Linear Regression

Intro to Machine Learning: Beginner Track #2

Slides: tinyurl.com/f20btrack2
Attendance code: **toystory**
Discord: bit.ly/ACMdiscord

acm.ai

Beginner Track

Who's it for?

- no experience in machine learning
- minimal experience coding
- want a solid foundation in the theory behind ML

What's covered?

- basics of machine learning
- theory and implementation of simple models
- introduction to useful ML libraries

When and where are meetings?

- **Location:** https://ucla.zoom.us/j/98508489562
- **Time:** Tuesdays 7-9 PM (PST)

Jenson Choi

Sudhanshu Agrawal

Adithya Nair

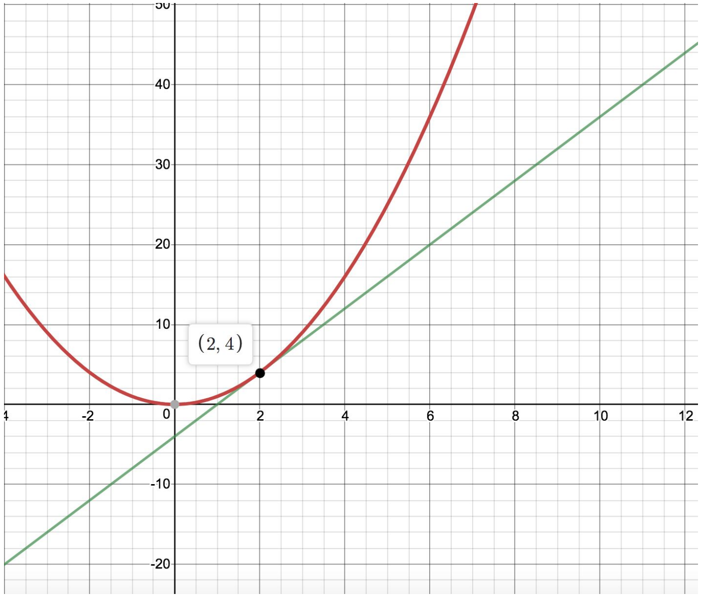
acm.ai

Let’s talk math

Why all this math?

- Machine learning is largely based on mathematics, statistics, and probability
- Thus, having a solid background in these concepts is very helpful!
- We'll be covering the basics - gradient descent, basic probability - so don't worry if you're a bit rusty
- Any q's? Ask any of the officers / post on Discord and we'll help you out + get you the resources you need

Derivatives



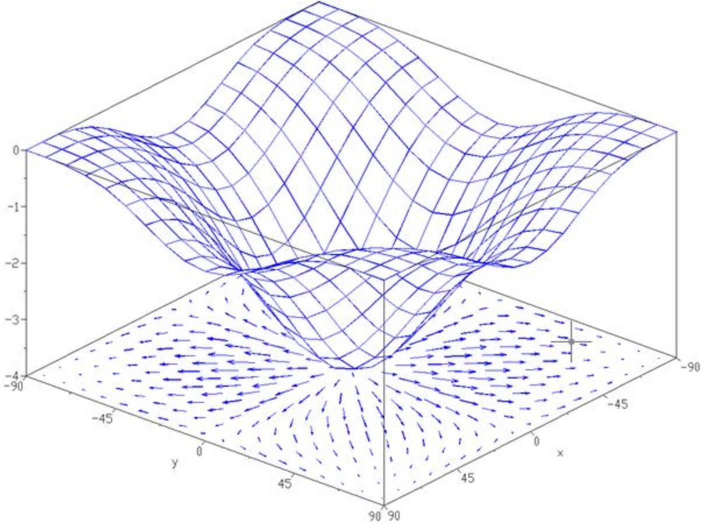
- The derivative of a function is the **rate of change** of the function
- If the derivative is **positive**, the function is **increasing**
- If the derivative is **negative,** the function is **decreasing**

acm.ai

Partial Derivatives

- To take the partial derivative of **f(x,y)** with respect to **x**, we assume **y** to be **constant** and take the derivative as you would for a single variable function

- To take the partial derivative of **$f(x_1,x_2,\ldots,x_n)$** with respect to some **x_i** we **take every other variable to be constant**, and continue.



