CIS 678 Machine Learning

ML Introduction: Linear Regression (part 3)

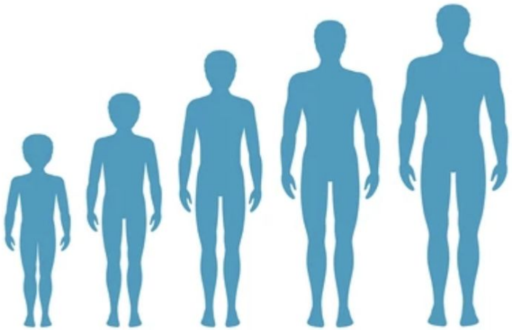
What we'd like to accomplish today

- ~~General Concepts: ***Straight Line** to **Linear Regression***~~
- Gradient Descent Algorithm
  - ➜ A simple two parameter **Linear Regression** model
  - ➜ Hands on **Notebook implementation**
- QA

We will be learning today about Regression

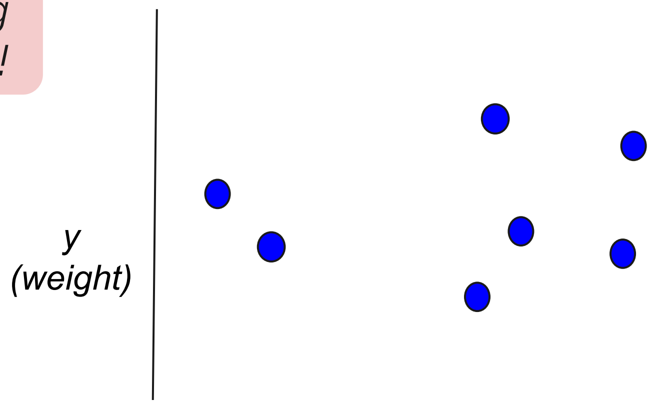
- *Person's weight :* $y \in R$

$$f\,(y|x=height)$$



Select one among a set of Models!

Select one among infinite (∞) Models!



Model, θ: $\{\beta_0, \beta_1\}$

Model, θ: $\{\beta_0, \beta_1\}$

Model, θ: $\{\beta_0, \beta_1\}$

Model, θ: $\{\beta_0, \beta_1\}$

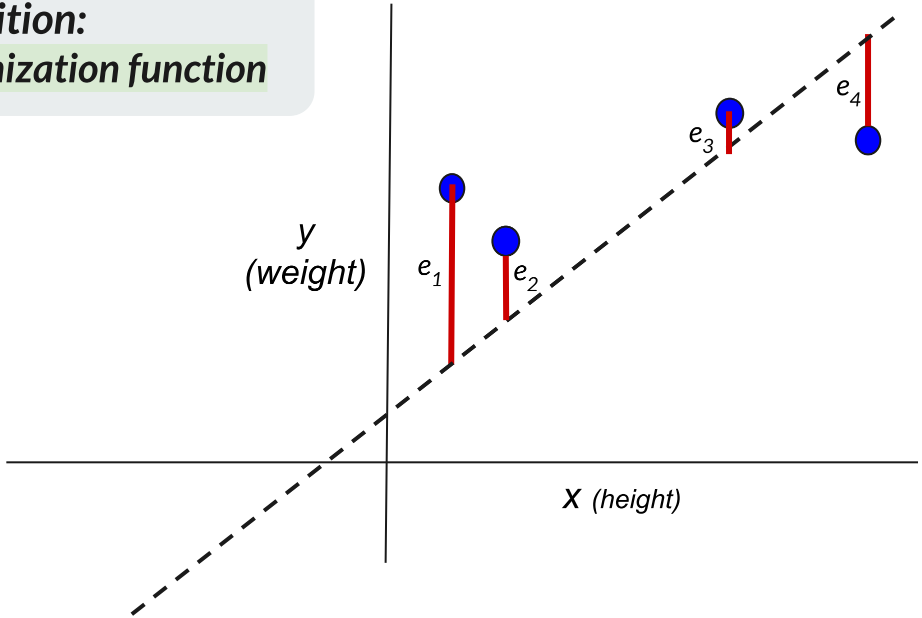
________ ***How to Generalize the Idea for***

- any number of points, and/or

- any models (remember, we have infinite number of possible models)

Linear Regression

Definition:
Optimization function



Prediction function & Model

$$\hat{y} = \beta_0 + \beta_1 x$$

$$\Theta = \{\beta_0, \beta_1\}$$

Fitting Error

$$\epsilon = |\hat{y} - y|$$

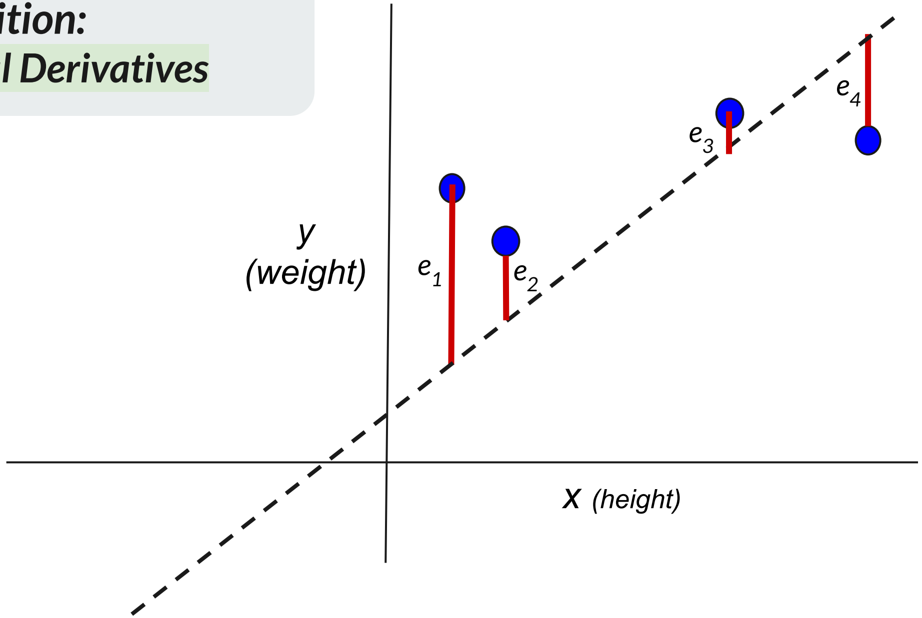
Optimization/loss/error function

$$E_\Theta = \frac{1}{2}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

$$\Theta^* = \text{argmin}_\Theta\, E\{(x_i, y_i)\}_{i=1,\cdots,N}$$

Linear Regression

Definition:
Partial Derivatives



- **Optimization/Loss Function:**

$$E_\Theta = \frac{1}{2}\sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)^2$$

