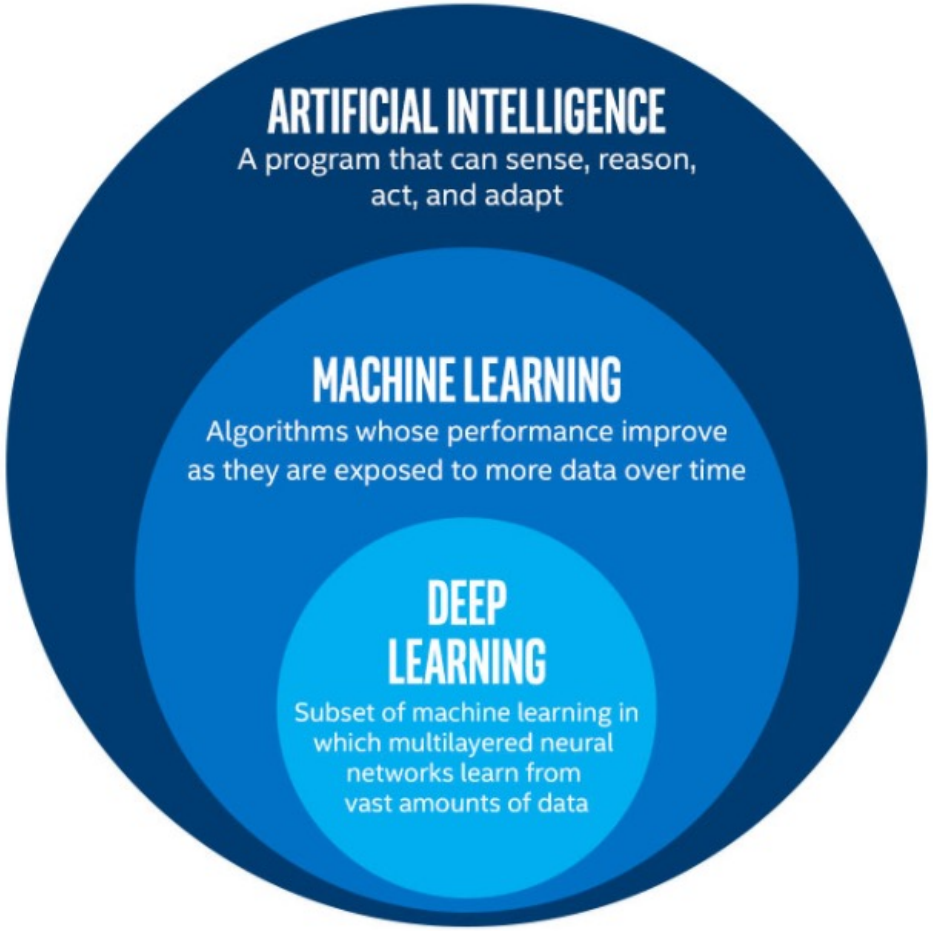
INTRODUCTION TO MACHINE LEARNING

Nipuna Senanayake

APPLICATIONS

AI AND MACHINE



AI but Not ML:

- Searching / Path finding Algorithms / Robotics
- Ontology Engineering / Semantic Web
- Prepositional / First order logic

ML but not Deep Learning

- Regression
- Decision Tree
- SVM
- K-means
- KNN
- Neural networks (Few layers)

