Machine Learning with Logistic Regression

Deep Learning

Brad Quinton, Scott Chin

Case Study: Toronto Raptors Internship

- Good news! You have been hired as an intern with the Toronto Raptors. Although you know nothing about basketball, you are keen to help Canada's only NBA basketball team repeat their championship…

Case Study: Toronto Raptors Internship

- **Day 1:** First meeting with the coaching staff:

“I’m gald you are here! We need every advantage we can get. The entry draft is coming up, and we need some of this ’AI’ that I keep hearing about to help us decide which college players we should select…

*…I’ve heard that AI needs data, so here are the prospects and the results from this year’s **Draft Combine**, good luck!”*

Case Study: NBA Draft Combine

- A quick search on web shows us the Draft Combine is a set of stats on each player who will be eligible for the draft, there are 4 categories: [Anthropometric Stats](), [Spot-Up Shooting Stats](), [Non-Stationary Shooting Stats](), [Strength & Agility Stats]()

PLAYER	POS	BODY FAT %	HAND LENGTH (INCHES)	HAND WIDTH (INCHES)	HEIGHT W/O SHOES	HEIGHT W/ SHOES	STANDING REACH	WEIGHT (LBS)	WINGSPAN
Nickeil Alexander-Walker	SG	5.90%	8.50	8.75	6' 4.25"	6' 5.5"	8' 6"	203.8	6' 9.5"
RJ Barrett	SF	-%	-	-				-	
Charles Bassey	C	8.50%	9.25	9.50	6' 8.75"	6' 10"	9' 1.5"	239.0	7' 3.5"
Darius Bazley	PF	3.60%	9.00	9.75	6' 7.75"	6' 9"	8' 11"	208.4	7' 0"
Bol Bol	C	7.10%	9.25	9.50	7' 0.75"	7' 2.5"	9' 7.5"	208.0	7' 7"
Jordan Bone	SG	5.00%	7.50	9.25	6' 1.5"	6' 2.75"	7' 11"	179.0	6' 3.25"
Brian Bowen II	SF	6.50%	8.50	9.75	6' 6.25"	6' 7.5"	8' 7"	200.0	6' 10"

https://stats.nba.com/draft/combine-anthro/?SeasonYear=2019-20

Brad Quinton, Scott Chin

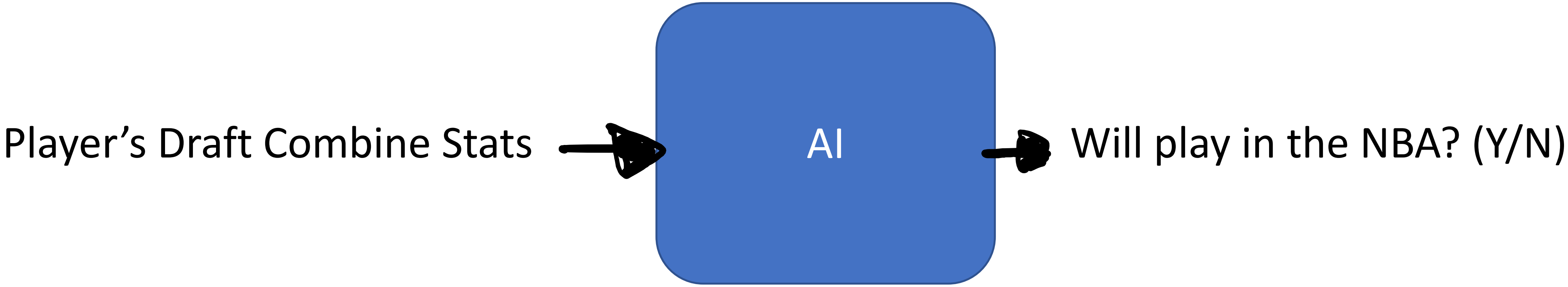
Case Study: Uh-oh. Now What!?!

- Well, let’s frame the problem… what are the inputs and outputs
- **Input:** Set of data about each player in the draft.
- **Output:** A list of players that we believe should be drafted as potential new players.
- With a little more digging we find out that as good as they are in college, most drafted players are never actually play in the NBA

Case Study: The “AI”

- So an easy and reasonable simplification is to try to predict which players will actually play in the NBA, and then suggest those as good draft picks….

- It is likely the Coaching staff want something like this:



Case Study: How can we build this AI?

- **Option #1:** Since we don't know anything about basketball we could hire a *basketball expert* to help us understand what is important when selecting a draft picks. Working with the expert we could write the prediction software. (This is usually called an Expert System)

```
if (height > 2.1) AND (weight > 200) then
  if (wingsnap/reach) > 0.9 then

    . . .

    set play_in_NBA 1
   else
    set play_in_NBA 0
```

Case Study: How can we build this AI?

- Unfortunately, interns don't get budgets to hire basketball experts, and besides, we want to do *better* than the experts!

- Luckily, we remember reading about machine learning. Maybe there is another option!

- **Option #2:** Use machine learning (ML) to train the AI, thus avoiding the need for the expert. And maybe even beating them.

Case Study: How can we build this AI?

- But, wait. What do we train with?

- This is **the fundamental challenge of ML:** The machine can only 'learn' if we have examples that we can use to train it!

- Then we remember, there is a draft ever year! We can look at past results to build examples: if we find out who eventually played in the NBA (and critically, who didn't) along with their Draft Combine results we will have the examples we need.