$$f(x, y) = 2x + 3y^2$$

$$\frac{\partial f}{\partial x} = 2$$

$$\frac{\partial f}{\partial y} = 6y$$

acm.ai

Calculating a gradient

A gradient of an **n-dimensional function** is an **n-dimensional vector** of the partial derivatives of the function with respect to each variable

$$f(x, y, z) = xsin(y) + 2z^3$$

$$\nabla f(x, y, z) =< sin(y), xcos(y), 6z^2 >$$

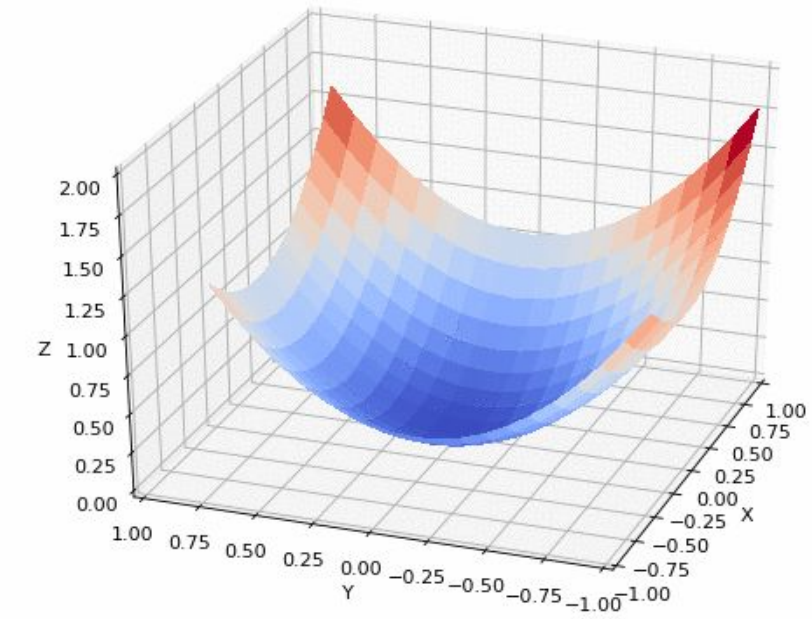
A Quick quiz

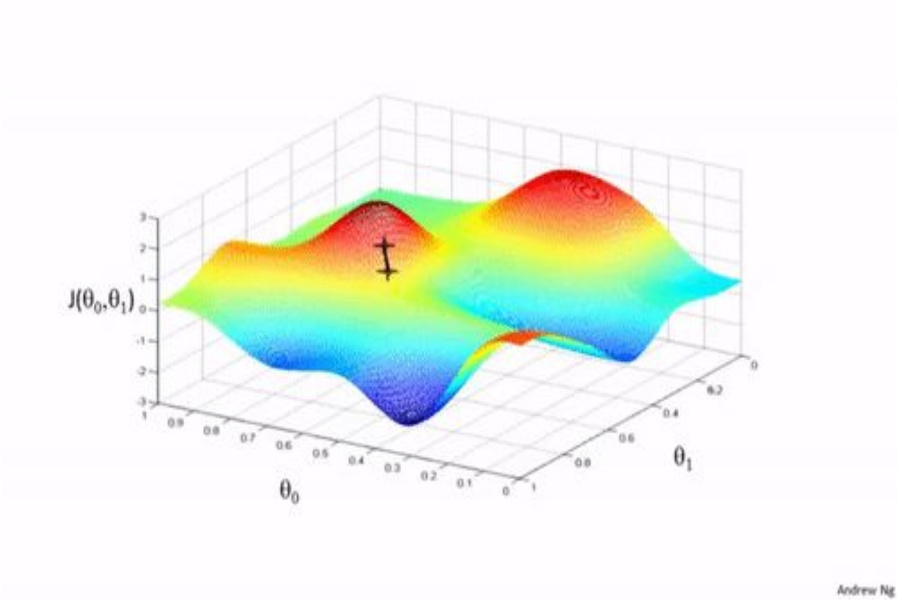
What is the gradient of $f(x,y,z) = x^2 + y^2 + z^2$?