- **Gradient w.r.t.** β_0**:**

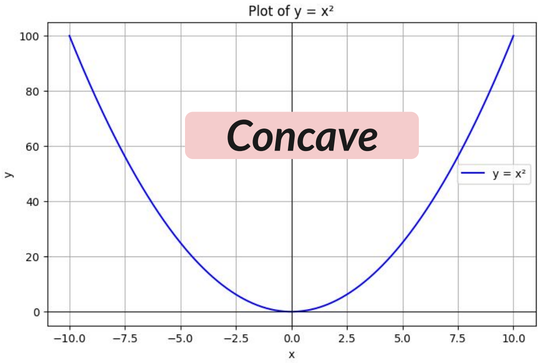
$$\frac{\partial E_\Theta}{\partial \beta_0} = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)$$

- **Gradient w.r.t.** β_1**:**

$$\frac{\partial E_\Theta}{\partial \beta_1} = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)\, x_i$$

Linear Regression

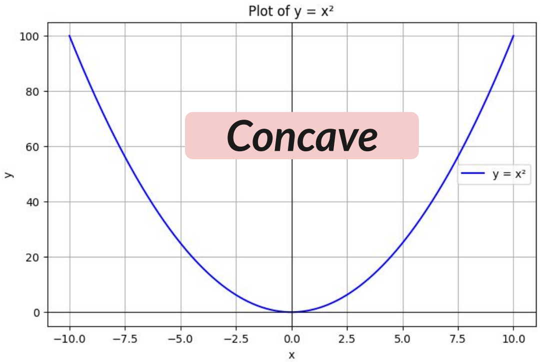
Quadratic functions



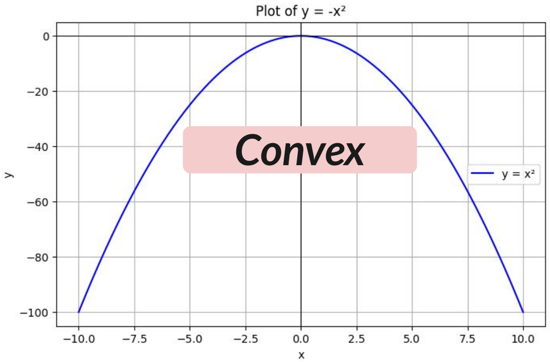
$$y = x^2$$

Linear Regression

Quadratic functions



$$y = x^2$$

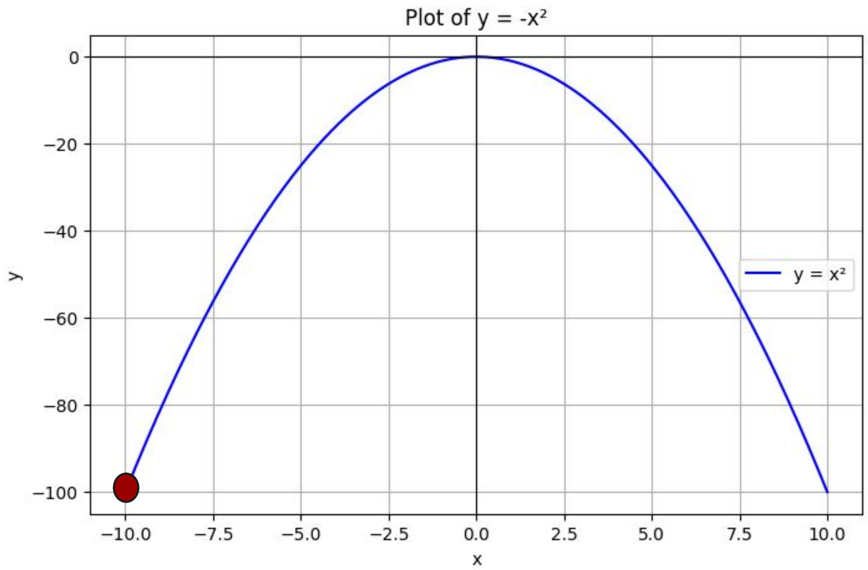


$$y = -x^2$$

Linear Regression

Gradient ascent - General idea:

- *One wants to reach the top starting from the red point*

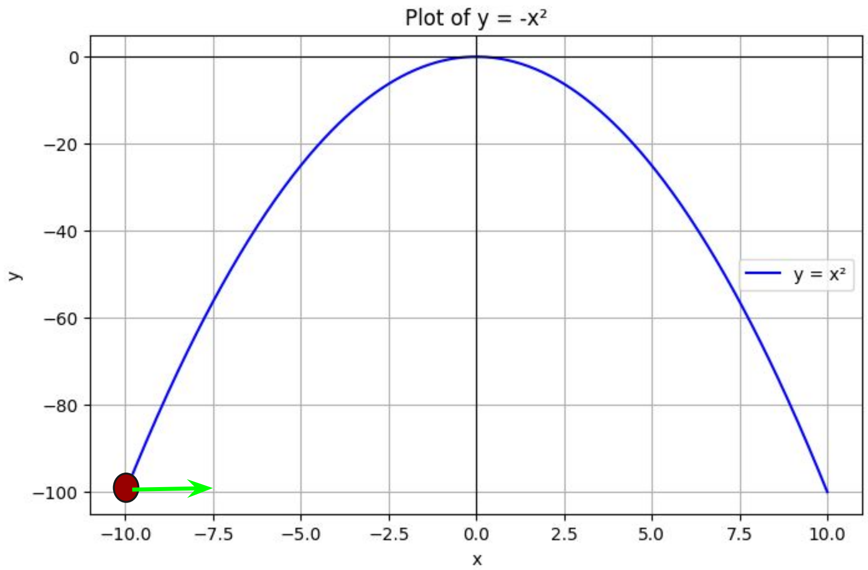


$y = -x^2$ (Quadratic function)

Linear Regression

Gradient ascent - General idea:

- *One wants to reach the top starting from the red point*
- *If we take the derivative, with respect to x,*



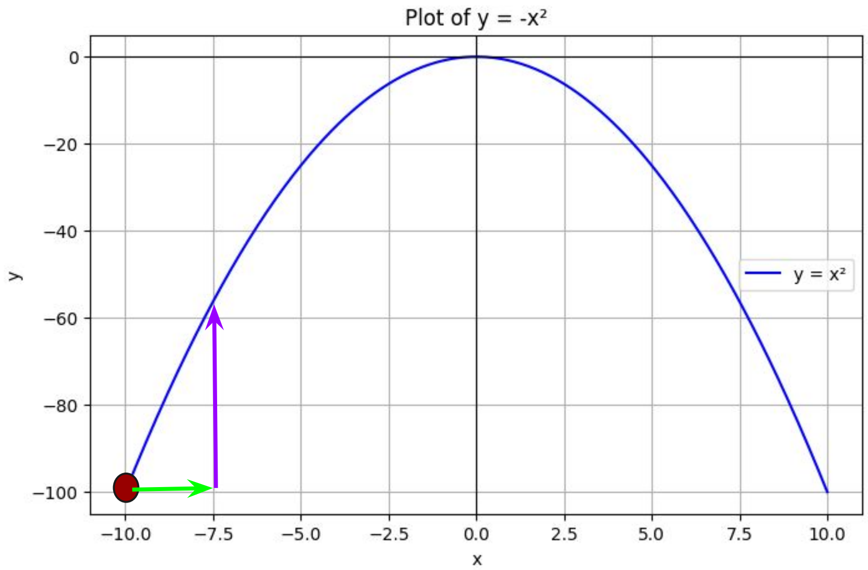
$$y = -x^2 \qquad \text{(Quadratic function)}$$

$$\frac{dy}{dx} = -2x \qquad \text{(First Derivative)}$$

Linear Regression

Gradient ascent - General idea:

- *One wants to reach the top starting from the red point*
- *If we take the derivative, with respect to x,*



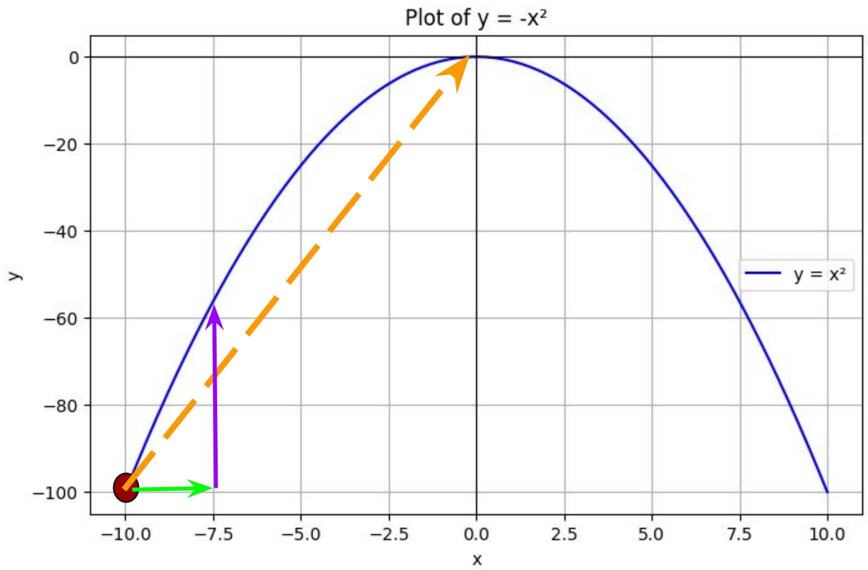
$$y = -x^2 \qquad \text{(Quadratic function)}$$

$$\frac{dy}{dx} = -2x \qquad \text{(First Derivative)}$$

Linear Regression

Gradient ascent - General idea:

- *One wants to reach the top starting from the red point*
- *If we take the derivative, with respect to x, we will know which direction to move*



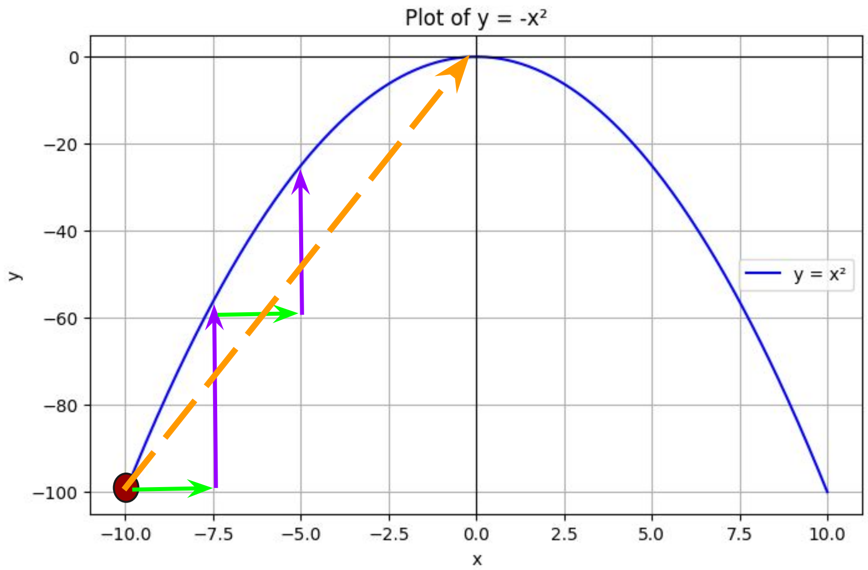
$$y = -x^2 \qquad \text{(Quadratic function)}$$

$$\frac{dy}{dx} = -2x \qquad \text{(First Derivative)}$$

Linear Regression

Gradient ascent - General idea:

- *One wants to reach the top starting from the red point*
- *If we take derivatives, with respect to x, we will know which direction to move*
- *We keep iterating*



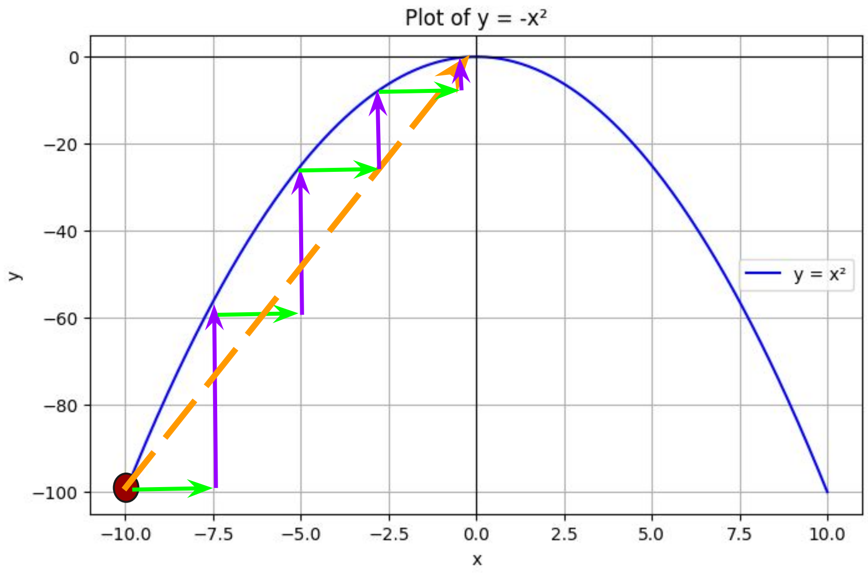
$$y = -x^2 \qquad \text{(Quadratic function)}$$

$$\frac{dy}{dx} = -2x \qquad \text{(First Derivative)}$$

Linear Regression

Gradient ascent - General idea:

- *One wants to reach the top starting from the red point*
- *If we take derivatives, with respect to x, we will know which direction to move*
- *We keep iterating*
- *It's like building stairs of equal width and varying height (touching f(x)).*
- *Usually until f(x) ~ 0*
- *The width of each stair is called 'step-size'/'learning-rate in many ML algorithms.*
- *…..*
- *What will happen if the step-size is too big?*
- *What will happen if the step size is too small?*