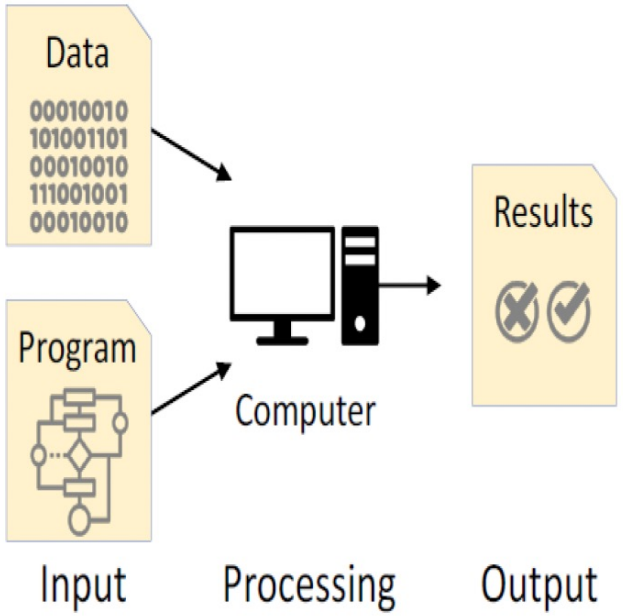
Deep Learning

- Deep NN
- CNN
- RNN
- LSTM

ML VS TRADITIONAL PROGRAMMING

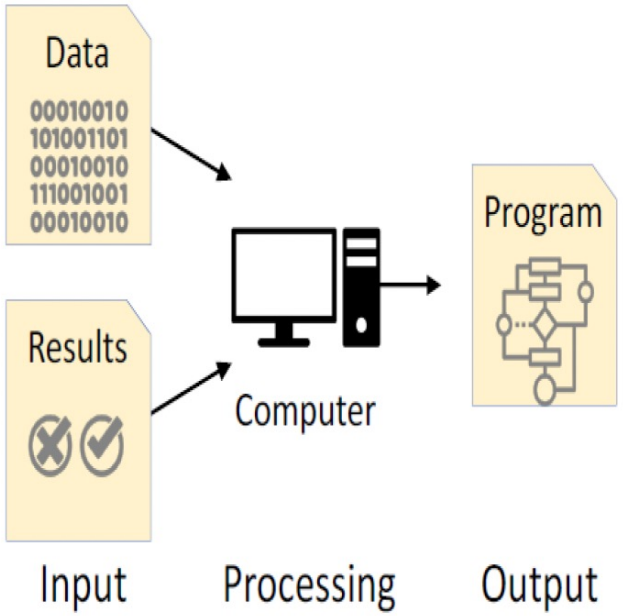
Traditional Programming

Works well when we know how to specify the program

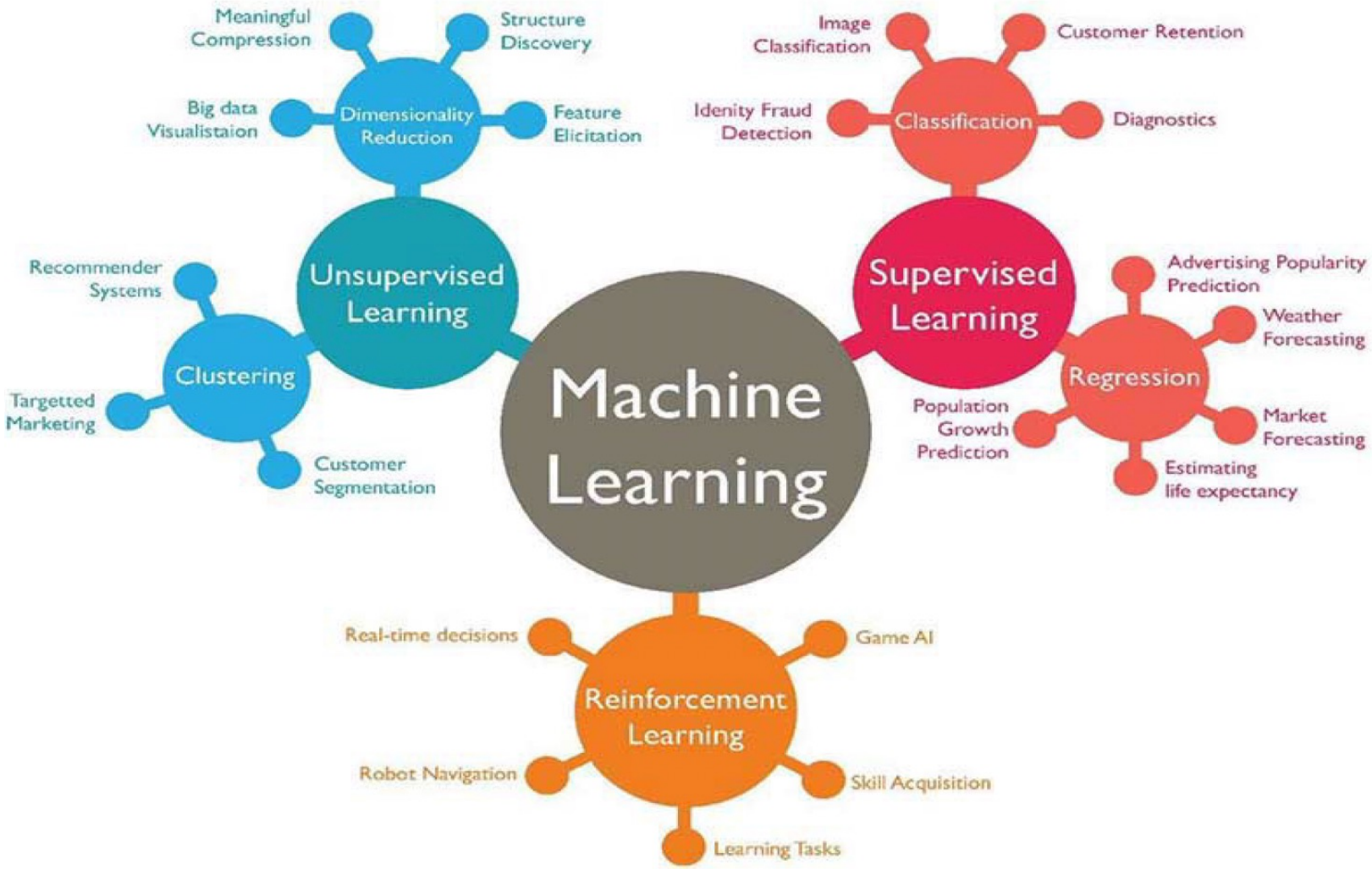


Machine Learning

Needed when we don't know how to specify the program



TYPES OF ML


Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Market Forecasting
Estimating life expectancy
Population Growth Prediction
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

KEY TERMS

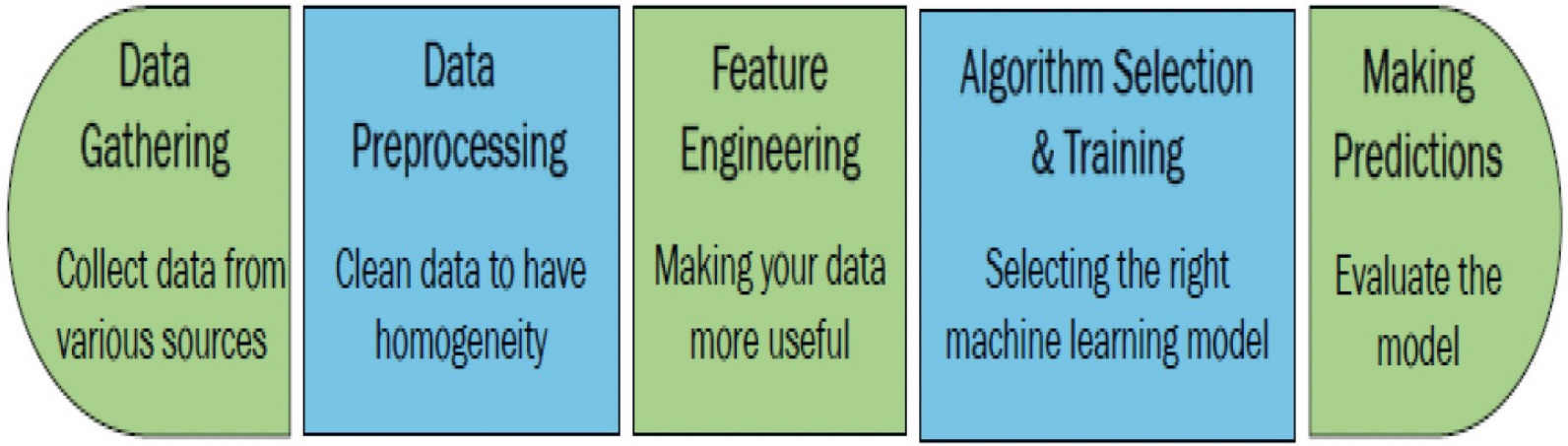
- **Supervised Learning:** Use of labeled dataset for training algorithms so that they can classify data or predict the outcome.
- **Label:** What we are predicting. The "y" variable in simple linear regression. The label could be the future price of wheat, the kind of animal shown in a picture, the meaning of an audio clip, or just about anything.
- **Feature:** is an input variable—the x variable in simple linear regression. A simple machine learning project might use a single feature, while a more sophisticated machine learning project could use millions of features
- **Models:** A model defines the relationship between features and label. For example, a spam detection model might associate certain features strongly with "spam".

TWO PHASES OF LIFE (OF A MODEL!)

