Course series: Deep Learning for Machine Translation

Machine Learning

Lecture # 3

Hassan Sajjad and Fahim Dalvi
Qatar Computing Research Institute, HBKU

Machine Learning

A technique that gives **machines** the **ability** to **learn**, without any explicit programming

Machine Learning

In simpler terms, a machine should be able to **see** some **data**, and **learn to make decisions** based on what it has seen

Machine Learning

An example: You are a car dealer, and you have a historical record of **which cars are accident prone**. How can you "teach" a computer to predict which *new* cars will be accident prone?

	Maximum Speed	Acceleration	Color	Car age	Accident Prone?
Car 1	240 km/h	Fast	Red	2 yrs	Yes
Car 2	100 km/h	Fast	Yellow	2 yrs	No
Car 3	240 km/h	Fast	Blue	1 yr	No
Car 4	200 km/h	Slow	Blue	5 yrs	Yes
Car 5	100 km/h	Fast	Yellow	5 yrs	Yes
Car 6	100 km/h	Slow	Black	6 yrs	No
Car 7	150 km/h	Fast	Red	2 yrs	?

Machine Learning

How did you make your decision?

Search for closest vehicle in the past?

Come up with a set of rules?

How did you decide what knowledge is important?

Machine Learning

Historically, rule based systems were common:

```
if (car.acceleration = fast and car.age > 1 and ...)
    print (“accident prone”)
else if (car.acceleration = slow and car.maxspeed > 150 and ...)
    print (“accident prone”)
else if (car.acceleration = slow and car.maxspeed < 50)
    print (“not accident prone”)
else if
 ...
 ...
```

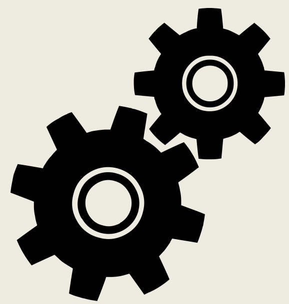
Domain Specific | Cumbersome | Not easy to learn from new data

Machine Learning

Then, machine learning techniques came about...

	...	Accident Prone?
Car 1	...	Yes
Car 2	...	No
Car 3	...	No
Car 4	...	Yes
Car 5	...	Yes
Car 6	...	No

Training data



Machine Learning Algorithms

$$f$$

“Model”

Domain Agnostic

Robust

Easy to learn from new data

Training Data

	Maximum Speed	Acceleration	Color	Car age	Accident Prone?
Car 1	240 km/h	Fast	Red	2 yrs	Yes
Car 2	100 km/h	Fast	Yellow	2 yrs	No
Car 3	240 km/h	Fast	Blue	1 yr	No
Car 4	200 km/h	Slow	Red	5 yrs	Yes
Car 5	100 km/h	Fast	Yellow	5 yrs	Yes
Car 6	100 km/h	Slow	Black	6 yrs	No

Input Features Labels

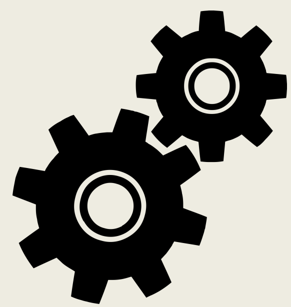
We use training examples with labels to train a model

Machine Learning

Then, machine learning techniques came about...

	...	Accident Prone?
Car 1	...	Yes
Car 2	...	No
Car 3	...	No
Car 4	...	Yes
Car 5	...	Yes
Car 6	...	No

Training data



Machine Learning Algorithms

$$f$$

"Model"

Domain Agnostic

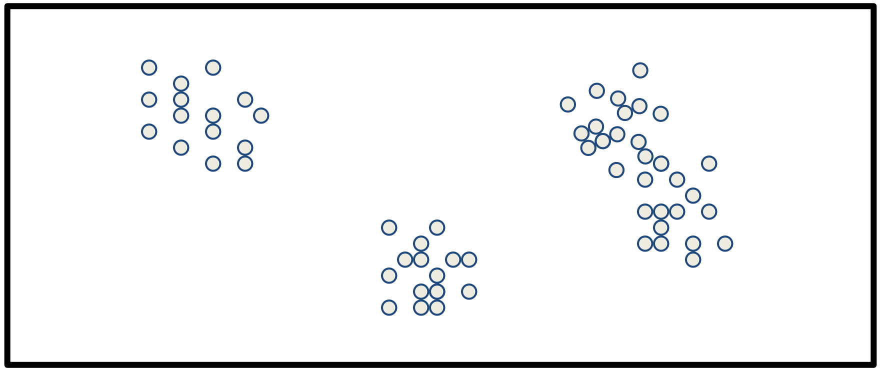
Robust

Easy to learn from new data

Algorithms

In this case, we have labels for each car. This class of problems is handled by *supervised learning algorithms*.

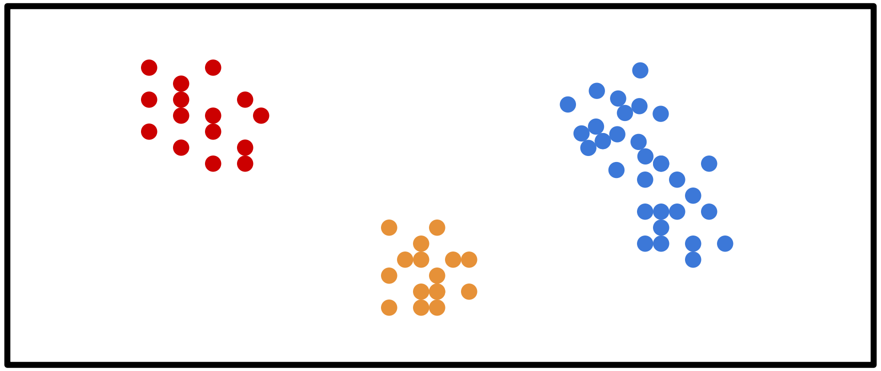
Unsupervised learning algorithms work on unlabelled data



Algorithms

In this case, we have labels for each car. This class of problems is handled by *supervised learning algorithms*.

Unsupervised learning algorithms work on unlabelled data



Algorithms

In this case, we have labels for each car. This class of problems is handled by *supervised learning algorithms*.

Unsupervised learning algorithms work on unlabelled data

Algorithms

Many techniques exist to build models:

- "Finding similar cars" type methods:
  - K-means clustering
  - Hierarchical clustering
- "Create set of rules" type methods:
  - Support vector machines
  - Logistic Regression
  - Neural Networks

Algorithms

Many techniques exist to build models:

- “Finding similar cars” type methods:
  - K-means clustering
  - Hierarchical clustering

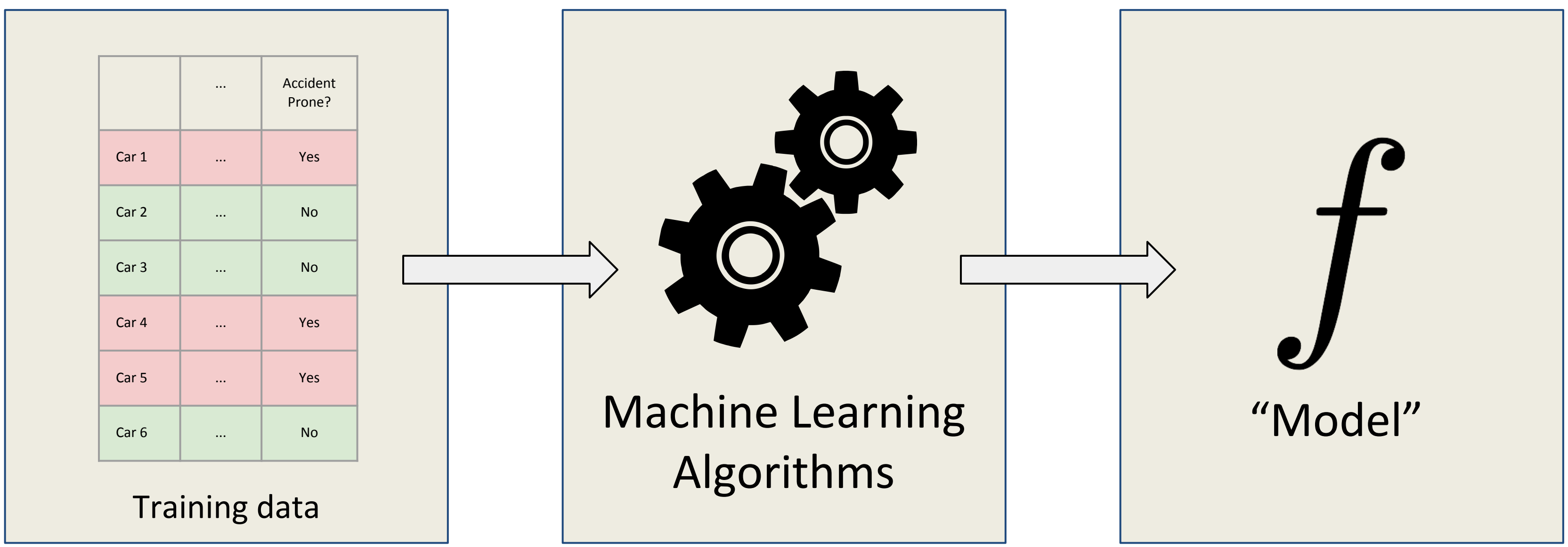
Unsupervised

- “Create set of rules” type methods:
  - Support vector machines
  - Logistic Regression
  - Neural Networks

Supervised

Machine Learning

Then, machine learning techniques came about...



	...	Accident Prone?
Car 1	...	Yes
Car 2	...	No
Car 3	...	No
Car 4	...	Yes
Car 5	...	Yes
Car 6	...	No

Domain Agnostic

Robust

Easy to learn from new data

Machine Learning Model

Example of a model: *A function that takes information about a car, and predicts whether it's accident prone or not*

`f(`  `)  =  No`

`f(`  `)  =  Yes`

Machine Learning Model

Example of a model: *A function that takes a word, and predicts it's part of speech tag*

```
      f(“car”)  =  Noun
f(“beautiful”)  =  Adjective
      f(“she”)  =  Pronoun
```

Machine Learning Model

At a high level, the basic idea is to figure out which features are important, and how important are they for prediction

```
  0.3 x car.maxspeed
+ 0.2 x car.acceleration
+ 0.0 x car.color
+ 0.5 x car.age
```

Machine Learning Model

At a high level, the basic idea is to figure out which features are important, and how important are they for prediction

```
  0.3 x car.maxspeed
+ 0.2 x car.acceleration
+ 0.0 x car.color
+ 0.5 x car.age
```

An older car is more likely to be accident prone

Machine Learning Model

At a high level, the basic idea is to figure out which features are important, and how important are they for prediction

```
  0.3 x car.maxspeed
+ 0.2 x car.acceleration
+ 0.0 x car.color
+ 0.5 x car.age
```

The color of a car has no impact on accidents

Supervised Learning: Classification

Process of assigning objects to categories

For example, Car 1 belongs to category *"Accident Prone"*

	Maximum Speed	Acceleration	Color	Car age	Accident Prone?
Car 1	240 km/h	Fast	Red	2 yrs	Yes
Car 2	100 km/h	Fast	Yellow	2 yrs	No
Car 3	240 km/h	Fast	Blue	1 yr	No
Car 4	200 km/h	Slow	Blue	5 yrs	Yes
Car 5	100 km/h	Fast	Yellow	5 yrs	Yes
Car 6	100 km/h	Slow	Black	6 yrs	No

