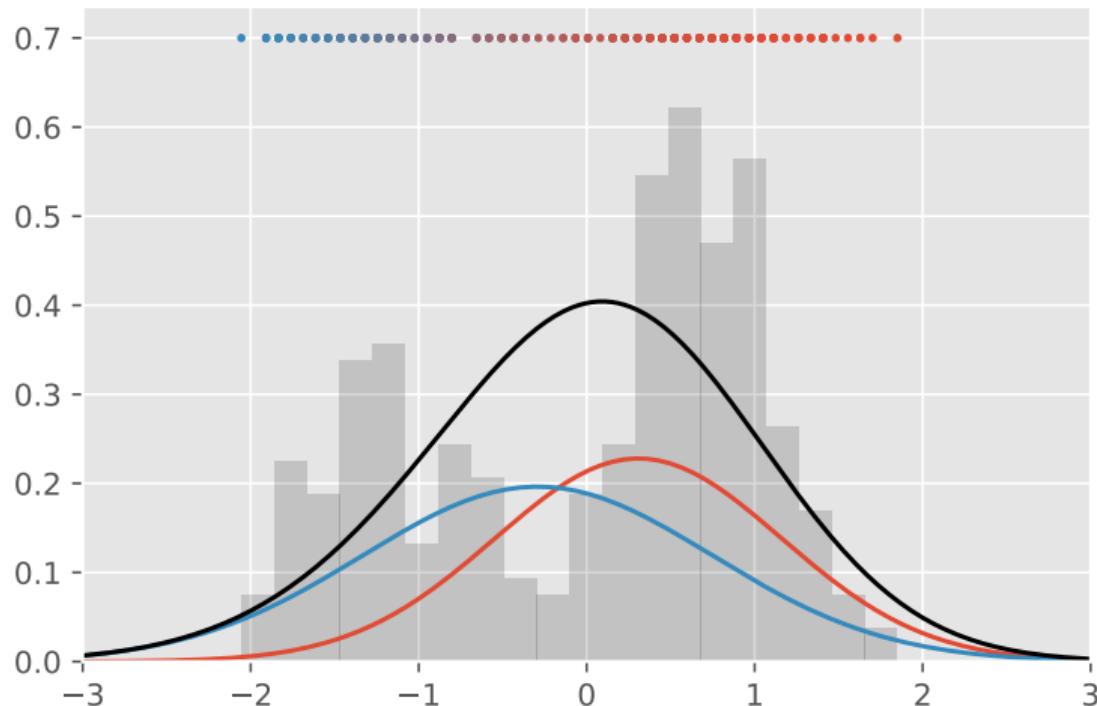
E-M Algorithm Demo

Iteration #2



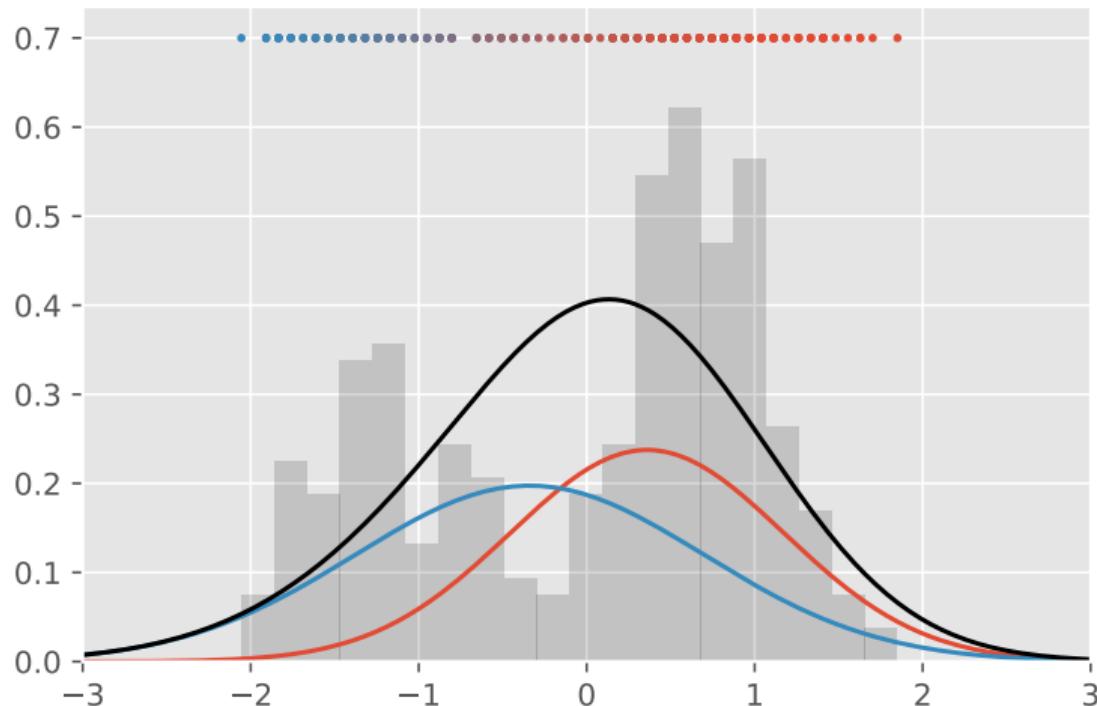
E-M Algorithm Demo

Iteration #3



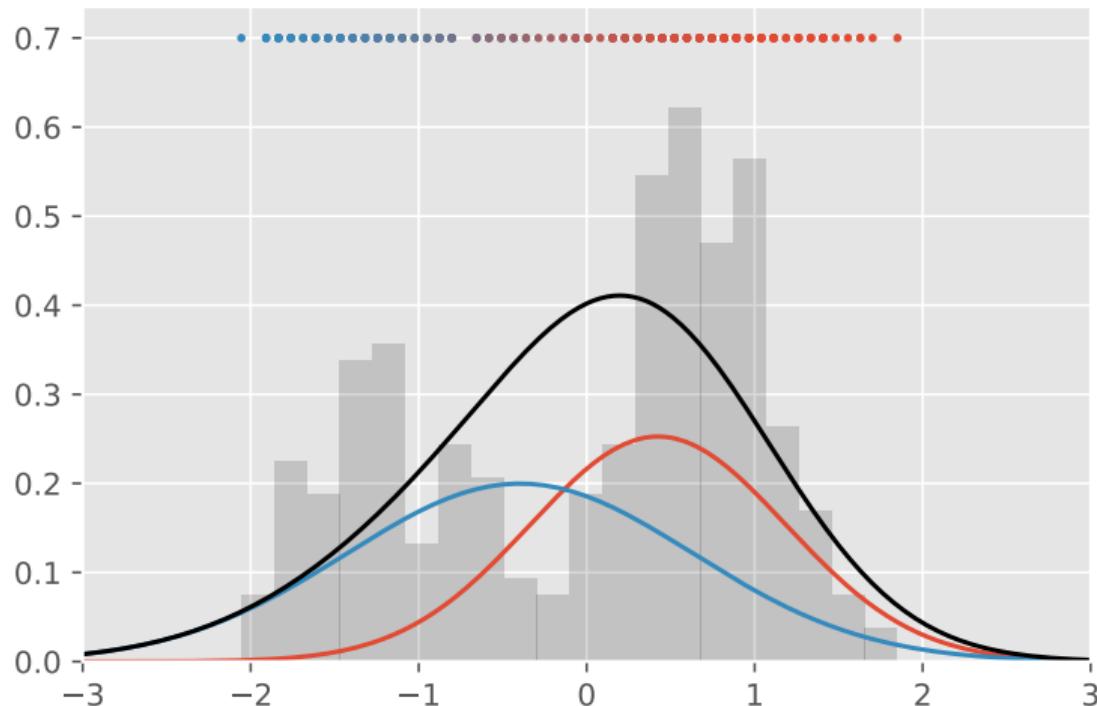
E-M Algorithm Demo

Iteration #4



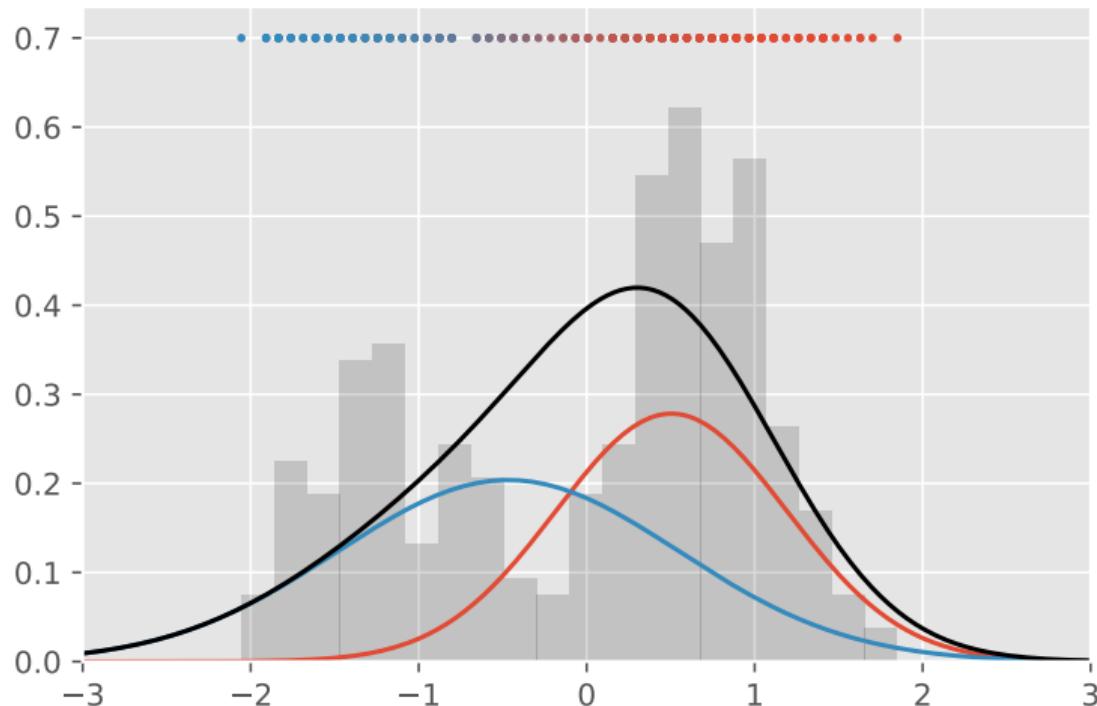
E-M Algorithm Demo

Iteration #5



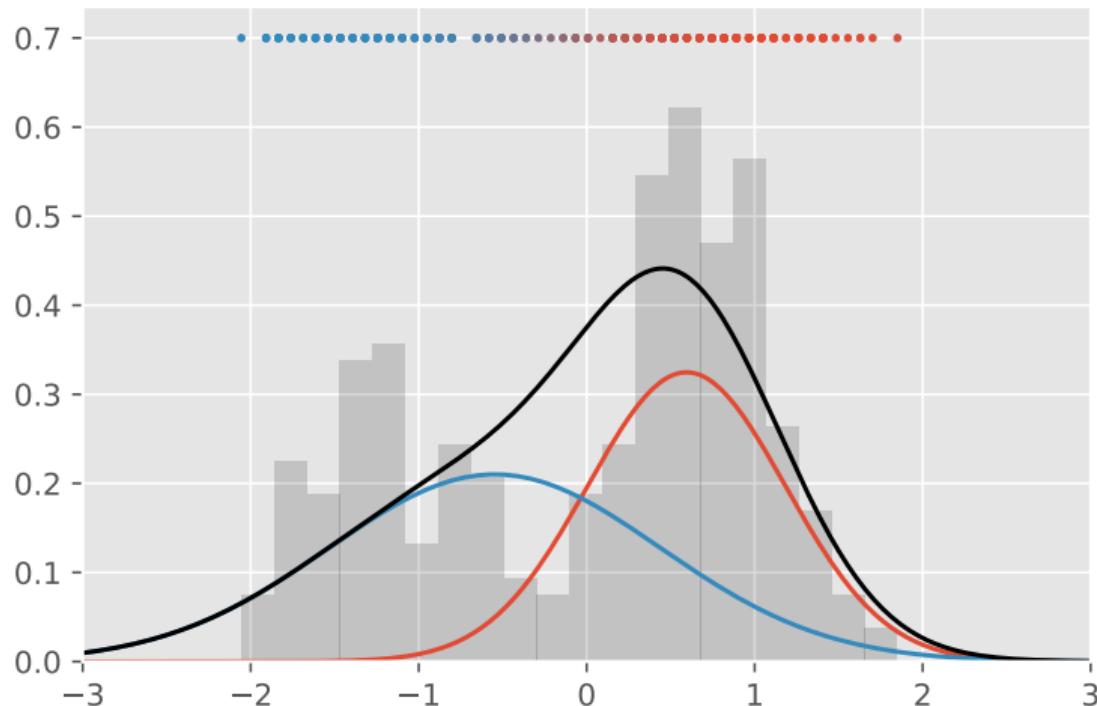
E-M Algorithm Demo

Iteration #6



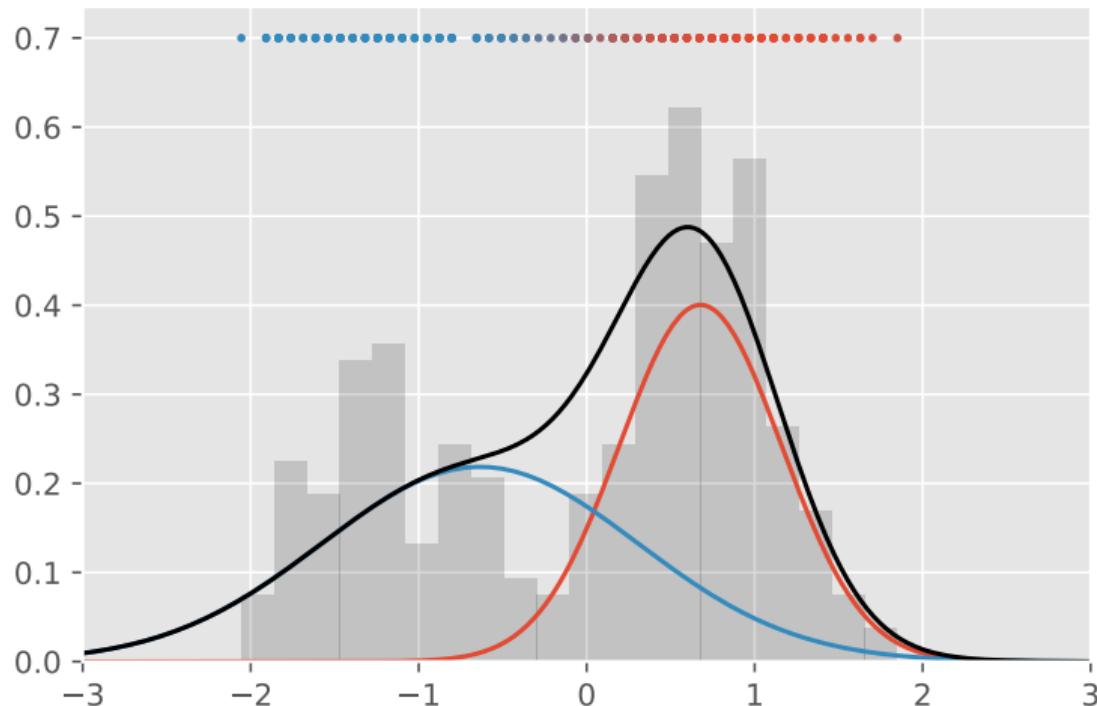
E-M Algorithm Demo

Iteration #7



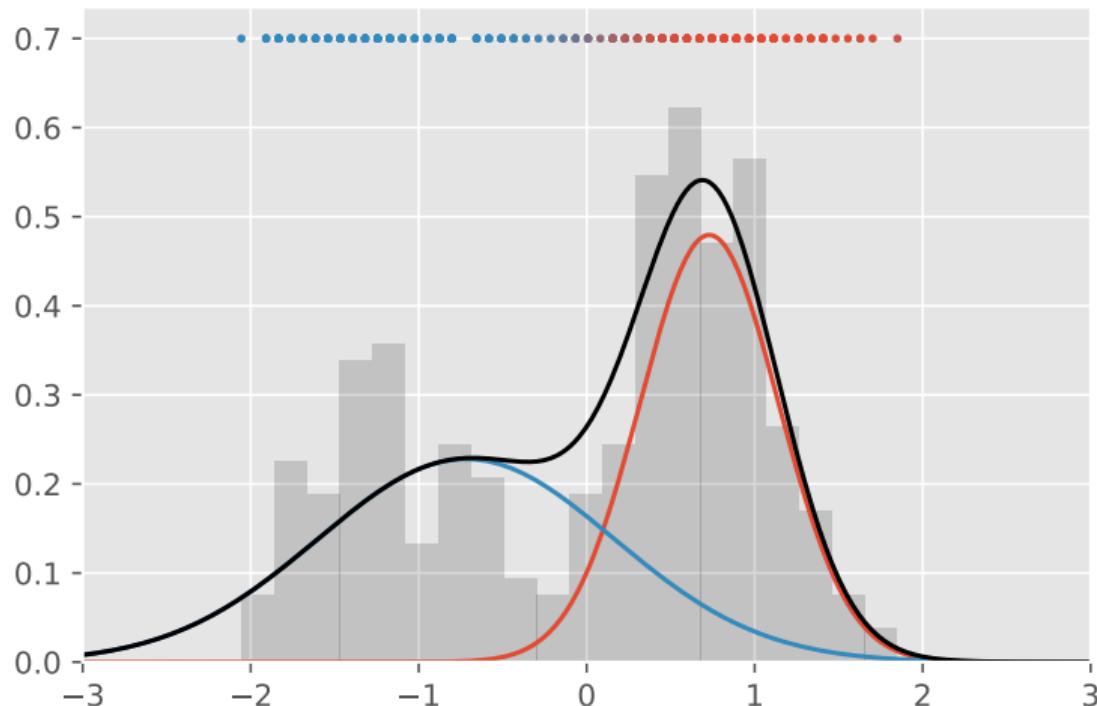
E-M Algorithm Demo

Iteration #8



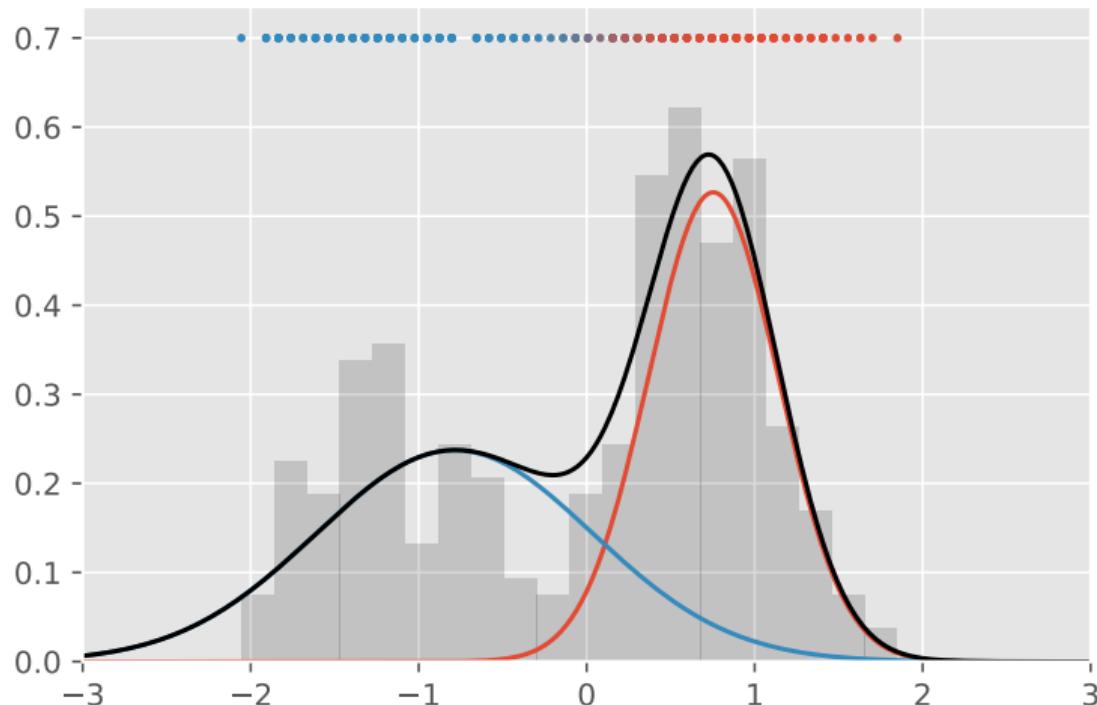
E-M Algorithm Demo

Iteration #9



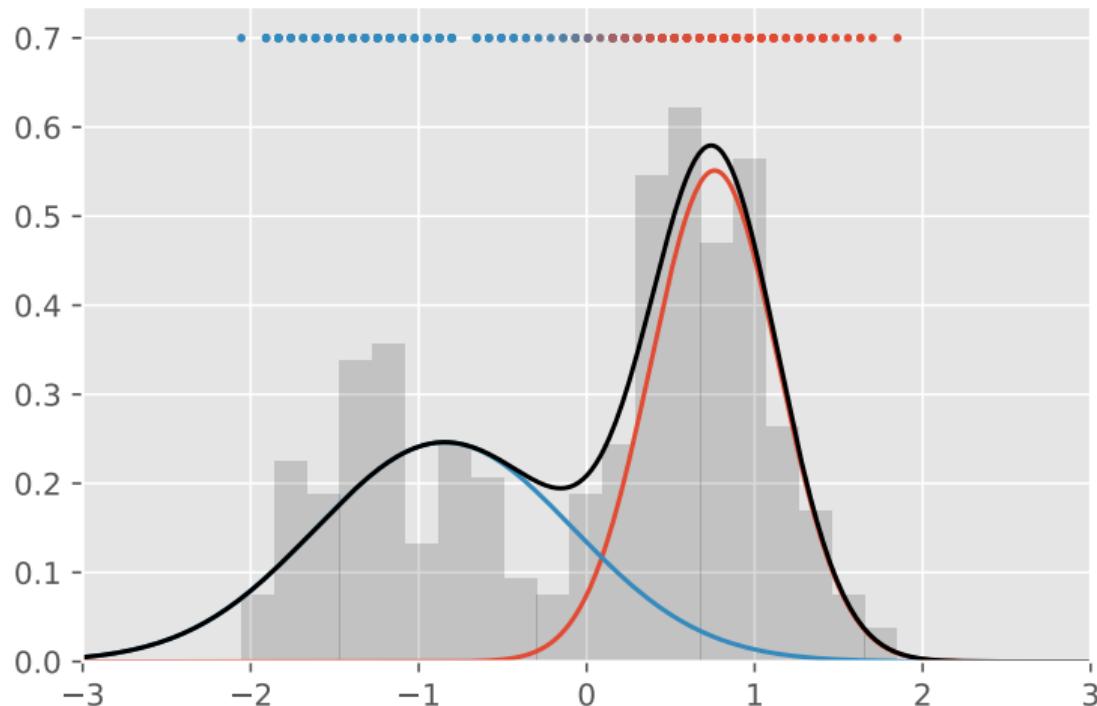
E-M Algorithm Demo

Iteration #10



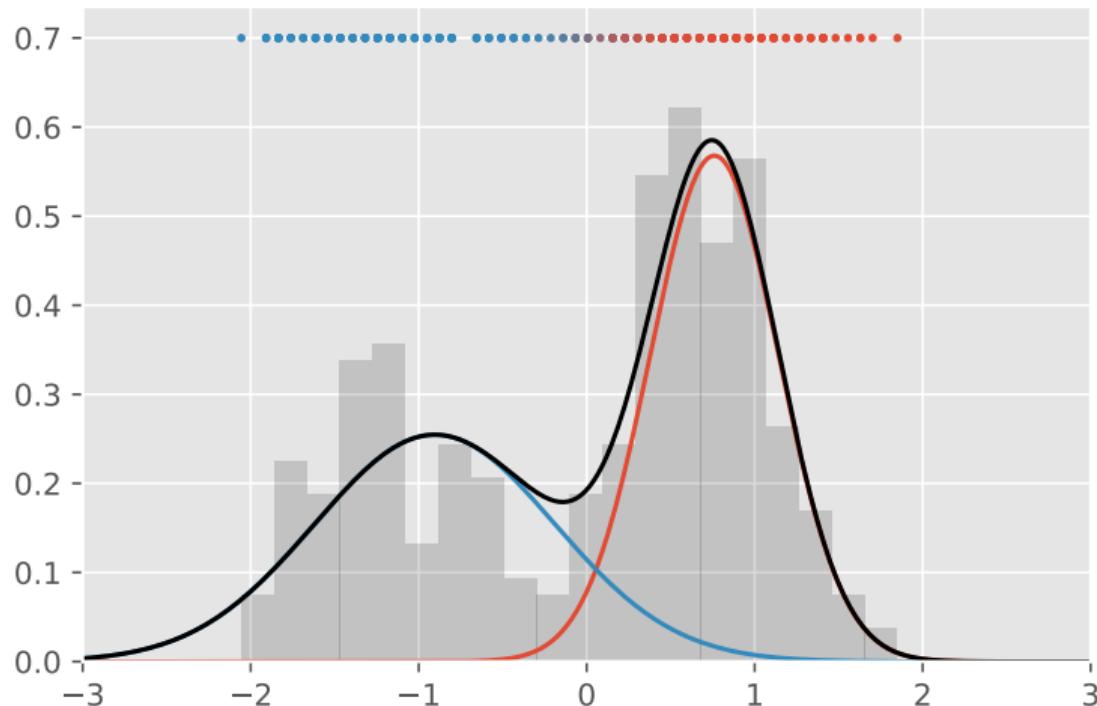