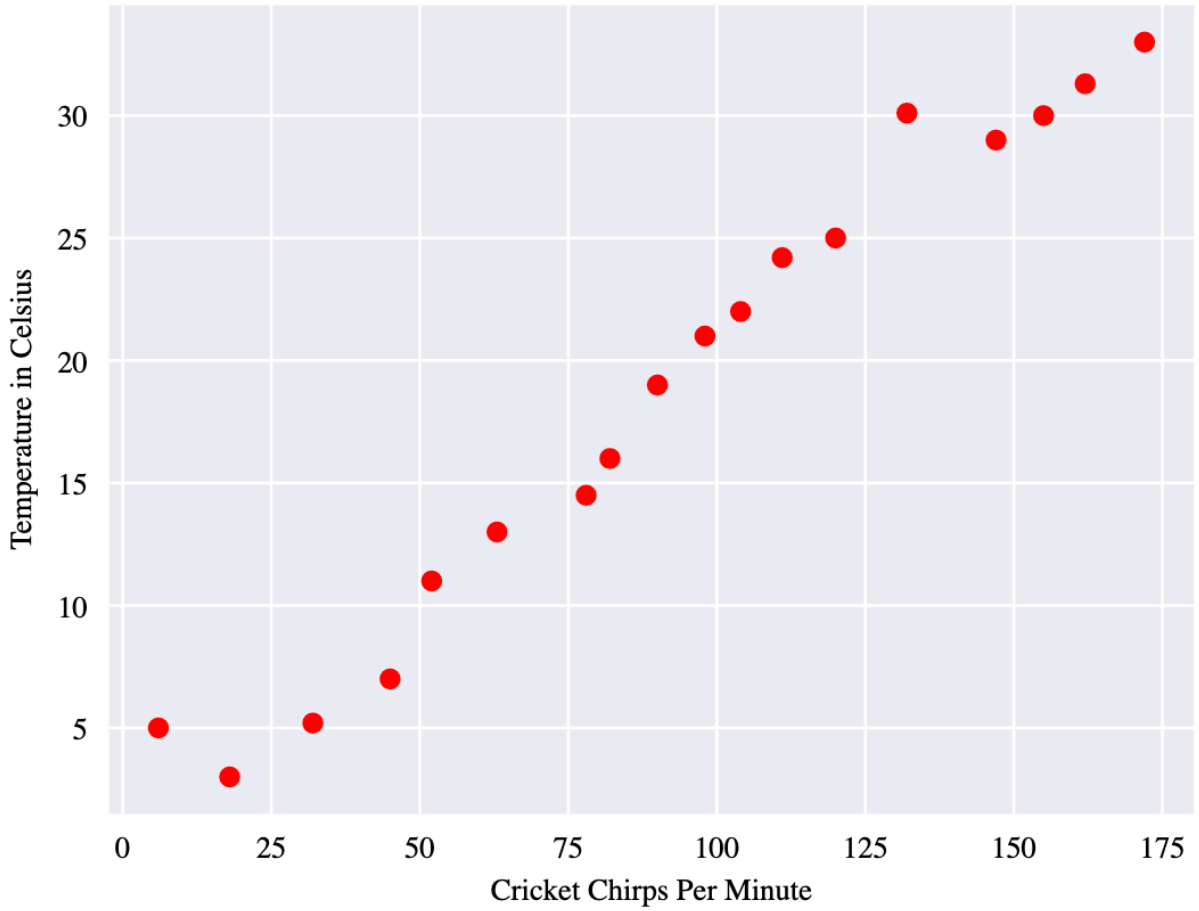
- **Training:**
  - Creating or learning the model.
  - Showing model labeled examples and enable it to gradually learn the relationships between features and the label

- **Inference:**
  - Applying the trained model to unlabeled examples. (Using trained model for make useful predictions)
  - *Classification:* Predicts discrete values (Is a given email message spam or not spam?)
  - *Prediction* (Regression): Predicts contiguous values (What is the value of a house in California?)

