

Implementation and Evaluation of Logistic Regression

Deep Learning

[Brad Quinton](#), [Scott Chin](#)

Raptors Case Study: Recap

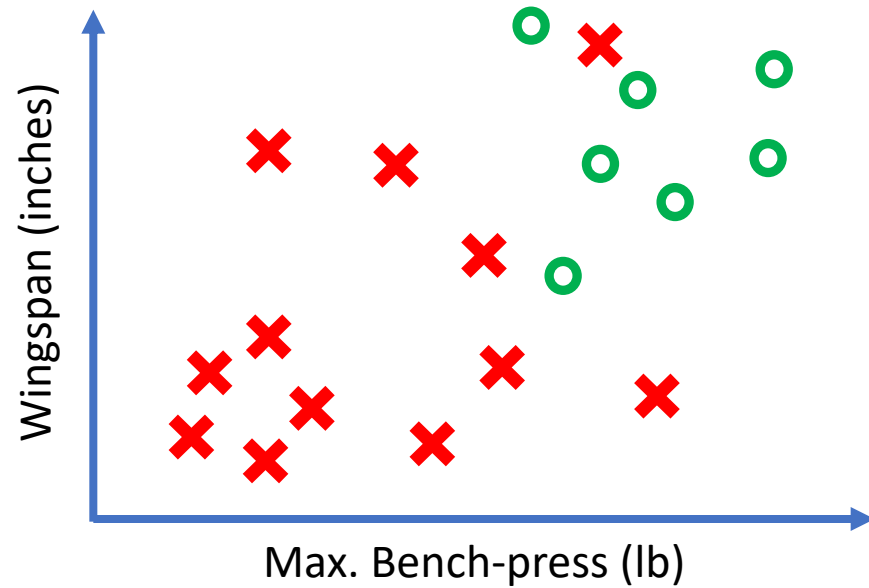


- After landing a "dream" internship with the Toronto Raptors basketball team, you have been asked to build an AI to help select NBA draft picks
- You have quickly studied up on Logistic Regression as a ML technique that seems to be a good candidate for this particular AI
- But after a second meeting with the coaching staff you realize:
 1. You don't know how accurate your AI will be.
 2. You don't really know how to implement it (at least efficiently!)

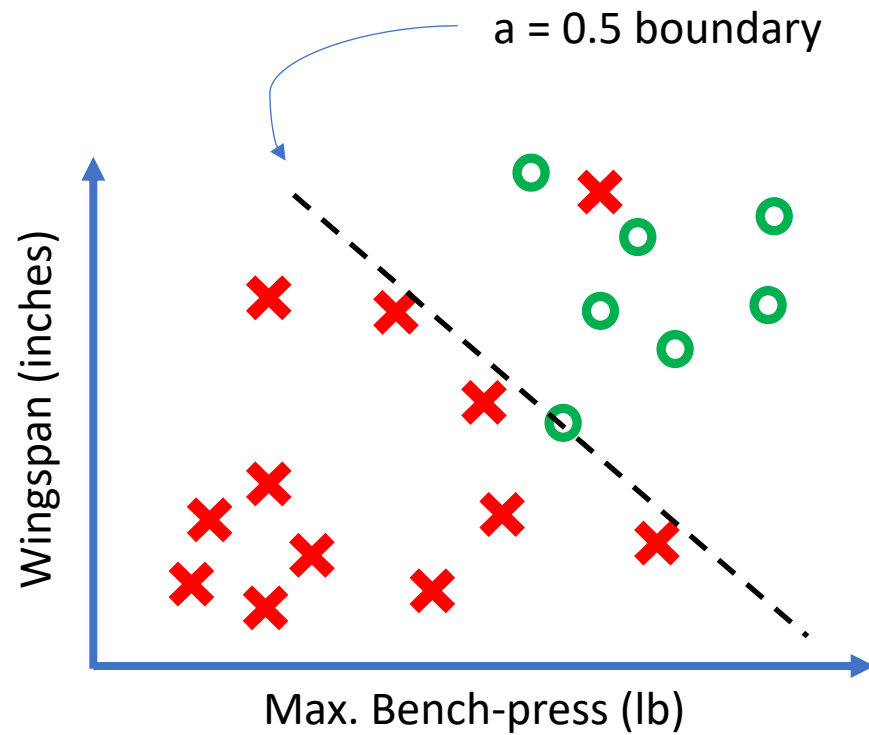
Accuracy of Logistic Regression based ML

- In machine learning (ML) we are using a set of example data to build an AI model of expected relationships (*i.e.* an approximate function)
- In the end, it is the performance of the model on **NEW DATA** that really matters
- That's the “intelligence” part of Artificial Intelligence (*e.g.* can the AI do something that we consider useful?)

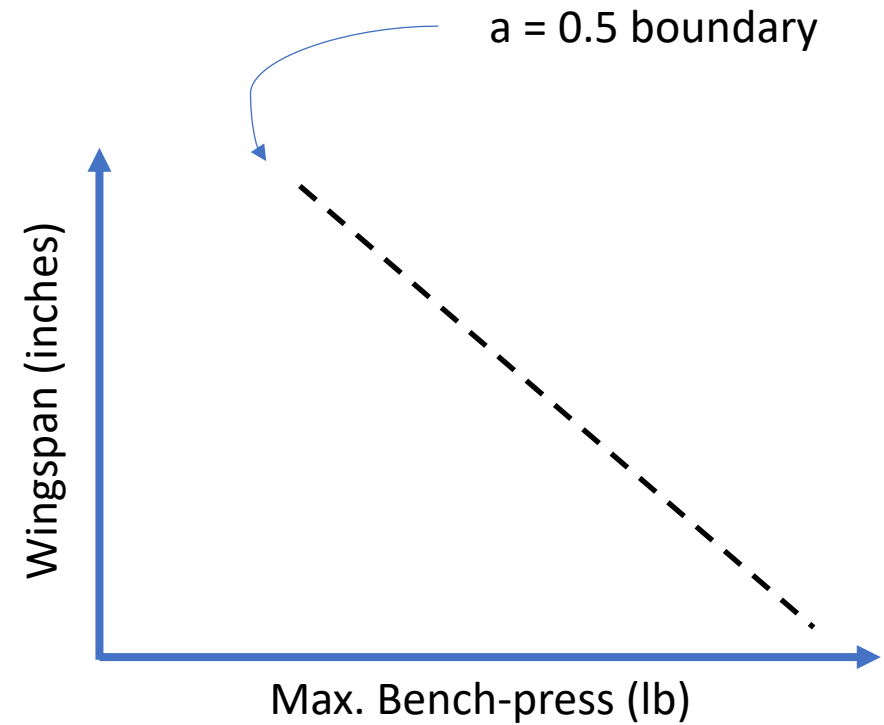
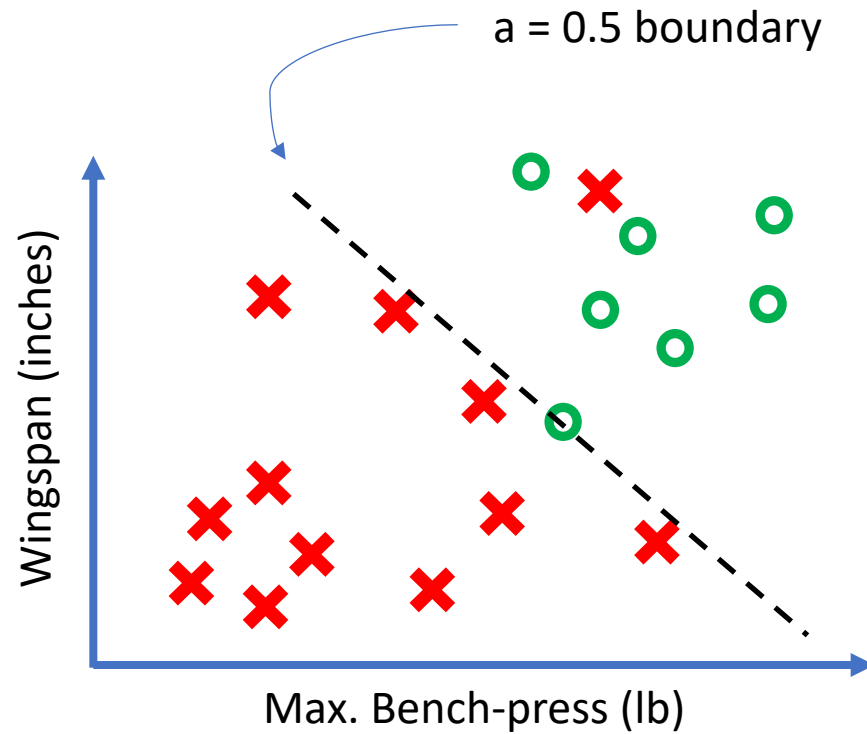
Look at the data again...



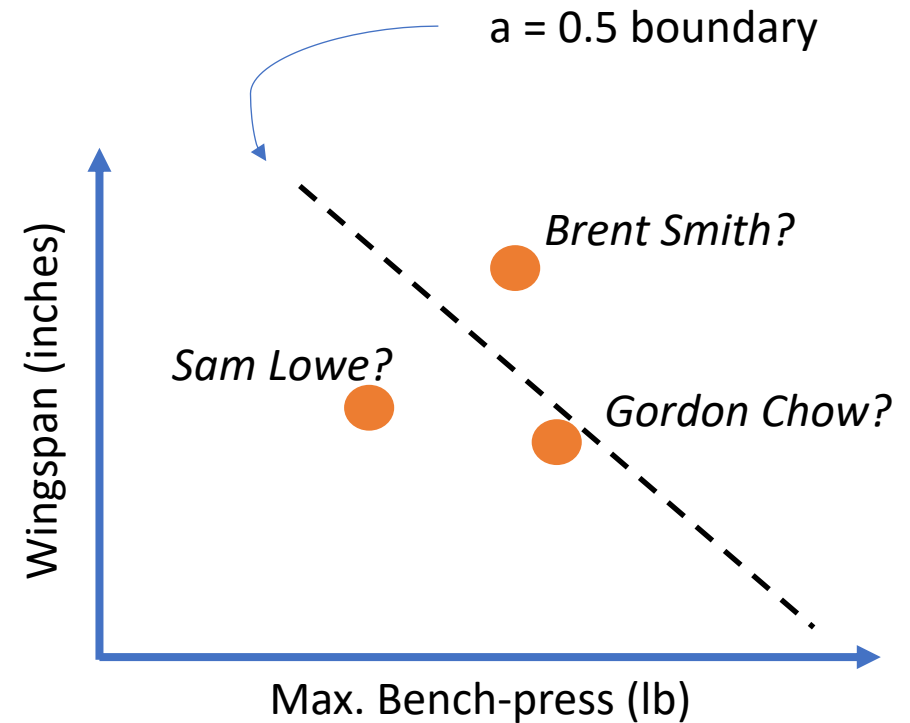
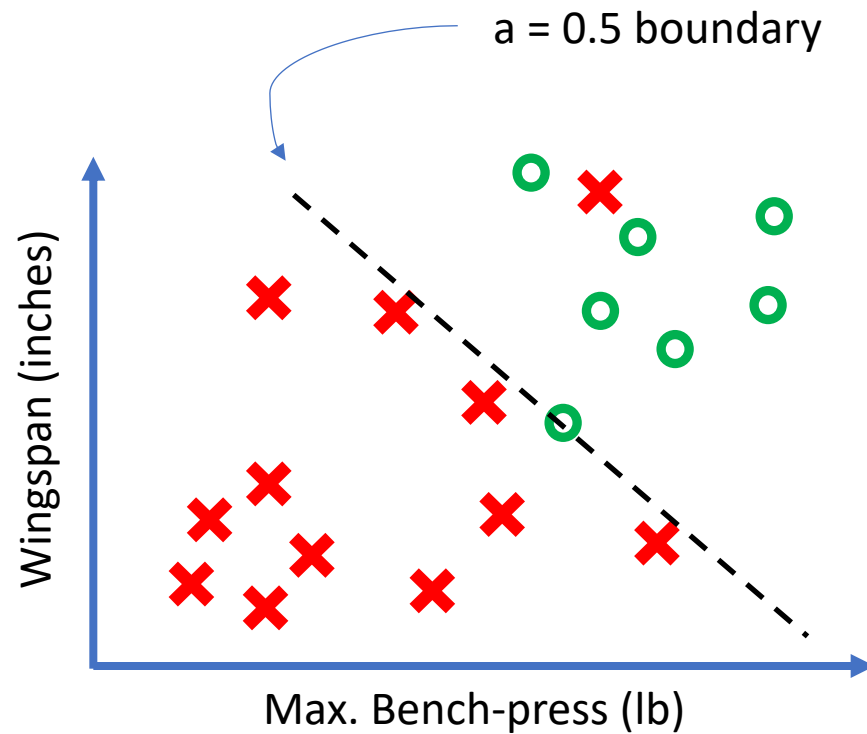
Look at the data again...