E-M Algorithm Demo

Iteration #11



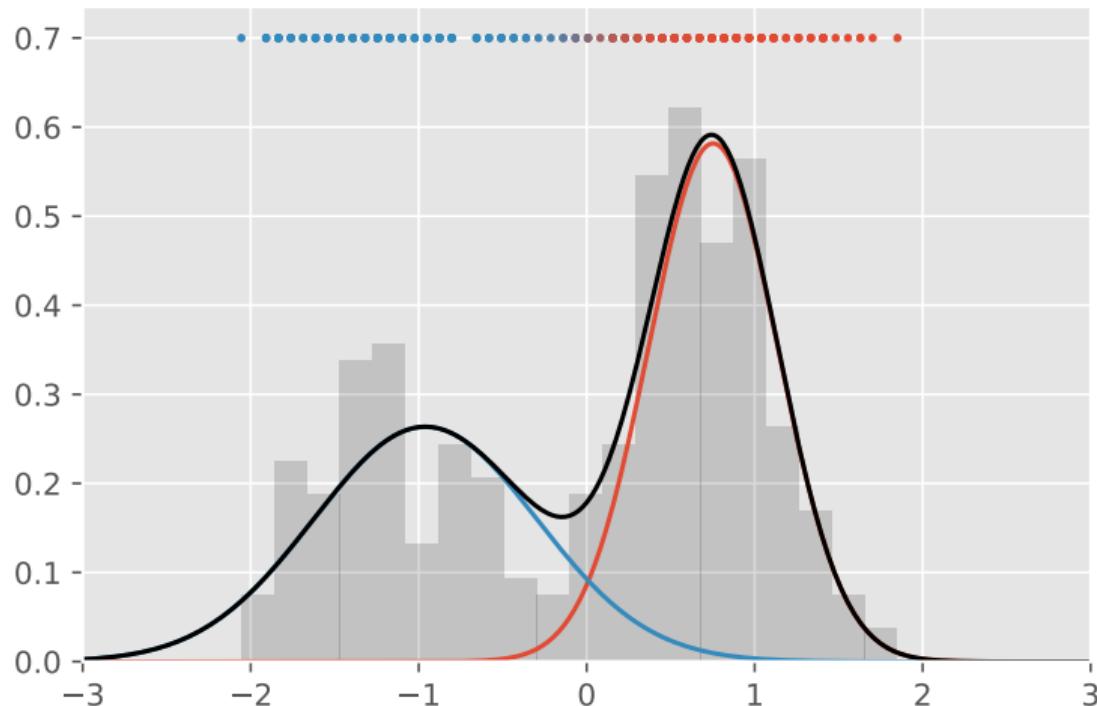
E-M Algorithm Demo

Iteration #12



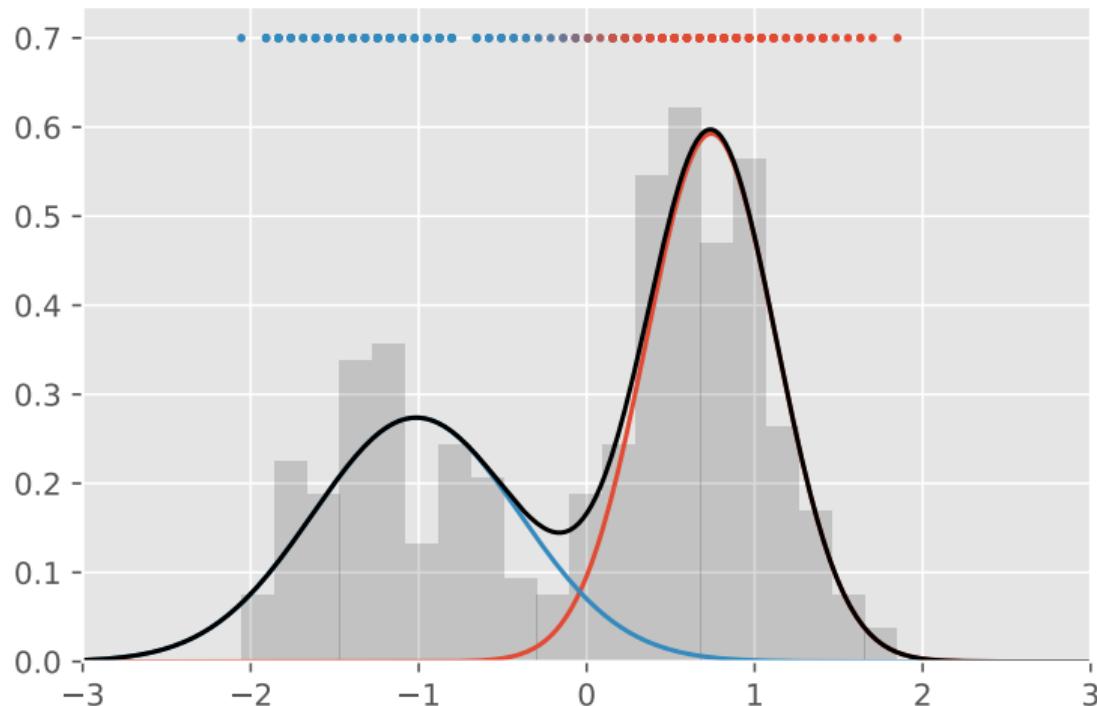
E-M Algorithm Demo

Iteration #13



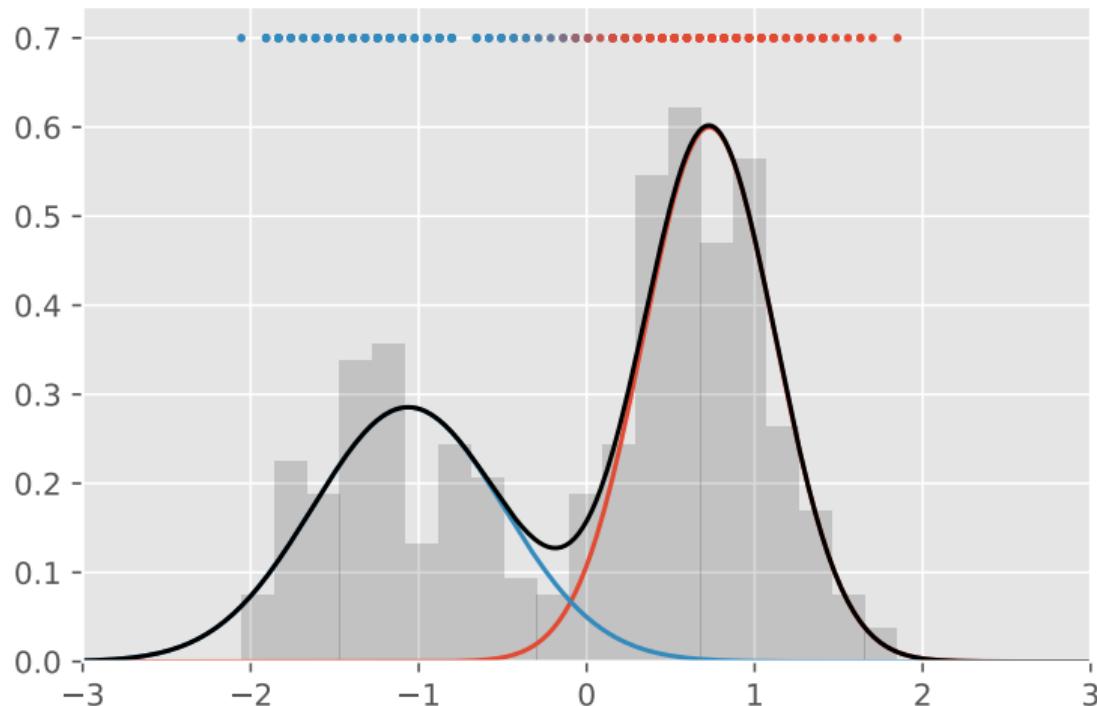
E-M Algorithm Demo

Iteration #14



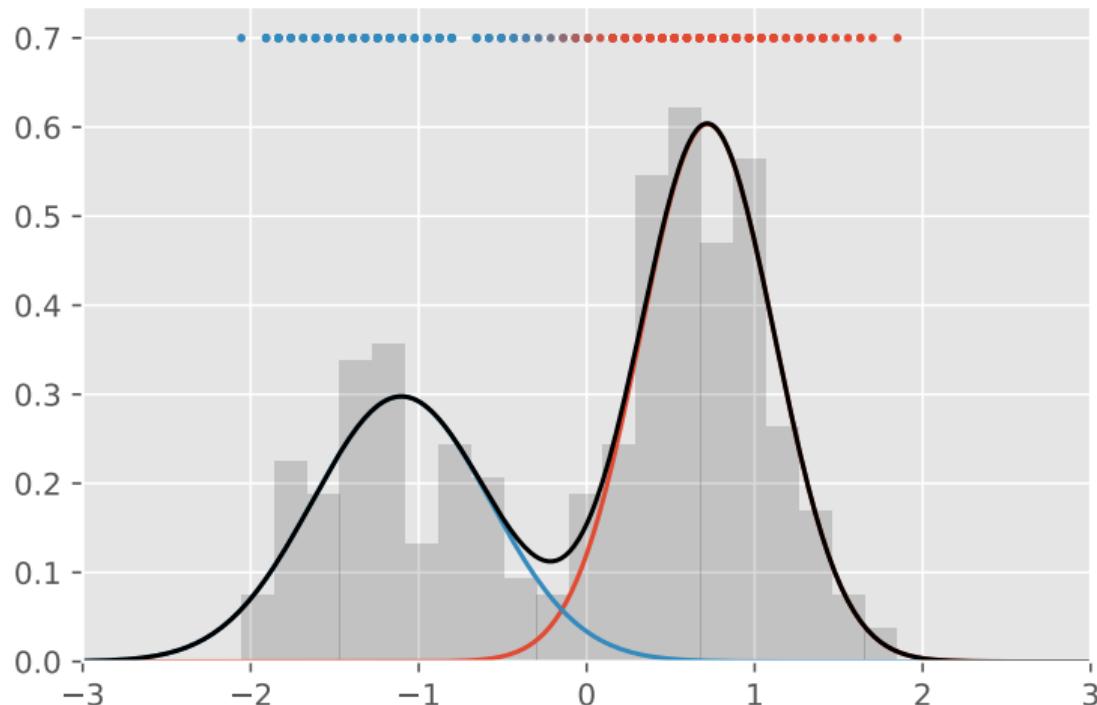
E-M Algorithm Demo

Iteration #15

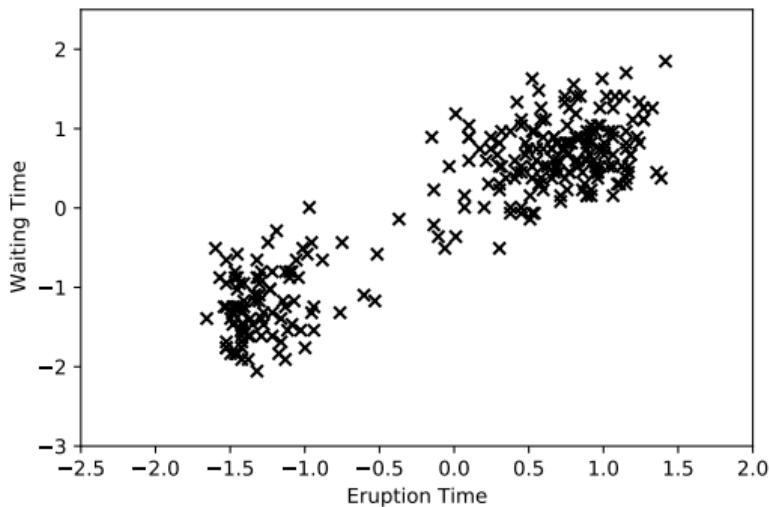


E-M Algorithm Demo

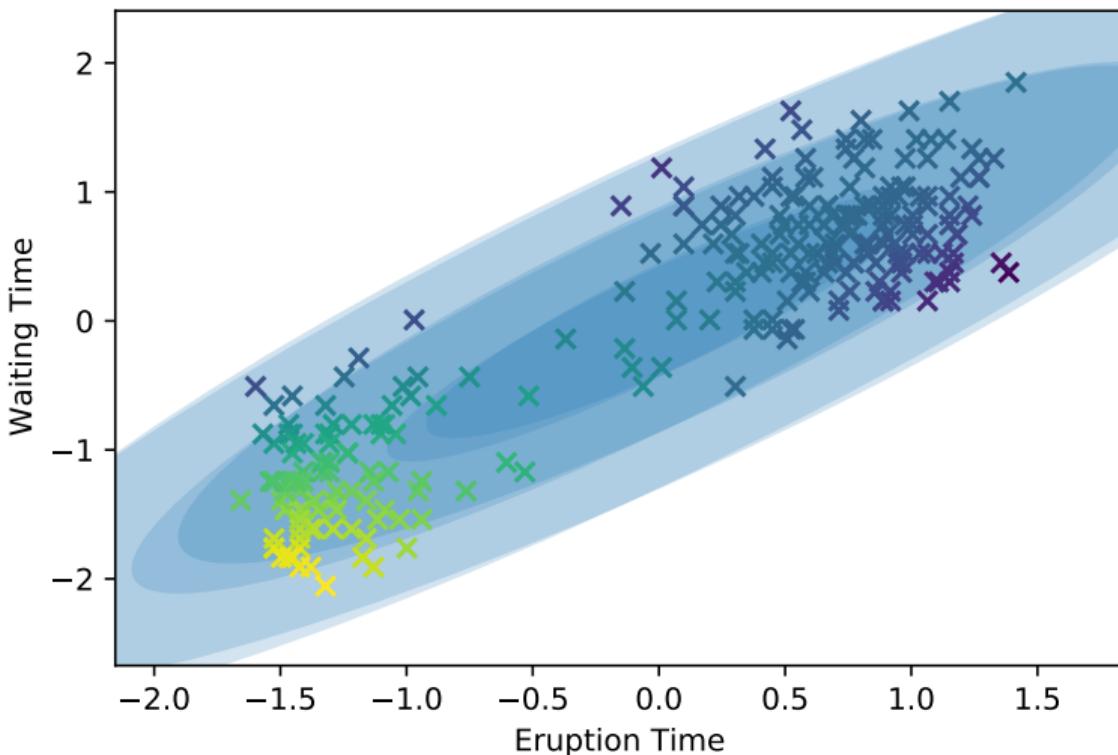
Iteration #16



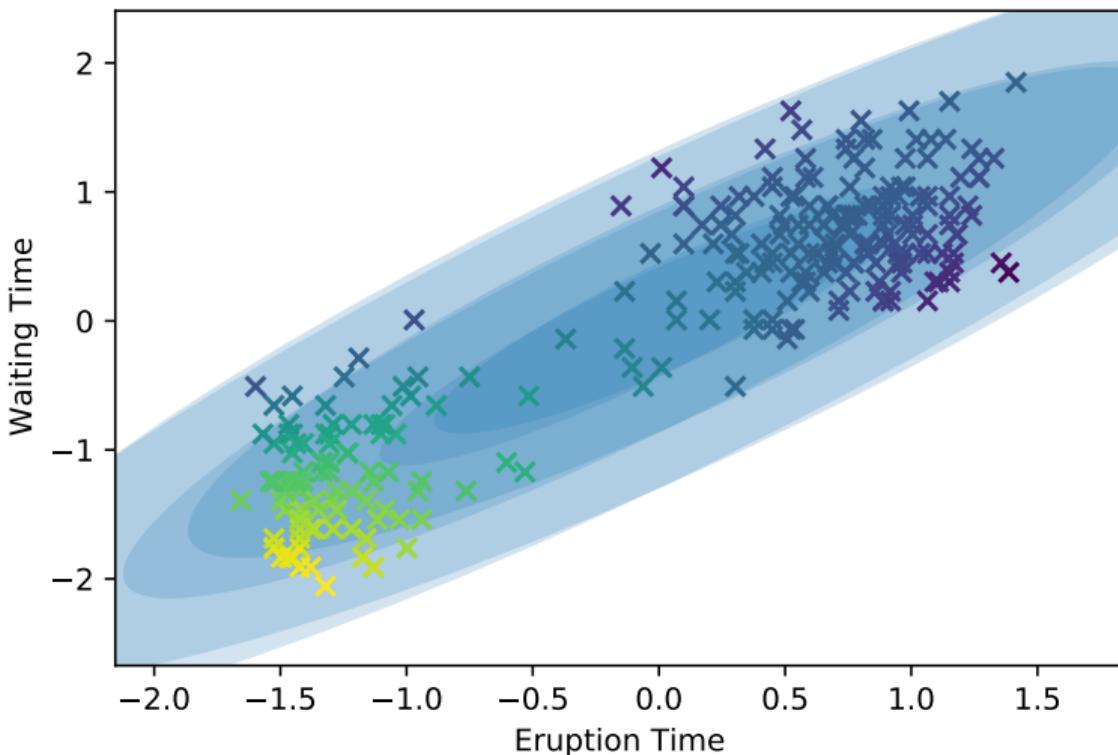
Geyser Eruptions



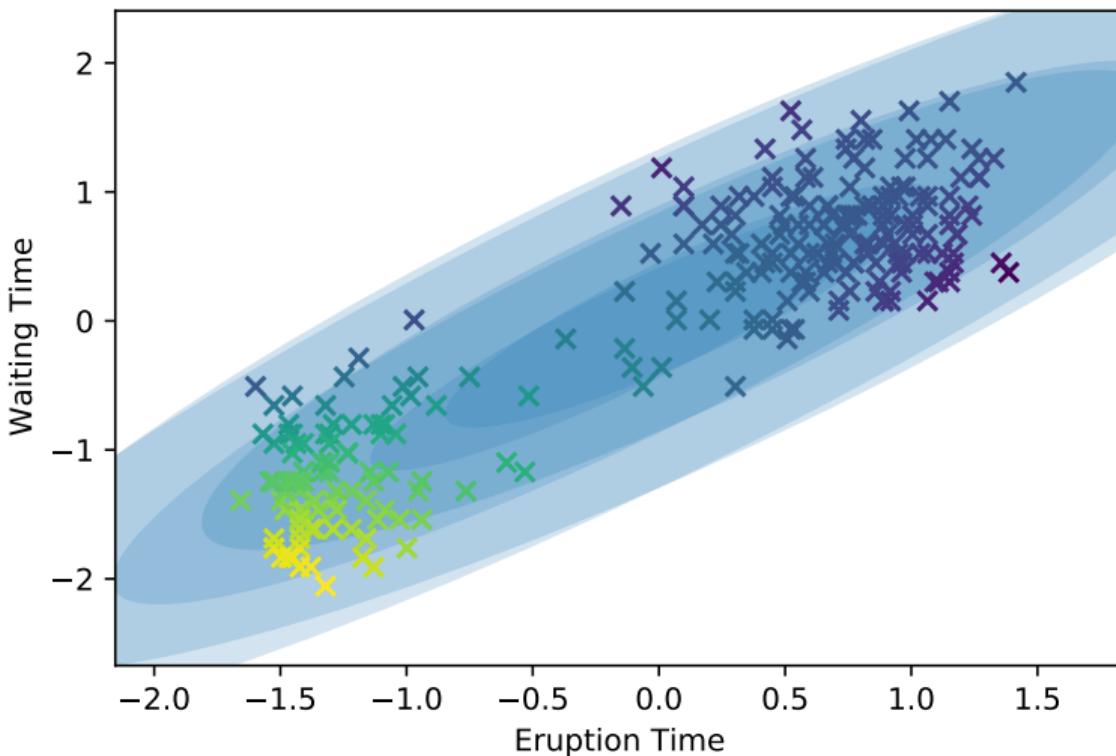
Iteration #1



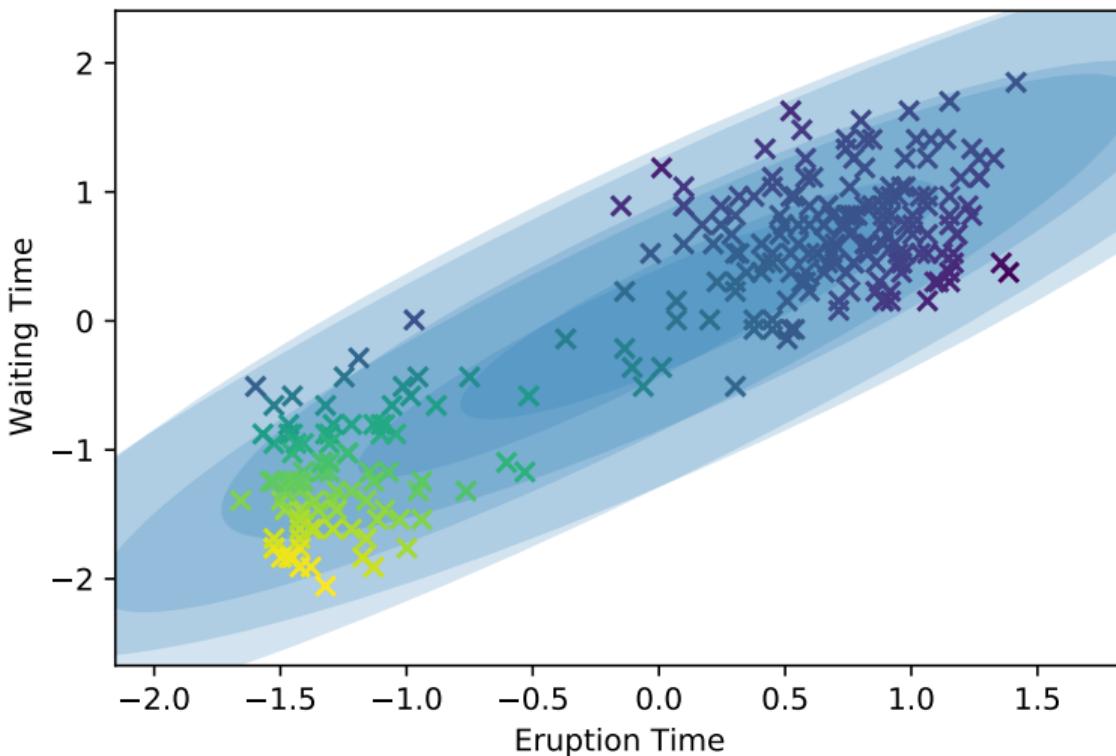
Iteration #2



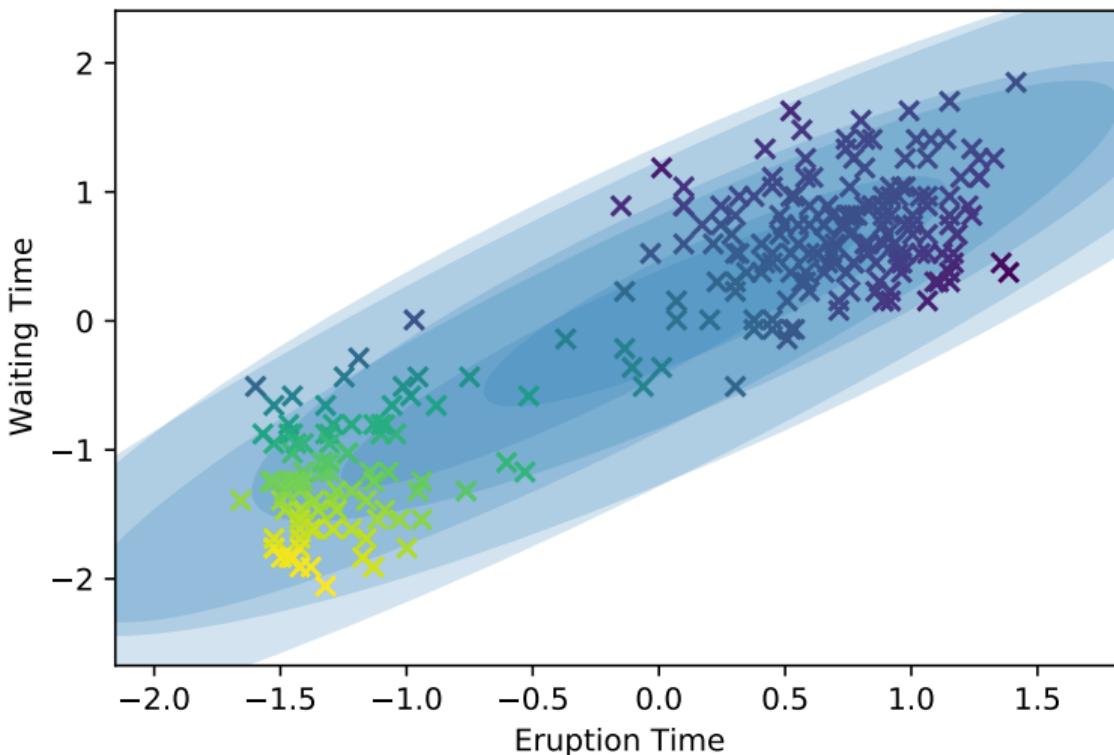
Iteration #3



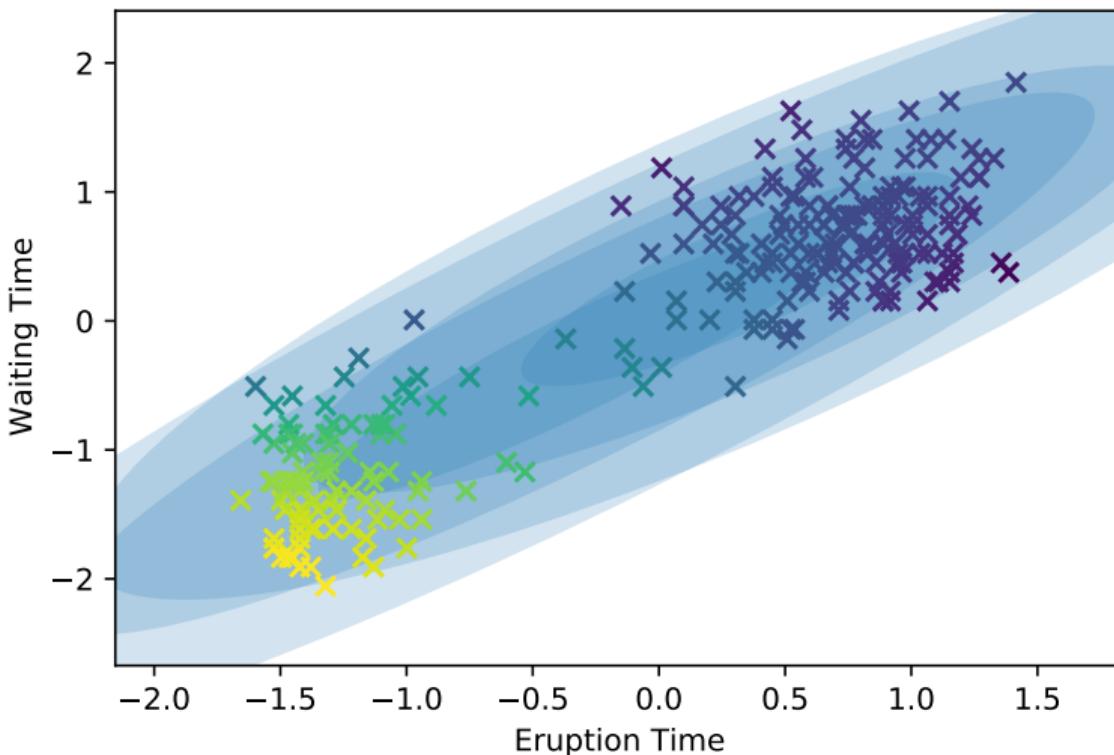