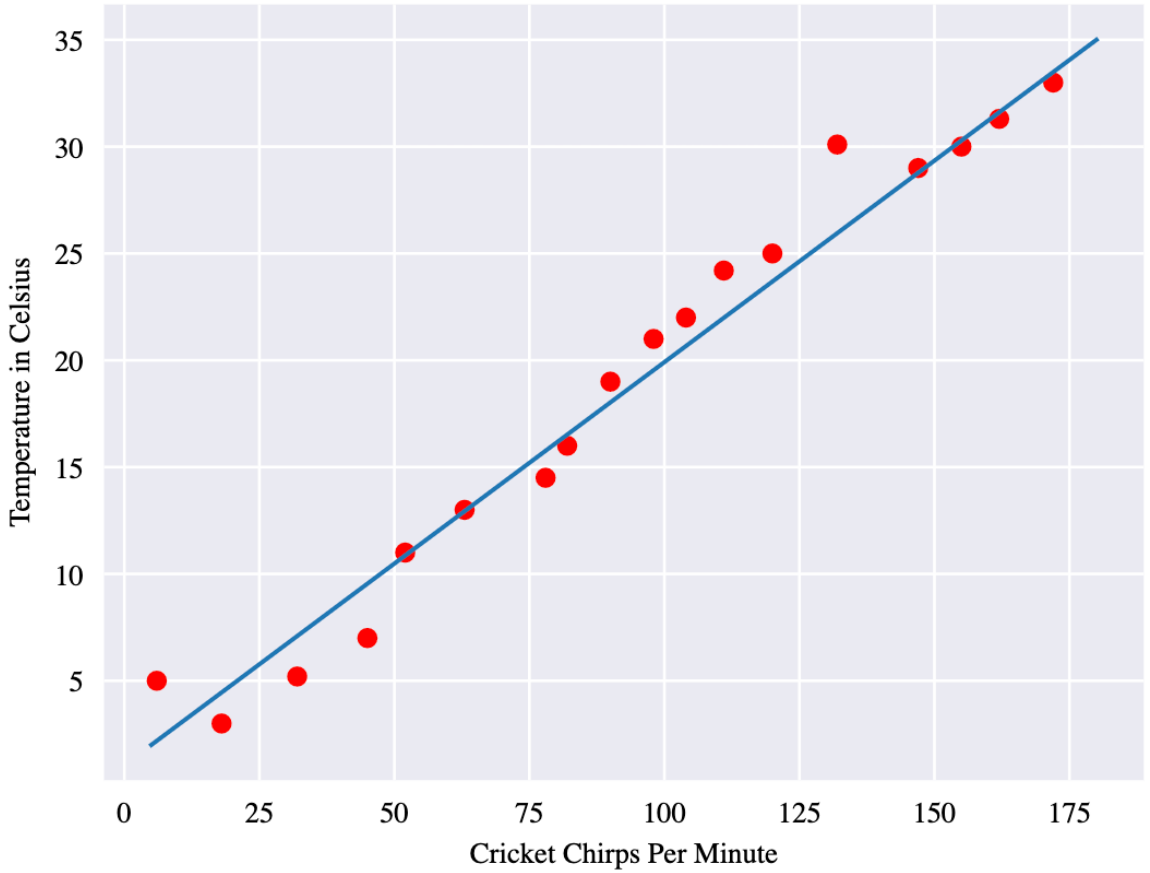
STEPS OF ML WORKFLOW



FROM STATISTICS TO COMPUTING



LINEAR REGRESSION

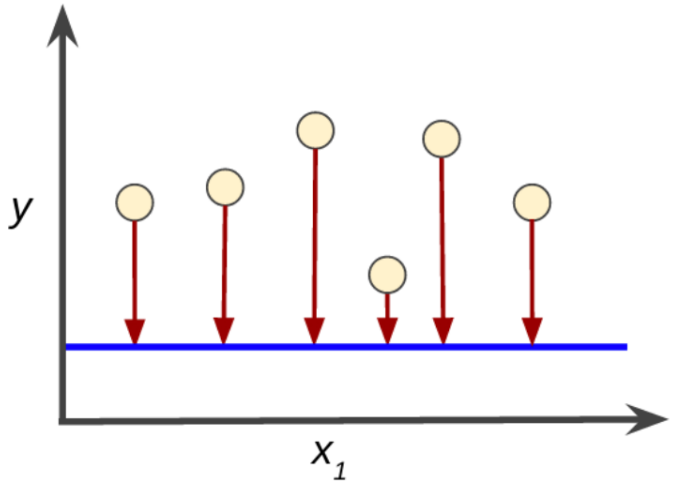


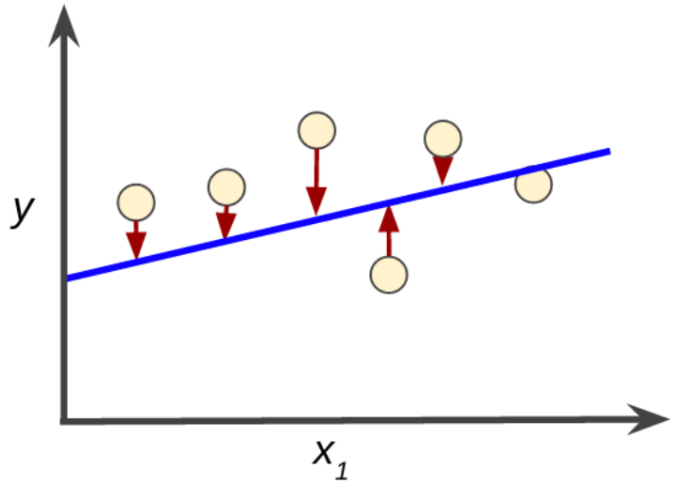
Stat/Math World	Computing / ML World
$y = mx + b$	$y' = b + w_1x_1$
y = the temperature in Celsius the value we're trying to predict.	y' = the predicted label (a desired output).
b = the y-intercept.	b = is the bias (the y-intercept), sometimes referred to as w0.
m = the slope of the line. x = the number of chirps per minute, the value of our input feature.	w_1 = the weight of feature 1. Weight is the same concept as the "slope" m in the traditional equation of a line. x_1 = a feature (a known input).

$y' = b + w_1x_1 + w_2x_2 + w_3x_3 + \cdots$ (A complex model with multiple features)

EMPIRICAL RISK MINIMIZATION

- New definition for **"Training":** learning/finding good values for weights and bias from labeled examples.
- **Loss:**
  - Penalty for wrong prediction.
  - Perfect prediction -> Zero loss, otherwise greater loss
  - Goal-> identifying set of weights and biases that have lower loss





SQUARED LOSS AND MSE FOR REGRESSION

- Squared Error (L2 loss): $L = (y - y')^2$

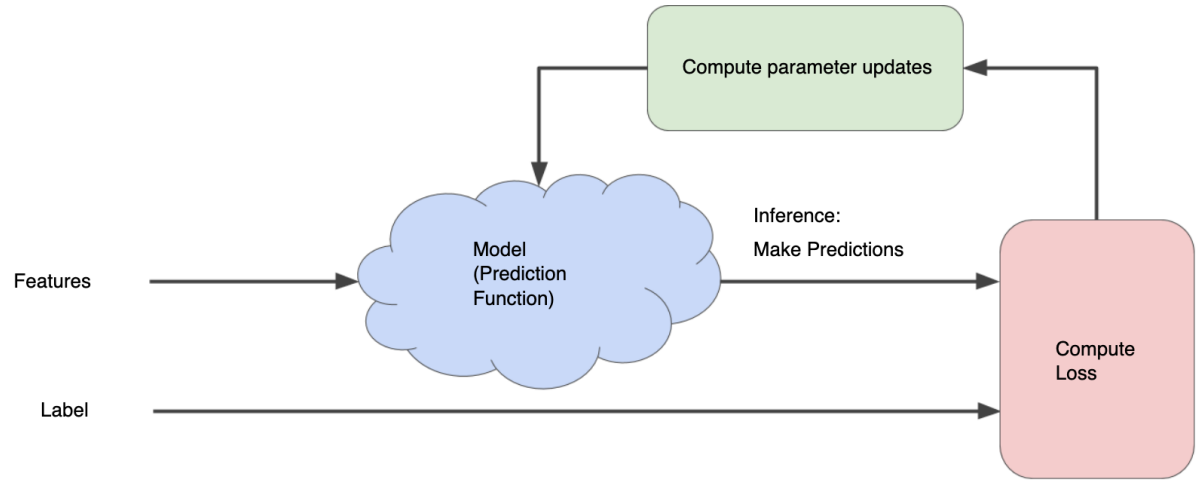
- Mean Squared Error (MSE)

$$MSE = \frac{1}{N} \sum_{(x,y) \in D} (y - prediction(x))^2$$

MSE is a common error function in ML but probably not the most practical / best loss function in all situations!!

FROM STATISTICAL APPROACH TO A COMPUTING APPROACH

- Hot and Cold game:
  - Start with a wild guess ($w_1 = 1$)
  - Wait for the system to tell the loss
  - Make another guess ($w_1 = 0.5$), what is the loss now?
  - Getting warmer or cold?
  - This goes on and on…

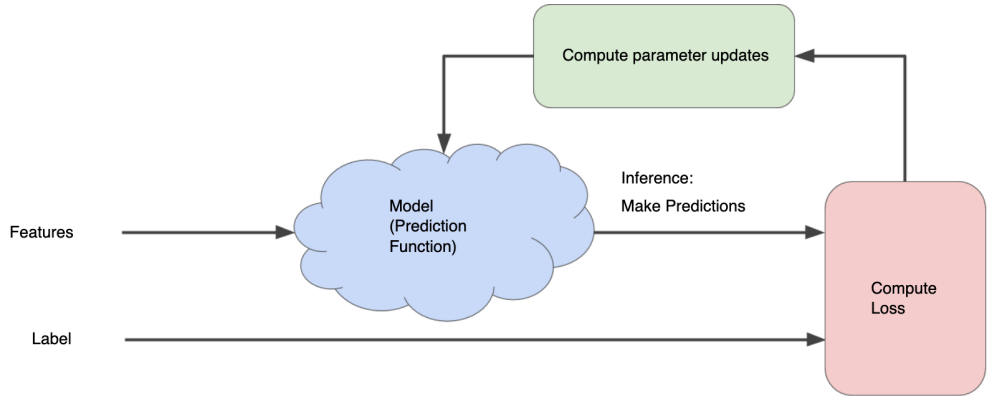


- An iterative process to reach the best model

TRAINING A MODEL

- Simple Regression model with one feature

$$y' = b + w_1 x_1$$



- Random values for wight and bias!