Look at the data again...



Look at the data again...



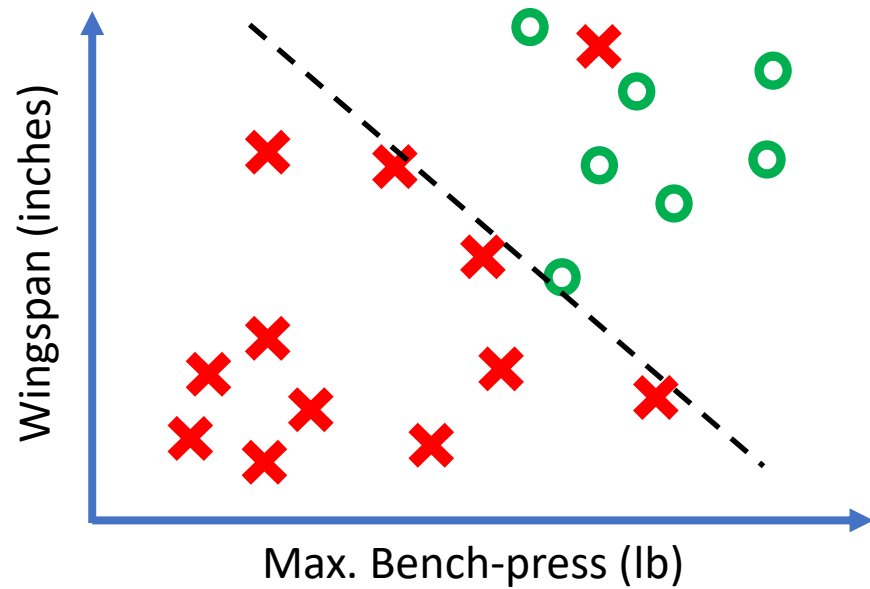
Sources of Inaccuracy in ML

- Let's first work through some intuition and then we will lay out some detailed measurements and prescriptions for various scenarios.
- Let's think. What are the sources of inaccuracy in the final prediction?:
 1. The AI model does not match the underlying nature of the data (in this case, for instance, maybe the data is not **linearly separable**.)
 2. Our learning algorithm did not find the "best" set of parameters for our model.
 3. The example data is **not representative** of the new data.

Inaccuracy from the Underlying AI Model

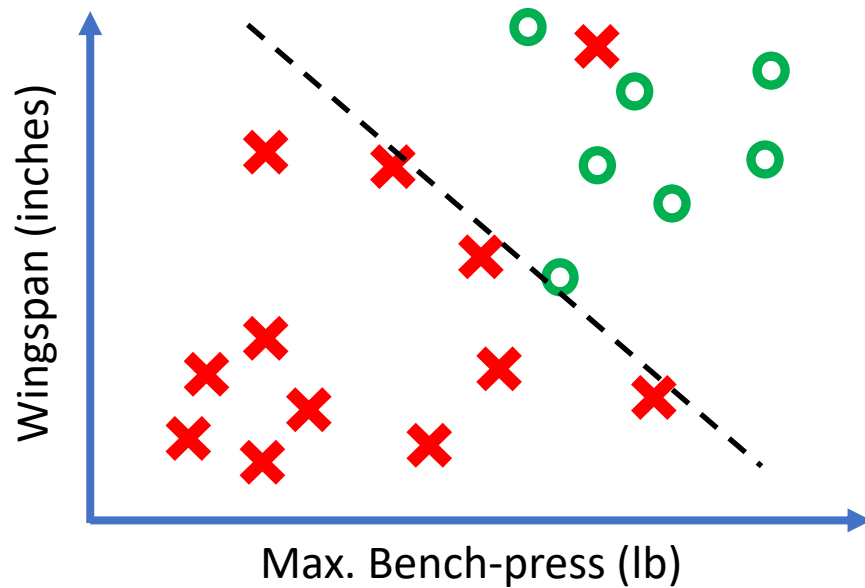
- It is possible that we have selected an underlying architecture for our ML that is not a good match for the problem at hand
- Remember, for Logistic Regression we made the *assumption* that we can predict based on the weighted sum of the input data
- For some problems, this is a good assumption, but for others it is not...

Consider the Linear Assumption

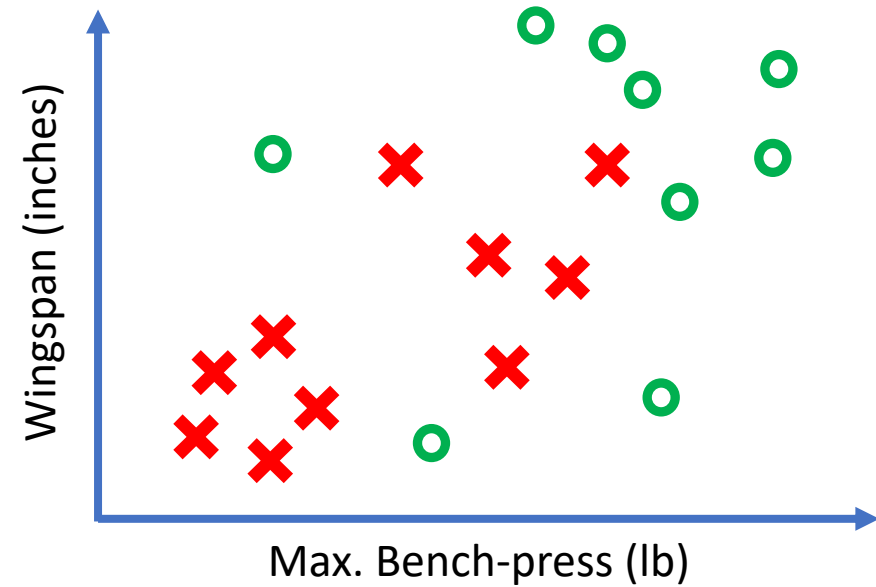


It is possible that the data is linearly separable....

Consider the Linear Assumption

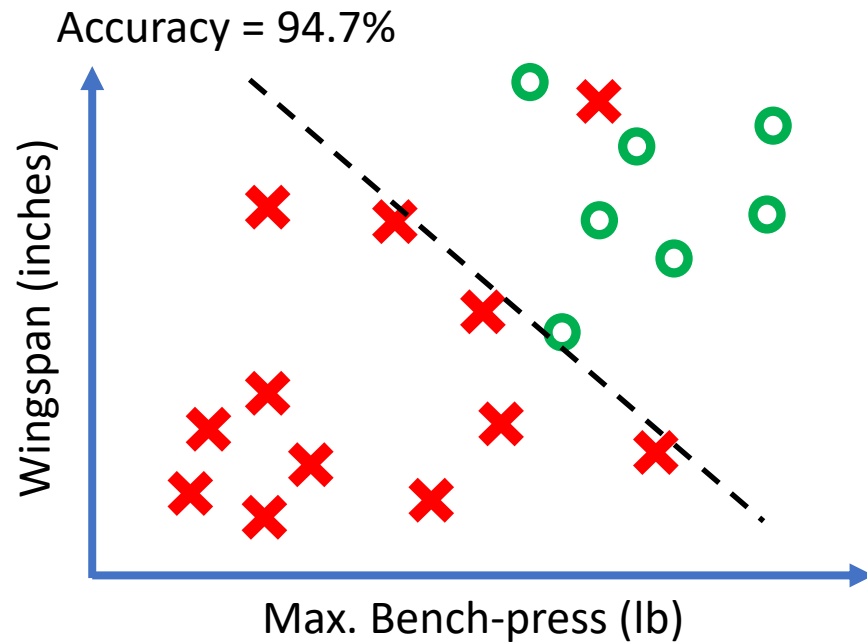


It is possible that the data is linearly separable....

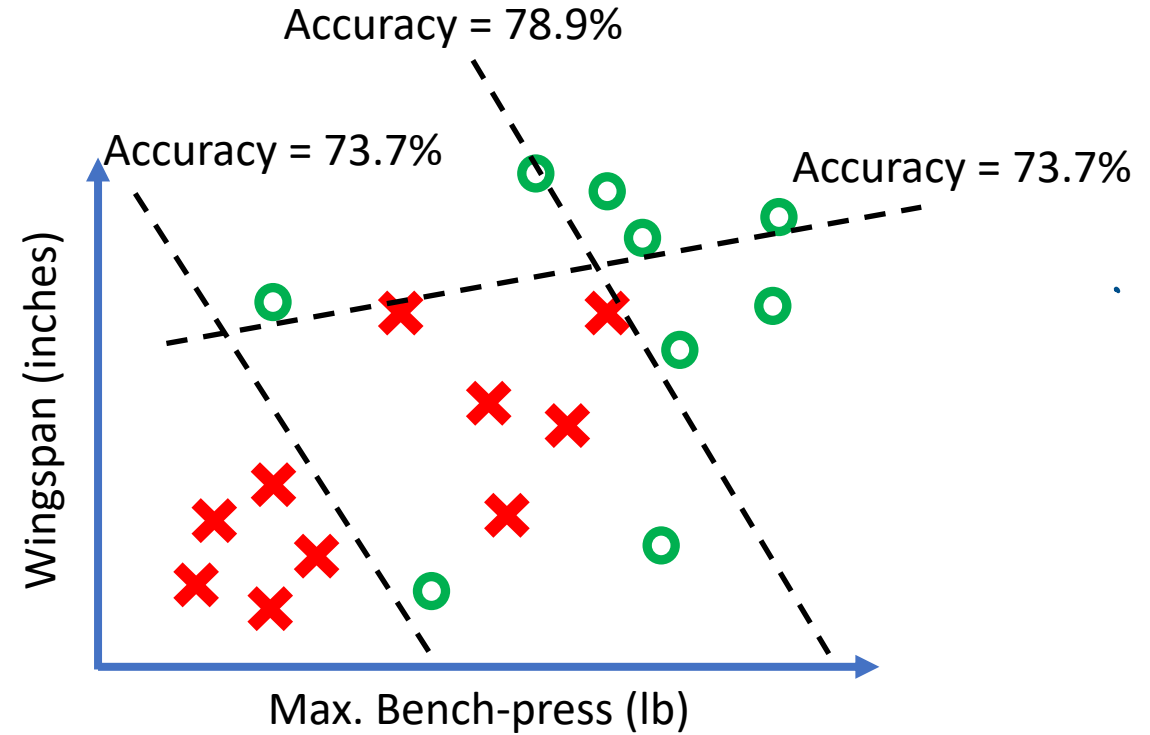


But, there is no guarantee!