a. $\nabla f = \ < 2x,\ 2y,\ 2z >$
b. $\nabla f = \ 2x + 2y + 2z$
c. $\nabla f = \ 1/3 < x^3,\ y^3,\ z^3 >$
d. $\nabla f = \ < x^2,\ y^2, z^2 >$

acm.ai

Gradient Descent: How we minimize the value of a function





acm.ai

Single Variable Gradient Descent

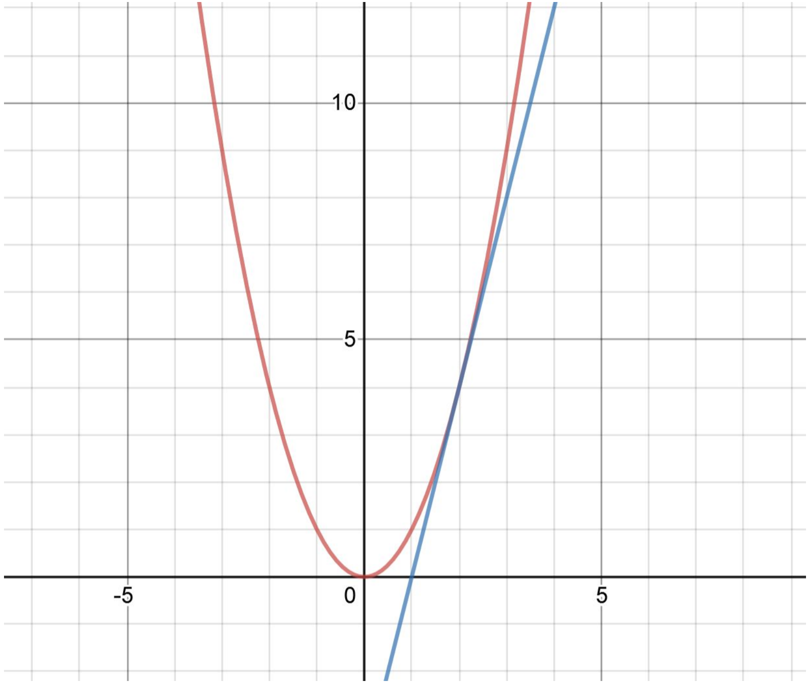
f(x) is a function of one variable: **x**

f'(x), the derivative, indicates whether the function is increasing or decreasing

If **f'(x)** is positive, the function is increasing. So if **x** increases, **f(x)** increases.

We want to decrease **f(x)** so we **decrease** x. i.e. we **subtract** *something* from x

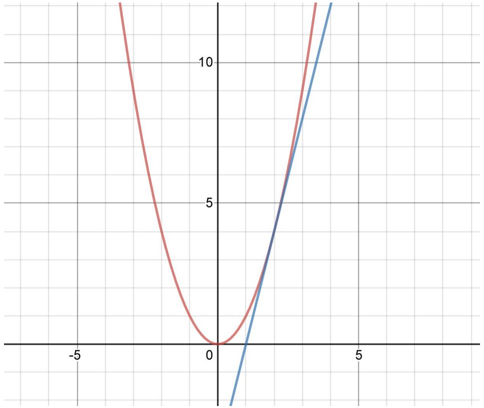
Similarly if **f'(x)** is negative, if we want to decrease f(x) we **increase** x



acm.ai

Single Variable Gradient Descent

To summarize: We want to **minimize** f(x), so
if **f'(x)** is **positive**, we want to **subtract** *something* from x
if **f'(x)** is **negative**, we want to **add** *something* to x



How do we do that?
Use **f'(x)** itself!
But careful! We want to do the **opposite** of what f'(x) tells us to