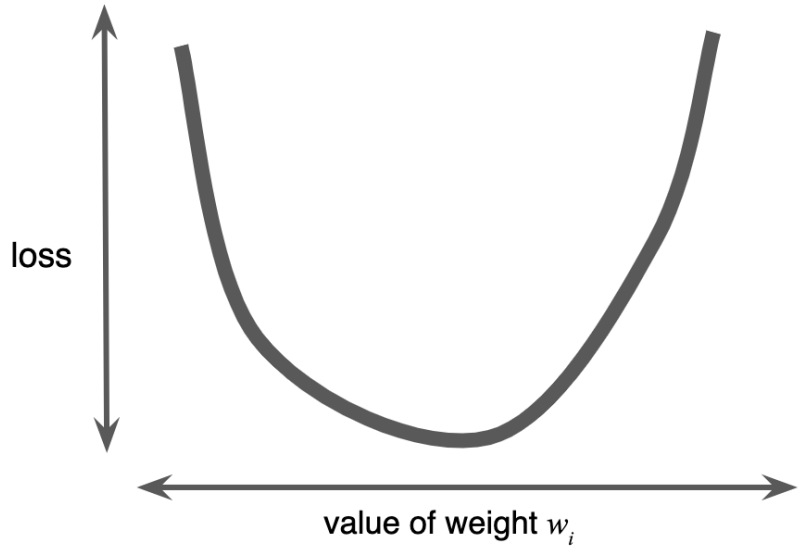
$$b = 0$$

$$w_1 = 0$$

- Plugging them into prediction function: $y' = 0 + 0 \cdot 10 = 0$
- Then use the loss function to calculate the loss. (Ex: the squared loss)
- For now, let's say: based on the loss value, system will calculate new b *and* w_1
- This continues till the lowest loss is reached (loss stops changing / starts to change extremely slow)

REDUCING THE LOSS WITH GRADIENT DECENT

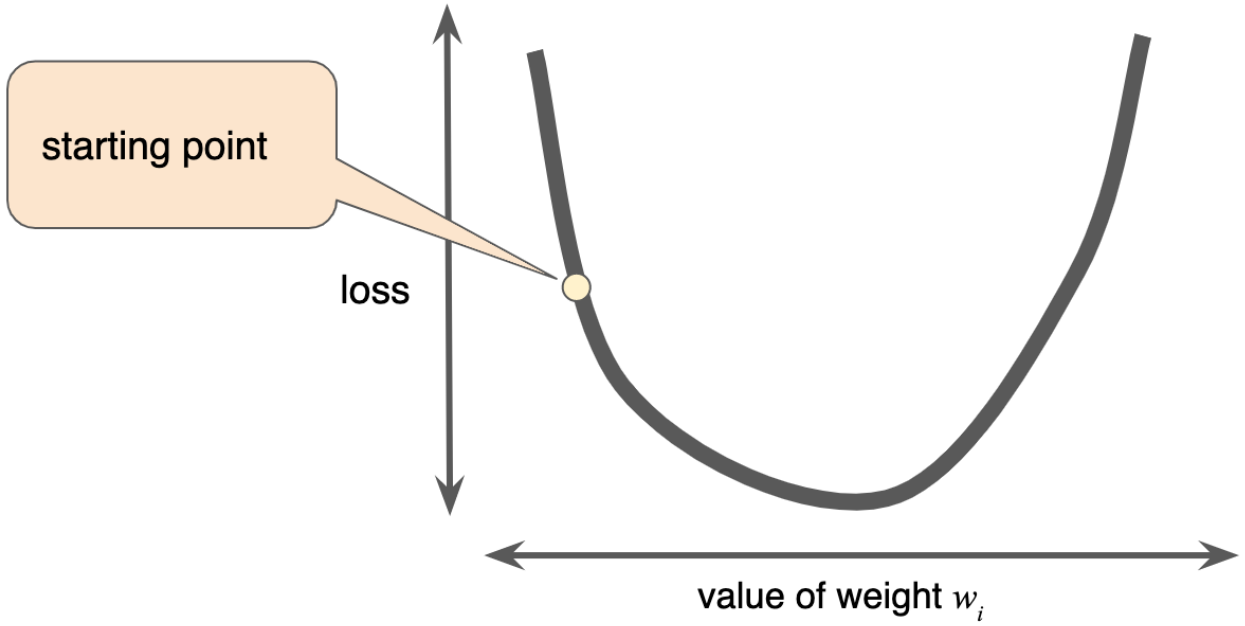
- Maths says:
  - If we have enough resources to calculate loss for all possible w1,
    - For a regression problem, loss vs w1 will be convex all the time



- Convex function only has one minimum.
- It is when the slope is 0
- Calculating the loss for all possible w1 is very inefficient. So, gradient decent is to rescue!

GRADIENT DECENT

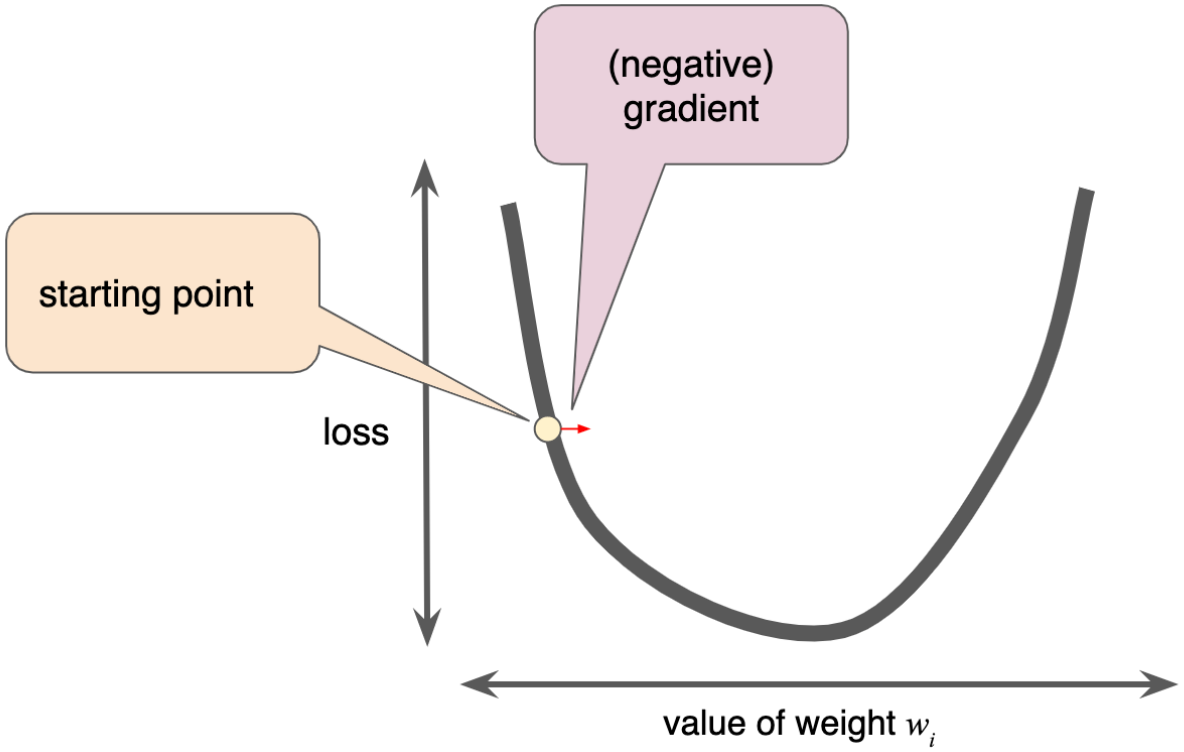
- Starting point for Gradient Decent



- Does not really matter what the starting point is.
- Many algorithms just take w1=0 for simplicity.
- Here we take something little lower than 0.