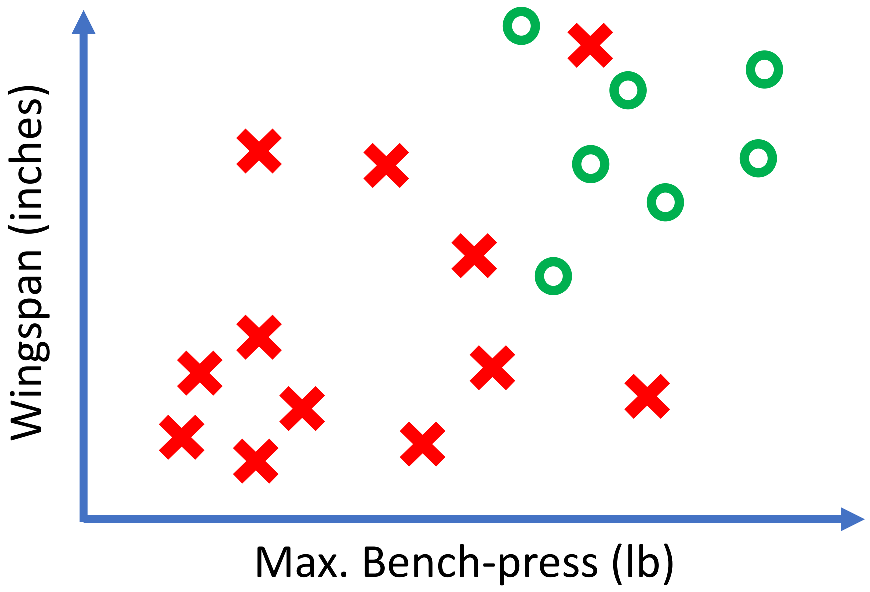
Case Study: The Data

- Great, now we have some data!

- For instance, we could find each player who participated in the Draft Combine over the last 20 years, and check if they eventually played in the NBA or not

- It is intuitive that there should be a relationship between these stats and ability to play in the NBA (for our sake, lets hope so!)

Case Study: Take a Peak at the Data*



*note this not the real data, just an example to give intuition for now....

Case Study: Dealing with Many Variables

- But we have dozens of data points for each player, not just 2 … there is no good way to just "eye-ball" in dozens of dimensions

- How do we find the "best fit" across an arbitrary number of dimensions? **Logistic Regression**.

- Let's learn about Logistic Regression and then get back to our case study…

Logistic Regression

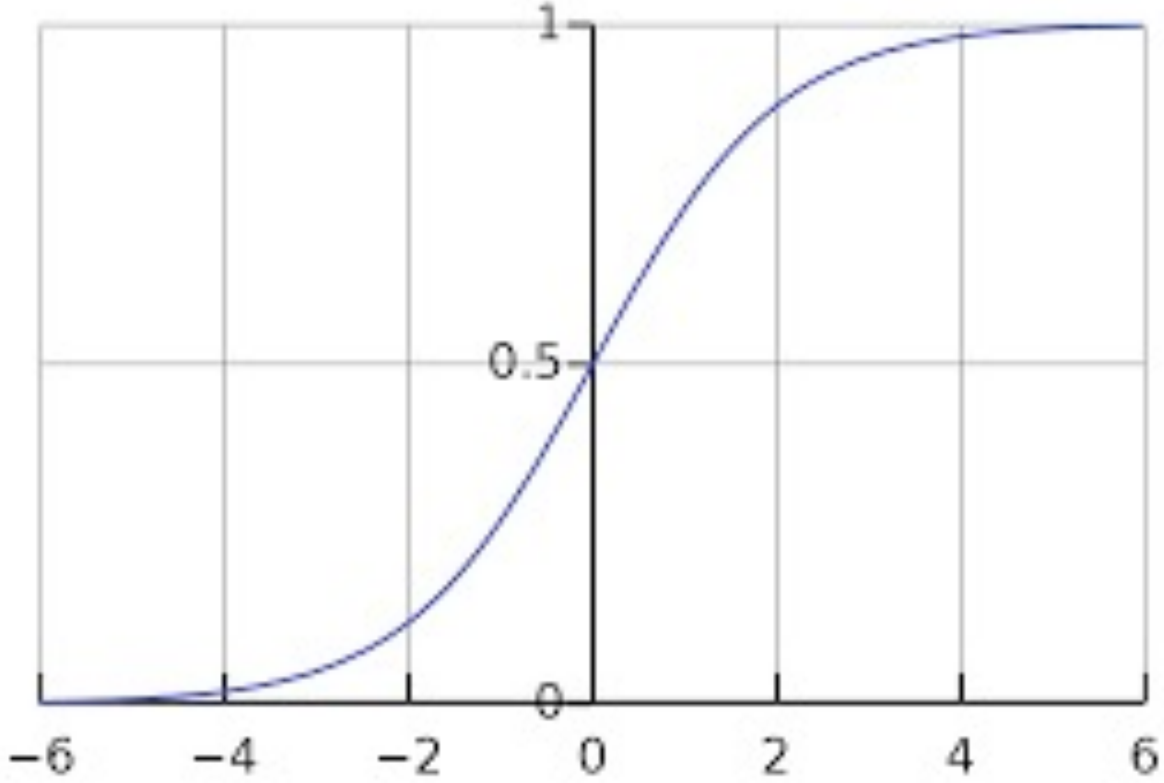
- Logistic regression is a technique that **assumes** that we can make a prediction (often called hypothesis in ML), based on a **linear combination** of the inputs*:*

$$z = w_0 x_0 + w_1 x_1 + \ldots + w_n x_n + b$$

- Since we are trying to classify into 2 groups (**Binary Classification**), we can use 0 and 1 to represent each. It turns out, that this is easier if we impose a function that only outputs values between 0 and 1, for example the *sigmoid* function

Sigmoid Function

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



- This sigmoid "forces" large values to be '1', and small values to be '0'

Logistic Regression

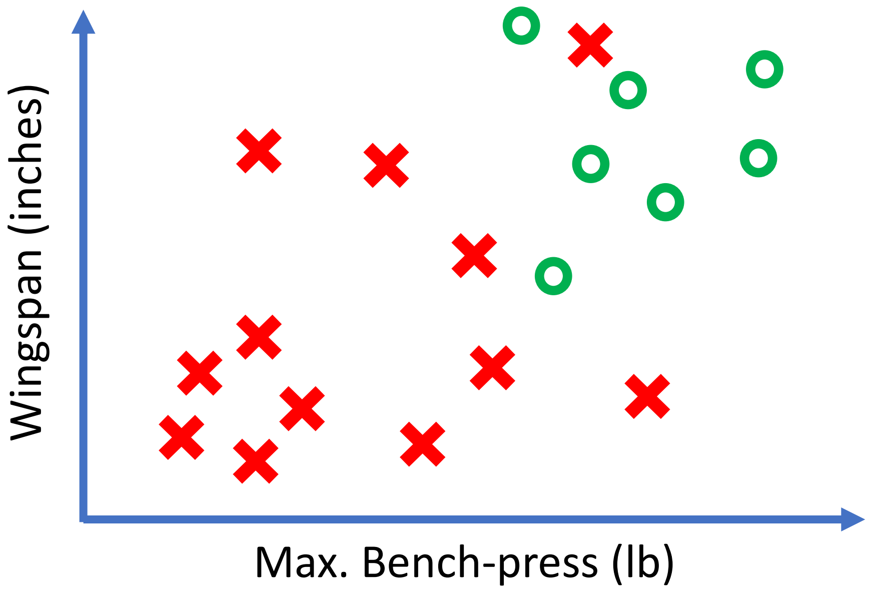
- Adding the sigmoid we get our final working logistic regression equation:

$$a \ = \sigma(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b)$$

- Assuming we had the correct values for $w_1, w_2, \ldots, w_n$ and b we could use this equation to predict as follows:

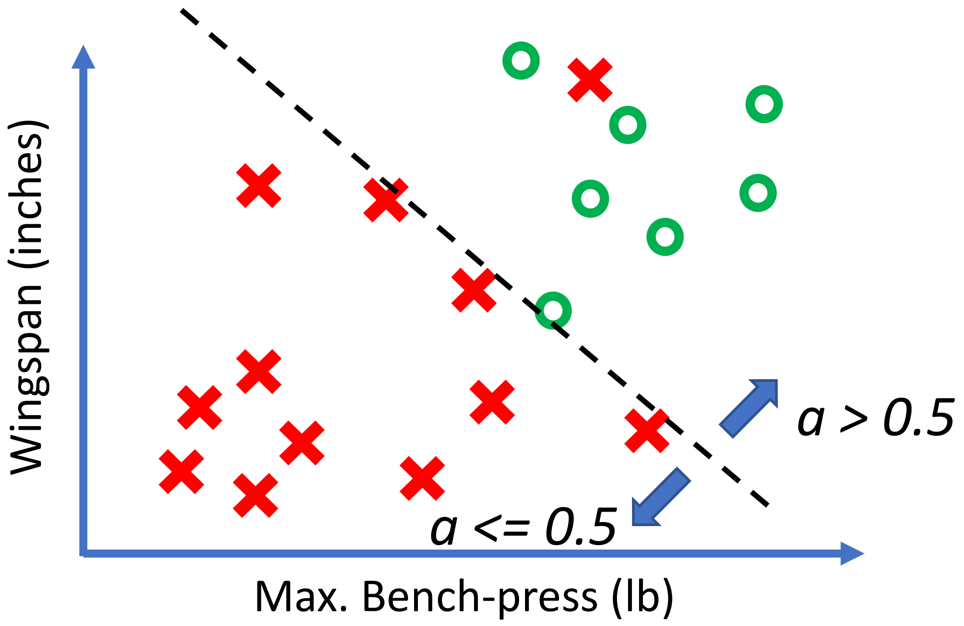
  - If a > 0.5 we predict '1' or 'Plays in NBA"
  - If a <= 0.5 we predict '0' or "Doesn't play in the NBA"

Logistic Regression



Logistic Regression

defined by some $w_1, w_2, \dots, w_n, b$



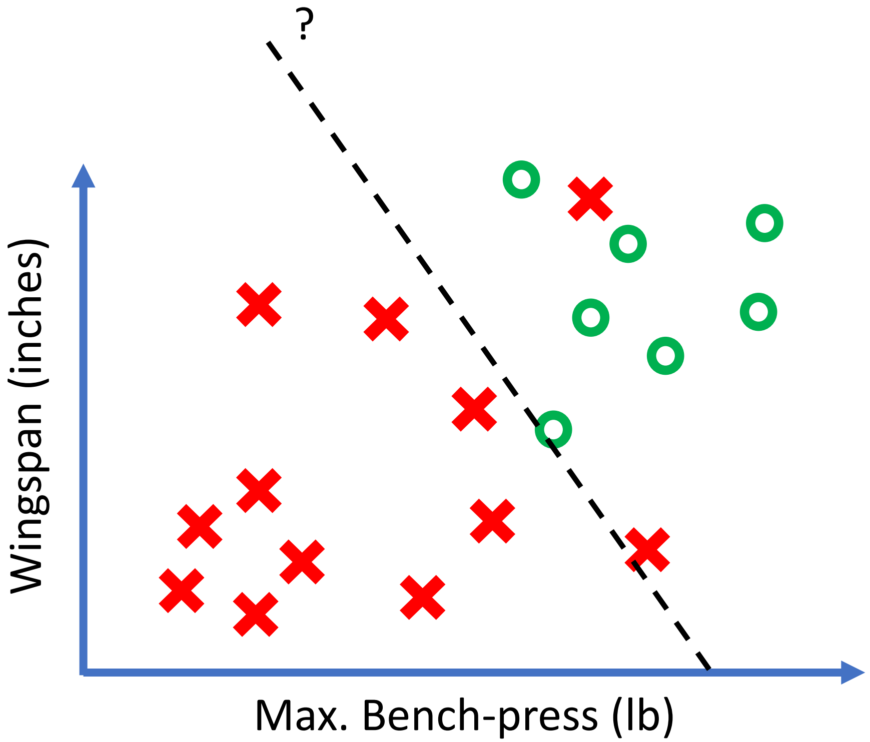
Logistic Regression

- Of course, the question now is: “How to we find $w_1, w_2, \dots, w_n$ and b?”