Consider Linear Assumption

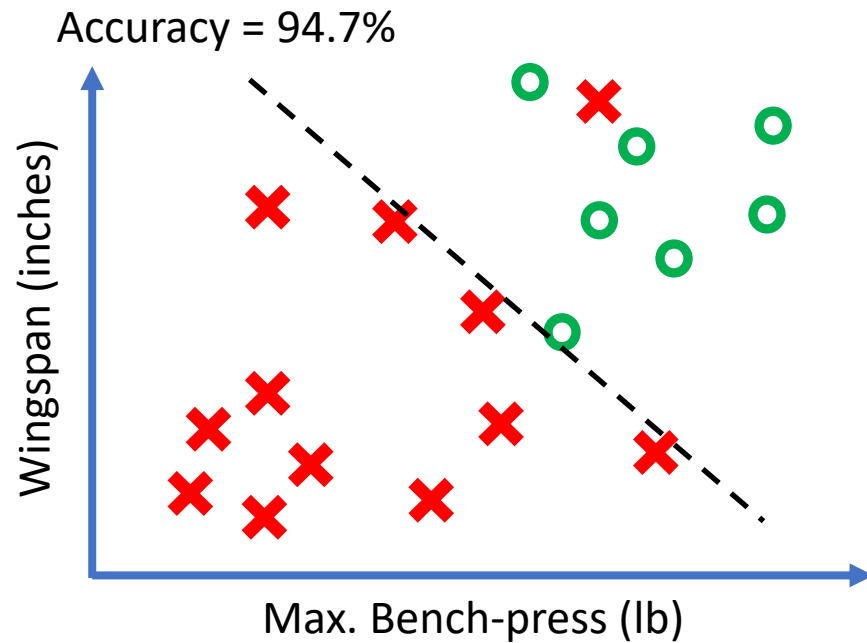


It is possible that the data is linearly separable....

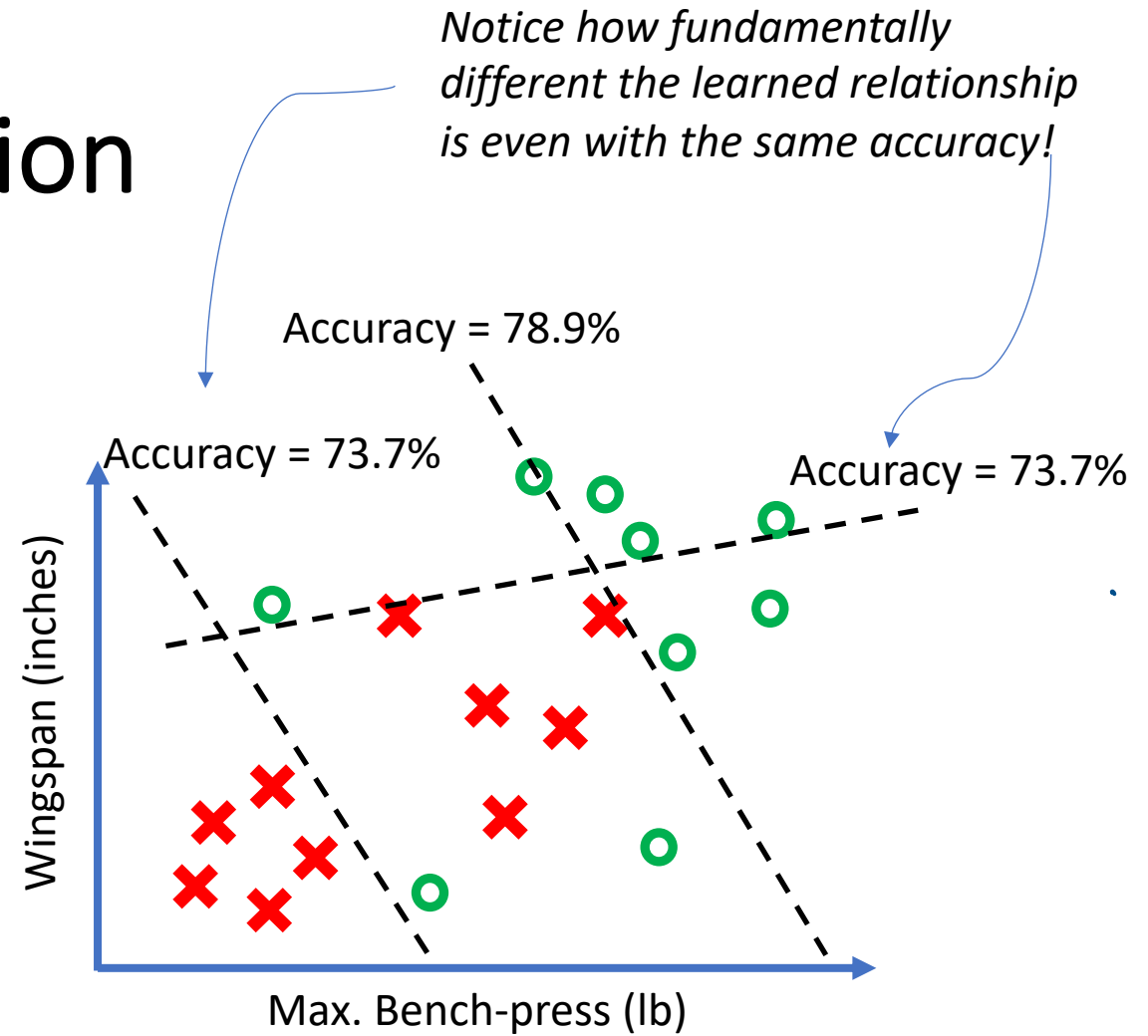


But, there is no guarantee!

Consider Linear Assumption



It is possible that the data is linearly separable....

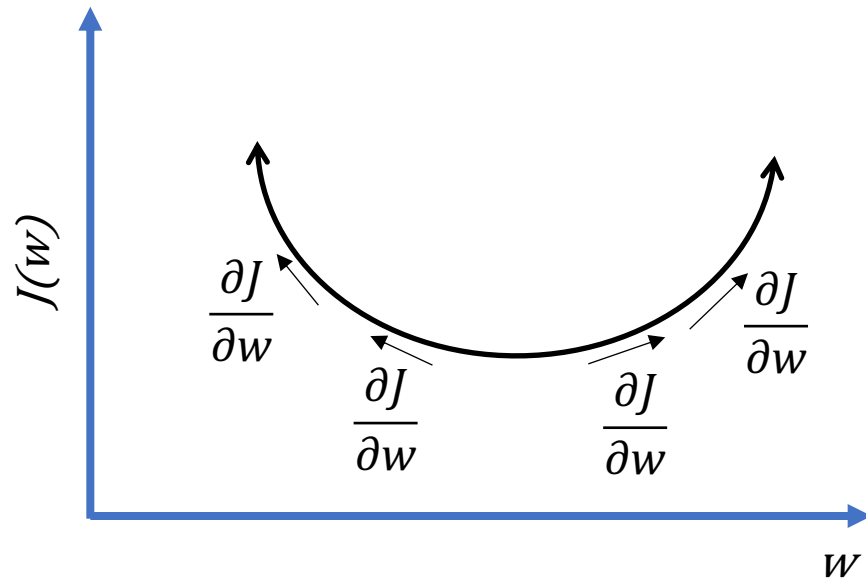


But, there is no guarantee!

Failure to Find Best Parameters

- Even when the underlying model is a good fit for the data, we still have to learn the parameters
- The accuracy of our prediction will be directly impacted by the ability to get to “good” parameters
- For logistic regression there are two ‘tuneable’ components in our algorithm: **learning rate**, α , and the **number of iterations** (in ML these are called **hyperparameters**)

Impact of Learning Rate and # of Iterations

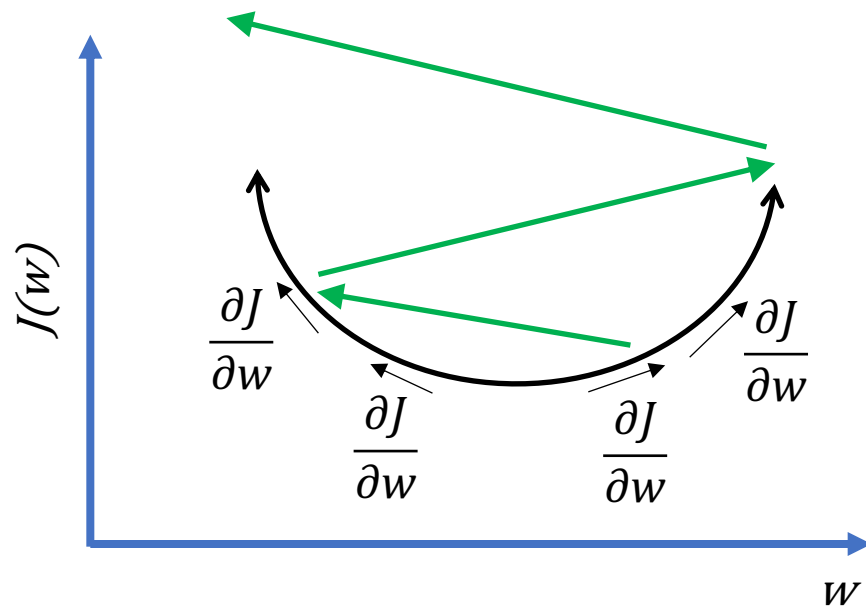


$$w = w - \alpha \frac{\partial J(w, b)}{\partial w}$$

```
for k = 1:num_iterations  
    w = w - learning_rate * dw
```

Impact of Learning Rate and # of Iterations

1. Learning rate is too large.



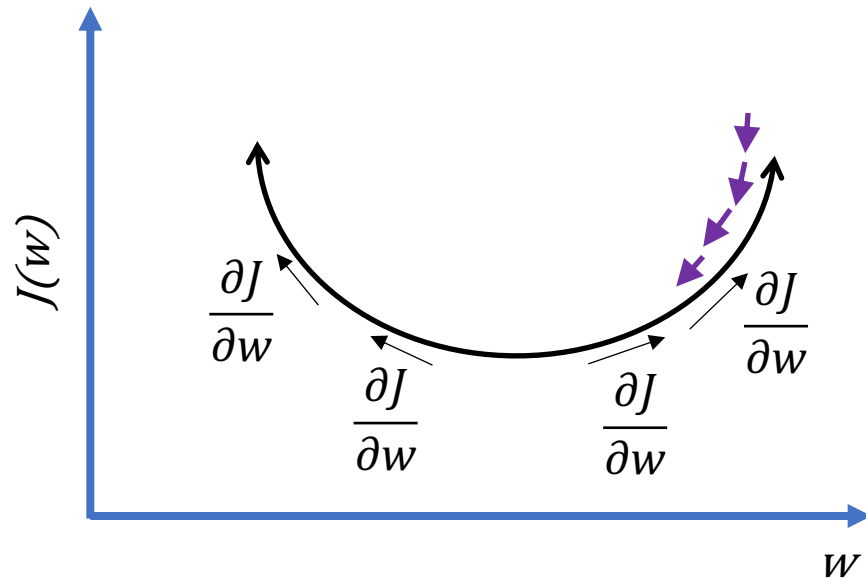
$$w = w - \alpha \frac{\partial J(w, b)}{\partial w}$$

```
for k = 1:num_iterations
    w = w - learning_rate * dw
```

Result: Final parameters are **WORSE** than random!