Supervised Learning: Classification

Process of assigning objects to categories

Car Example
Accident Prone
Not Accident Prone

Binary Classification

POS Example	
Noun	Verb
Pronoun	Adjective

Multiclass Classification

Supervised Learning: Classification

Process of assigning objects to categories

Car Example

Accident Prone

Not Accident Prone

Binary Classification

Two classes: yes or no

POS Example

Noun

Verb

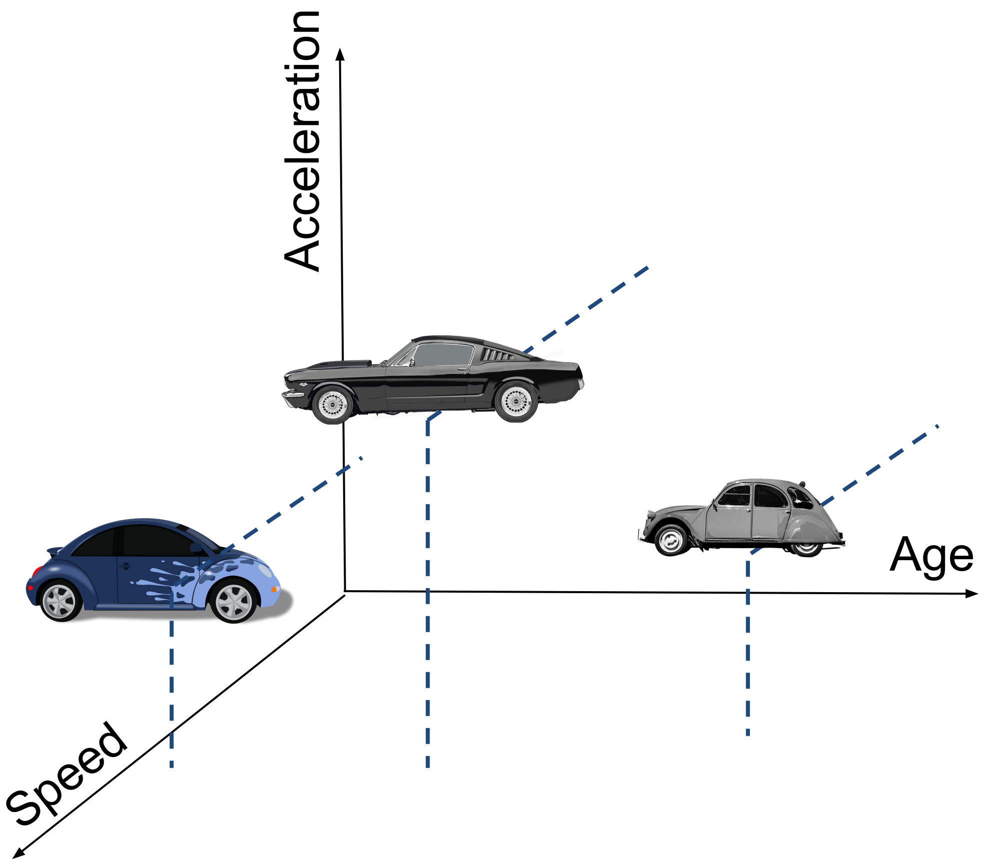
Pronoun

Adjective

Multiclass Classification

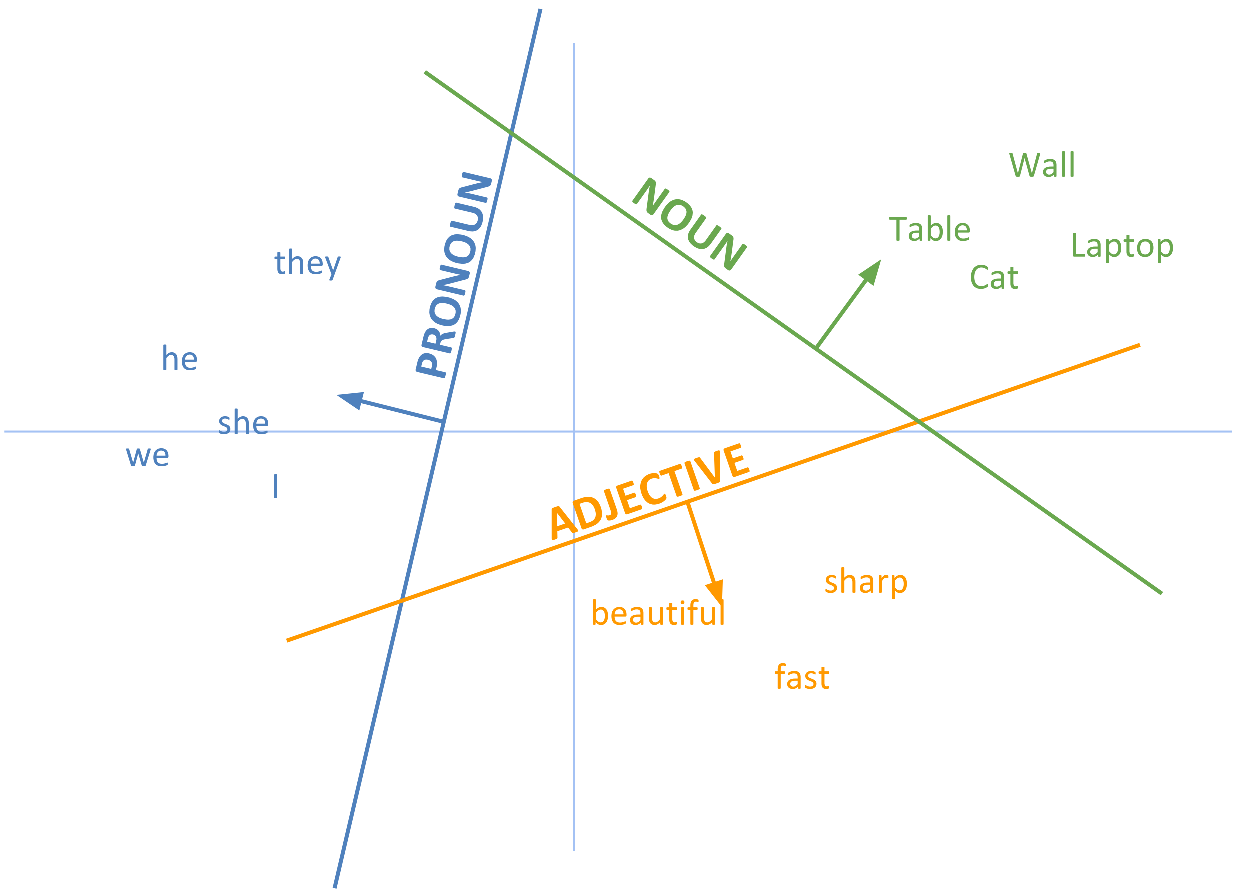
More than two classes to choose from

Classification: Vector Spaces

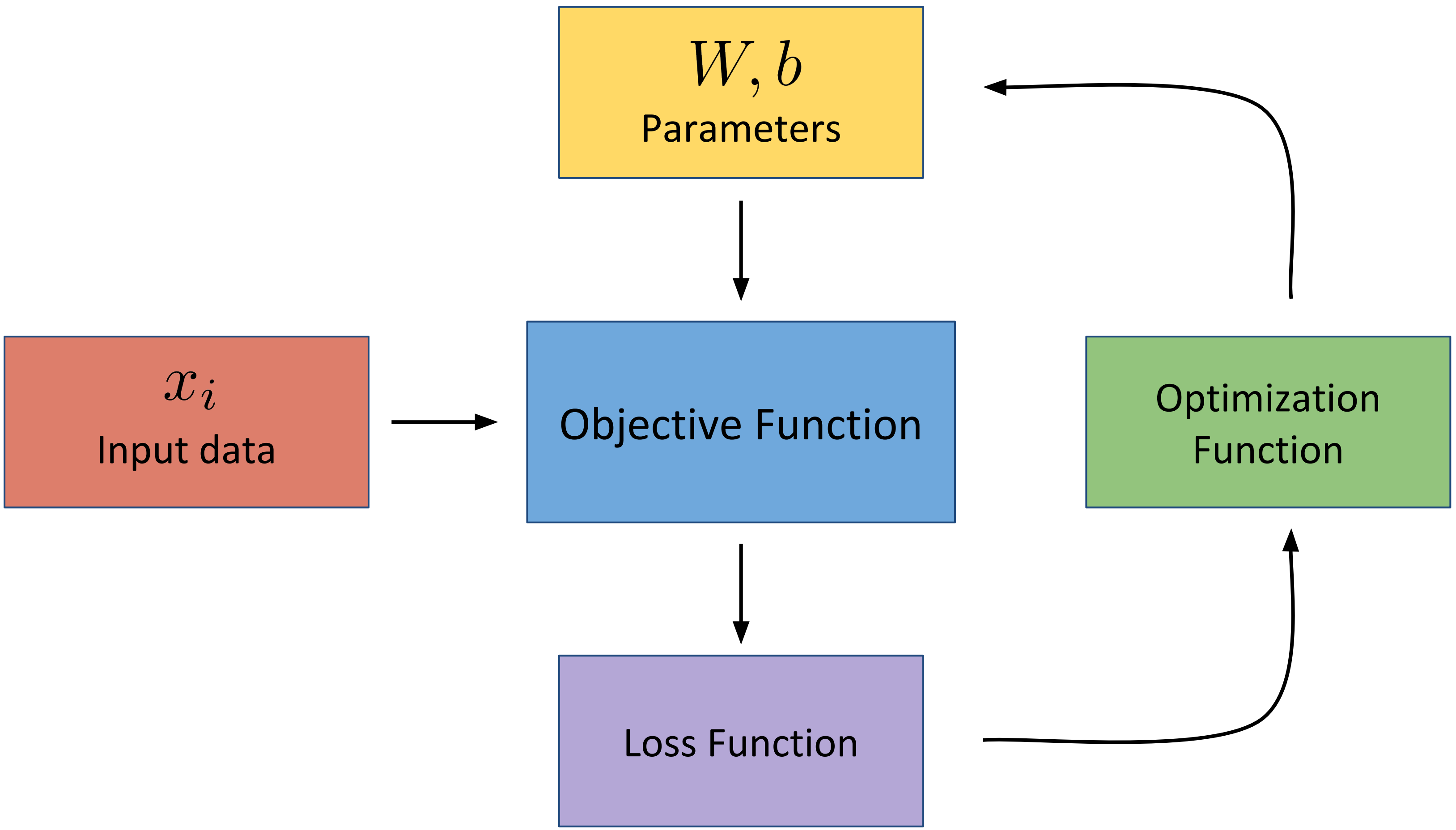


Imagine every feature as a dimension in space
Every object (car) can be represented as a point in space

Classification: Vector Spaces



Ingredients of a Classifier



Classification Exercise

- We will start by looking at a simple technique - ***Linear regression* for binary classification**

Classification Exercise

- We will start by looking at a simple technique - ***Linear regression*** **for binary classification**
- In Linear regression, our model/function predicts just one real number
- If this number is < 0, we will consider it to belong to **Class 1**. If it is ≥ 0, we will consider it to belong to **Class 2**.

Classification Exercise

- Choose w_0, w_1 and b such that positive examples give a result > 0 and negative examples give a result < 0

$$w_0 \times x_0 + w_1 \times x_1 + b$$

x_0	x_1	**Class**
2	0	Positive
5	-2	Positive
-2	2	Negative
-1	-3	Negative

Classification Exercise

- Choose w_0, w_1 and b such that positive examples give a result > 0 and negative examples gi

```
  0.3 x car.maxspeed
+ 0.2 x car.acceleration
+ 0.0 x car.color
+ 0.5 x car.age
```

$$w_0 \times \quad + b$$

x_0	x_1	Class
2	0	Positive
5	-2	Positive
-2	2	Negative
-1	-3	Negative

Classification Exercise

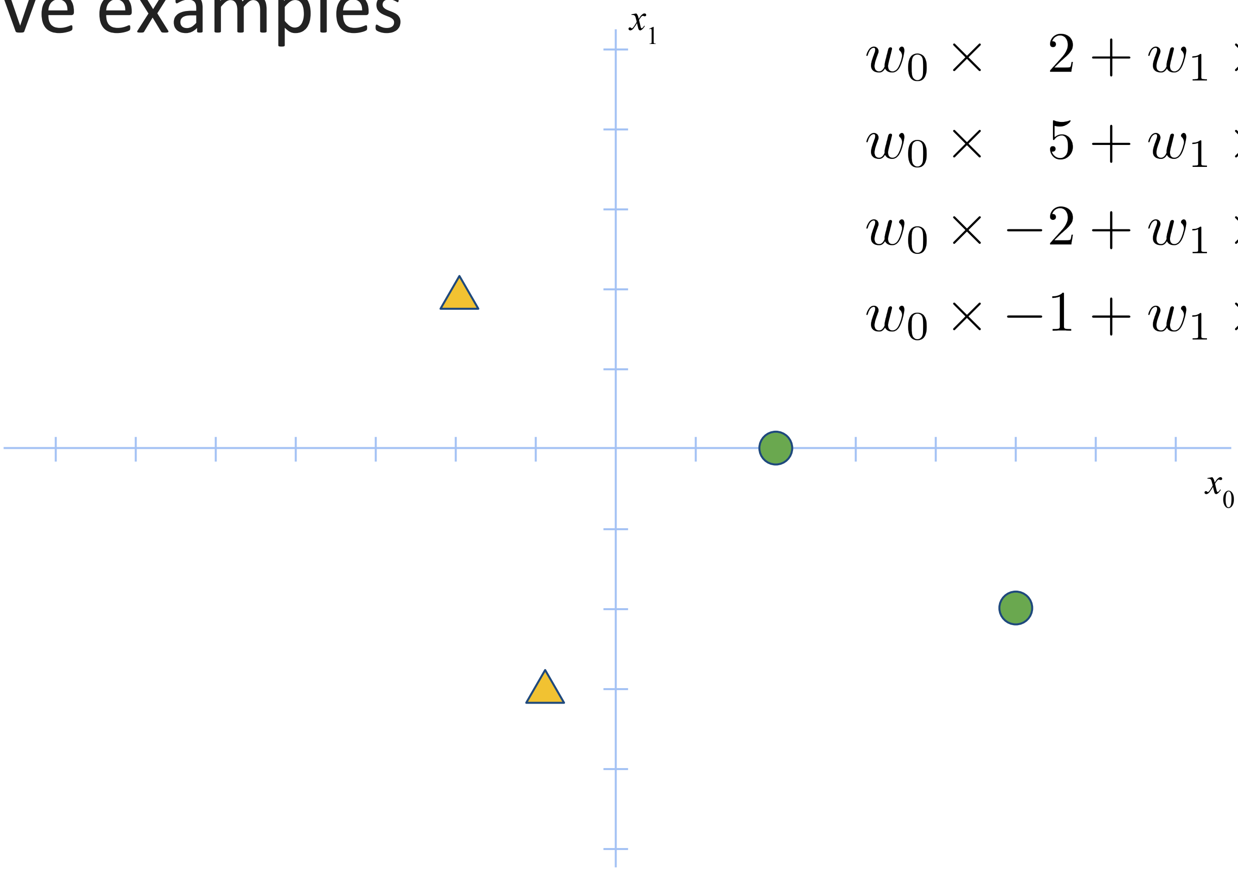
- Choose w_0, w_1 and b such that positive examples give a result > 0 and negative examples give a result < 0

$$w_0 \times x_0 + w_1 \times x_1 + b$$

x_0	x_1	**Class**
2	0	Positive
5	-2	Positive
-2	2	Negative
-1	-3	Negative

Classification Exercise

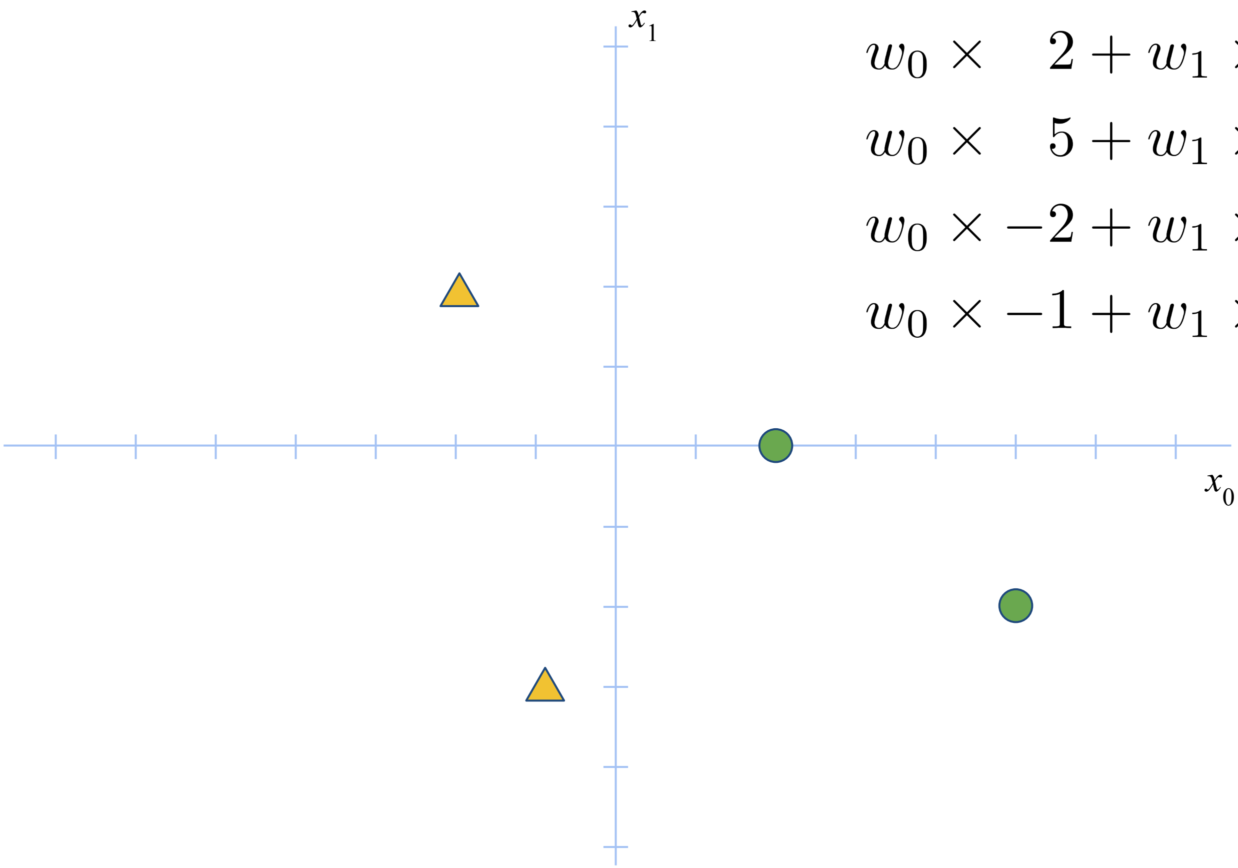
Find weights that separate positive examples from negative examples



$$w_0 \times 2 + w_1 \times 0 + b > 0$$
$$w_0 \times 5 + w_1 \times -2 + b > 0$$
$$w_0 \times -2 + w_1 \times 2 + b < 0$$
$$w_0 \times -1 + w_1 \times -3 + b < 0$$

Classification Exercise

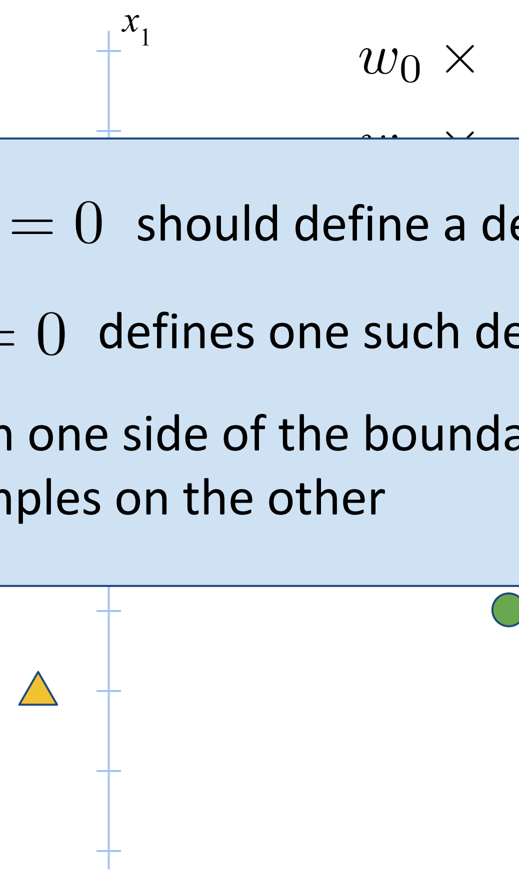
- Potential Solution: $w_0 = 3, w_1 = 1$ and $b = 3$



$$
\begin{aligned}
w_0 \times 2 + w_1 \times 0 + b &> 0 \\
w_0 \times 5 + w_1 \times -2 + b &> 0 \\
w_0 \times -2 + w_1 \times 2 + b &< 0 \\
w_0 \times -1 + w_1 \times -3 + b &< 0
\end{aligned}
$$

