



DATA ANALYSIS AND VISUALIZATION

INSTRUCTOR: UMME AMMARAH





LOGISTIC REGRESSION



DERIVATIVES

$$f(x) = 4x$$

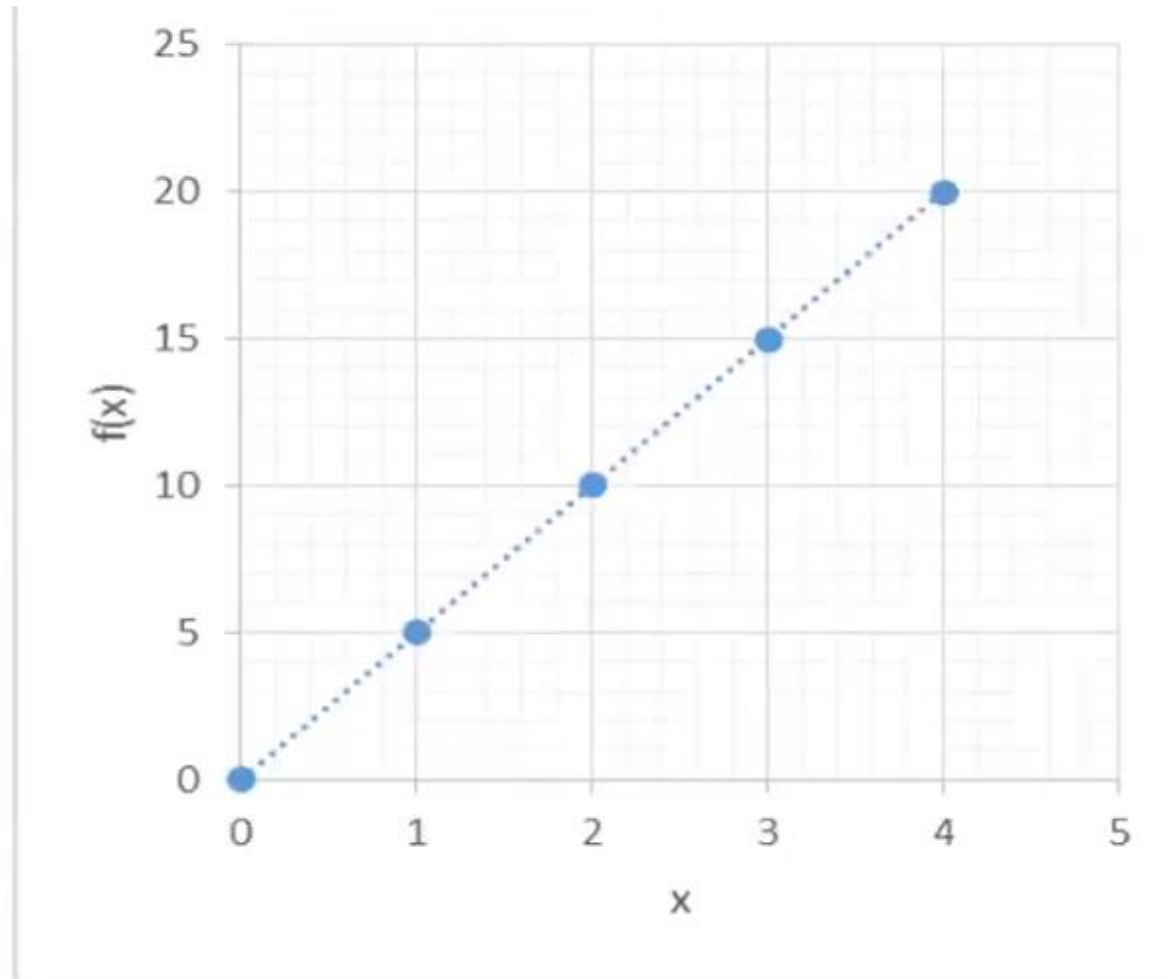
$$f(x) = x^3$$

$$f(x) = (x + 2)^4$$

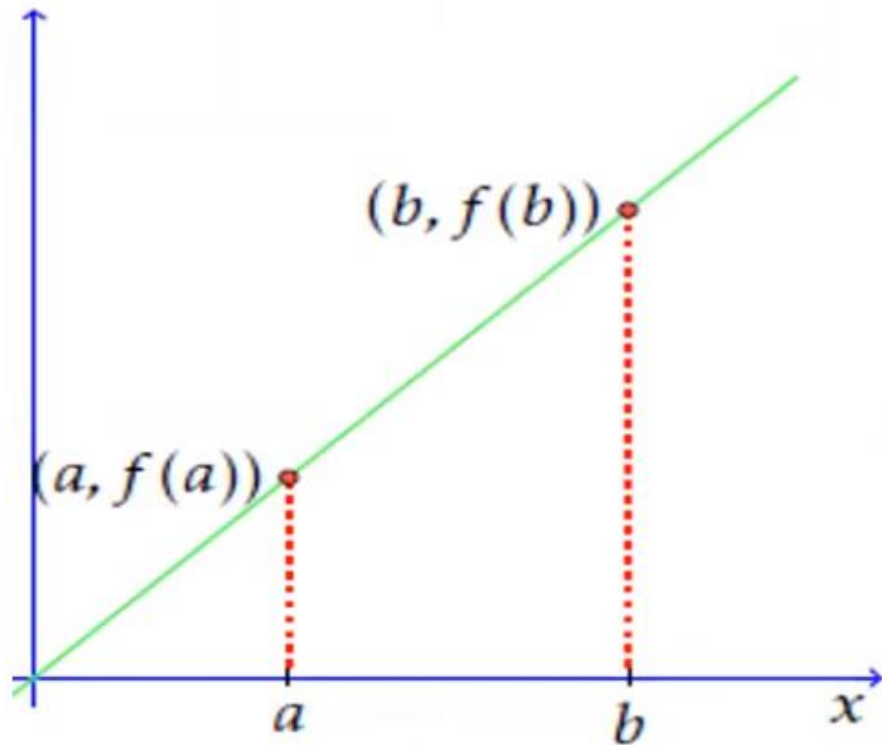
$$f(x) = (3x + 2y + 2)^2$$

SLOPE OF LINE

x	$f(x)$
0	0
1	5
2	10
3	15
4	20

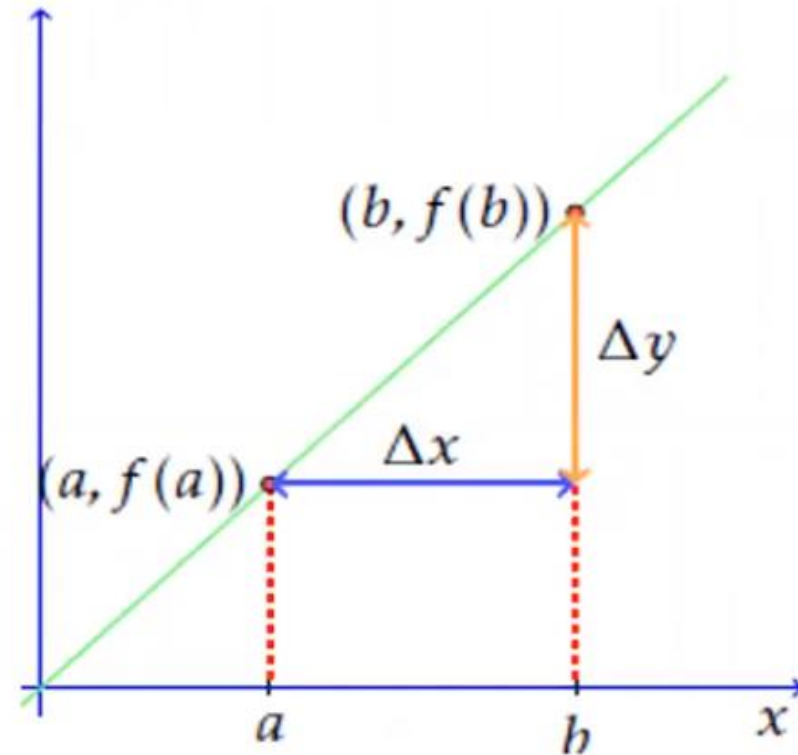


SLOPE OF LINE



$$\Delta x = b - a$$

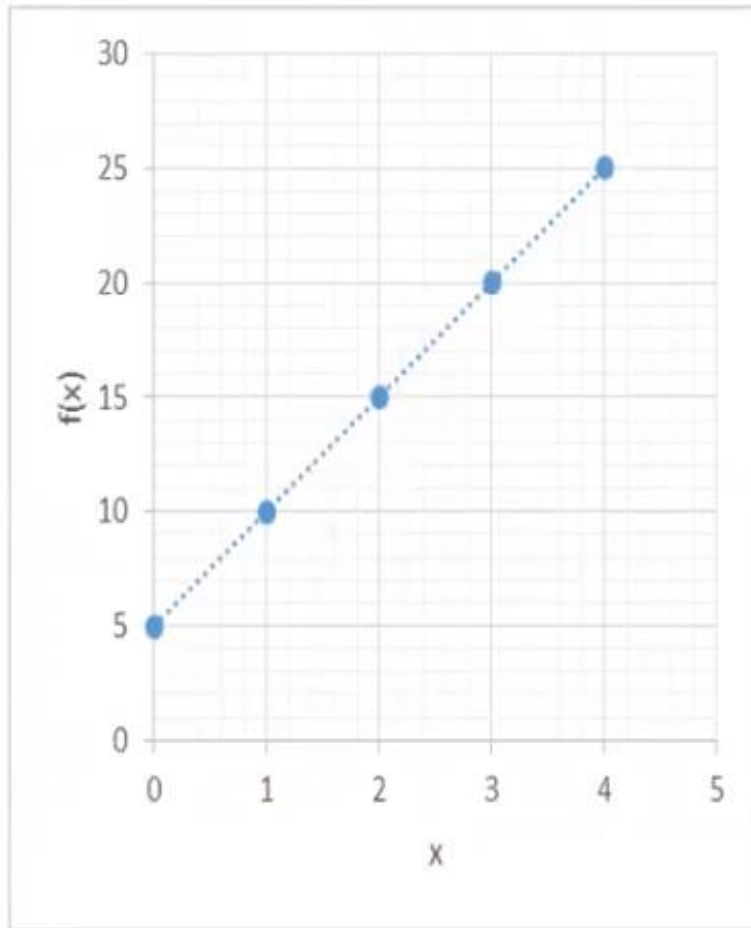
$$\Delta y = f(b) - f(a)$$



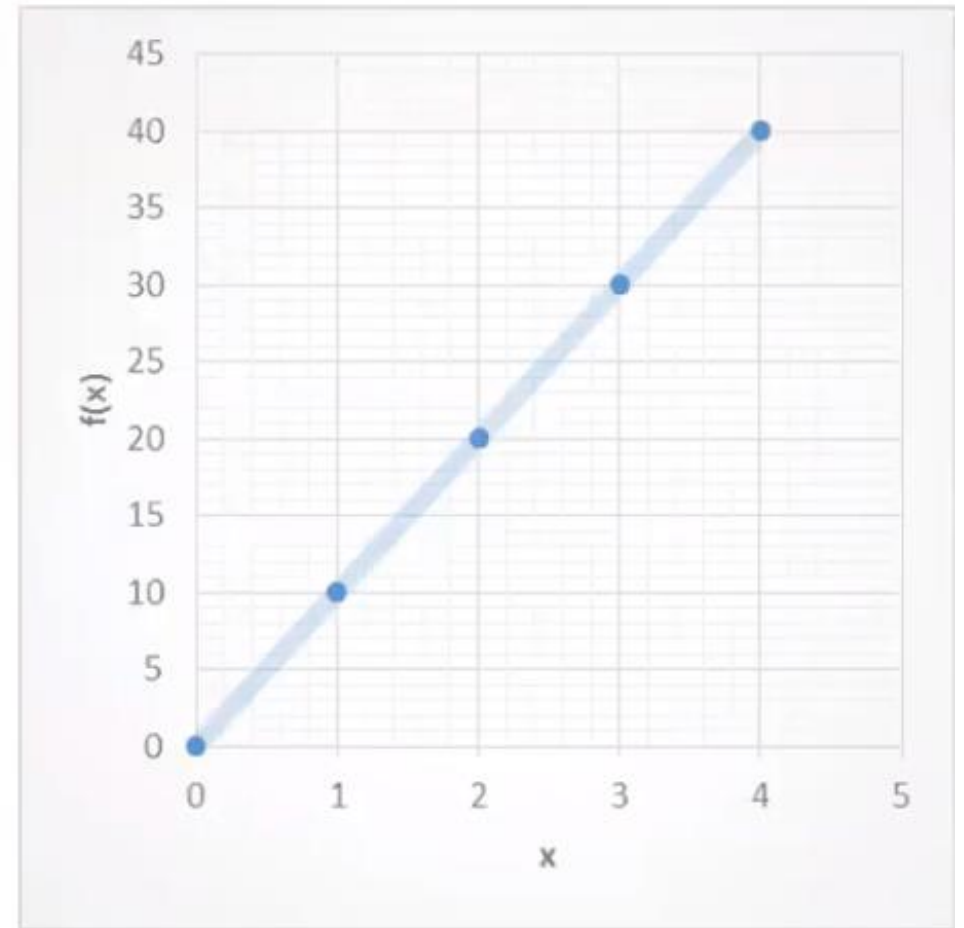
$$\text{slope} = \Delta y / \Delta x$$

SLOPE OF LINE