Impact of Learning Rate and # of Iterations

2. Learning rate is too small.



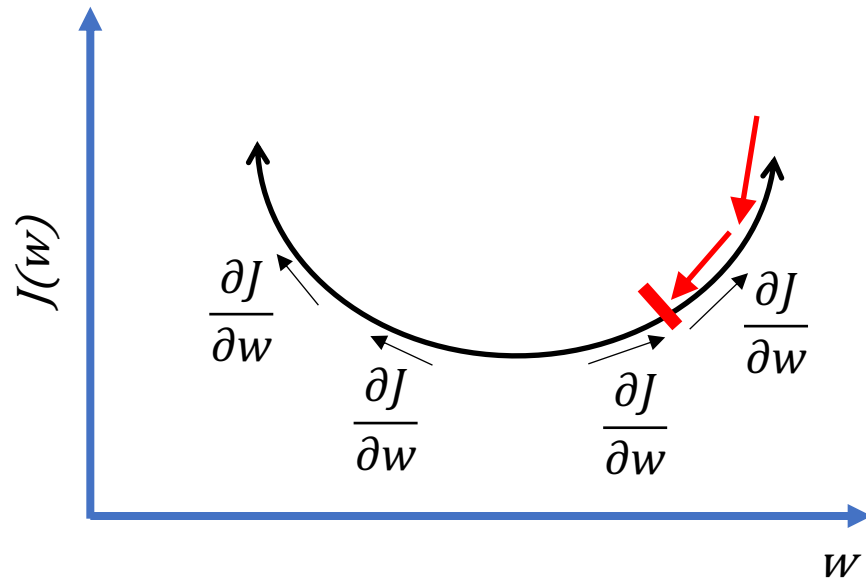
$$w = w - \alpha \frac{\partial J(w, b)}{\partial w}$$

```
for k = 1:num_iterations
    w = w - learning_rate * dw
```

Result: Final parameters are better than random, but not optimal.

Impact of Learning Rate and # of Iterations

3. Number of iterations is too small



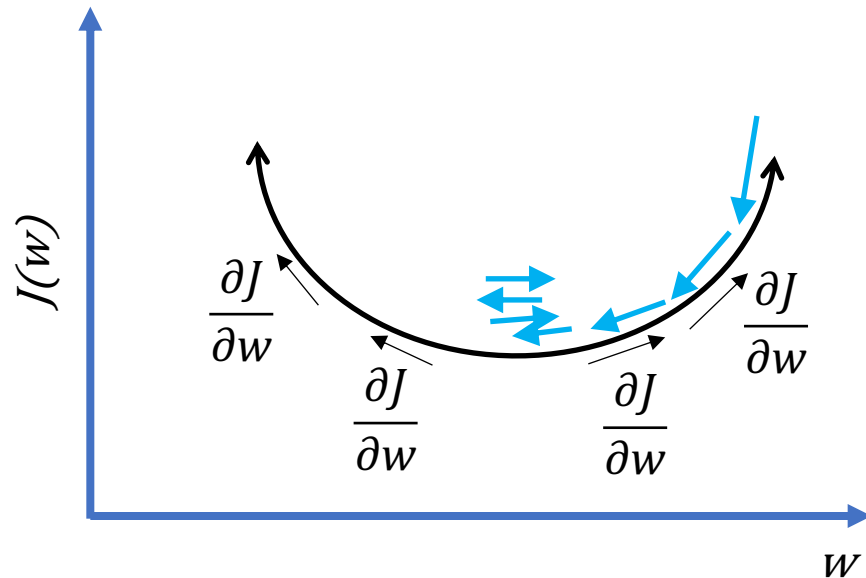
$$w = w - \alpha \frac{\partial J(w, b)}{\partial w}$$

```
for k = 1:num_iterations  
    w = w - learning_rate * dw
```

Result: Final parameters are better than random, but not optimal.

Impact of Learning Rate and # of Iterations

4. Number of iterations is too large



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

```
for k = 1:num_iterations
    w = w - learning_rate * dw
```

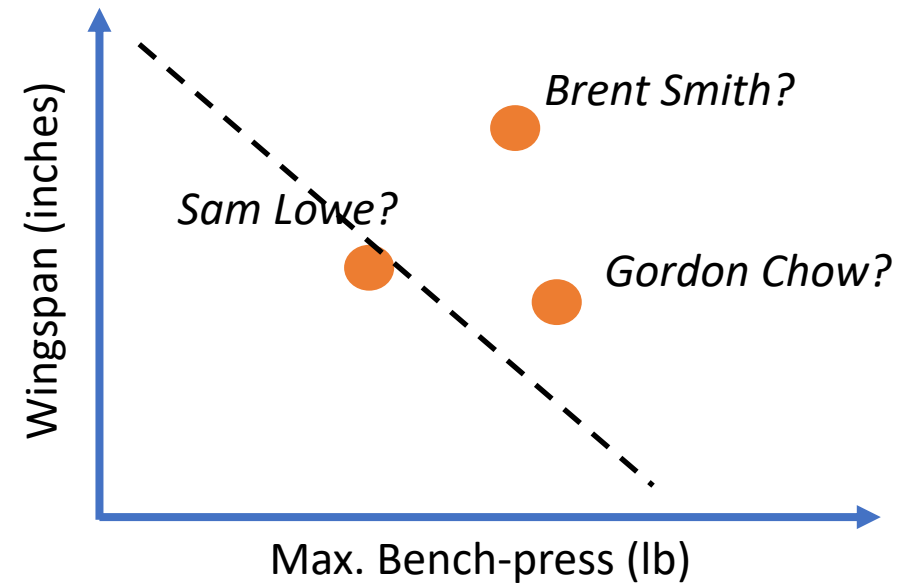
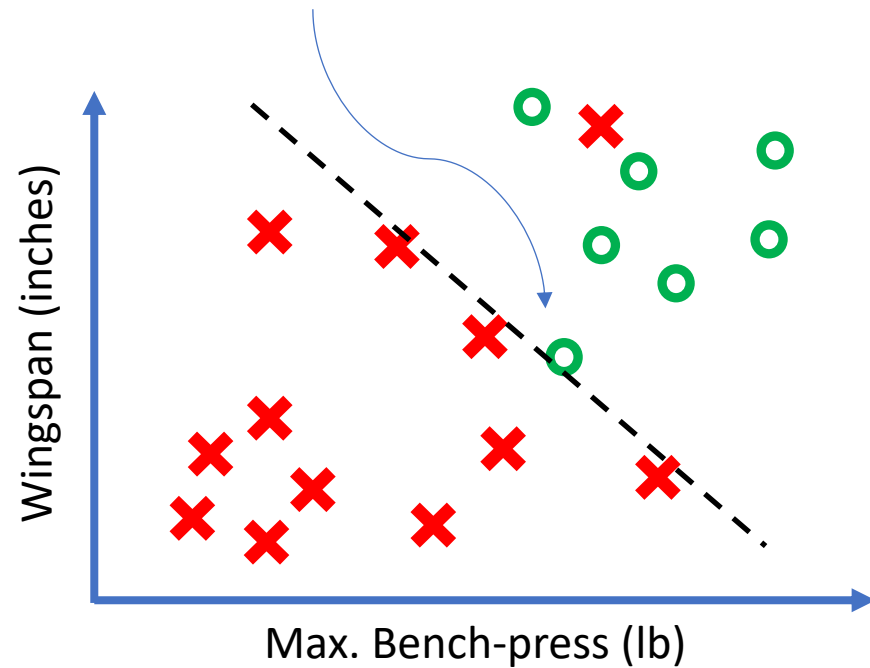
Result: As long as the learning rate is small enough, this only costs CPU cycles.

Example data is not representative of the data

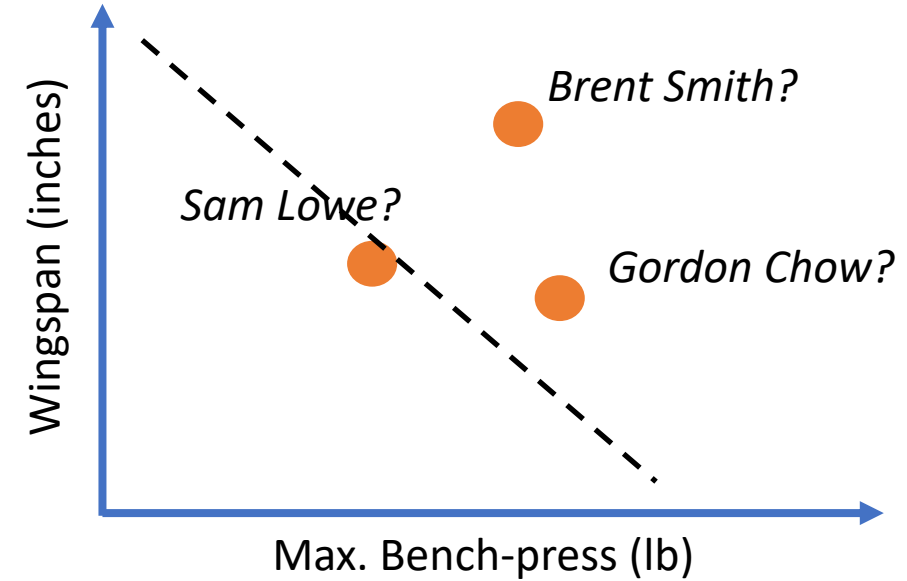
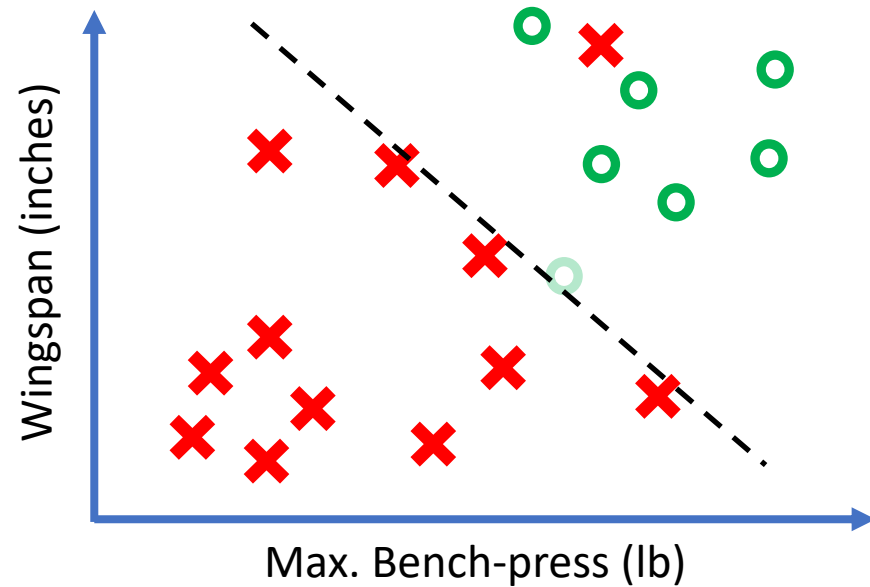
- There are a number of reasons:
 1. Not enough data to represent function.
 2. The data is noisy.
 3. The underlying behavior is not deterministic (at least with respect to what we want to predict)