Iteration #4



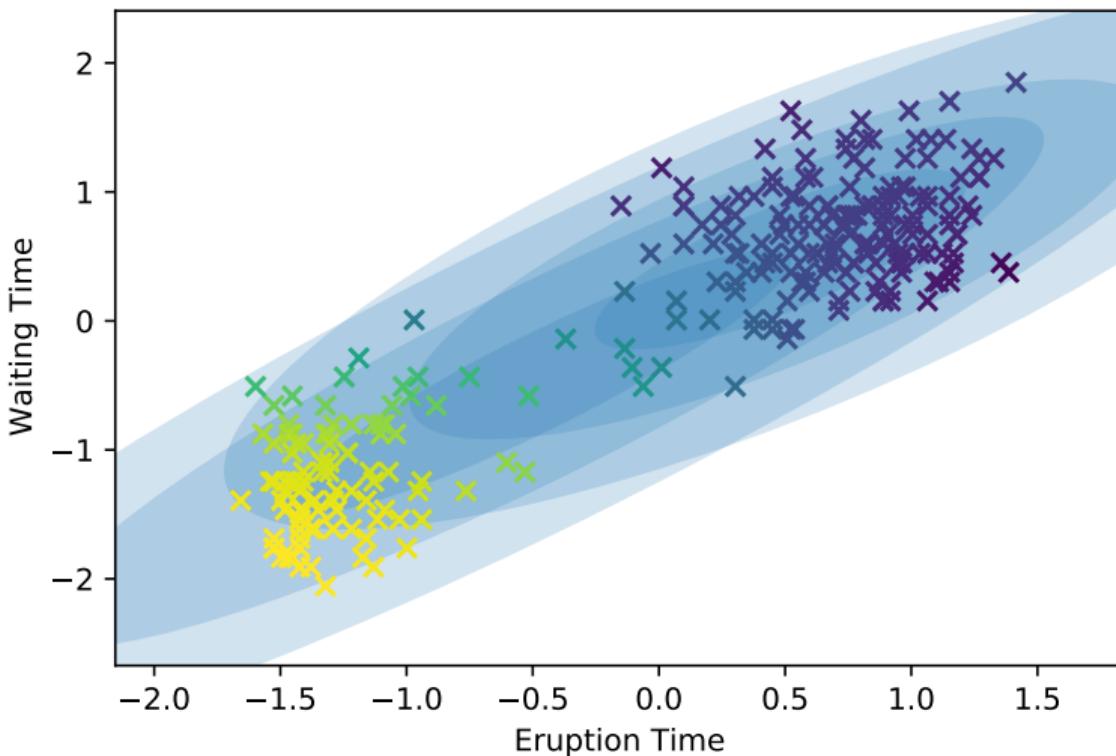
Iteration #5



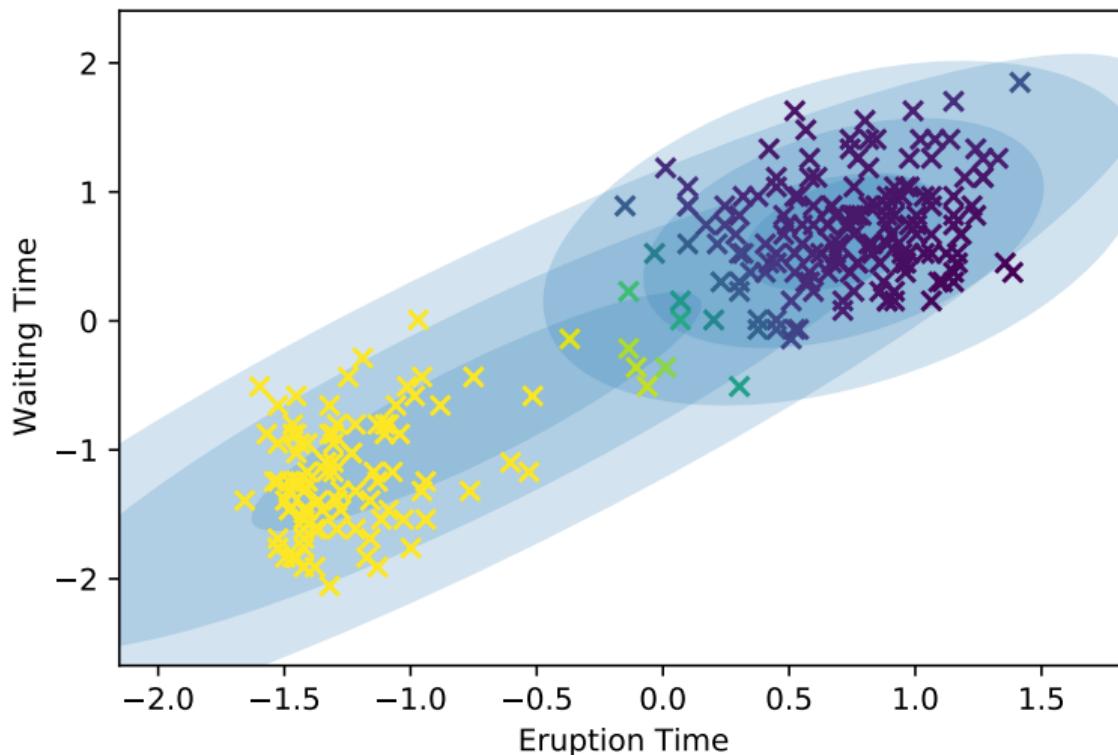
Iteration #6



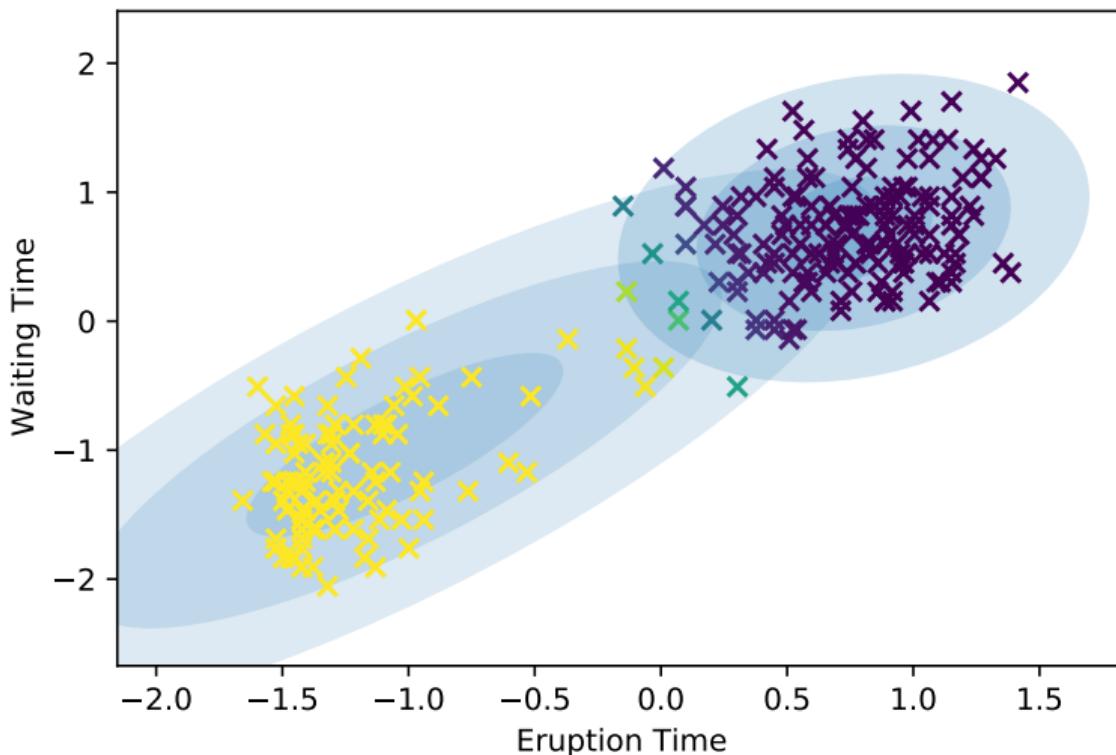
Iteration #7



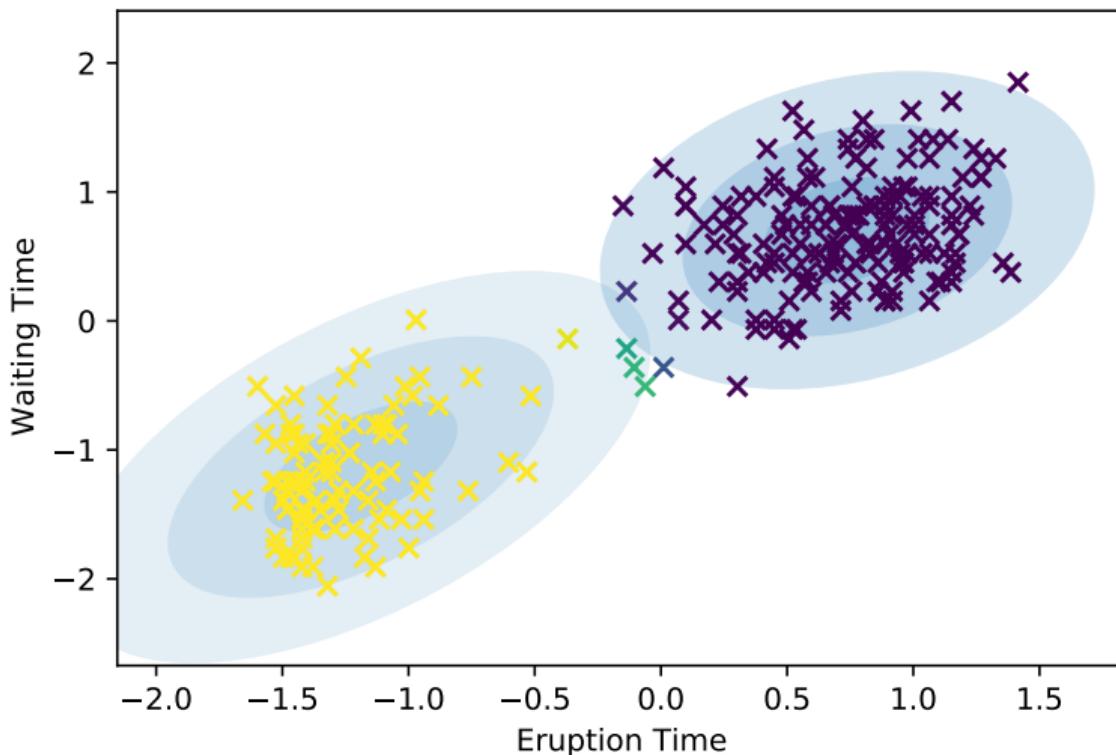
Iteration #8



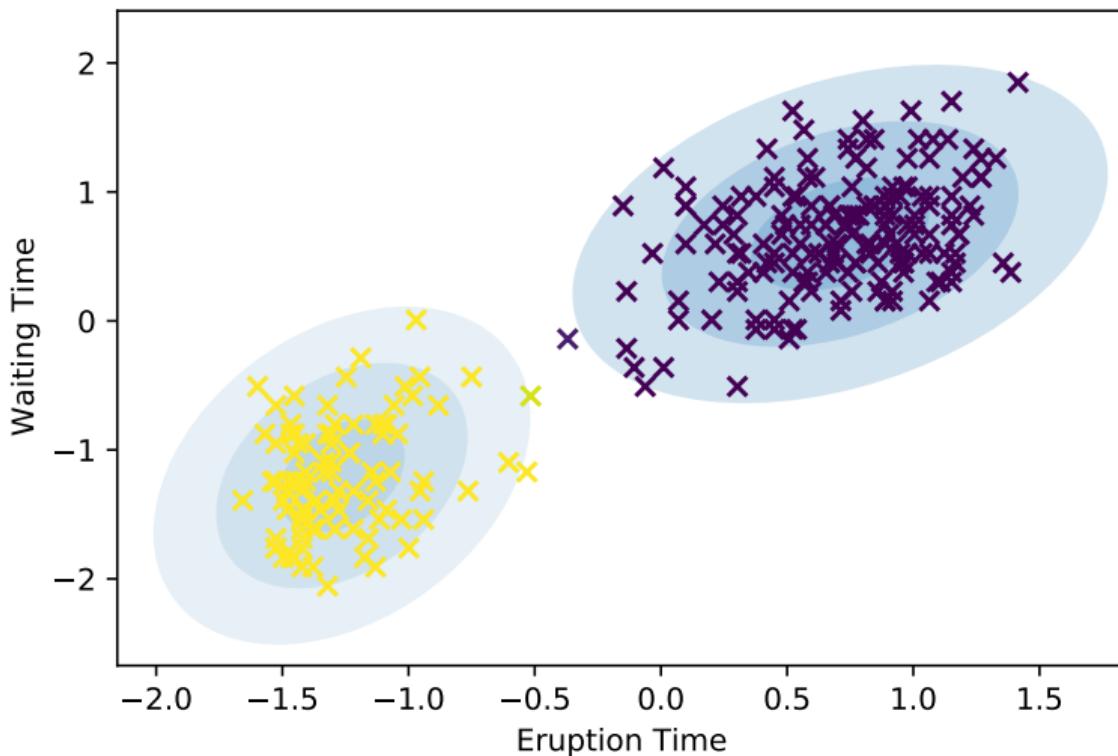
Iteration #9



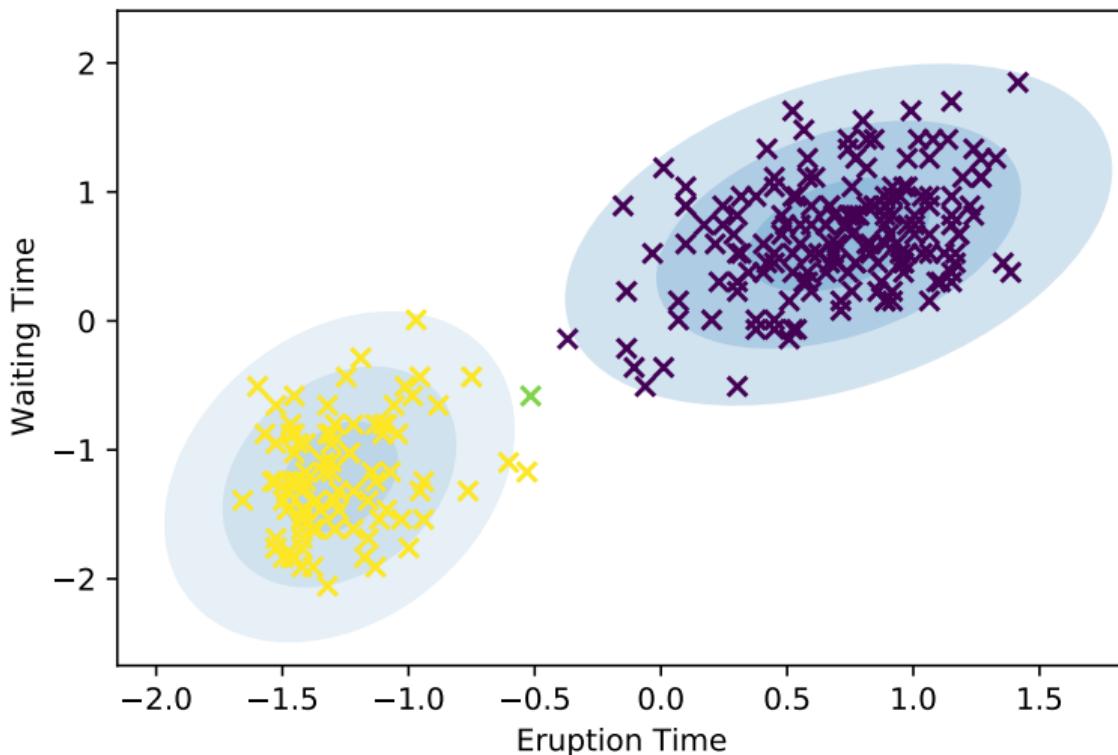
Iteration #10



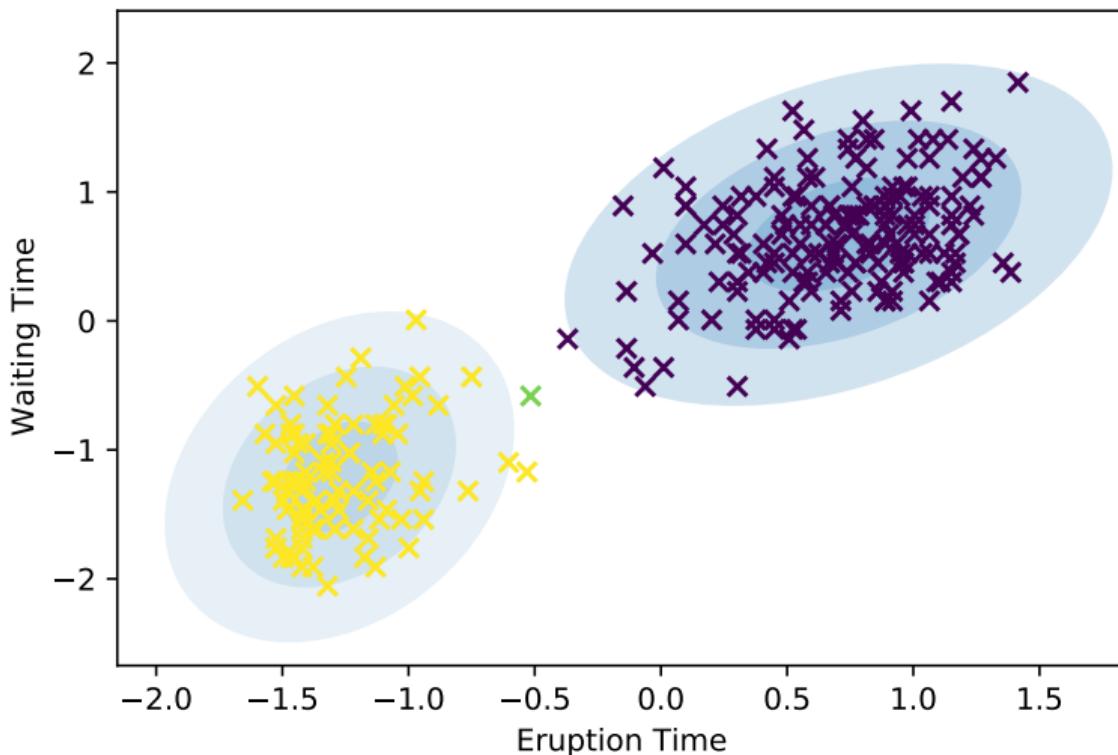
Iteration #11



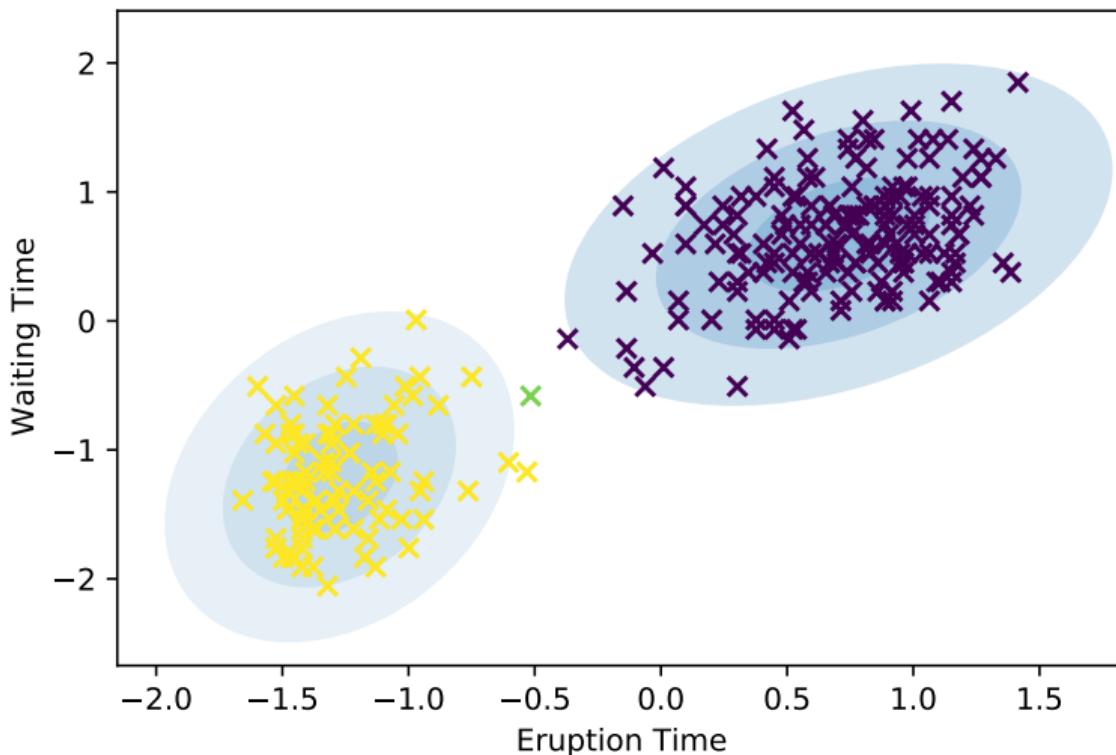
Iteration #12



Iteration #13



Iteration #14

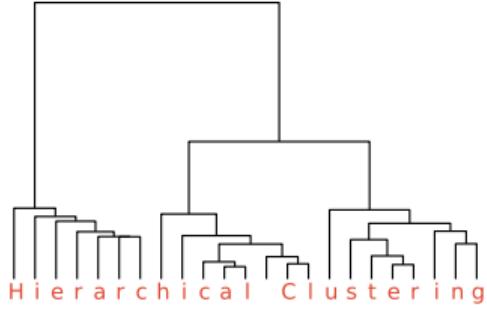


Clustering with EM

- ▶ Like with LDA/QDA, can assume spherical, diagonal, full covariance.
- ▶ May require many initializations.
- ▶ One way to initialize: k-means.