x	$f(x)$
0	5
1	10
2	15
3	20
4	25



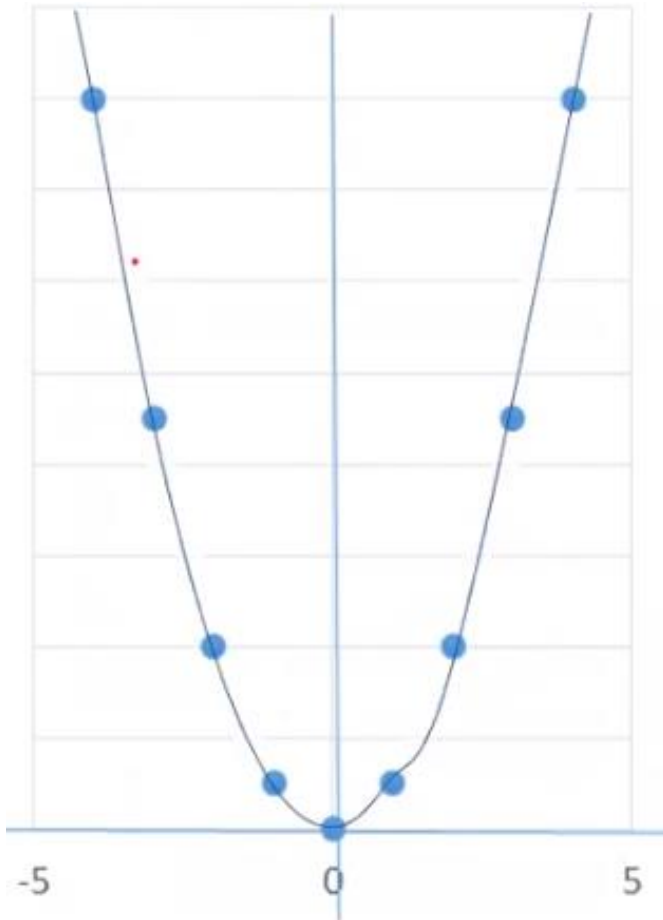
x	$f(x)$
0	0
1	10
2	20
3	30
4	40



SLOPE OF CURVE

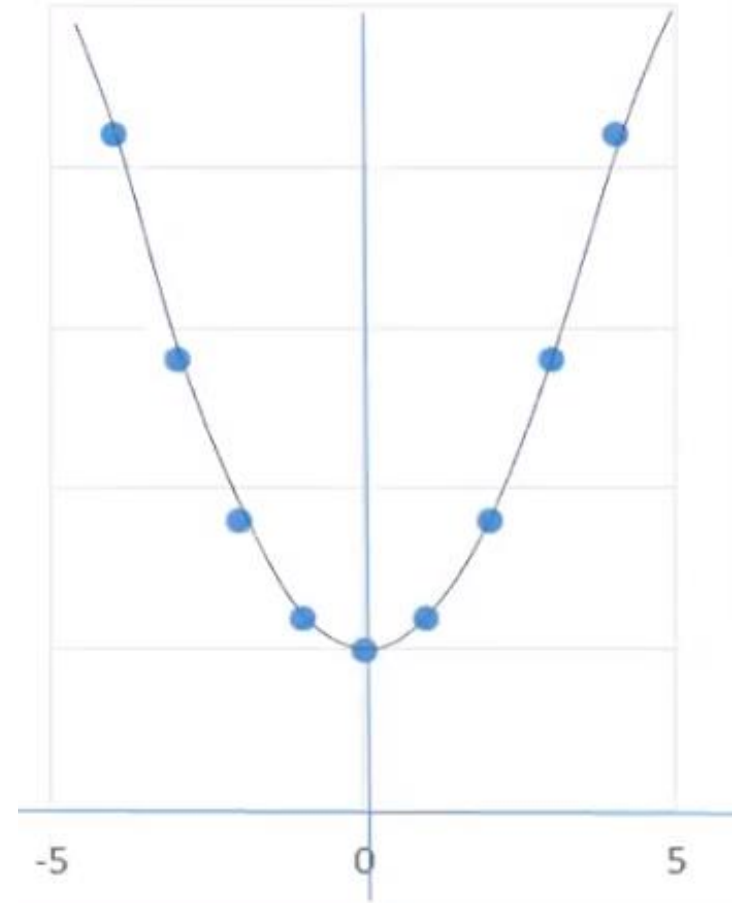
$$f(x) = x^2$$

x	f(x)
0	0
1	1
-1	1
2	4
-2	4
3	9
-3	9
4	16
-4	16



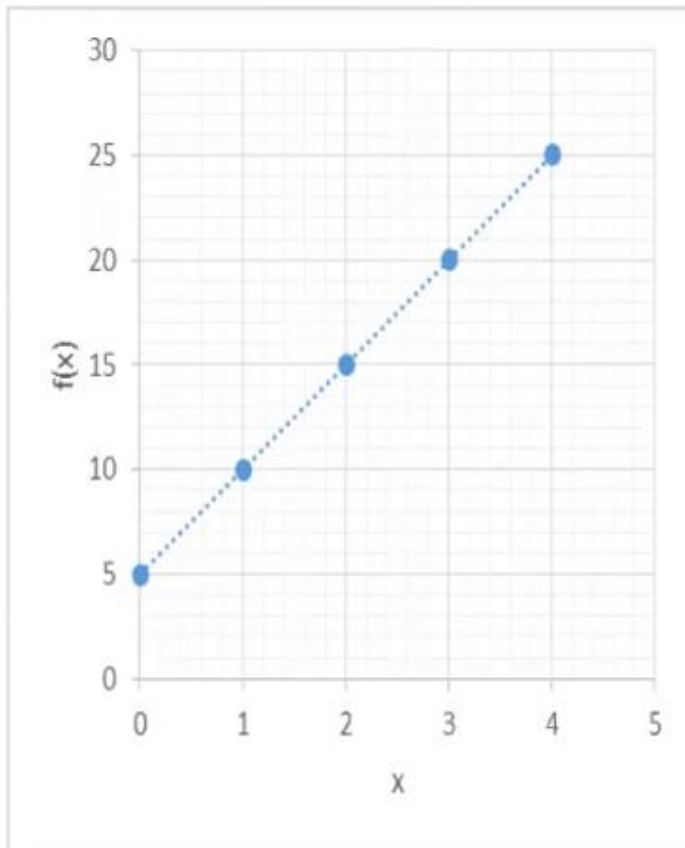
$$f(x) = x^2 + 5$$

x	f(x)
0	5
1	6
-1	6
2	9
-2	9
3	14
-3	14
4	21
-4	21



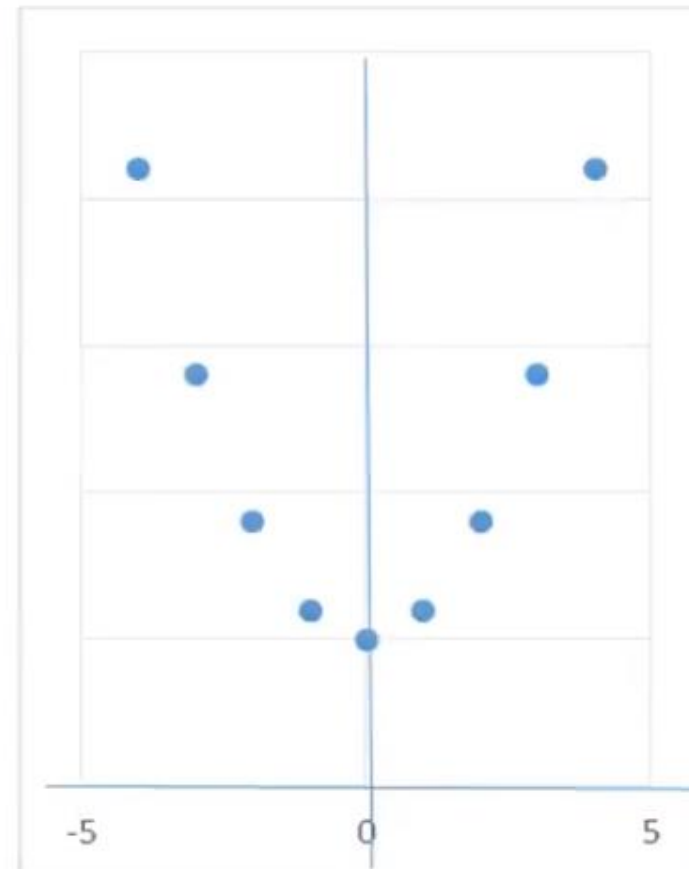
FINDING MINIMA AND MAXIMA

x	f(x)
0	5
1	10
2	15
3	20
4	25



$$f(x) = x^2 + 5$$

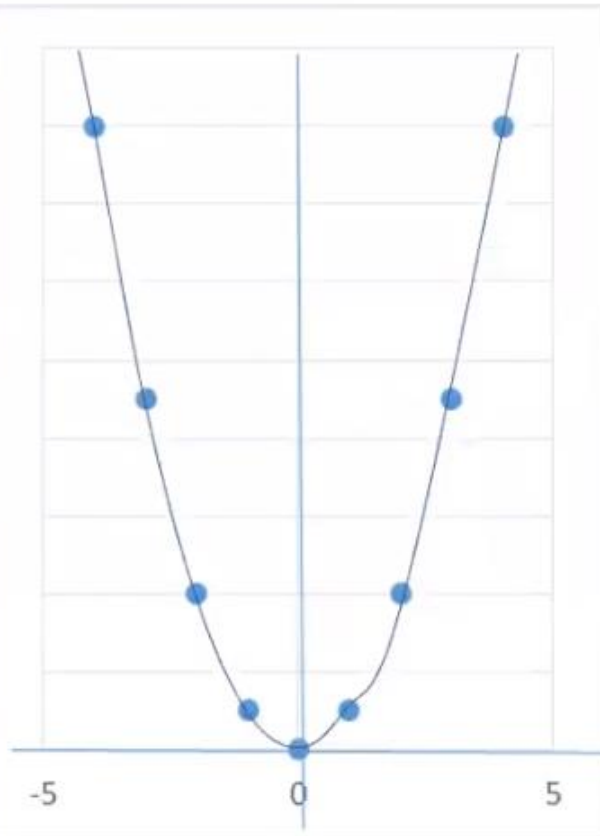
x	f(x)
0	5
1	6
-1	6
2	9
-2	9
3	14
-3	14
4	21
-4	21