Problem of Insufficient Data

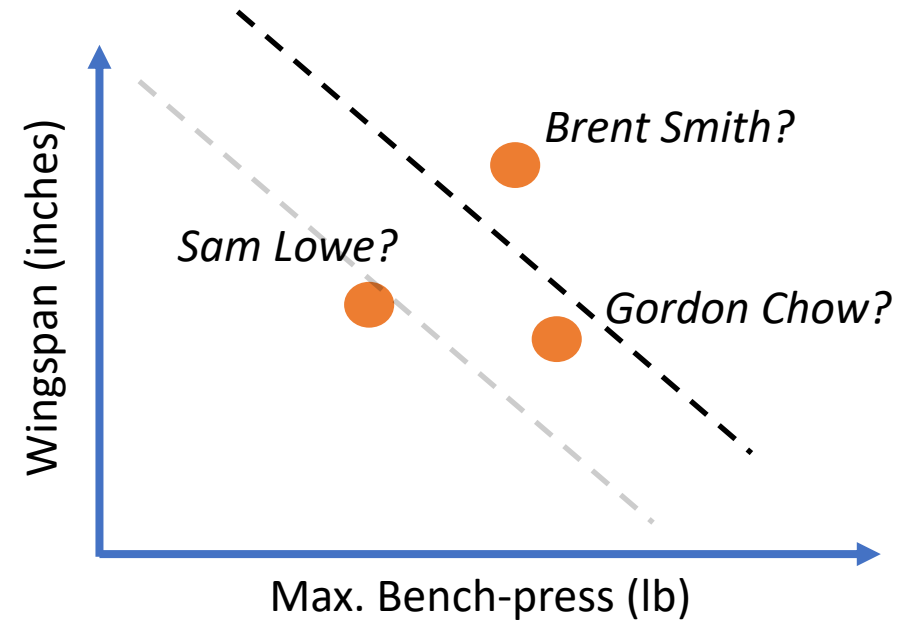
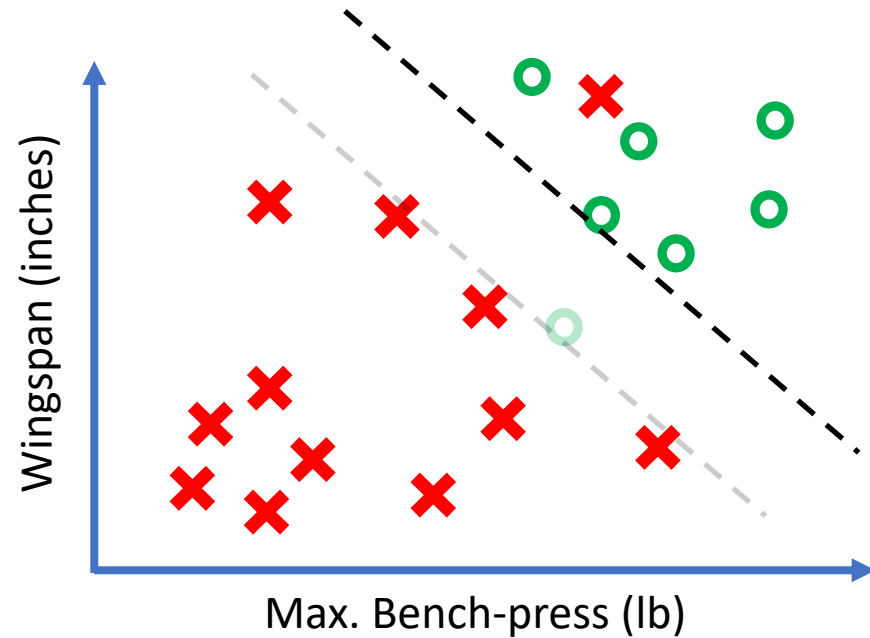
What if we failed to collect this data point?



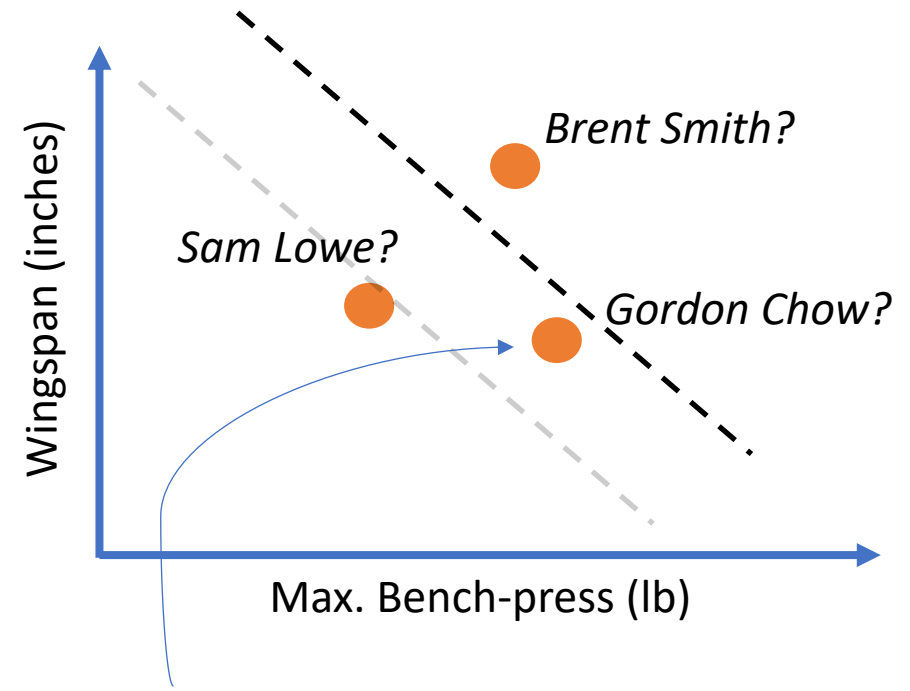
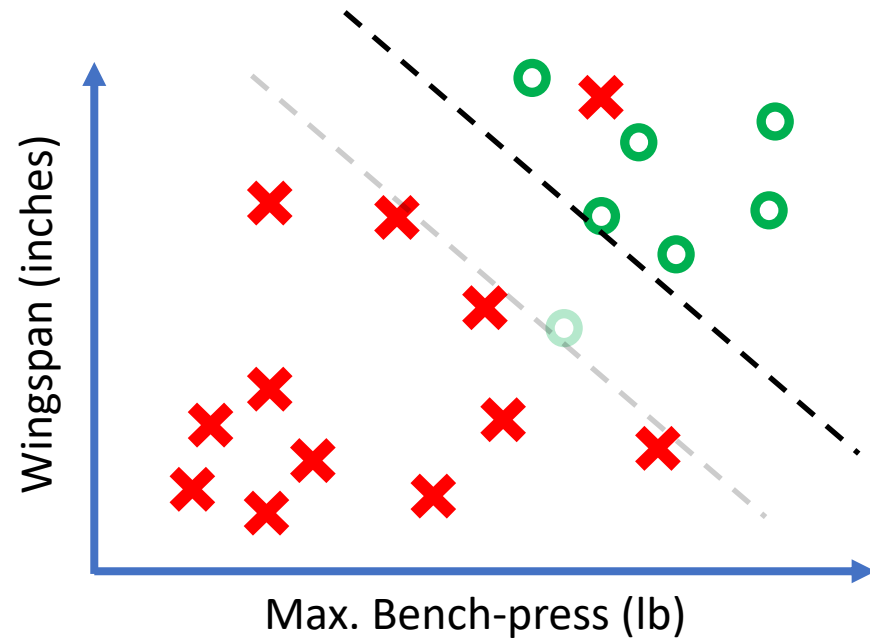
Problem of Insufficient Data



Problem of Insufficient Data

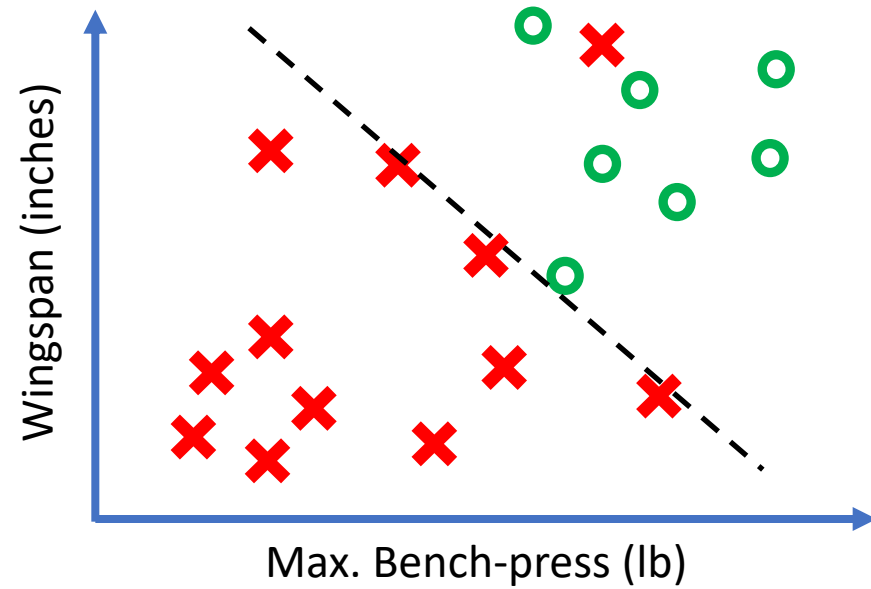


Problem of Insufficient Data

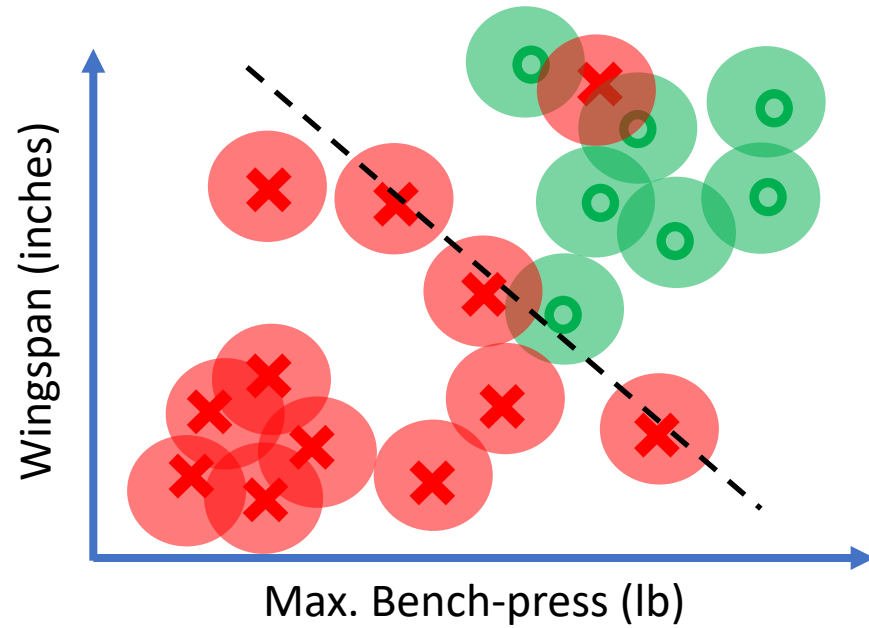


Gordon Chow's skills have not changed, but now is not headed to the NBA!?