GRADIENT DECENT

- Relies on negative gradient



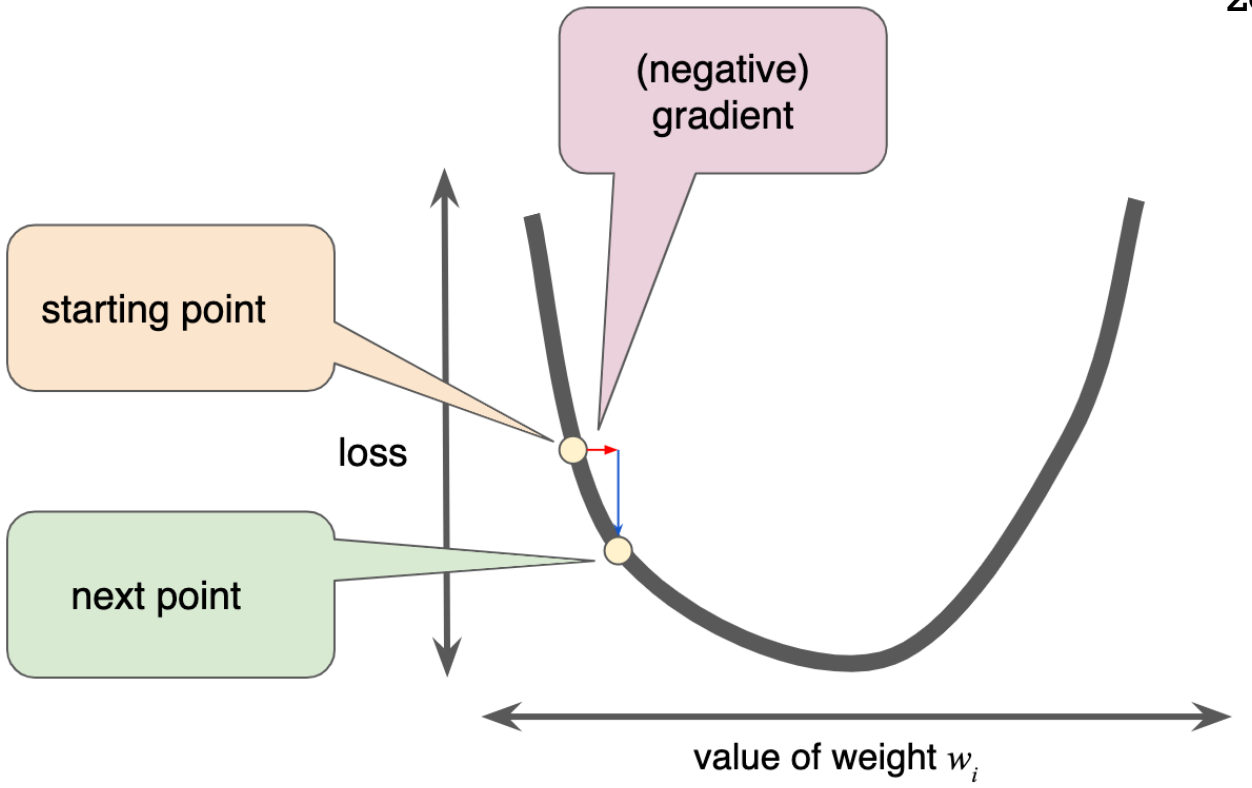
- Gradient is a vector: has direction + magnitude
- Points to the steepest increase in the loss function
- Hence, GD takes a step in direction of negative gradient to reduce the loss as quick as possible!

$$^{*}W_x = W_x - a\left(\frac{\partial Error}{\partial W_x}\right)$$

Old weight
Derivative of Error with respect to weight
New weight
Learning rate

GRADIENT DECENT

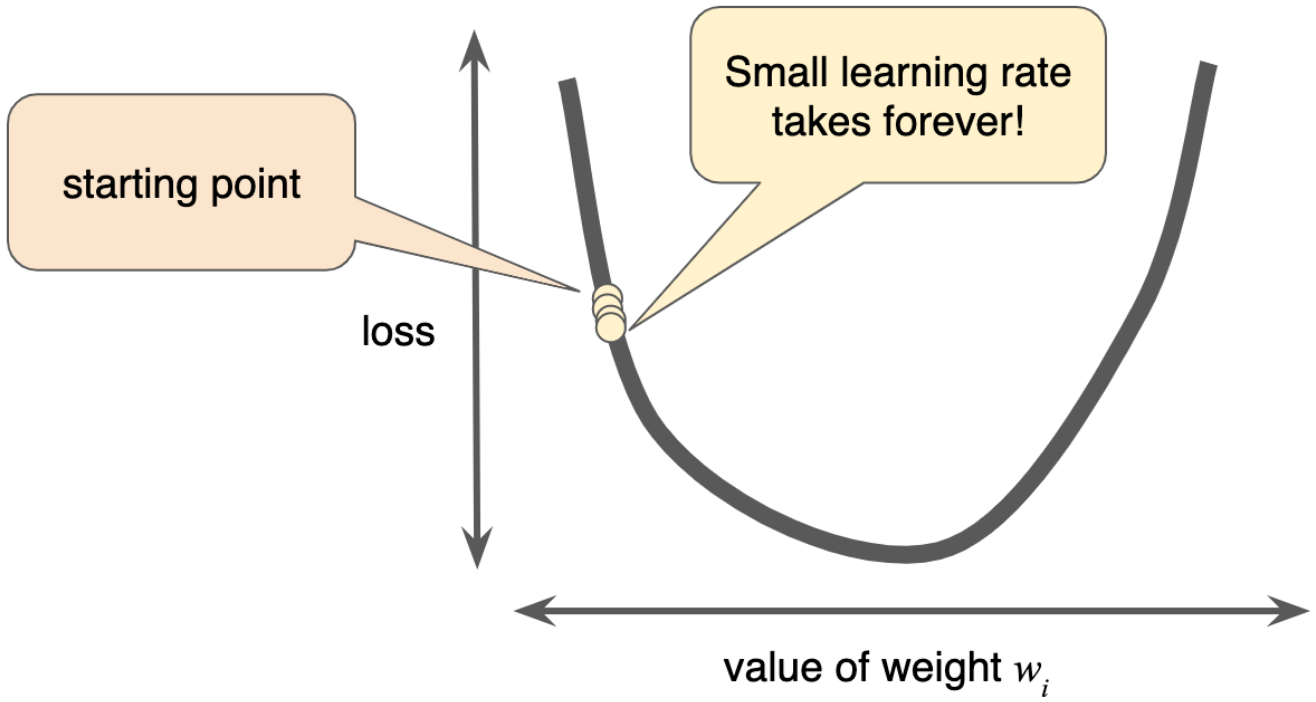
- GD taking the next step towards min loss