MINIMA AND MAXIMA

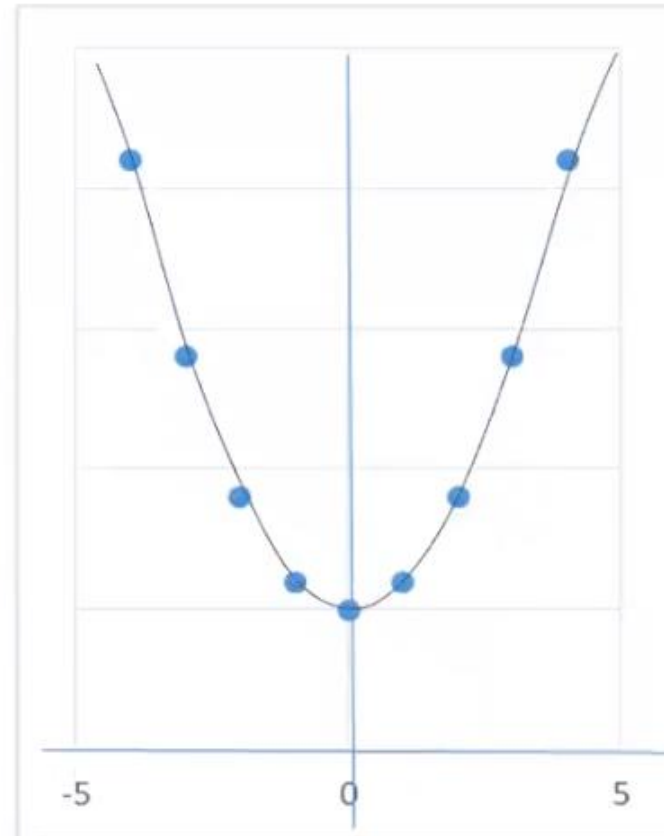
$$f(x) = x^2$$

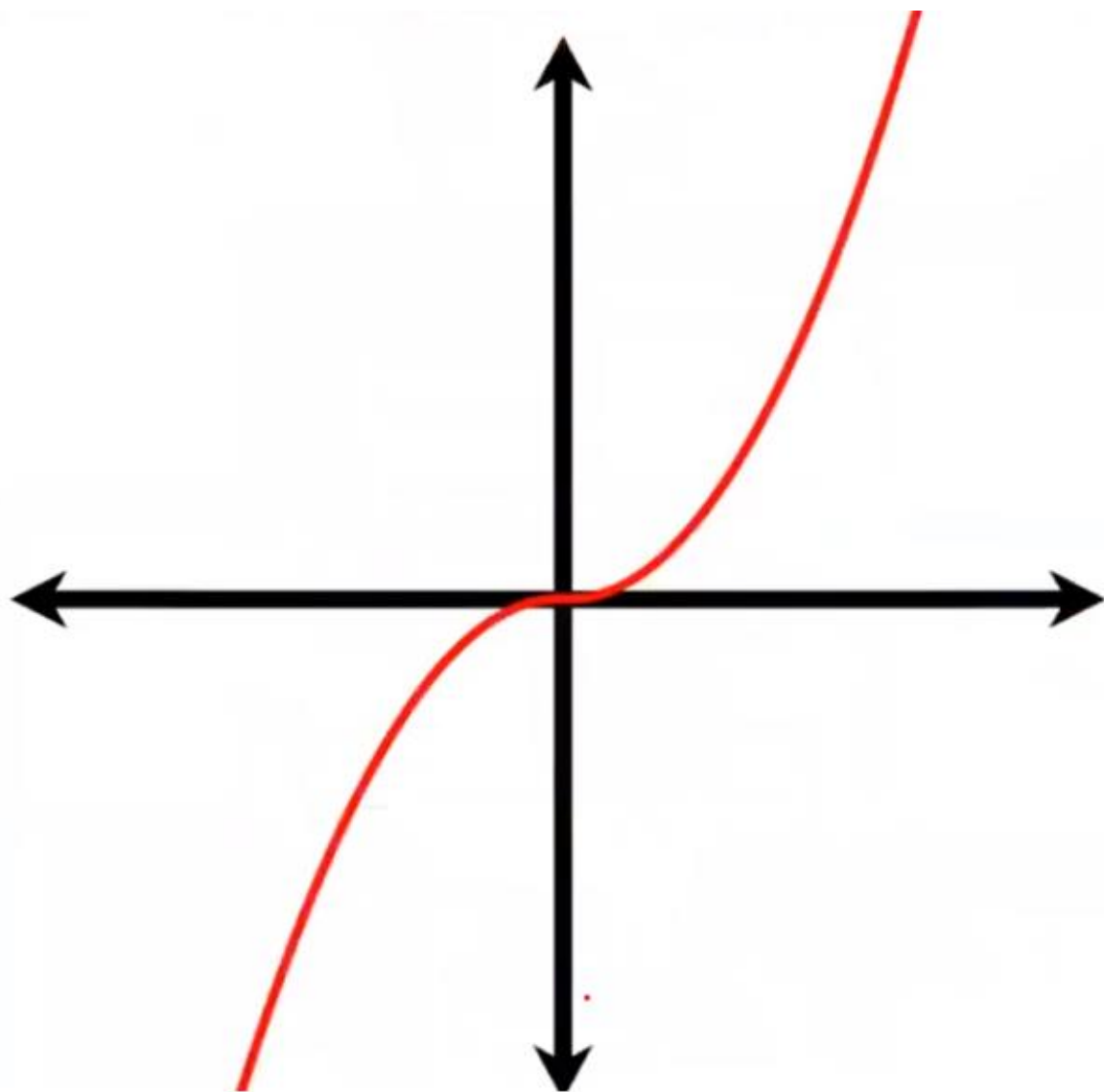
x	f(x)
0	0
1	1
-1	1
2	4
-2	4
3	9
-3	9
4	16
-4	16