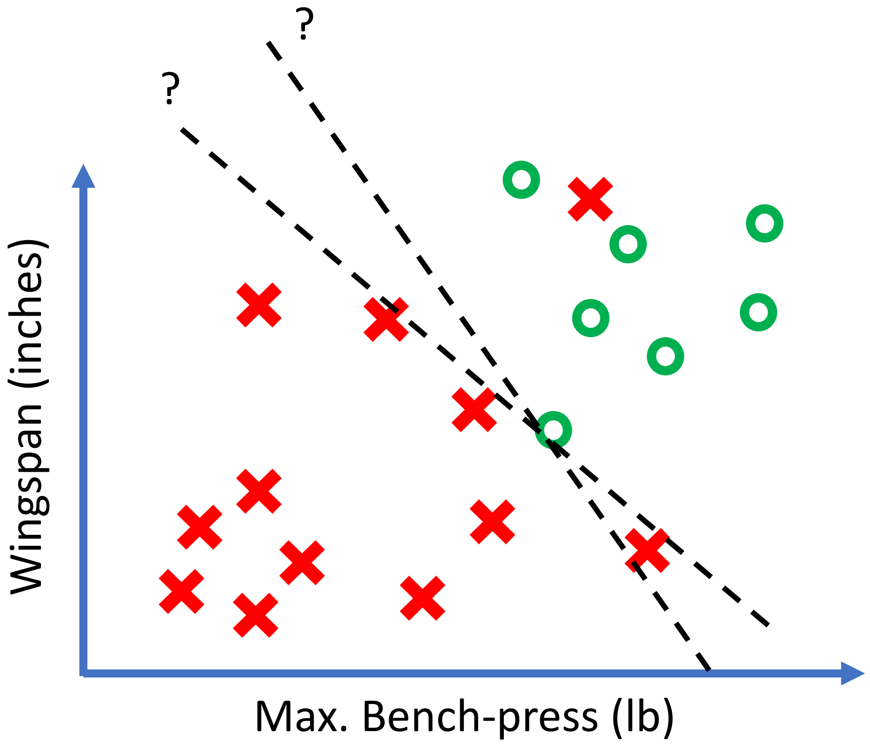
- In fact, that question is the heart of ML; $w_1, w_2, \dots, w_n$ and b are called parameters, and finding the “correct” parameters is what ML is all about

- There are many ways to find parameters: guess and test, simulated annealing, genetic algorithms, *etc.* but a technique called **Gradient Descent** works very well, as we will see…

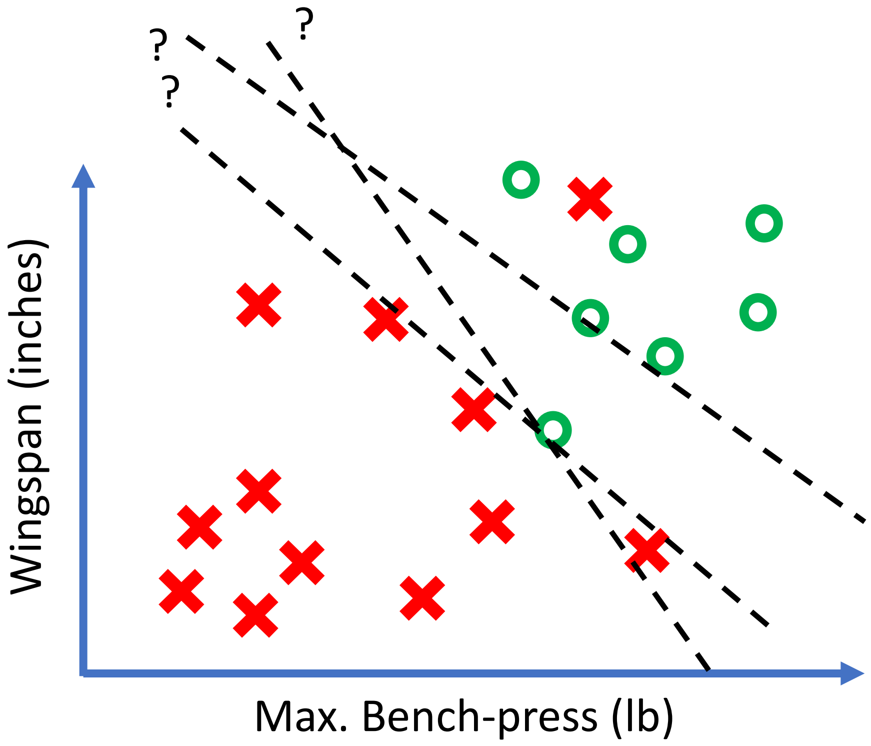
Logistic Regression



Logistic Regression



Logistic Regression



Cost Function

- First, what we need is a way to compare combinations of $w_1, w_2, \ldots, w_n$ and b to know which works best

- We need to create a **cost function**, J which will be a 'measure of fitness' of any given selection of $w_1, w_2, \ldots, w_n$ and b

- Then if $J(w_1', w_2', \ldots, w_n', b') < J(w_1, w_2, \ldots, w_n, b)$, then $w_1', w_2', \ldots, w_n', b'$ is a better set of parameters selection than $w_1, w_2, \ldots, w_n, b$

Cost Function

- What should use for the cost function?

- One solution is that we simply look at our classification accuracy (*i.e.* how many are right and how many are wrong?)

Accuracy = (right answers/total answers)

- This will certainly help us compare solutions, and for random "guess and test" this would be OK, however there are **A LOT** of combinations of parameters to try and we need to be smarter

Cost Function

- Ideally, we want a cost function that provides us guidance as to the next guess...
- But how? **Calculus to the rescue!** Recall, the derivative of a function represents the rate of change of the function, so it tells you “where the function is going”
- The heart of Gradient Descent is adjusting parameters, with an understanding of “where you are going”

Parameter Adjustment: Where to go next?