K-Means and EM

- ▶ K-Means is a limit case of EM!
- ▶ Spherical Gaussians, variance $\rightarrow 0$.

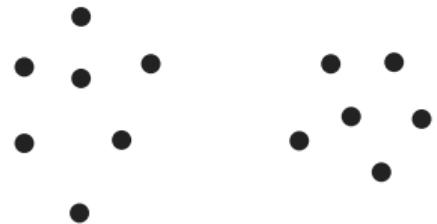


CSE 151A
Intro to Machine Learning

Lecture 16 – Part 02
Hierarchical Clustering

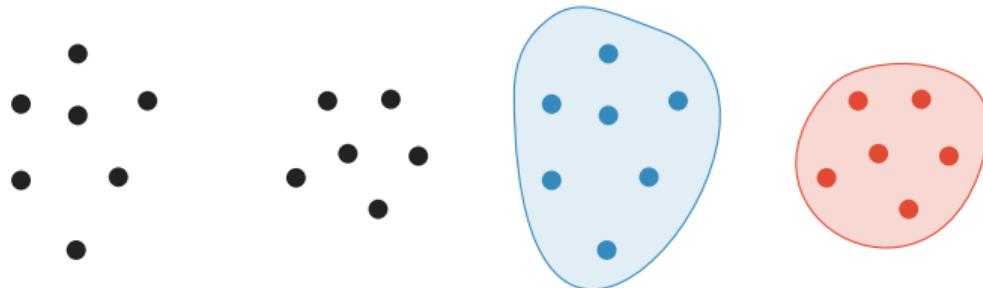
The goal of clustering:

Identify **structure** in data by grouping it into **clusters**.



Flat Clustering

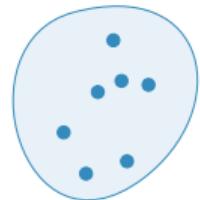
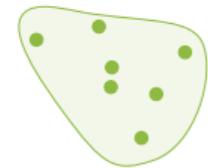
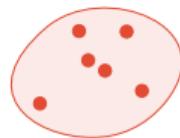
Partitioning of \mathcal{X} into **disjoint** sets called **clusters** s.t.
each point $x \in \mathcal{X}$ is in exactly one cluster.



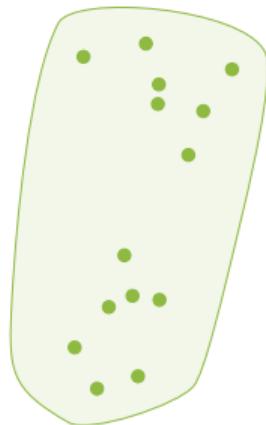
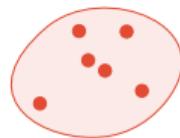
How many clusters are there?



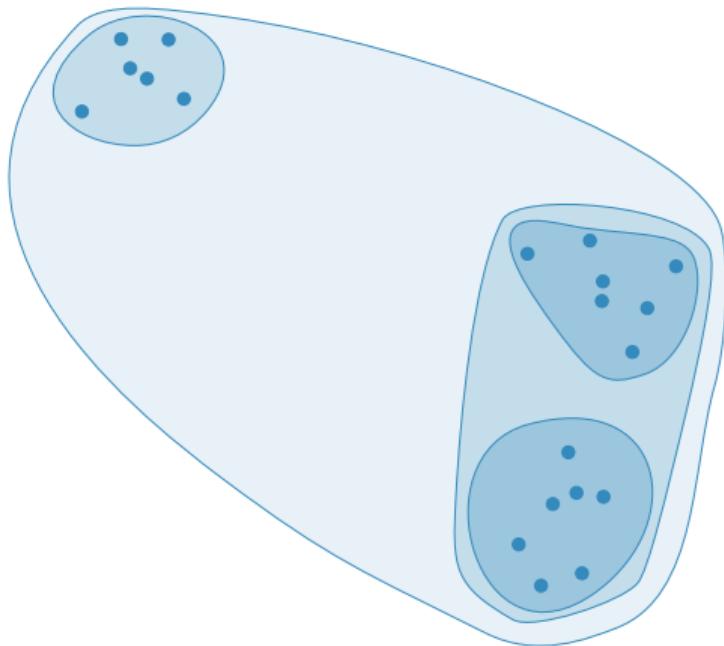
How many clusters are there?



How many clusters are there?

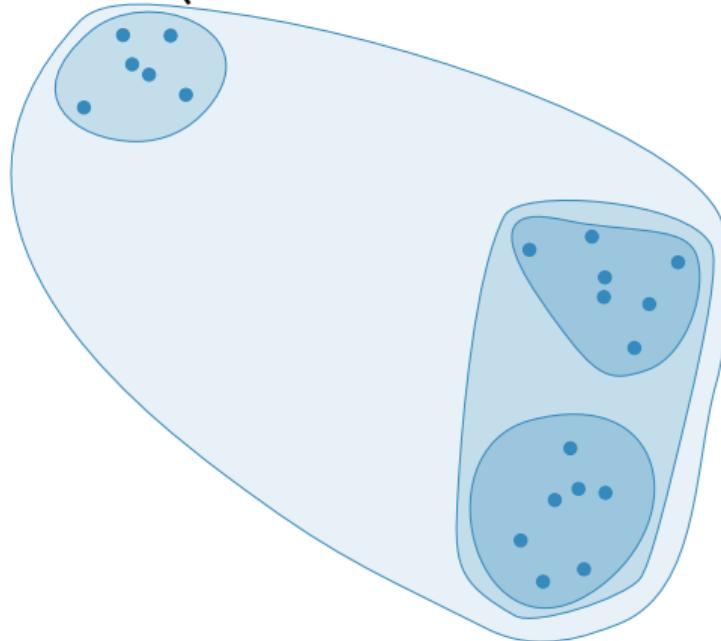


Allow clusters to nest...



A **hierarchical** clustering:

Collection \mathbb{C} of clusters s.t. any two are either **disjoint**, or **nested** (one is contained in the other).



How do we build a hierarchical clustering?

- ▶ There are two general approaches...
 - ▶ **Agglomerative (bottom-up):**
Start with each point in own cluster, iteratively **merge** them.
 - ▶ **Divisive (top-down):**
Start with all points in single cluster, recursively **divide** them.

Hierarchical Clustering

Input is a set of **objects** \mathcal{X} and a **dissimilarity** d :

$$d(x, x') \geq 0$$

non-negativity

$$d(x, x') = d(x', x)$$

symmetry

Linkage algorithms

- ▶ **Linkage algorithms** are a class of **agglomerative** approaches
- ▶ Idea:
 1. Start with each point in own cluster.
 2. Merge the two “**closest**” clusters.
 3. Repeat step 2 until we have a single cluster.

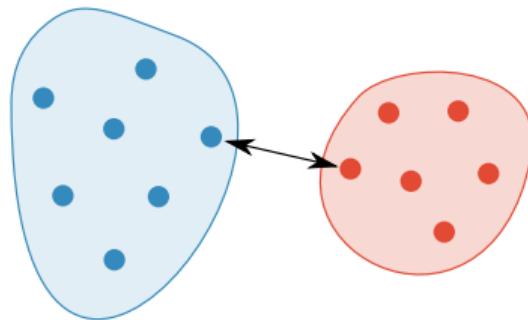
Linkage Algorithms

- ▶ How do we **measure** how **close** two clusters are?
- ▶ We use a **linkage function** \mathcal{L} taking pairs of clusters to \mathbb{R} .
- ▶ Single-linkage, complete-linkage,
average-linkage...

Single Linkage

The **smallest** distance between the clusters.

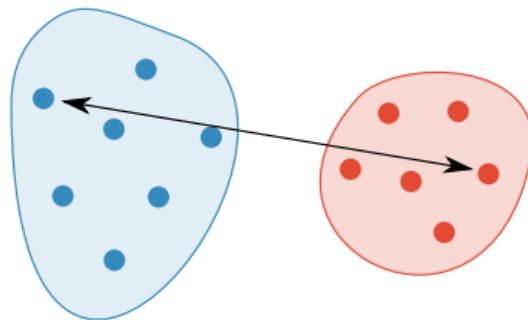
$$\mathcal{L}(C, C') = \min_{x, x' \in C \times C'} d(x, x')$$



Complete Linkage

The biggest distance between the clusters.

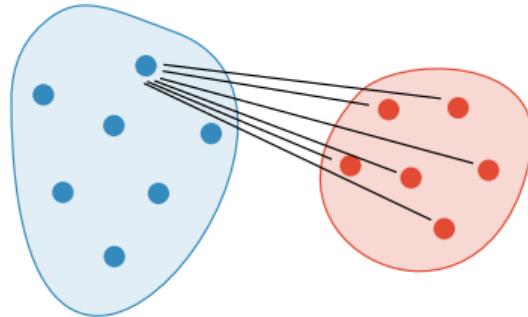
$$\mathcal{L}(C, C') = \max_{x, x' \in C \times C'} d(x, x')$$



Average-linkage (UPGMA)

The mean distance between the clusters.

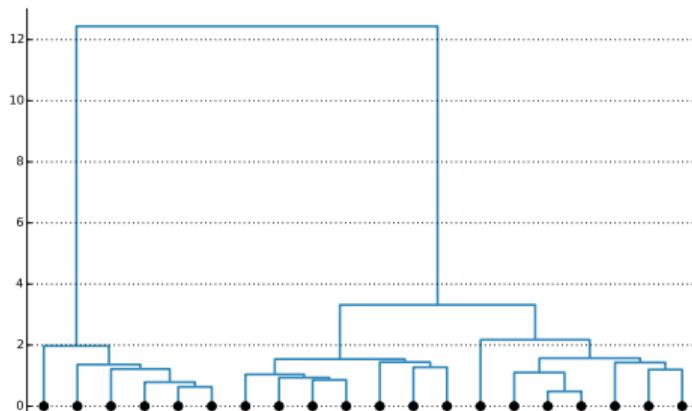
$$\mathcal{L}(C, C') = \frac{1}{|C \times C'|} \sum_{x, x' \in C \times C'} d(x, x')$$



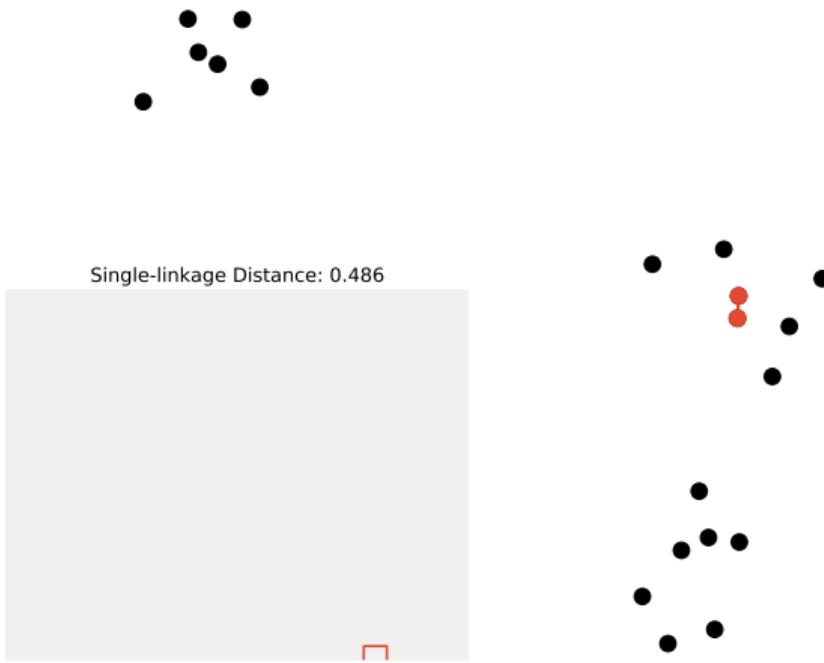
Dendrograms

Linkage clustering gives rise to a **dendrogram**.

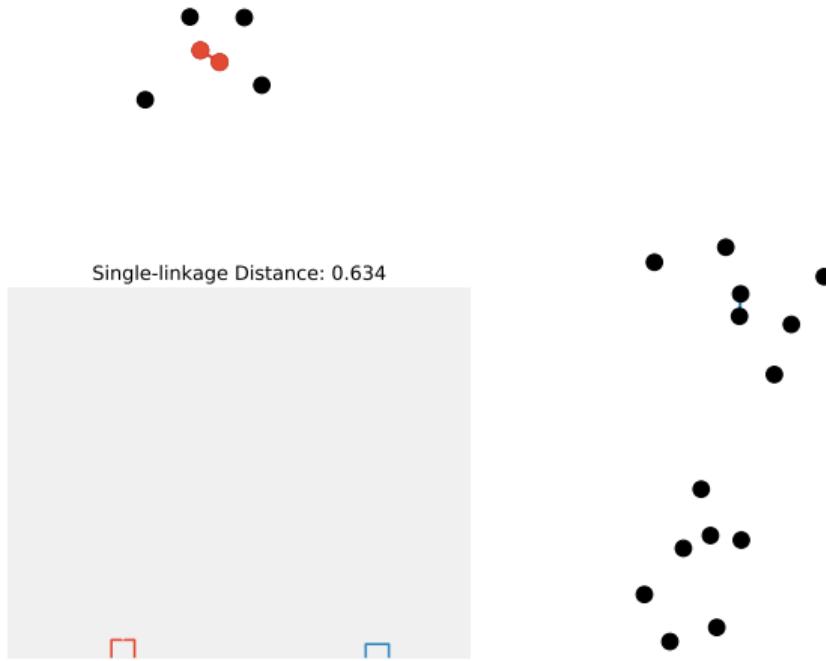
- ▶ Rooted tree whose leaves are points in \mathcal{X} .
- ▶ Can read off the linkage at which any pair of points merge.
- ▶ Cutting the dendrogram at any height produces **flat** clustering.