Noisy Data

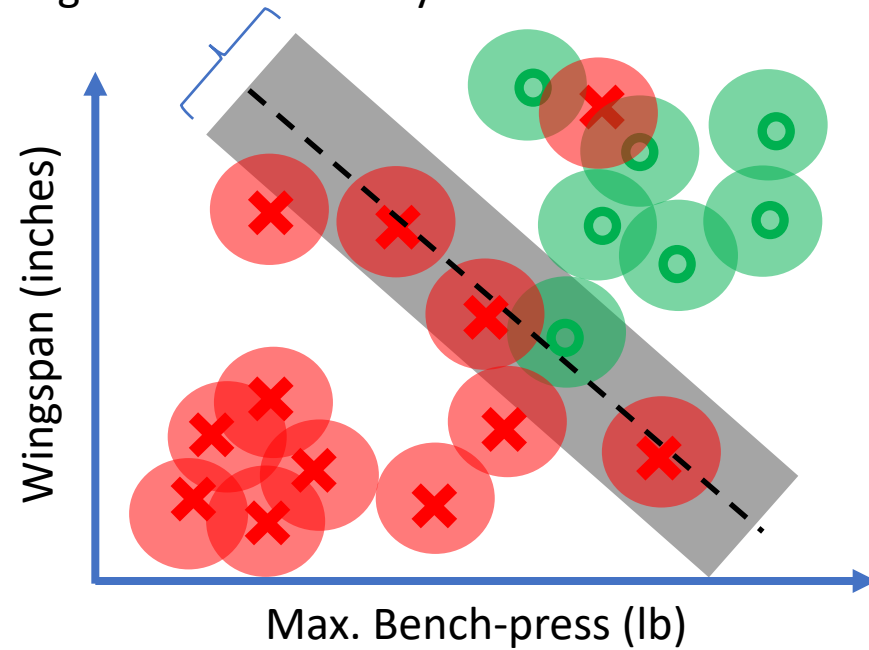


Noisy Data



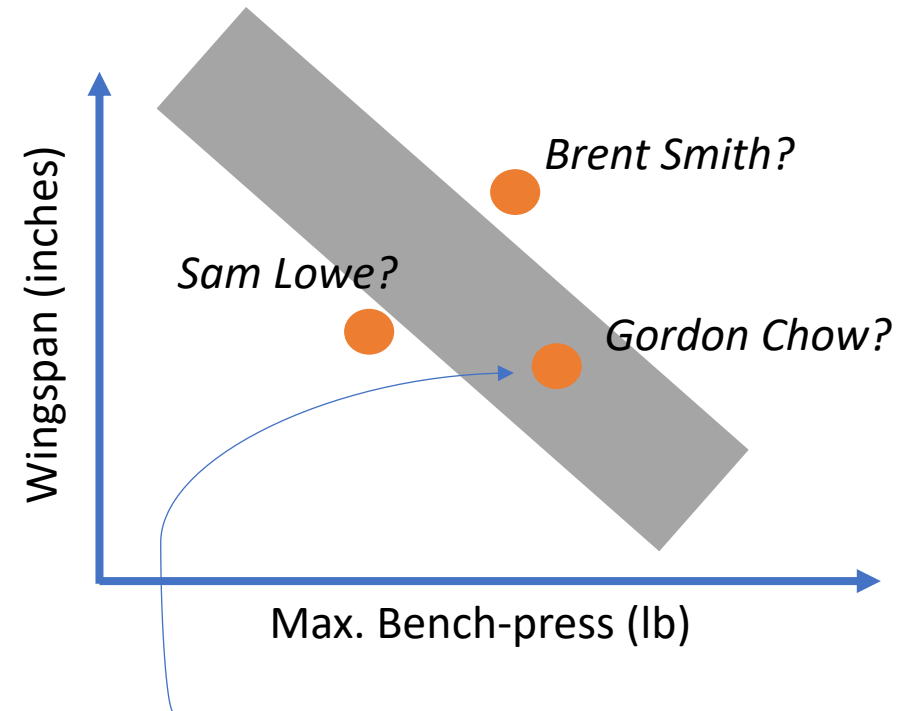
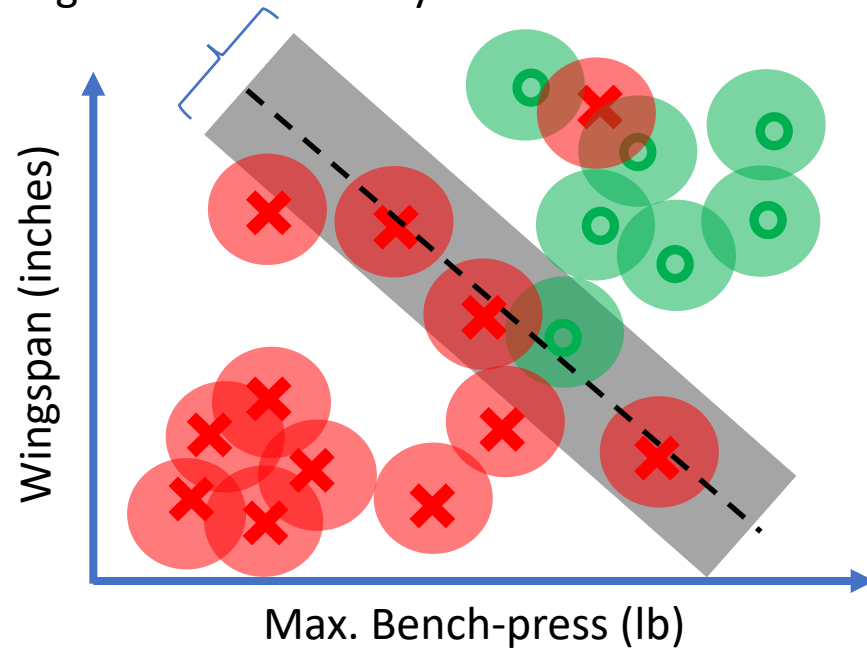
Noisy Data

Decision threshold could fall anywhere in this region depending on the which way the error falls....



Noisy Data

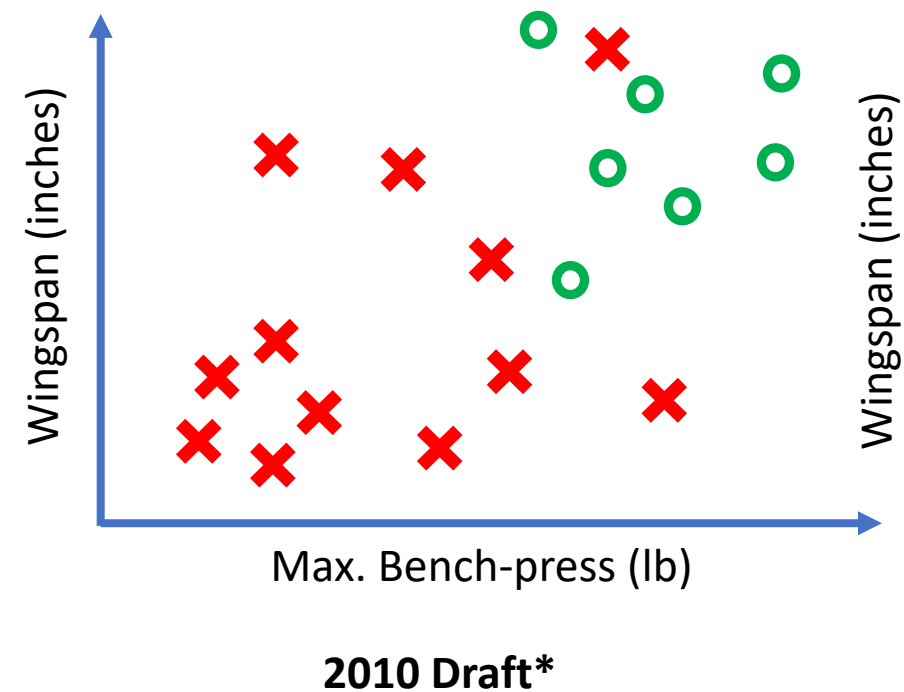
Decision threshold could fall anywhere in this region depending on the which way the error falls....



Again, what about Gordon?

Is there really a relationship between X and y?

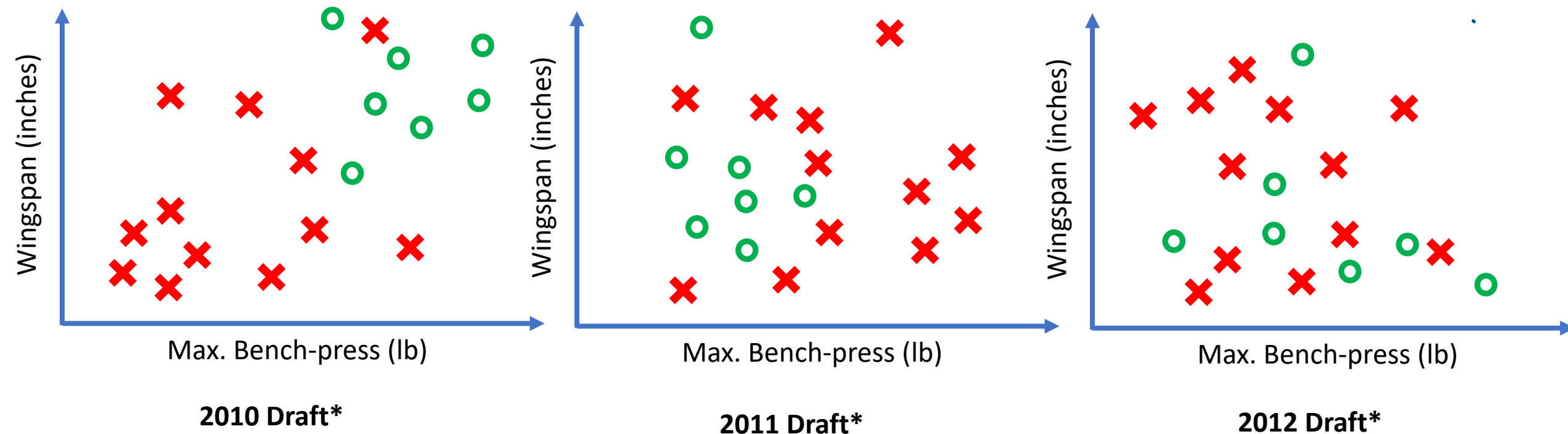
It is always possible that there is not really a deterministic relationship between the inputs and output you want to predict with your AI.



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

Is there really a relationship between X and y?

It is always possible that there is not really a deterministic relationship between the inputs and output you want to predict with your AI. For instance, in our example, maybe 'heart' is the only key to success in the NBA, in which case the Draft Combine results wouldn't matter....

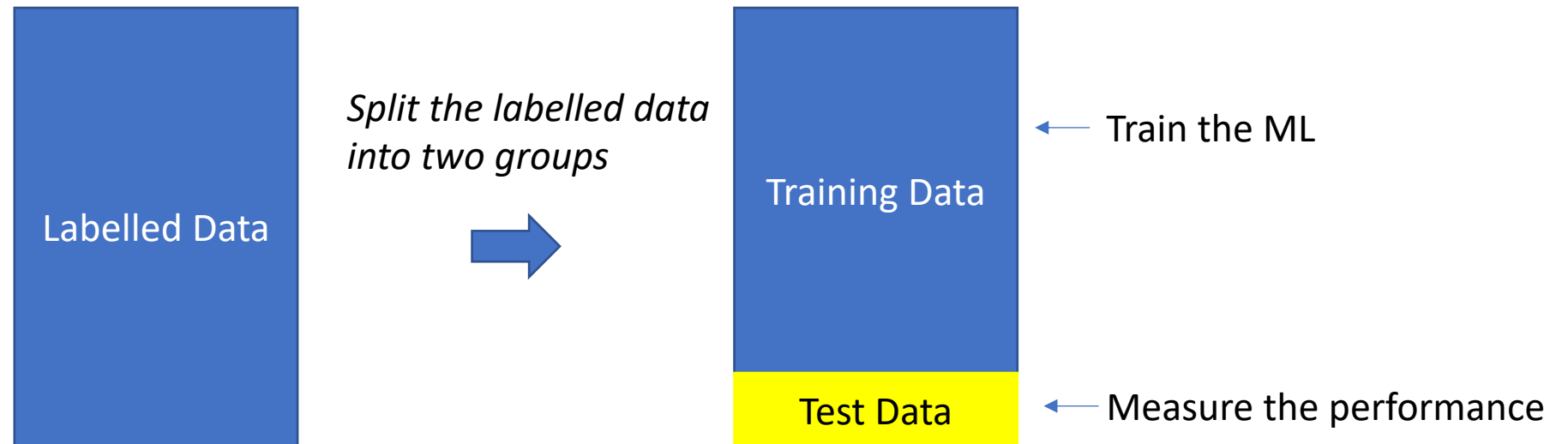


Back to Case Study

- *“Oh, no! How are we every going to go back to the Raptors coaching staff, now. It seems like this whole Machine Learning is on shaky ground!”*
- The good news is that if we stay disciplined with our approach and are careful to analyze and understand our results we can build a valuable AI and anticipate its accuracy
- Let’s find out how...