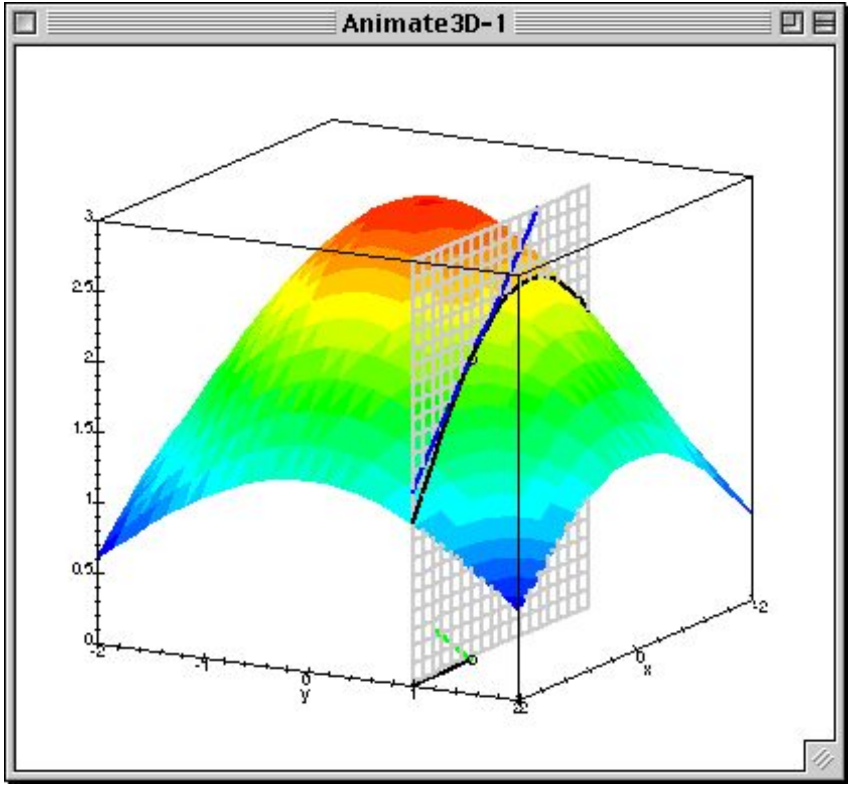
So we can **update** x like this $$x = x - \alpha f'(x)$$

alpha is just a constant we choose to scale f'(x). We call it the **learning rate.**

acm.ai

Multivariable Gradient Descent

- The gradient is the direction of **steepest ascent**
- Meaning that if we go in the same direction as the gradient we **increase** the value of the function
- But we want to **decrease** the value of the function
- So we go in the opposite direction as the gradient i.e. **gradient descent!**



acm.ai

Multivariable Gradient Descent

x is now a **vector**

$$\vec{x} = [x_1, x_2, \ldots . x_n]$$

The **gradient** is also a **vector**

$$\nabla f(\vec{x}) = [\frac{\delta f}{\delta x_1}, \frac{\delta f}{\delta x_2} \ldots . . \frac{\delta f}{\delta x_n}]$$

So we **update** the x vector using the gradient vector

$$\vec{x} = \vec{x} - \alpha \nabla f(\vec{x})$$

acm.ai

Let's get into the ML

An example problem - Housing Prices

- Say we wanted to predict the price of a house
- This requires knowing some information about the house:
  - What's the square footage, how many rooms does it have, etc.
- We call this information about the house **features**
- The **price** of the house **depends** on its **features.** But what is this relation?
- We find this relation using Machine Learning

Which house is worth more?