Example: Single-linkage clustering



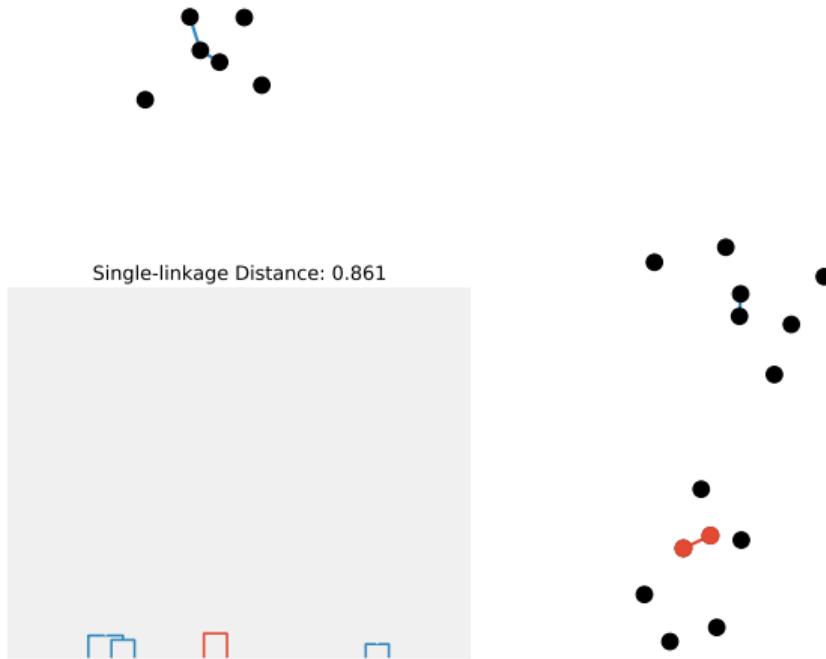
Example: Single-linkage clustering



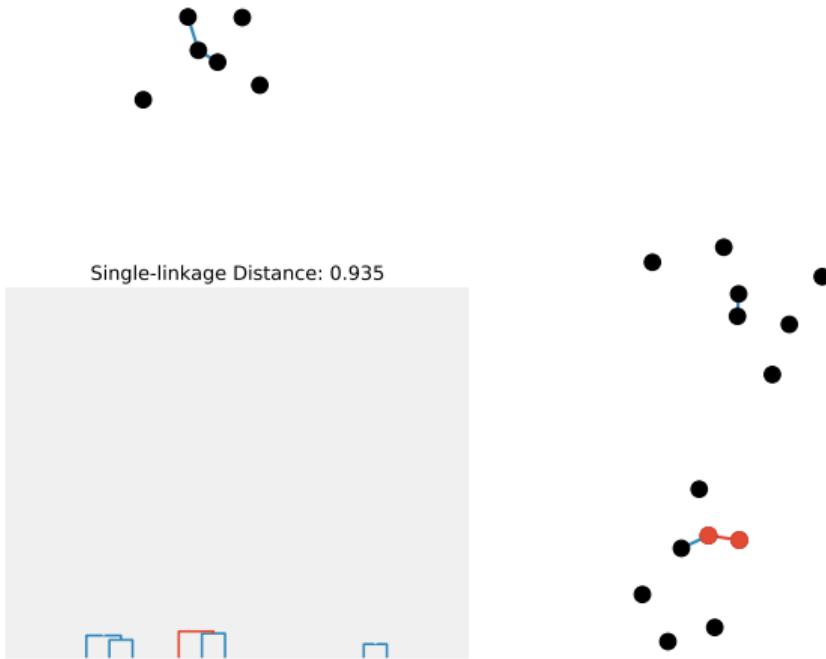
Example: Single-linkage clustering



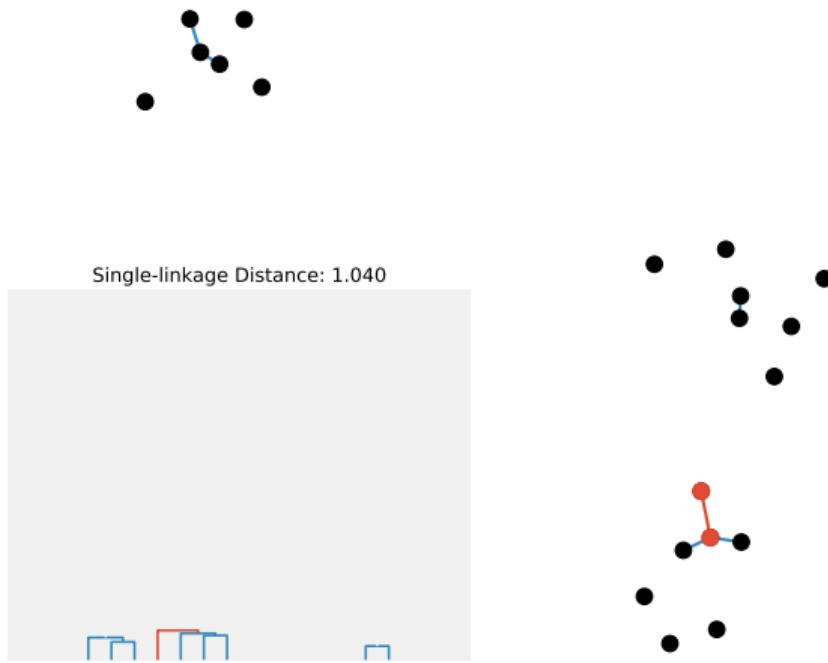
Example: Single-linkage clustering



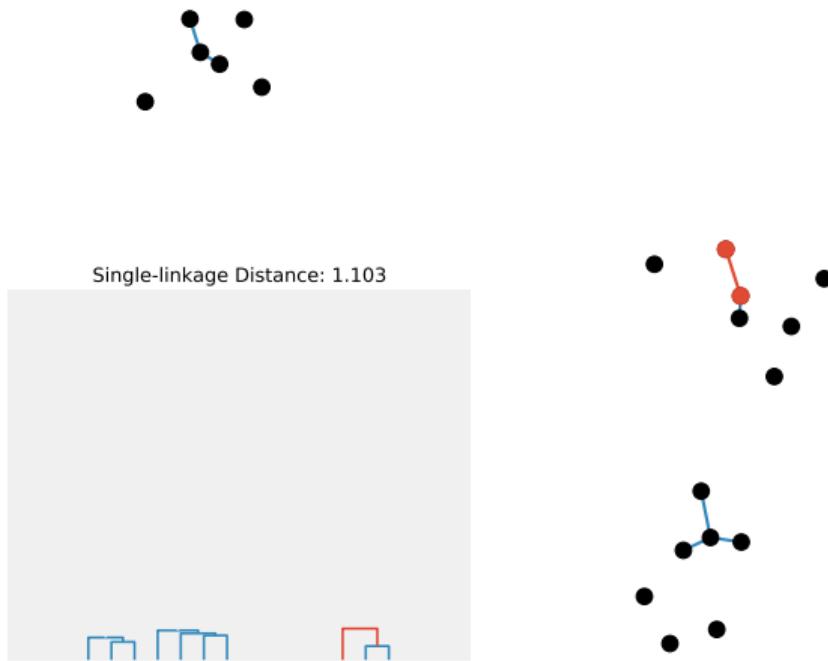
Example: Single-linkage clustering



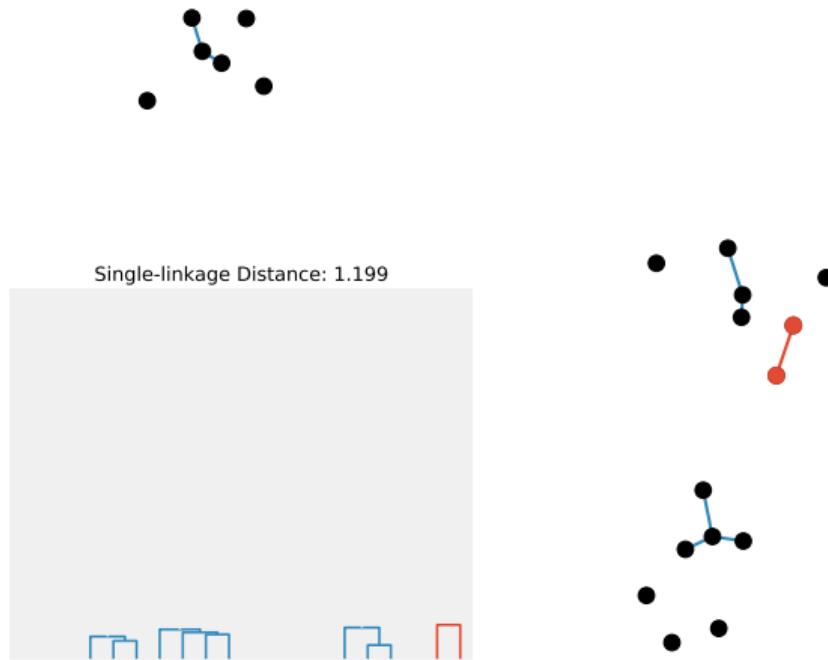
Example: Single-linkage clustering



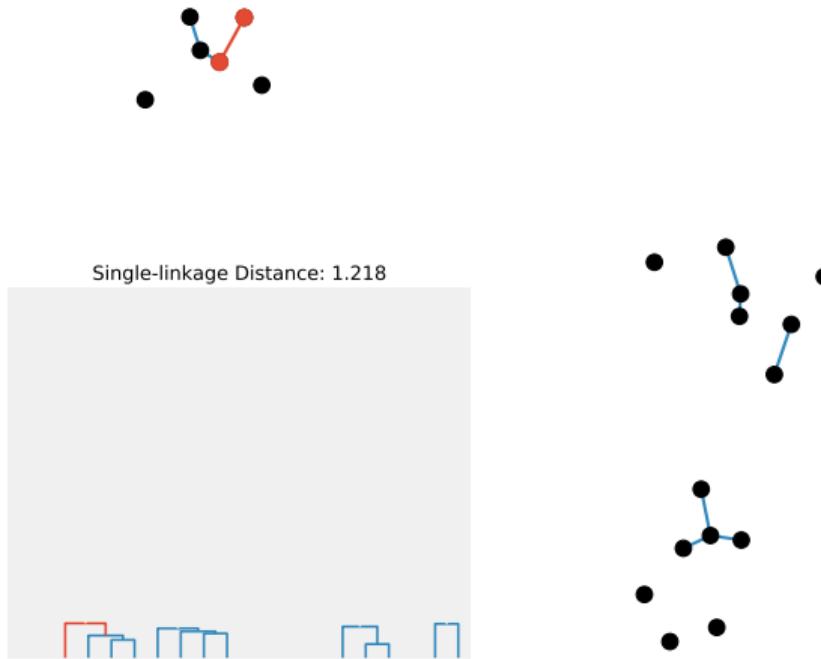
Example: Single-linkage clustering



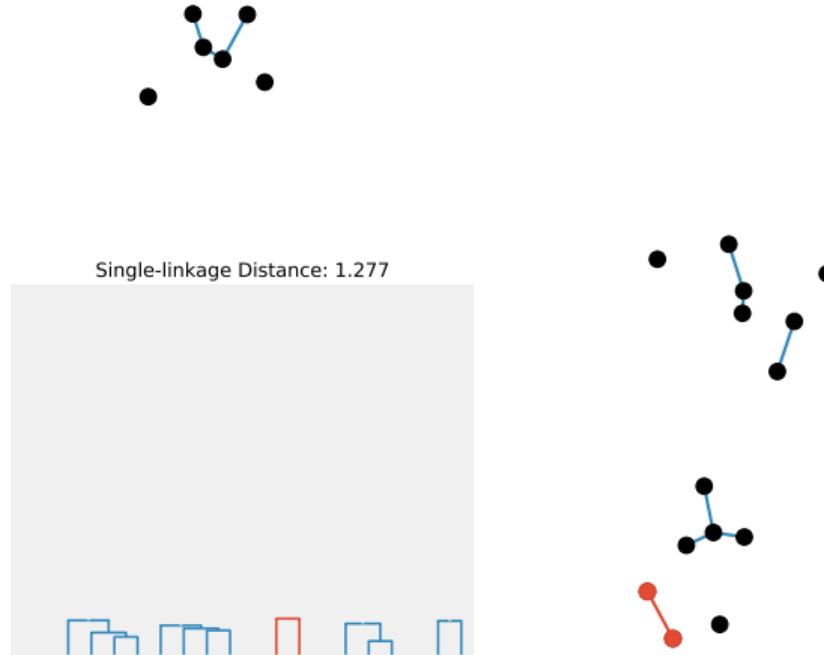
Example: Single-linkage clustering



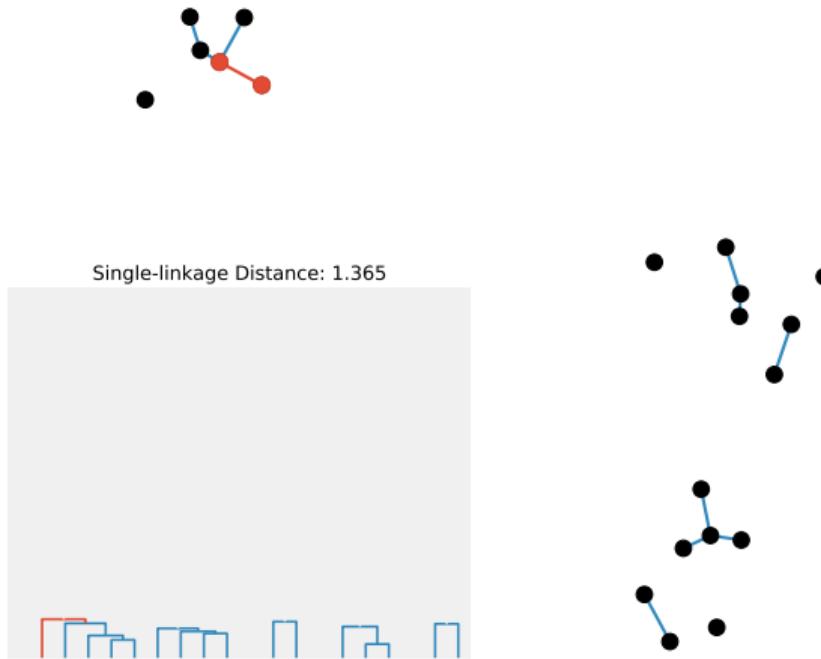
Example: Single-linkage clustering



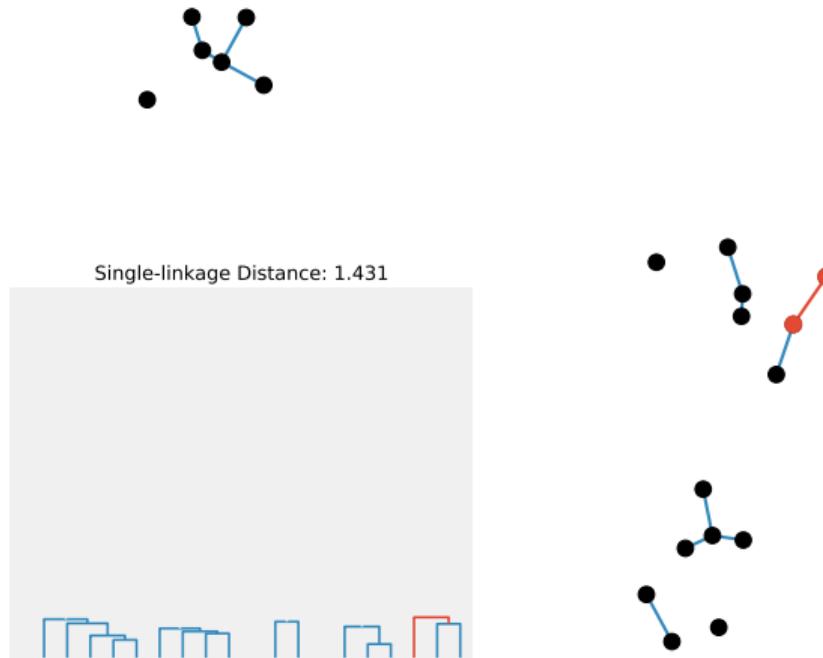
Example: Single-linkage clustering



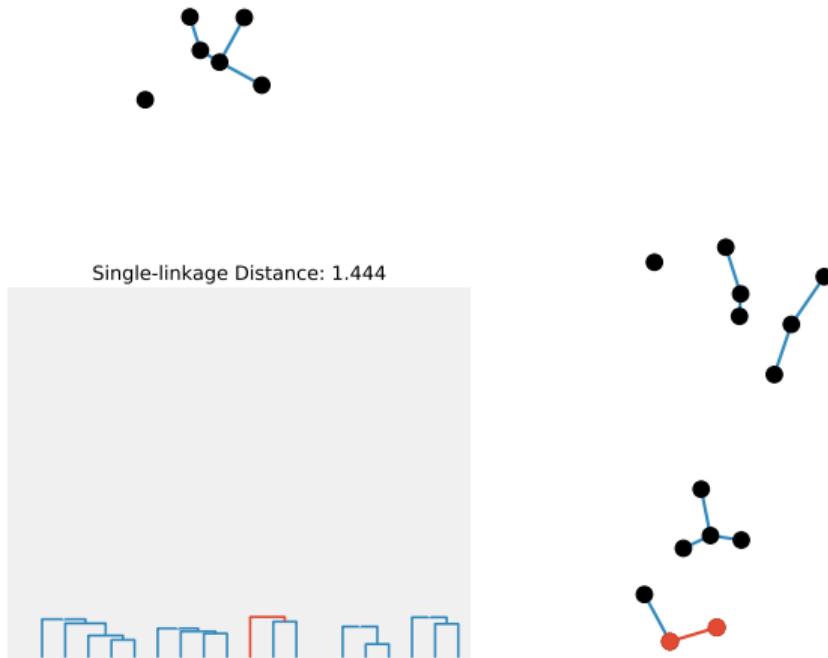
Example: Single-linkage clustering



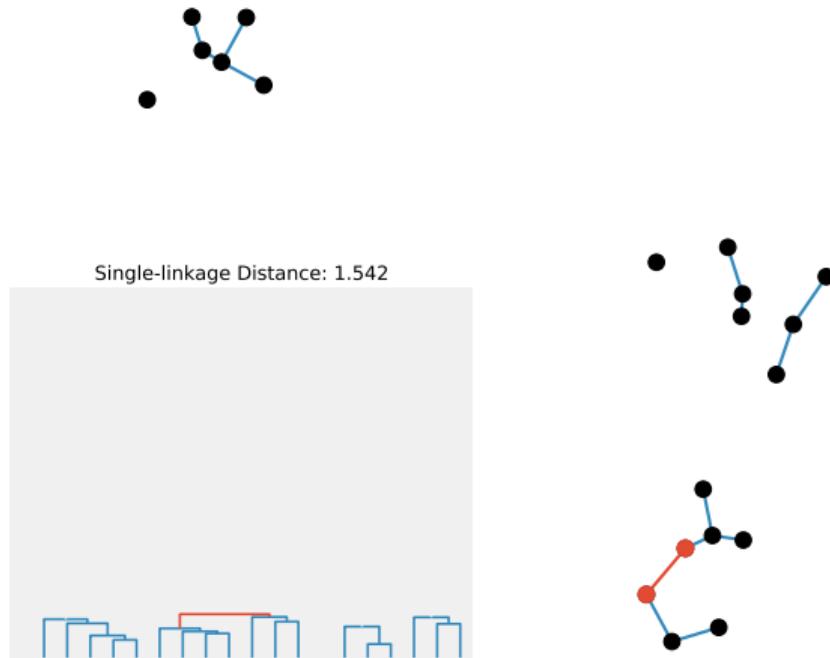
Example: Single-linkage clustering



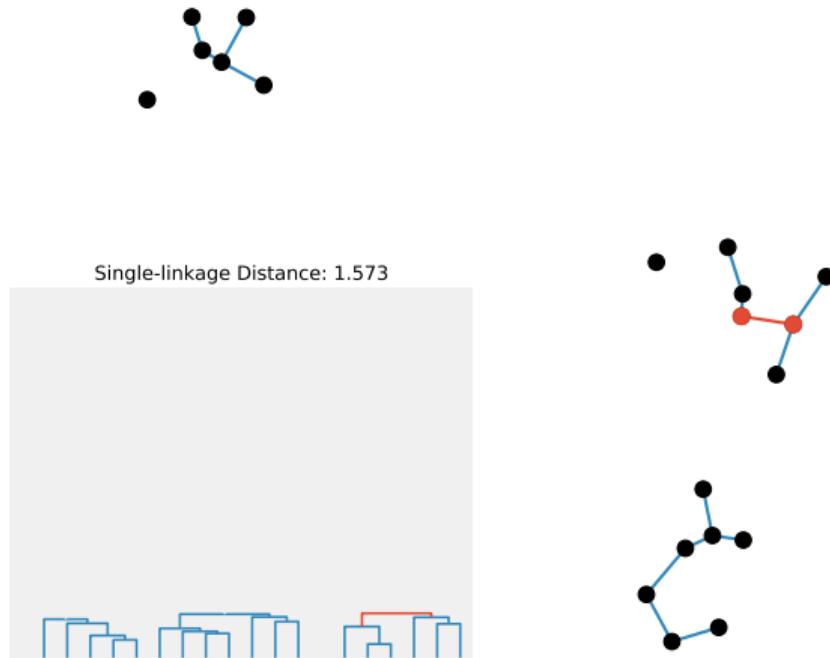
Example: Single-linkage clustering



Example: Single-linkage clustering



Example: Single-linkage clustering



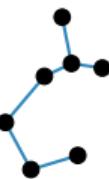
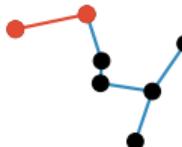
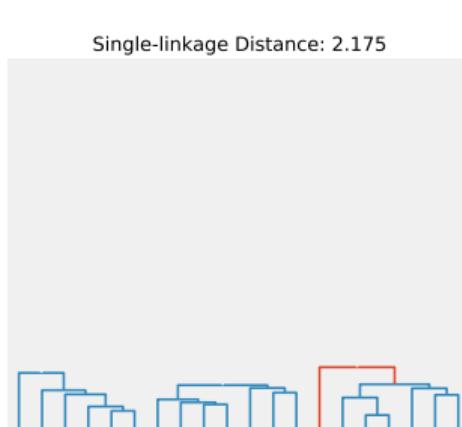
Example: Single-linkage clustering



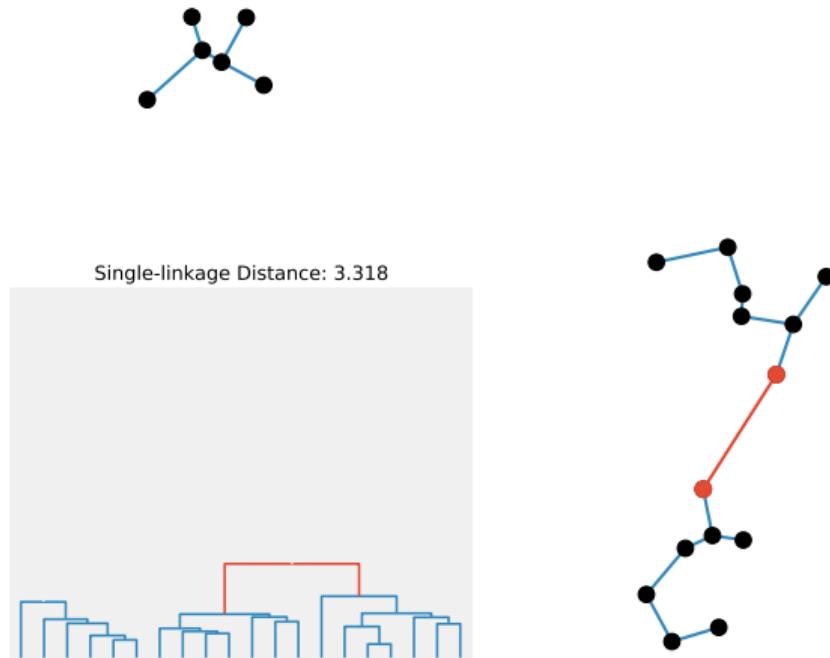
Single-linkage Distance: 1.975



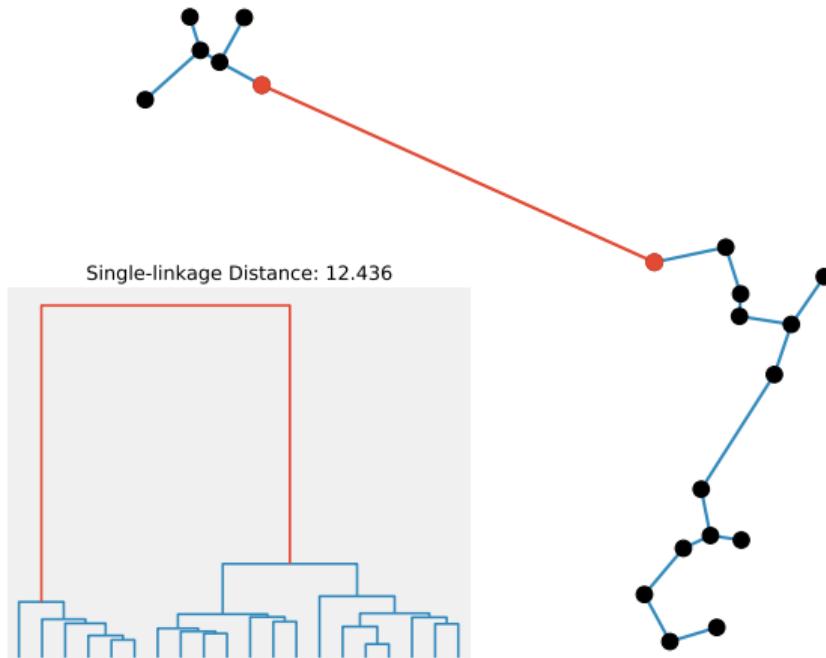
Example: Single-linkage clustering



Example: Single-linkage clustering



Example: Single-linkage clustering



Remember Kruskal's Algorithm?

- ▶ Build minimum spanning tree of weighted graph.
- ▶ Every step, add “lightest edge”.

Graph-Theoretic SLC

- ▶ Define complete weighted graph
 - ▶ Nodes are data points
 - ▶ Edge weights are distances
- ▶ For any number λ , delete all edges of weight $> \lambda$.
- ▶ Connected components of resulting graph are **single-linkage clusters** at level λ .

Practical considerations

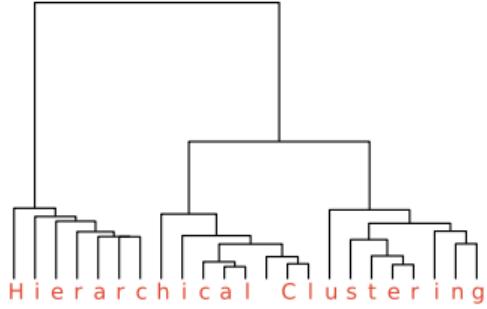
- ▶ Naïve implementations take $\Theta(n^3)$ time.

Practical considerations

- ▶ Naïve implementations take $\Theta(n^3)$ time.
- ▶ Some linkages have more efficient algorithms:
 - ▶ Single-linkage: $\Theta(n^2)$, since Prim's algorithm is $\Theta(n^2)$ on a complete graph.
 - ▶ Complete-, Average-linkage: $O(n^2 \log n)$.

Practical considerations

- ▶ Naïve implementations take $\Theta(n^3)$ time.
- ▶ Some linkages have more efficient algorithms:
 - ▶ Single-linkage: $\Theta(n^2)$, since Prim's algorithm is $\Theta(n^2)$ on a complete graph.
 - ▶ Complete-, Average-linkage: $O(n^2 \log n)$.
- ▶ Single-linkage is insensitive to density, exhibits chaining.



CSE 151A
Intro to Machine Learning

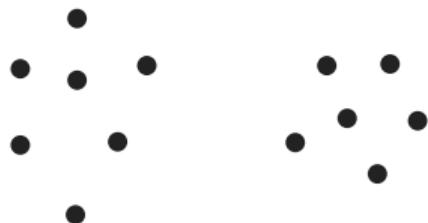
Lecture 16 – Part 03
Density Cluster Trees

The goal of clustering:

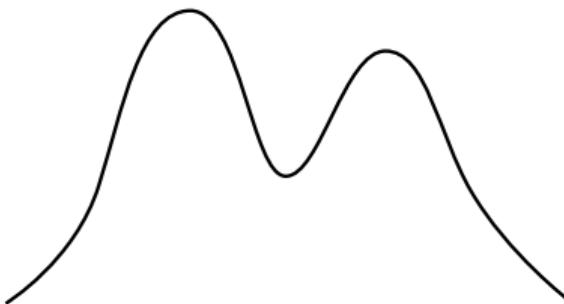
Identify structure in data by grouping it into **clusters**



The goal of clustering:
Identify structure in data by grouping it into **clusters**

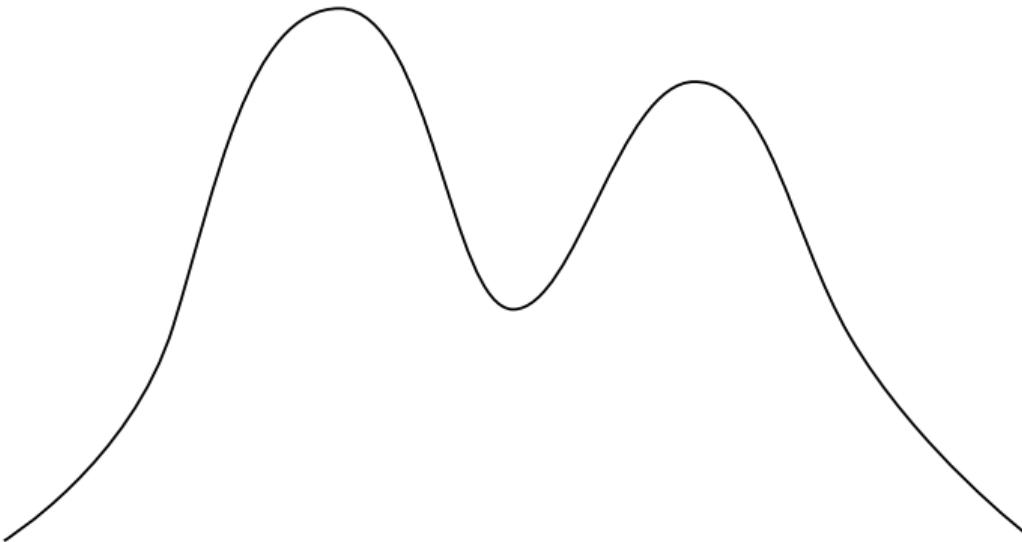


Assumption: data is drawn from some **density**.



What **structure** do we wish to recover?