Build a Test Data Set

- The MOST important thing you can do to make sure you understand your ML is to take some of your data and “put it off to the side”
- Don’t forget our labelled data has the right answer! It is very valuable.

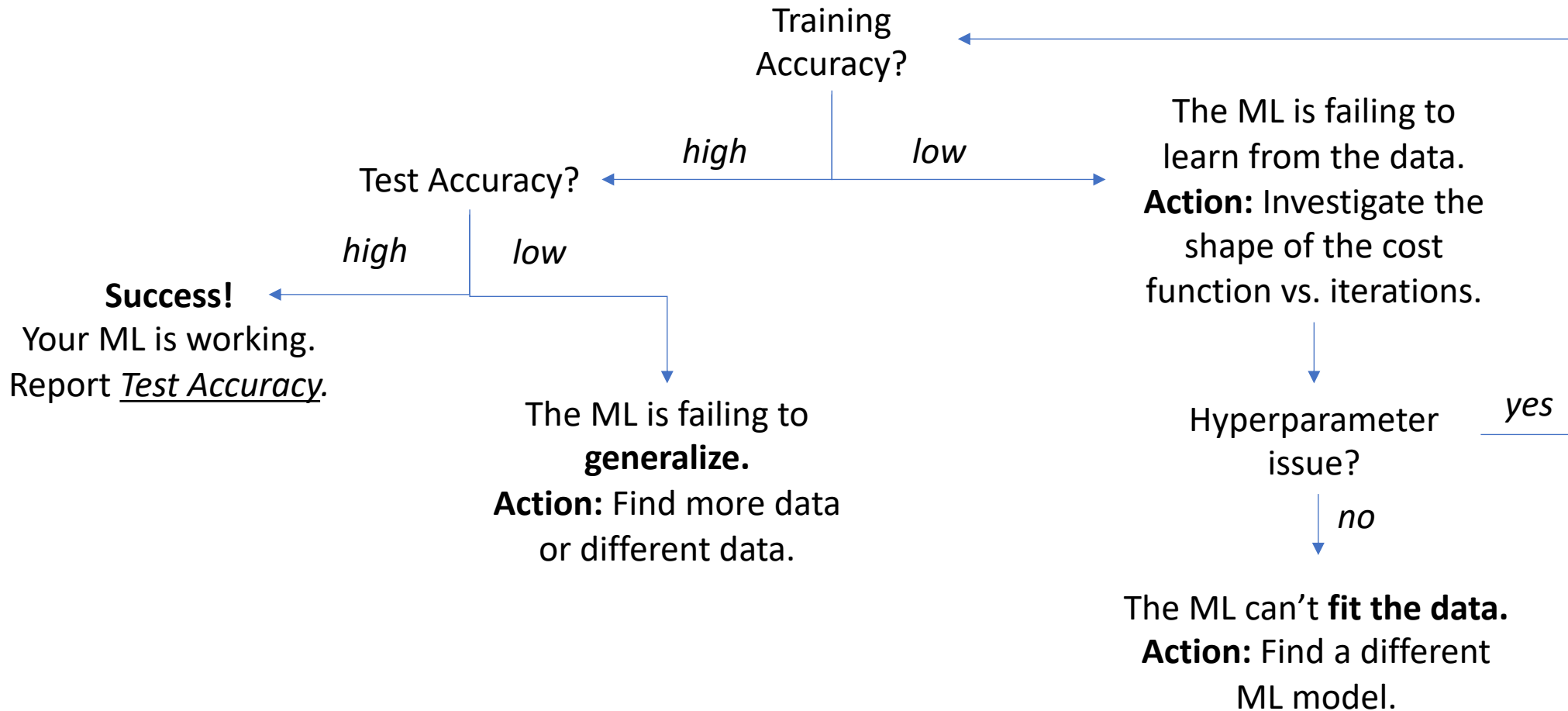


Test Data Set Accuracy

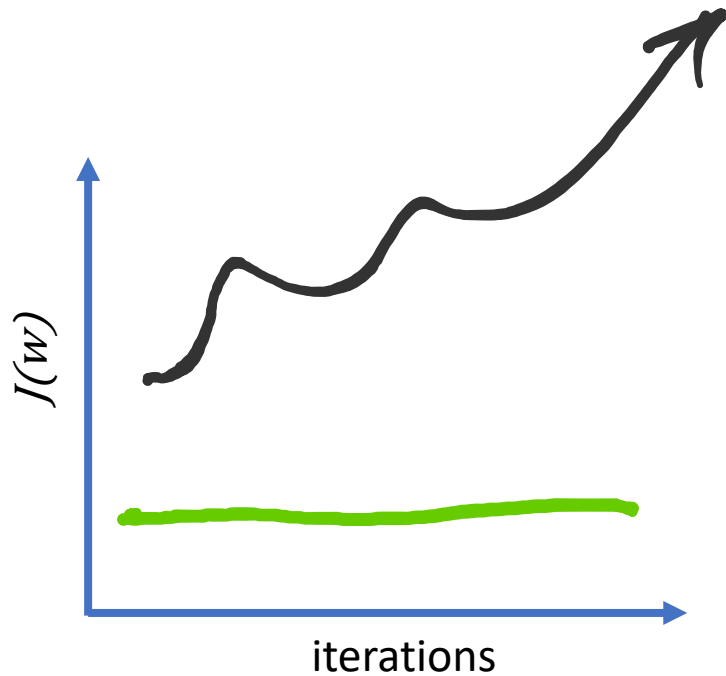
- In the context of our case study, if you want to know how well we will predict the 2020 NBA Draft outcomes, why not try to predict the 2015 outcomes and then compare against the known good answer! -- This is the Test Data Set*.
- Now we have two critical pieces of information: Accuracy on the training data set and the test data set.

*Note: To get an accurate view it is critical that that we don't use the Test Data to train the ML.

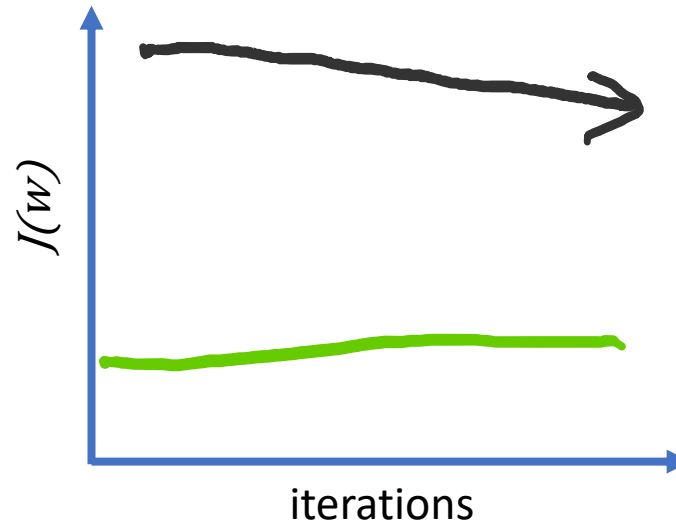
Logistic Regression ML Decision Tree



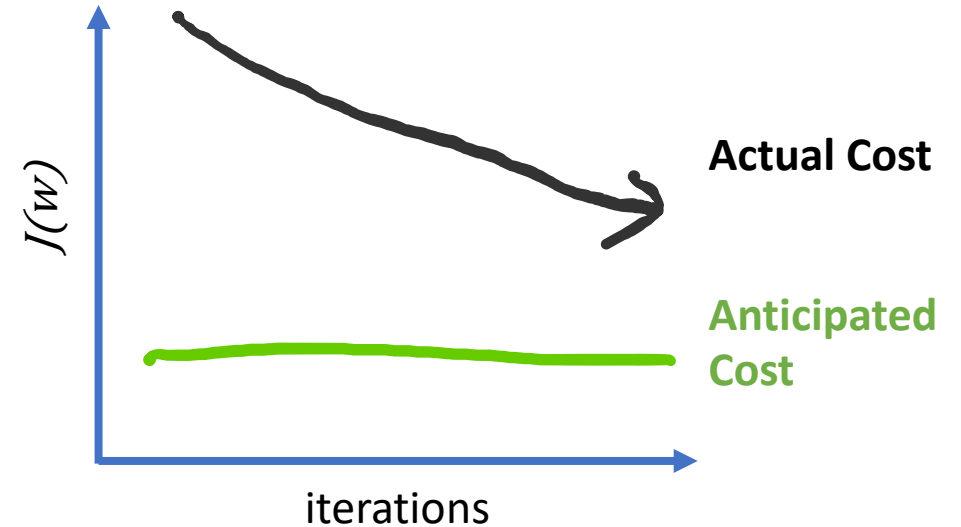
Hyperparameter Issues



Learning rate is too **high**.

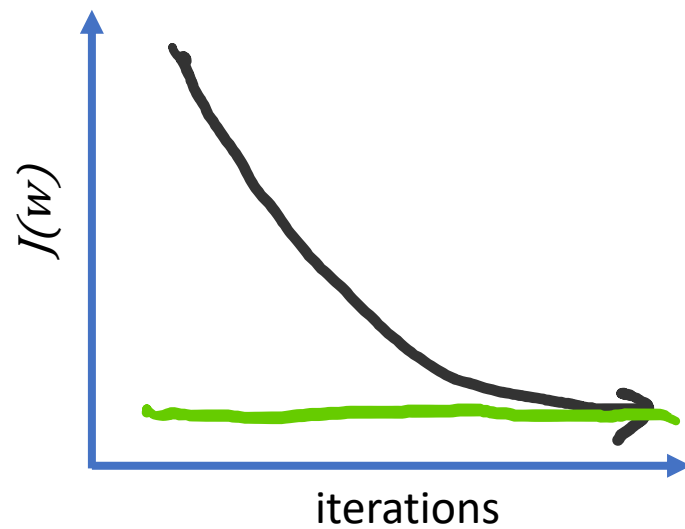


Learning rate is too **low**.