Classification Exercise

- Potential Solution: $w_0 = 3, w_1 = 1$ and $b = 3$



$$w_0 \times 2 + w_1 \times 0 + b > 0$$

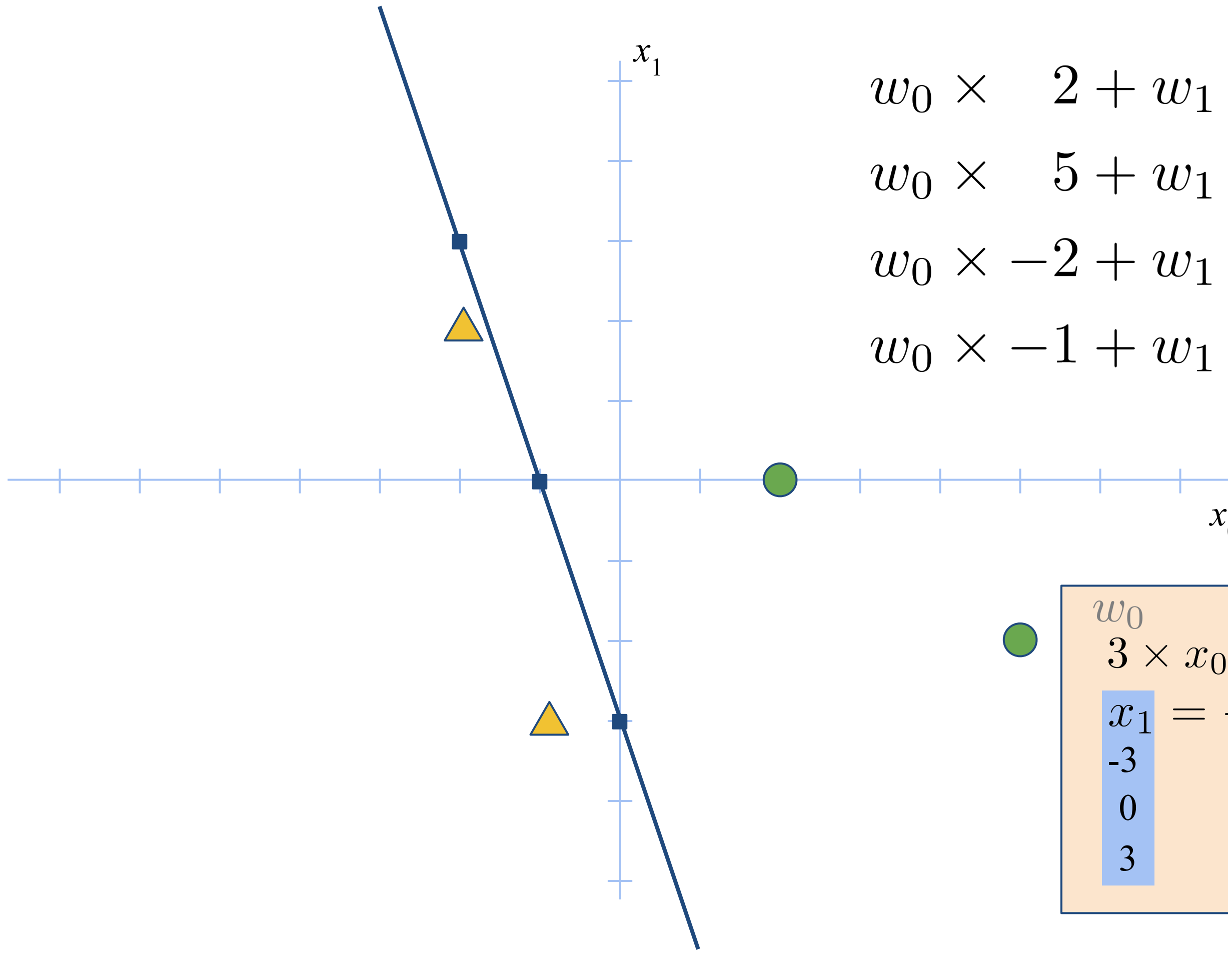
$w_0 \times x_0 + w_1 \times x_1 + b = 0$ should define a decision boundary

$3 \times x_0 + 1 \times x_1 + 3 = 0$ defines one such decision boundary

Positive examples will be on one side of the boundary, and negative examples on the other

Classification Exercise

- Potential Solution: $w_0 = 3, w_1 = 1$ and $b = 3$



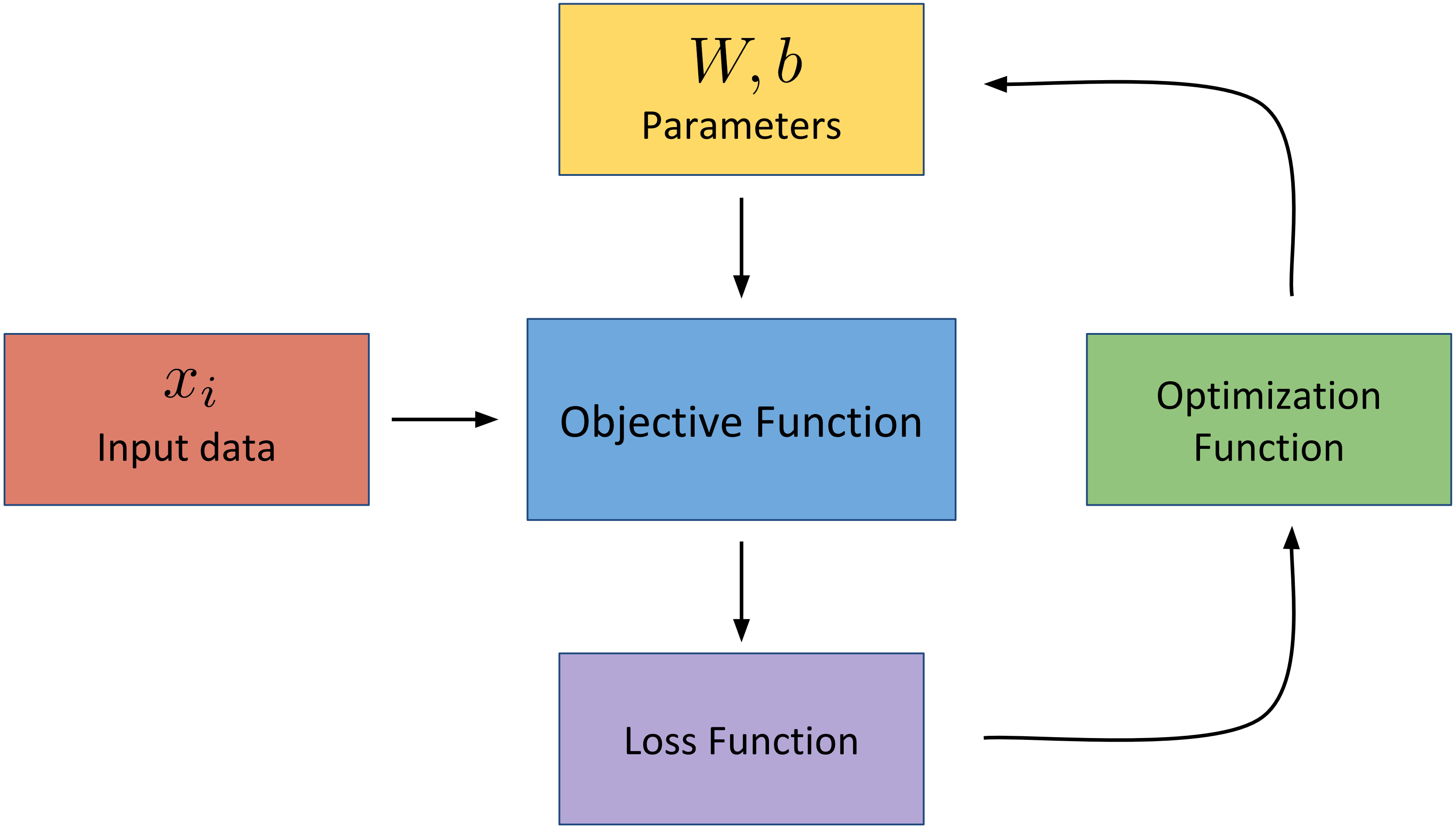
$$
\begin{aligned}
w_0 \times \ \ 2 + w_1 \times \ \ 0 + b > 0 \\
w_0 \times \ \ 5 + w_1 \times -2 + b > 0 \\
w_0 \times -2 + w_1 \times \ \ 2 + b < 0 \\
w_0 \times -1 + w_1 \times -3 + b < 0
\end{aligned}
$$

$$\overset{w_0}{3} \times x_0 + \overset{w_1}{1} \times x_1 + \overset{b}{3} = 0$$

$$x_1 = -3 \times x_0 - 3$$

x_1	x_0
-3	0
0	-1
3	-2

Ingredients of a Classifier



Objective Function

$$f(x, W, b) = W \cdot x + b$$

Objective function defines our *goal*

`f(`  `, W, b) = 2 Numbers`

Input

Parameters
P

Outputs
The scores for the two classes - *accident prone* and *not accident prone*

Objective Function

$$f(x, W, b) = W \cdot x + b$$

Objective function defines our *goal*

Can also be just one number as in the case of *Linear regression*, but in general for classification, we have **n** numbers if we want to classify between **n** classes

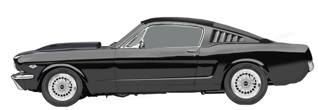
`=   2 Numbers`

Outputs
The scores for the two classes - *accident prone* and *not accident prone*

Objective Function

$$f(x, W, b) = W \cdot x + b$$

Objective function defines our *goal*

`f(`  `, W, b) = 2 Numbers`

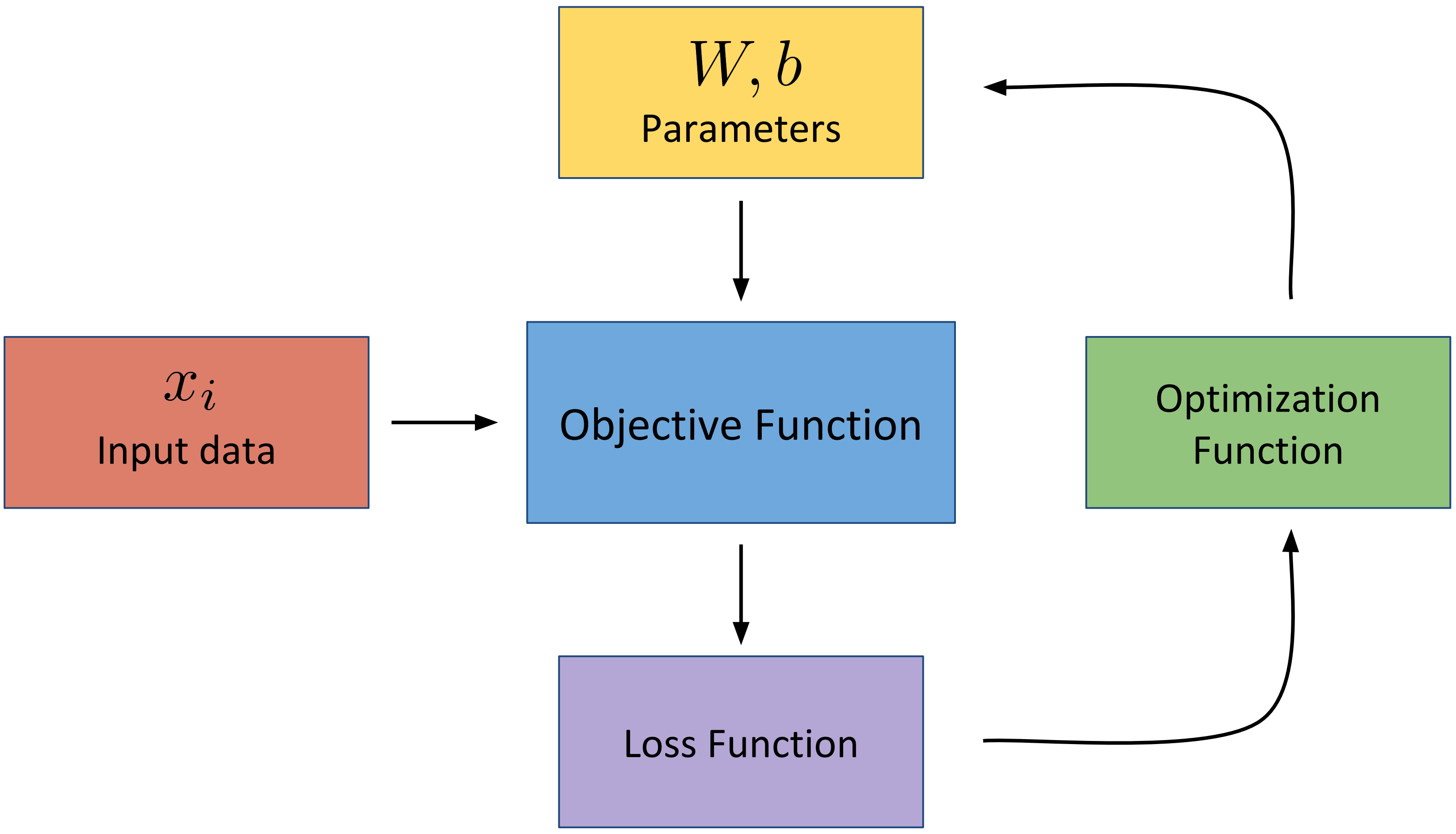
Input

Parameters
P

Outputs
The scores for the two classes - *accident prone* and *not* *rone*

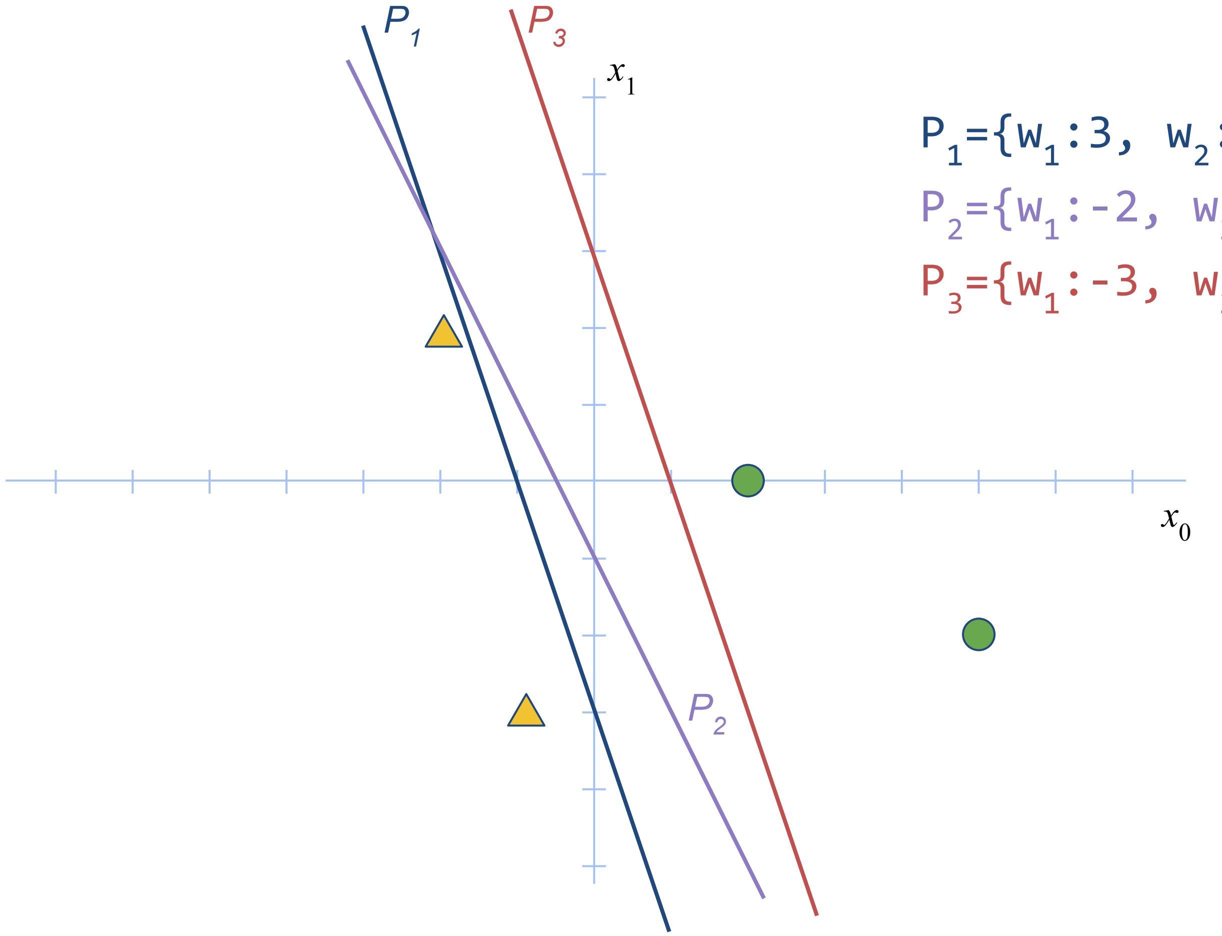
Learned by the algorithm, just like you learned `W` and b in the previous exercise!