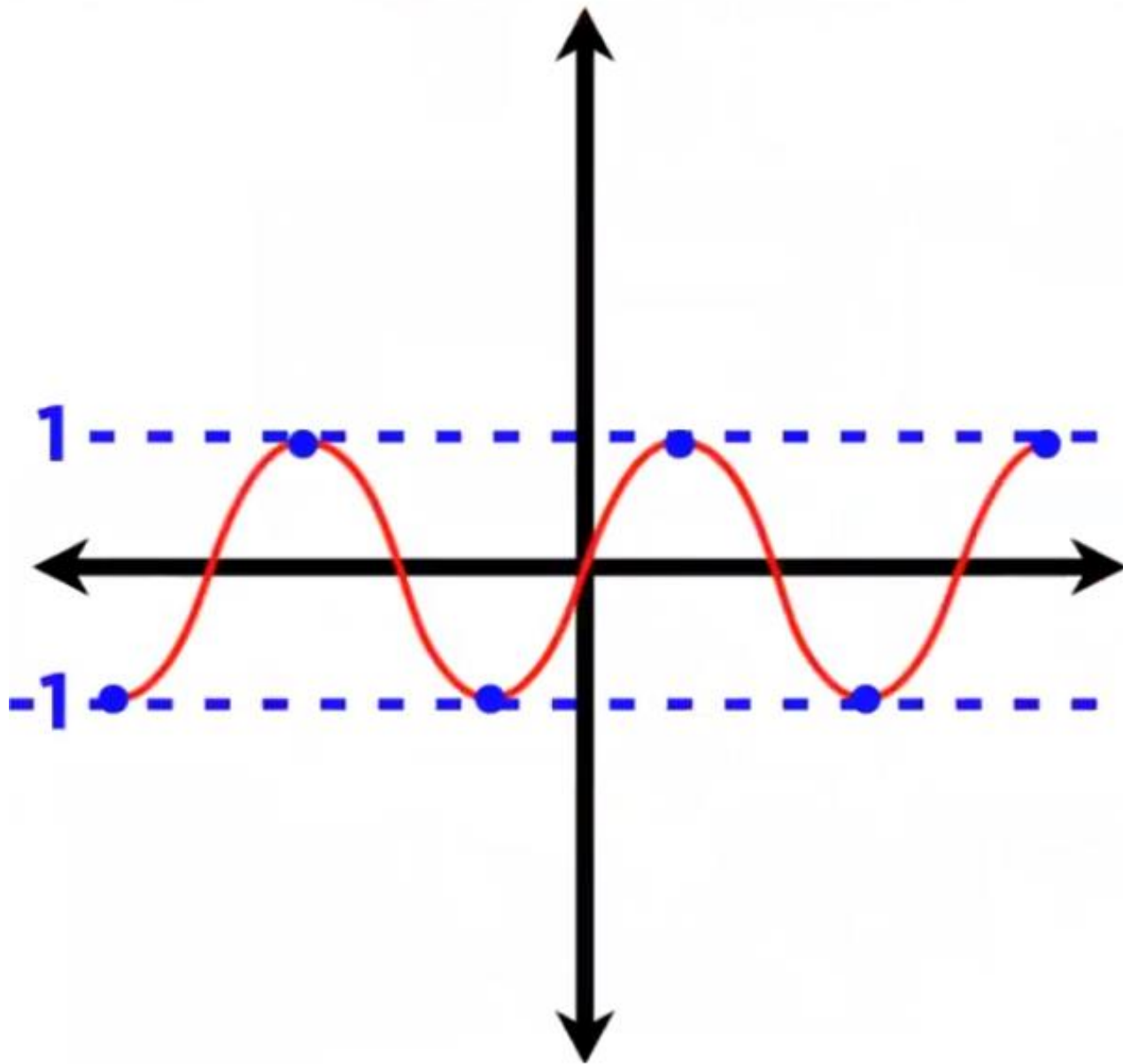
$$f(x) = x^2 + 5$$

x	f(x)
0	5
1	6
-1	6
2	9
-2	9
3	14
-3	14
4	21
-4	21

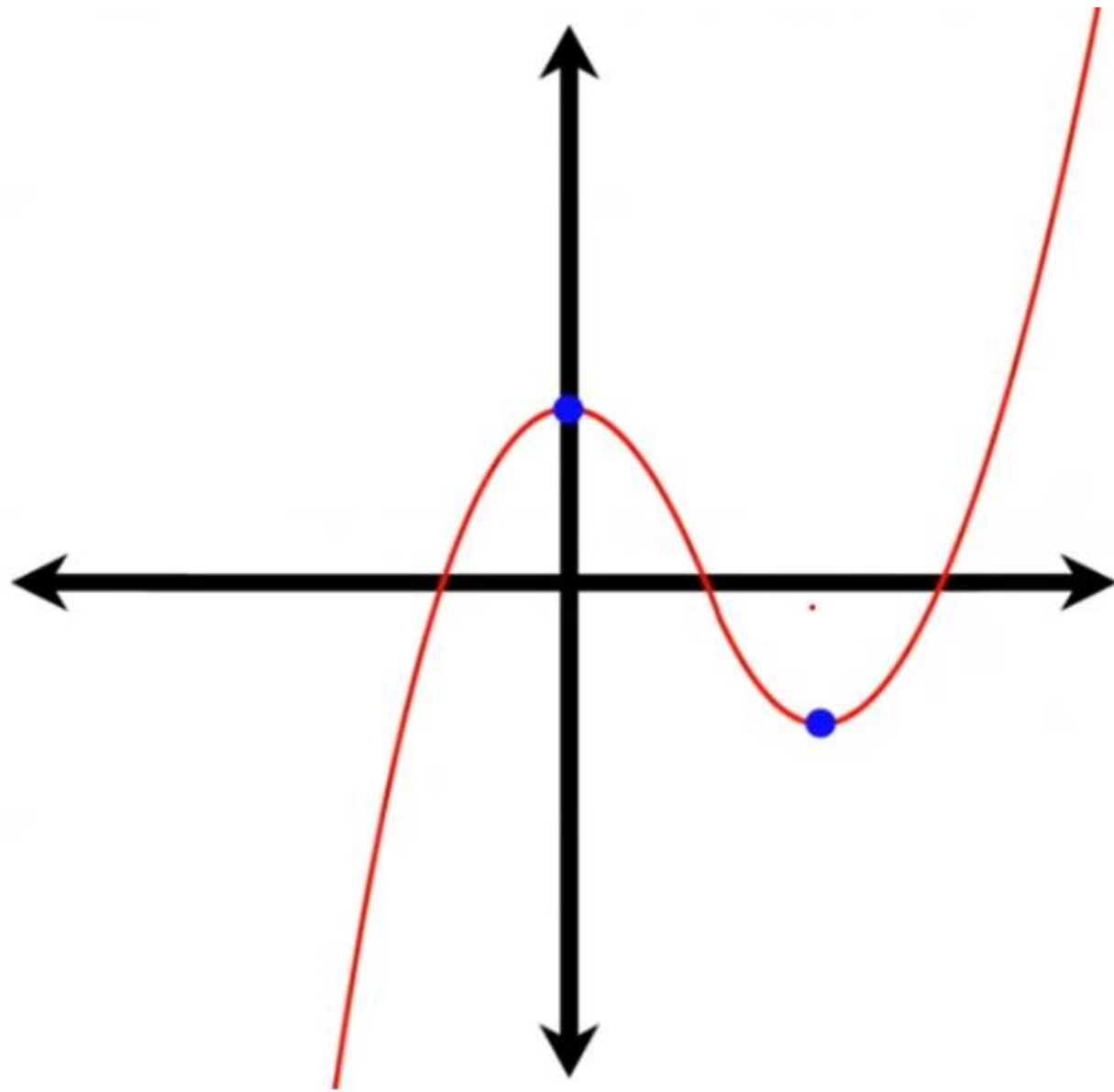




$f(x) = x^3$ has no
maxima/minima



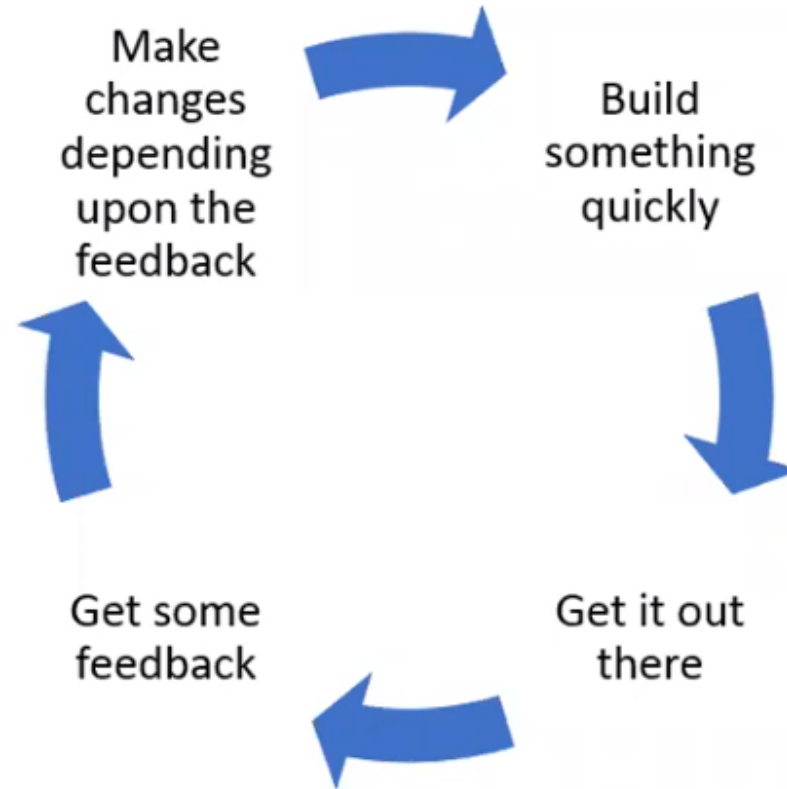
$f(x) = \sin(x)$ has an
absolute maximum
and absolute minimum



$$f(x) = x^3 - 3x^2 + 1$$

GRADIENT DESCENT

Just like Agile
Methodology



GRADIENT DESCENT

Lets have some function $J(\vartheta)$

Want to $\min_{\vartheta} J(\vartheta)$

Algorithm:

- initialize ϑ 's randomly
 - keep chaining ϑ 's to reduce $J(\vartheta)$
- until we hopefully end up at a minimum

GRADIENT DESCENT

Lets have some function $J(\vartheta)$

Want to $\min_{\vartheta} J(\vartheta)$

Algorithm:

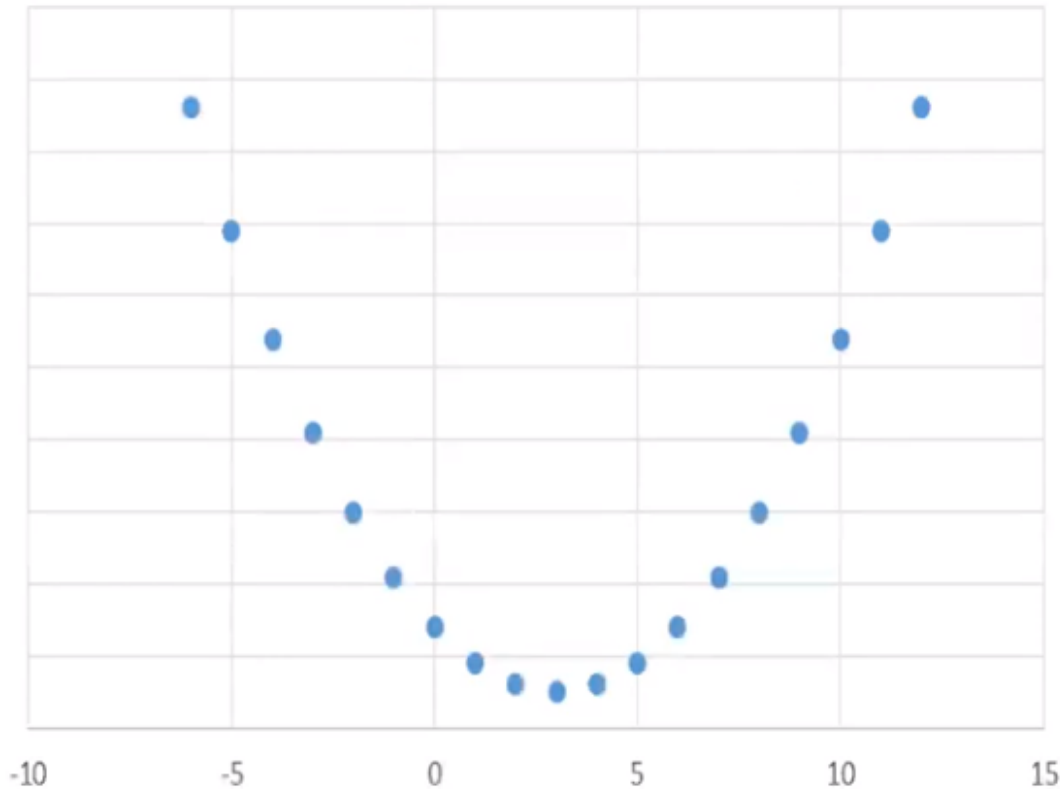
- initialize ϑ 's randomly
- repeat until convergence {

$$\vartheta_i := \vartheta_i - \alpha \frac{\partial}{\partial \vartheta_i} J(\vartheta_i)$$

GRADIENT DESCENT

$$J(\vartheta_1) = (\vartheta_1 - 3)^2 + 5$$

ϑ_1	$J(\vartheta_1)$
0	14
1	9
-1	21
2	6
-2	30
3	5
-3	41
4	6
-4	54
5	9
-5	69
6	14
-6	86
7	21
8	30
9	41
10	54
11	69
12	86
13	105



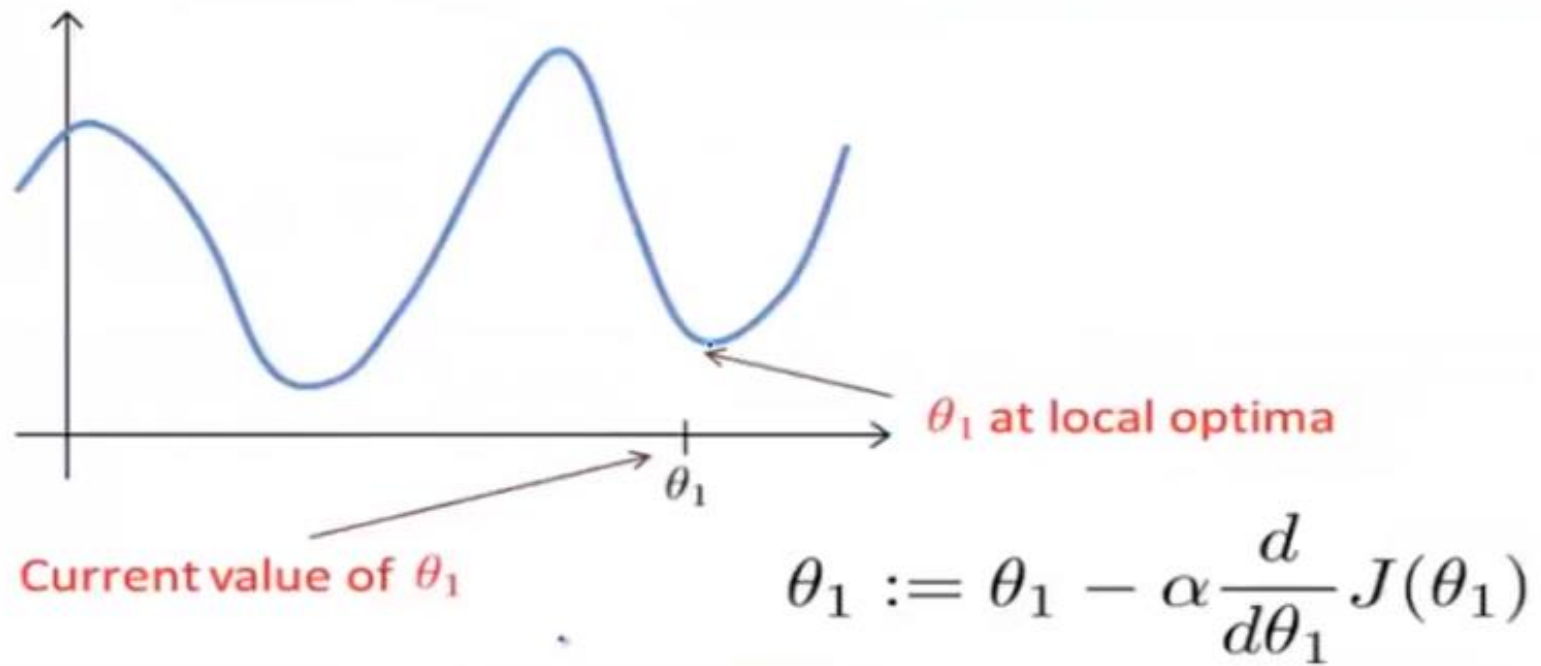
$$\vartheta_1 := \vartheta_1 - \alpha \frac{\partial}{\partial \vartheta_1} J(\vartheta_1)$$

$$\frac{\partial}{\partial \vartheta_1} J(\vartheta_1) = 2(\vartheta_1 - 3)$$

$$\bar{\alpha} = 0.1$$

If $\vartheta_1 = 10$

STUCK AT LOCAL OPTIMA

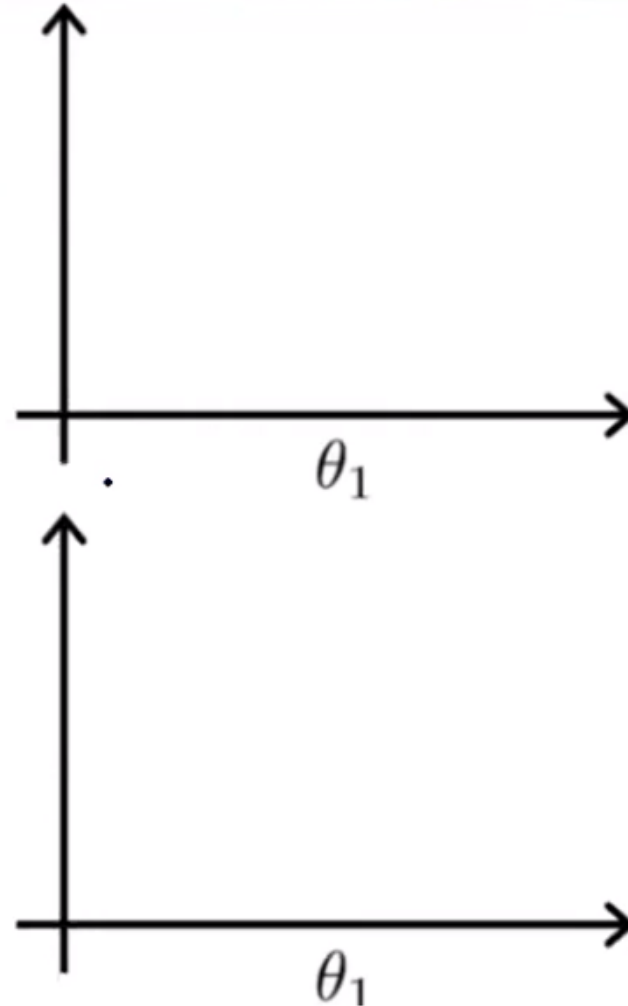


IMPACT OF LEARNING RATE

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow.