- GD algorithm adds some fraction of the gradient's magnitude to the start point and gets the next point
- This process repeats until the loss gets close to zero

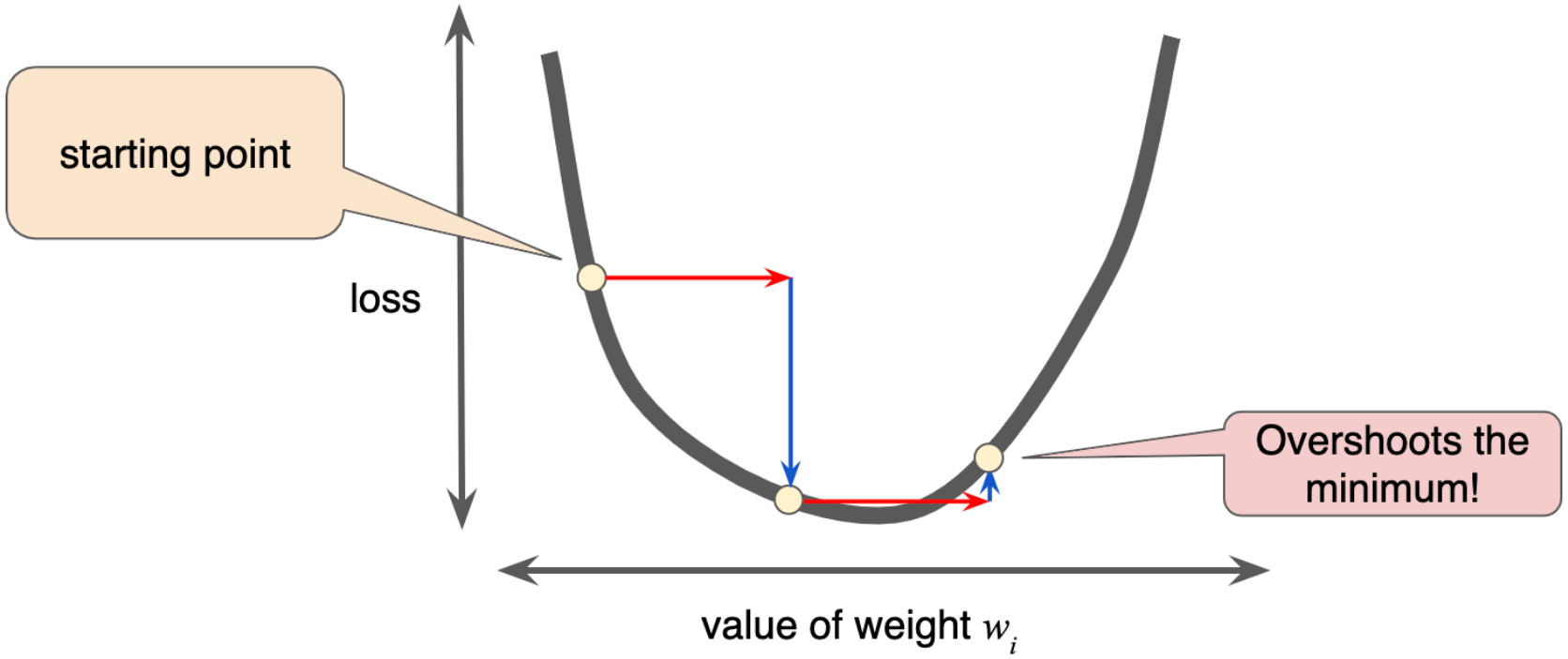
LEARNING RATE

- Determining the next step of GD :
  - Δ = Learning Rate * Gradient
  - GD take the next point Δ distance away



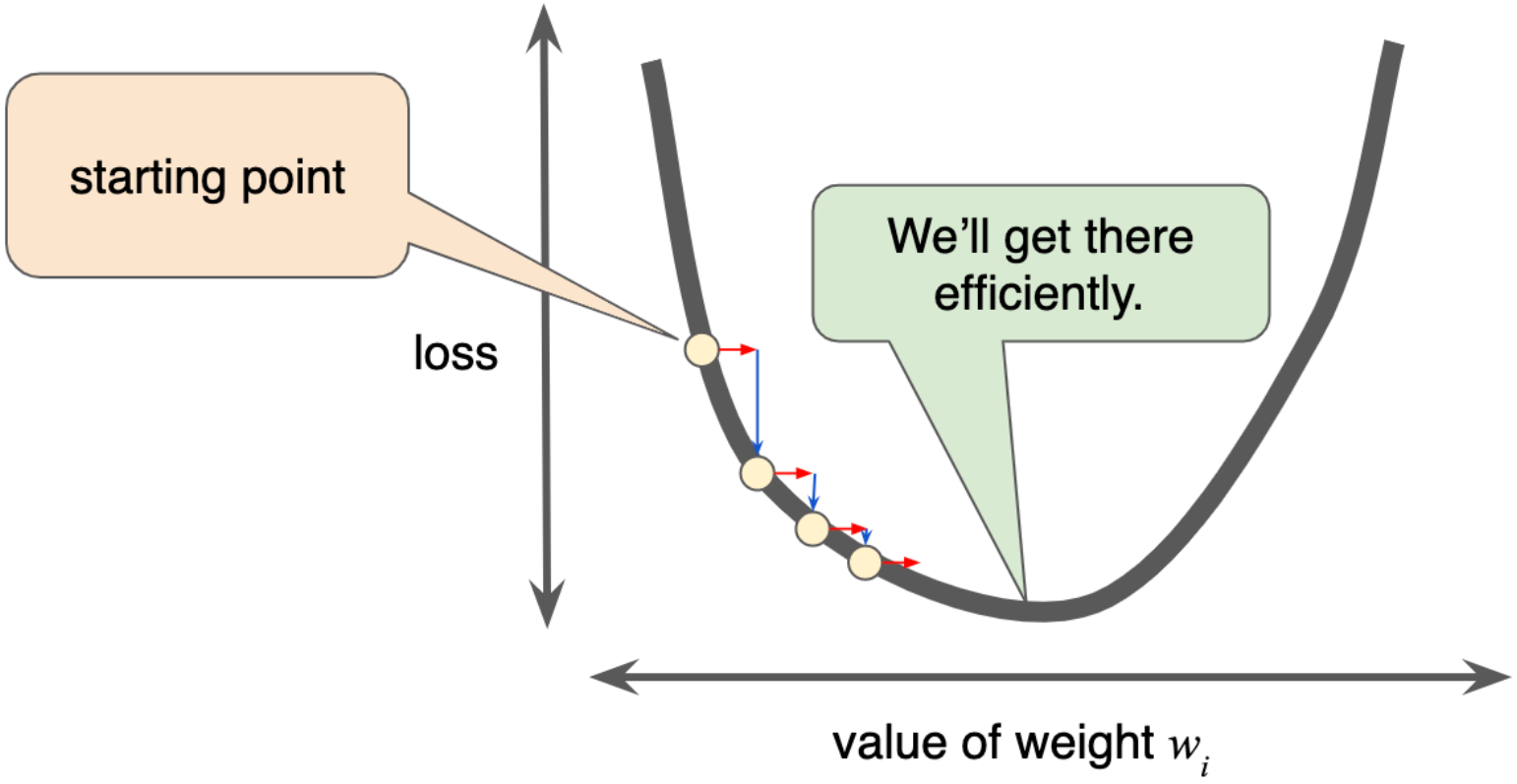
Too small learning rate ->
Too much time to finish learning