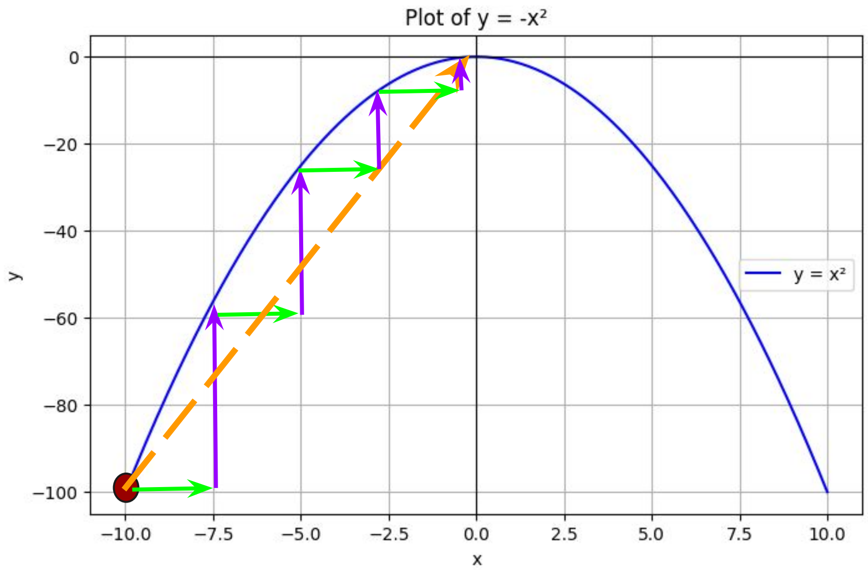
$$y = -x^2 \qquad \text{(Quadratic function)}$$

$$\frac{dy}{dx} = -2x \qquad \text{(First Derivative)}$$

Linear Regression

Gradient ascent - General idea:

- *One wants to reach the top starting from the red point*
- *If we take derivatives, with respect to x, we will know which direction to move*
- *We keep iterating*
- *It's like building stairs of equal width and varying height (touching f(x)).*
- *Usually until f(x) ~ 0*
- *The width of each stair is called 'step-size'/'learning-rate in many ML algorithms.*
- …..
- *What will happen if the step-size is too big?*
- *What will happen if the step size is too small?*



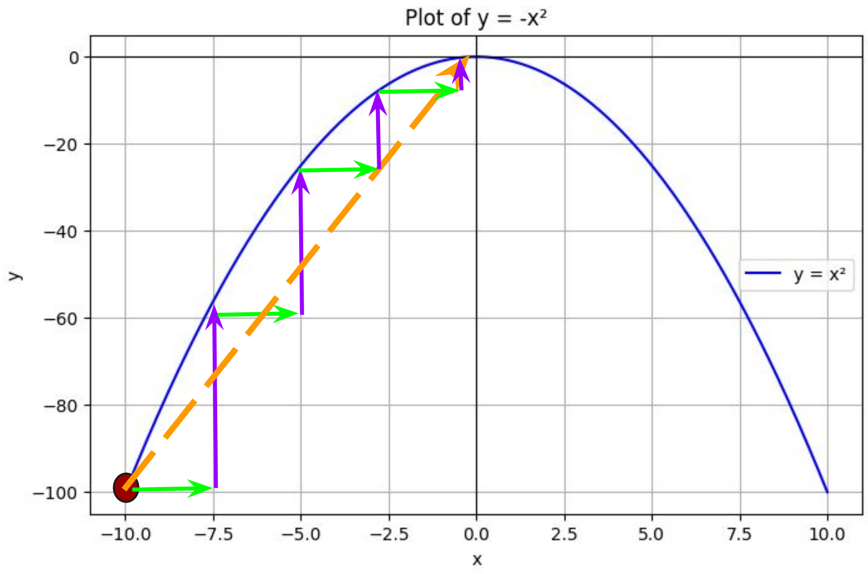
$$y = -x^2 \qquad \text{(Quadratic function)}$$

$$\frac{dy}{dx} = -2x \qquad \text{(First Derivative)}$$

Linear Regression

Gradient ascent - General idea:

- *One wants to reach the top starting from the red point*
- *If we take derivatives, with respect to x, we will know which direction to move*
- *We keep iterating*
- *It's like building stairs of equal width and varying height (touching f(x)).*
- *Usually until f(x) ~ 0*
- *The width of each stair is called 'step-size'/'learning-rate in many ML algorithms.*
- …..
- *What will happen if the step-size is too big?*
- *What will happen if the step size is too small?*



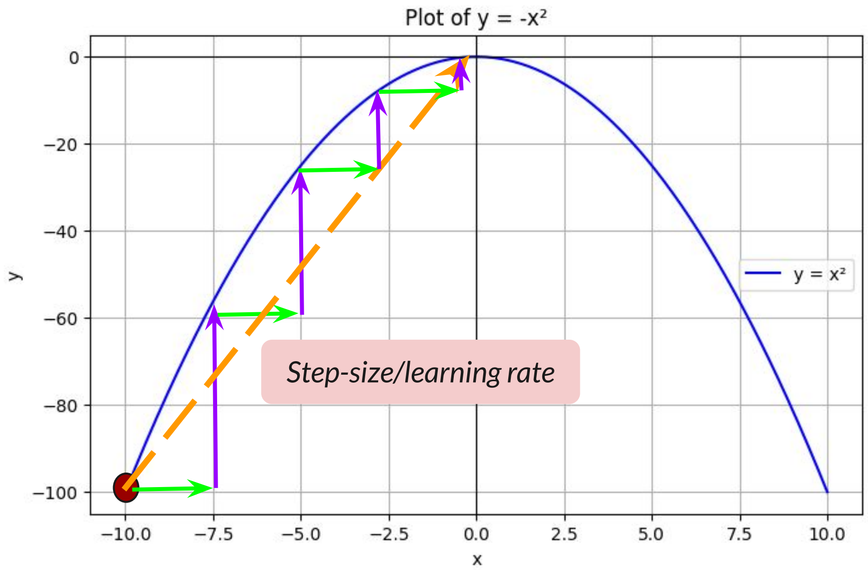
$$y = -x^2 \qquad \text{(Quadratic function)}$$

$$\frac{dy}{dx} = -2x \qquad \text{(First Derivative)}$$

Linear Regression

Gradient ascent - General idea:

- *One wants to reach the top starting from the red point*
- *If we take derivatives, with respect to x, we will know which direction to move*
- *We keep iterating*
- *It's like building stairs of equal width and varying height (touching f(x)).*
- *Usually until f(x) ~ 0*
- *The width of each stair is called 'step-size'/'learning-rate in many ML algorithms.*
- …..
- *What will happen if the step-size is too big?*
- *What will happen if the step size is too small?*



$$y = -x^2 \qquad \text{(Quadratic function)}$$

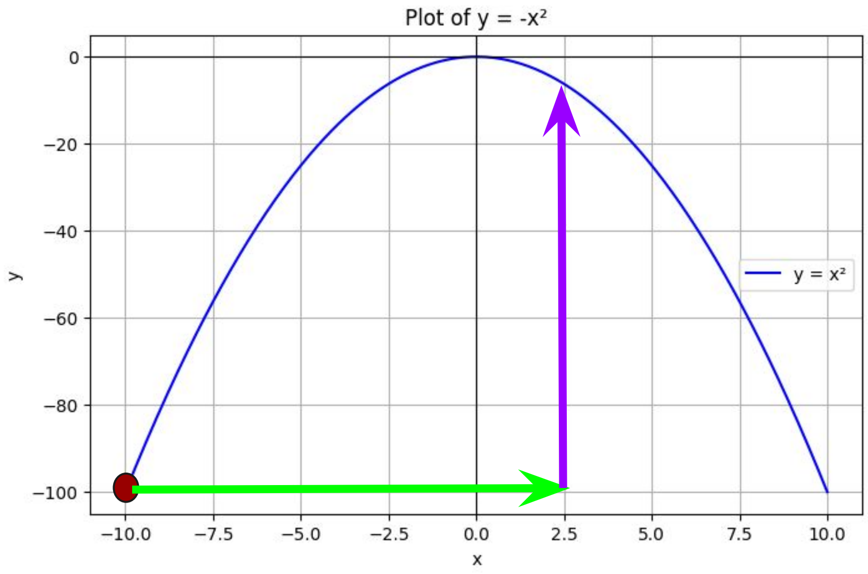
$$\frac{dy}{dx} = -2x \qquad \text{(First Derivative)}$$