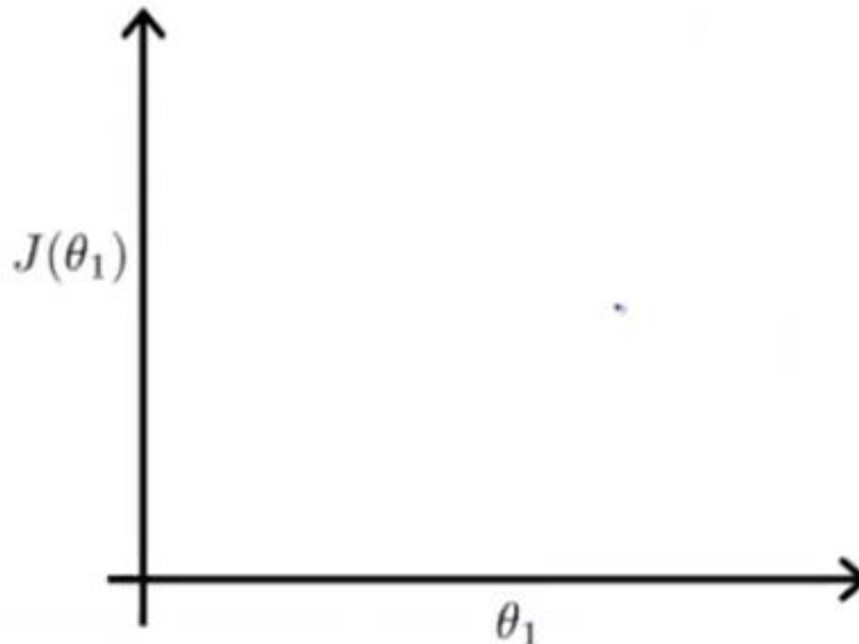
If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



Gradient descent can converge to a local minimum, even with the learning rate α fixed.

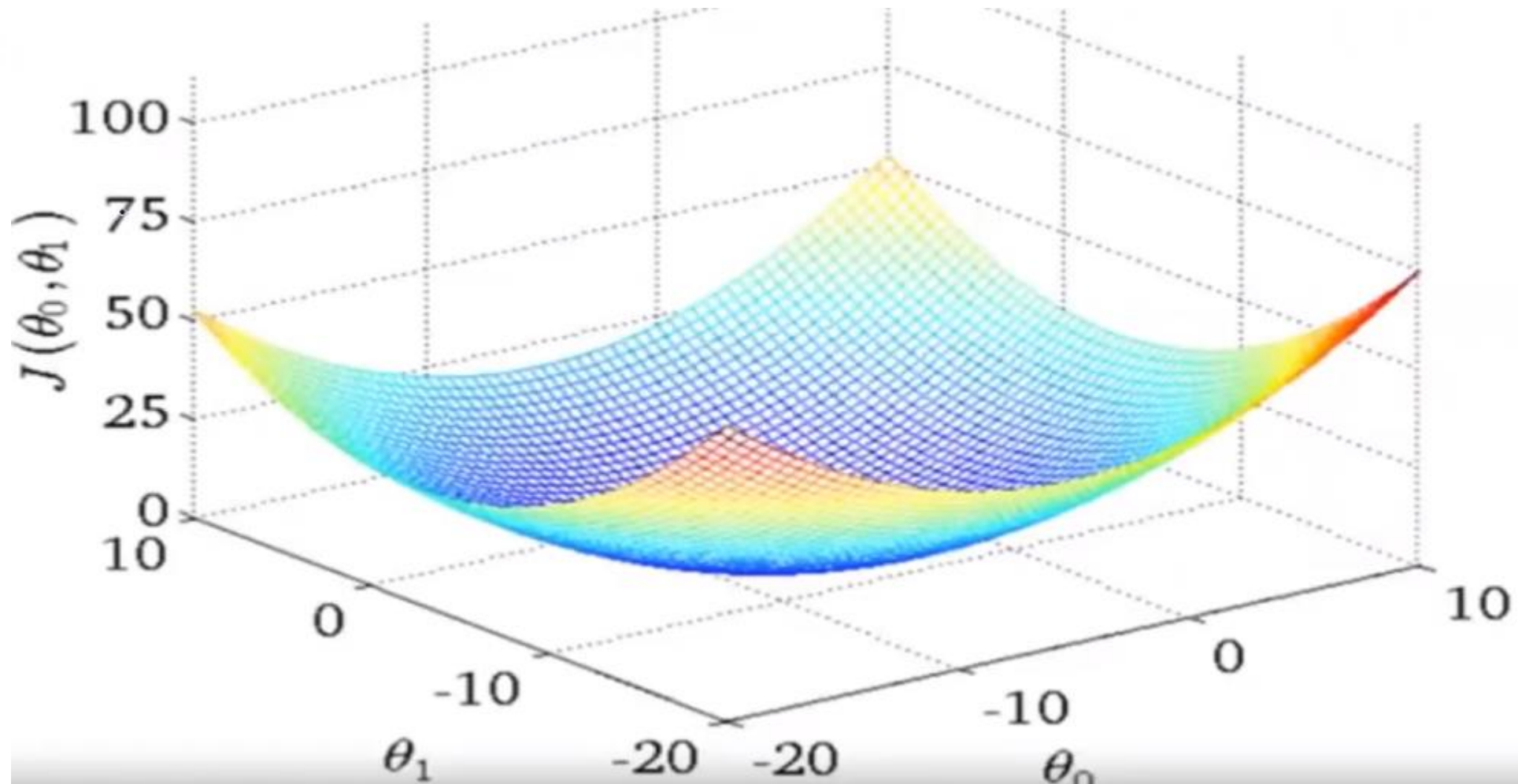
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.



Cost function: $J(\vartheta_0, \vartheta_1)$

$$\min_{\vartheta_0, \vartheta_1} J(\vartheta_0, \vartheta_1)$$



IMPLEMENTATION

Algorithm:

- initialize ϑ 's randomly
- repeat until convergence {

$$\vartheta_i := \vartheta_i - \alpha \frac{\partial}{\partial \vartheta_i} J(\vartheta_0, \vartheta_1)$$

Correct: Simultaneous Update

$$temp_0 := \vartheta_0 - \alpha \frac{\partial}{\partial \vartheta_0} J(\vartheta_0, \vartheta_1)$$

$$temp_1 := \vartheta_1 - \alpha \frac{\partial}{\partial \vartheta_1} J(\vartheta_0, \vartheta_1)$$

$$\vartheta_0 := temp_0$$

$$\vartheta_1 := temp_1$$

Cost function: $J(\vartheta_0, \vartheta_1)$

$$\min_{\vartheta_0, \vartheta_1} J(\vartheta_0, \vartheta_1)$$

Incorrect

$$temp_0 := \vartheta_0 - \alpha \frac{\partial}{\partial \vartheta_0} J(\vartheta_0, \vartheta_1)$$

$$\vartheta_0 := temp_0$$

$$temp_1 := \vartheta_1 - \alpha \frac{\partial}{\partial \vartheta_1} J(\vartheta_0, \vartheta_1)$$

$$\vartheta_1 := temp_1$$

SIGMOID DERIVATIVE

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

$$\sigma'(x) = \frac{d}{dx}\sigma(x) = \sigma(x)(1 - \sigma(x))$$

BINARY CROSS ENTROPY LOSS

$$L_{\text{CE}}(\hat{y}, y) = -\frac{1}{m} \sum_{i=1}^m \mathbf{y} \log(\hat{\mathbf{y}}) + (1 - \mathbf{y}) \log(1 - \hat{\mathbf{y}})$$

$$\hat{\mathbf{y}} = \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\mathbf{z} = \mathbf{w}_0 + \mathbf{x}_1 * \mathbf{w}_1 + \mathbf{x}_2 * \mathbf{w}_2 + \cdots + \mathbf{x}_k * \mathbf{w}_k = \mathbf{W}^T \mathbf{X}$$

DERIVATIVE OF LOSS FUNCTION

$$\frac{\partial L(W)}{\partial W} = \frac{\partial L(W)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial W}$$

$$\frac{\partial L(W)}{\partial \hat{y}} = -\left(\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right) = \frac{\hat{y} - y}{\hat{y}(1-\hat{y})}$$

$$\frac{\partial \hat{y}}{\partial z} = \hat{y}(1-\hat{y})$$

$$\frac{\partial z}{\partial W} = X$$

$$\frac{\partial L(W)}{\partial W} = \frac{\hat{y} - y}{\hat{y}(1-\hat{y})} \hat{y}(1-\hat{y})X = (\hat{y} - y)X$$

LOGISTIC REGRESSION

- Important analytic tool in natural and social sciences
- Baseline supervised machine learning tool for classification
- Is also the foundation of neural networks

LOGISTIC REGRESSION

- Instead of our output vector y being a continuous range of values, it will only be 0 or 1.
 $y \in \{0, 1\}$
- Where 0 is usually taken as the "negative class" and 1 as the "positive class", but you are free to assign any representation to it.
- One method is to use linear regression and map all predictions greater than 0.5 as a 1 and all less than 0.5 as a 0.
- This method doesn't work well because classification is not actually a linear function.

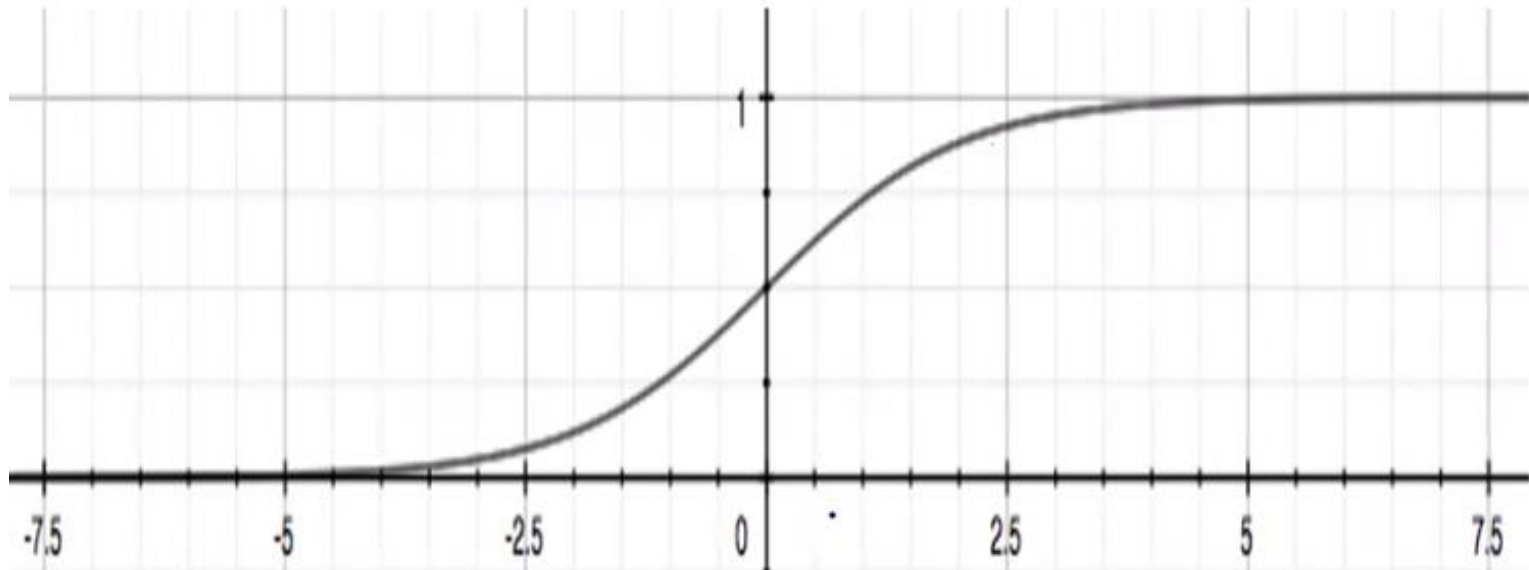
LOGISTIC REGRESSION

- Our hypothesis should satisfy:
- $0 \leq h\theta(x) \leq 1$
- Our new form uses the “sigmoid function” also called the “Logistic Function”

$$h\theta(x) = g(\theta^T x)$$

$$z = \theta^T x$$

$$g(z) = 1 / (1 + e^{-z})$$



z	sig(z)
-2	0.12
-1.5	0.18
-1	0.27
-0.5	0.38
0	0.50
0.5	0.62
1	0.73
1.5	0.82
2	0.88
2.5	0.92

LOGISTIC REGRESSION

- We start with our old hypothesis (linear regression), except that we want to restrict the range to 0 and 1.
- This is accomplished by plugging $\vartheta^T x$ into the Logistic Function.
 - $h\vartheta$ will give us the **probability** that our output is 1.
- For example, $h\vartheta(x)=0.7$ gives us the probability of 70% that our output is 1.
 - $h\vartheta(x) = P(y=1 | x; \vartheta) = 1 - P(y=0 | x; \vartheta)$
 - $P(y=0 | x; \vartheta) + P(y=1 | x; \vartheta) = 1$
- Our probability that our prediction is 0 is just the complement of our probability that it is 1 (e.g. if probability that it is 1 is 70%, then the probability that it is 0 is 30%).