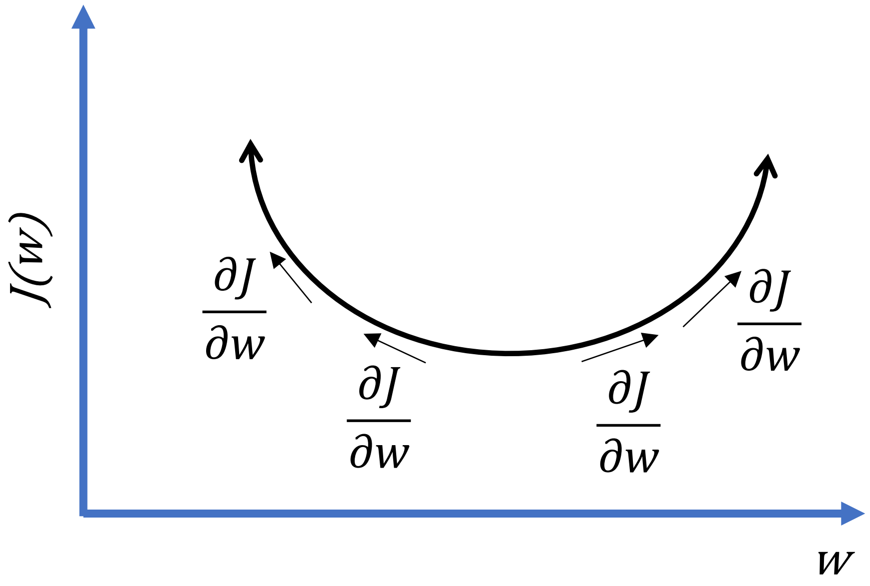
- So if with a cost function in terms of the parameters, the derivative would tell us which direction to adjust the parameters…

if, $J(w_1, w_2, \ldots, w_n, b)$ is the overall cost, then

$\frac{\partial J(w_1, w_2, \ldots, w_n, b)}{\partial w_1}$ is the rate of change of the cost w.r.t w_1

- This this **very valuable** if you are asking the question "how should I change w_1 to improve the cost?"

Adjusting the Parameters



- We can improve our parameters by incrementally adjusting them, based on the derivative:

$$w = w - \propto \frac{\partial J(w,b)}{\partial w}$$

$$b = b - \propto \frac{\partial J(w,b)}{\partial b}$$

- Where, ∝, is size of the adjustment, normally called the **learning rate**.

Building a Cost Function

- Ok, so what we need is a differentiable, "well behaved" (*i.e.* convex) function to represent our cost

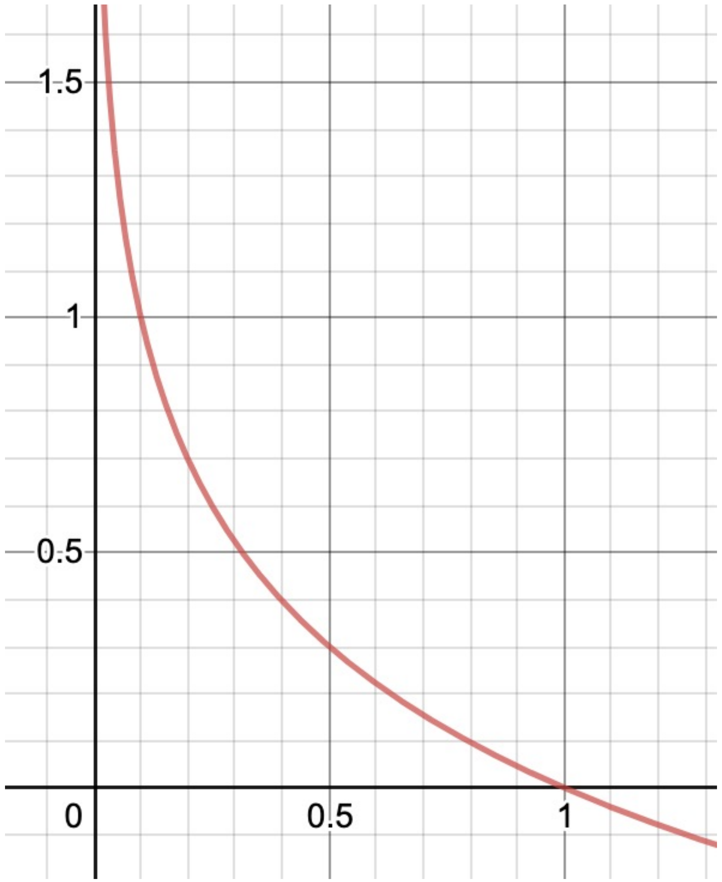
- Let's step back and take stock:

    - we have a number of examples, *m*,
    - each example is X=>y mapping,
    - X is a vector of dimension, *n:* X = $[x_1, x_2, \ldots, x_n]$
    - y is one of two values {0,1}

- As reasonable assumption is that for each of the *m* examples we want $a(X, W, b)$ to be as close to y as possible

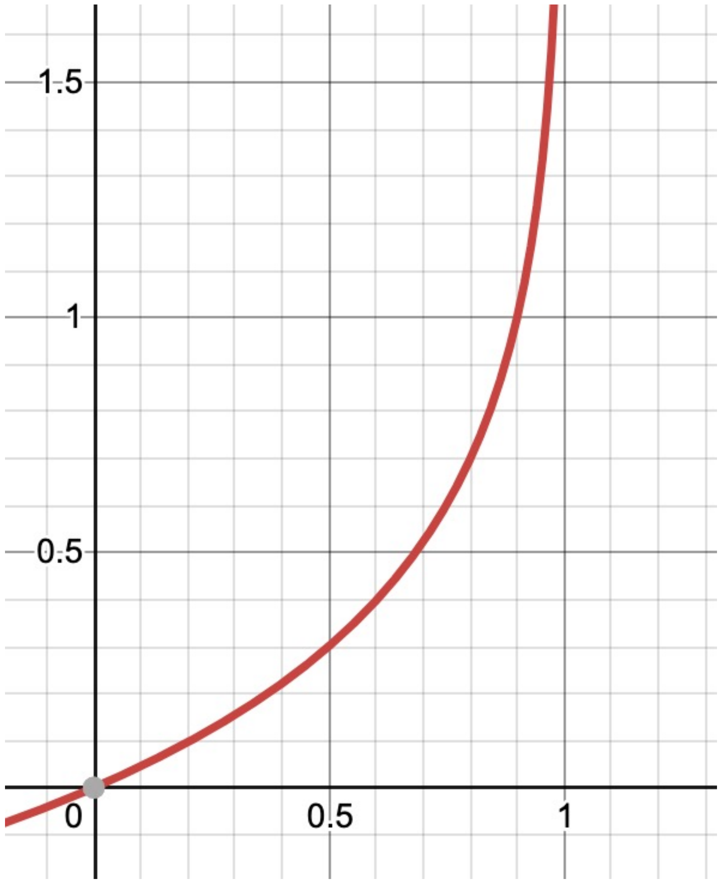
Consider the Log Function

$-\log(a)$



0 when a=1, but large when a => 0

$-\log(1-a)$



0 when a=0, but large when a => 1

- Notice also the simple and well-defined derivative:

$$\frac{d}{da} log(a) = \frac{1}{a}$$

Brad Quinton, Scott Chin

Using the Log function create a Loss Function

- Consider the cost for a data point (this is called the **Loss**, L) using the log function:

  When y = 1: $L(a, y) = -\log(a)$
  When y = 0: $L(a, y) = -\log(1 - a)$

- Since we know that y will be strictly 0, or 1, then can easily combine these into one function:

$$L(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$

Finding the Derivative w.r.t. Each Parameter

- We can calculate the partial derivatives as follows:

$$L(a, y) = -(y \log a + (1 - y) \log(1 - a)) \Rightarrow \frac{\partial L}{\partial a} = -\frac{y}{a} + \frac{1 - y}{1 - a} = \frac{a - y}{a(1 - a)}$$

$$a = \sigma(z) = \frac{1}{1 + e^{-z}} \Rightarrow \frac{\partial a}{\partial z} = \sigma(z)(1 - \sigma(z)) = a(1 - a)$$

$$z = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b \Rightarrow \frac{\partial z}{\partial w_1} = x_1, \frac{\partial z}{\partial w_2} = x_2, \ldots, \frac{\partial z}{\partial b} = 1$$

- Then Using the *chain rule* we get:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w_1} = x_1(a - y)$$

Brad Quinton, Scott Chin

Finding the Derivative w.r.t. Each Parameter

- By analogy we find:

$$\frac{\partial L}{\partial w_n} = x_n(a - y) \qquad\qquad \frac{\partial L}{\partial b} = (a - y)$$

- Notice how well this worked out! For each data point, the rate of change of the Loss, *L*, is only dependent on the different between the hypothesis, *a,* and the actual value, *y,* multiplied by the value of the input, x_n.

- This will be very important later when we want to build an efficient Neural Network training algorithm.

Final Cost Function

- We can pull this together across each of the *m* data points to create the cost function, *J*:

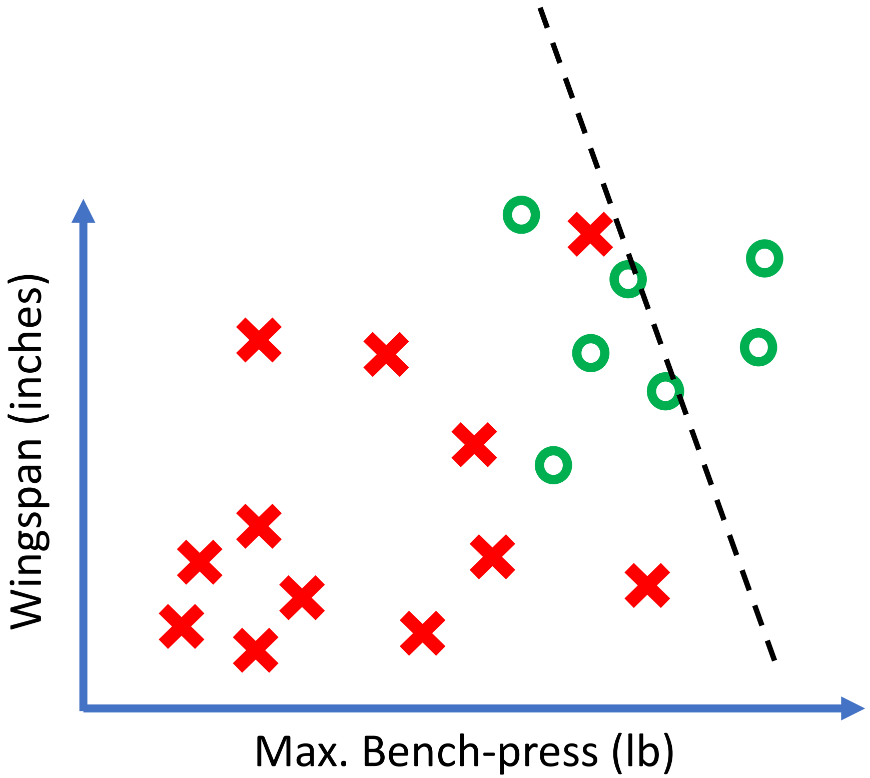
$$J = -\frac{1}{m}\left(\sum_{i=1}^{m} y^i \log\left(a^{(i)}\right) + \sum_{i=1}^{m}\left(1 - y^{(i)}\right)\log(1 - a^{(i)})\right)$$

- Therefore:

$$\frac{\partial J}{\partial w_n} = -\frac{1}{m}\sum_{i=1}^{m} x_n^{(i)}\left(a^{(i)} - y^i\right)$$