3.5 million

200k

acm.ai

Some terms

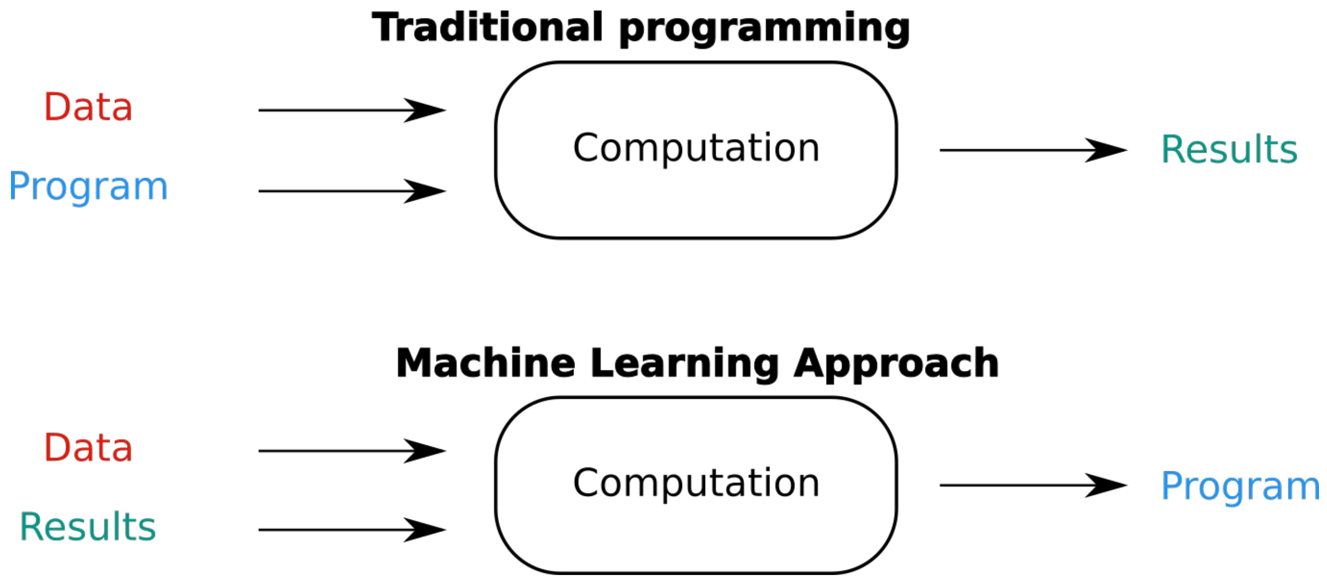
- **Feature** - some **property** of the object we are working with.

  Eg: For houses, square footage is a feature.

- **Target/Label** - the **true** value of what we are trying to predict.

  Eg: For houses, the target would be the price of the house.

The ML way vs The Old Fashioned way



The ML way - Pattern Recognition

- In the machine learning way, we assume that for any new house that we are given, we'll be able to tell its price if we find some **pattern** in the prices of some other houses.
- That is, given some houses, with some **features** and **targets**, we think that some new house (whose price we don't know) will follow the trend of the houses of which we do know the price.

ML pipeline

Input data	Preprocessing	Training	Testing	Interpretation
Training and testing data The more data we have, the better	Scaling data, removing outliers, etc.	Choosing the **function** and representing the data in that form; Training data is used to develop model	Evaluating the model using testing data	Choosing a new model, introducing more variables, etc.

How do we represent our data?

- Each **row** represents the information for **one house** in our data set
- Each **column** represents one **feature**: Like number of bedrooms
- The **target** is the list of **prices** of the houses. It is what we want to predict. We call this vector **y**

y :target

Bedrooms	Sq. feet	Neighborhood	Sale price
3	2000	Normaltown	$250,000
2	800	Hipsterton	$300,000
2	850	Normaltown	$150,000
1	550	Normaltown	$78,000
4	2000	Skid Row	$150,000

This is our "training data."

acm.ai

Hypothesis

- We want to learn what **function** will **output the desired price**, given some **features as input.**
- This function is called the **hypothesis**. It models the **pattern** we are interested in.
- Let's call our hypothesis **y-hat**

$$\hat{y}(x_1, x_2 \ldots x_n)$$

and the features would be the inputs: x_1, x_2, … , x_n

- Our goal now is to determine y-hat

acm.ai

What model should we use?

- First, what is a model?
- The word *model* is thrown around a lot in ML, and there doesn't seem to be one rigorous definition.
- Think of the model as the machine's interpretation of the problem, how it "models" the situation provided.

What model should we use? (contd.)

- Now, for housing prices, one might assume that features like square footage and the number of rooms are proportional to the price
- We might think that features like crime rate will negatively affect the price
- For all these features, there seems to be a direct relationship: the feature either directly increases or directly decreases the price.
- A **linear model** might be a good choice: i.e. something like y = mx+b

$$\hat{y}(x) = b + w_1x_1 + w_2x_2 + \ldots\ldots w_nx_n$$

- This is our **hypothesis**

acm.ai

Weights

$$\hat{y}(x) = b + w_1 x_1 + w_2 x_2 + \ldots\ldots w_n x_n$$

- The **weights** are $\mathbf{w_1, w_2 \ldots. w_n}$ and $\mathbf{b}$.
- They are the **learnable parameters** of our model.
- In the hypothesis above, we can change the function by changing the values of $\mathbf{b}$ and $\mathbf{w_1, w_2, \ldots. w_n}$
- We need to find the *best* possible weights for our model.

acm.ai

Weights - Quick Poll

- Let's assume a house's price depends solely on four features: the number of rooms it has, how long ago it was constructed, the crime rate in the neighborhood and the distance of the house from the closest hospital. The neighborhood crime rate and the number of rooms affect the house price the most. Which feature would have the most negative weight?