Ingredients of a Classifier



Loss Function

Given a set of parameters P={P_1,P_2,…}, how do you know which one to use?



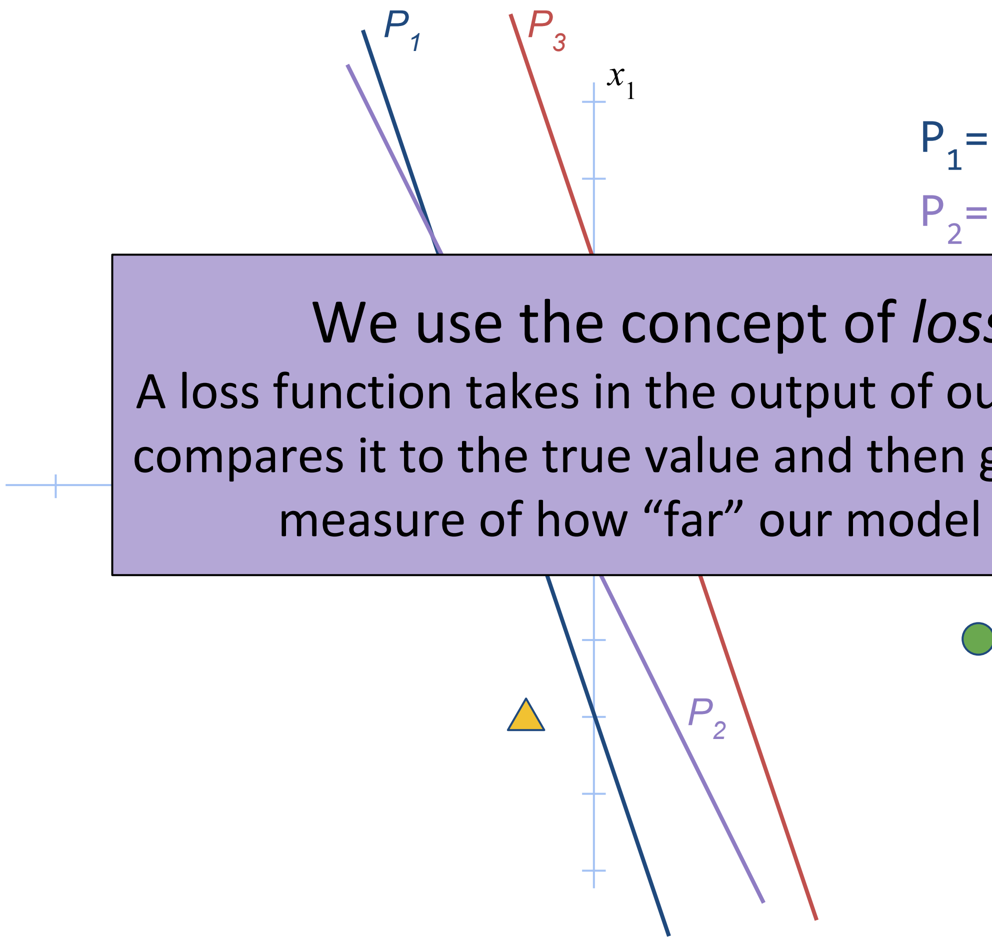
P_1={w_1:3, w_2:1, b:3}

P_2={w_1:-2, w_2:-1, b:-1}

P_3={w_1:-3, w_2:-1, b:3}

Loss Function

Given a set of parameters P={P_1,P_2,…}, how do you know which one to use?



P_1={w_1:3, w_2:1, b:3}

P_2={w_1:-2, w_2:-1, b:-1}

$_2$:-1, b:3}

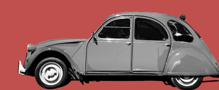
We use the concept of *loss*

A loss function takes in the output of our model, compares it to the true value and then gives us a measure of how “far” our model is.

Loss Function Exercise

Consider two cars and three sets of parameters

Not accident prone



Accident prone

P_1 P_2 P_3

Which set of parameters is the best?

f( , P_1) = [0.5, 0.5]

f( , P_2) = [0.7, 0.3]

f( , P_3) = [0.1, 0.9]

f( , P_1) = [0.1, 0.9]

f( , P_2) = [0.3, 0.7]

f( , P_3) = [0.9, 0.1]

Loss Function Exercise

Consider two cars and three sets of parameters

Not accident prone

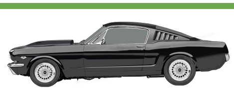
Accident prone

P_1 P_2 P_3

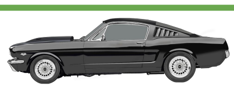
Which set of parameters is the best?

f( , P_1) = [0.5, 0.5]

f( , P_1) = [0.1, 0.9]

f( , P_2) = [0.7, 0.3]

f( , P_2) = [0.3, 0.7]

f( , P_3) = [0.1, 0.9]

f( , P_3) = [0.9, 0.1]

Loss Function

A loss function is *any function that gives a measure of how far your scores are from their true values*

Loss Function

A loss function is *any function that gives a measure of how far your scores are from their true values*

Not accident prone

Accident prone

`[1.0, 0.0]` ← True values → `[0.0, 1.0]`

Loss Function

A potential loss function in this case is the *sum of the absolute difference* of scores:

L( , P_1) = sum(f( , P_1) - [1.0, 0.0])

= sum([|-0.5| , |0.5|]) = 1

L( , P_1) = sum(f( , P_1) - [0.0, 1.0])

= sum([|0.1|, |-0.1|]) = 0.2

$$L = \sum_{i=1}^{n} |x_i - y_i|$$

Loss Function

A potential loss function in this case is the *sum of the absolute difference* of scores:

L( , P_1) = sum(f( , P_1) - [1.0, 0.0])

= sum([|-0.5| , |0.5|]) = 1

L( , P_1) = sum(f( , P_1) - [0.0, 1.0])

= sum([|0.1|, |-0.1|]) = 0.2

L( , P_2) = 0.6

L( , P_3) = 1.8

L( , P_2) = 0.6

L( , P_3) = 1.8

Loss Function

A potential loss function in this case is the *sum of the absolute difference* of scores:

L( , P_1) = sum(f( , P_1) - [1.0, 0.0])
= sum([|-0.5| , |0.5|]) = 1

L( , P_1) = sum(f( , P_1) - [0.0, 1.0])
= sum([|0.1|, |-0.1|]) = 0.2

L( , P_2) = 0.6
L( , P_3) = 1.8

L( , P_2) = 0.6
L( , P_3) = 1.8

Average loss for both cars

L(P_1) = 0.6   L(P_2) = 0.6   L(P_3) = 1.8

Loss Function

```
Average loss for both cars
L(P1) = 0.6    L(P2) = 0.6     L(P3) = 1.8
```

A lower value of the loss indicates a better model

i.e. we are closer to the true values

In this case, P_1 and P_2 have the *lower value* of 0.6, so we know they are better than P_3. However, we also know that P_2 is better than P_1 , and this implies our loss function is not very good right now!

Loss Function

Better loss function:

Mean Squared Error

Loss is equal to the sum of the square of the differences in the scores

$$L = \sum_{i=1}^{n} (x_i - y_i)^2$$

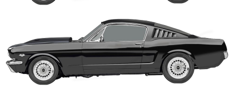
Loss Function

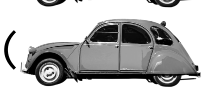
Better loss function:

Mean Squared Error

Loss is equal to the sum of the square of the differences in the scores

f( , P_1) = [0.5, 0.5] f( , P_1) = [0.1, 0.9]

f( , P_2) = [0.7, 0.3] f( , P_2) = [0.3, 0.7]

f( , P_3) = [0.1, 0.9] f( , P_3) = [0.9, 0.1]

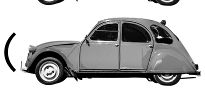
Loss Function

Better loss function:

Mean Squared Error

Loss is equal to the sum of the square of the differences in the scores

MSE( , P_1) = 0.50   MSE( , P_1) = 0.02

MSE( , P_2) = 0.18   MSE( , P_2) = 0.18

MSE( , P_3) = 1.62   MSE( , P_3) = 1.62

Loss Function

Better loss function:

Mean Squared Error

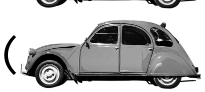
Loss is equal to the sum of the square of the differences in the scores

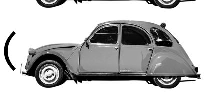
MSE( , P_1) = 0.50

MSE( , P_2) = 0.18

MSE( , P_3) = 1.62

MSE( , P_1) = 0.02

MSE( , P_2) = 0.18

MSE( , P_3) = 1.62

Average loss

$L(P_1) = 0.26$ $L(P_2) = 0.18$ $L(P_3) = 1.62$

Loss Function

Mean Squared Error works better, as it penalizes values that are further away more.

Loss Function

Many other choices for loss functions:

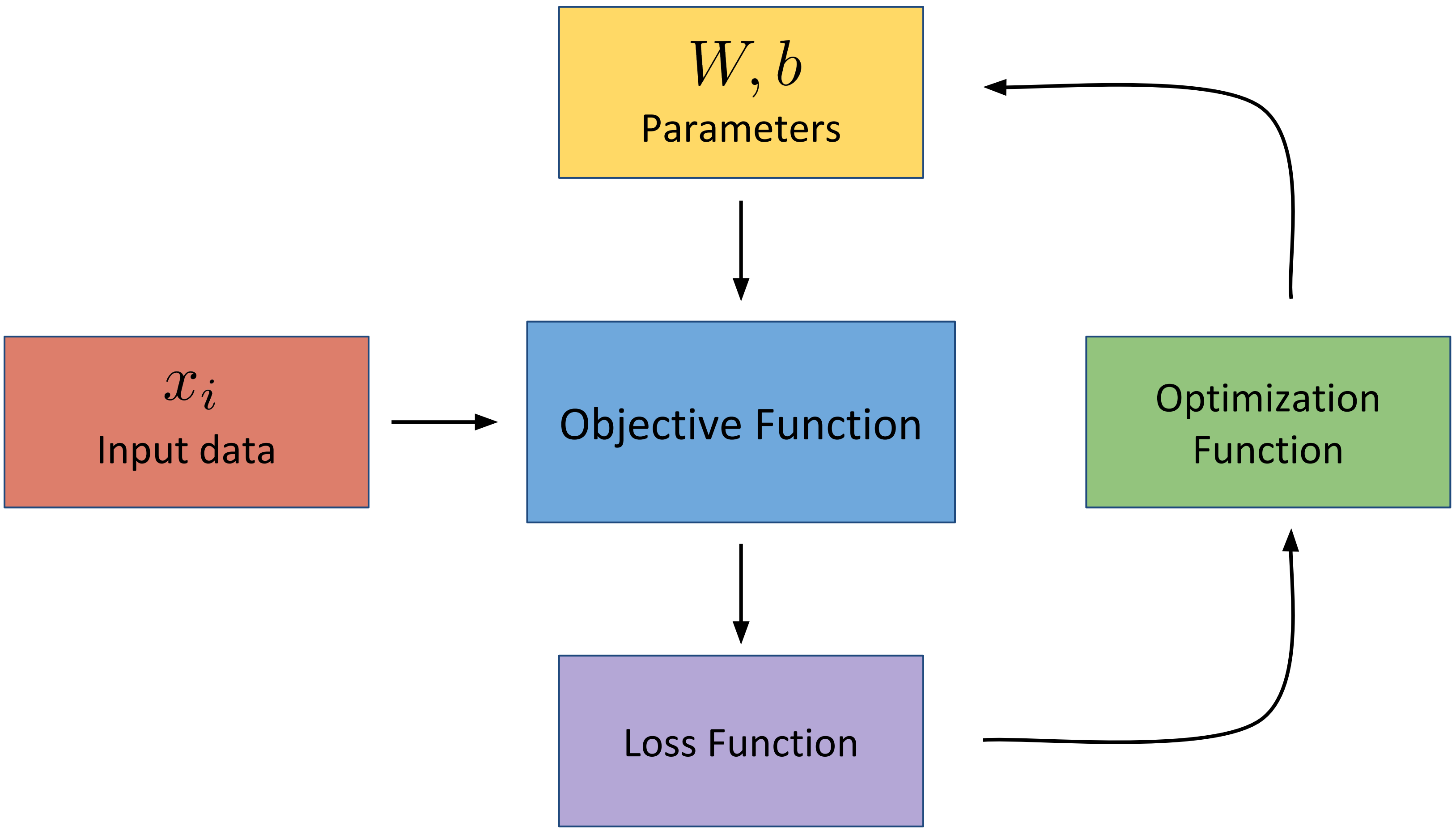
- Absolute Distance loss
- Hinge loss
- Logistic loss
- Cross Entropy loss

⋮

Loss Function

Loss function is also known as the *cost function* in some literature

Ingredients of a Classifier



Optimization

Now that we have a way of defining loss, we need a way to use it to improve our parameters

This process is called optimization - where our goal is to “minimize” the loss function, i.e. bring it as close to zero as possible