Questions

Linear Regression

Gradient ascent - General idea:

- *One wants to reach the top starting from the red point*
- *If we take derivatives, with respect to x, we will know which direction to move*
- *We keep iterating*
- *It's like building stairs of equal width and varying height (touching f(x)).*
- *Usually until f(x) ~ 0*
- *The width of each stair is called 'step-size'/'learning-rate in many ML algorithms.*
- *.....*
- *What will happen if the step-size is too big?*
- *What will happen if the step size is too small?*



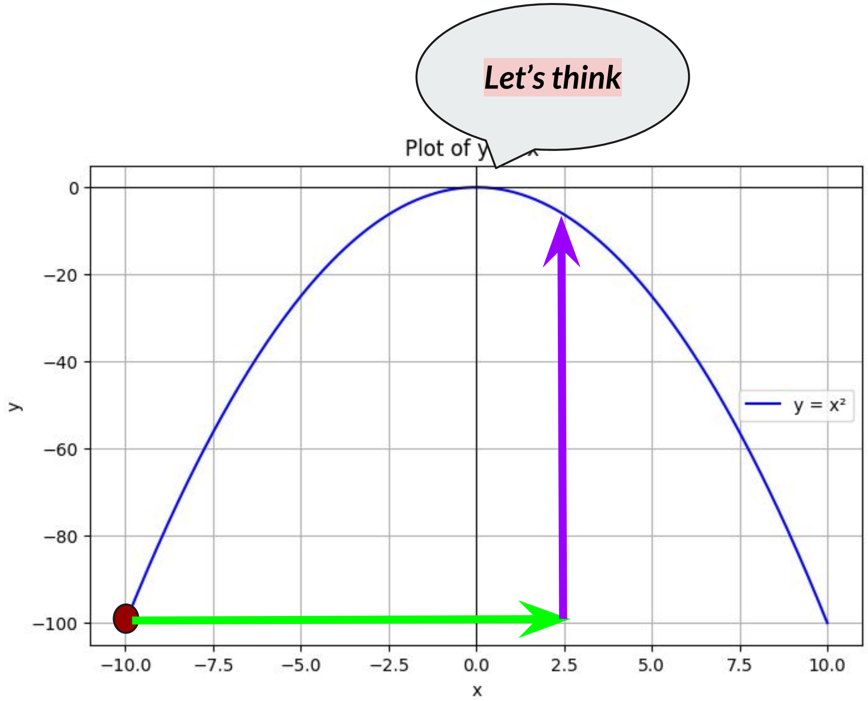
$$y = -x^2 \qquad \text{(Quadratic function)}$$

$$\frac{dy}{dx} = -2x \qquad \text{(First Derivative)}$$

Linear Regression

Gradient ascent - General idea:

- *One wants to reach the top starting from the red point*
- *If we take derivatives, with respect to x, we will know which direction to move*
- *We keep iterating*
- *It's like building stairs of equal width and varying height (touching f(x)).*
- *Usually until f(x) ~ 0*
- *The width of each stair is called 'step-size'/'learning-rate in many ML algorithms.*
- *.....*
- *What will happen if the step-size is too big?*
- *What will happen if the step size is too small?*



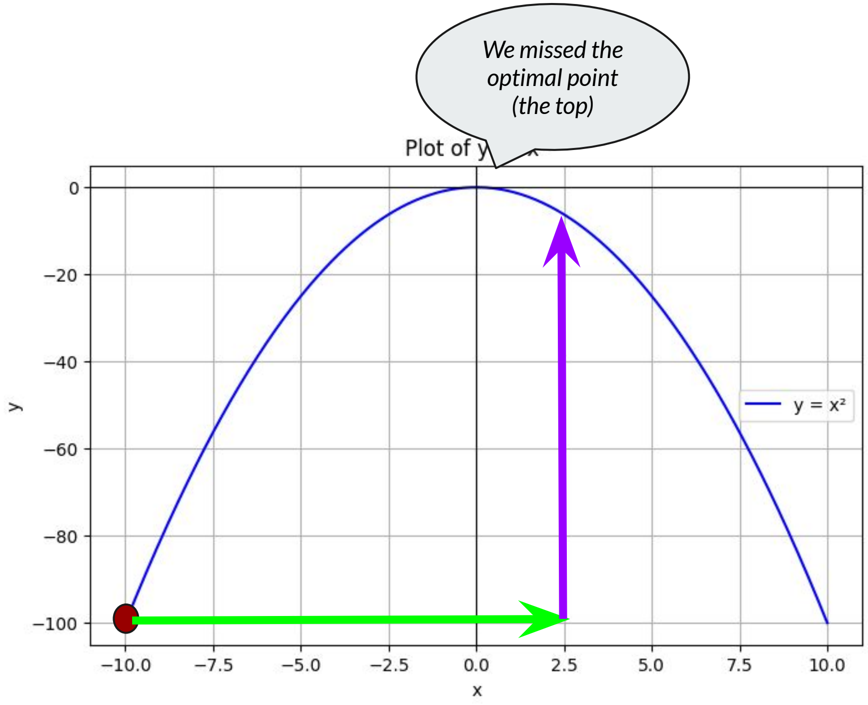
$$y = -x^2 \qquad \text{(Quadratic function)}$$

$$\frac{dy}{dx} = -2x \qquad \text{(First Derivative)}$$

Linear Regression

Gradient ascent - General idea:

- *One wants to reach the top starting from the red point*
- *If we take derivatives, with respect to x, we will know which direction to move*
- *We keep iterating*
- *It's like building stairs of equal width and varying height (touching f(x)).*
- *Usually until f(x) ~ 0*
- *The width of each stair is called 'step-size'/'learning-rate in many ML algorithms.*
- *.....*
- *What will happen if the step-size is too big?*
- *What will happen if the step size is too small?*



$$y = -x^2 \qquad \text{(Quadratic function)}$$

$$\frac{dy}{dx} = -2x \qquad \text{(First Derivative)}$$

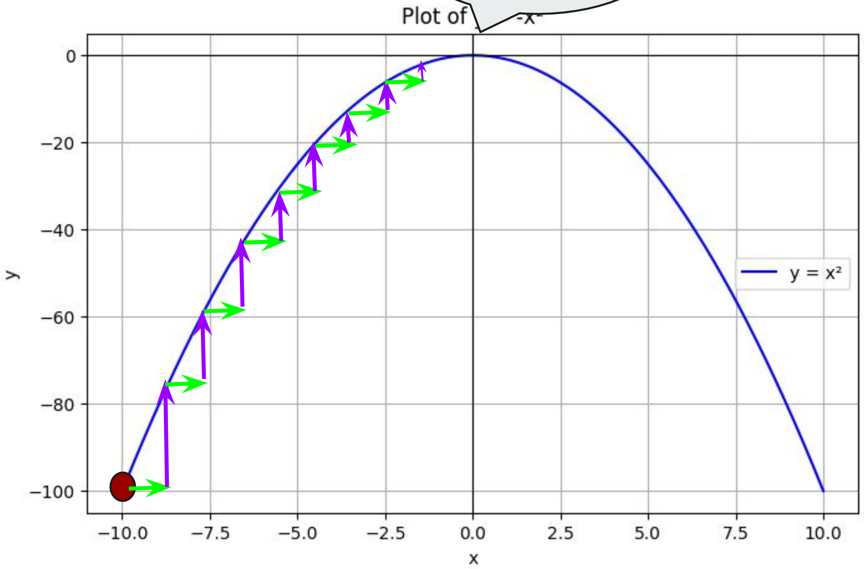
Another Question!

Linear Regression

Gradient ascent - General idea:

- *One wants to reach the top starting from the red point*
- *If we take derivatives, with respect to x, we will know which direction to move*
- *We keep iterating*
- *It's like building stairs of equal width and varying height (touching f(x)).*
- *Usually until f(x) ~ 0*
- *The width of each stair is called 'step-size'/'learning-rate in many ML algorithms.*
- .....
- *What will happen if the step-size is too big?*
- *What will happen if the step size is too small?*

Let's think



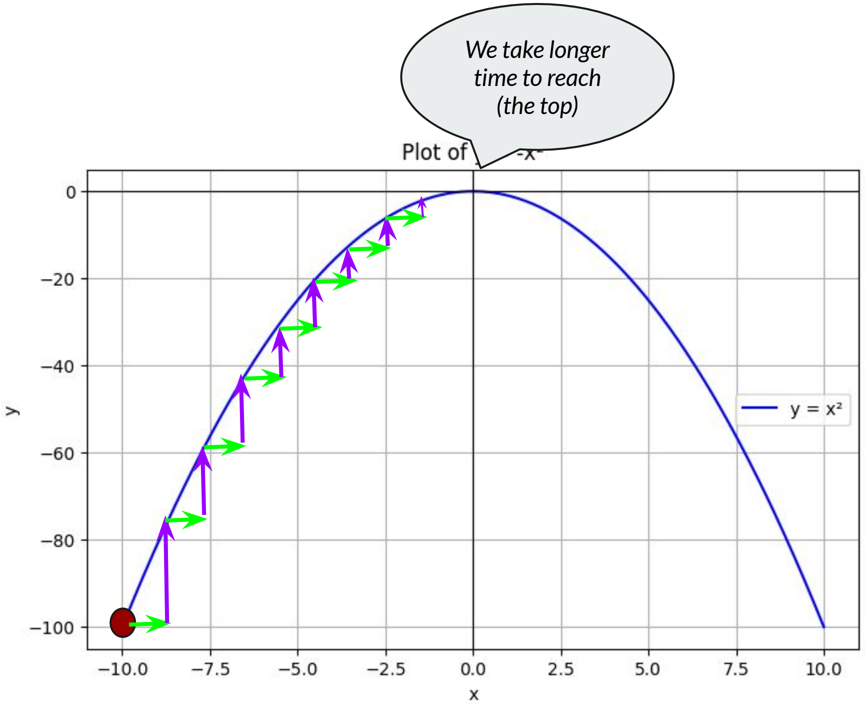
$$y = -x^2 \qquad \text{(Quadratic function)}$$

$$\frac{dy}{dx} = -2x \qquad \text{(First Derivative)}$$

Linear Regression

Gradient ascent - General idea:

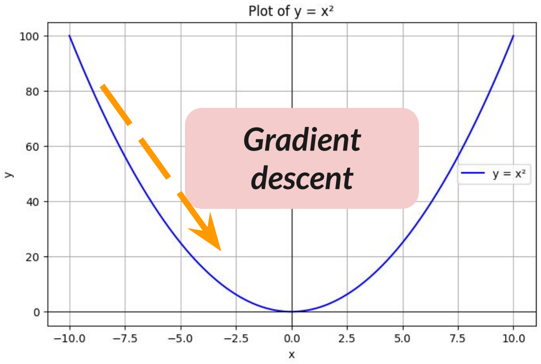
- *One wants to reach the top starting from the red point*
- *If we take derivatives, with respect to x, we will know which direction to move*
- *We keep iterating*
- *It's like building stairs of equal width and varying height (touching f(x)).*
- *Usually until f(x) ~ 0*
- *The width of each stair is called 'step-size'/'learning-rate in many ML algorithms.*
- *.....*
- *What will happen if the step-size is too big?*
- *What will happen if the step size is too small?*



$$y = -x^2 \qquad \text{(Quadratic function)}$$

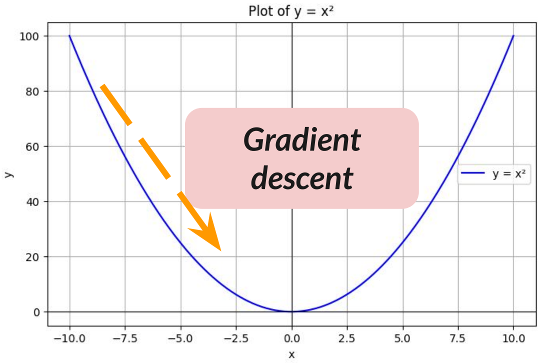
$$\frac{dy}{dx} = -2x \qquad \text{(First Derivative)}$$

Now we are ready for the

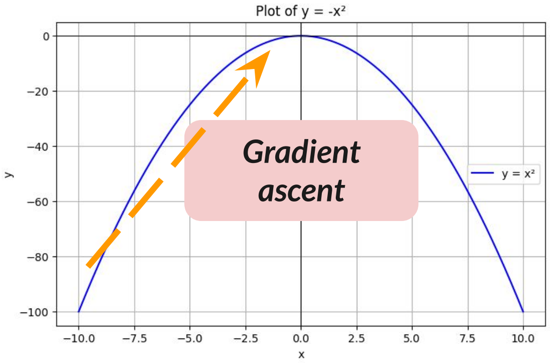


$$y = x^2$$

Now we are ready for the



$$y = x^2$$



$$y = -x^2$$

Gradient Descent Algorithm

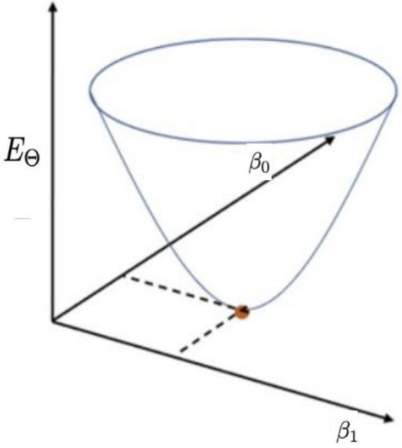
Gradient descent (- ascent)