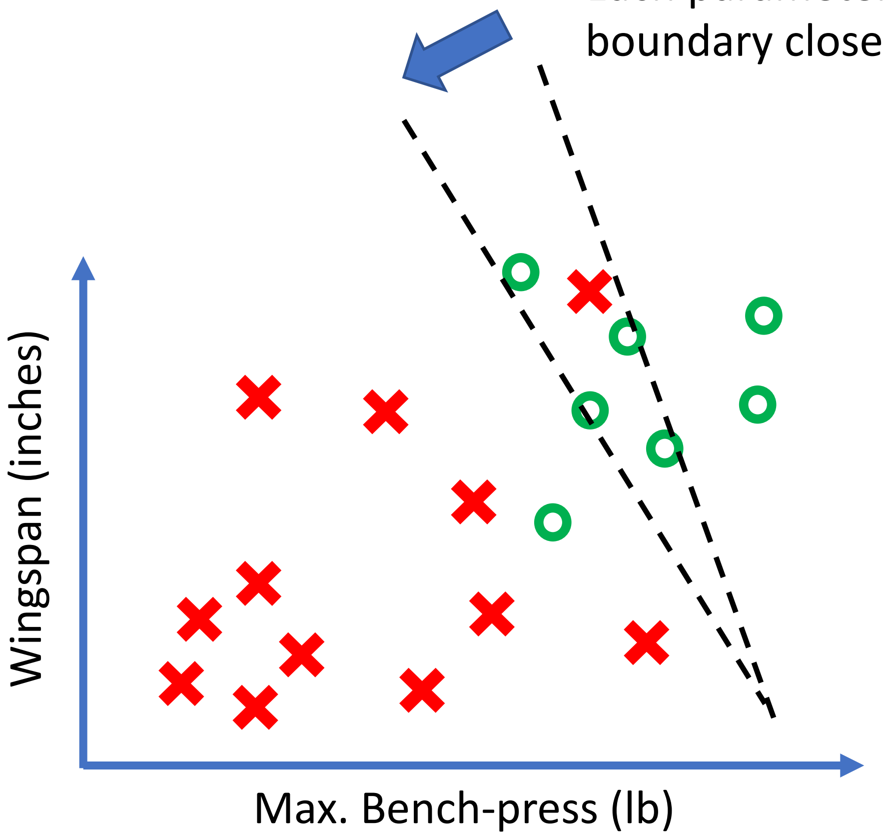
$$\frac{\partial J}{\partial b} = -\frac{1}{m}\sum_{i=1}^{m}\left(a^{(i)} - y^i\right)$$

Logistic Regression



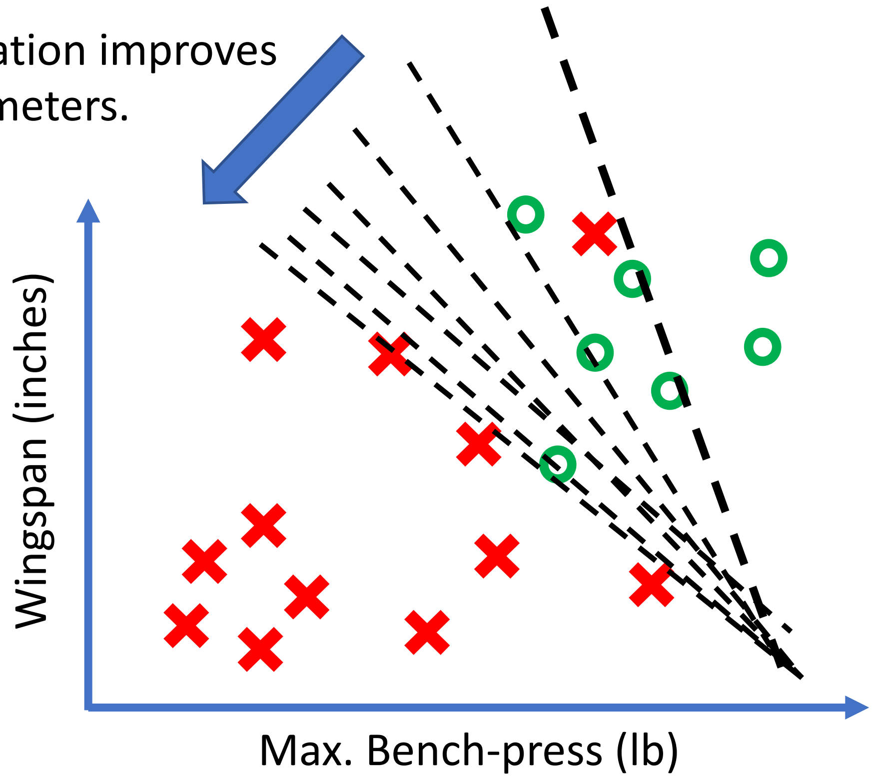
Logistic Regression

Each parameter update moves the decision boundary closer to the "low cost" point.



Logistic Regression

Each iteration improves the parameters.



Pulling it all together

1. Assume: $a = \sigma(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b)$

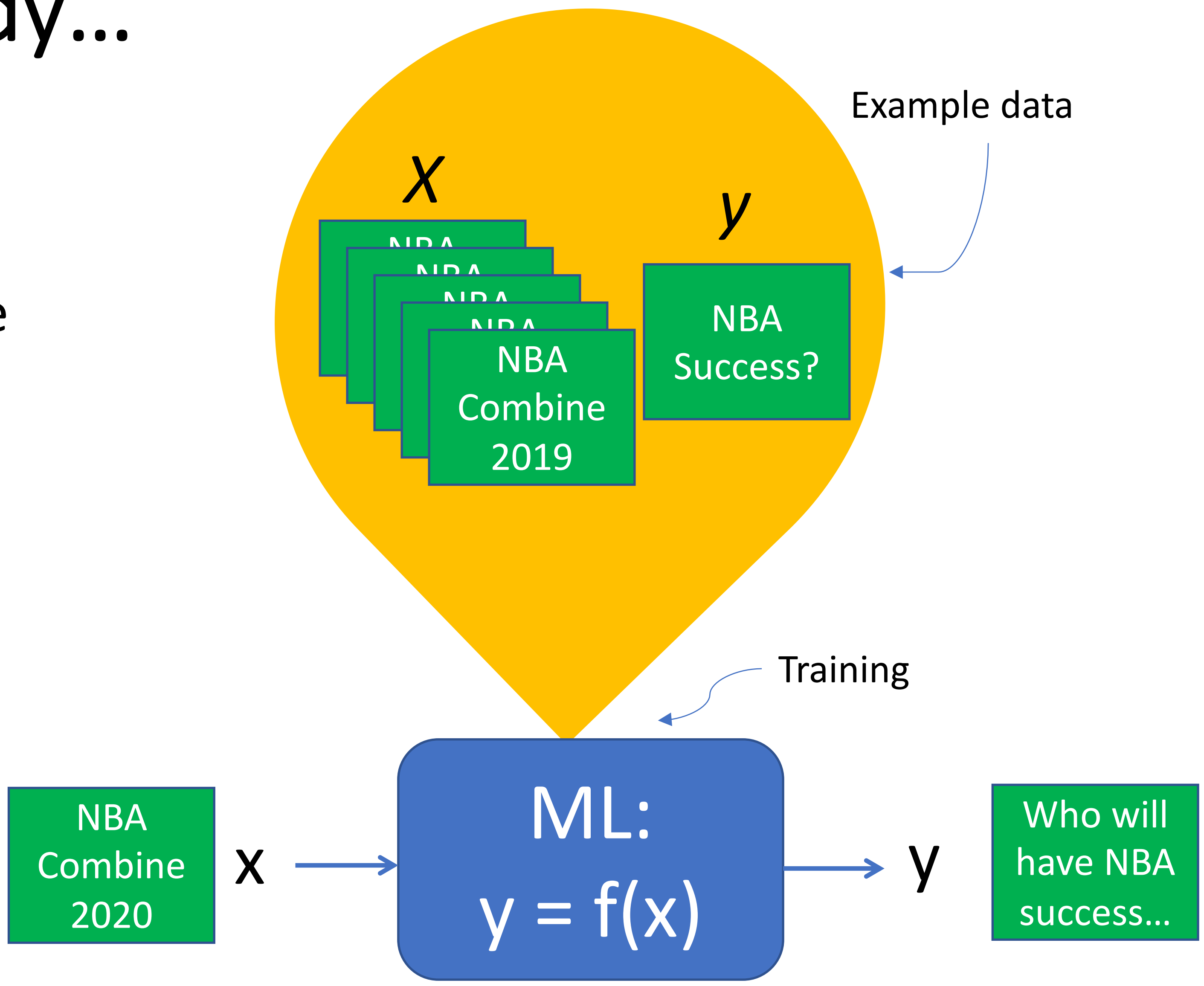
2. Initialize w_{1-n}, b to random values (or zero)