Optimization Exercise

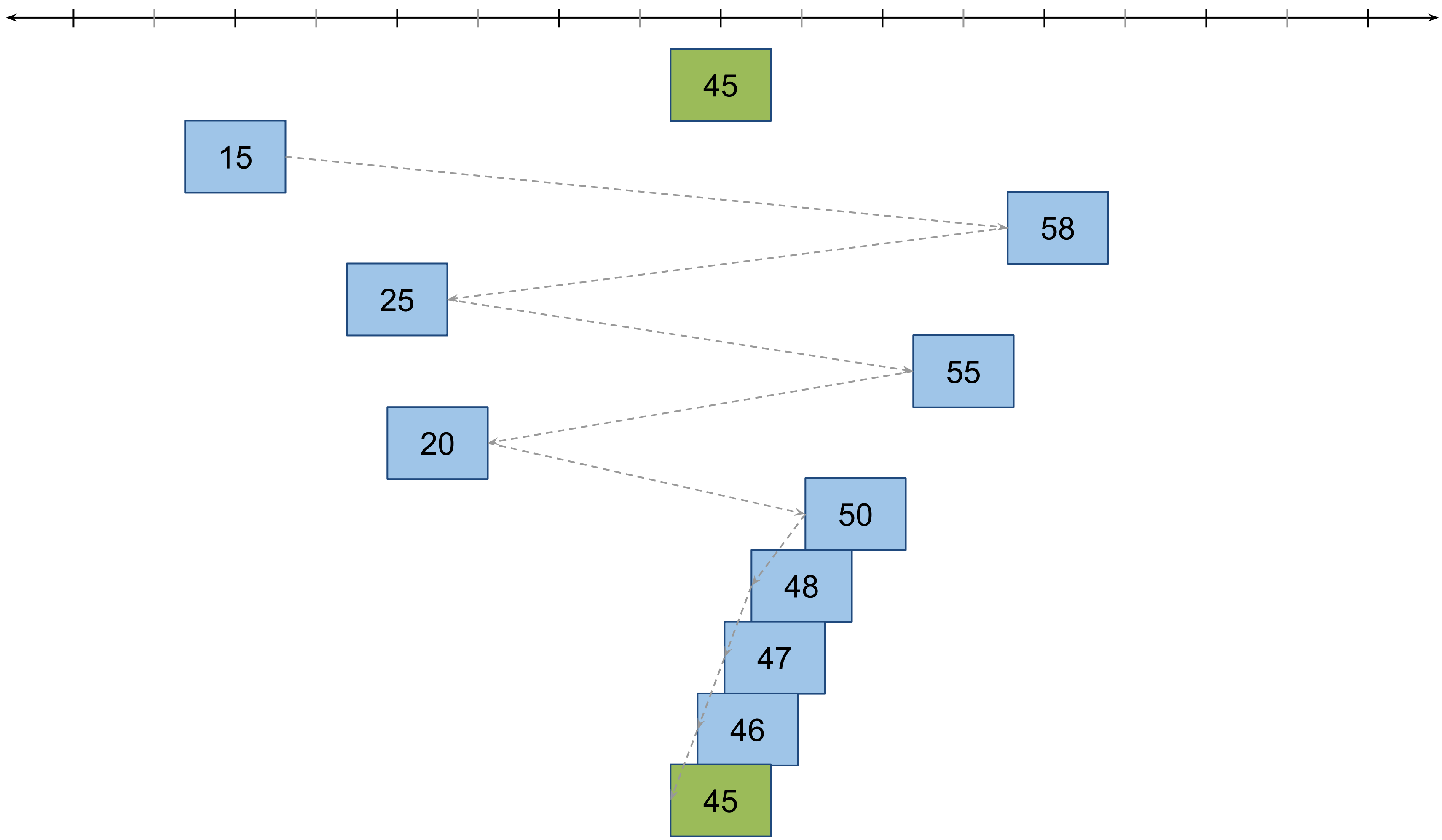
Find the value of x in the following equation:

$$x + 5 = ?$$

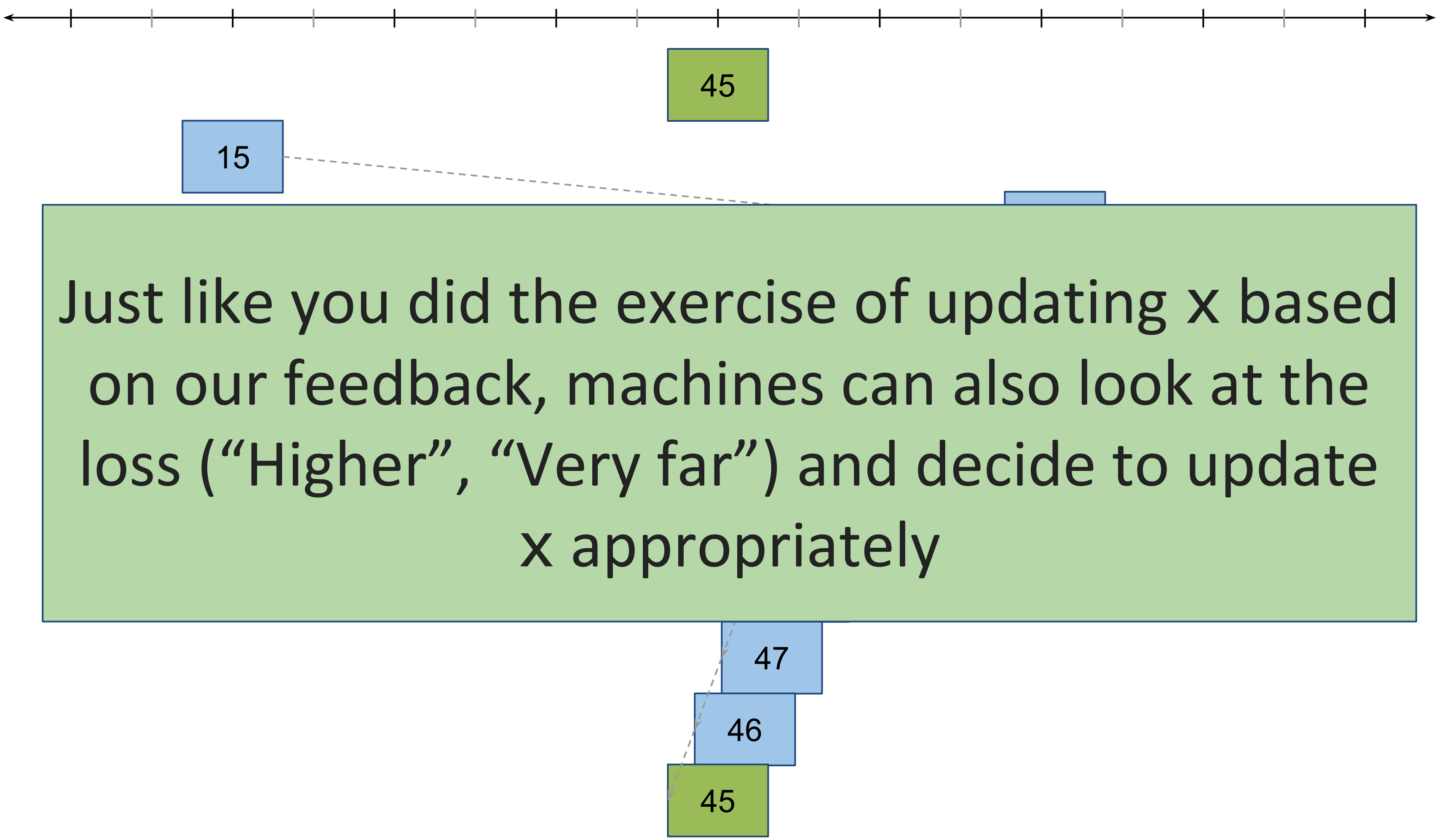
For every guess you will get the following hints:

Direction:	Higher	Lower		
Error:	Very far	Far	Close	Very close

Optimization Example

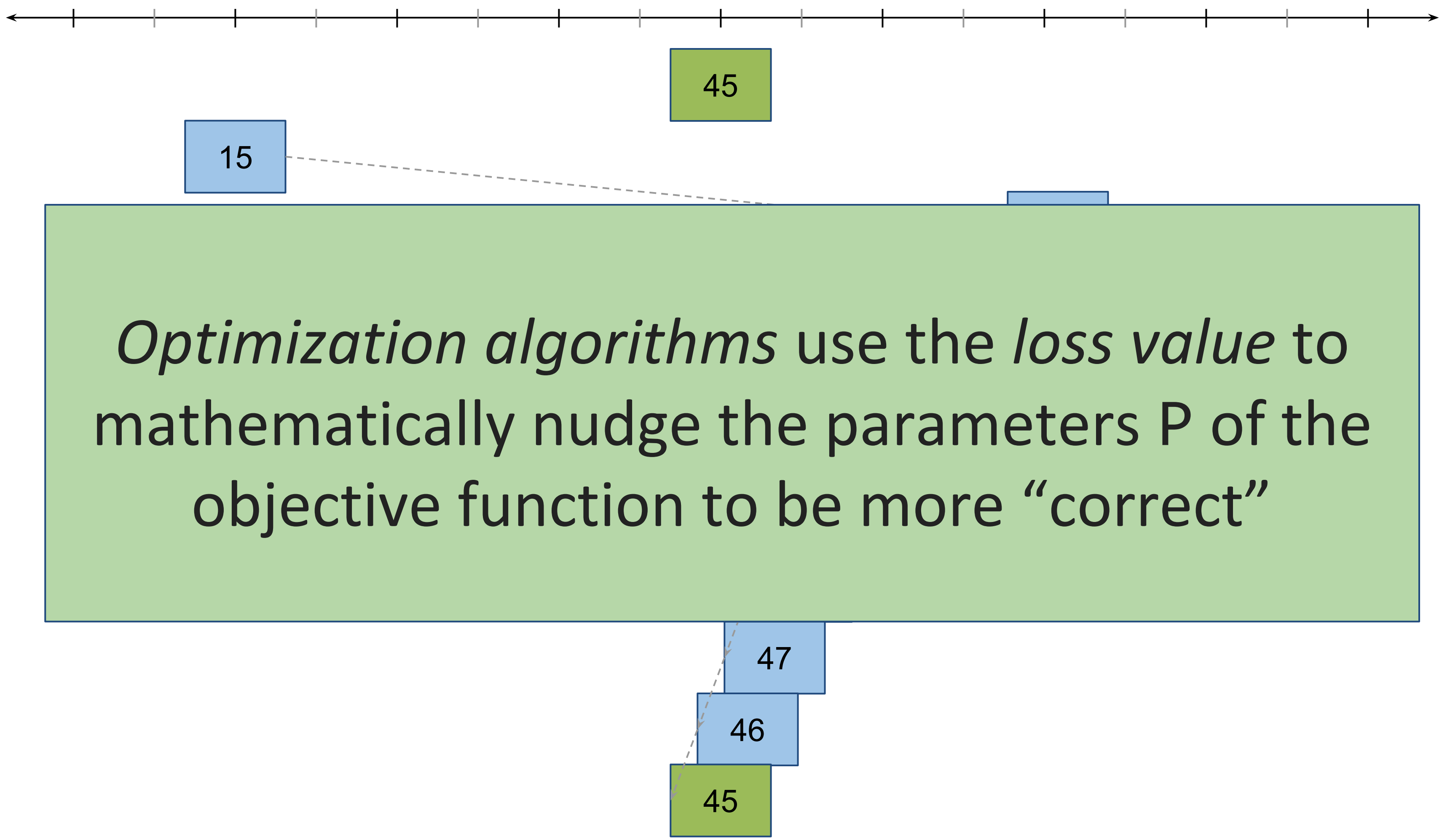


Optimization Example



Just like you did the exercise of updating x based on our feedback, machines can also look at the loss (“Higher”, “Very far”) and decide to update x appropriately

Optimization Example



Optimization algorithms use the *loss value* to mathematically nudge the parameters P of the objective function to be more “correct”

Optimization

What are some strategies you used to optimize x?

Optimization: Random Search

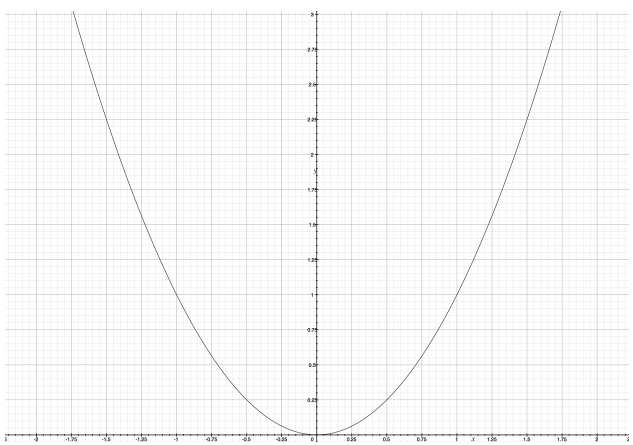
- Potential Solution: Guess randomly each time
- Pros:
  - Very simple
- Cons:
  - Not very efficient
  - *loss* value is unused
  - Potentially may never find a good solution

Optimization: Gradient Search

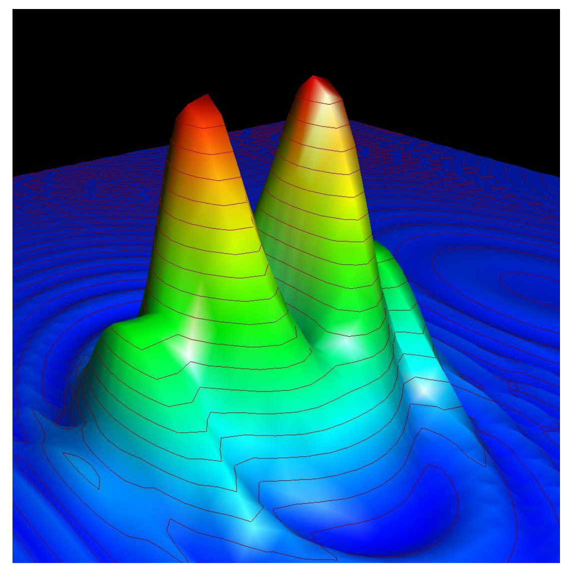
- Better Solution: Gradient based search

Optimization: Gradient Search

- Better Solution: Gradient based search
- Every function can be represented in space:



$$y = x^2$$

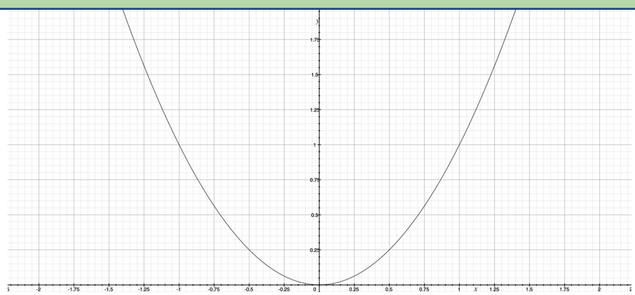


$$z = \frac{\sin(x^2 + 3y^2)}{0.1 + r^2} + (x^2 + 5y^2) \cdot \frac{\exp(1 - r^2)}{2}, r = \sqrt{x^2 + y^2}$$

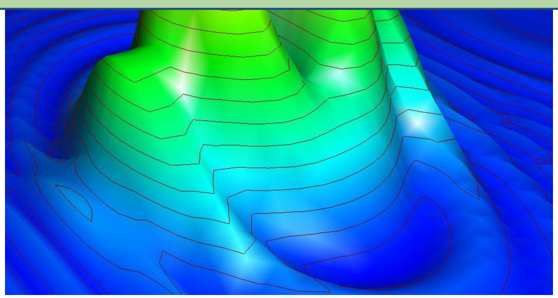
Optimization: Gradient Search

- Better Solution: Gradient based search
- Every function can be represented in space:

Any *loss* function can also be represented in space



$y = x^2$



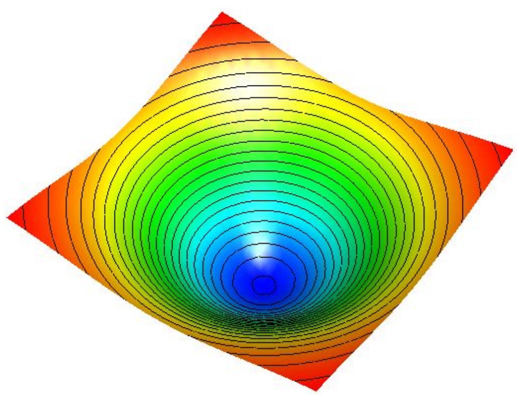
$$z = \frac{\sin(x^2 + 3y^2)}{0.1 + r^2} + (x^2 + 5y^2) \cdot \frac{\exp(1 - r^2)}{2}, r = \sqrt{x^2 + y^2}$$