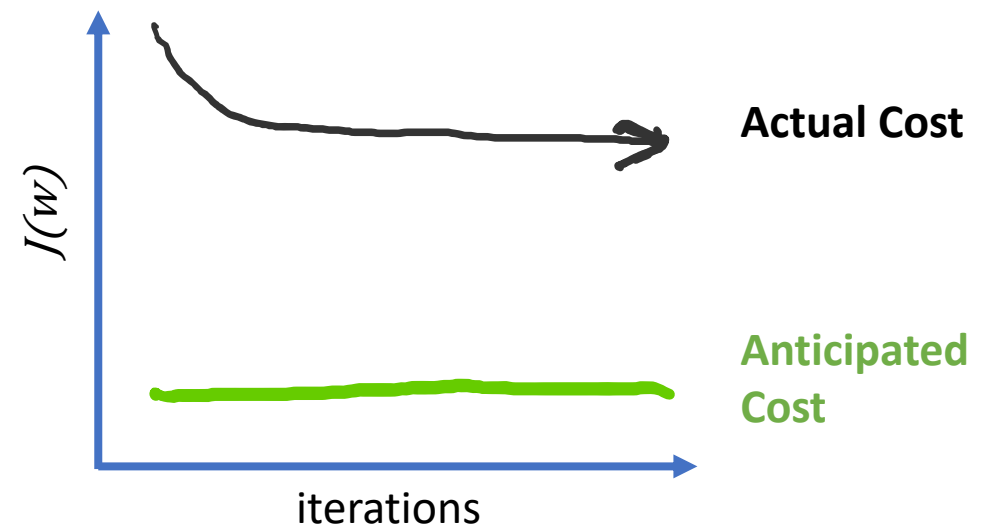


Number of iterations too **low**.

Model Issues



Good behavior.



Learning unsuccessful.
(Data/model fit issue)

Summarizing Potential Issues

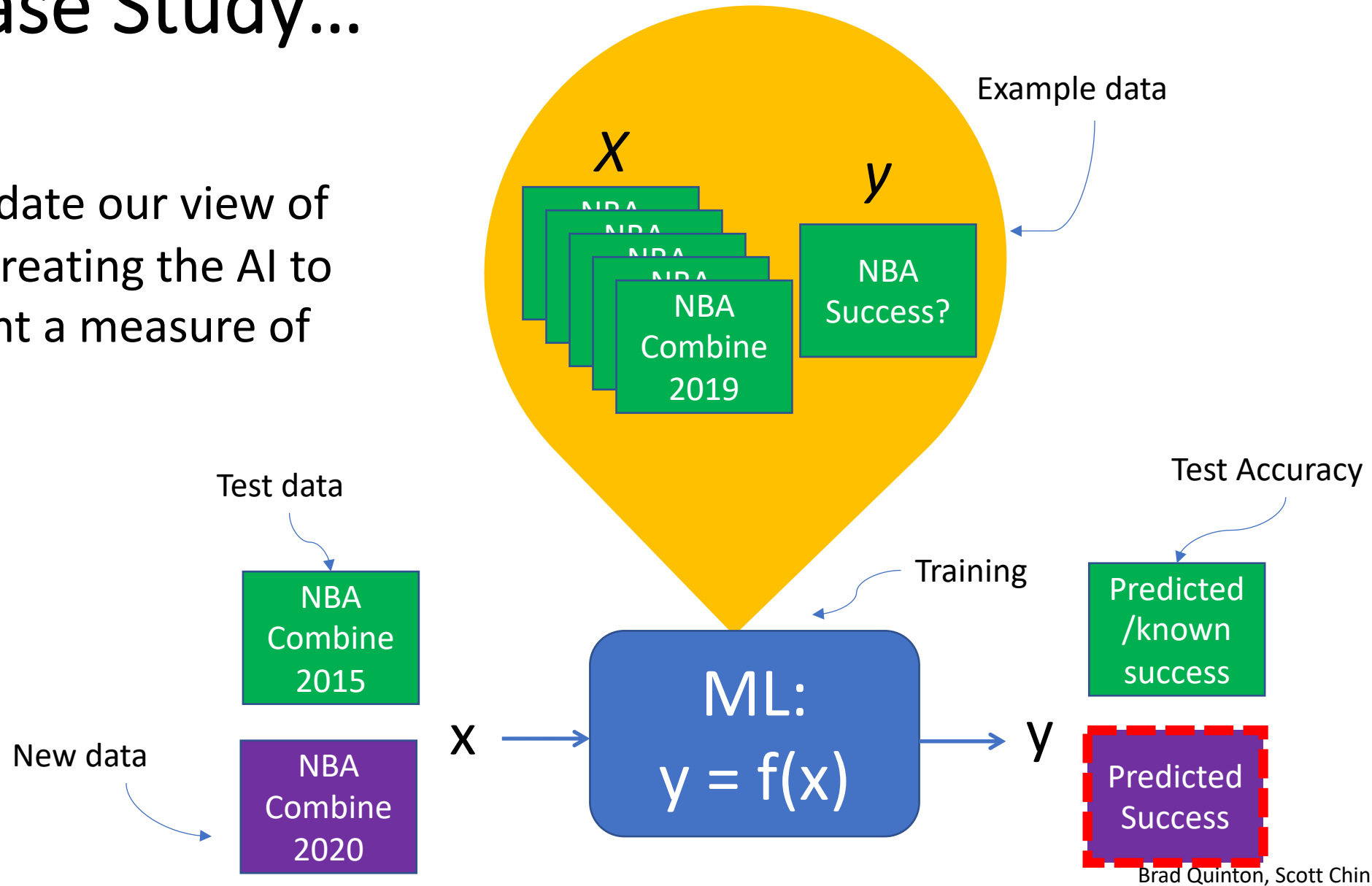
Issue	Symptom	Action
AI Model Doesn't Fit Data.	Training accuracy is low and hyperparameter tuning doesn't help.	Consider a different AI model (NN, for instance)
We are not finding the best parameters.	Unexpected shape of cost/iterations graph.	Tune the hyperparameters.
Example data does not represent the new data. (Lack of data, noisy data, non-deterministic data)	High training accuracy but test accuracy is low.	Try to find more data, better data or different data.

Reporting Logistic Regression Accuracy

- As we have emphasized previously the point of ML is to address new data.
- The key to reporting the accuracy of your ML system correctly is to:
 1. Select a *representative* test data set from your labelled data,
 2. Make sure you don't use the test data to train you ML,
 3. Report the accuracy of the test data set (not the training data set).

Back to Case Study...

- We can now update our view of the process of creating the AI to take into account a measure of accuracy..



Case Study: Meeting #3

- **Day 9:** Third meeting with the coaching staff:

You: *“Thanks for meeting again. I wanted to let you know that I’ve now built an AI using 20 years worth of historic Draft Combine results.”*

Coaches: *“Sounds great. What was that you said about accuracy again..?”*

You: *“Applying our AI to, for example, the 2015 Draft Combine the prediction rate for who would eventually play in the NBA was 87%”*

Coaches: *“Nice. We will certainly want to role this software out to the scouting staff... when will it be ready?”*

Case Study: Project Suggestion

- I have not had time to pull together the real data about NBA player success to be able to do ML on real data (the NBA Combine data is easy, it is here: <https://stats.nba.com/draft/combine-anthro/>)
- If you are a basketball fan and have the time/interest in pulling together the example data and I would be very interested in seeing it (would add your name to these slides going forward!)
- This would also make a nice GitHub example for those who want to have something to reference in on their CV...

ML Implementation and Coding

- In this course we are going to mostly avoid trying to teaching detailed coding techniques in the lectures (because it is not very effective)
- Instead, Jupyter Notebook based assignments will cover detailed implementation and coding issues
- However, for certain very key issues we will cover the conceptual aspects in the lectures as well to assist with the assignments
- Vectorization is one of those key issues....

Vectorization

- Machine Learning, and especially Deep Learning, are very computationally expensive
- This is because the best (most accurate) solutions comes from using:
 1. A **lot of example data**, with
 2. models that contain a **lot of parameters**,
 3. trained over a **lot of iterations**.
- Therefore implementing your code in a computationally efficient way is critical to finding high quality solutions in a reasonable timeframe

Vectorization

- Modern CPUs, GPUs, and special purpose AI silicon all gain computational efficiency by grouping key operations (in this case multiplies) together so they can be executed in parallel
- To leverage these hardware facilities it is important to structure machine learning code correctly
- This process is called **vectorization**

Iterative Operations vs. Matrix Operations


- Consider updating the hypothesis, a , for each of the m examples

```
for i = 1:m
    z = 0
    for k = 1:n
        z = z + w[k]* x[k][i]
    a[i] = sigmoid(z + b)
```

Iterative Operations vs. Matrix Operations

- Consider updating the hypothesis, a , for each of the m examples

```
for i = 1:m
    z = 0
    for k = 1:n
        z = z + w[k]*x[k][i]
    a[i] = sigmoid(z + b)
```



There are $(m * n)$ multiplies here. But they are each updated “one at a time” in nested for loops.

Matrix operations vs. iteration operations

- If we instead look at updating the hypothesis as a matrix operation, we can write:

$$A = \sigma(w^T X + b)$$

- Where:

X is a (n, m) input matrix (*i.e.* all of the input features for each example)

n is the number of features in the input data

m is the number of examples in the training data

w is a $(n, 1)$ parameter matrix

b is the bias

A is a $(1, m)$ vector which represents the hypothesis for each of the m samples

Matrix operations vs. iteration operations

- We can expand this equation:

$$w^T X = [w_1 \ w_2 \ \dots \ w_n] \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(m)} \\ \vdots & \ddots & & \vdots \\ x_n^{(1)} & x_n^{(2)} & \dots & x_n^{(m)} \end{bmatrix}$$

$$w^T X = \left[w_1 x_1^{(1)} + \dots + w_n x_n^{(1)} \quad w_1 x_1^{(2)} + \dots + w_n x_n^{(2)} \quad \dots \quad w_1 x_1^{(m)} + \dots + w_n x_n^{(m)} \right]$$

Matrix operations vs. iteration operations

- And further for A:

$$A = \sigma(w^T X + b)$$

$$A = \sigma\left(\begin{bmatrix} w_1 x_1^{(1)} + \dots + w_n x_n^{(1)} & w_1 x_1^{(2)} + \dots + w_n x_n^{(2)} & \dots & w_1 x_1^{(m)} + \dots + w_n x_n^{(m)} \end{bmatrix} + b\right)$$

$$A = \left[\sigma\left(w_1 x_1^{(1)} + \dots + w_n x_n^{(1)} + b\right) \underbrace{\sigma\left(w_1 x_1^{(2)} + \dots + w_n x_n^{(2)} + b\right)} \dots \sigma\left(w_1 x_1^{(m)} + \dots + w_n x_n^{(m)} + b\right) \right]$$

$A[1, i]$ is the hypothesis for the i^{th} example in the train data set.

Using NumPy for Matrix Operations

- By reformatting the equation to use matrix operations, we leverage the specially written NumPy package:

```
A = np.sigmoid(np.matmul(w.transpose(), X) + b)
```

- As you will see in the assignment, this will substantially increase the runtime performance, which will be critical when we get into Deep Learning

Aside #1: Assumptions, and more Assumptions...

- Deep Learning and Machine Learning is full of **assumptions** and **approximations**
- There has been a tendency amongst some in the field to minimize or ignore these assumptions (*“who cares, it just works...”*)
- My personal view is that in the long-term this is **not productive**, nor is it **acceptable** for many engineering applications.
- Rather, we should strive to be aware of our assumptions, and approximations, and ask ourselves ***“how would we know if this is wrong?”***

Aside #2: A Data Science Point of View

- It is worth considering that AI/ML is not the only way to get value from example data, from a Data Science point of view we can use data to build **understanding** that can be used by humans to make better choices
- So, and different response to the Raptors problem would be to do a statistical analysis of historical data to draw conclusions for the talent scouts, something like:

“data shows that the best draft picks are those who wingspan exceeds their height, but have a strength/weight ratio 1 ...”

Key Take-Aways

- It is important to use known good data to test your ML, but it is critical that the test data is not used to train the ML
- Optimal ML results require hyperparameter tuning (even more so in Deep Learning)
- For computational efficiency we need to use vectorized implementations of our algorithms and leverage specialized packages such as NumPy