3. Repeatedly apply: $w = w - \propto \frac{\partial J(w,b)}{\partial w}$ $\quad b = b - \propto \frac{\partial J(w,b)}{\partial b}$

4. Stop when J < target error

Back to Case Study…

- Looks like we have a plan to build that AI the Raptors are looking for!
- Let's go tell them…



Case Study: Meeting #2

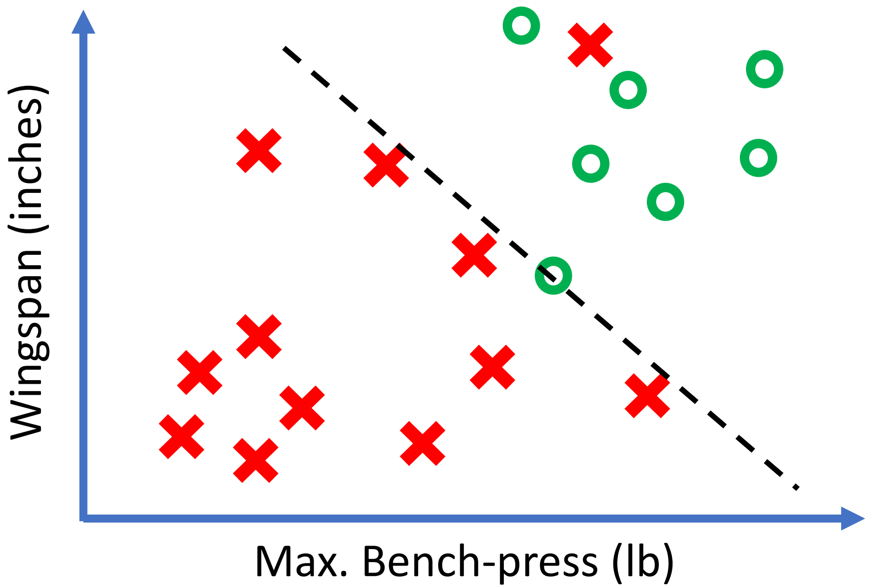
- **Day 5:** Second meeting with the coaching staff:

You: *"I've figured it all out! We can use Machine Learning to build the AI. It will predict who will and won't play in the NBA using the Draft Combine"*

Coaches: *"Wow! That's great. So it will be 100% accurate? This AI stuff sure is great."*

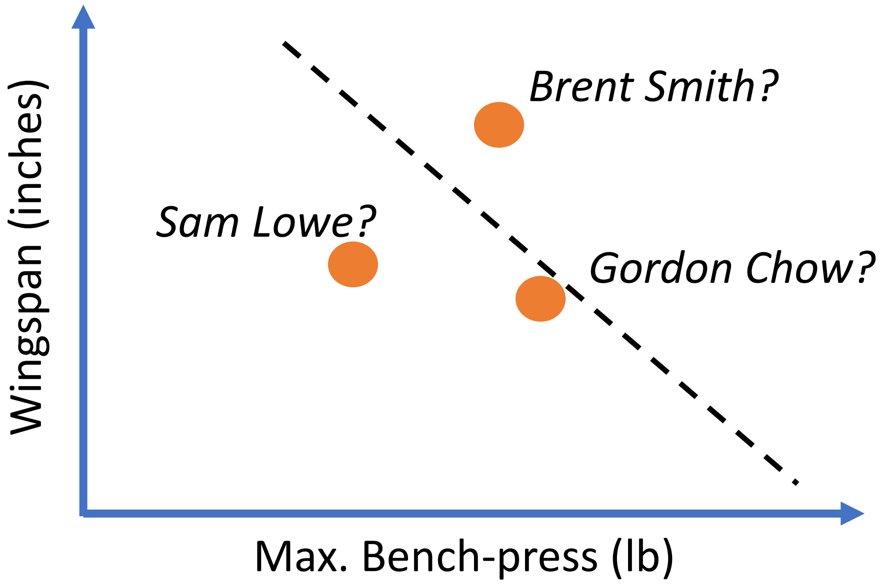
You: *"Ah…uh…well, I guess there will be 'some' error. Umm…Let me get back to you."*

Don't forget the goal is Prediction



- This sure looks like a nice fit but remember we all ready know all the answers here!
- And, in fact, the fit is already not perfect…

Don't forget the goal is <u>Prediction</u>



- In ML it does NOT matter how well we fit the training data, it ONLY matters if it works for **<u>NEW data</u>**!

How *do* we know if the AI will work?

- Oh, and how do we implement it?

- Stay tuned for the next lecture, where we will look and implementation and evaluation of logistic regression….