DECISION BOUNDARY

- In order to get our discrete 0 or 1 classification, we can translate the output of the hypothesis function as follows:
 - $h\vartheta(x) \geq 0.5 \rightarrow y = 1$
 - $h\vartheta(x) < 0.5 \rightarrow y = 0$
- The way our logistic function g behaves is that when its input is greater than or equal to zero, its output is greater than or equal to 0.5:
 - $g(z) \geq 0.5$ when $z \geq 0$
- From these statements we can now say:
 - $\vartheta^T x \geq 0 \Rightarrow y = 1$
 - $\vartheta^T x < 0 \Rightarrow y = 0$

The **decision boundary** is the line that separates the area where $y=0$ and where $y=1$. it is created by our hypothesis function.

DECISION BOUNDARY

- $\Theta = \begin{bmatrix} 5 \\ -1 \\ 0 \end{bmatrix}$

$$y=1 \text{ if } 5 + (-1) * x_1 + 0 * x_2 \geq 0$$

$$5 - x_1 \geq 0$$

$$-x_1 \geq -5$$

$$x_1 \leq 5$$

In this case, our decision boundary is a straight vertical line placed on the graph where $x_1 = 5$, and everything to the left of that denotes $y=1$, while everything to the right denotes $y=0$.

- Again, the input to the sigmoid function $g(z)$ (e.g. $\vartheta^T X$) doesn't need to be linear, and could be a function that describes a circle (e.g. $z = \vartheta_0 + \vartheta_1 x_1^2 + \vartheta_2 x_2^2$) or any shape to fit our data.

COMPONENTS OF A PROBABILISTIC MACHINE LEARNING CLASSIFIER

Given m input/output pairs $(x^{(i)}, y^{(i)})$:

1. A **feature representation** of the input. For each input observation $x^{(i)}$, a vector of features $[x_1, x_2, \dots, x_n]$. Feature j for input $x^{(i)}$ is x_j , more completely $x_j^{(i)}$, or sometimes $f_j(x)$.
2. A **classification function** that computes \hat{y} , the estimated class, via $p(y|x)$, like the **sigmoid** or **softmax** functions.
3. An objective function for learning, like **cross-entropy loss**.
4. An algorithm for optimizing the objective function: **stochastic gradient descent**.

THE TWO PHASES OF LOGISTIC REGRESSION

- **Training:** we learn weights w and b using **stochastic gradient descent** and **cross-entropy loss**.
- **Test:** Given a test example x we compute $p(y|x)$ using learned weights w and b , and return whichever label ($y = 1$ or $y = 0$) is higher probability



Classification in Logistic Regression



TEXT CLASSIFICATION: DEFINITION

- *Input:*

- a document x
- a fixed set of classes $C = \{c_1, c_2, \dots, c_J\}$

- *Output:* a predicted class $\hat{y} \in C$

BINARY CLASSIFICATION IN LOGISTIC REGRESSION

- Given a series of input/output pairs:
 - $(x^{(i)}, y^{(i)})$
- For each observation $x^{(i)}$
 - We represent $x^{(i)}$ by a **feature vector** $[x_1, x_2, \dots, x_n]$
 - We compute an output: a predicted class $\hat{y}^{(i)} \in \{0, 1\}$

FEATURES IN LOGISTIC REGRESSION

- For feature x_i , weight w_i tells is how important is x_i
 - x_i = "review contains 'awesome'": $w_i = +10$
 - x_j = "review contains 'abysmal'": $w_j = -10$
 - x_k = "review contains 'mediocre'": $w_k = -2$

LOGISTIC REGRESSION FOR ONE OBSERVATION x

- Input observation: vector $x = [x_1, x_2, \dots, x_n]$
- Weights: one per feature: $W = [w_1, w_2, \dots, w_n]$
 - Sometimes we call the weights $\theta = [\theta_1, \theta_2, \dots, \theta_n]$
- Output: a predicted class $\hat{y} \in \{0, 1\}$

(multinomial logistic regression: $\hat{y} \in \{0, 1, 2, 3, 4\}$)

HOW TO DO CLASSIFICATION

- For each feature x_i , weight w_i tells us importance of x_i
 - (Plus we'll have a bias b)
- We'll sum up all the weighted features and the bias

$$Z = \sum_{i=1}^n w_i x_i + b$$

$$Z = w \cdot x + b$$

- If this sum is high, we say $y=1$; if low, then $y=0$

BUT WE WANT A PROBABILISTIC CLASSIFIER

- We need to formalize “sum is high”.
- We’d like a principled classifier that gives us a probability, just like Naive Bayes did
- We want a model that can tell us:

$$p(y=1|x; \theta)$$

$$p(y=0|x; \theta)$$

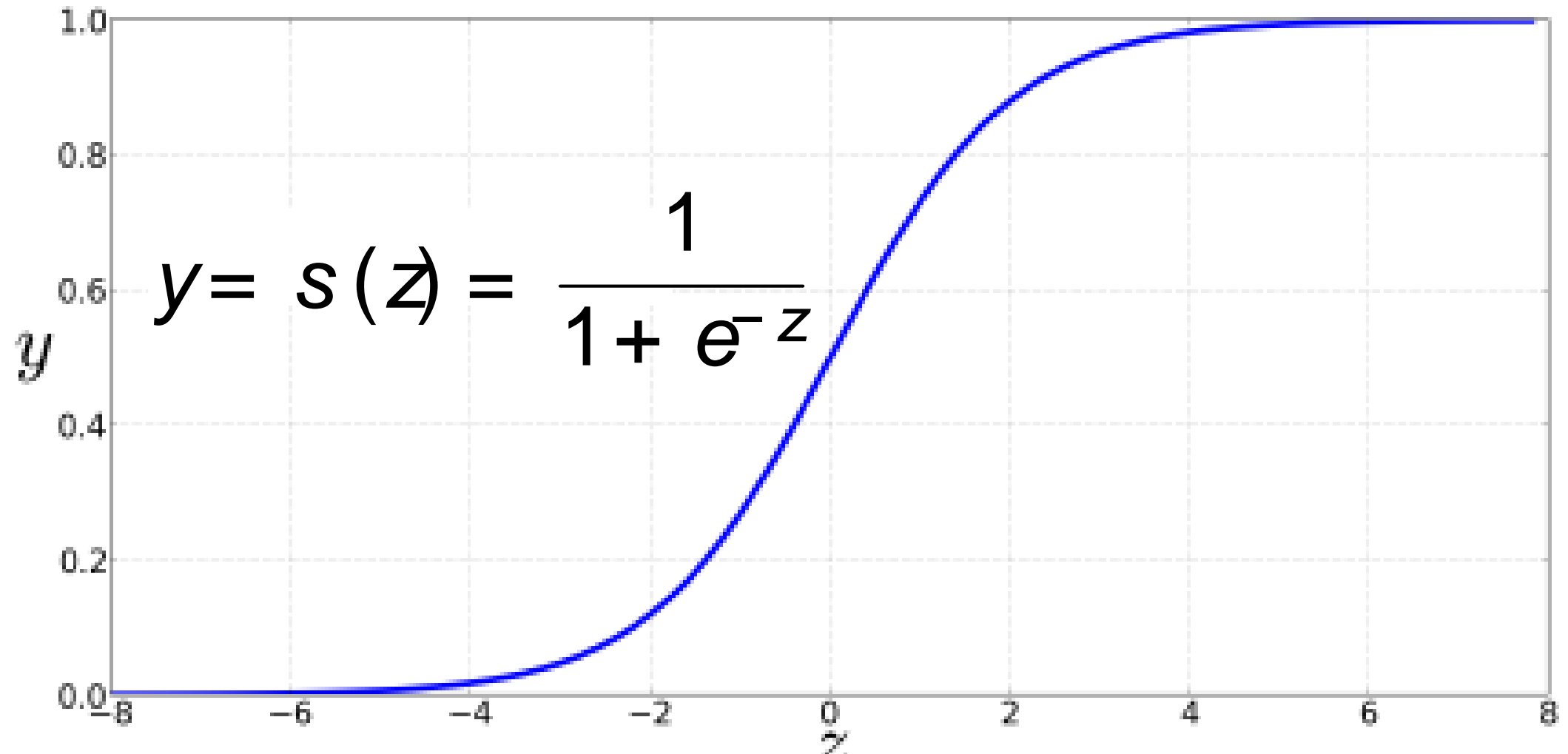
THE PROBLEM: z ISN'T A PROBABILITY, IT'S JUST A NUMBER!

$$z = w \cdot x + b$$

$$y = s(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + \exp(-z)}$$

- Solution: use a function of z that goes from 0 to 1

THE VERY USEFUL SIGMOID OR LOGISTIC FUNCTION



IDEA OF LOGISTIC REGRESSION

- We'll compute $w \cdot x + b$
- And then we'll pass it through the sigmoid function:
- $\sigma(w \cdot x + b)$
- And we'll just treat it as a probability

MAKING PROBABILITIES WITH SIGMOIDS

$$\begin{aligned} P(y = 1) &= \sigma(w \cdot x + b) \\ &= \frac{1}{1 + \exp(-(w \cdot x + b))} \end{aligned}$$

$$\begin{aligned} P(y = 0) &= 1 - \sigma(w \cdot x + b) \\ &= 1 - \frac{1}{1 + \exp(-(w \cdot x + b))} \\ &= \frac{\exp(-(w \cdot x + b))}{1 + \exp(-(w \cdot x + b))} \end{aligned}$$

BY THE WAY:

$$P(y = 0) = 1 - \sigma(w \cdot x + b) = \sigma(-(w \cdot x + b))$$

$$= 1 - \frac{1}{1 + \exp(-(w \cdot x + b))}$$

$$= \frac{\exp(-(w \cdot x + b))}{1 + \exp(-(w \cdot x + b))}$$

Because

$$1 - \sigma(x) = \sigma(-x)$$

TURNING A PROBABILITY INTO A CLASSIFIER

$$\hat{y} = \begin{cases} 1 & \text{if } P(y = 1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

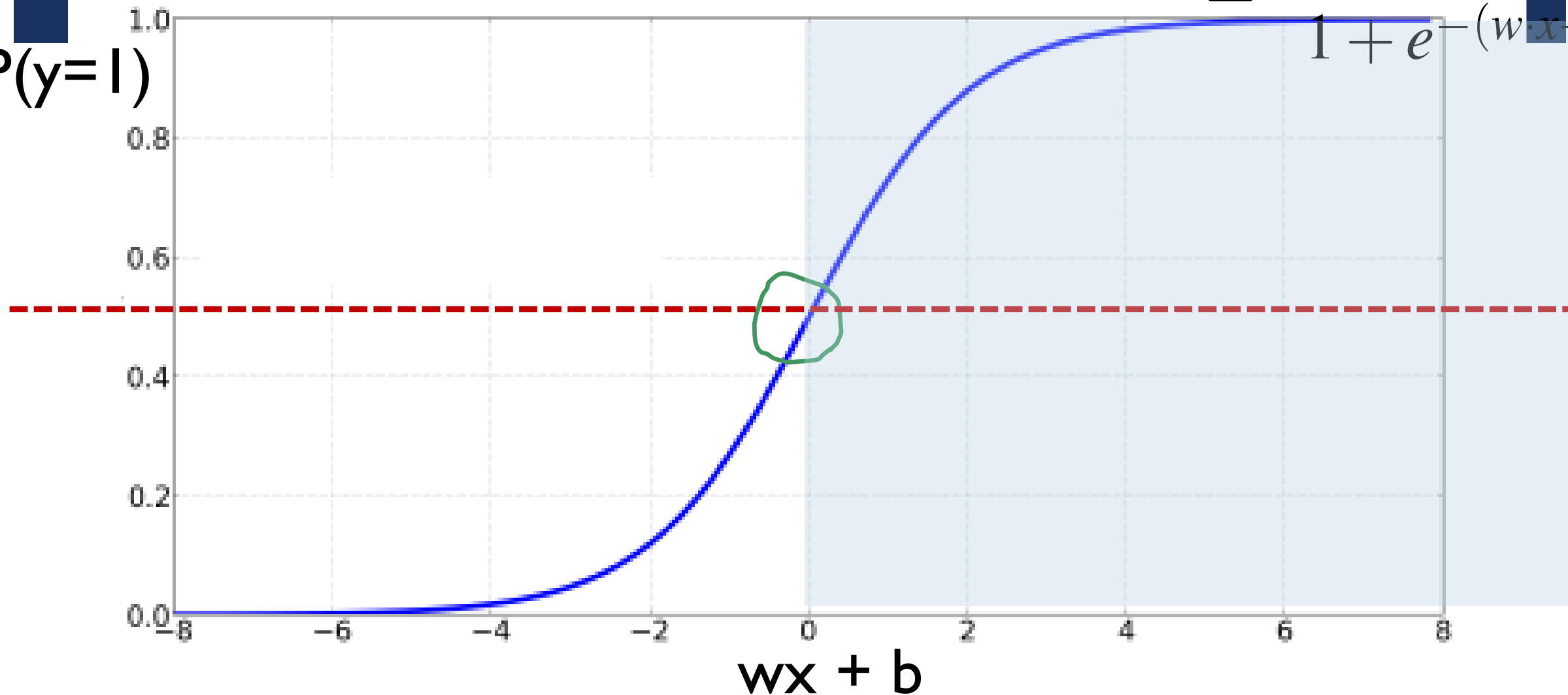
0.5 here is called the **decision boundary**

$$P(y=1) = \sigma(w \cdot x + b)$$

THE PROBABILISTIC CLASSIFIER

$P(y=1)$

$$= \frac{1}{1 + e^{-(w \cdot x + b)}}$$



TURNING A PROBABILITY INTO A CLASSIFIER

$$\hat{y} = \begin{cases} 1 & \text{if } P(y = 1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases} \quad \begin{array}{l} \text{if } w \cdot x + b > 0 \\ \text{if } w \cdot x + b \leq 0 \end{array}$$



Text example on sentiment classification



SENTIMENT EXAMPLE: DOES $Y=1$ OR $Y=0$?