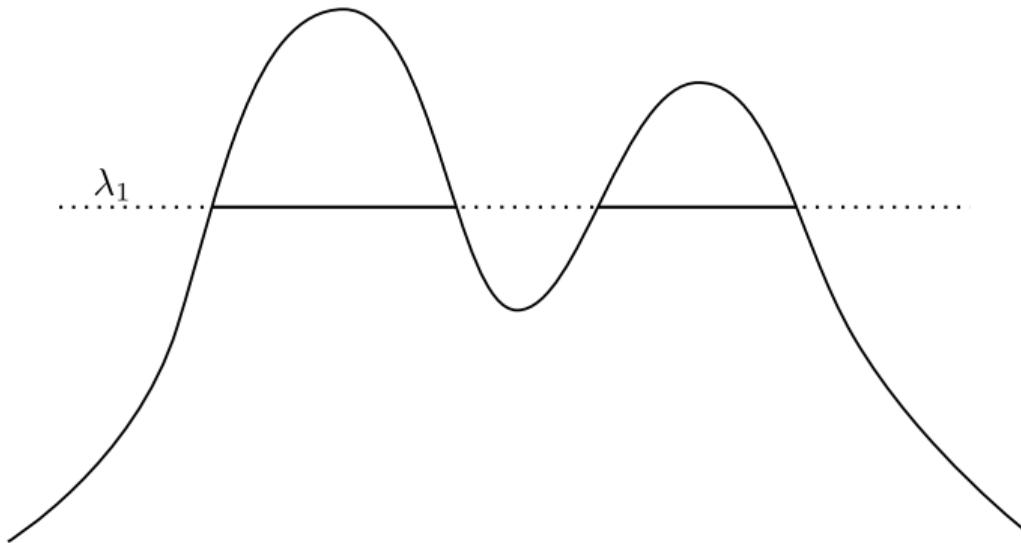
A **cluster** of a density is a *region of high probability*.¹



¹Hartigan (1981), Wishart (1969)...

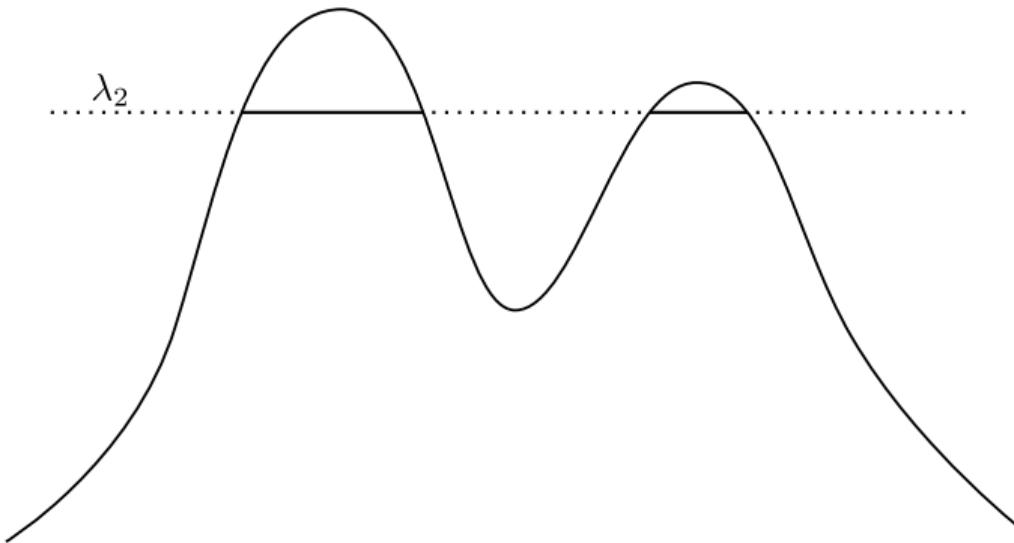
High-density clusters

Connected components of $\{f \geq \lambda_1\}$?



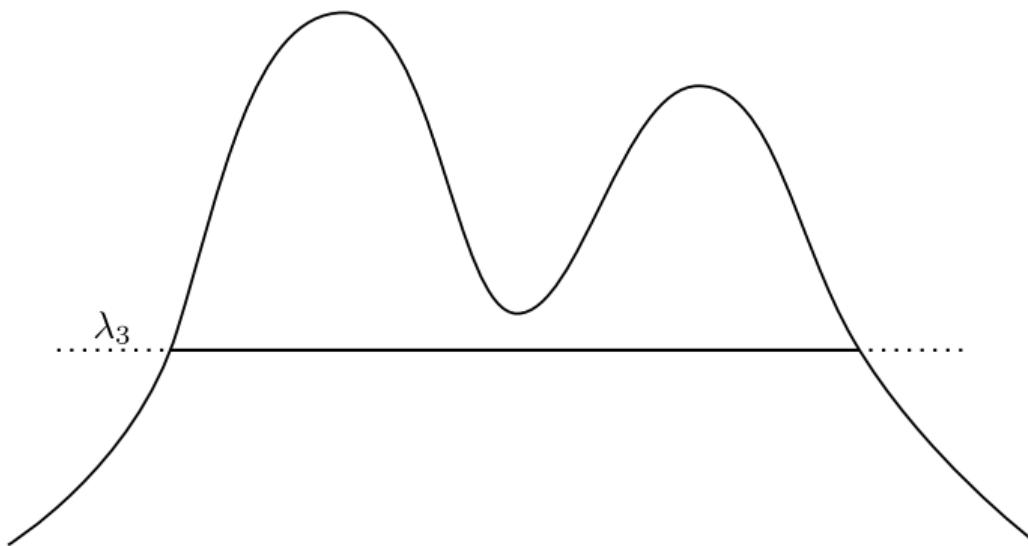
High-density clusters

Connected components of $\{f \geq \lambda_2\}$?



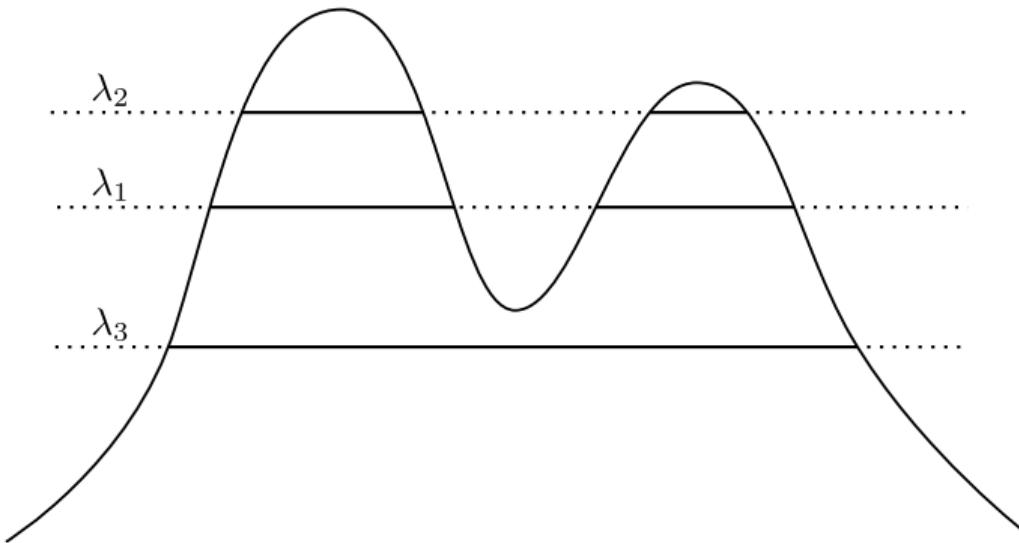
High-density clusters

Connected components of $\{f \geq \lambda_3\}$?



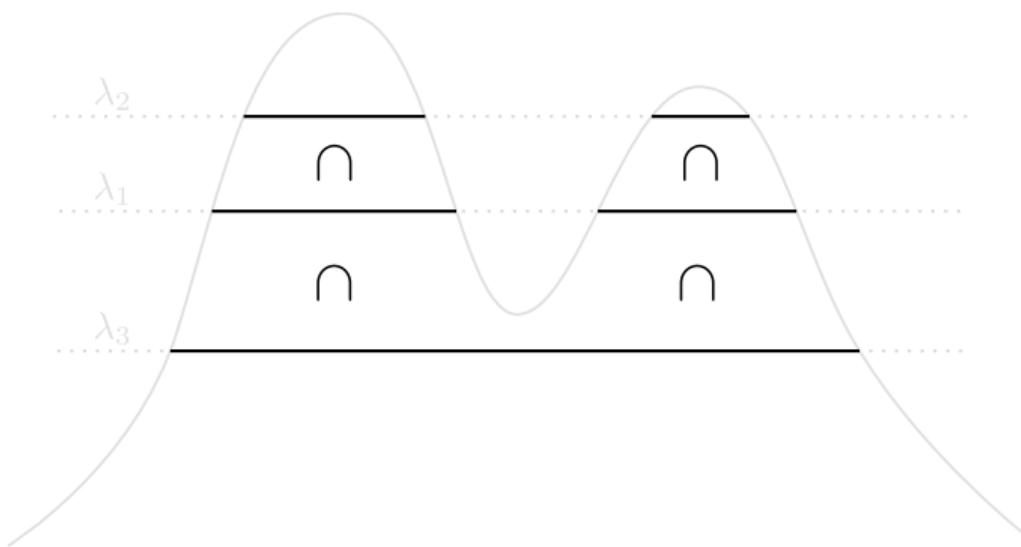
High-density clusters

A **cluster** is a connected component of $\{f \geq \lambda\}$ for any $\lambda > 0$.



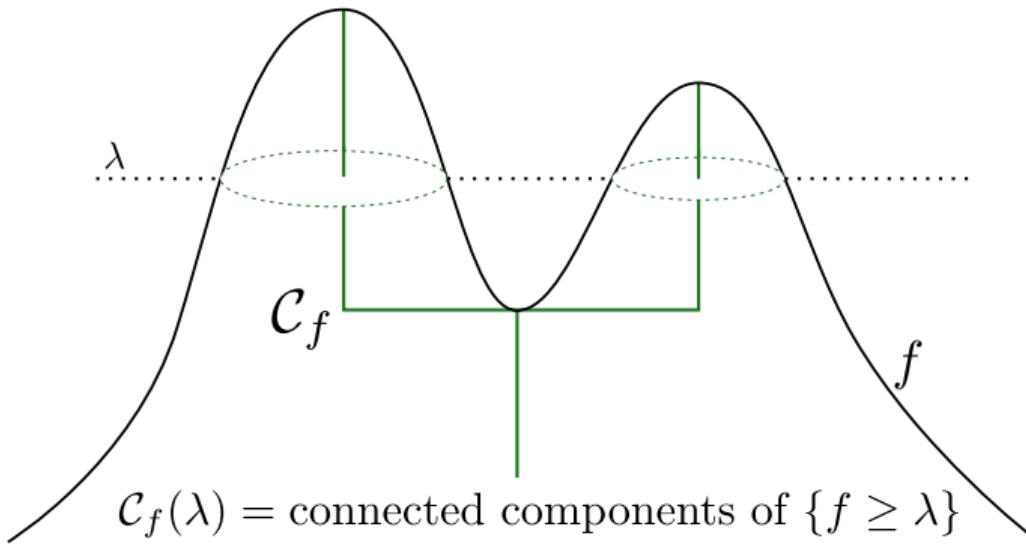
A hierarchy of clusters

Clusters from higher levels nest within clusters from lower levels.



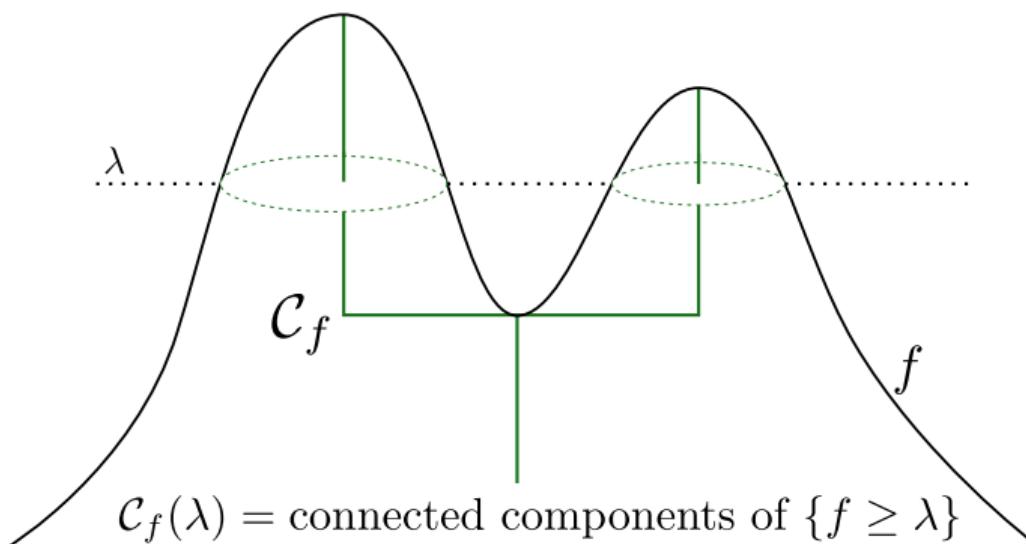
The density cluster tree

This gives rise to a tree structure called the **density cluster tree**.



What **structure** do we wish to recover?

This **density cluster tree** is what we hope to recover from data.



Robust single-linkage

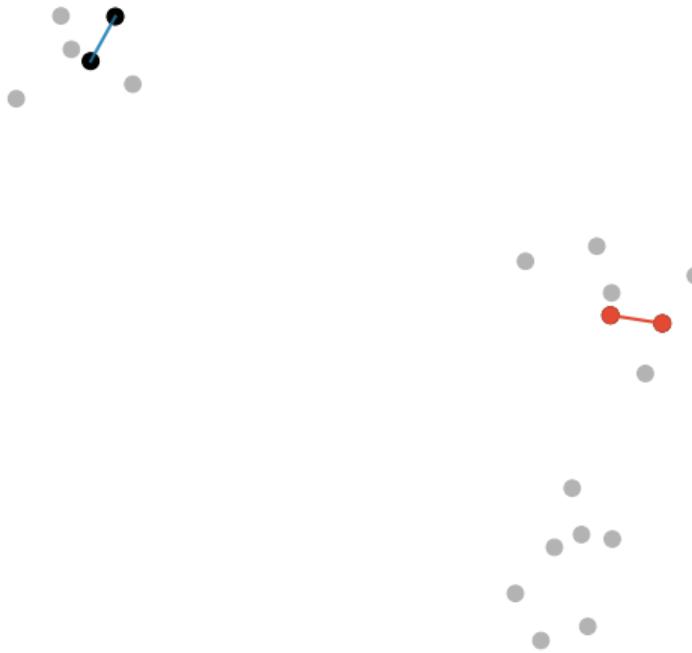
Intuition: At first, only admit **high-density** points into graph.

- ▶ Choose parameters α and k
- ▶ For each $x \in \mathcal{X}$, let $r_k(x)$ be the distance to x 's k -th nearest neighbor.
- ▶ As r grows from 0 to ∞ :
 - ▶ Let $V = \{x : r_k(x) \leq r\}$.
 - ▶ Let $E = \{(x, x') : d(x, x') \leq \alpha r\}$.
 - ▶ Build the graph $G_r = (V, E)$.
- ▶ The **clusters** at time r are the **connected components** of G_r .

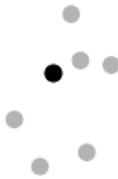
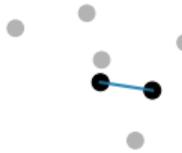
Example: Robust single-linkage



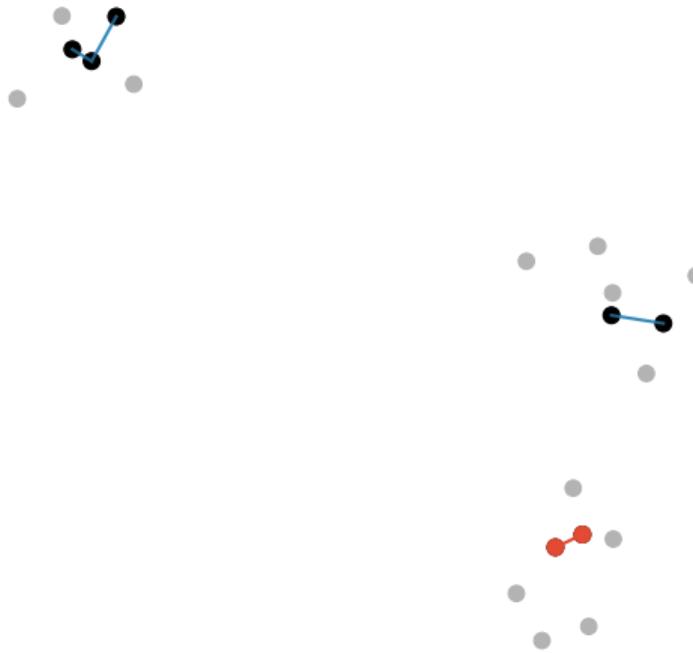
Example: Robust single-linkage



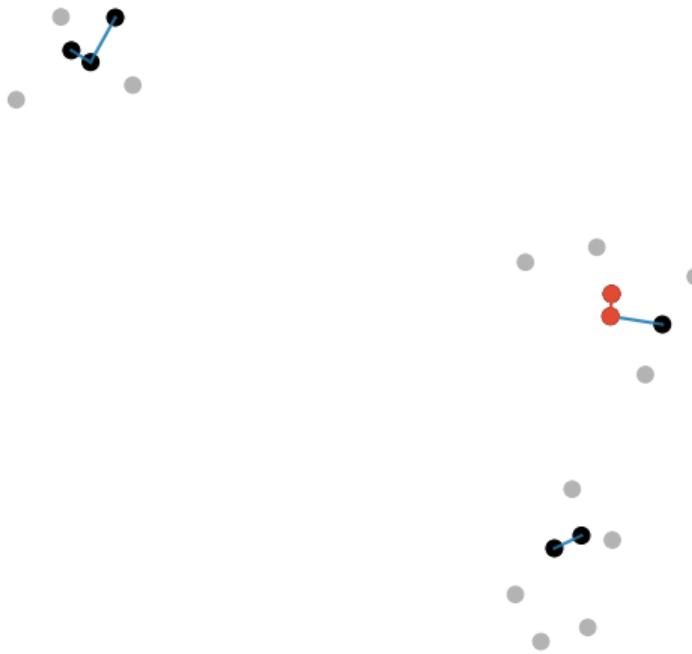
Example: Robust single-linkage



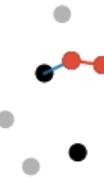
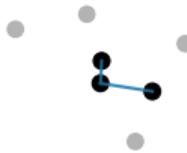
Example: Robust single-linkage



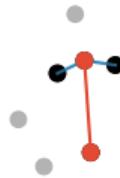
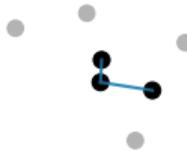
Example: Robust single-linkage



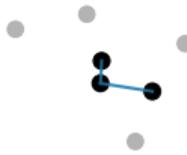
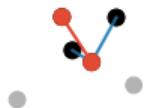
Example: Robust single-linkage



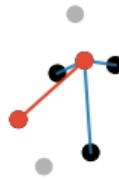
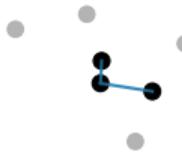
Example: Robust single-linkage



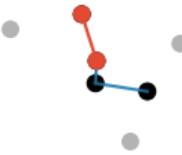
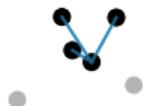
Example: Robust single-linkage



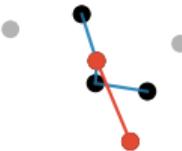
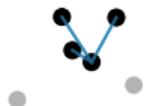
Example: Robust single-linkage



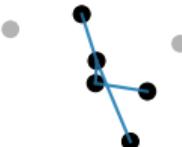
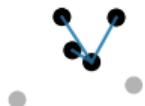
Example: Robust single-linkage



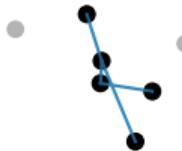
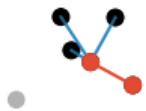
Example: Robust single-linkage



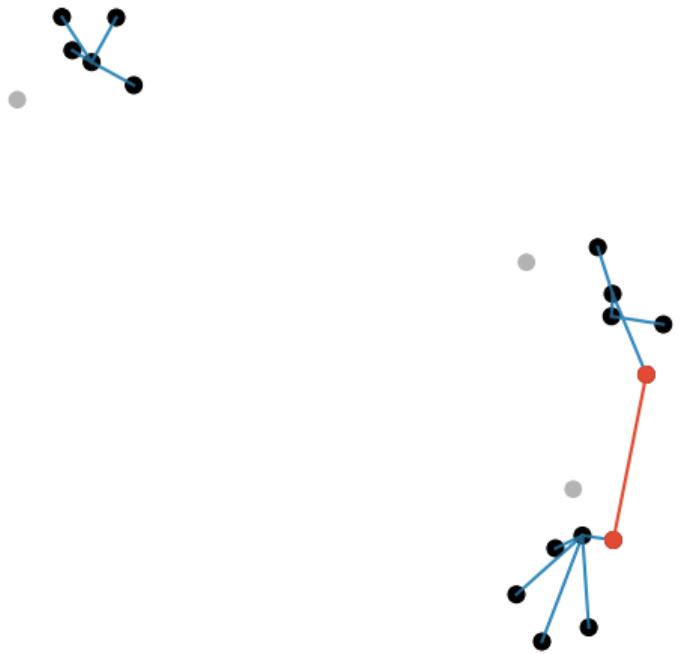
Example: Robust single-linkage



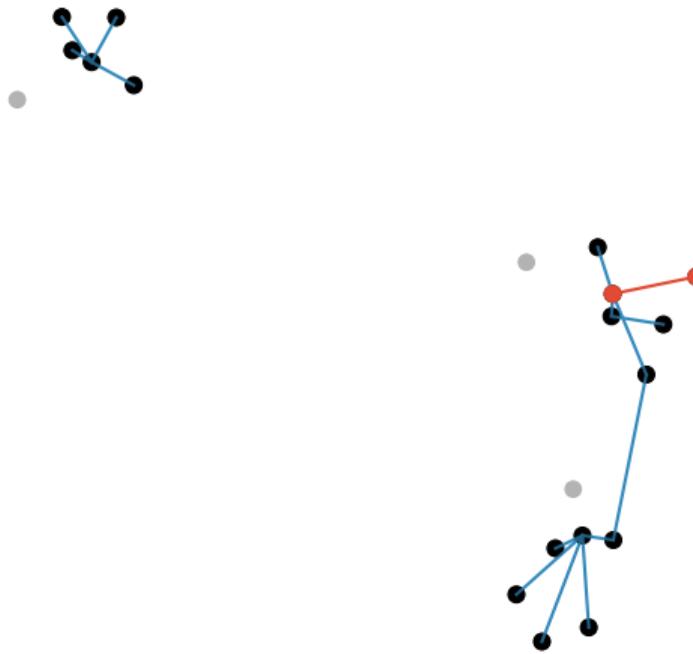
Example: Robust single-linkage



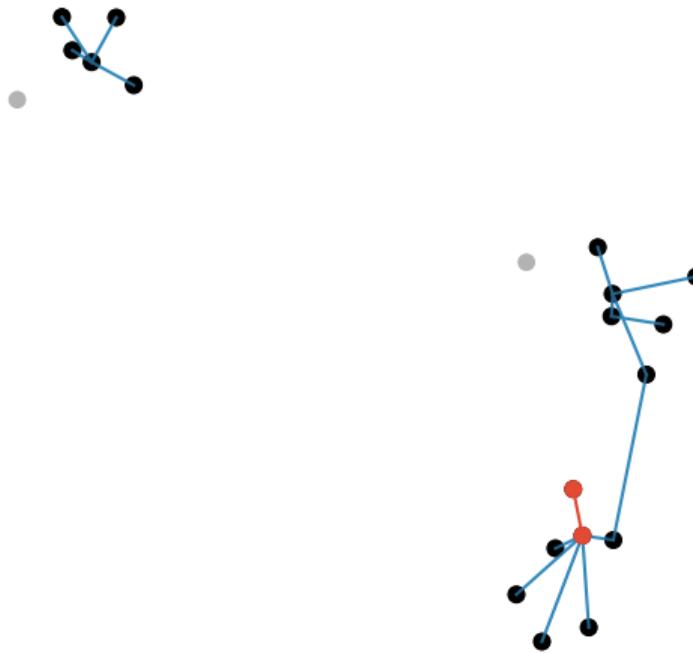
Example: Robust single-linkage



Example: Robust single-linkage



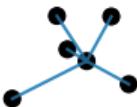
Example: Robust single-linkage



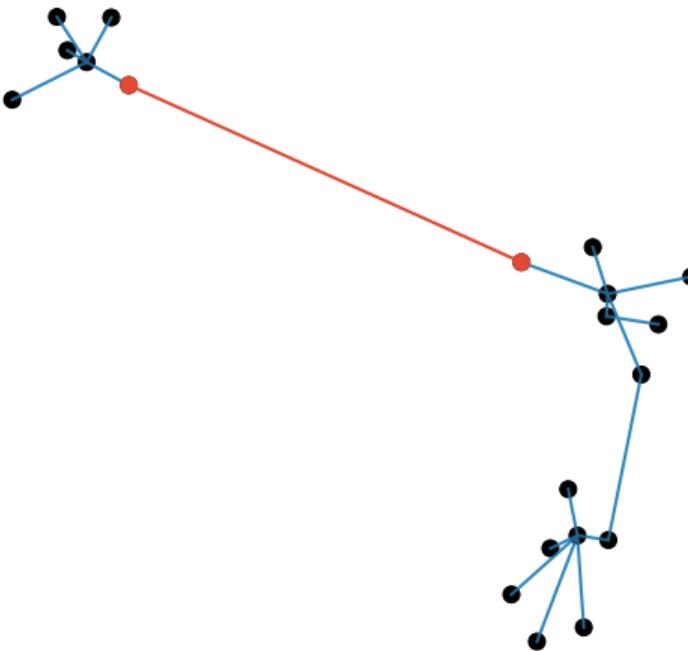
Example: Robust single-linkage



Example: Robust single-linkage



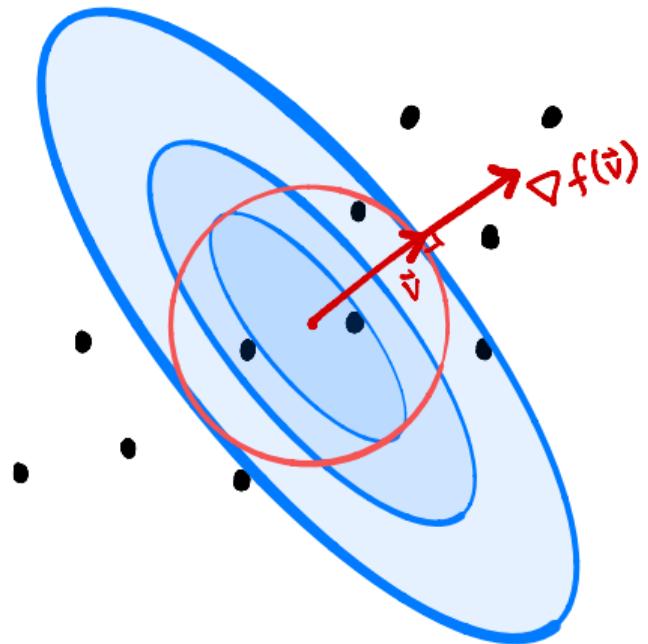
Example: Robust single-linkage



- ▶ **Robust single-linkage** recovers the density cluster tree (Chaudhuri and Dasgupta, 2010; Eldridge, Belkin, Wang 2015).
- ▶ Can be viewed as a transformation of metric, followed by single-linkage:

$$\tilde{d}(x, x') = \max \left\{ r_k(x), r_k(x'), \frac{1}{\alpha} d(x, x') \right\}.$$

- ▶ And therefore can be computed in $\Theta(kn^2)$ time.