The parameters

$$\hat{y}(x_1, x_2 \ldots x_n) = b + w_1 x_1 + w_2 x \ldots + w_n x_n$$

An input **X** is an **n-dimensional vector** for the n features in the example

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

The weight **W** is also an n-dimensional vector.

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix}$$

The bias **b** is a real number.

$$b$$

acm.ai

Model

$$\hat{y}(x) = b + w_1x_1 + w_2x_2 + \ldots\ldots w_nx_n$$

So our model, **yhat(x)** can also be represented in the following manner :

$$W.X + b$$

i.e. take the dot product of the W vector and the X vector and then add the scalar b

In matrix notation this is commonly written as

$$W^TX + b$$

acm.ai

Loss Function: a measure of error

- To update W, we need to first talk about something called the loss function.
- This is sometimes referred to as the cost function.
- It measures the **error** in your predictions compared to the target values.
- There are many choices of function to measure the error. We will go with the **Mean Squared Error (MSE)**

$$L(\hat{y_1}, \hat{y_2}, \ldots \hat{y_m}) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y_i})^2$$

y: the actual **target** value

yhat: the output **predicted value**

i: the ith **training sample**

acm.ai

Loss Function as a function of weights and bias

- The loss function is a function of your predictions
- Your predictions are functions of the weights and bias of your model
- The loss function can also be thought as a function of the weights and bias of your model

$$L(\hat{y_1}, \hat{y_2}, \ldots \hat{y_m}) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y_i})^2 \qquad \hat{y}(x) = b + w_1x_1 + w_2x_2 + \ldots.. w_nx_n$$

acm.ai

Loss Function - Quick Poll

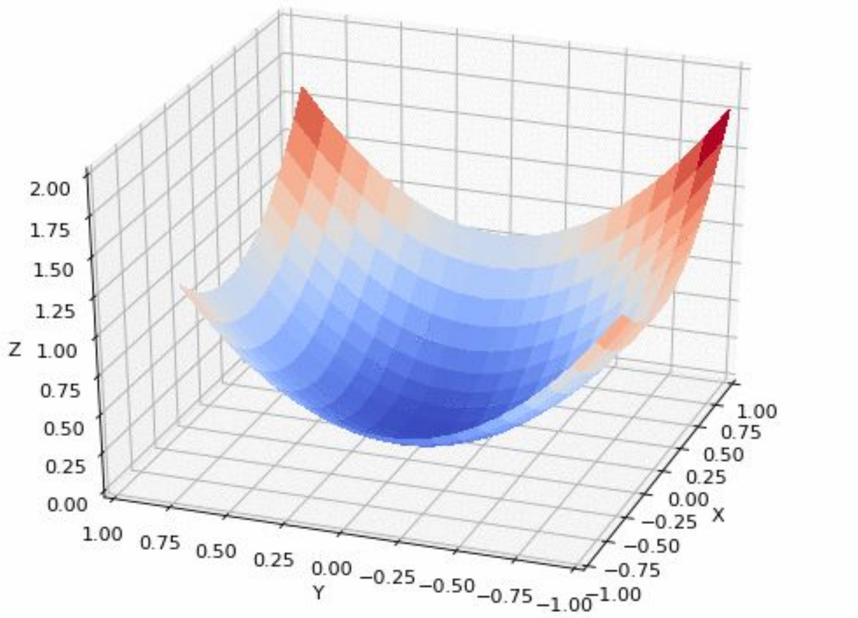
- What is the loss function (MSE) measuring?
  - Error of the predictions that your model makes (compared to true value)
  - The inherent error in your dataset (such as when the dataset is not properly cleaned)
  - Weight(s) of your model
  - Your predictions