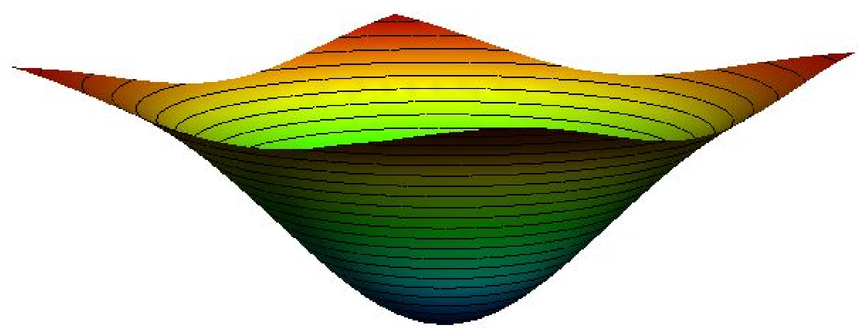
Optimization: Gradient Search

- Better Solution: Gradient based search
- Our goal is to minimize the loss, i.e. find a set of parameters P such that the loss is close to zero

Optimization: Gradient Search

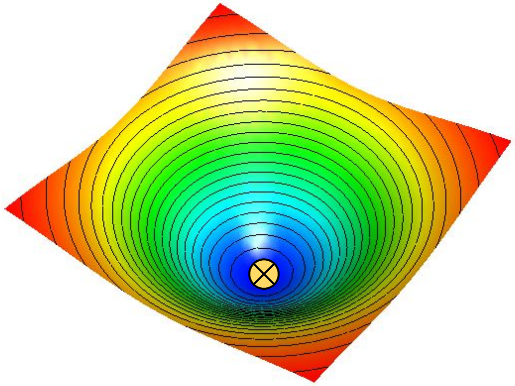
- Better Solution: Gradient based search
- Our goal is to minimize the loss, i.e. find a set of parameters P such that the loss is close to zero

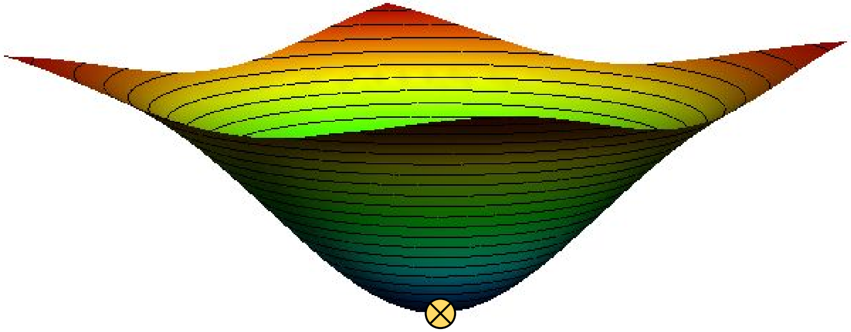




Optimization: Gradient Search

- Better Solution: Gradient based search
- Our goal is to minimize the loss, i.e. find a set of parameters P such that the loss is close to zero





⊗ Minimum value of the function

Optimization: Gradient Search

Functions are just like terrain - they have mountains and valleys

We want to minimize loss,

i.e. go to the bottom of the terrain

Optimization: Gradient Search

Q: Imagine you are blindfolded on a mountain, how will you go to the bottom?

Optimization: Gradient Search

Q: Imagine you are blindfolded on a mountain, how will you go to the bottom?

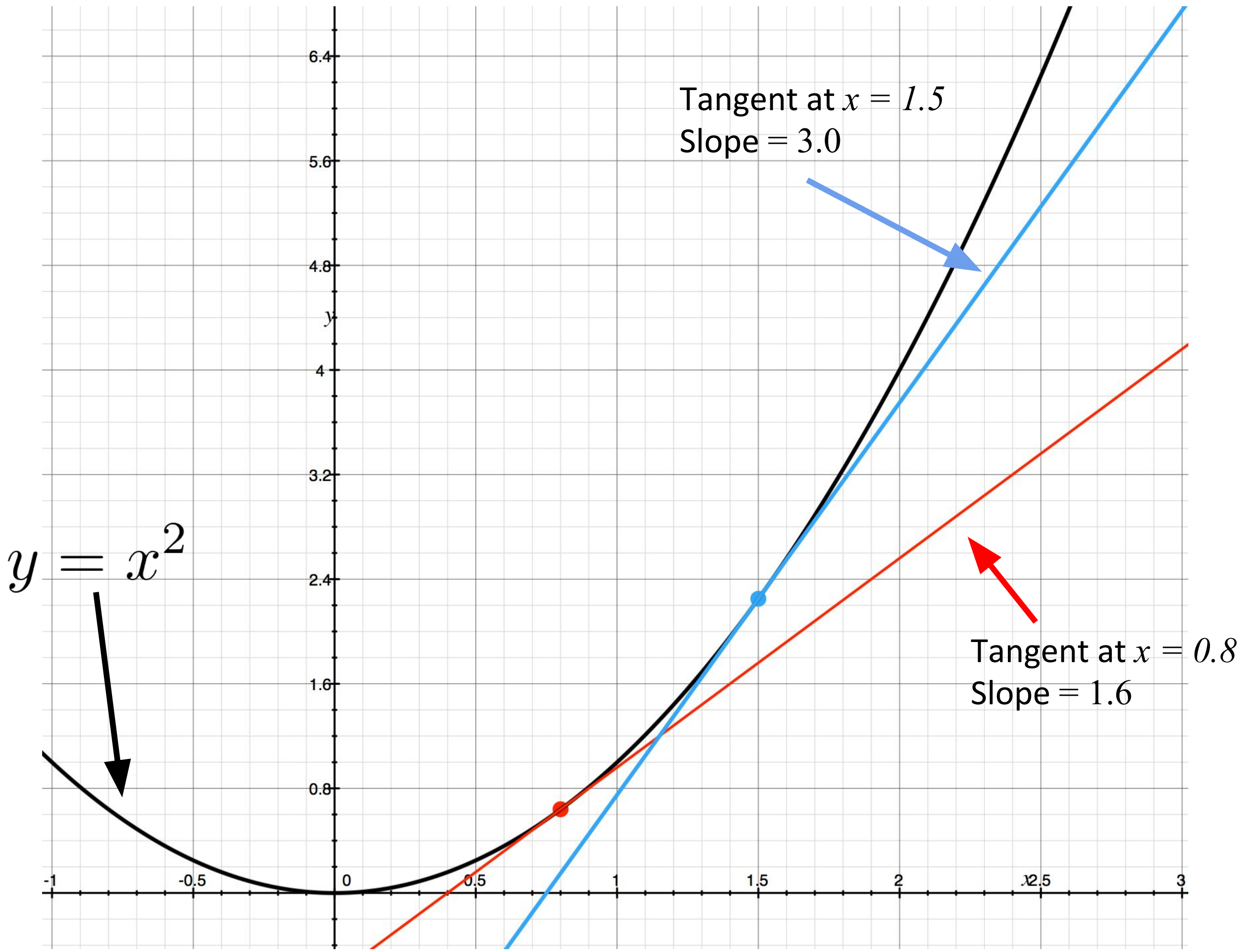
A: Sense the slope around you, and move in the direction where the slope points downwards

Optimization: Gradient Search

Concept of gradient == “your sense of slope” for the loss function

The gradient of a function is mathematically defined as the slope of the tangent i.e. slope at any given point on the function

Optimization: Gradient Search



Optimization: Gradient Search

Once we know the direction, we can move towards the minimum.

Are we done?

Optimization: Learning Rate

How far should we move?

The *step size* or *learning rate* defines how big a step we should take in the direction of the gradient

Optimization: Learning Rate

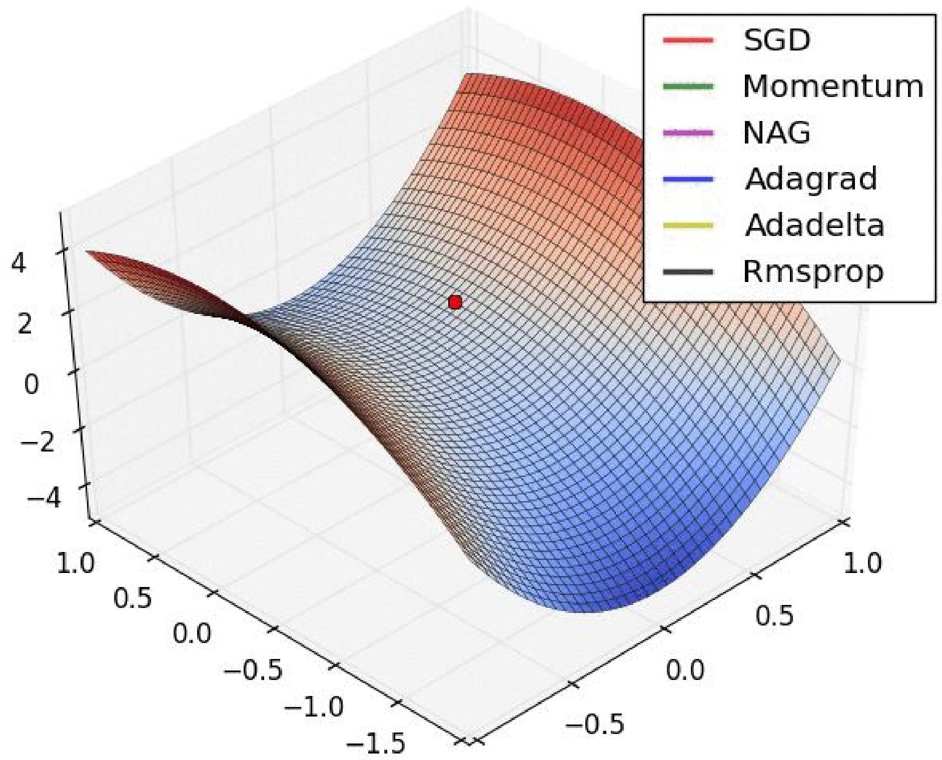
How far should we move?

The *step size* or *learning rate* defines how big a step we should take in the direction of the gradient

It must be well controlled - too small a step and it may take a long time to reach the bottom - too big a step and we may miss the minimum all together!

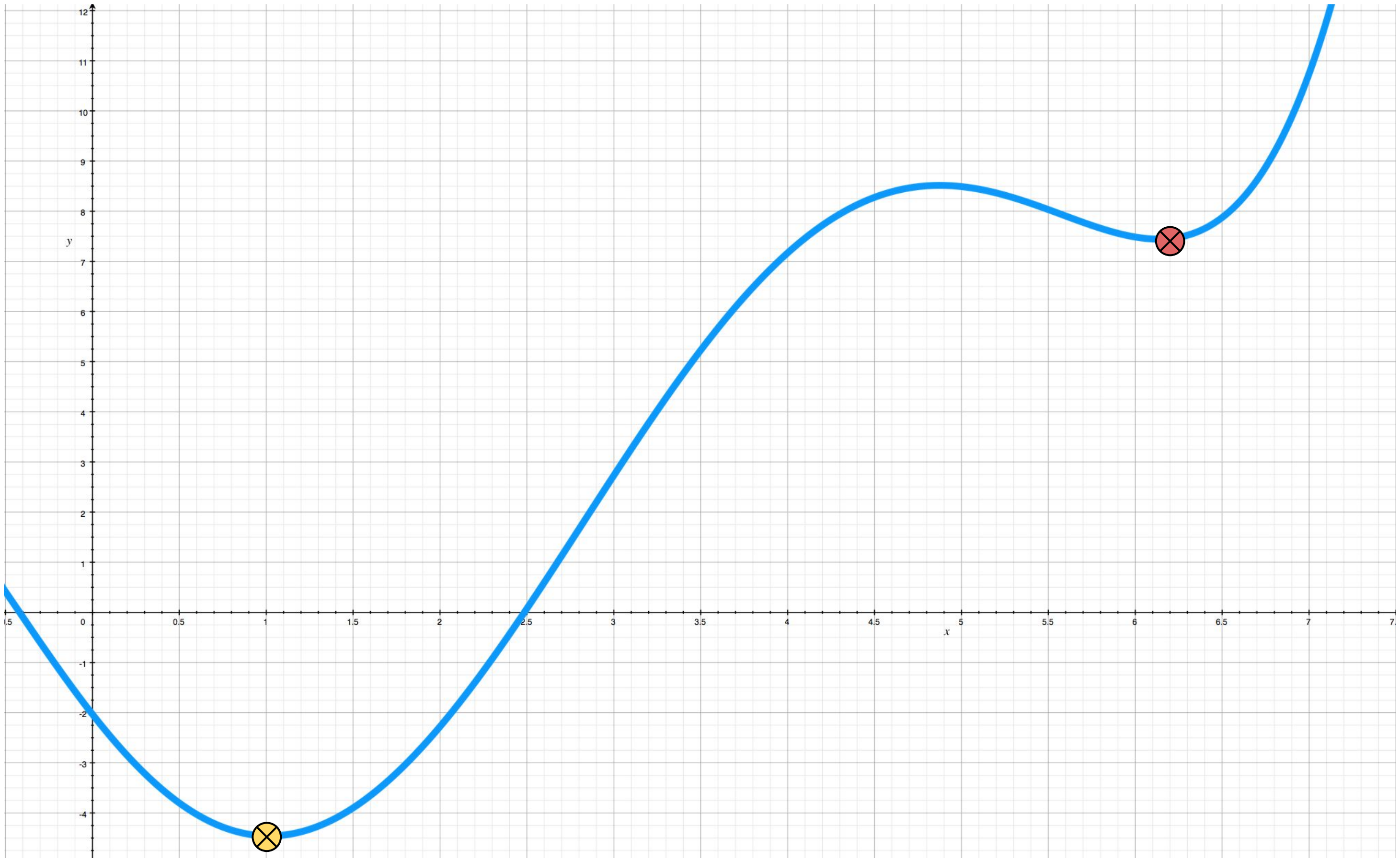
Optimization

Various optimization algorithms



Alec Radford ([Reddit])