1. Start with an initial ***(β_0, β_1)*** and a ***learning rate (L)*** , a scalar, which controls the gradient step.
2. For ***N*** training data points, estimate the **model loss**
3. Estimate the gradient (vector of partial derivatives): $\nabla E_\Theta = \left[\frac{\delta}{\delta \beta_0}(E_\Theta), \frac{\delta}{\delta \beta_1}(E_\Theta)\right]$
4. Update parameters

$$\beta_0 \leftarrow \beta_0 - L * \frac{\delta}{\delta \beta_0}(E_\Theta)$$

$$\beta_1 \leftarrow \beta_1 - L * \frac{\delta}{\delta \beta_1}(E_\Theta)$$

4. Go to step 2) and iterate until the model loss reaches a predefined **threshold** or a certain **number of iterations** are executed.



- **Optimization/Loss Function:**

$$E_\Theta = \frac{1}{2}\sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)^2$$

- **Gradient w.r.t.** β_0**:**

$$\frac{\partial E_\Theta}{\partial \beta_0} = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)$$

- **Gradient w.r.t.** β_1**:**

$$\frac{\partial E_\Theta}{\partial \beta_1} = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)\, x_i$$

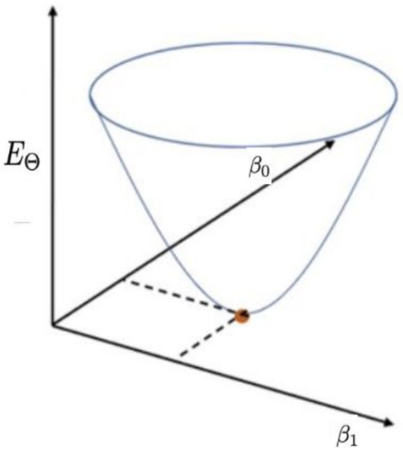
Gradient descent (- ascent)

1. Start with an initial ***(β_0, β_1)*** and a ***learning rate (L)*** , a scalar, which controls the gradient step.
2. For ***N*** training data points, estimate the **model loss**
3. Estimate the gradient (vector of partial derivatives): $\nabla E_\Theta = \left[\frac{\delta}{\delta\beta_0}(E_\Theta), \frac{\delta}{\delta\beta_1}(E_\Theta)\right]$
4. Update parameters

$$\beta_0 \leftarrow \beta_0 - L * \frac{\delta}{\delta\beta_0}(E_\Theta)$$

$$\beta_1 \leftarrow \beta_1 - L * \frac{\delta}{\delta\beta_1}(E_\Theta)$$

4. Go to step 2) and iterate until the model loss reaches a predefined **threshold** or a certain **number of iterations** are executed.



- **Optimization/Loss Function:**

$$E_\Theta = \frac{1}{2}\sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)^2$$

- **Gradient w.r.t. β_0:**

$$\frac{\partial E_\Theta}{\partial \beta_0} = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)$$

- **Gradient w.r.t. β_1:**

$$\frac{\partial E_\Theta}{\partial \beta_1} = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)\, x_i$$

Gradient descent (- ascent)

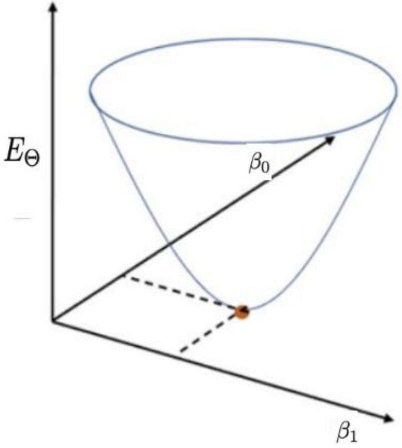
1. Start with an initial ***(β_0, β_1)*** and a ***learning rate (L)*** , a scalar, which controls the gradient step.
2. For ***N*** training data points, estimate the **model loss**
3. Estimate the gradient (vector of partial derivatives): $\nabla E_\Theta = \left[\frac{\delta}{\delta \beta_0}(E_\Theta), \frac{\delta}{\delta \beta_1}(E_\Theta)\right]$

4. Update parameters

$$\beta_0 \leftarrow \beta_0 - L * \frac{\delta}{\delta \beta_0}(E_\Theta)$$

$$\beta_1 \leftarrow \beta_1 - L * \frac{\delta}{\delta \beta_1}(E_\Theta)$$

4. Go to step 2) and iterate until the model loss reaches a predefined **threshold** or a certain **number of iterations** are executed.



- **Optimization/Loss Function:**

$$E_\Theta = \frac{1}{2}\sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)^2$$

- **Gradient w.r.t. β_0:**

$$\frac{\partial E_\Theta}{\partial \beta_0} = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)$$

- **Gradient w.r.t. β_1:**

$$\frac{\partial E_\Theta}{\partial \beta_1} = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)\, x_i$$

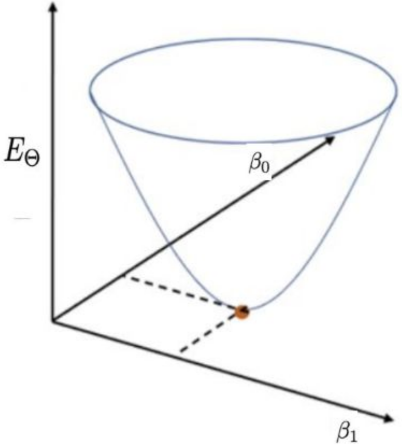
Gradient descent (- ascent)

1. Start with an initial ***(β$_0$, β$_1$)*** and a ***learning rate (L)***, a scalar, which controls the gradient step.
2. For ***N*** training data points, estimate the **model loss**
3. Estimate the gradient (vector of partial derivatives): $\nabla E_\Theta = \left[\frac{\delta}{\delta\beta_0}(E_\Theta), \frac{\delta}{\delta\beta_1}(E_\Theta)\right]$
4. Update parameters

$$\beta_0 \leftarrow \beta_0 - L * \frac{\delta}{\delta\beta_0}(E_\Theta)$$

$$\beta_1 \leftarrow \beta_1 - L * \frac{\delta}{\delta\beta_1}(E_\Theta)$$

4. Go to step 2) and iterate until the model loss reaches a predefined **threshold** or a certain **number of iterations** are executed.