Learning Weights: Minimize Loss

- Our loss is a function of the **weights** and **bias** of our model
- Our model learns by updating its **weights** and **bias**
- If loss function outputs a small value → our model is **accurate**
- So, we need to find those **weights** for which the **loss is minimized**
- How do we do that?

acm.ai

Gradient Descent



$$\vec{x} = [x_1, x_2, \ldots . x_n]$$

$$\nabla f(\vec{x}) = [\frac{\delta f}{\delta x_1}, \frac{\delta f}{\delta x_2} \ldots . \frac{\delta f}{\delta x_n}]$$

$$\vec{x} = \vec{x} - \alpha \nabla f(\vec{x})$$

acm.ai

Minimize loss using gradient descent!

Taking the **gradient** of our MSE loss function

$$\frac{\partial L}{\partial w_j} = \frac{2}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)x_{ij} \qquad w_j = w_j - \alpha\frac{\partial L}{\partial w_j}$$

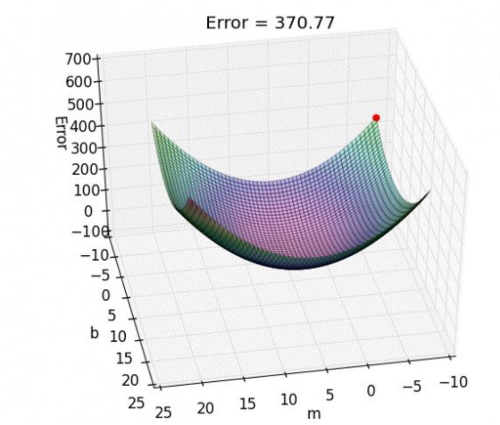
$$\frac{\partial L}{\partial b} = \frac{2}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i) \qquad b = b - \alpha\frac{\partial L}{\partial b}$$

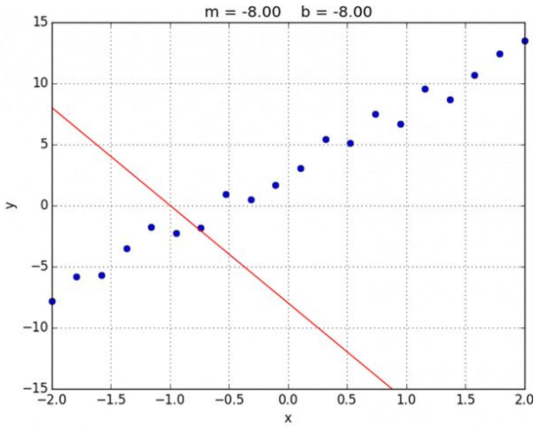
i refers to the **i**th training sample, **j** refers to the **j**th feature
Here's the full derivation of the gradients of Loss function

acm.ai

Best Fit

- Finding the optimal hypothesis function can be thought of as finding the **best fit “curve”** for your data. The best fit curve ensures minimum loss.





- This is linear regression with one feature. We are trying to fit a line with the given data.
- You will implement this in a project later in the quarter!

After we learn weights: Testing

- To test our model, we first select some of the data points we have.
- We then feed the **input features** of those data points into our model and keep aside the true **y** values
- Our model generates **predictions** using the input features
- We calculate the **loss** between our predictions and the true values
- And that loss tells us how well our model has performed!

What we just did: Supervised learning

- In very simple terms, we told our model what the right answer was
- Types of supervised learning
  - Classification: output labels
  - Regression: map input to continuous output
- Classification or regression?
  - Cat vs dog?
  - Number of fish in certain reef?
  - Normal mail or spam?

So there you have it

- What we did today was a form of Supervised Learning
- We’ll be concentrating on Supervised Learning in this series.
- The next topic to learn is Logistic Regression, where we’ll be classifying objects, instead of predicting values.

Answer to Poll questions

1. A
2. Neighborhood crime rate
3. Error of the predictions that your model makes

Thank you! We’ll see you next week!

Please fill out our feedback form:
forms.gle/mEUVe9bpGgsYDXfk7

Next week: Logistic Regression

How does a computer recognise cats and dogs?

Today’s event code: **toystory**

FB group: facebook.com/groups/uclaacmai

Github: github.com/uclaacmai/beginner-track-fall-2020

ACM AI