CSE 151A
Intro to Machine Learning

Lecture 17 – Part 01
Dimensionality
Reduction

Announcements

- ▶ Midterm 02 is Friday! Covers Week 05 – Week 08.
 - ▶ Canvas Quiz. Tip: **don't** use Safari.
- ▶ There are no more mandatory homeworks.
 - ▶ I **will** be posting more plus problems (maybe another competition...).
 - ▶ I'll post some “essential” conceptual questions about Weeks 09 and 10, but they will not be turned in. Preparation for final exam.

Dimensionality Reduction

- ▶ Too many features hurts performance.
- ▶ Einstein: “Everything should be made as simple as possible, but no simpler.”

Dimensionality Reduction

- ▶ **Given:** a data set in high dimensions.
- ▶ **Reduce** dimensionality while preserving information.
- ▶ Why?
 - ▶ Faster, less memory.
 - ▶ High-dimensional data usually has redundancy.
 - ▶ Remove noisy/irrelevant features.

Example

0

1

2

3

4

5

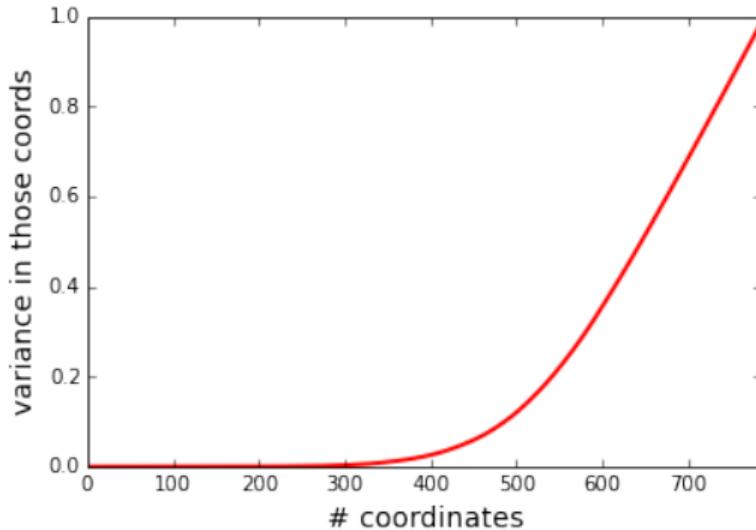
6

7

8

9

Example



Assumption

- ▶ Variance is **interesting**.
 - ▶ More variable features are more useful.

D-R Approach #1

- ▶ Start with data in d dimensions.
- ▶ Compute variance of each feature.
- ▶ Keep only the k features with most variance.

This is OK...

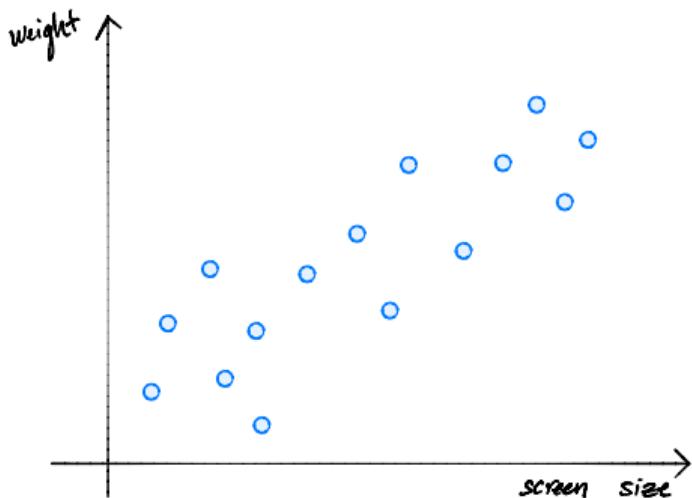
- ▶ ...but we can do better.
- ▶ **Problem:** features are often redundant.
- ▶ **Example:** height and weight

D-R Approach #2

- ▶ Find features which vary **together**.
 - ▶ **Example:** height and weight.
- ▶ Create **new** features which are combinations of old features.
- ▶ Keep best k combinations.

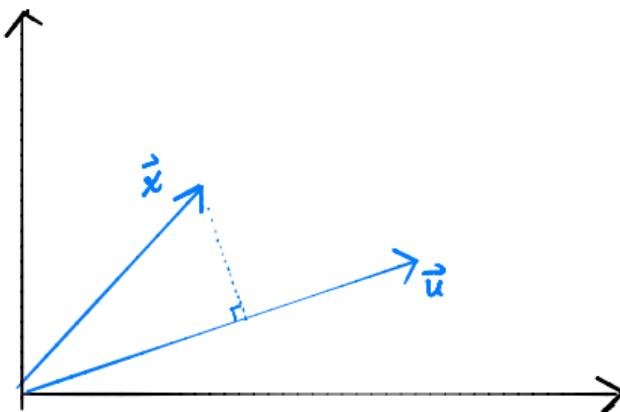
Example

Suppose we want just one feature.

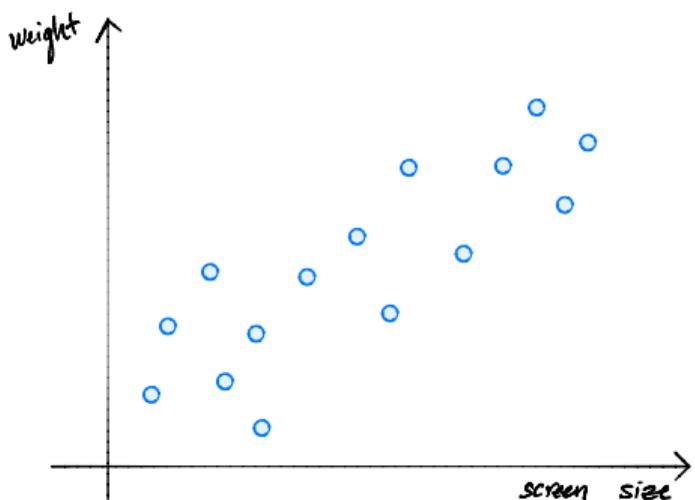


Projections

- ▶ The **projection** of $\vec{x} \in \mathbb{R}^d$ along the direction $\vec{u} \in \mathbb{R}^d$ (where \vec{u} is a unit vector) is $(\vec{x} \cdot \vec{u})\vec{u}$.



Projections



The Problem

- ▶ **Given:** data $\vec{x}^{(1)}, \dots, \vec{x}^{(n)} \in \mathbb{R}^d$
- ▶ **Map:** each data point $\vec{x}^{(i)}$ to a single feature, z_i .
 - ▶ Later: $\vec{z}^{(i)} \in \mathbb{R}^{d'}, d' \leq d$.
- ▶ **Idea:** map $\vec{x}^{(i)}$ by projecting it onto direction \vec{u} of maximum variance.
 - ▶ $z_i = \vec{x}^{(i)} \cdot \vec{u} = \sum_{j=1}^d u_j x_j^{(i)}$

Variance in a Direction

- ▶ Let \vec{u} be a unit vector.
- ▶ $\vec{x}^{(i)} \cdot \vec{u}$ is the new feature for $\vec{x}^{(i)}$.
- ▶ The variance of the new features is:

$$\begin{aligned}\text{Var}(z_1, \dots, z_n) &= \frac{1}{n} \sum_{i=1}^n z_i^2 \\ &= \frac{1}{n} \sum_{i=1}^n (\vec{x}^{(i)} \cdot \vec{u})^2\end{aligned}$$

Claim

- ▶ Suppose the data is **centered**.
 - ▶ The average of each feature is zero.
- ▶ Let C be the data's covariance matrix.
- ▶ Then the variance in the direction of \vec{u} is:

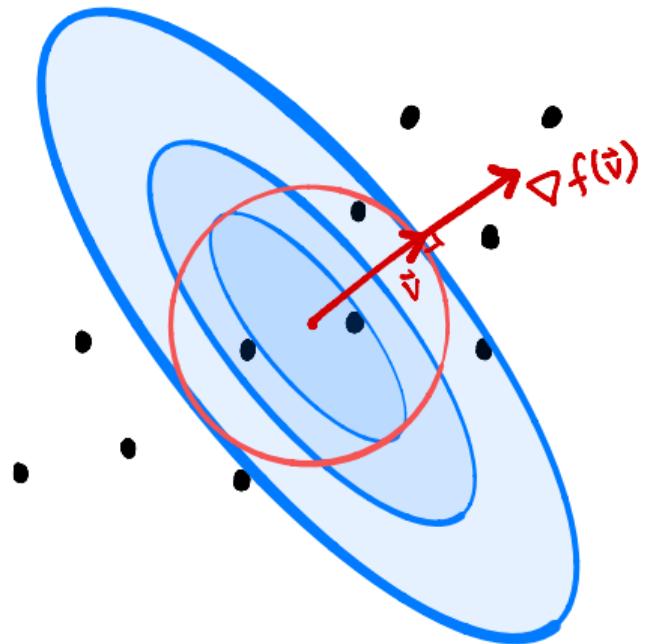
$$\text{Var}(z_1, \dots, z_n) = \vec{u}^T C \vec{u}$$

The Problem (More Formally)

- ▶ **Given:** covariance matrix C of centered data
 $\vec{x}^{(1)}, \dots, \vec{x}^{(n)} \in \mathbb{R}^d$
- ▶ **Find:** the **unit** vector \vec{u} maximizing $\vec{u}^T C \vec{u}$

The Problem (More Formally)

- ▶ **Given:** covariance matrix C of centered data
 $\vec{x}^{(1)}, \dots, \vec{x}^{(n)} \in \mathbb{R}^d$
- ▶ **Find:** the **unit** vector \vec{u} maximizing $\vec{u}^T C \vec{u}$
- ▶ **How?**



CSE 151A
Intro to Machine Learning

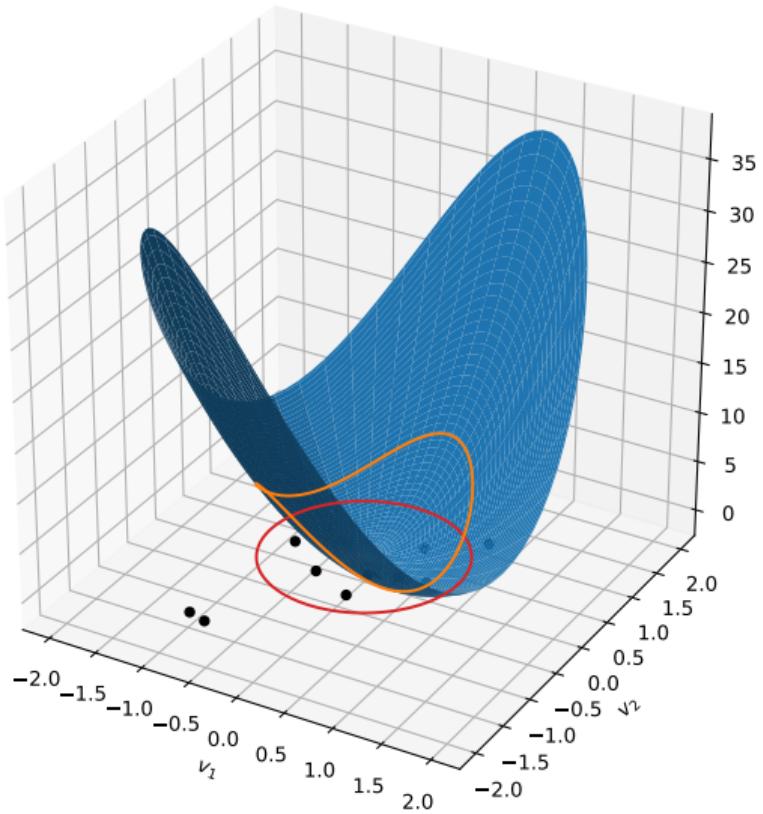
Lecture 17 – Part 02
Optimization

The Problem

- ▶ **Given:** covariance matrix C of centered data
 $\vec{x}^{(1)}, \dots, \vec{x}^{(n)} \in \mathbb{R}^d$
- ▶ **Find:** the **unit** vector \vec{u} maximizing $\vec{u}^T C \vec{u}$

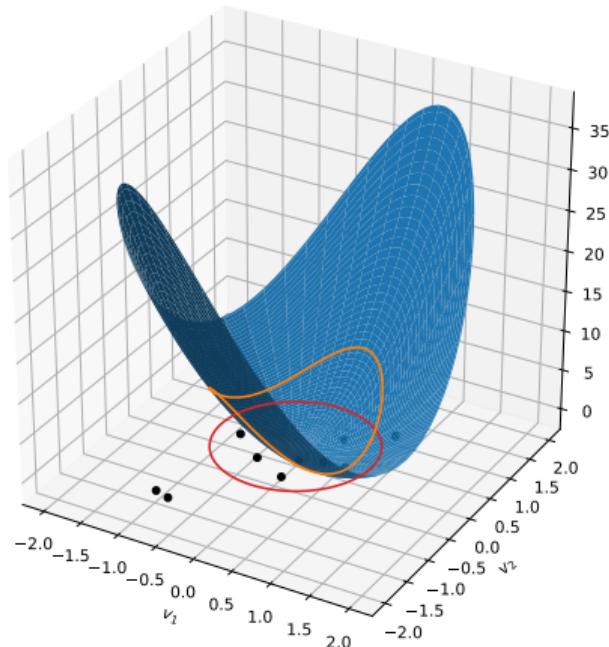
The Variance Function

- ▶ Define $f(\vec{v}) = v^T Cv$.
- ▶ **Claim:** f is **paraboloidal**.



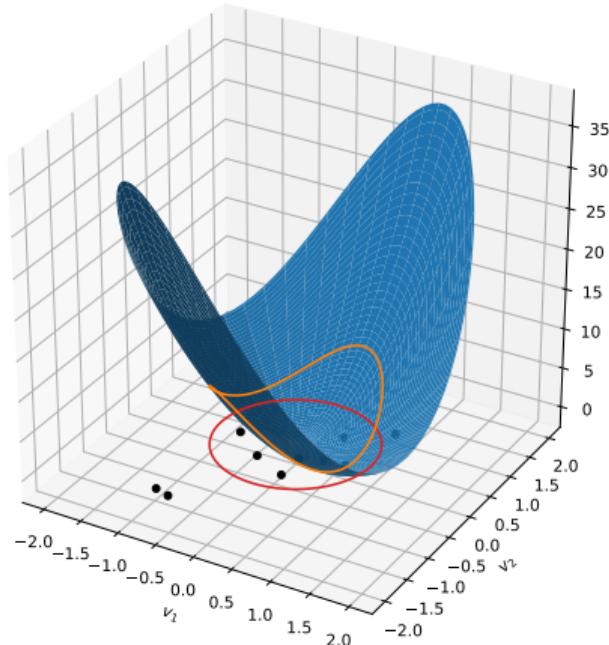
Optimization

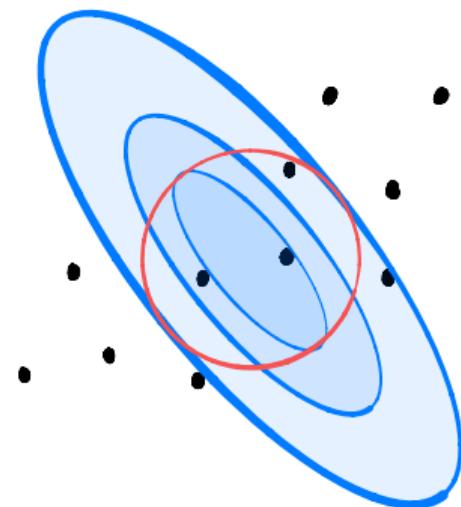
- ▶ Set gradient to zero, solve?

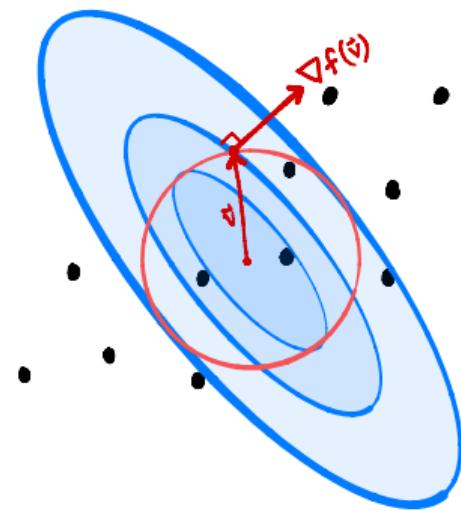


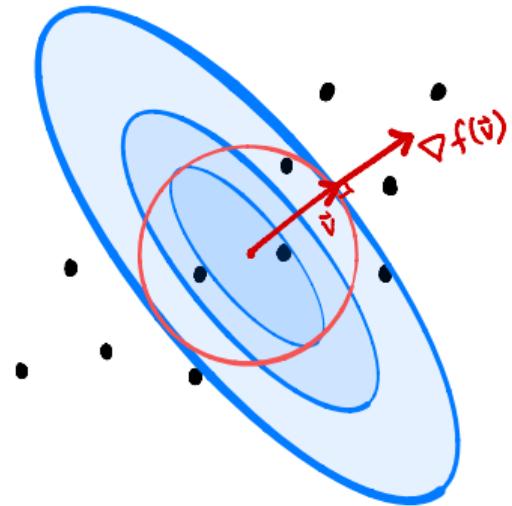
Optimization

- ▶ Set gradient to zero, solve? **No.**









The Solution

- ▶ We want to maximize $f(\vec{v})$ subject to $\|\vec{v}\| = 1$.
- ▶ Necessary: \vec{v} is in same direction as $\nabla f(\vec{v})$.

$$\nabla f(\vec{v}) = \lambda \vec{v} \tag{1}$$

- ▶ **Lagrange Multipliers**

Claim

► Remember: $f(\vec{v}) = \vec{v}^T C \vec{v}$

► Claim: $\nabla f(\vec{v}) = 2C\vec{v}$

► Condition (1) becomes:

$$2C\vec{v} = \lambda\vec{v}$$

► \vec{v} must be an **eigenvector** of C .

Remember: Eigenvectors

- ▶ An **eigenvector** of a matrix A is a vector \vec{u} such that $A\vec{u} = \lambda\vec{u}$. λ is called the **eigenvalue**.
- ▶ Matrices can have many eigenvector/eigenvalue pairs.
- ▶ If A ($d \times d$) is symmetric, positive definite, there is a set of d mutually orthogonal eigenvectors.

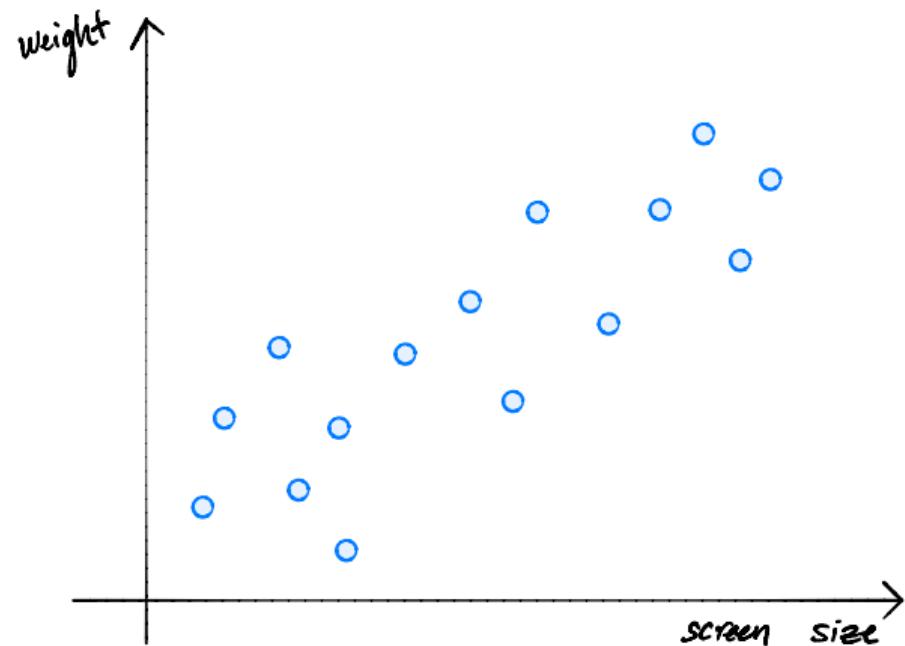
Recap

- ▶ **Goal:** Find unit vector \vec{u} maximizing $\vec{u}^T C \vec{u}$.
 - ▶ I.e., find unit vector in direction of maximum variance.
- ▶ Any solution must satisfy $2C\vec{u} = \lambda\vec{u}$
 - ▶ I.e., it must be an eigenvector of C .
- ▶ The **top eigenvector** of the covariance matrix points in direction of maximum variance.