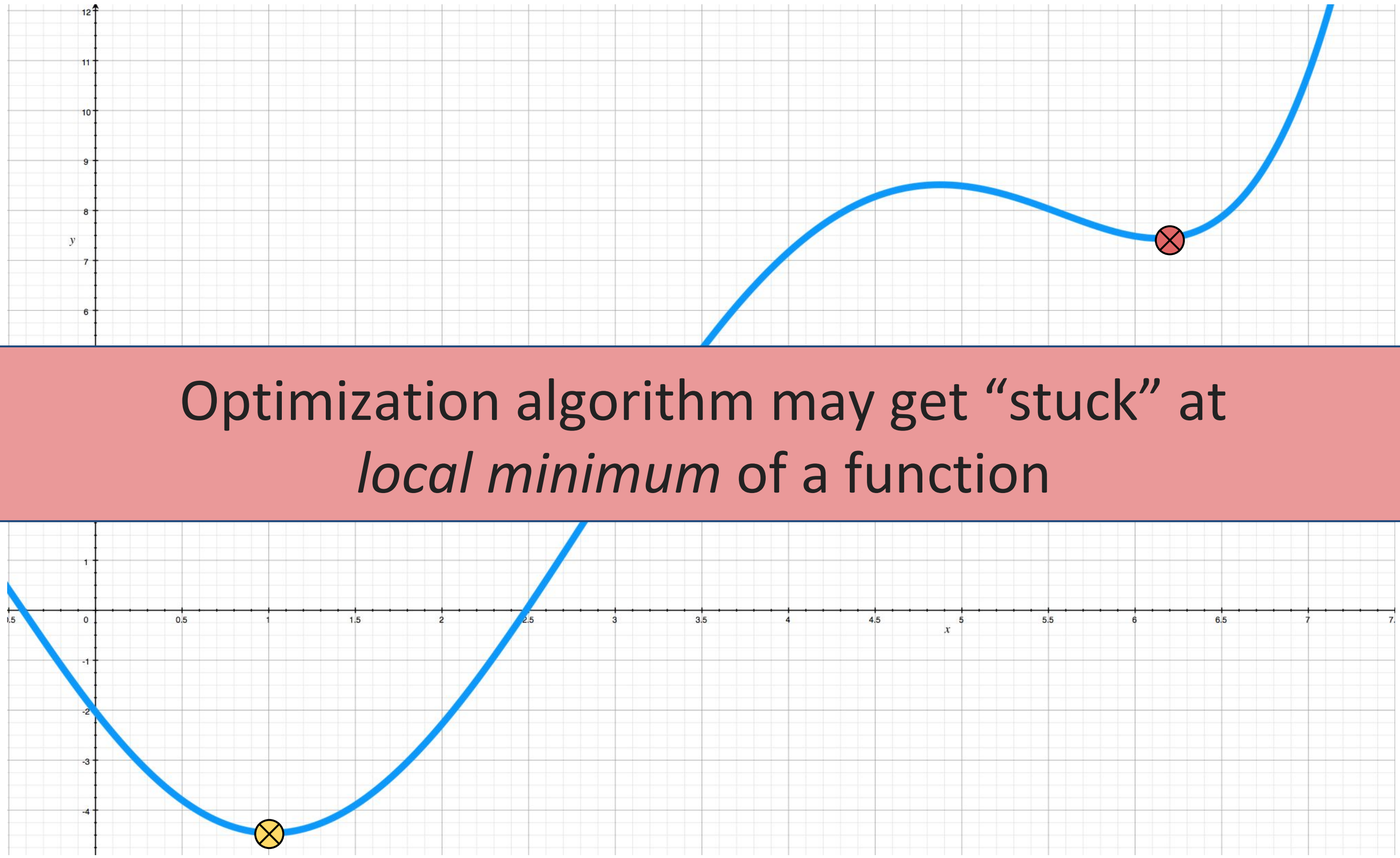
Local Minima



Minimum value of the function

Local minimum value of the function

Local Minima



Optimization algorithm may get “stuck” at
local minimum of a function

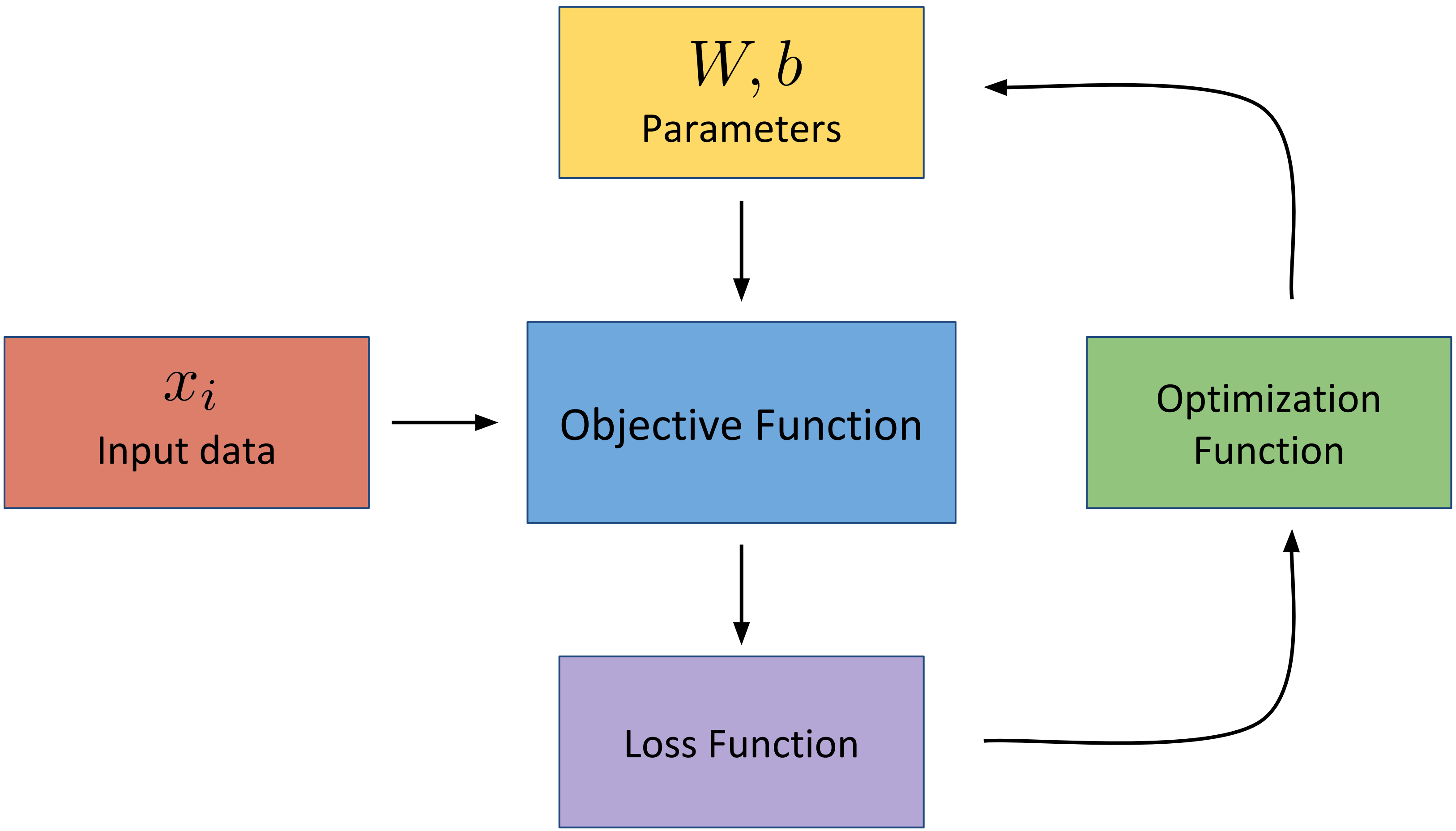
Minimum value of the function

Local minimum value of the function

Optimization: Demo

Let’s look at an actual optimization in real time!

Ingredients of a Classifier

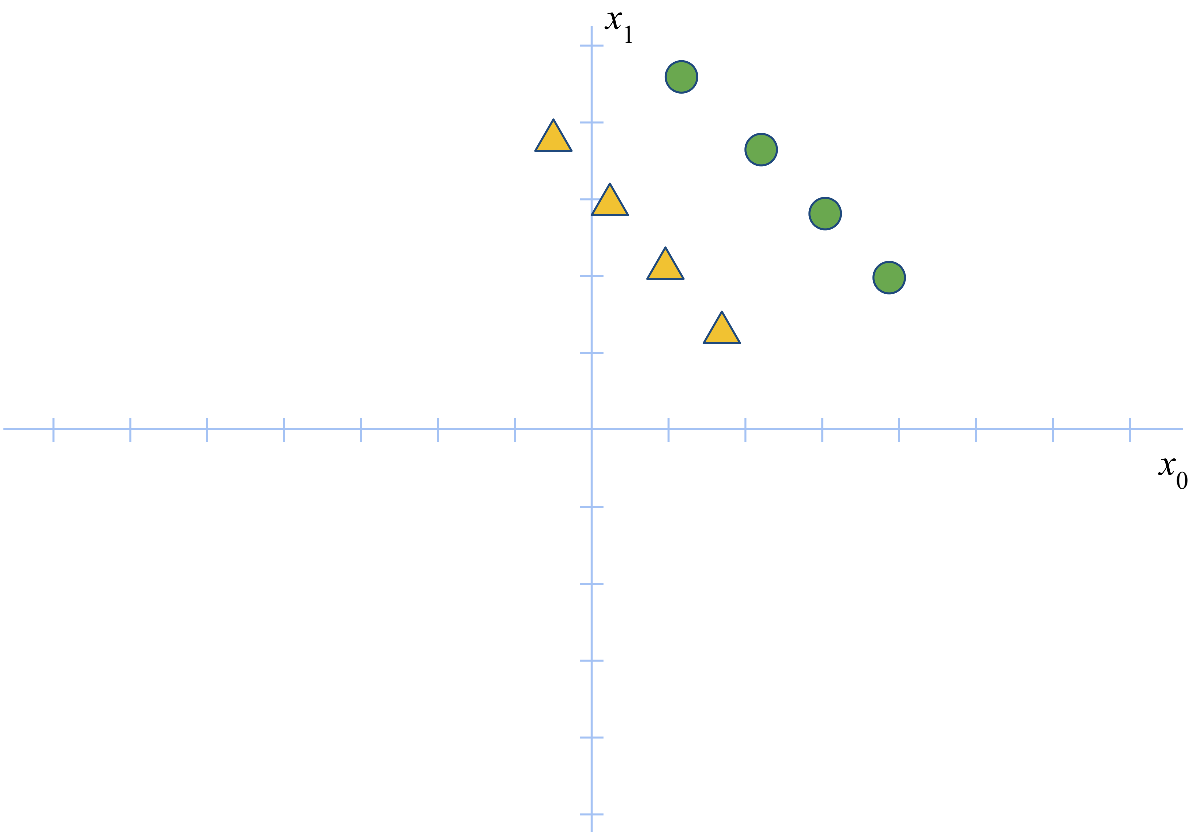


Bias

- What was the "b"?
  - A parameter that allows you to "shift" your decision boundary
  - In the case of a linear boundary (`f = Wx + b`), the W can only control the slope of the boundary - but that may not be enough

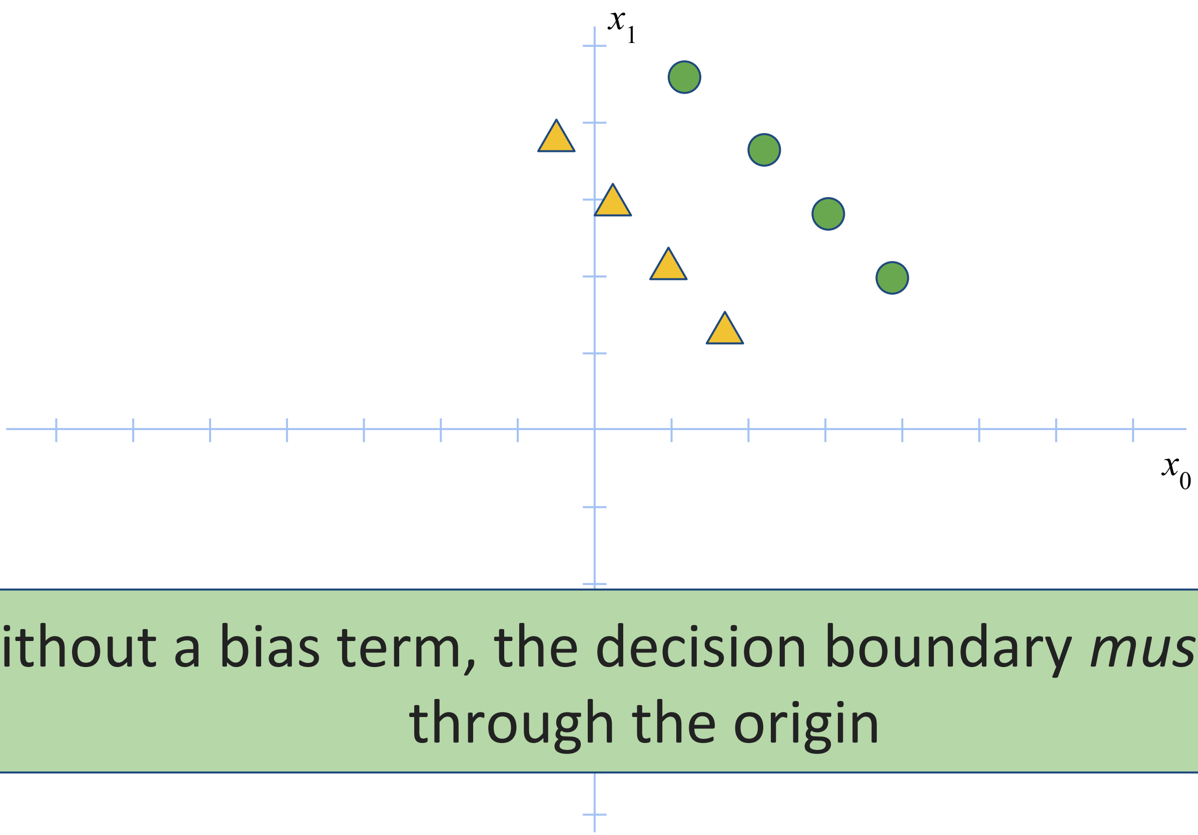
Bias

Consider the following dataset:



Bias

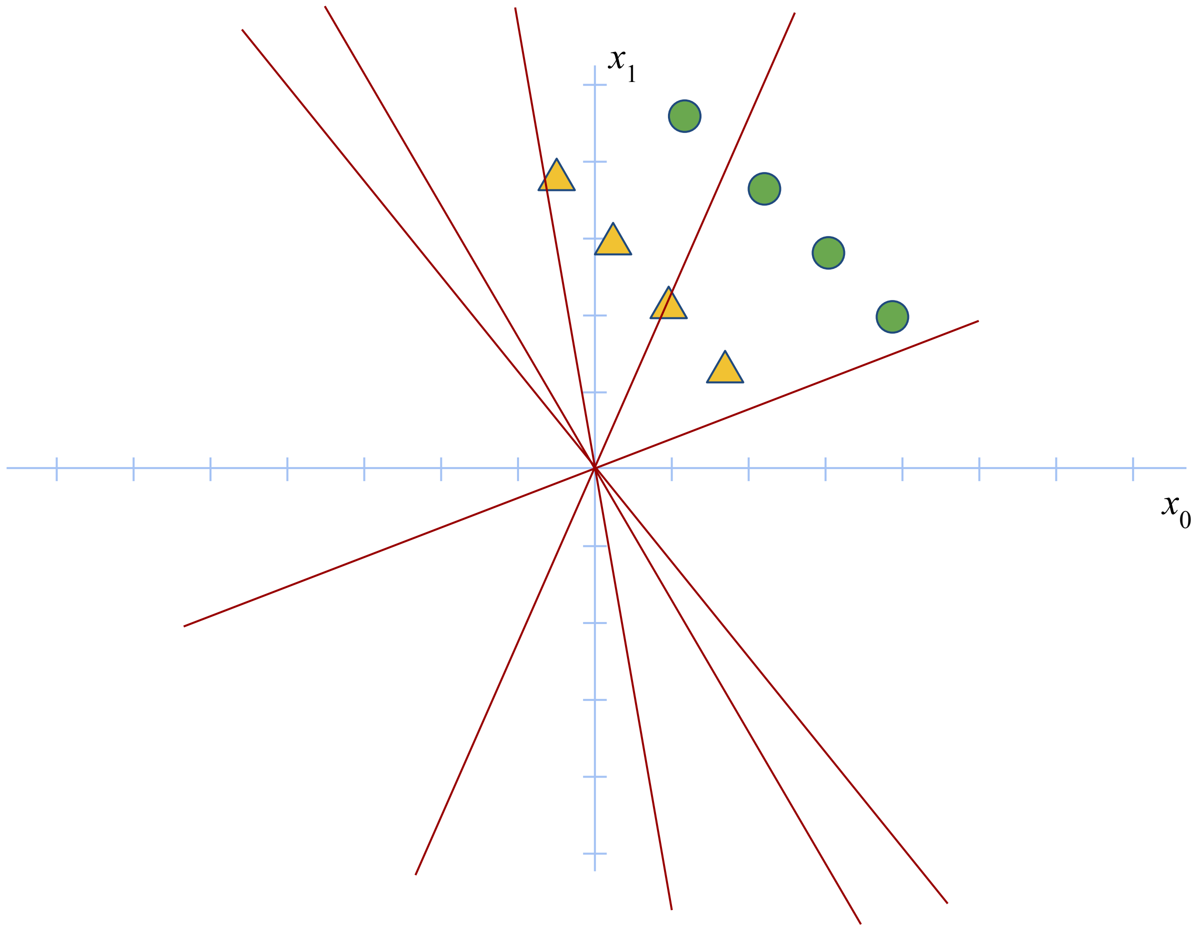
Consider the following dataset:



Without a bias term, the decision boundary *must* pass through the origin

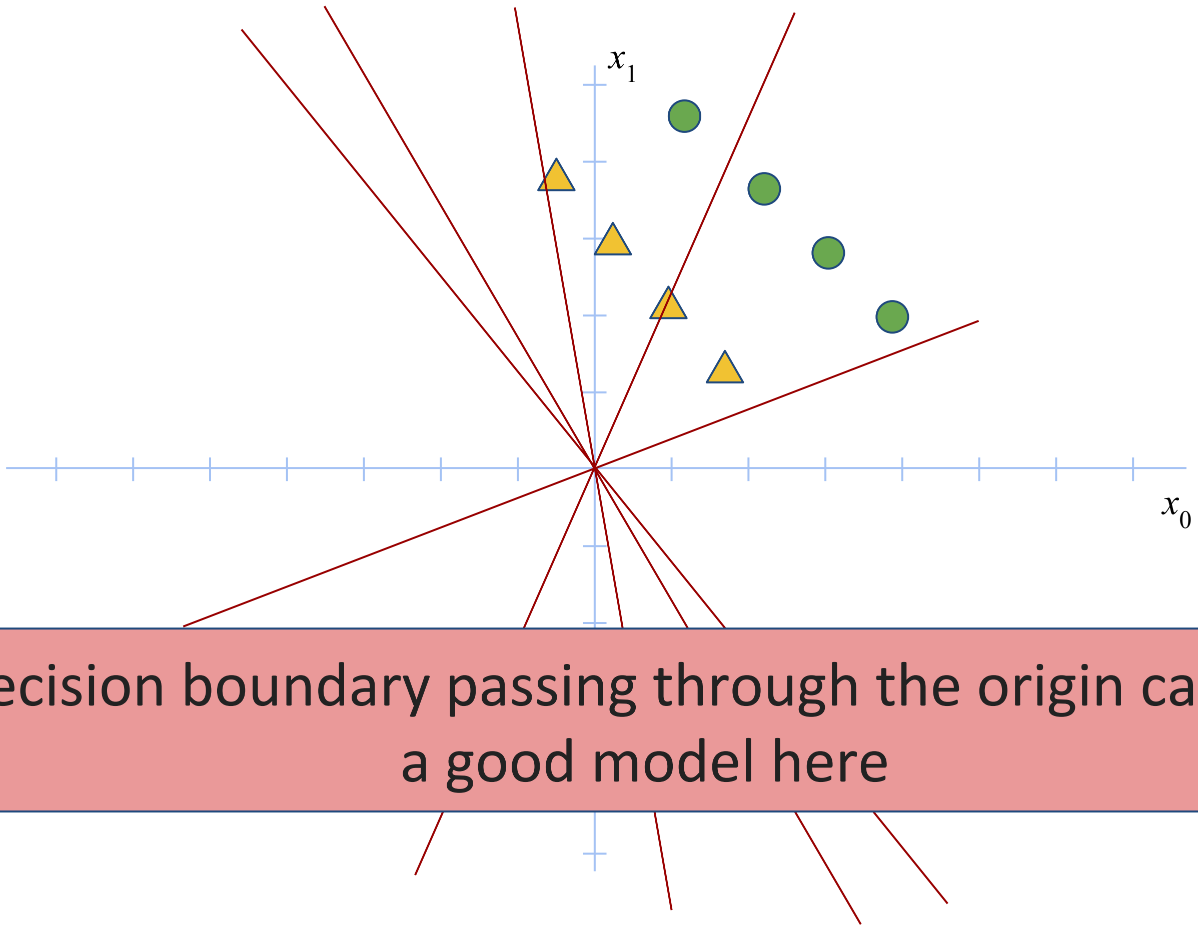
Bias

Consider the following dataset:



Bias

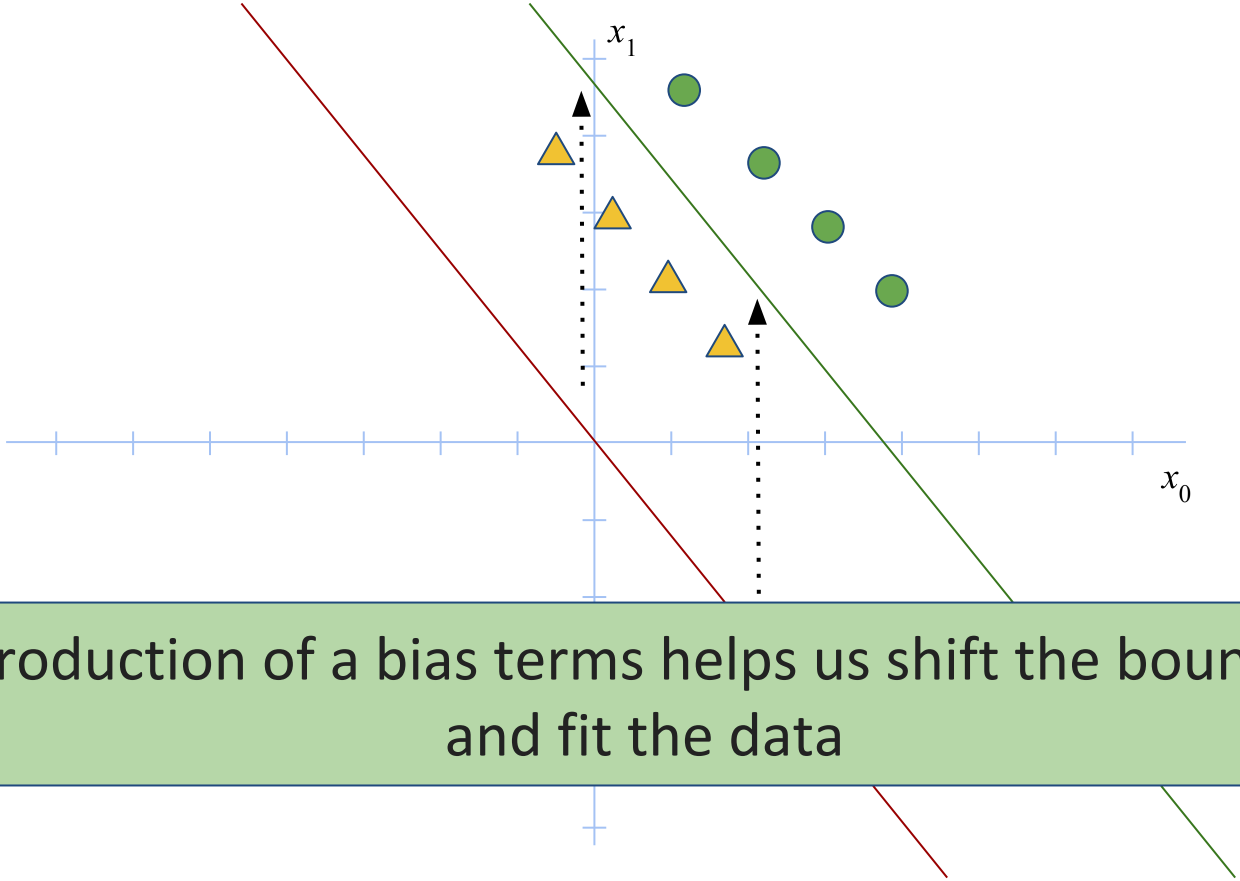
Consider the following dataset:



No decision boundary passing through the origin can lead to a good model here

Bias

Consider the following dataset:



Introduction of a bias terms helps us shift the boundary and fit the data

Parameter Initialization

- We’ve learned that we need to move in the direction of the gradient to reach the minimum value for a given loss function
- But where do we start?

Parameter Initialization

- Initial values of W and b dictate where in the terrain we begin
- If we start near a minima, we can optimize very quickly - If we start too far, it may take a long time to find a good model
- We may even start near a local minimum and never find the global minimum for a given function

Parameter Initialization

- Zero initialization?
- Random initialization?
- Something more complicated?

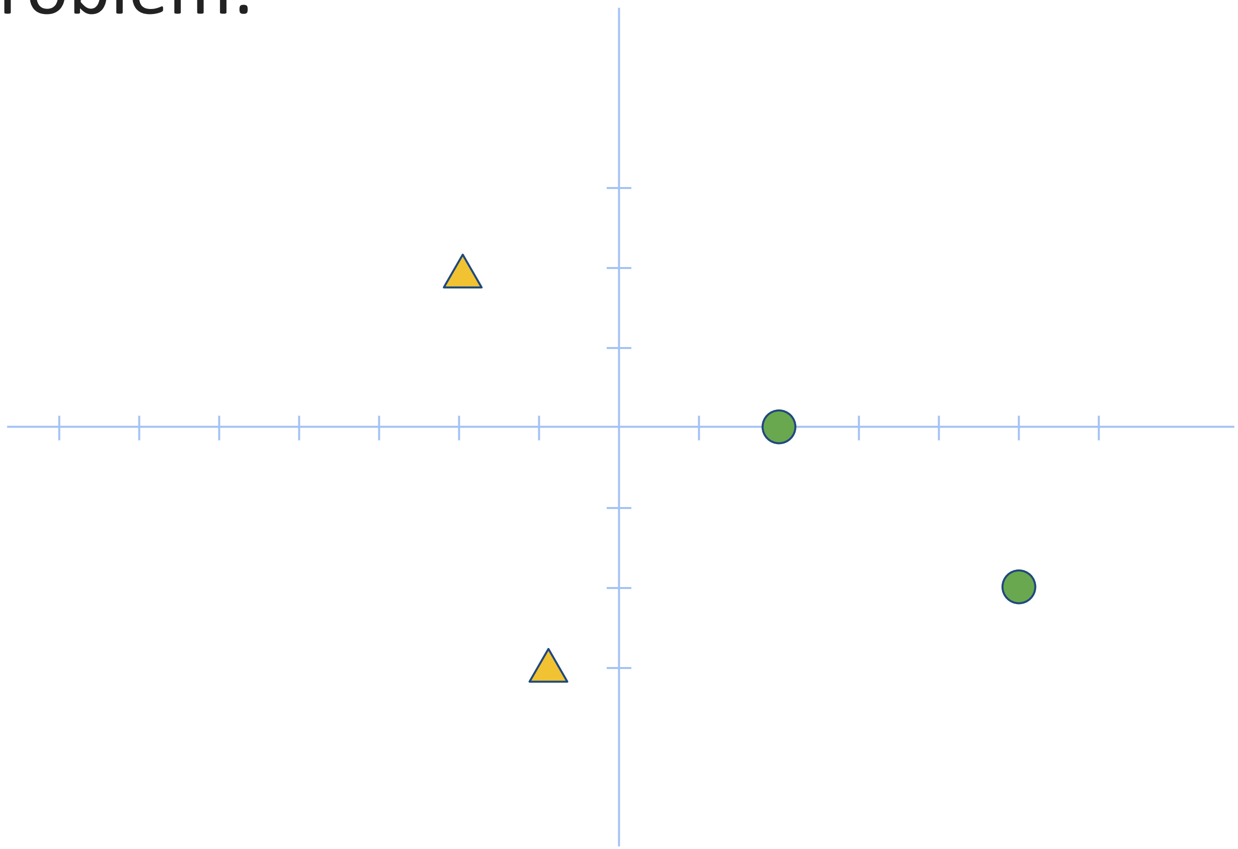
Parameter Initialization

- Zero initialization
- Random initialization
- Something more complicated:
  - Gaussian distributed
  - Xavier Initialization

More on this later!

Regularization

As we’ve seen, there are many potential solutions to a problem:



Regularization

As we've seen, there are many potential solutions to a problem:

$P_1: w_0 = 3; w_1 = 1; b = 3$

$P_2: w_0 = 300; w_1 = 100; b = 300$

$P_3: w_0 = 300; w_1 = 99; b = 300$

Which set of parameters is better here?

Regularization

Some loss functions are sensitive to the magnitude of weights:

```
Average losses (MSE)
```

$L(P_1) = 73.25$ $L(P_2) = 866051.0$ $L(P_3) = 867304.75$

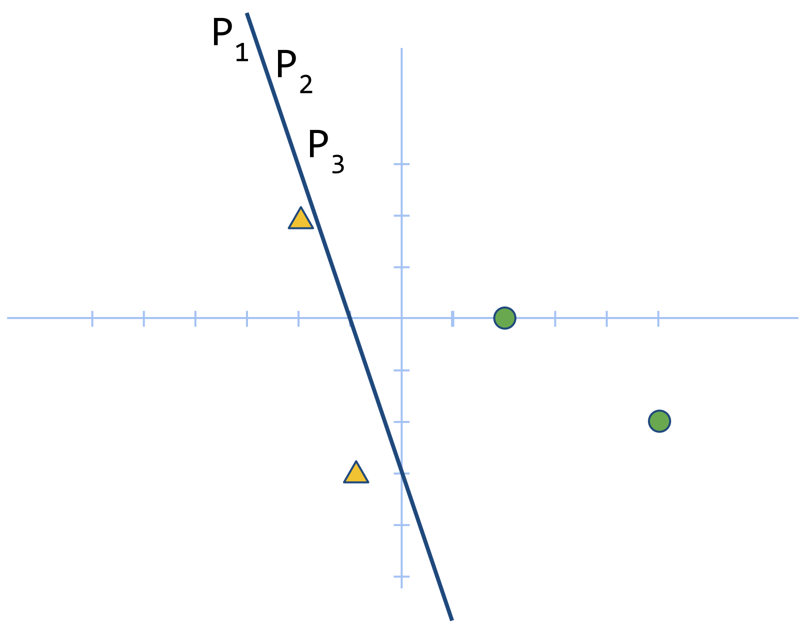
Regularization

Some loss functions are sensitive to the magnitude of weights:

Average losses (MSE)

$L(P_1) = 73.25$ $L(P_2) = 866051.0$ $L(P_3) = 867304.75$

But all three represent almost exactly the same boundary!



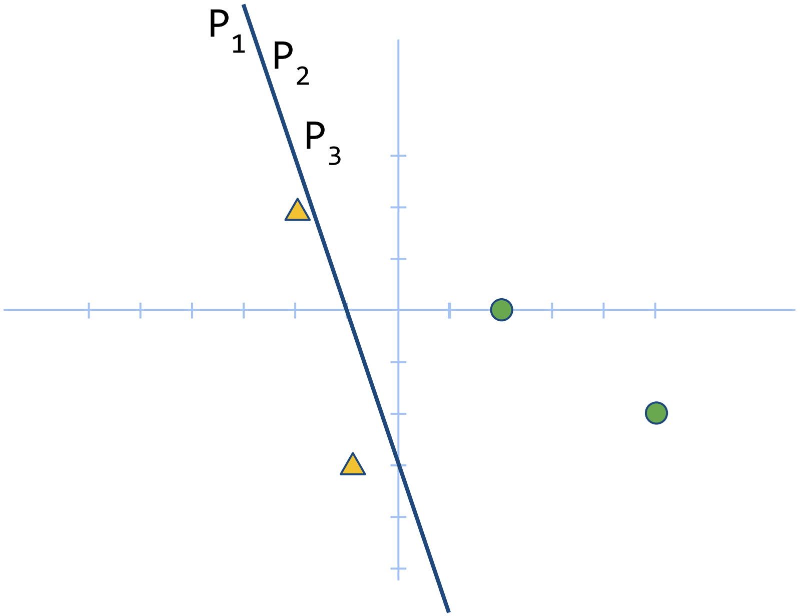
Regularization

Some loss functions are sensitive to the magnitude of weights:

```
Average losses (MSE)
```

$L(P_1) = 73.25$ $L(P_2) = 866051.0$ $L(P_3) = 867304.75$

Loss is different for each set of parameters, even though conceptually they are all equally good



Regularization

Solution: Since all these solutions are equally good, constrain our model to weights with small magnitude

Regularization

Solution: Since all these solutions are equally good, constrain our model to weights with small magnitude

$$L = \texttt{Normal loss} + \lambda \sum_{w} w_i^2$$

Penalizes weights that are too large
λ defines how much importance you want to give to regularization

Summary

- Classification - supervised vs. unsupervised
- Linear regression
- Objective function
- Loss function
  - sum of absolute differences
  - mean squared error
- Optimization
  - random search
  - gradient search