■ It's hokey . There are virtually no surprises , and the writing is second-rate . So why was it so enjoyable ? For one thing , the cast is great . Another nice touch is the music . I was overcome with the urge to get off the couch and start dancing . It sucked me in , and it'll do the same to you .

It's hokey. There are virtually no surprises, and the writing is second-rate. So why was it so enjoyable? For one thing, the cast is great. Another nice touch is the music. I was overcome with the urge to get off the couch and start dancing. It sucked me in, and it'll do the same to you.

$x_2=2$
 $x_3=1$
 $x_1=3$ $x_5=0$ $x_6=4.19$ $x_4=3$

Var	Definition	Value in Fig. 5.2
x_1	count(positive lexicon) \in doc)	3
x_2	count(negative lexicon) \in doc)	2
x_3	$\begin{cases} 1 & \text{if "no" } \in \text{ doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns \in doc)	3
x_5	$\begin{cases} 1 & \text{if "!" } \in \text{ doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(66) = 4.19$

CLASSIFYING SENTIMENT FOR INPUT X

Var	Definition	Value
x_1	count(positive lexicon) \in doc)	3
x_2	count(negative lexicon) \in doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns \in doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(66) = 4.19$

Suppose $w = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$

$b = 0.1$

CLASSIFYING SENTIMENT FOR INPUT X

$$\begin{aligned} p(+|x) &= P(Y=1|x) = s(w \cdot x + b) \\ &= s([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.19] + 0.1) \\ &= s(.833) \\ &= 0.70 \end{aligned}$$

$$\begin{aligned} p(-|x) &= P(Y=0|x) = 1 - s(w \cdot x + b) \\ &= 0.30 \end{aligned}$$

CLASSIFICATION IN (**BINARY**) LOGISTIC REGRESSION: SUMMARY

■ Given:

- a set of classes: (+ sentiment, - sentiment)
- a vector \mathbf{x} of features $[x_1, x_2, \dots, x_n]$
 - $x_1 = \text{count}(\text{"awesome"})$
 - $x_2 = \log(\text{number of words in review})$
- A vector \mathbf{w} of weights $[w_1, w_2, \dots, w_n]$
 - w_i for each feature f_i

$$\begin{aligned} P(y = 1) &= \sigma(w \cdot x + b) \\ &= \frac{1}{1 + e^{-(w \cdot x + b)}} \end{aligned}$$

WAIT, WHERE DID THE w 'S COME FROM?

- Supervised classification:
 - We know the correct label y (either 0 or 1) for each x .
 - But what the system produces is an estimate, \hat{y}
- We want to set w and b to minimize the **distance** between our estimate $\hat{y}^{(i)}$ and the true $y^{(i)}$.
 - We need a distance estimator: a **loss function** or a **cost function**
 - We need an optimization algorithm to update w and b to minimize the loss.

LEARNING COMPONENTS

- A loss function:

- **cross-entropy loss**

- An optimization algorithm:

- **stochastic gradient descent**

THE DISTANCE BETWEEN \hat{y} AND y

- We want to know how far is the classifier output:

- $\hat{y} = \sigma(w \cdot x + b)$

- from the true output:

- y [= either 0 or 1]

- We'll call this difference:

- $L(\hat{y}, y)$ = how much \hat{y} differs from the true y

INTUITION OF NEGATIVE LOG LIKELIHOOD LOSS = CROSS-ENTROPY LOSS

- A case of conditional maximum likelihood estimation
- We choose the parameters w, b that maximize
 - the log probability
 - of the true y labels in the training data
 - given the observations x

LET'S SEE IF THIS WORKS FOR OUR SENTIMENT EXAMPLE

- We want loss to be:
 - smaller if the model estimate is close to correct
 - bigger if model is confused
- Let's first suppose the true label of this is $y=1$ (positive)

It's hokey . There are virtually no surprises , and the writing is second-rate .
So why was it so enjoyable ? For one thing , the cast is great .
Another nice touch is the music . I was overcome with the urge to get off the couch and start dancing . It sucked me in , and it'll do the same to you .

LET'S SEE IF THIS WORKS FOR OUR SENTIMENT EXAMPLE

- True value is $y=1$. How well is our model doing?

$$\begin{aligned} p(+|x) &= P(Y=1|x) = s(w \cdot x + b) \\ &= s([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.19] + 0.1) \\ &= s(.833) \\ &= 0.70 \end{aligned} \tag{5.6}$$

- Pretty well! What's the loss?

$$\begin{aligned} L_{\text{CE}}(\hat{y}, y) &= -[y \log \sigma(w \cdot x + b) + (1 - y) \log (1 - \sigma(w \cdot x + b))] \\ &= -[\log \sigma(w \cdot x + b)] \\ &= -\log(.70) \\ &= .36 \end{aligned}$$

LET'S SEE IF THIS WORKS FOR OUR SENTIMENT EXAMPLE

- Suppose true value instead was $y=0$.

$$\begin{aligned} p(-|x) &= P(Y=0|x) = 1 - \sigma(w \cdot x + b) \\ &= 0.30 \end{aligned}$$

- What's the loss?

$$\begin{aligned} L_{\text{CE}}(\hat{y}, y) &= -[y \log \sigma(w \cdot x + b) + (1 - y) \log (1 - \sigma(w \cdot x + b))] \\ &= -[\log (1 - \sigma(w \cdot x + b))] \\ &= -\log (.30) \\ &= 1.2 \end{aligned}$$

LET'S SEE IF THIS WORKS FOR OUR SENTIMENT EXAMPLE

- The loss when model was right (if true $y=1$)

$$\begin{aligned} L_{\text{CE}}(\hat{y}, y) &= -[y \log \sigma(w \cdot x + b) + (1 - y) \log (1 - \sigma(w \cdot x + b))] \\ &= -[\log \sigma(w \cdot x + b)] \\ &= -\log(.70) \\ &= .36 \end{aligned}$$

- Is lower than the loss when model was wrong (if true $y=0$):

$$\begin{aligned} L_{\text{CE}}(\hat{y}, y) &= -[y \log \sigma(w \cdot x + b) + (1 - y) \log (1 - \sigma(w \cdot x + b))] \\ &= -[\log (1 - \sigma(w \cdot x + b))] \\ &= -\log(.30) \\ &= 1.2 \end{aligned}$$

- Sure enough, loss was bigger when model was wrong!



REGULARIZATION



OVERFITTING

- A model that perfectly match the training data has a problem.
- It will also **overfit** to the data, modeling noise
 - A random word that perfectly predicts y (it happens to only occur in one class) will get a very high weight.
 - Failing to generalize to a test set without this word.
- A good model should be able to **generalize**

OVERFITTING



- This movie drew me in, and it'll do the same to you.

Useful or harmless features

X1 = "this"

X2 = "movie"

X3 = "hated"

X4 = "drew me in"



I can't tell you how much I hated this movie. It sucked.

4gram features that just "memorize" training set and might cause problems

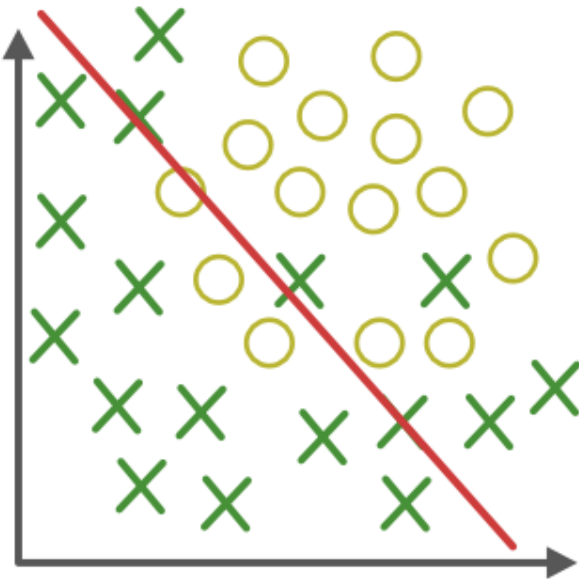
X5 = "the same to you"

X7 = "tell you how much"

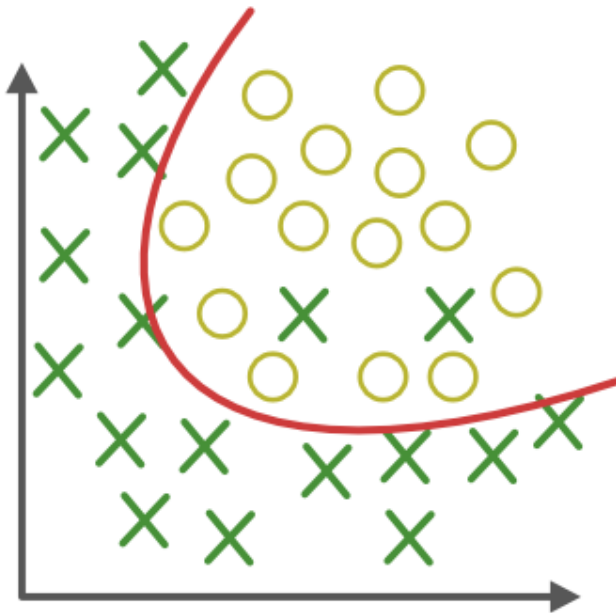
OVERFITTING

- 4-gram model on tiny data will just memorize the data
 - 100% accuracy on the training set
- But it will be surprised by the novel 4-grams in the test data
 - Low accuracy on test set
- Models that are too powerful can **overfit** the data
 - Fitting the details of the training data so exactly that the model doesn't generalize well to the test set
 - How to avoid overfitting?
 - Increase training data
 - Early Stopping
 - Regularization in logistic regression

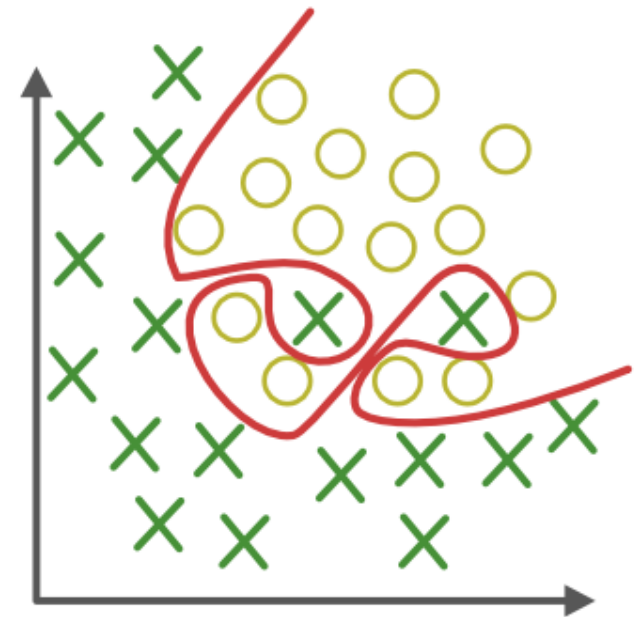
OVERFITTING VS UNDER FITTING



Under-fitting
(too simple to
explain the variance)