Principal Components

- ▶ The **top eigenvector** of the covariance matrix is called the **principal component**.
- ▶ It points in the direction of maximum variance.
- ▶ Idea: it is the “most interesting” direction.

Principal Component Projections

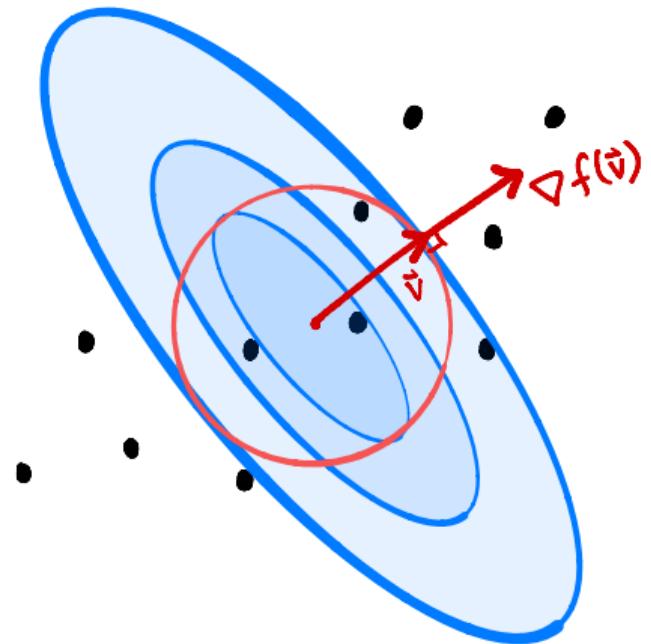


Principal Component Analysis

- ▶ **Given:** data $\vec{x}^{(1)}, \dots, \vec{x}^{(n)} \in \mathbb{R}^d$
- ▶ **Map:** each data point $\vec{x}^{(i)}$ to a single feature, z_i .
- ▶ **PCA:** Let $z_i = \vec{x}^{(i)} \cdot \vec{u}$, where \vec{u} is top eigenvector of covariance matrix.

Next Time

- ▶ **Given:** data $\vec{x}^{(1)}, \dots, \vec{x}^{(n)} \in \mathbb{R}^d$
- ▶ **Map:** each data point $\vec{x}^{(i)}$ to a lower-dimensional vector, $\vec{z}^{(i)} \in \mathbb{R}^{d'}, d' \leq d$.



CSE 151A
Intro to Machine Learning

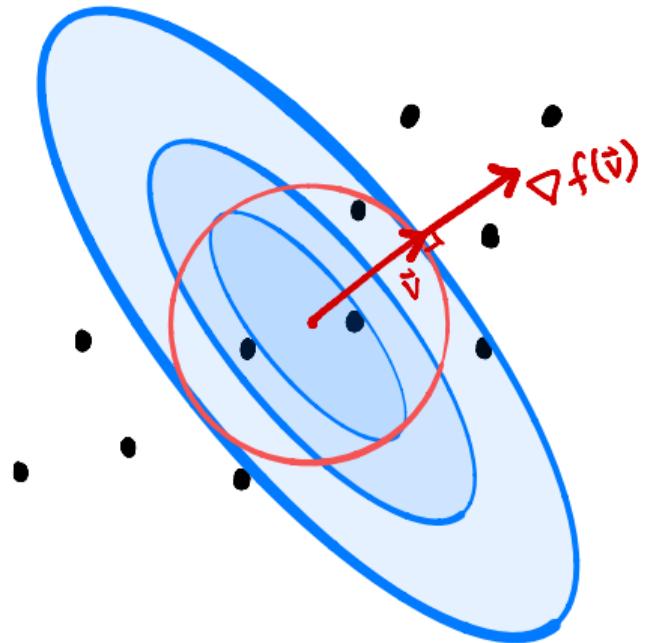
Lecture 18 – Part 01
Ending the Quarter

Final Exam

- ▶ Section of Final Exam covering Weeks 09 and 10 is cancelled.
- ▶ (Optional) Redemption Sections will still be given.

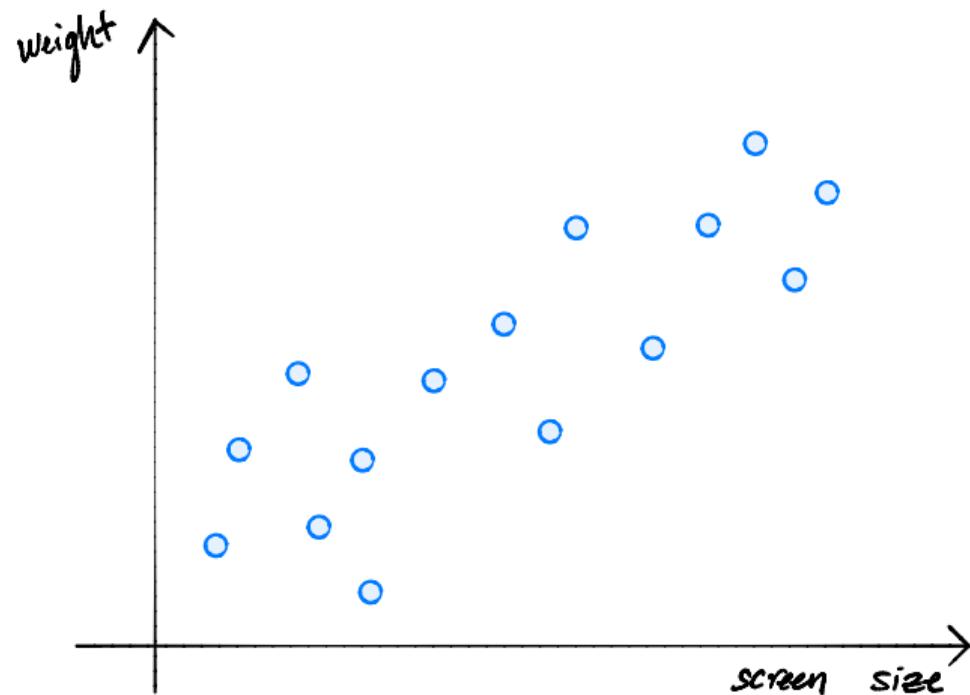
Grades

- ▶ Midterm 02 grades will be posted tonight or tomorrow.
- ▶ All HWs (except for HW 08) will be posted soon.



CSE 151A
Intro to Machine Learning

Lecture 18 – Part 02
**PCA in Many
Dimensions**



Recap: Principal Components

- ▶ **Goal:** Find unit vector \vec{u} maximizing $\vec{u}^T C \vec{u}$.
 - ▶ I.e., find unit vector in direction of maximum variance.
- ▶ Any solution must satisfy $2C\vec{u} = \lambda\vec{u}$
 - ▶ I.e., it must be an eigenvector of C .
- ▶ The **top eigenvector** of the covariance matrix points in direction of maximum variance.

Principal Component Analysis

- ▶ **Given:** data $\vec{x}^{(1)}, \dots, \vec{x}^{(n)} \in \mathbb{R}^d$
- ▶ **Map:** each data point $\vec{x}^{(i)}$ to a single feature, z_i .
- ▶ **PCA:** Let $z_i = \vec{x}^{(i)} \cdot \vec{u}$, where \vec{u} is top eigenvector of covariance matrix.

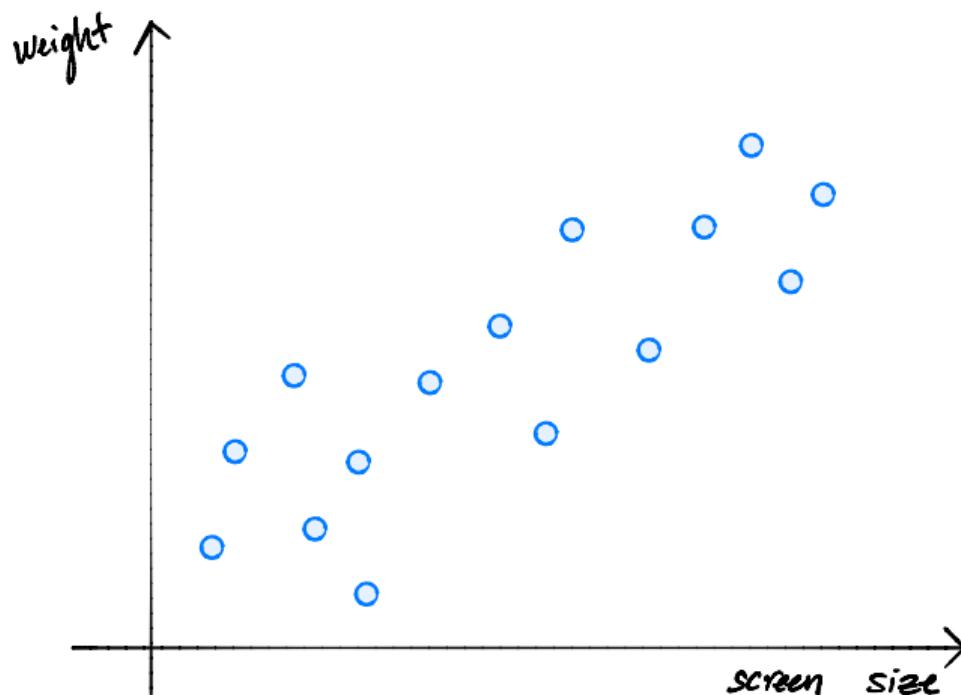
PCA in Many Dimensions

- ▶ We lose a lot of information when we project from \mathbb{R}^d to a single number.
- ▶ Instead of a single number, represent $\vec{x}^{(i)} \in \mathbb{R}^d$ as a smaller vector, $\vec{z}^{(i)} \in \mathbb{R}^{d'}$.

PCA in Many Dimensions

- ▶ First PCA feature: project $\vec{x}^{(i)}$ onto principal component, $\vec{u}^{(1)}$
 - ▶ $\vec{u}^{(1)}$ is vector maximizing $\vec{u}^T C \vec{u}$.
- ▶ Second PCA feature: project $\vec{x}^{(i)}$ onto ???
- ▶ Third PCA feature: project $\vec{x}^{(i)}$ onto ???
- ▶ ...

PCA in Many Dimensions



PCA in Many Dimensions

- ▶ Second PCA feature: project $\vec{x}^{(i)}$ onto $\vec{u}^{(2)}$.
 - ▶ $\vec{u}^{(2)}$ is the **second principal component**
 - ▶ Found by maximizing $\vec{u}^T C \vec{u}$ subject to constraint that $\vec{u}^{(2)}$ is orthogonal to $\vec{u}^{(1)}$

PCA in Many Dimensions

- ▶ Third PCA feature: project $\vec{x}^{(i)}$ onto $\vec{u}^{(3)}$.
 - ▶ $\vec{u}^{(3)}$ is the **third principal component**
 - ▶ Found by maximizing $\vec{u}^T C \vec{u}$ subject to constraint that $\vec{u}^{(3)}$ is orthogonal to $\vec{u}^{(1)}$ and $\vec{u}^{(2)}$

PCA in Many Dimensions

- ▶ **Goal:** find k unit vectors $\vec{u}^{(1)}, \dots, \vec{u}^{(k)}$ such that:
 - ▶ $(\vec{u}^{(1)})^T C \vec{u}^{(1)}$ is maximized
 - ▶ $(\vec{u}^{(2)})^T C \vec{u}^{(2)}$ is maximized s.t. $\vec{u}^{(2)} \perp \vec{u}^{(1)}$
 - ▶ ...
 - ▶ $(\vec{u}^{(k)})^T C \vec{u}^{(k)}$ is maximized s.t. $\vec{u}^{(k)} \perp \vec{u}^{(1)}, \dots, \vec{u}^{(k-1)}$

The New Features

- ▶ Suppose we have found $\vec{u}^{(1)}, \dots, \vec{u}^{(k)}$.
- ▶ The new feature vector for $\vec{x}^{(i)}$ is:

$$\vec{z}^{(i)} = (\vec{x}^{(i)} \cdot \vec{u}^{(1)}, \vec{x}^{(i)} \cdot \vec{u}^{(2)}, \dots, \vec{x}^{(i)} \cdot \vec{u}^{(k)})^T$$

The New Features

- ▶ Equivalently, define:

$$U = \begin{pmatrix} \leftarrow & \vec{u}^{(1)} & \rightarrow \\ \leftarrow & \vec{u}^{(2)} & \rightarrow \\ & \cdots & \\ \leftarrow & \vec{u}^{(k)} & \rightarrow \end{pmatrix}$$

- ▶ Then $\vec{z}^{(i)} = U\vec{x}^{(i)}$

The Solution

- ▶ How do we find $\vec{u}^{(1)}, \dots, \vec{u}^{(k)}$?
- ▶ We know $\vec{u}^{(1)}$ is the top eigenvector of C .
- ▶ Claim: $\vec{u}^{(i)}$ is the i th eigenvector of C .

Spectral Theorem

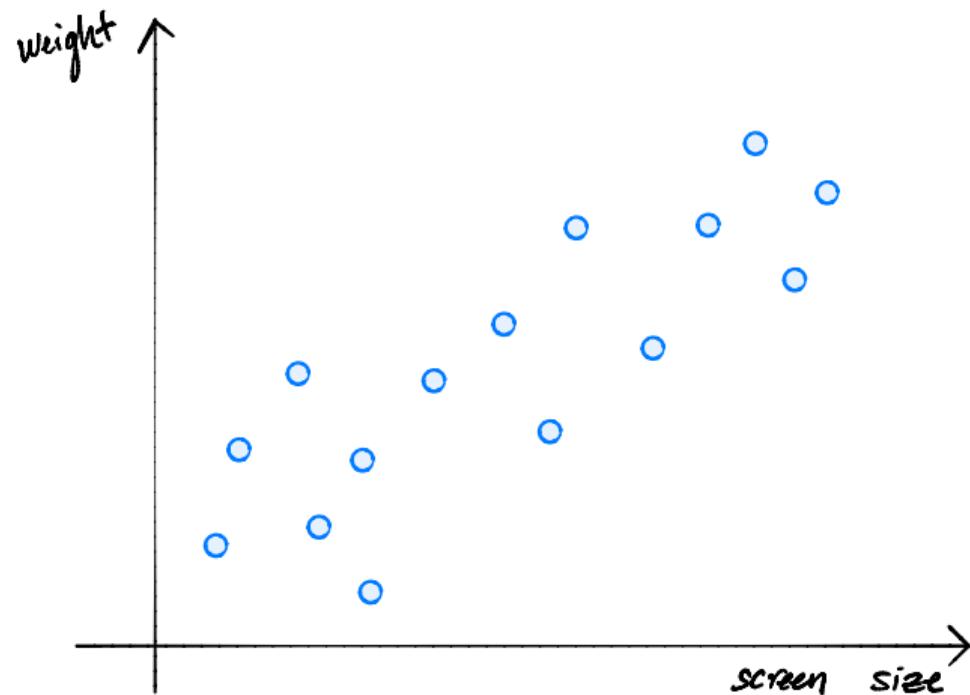
- ▶ Let C be the $d \times d$ covariance matrix of X .
- ▶ In $O(d^3)$ time, we can compute its **eigendecomposition**, consisting of
 - ▶ real **eigenvalues** $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$
 - ▶ corresponding **eigenvectors** $\vec{u}^{(1)}, \dots, \vec{u}^{(d)} \in \mathbb{R}^d$ that are orthonormal (unit length and at right angles to each other)

PCA

- ▶ Suppose we wish to map a data set $\vec{x}^{(1)}, \dots, \vec{x}^{(n)}$ to k dimensions while retaining as much variance as possible.
- ▶ **Solution:**
 - ▶ Compute top k eigenvectors of covariance.
 - ▶ Place them row-wise into a matrix U .
 - ▶ $\vec{x}^{(i)} \mapsto U\vec{x}^{(i)}$.

PCA Revisited

- ▶ We have seen PCA as an optimization problem.
- ▶ Another (equivalent) view: decorrelation.

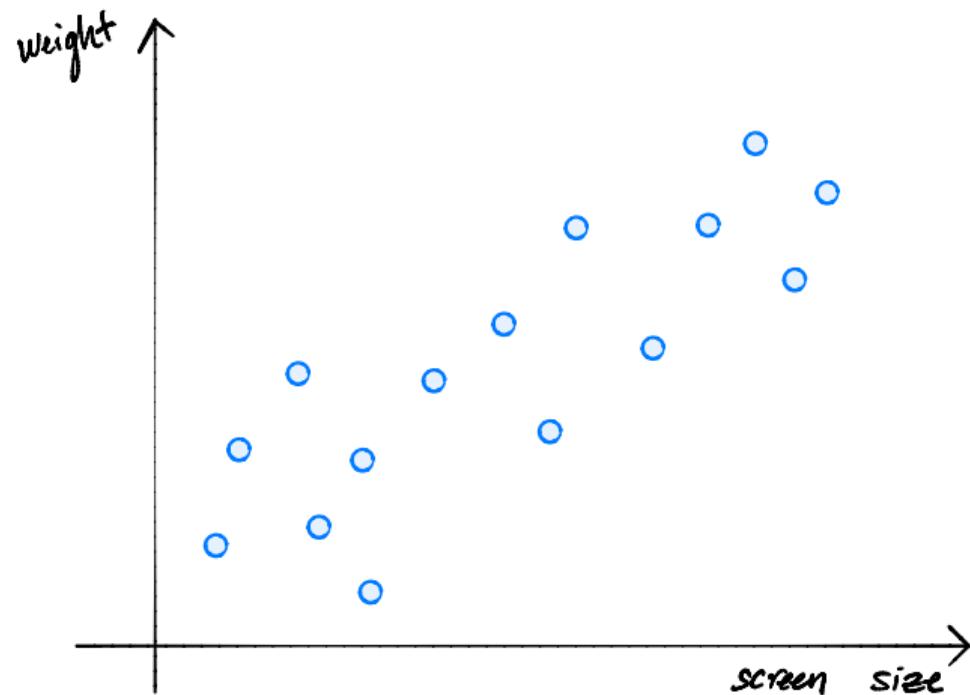


Decorrelation

- ▶ **Goal:** find orthonormal basis in which data is decorrelated
 - ▶ Covariance matrix of new features is diagonal.
- ▶ **Solution:** use as basis the eigenvectors of covariance matrix.

Reconstruction

- ▶ The whole goal of PCA is to reduce dimensionality.
 - ▶ Example: turn a 784-dimensional image vector into 200-dimensional feature vector.
- ▶ But sometimes it is fun to go the other direction:
 - ▶ Take result of PCA and **reconstruct** the original image.



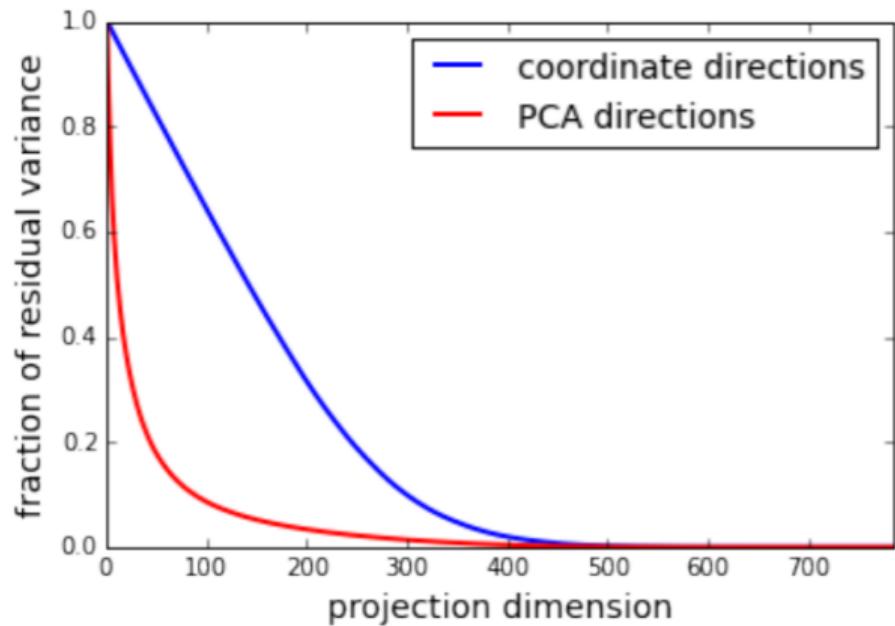
Reconstruction

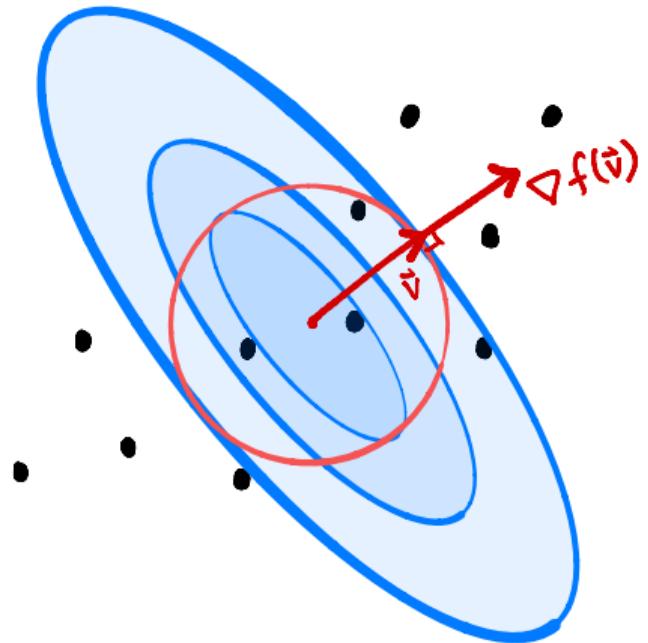
- ▶ Suppose original data is d dimensional.
- ▶ Project onto k eigenvectors $\vec{u}^{(1)}, \dots, \vec{u}^{(k)} \in \mathbb{R}^d$.
 - ▶ $\vec{z}^{(i)} = (\vec{x}^{(i)} \cdot \vec{u}^{(1)}, \dots, \vec{x}^{(i)} \cdot \vec{u}^{(k)})^T$
- ▶ Reconstruction of $\vec{x}^{(i)}$ from $\vec{z}^{(i)}$:

$$\vec{x}^{(i)} \approx \vec{z}_1^{(i)} \vec{u}^{(1)} + \vec{z}_2^{(i)} \vec{u}^{(2)} + \dots + \vec{z}_k^{(i)} \vec{u}^{(k)}$$

PCA in Practice

- ▶ PCA is often used in **preprocessing** before classifier is trained, etc.
- ▶ Must choose number of dimensions, k .
- ▶ One way: cross-validation.
- ▶ Another way: the elbow method.





CSE 151A
Intro to Machine Learning

Lecture 18 – Part 04
Demos

<https://go.ucsd.edu/3exnmrw>

CSE 151A

Intro to Machine Learning

Lecture (-1) – Part 01
**What is Machine
Learning?**

Final Exam Redemption

Please reach out if you're feeling overwhelmed

jeldridge@ucsd.edu

Today

1. Recap
2. How did you get into ML/DS?
3. Do I need an MS/PhD to get a job in ML?
4. How do I start an ML project?
5. What is the difference between AI/ML/Stats/DS?
6. Thank you

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.

What is Machine Learning?

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

What is Machine Learning?

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.

What is Machine Learning

- ▶ We have learned different ways of formulating classification, regression, clustering, etc.
- ▶ Knowing how a method is formulated helps you understand which methods should be used and how.

CSE 151A

Intro to Machine Learning

Lecture (-1) – Part 02
How did you get into
ML/DS?

CSE 151A

Intro to Machine Learning

Lecture (-1) – Part 03
**Do I need an MS/PhD to
get a job in ML?**

CSE 151A

Intro to Machine Learning

Lecture (-1) – Part 04
**How do I start an ML
project?**

CSE 151A

Intro to Machine Learning

Lecture (-1) – Part 05

**What is the difference
between
AI/ML/Stats/DS?**

CSE 151A

Intro to Machine Learning

Lecture (-1) – Part 06

Thank you