LEARNING RATE



Too large learning rate ->
perpetually bounce haphazardly across the bottom of the well

THE BEST LEARNING RATE?



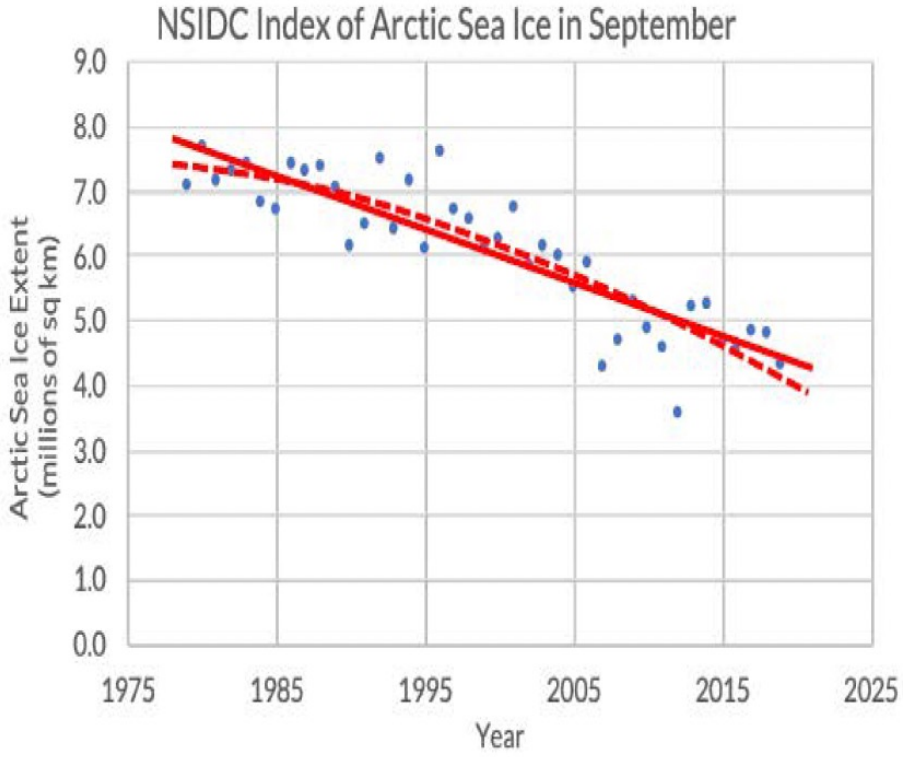
- **Optimizations:**
  - Stochastic Gradient Decent (SGD)
  - Mini-batch SGD

ML MODEL PERFORMANCE

		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

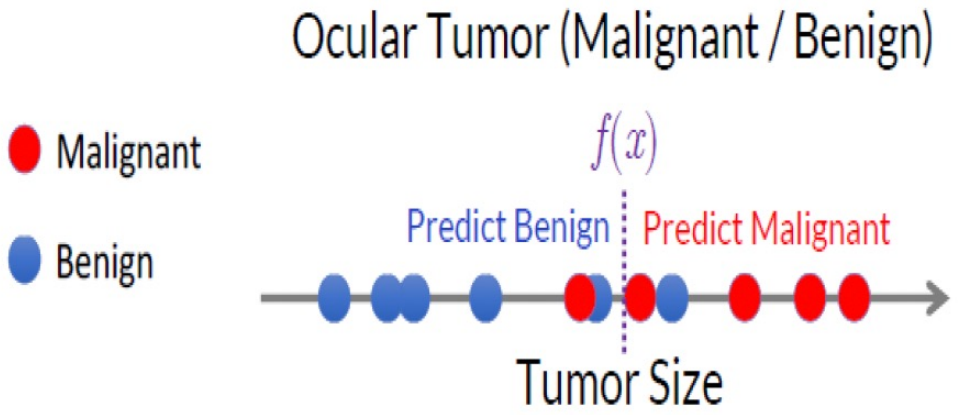
SUPERVISED LEARNING: REGRESSION

- Given $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Learn a function $f(x)$ to predict y given x
  - y is numeric == regression



SUPERVISED LEARNING: CLASSIFICATION

- Given $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Learn a function $f(x)$ to predict y given x
  - y is categorical == classification



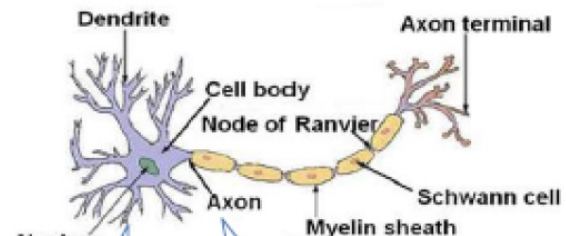
NN BASED CLASSIFICATION EXAMPLE

Pedestrian

Car

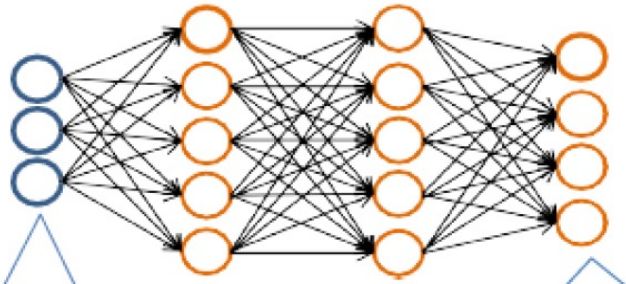
Motorcycle

Truck



Input

Output



Input (Image Pixel)

Output (Judgement)

$h_\Theta(x) \in \mathbb{R}^4$

Want $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.

when pedestrian

when car

when motorcycle

UNSUPERVISED LEARNING: CLUSTERING

- Given $x_1, x_2, \ldots, x_n$ (without labels)
- Output hidden structure behind the x's
  - E.g., clustering