Appropriate-fitting



Over-fitting
(forcefitting--too
good to be true)

REGULARIZATION


- A solution for overfitting
- Add a regularization term $R(\theta)$ to the loss function
- Idea: choose an $R(\theta)$ that penalizes large weights
 - fitting the data well with lots of big weights not as good as fitting the data a little less well, with small weights

L2 REGULARIZATION (= RIDGE REGRESSION)

- The sum of the squares of the weights
- The name is because this is the (square of the) **L2 norm** $\|\theta\|_2$, = **Euclidean distance** of θ to the origin.

$$R(\theta) = \|\theta\|_2^2 = \sum_{j=1}^n \theta_j^2$$

- L2 regularized objective function:

$$J(\theta) = \boxed{\frac{1}{m} \sum_{i=1}^m \left[-y^{(i)} \log(h_{\theta}(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) \right]} + \boxed{\frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2}$$


LI REGULARIZATION (= LASSO REGRESSION)

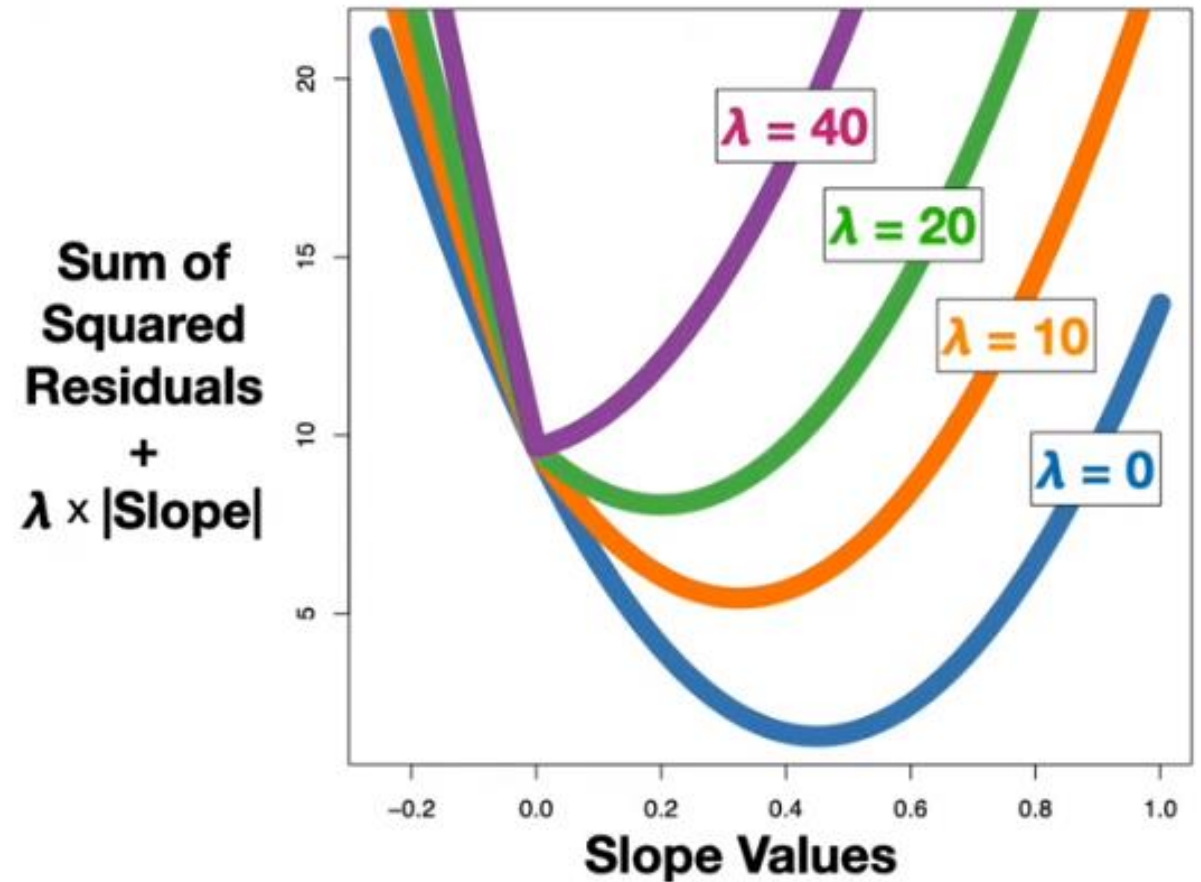
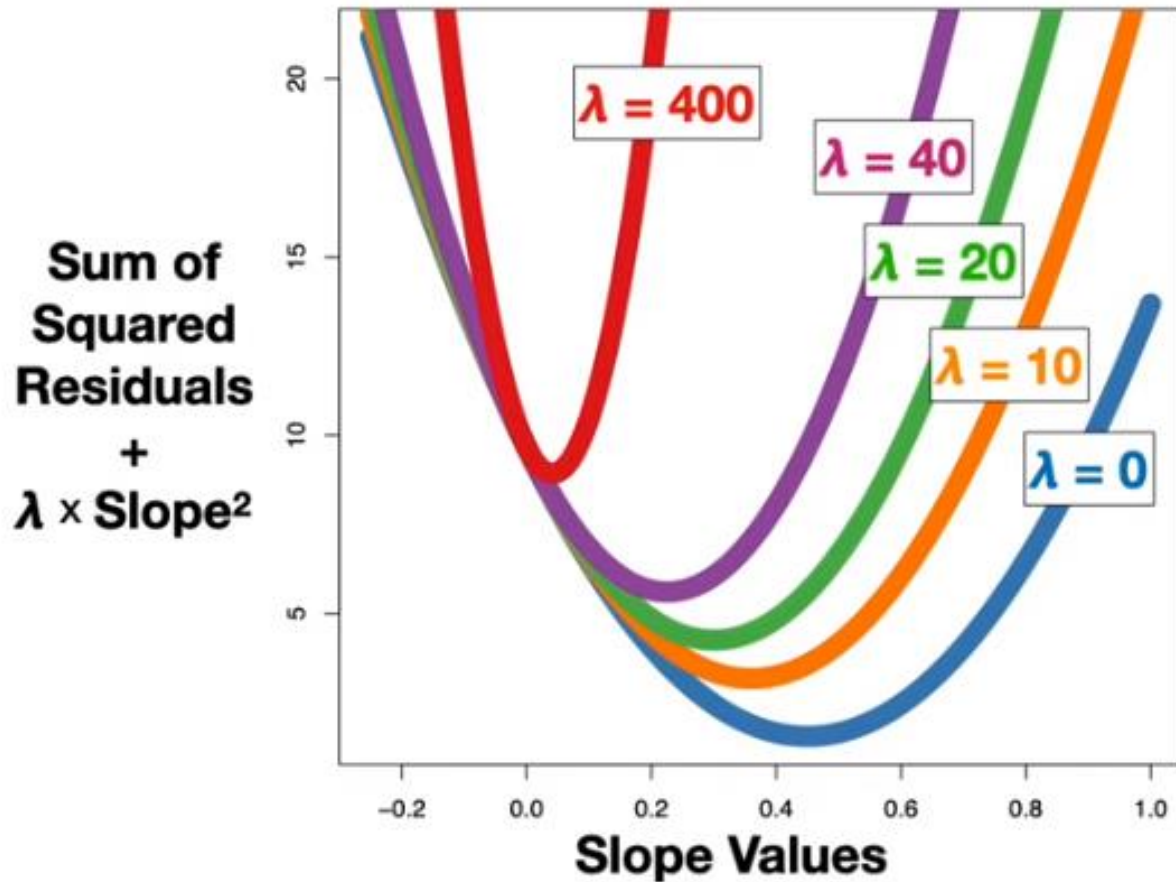
- The sum of the (absolute value of the) weights
- Named after the **L1 norm** $\|W\|_1$, = sum of the absolute values of the weights, = **Manhattan distance**

$$R(\theta) = \|\theta\|_1 = \sum_{i=1}^n |\theta_i|$$

- L1 regularized objective function:

$$J(\theta) = \underbrace{\frac{1}{m} \sum_{i=1}^m \left[-y^{(i)} \log(h_{\theta}(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) \right]}_{\text{Cross-Entropy Loss}} + \frac{\lambda}{m} \sum_{j=1}^n |\theta_j|$$

DIFFERENCE BETWEEN RIDGE AND LASSO





GENERATIVE VS DISCRIMINATIVE CLASSIFIER



GENERATIVE AND DISCRIMINATIVE CLASSIFIERS

- Naive Bayes is a **generative** classifier
- by contrast:
- Logistic regression is a **discriminative** classifier

GENERATIVE AND DISCRIMINATIVE CLASSIFIERS

Suppose we're distinguishing cat from dog images



GENERATIVE CLASSIFIER:

- Build a model of what's in a cat image
 - Knows about whiskers, ears, eyes
 - Assigns a probability to any image:
 - how cat-y is this image?



Also build a model for dog images

Now given a new image:

Run both models and see which one fits better

DISCRIMINATIVE CLASSIFIER

Just try to distinguish dogs from cats



Oh look, dogs have collars!
Let's ignore everything else

FINDING THE CORRECT CLASS C FROM A DOCUMENT D IN GENERATIVE VS DISCRIMINATIVE CLASSIFIERS

■ Naive Bayes

$$\hat{c} = \operatorname{argmax}_{c \in C} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

■ Logistic Regression

$$\hat{c} = \operatorname{argmax}_{c \in C} \overbrace{P(c/d)}^{\text{posterior}}$$



SIMILARITY BETWEEN TEXT



TEXT SIMILARITY METRIC IN NLP

- How the two text documents close to each other in terms of their context or meaning
- Various text similarity metric
 - cosine similarity
 - Euclidean distance
 - Jaccard similarity

COSINE SIMILARITY

- Cosine similarity is one of the metric to measure the text-similarity between two documents irrespective of their size in Natural language Processing.
- The Cosine similarity of two documents range from 0 to 1
- Two vectors have the same orientation if Cosine similarity score is 1.
- The value closer to 0 indicates that the two documents have less similarity.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

EXAMPLE

```
doc_1 = "Data is the oil of the digital economy"  
doc_2 = "Data is a new oil"
```

Vector representation of the document

```
doc_1_vector = [1, 1, 1, 1, 0, 1, 1, 2]  
doc_2_vector = [1, 0, 0, 1, 1, 0, 1, 0]
```

	data	digital	economy	is	new	of	oil	the
doc_1	1	1	1	1	0	1	1	2
doc_2	1	0	0	1	1	0	1	0

$$A \cdot B = \sum_{i=1}^n A_i B_i$$

$$= (1 * 1) + (1 * 0) + (1 * 0) + (1 * 1) + (0 * 1) + (1 * 0) + (1 * 1) + (2 * 0)$$

$$= 3$$

$$\sqrt{\sum_{i=1}^n A_i^2} = \sqrt{1 + 1 + 1 + 1 + 0 + 1 + 1 + 4} = \sqrt{10}$$

$$\sqrt{\sum_{i=1}^n B_i^2} = \sqrt{1 + 0 + 0 + 1 + 1 + 0 + 1 + 0} = \sqrt{4}$$

$$\text{cosine similarity} = \cos\theta = \frac{A \cdot B}{|A||B|} = \frac{3}{\sqrt{10} * \sqrt{4}} = 0.4743$$

Cosine Similarity between doc_1 and doc_2 is 0.47

JACCARD SIMILARITY

- Jaccard Similarity defined as an intersection of two documents divided by the union of that two documents that refer to the number of common words over a total number of words
- The Jaccard Similarity score is in a range of 0 to 1. If the two documents are identical, Jaccard Similarity is 1. The Jaccard similarity score is 0 if there are no common words between two documents.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

EXAMPLE

$$\begin{aligned} J(doc_1, doc_2) &= \frac{\{'data', 'is', 'the', 'new', 'oil', 'of', 'digital', 'economy'\} \cap \{'data', 'is', 'a', 'new', 'oil'\}}{\{'data', 'is', 'the', 'new', 'oil', 'of', 'digital', 'economy'\} \cup \{'data', 'is', 'a', 'new', 'oil'\}} \\ &= \frac{\{'data', 'is', 'new', 'oil'\}}{\{'data', 'a', 'of', 'is', 'economy', 'the', 'new', 'digital', 'oil'\}} \\ &= \frac{4}{9} = 0.444 \end{aligned}$$