- **Optimization/Loss Function:**

$$E_\Theta = \frac{1}{2}\sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)^2$$

- **Gradient w.r.t. β_0:**

$$\frac{\partial E_\Theta}{\partial \beta_0} = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)$$

- **Gradient w.r.t. β_1:**

$$\frac{\partial E_\Theta}{\partial \beta_1} = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)\,x_i$$

Batch Gradient descent (- ascent)

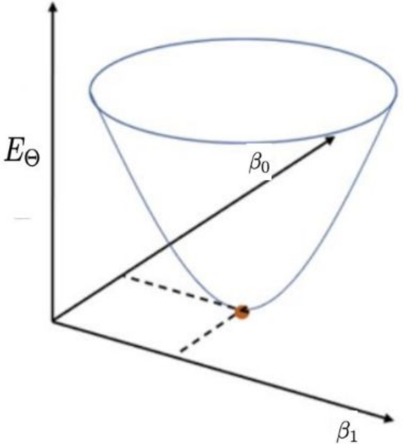
Gradient descent (- ascent) Variants

1. Start with an initial ***(β_0, β_1)*** and a ***learning rate (L)*** , a scalar, which controls the gradient step.
2. For ***N*** training data points, estimate the **model loss**
3. Estimate the gradient (vector of partial derivatives): $\nabla E_\Theta = \left[\frac{\delta}{\delta\beta_0}(E_\Theta), \frac{\delta}{\delta\beta_1}(E_\Theta)\right]$
4. Update parameters

$$\beta_0 \leftarrow \beta_0 - L * \frac{\delta}{\delta\beta_0}(E_\Theta)$$

$$\beta_1 \leftarrow \beta_1 - L * \frac{\delta}{\delta\beta_1}(E_\Theta)$$

4. Go to step 2) and iterate until the model loss reaches a predefined **threshold** or a certain **number of iterations** are executed.



- **Optimization/Loss Function:**

$$E_\Theta = \frac{1}{2}\sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)^2$$

- **Gradient w.r.t.** β_0**:**

$$\frac{\partial E_\Theta}{\partial \beta_0} = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)$$

- **Gradient w.r.t.** β_1**:**

$$\frac{\partial E_\Theta}{\partial \beta_1} = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)\, x_i$$

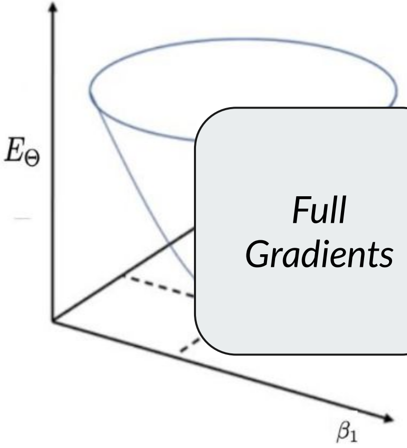
Gradient descent (- ascent) Variants

1. Start with an initial ***(β_0, β_1)*** and a ***learning rate (L)*** , a scalar, which controls the gradient step.
2. For ***N*** training data points, estimate the **model loss**
3. Estimate the gradient (vector of partial derivatives): $\nabla E_\Theta = \left[\frac{\delta}{\delta\beta_0}(E_\Theta), \frac{\delta}{\delta\beta_1}(E_\Theta)\right]$
4. Update parameters

$$\beta_0 \leftarrow \beta_0 - L * \frac{\delta}{\delta\beta_0}(E_\Theta)$$

$$\beta_1 \leftarrow \beta_1 - L * \frac{\delta}{\delta\beta_1}(E_\Theta)$$

4. Go to step 2) and iterate until the model loss reaches a predefined **threshold** or a certain **number of iterations** are executed.



Full Gradients

- **Optimization/Loss Function:**

$$E_\Theta = \frac{1}{2}\sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)^2$$

- **Gradient w.r.t. β_0:**

$$\frac{\partial E_\Theta}{\partial \beta_0} = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)$$

- **Gradient w.r.t. β_1:**

$$\frac{\partial E_\Theta}{\partial \beta_1} = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)\, x_i$$

Full Gradients

Gradient descent (- ascent) Variants

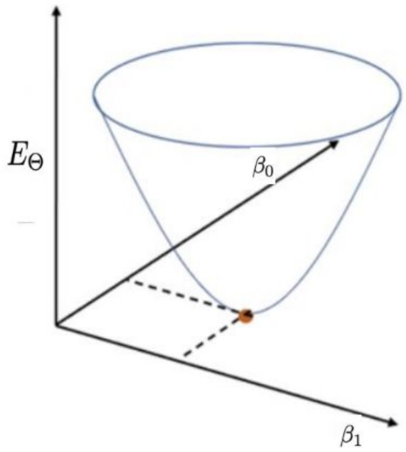
1. Start with an initial **_(β_0, β_1)_** and a **_learning rate (L)_** , a scalar, which controls the gradient step.
2. For ***N*** training data points, estimate the **model loss**
3. Estimate the gradient (vector of partial derivatives): $\nabla E_\Theta = \left[\frac{\delta}{\delta\beta_0}(E_\Theta), \frac{\delta}{\delta\beta_1}(E_\Theta)\right]$

4. Update parameters

$$\beta_0 \leftarrow \beta_0 - L * \frac{\delta}{\delta\beta_0}(E_\Theta)$$

$$\beta_1 \leftarrow \beta_1 - L * \frac{\delta}{\delta\beta_1}(E_\Theta)$$

4. Go to step 2) and iterate until the model loss reaches a predefined **threshold** or a certain **number of iterations** are executed.



- **Optimization/Loss Function:**

$$E_\Theta = \frac{1}{2}\sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)^2$$

- **Gradient w.r.t. β_0:**

$$\frac{\partial E_\Theta}{\partial \beta_0} = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)$$

- **Gradient w.r.t. β_1:**

$$\frac{\partial E_\Theta}{\partial \beta_1} = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)\, x_i$$

Batch Gradient descent: *N=n (random) and n< N*

Gradient descent (- ascent) Variants

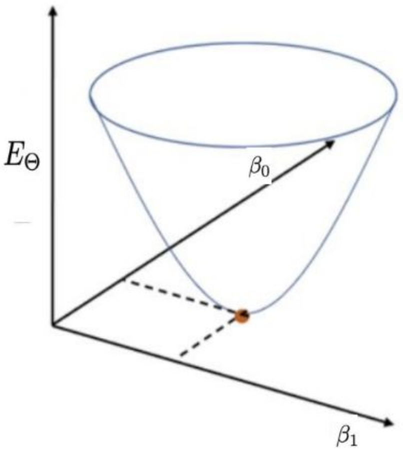
1. Start with an initial ***(β_0, β_1)*** and a ***learning rate (L)*** , a scalar, which controls the gradient step.
2. For ***N*** training data points, estimate the **model loss**
3. Estimate the gradient (vector of partial derivatives): $\nabla E_\Theta = \left[\frac{\delta}{\delta\beta_0}(E_\Theta), \frac{\delta}{\delta\beta_1}(E_\Theta)\right]$

4. Update parameters

$$\beta_0 \leftarrow \beta_0 - L * \frac{\delta}{\delta\beta_0}(E_\Theta)$$

$$\beta_1 \leftarrow \beta_1 - L * \frac{\delta}{\delta\beta_1}(E_\Theta)$$

4. Go to step 2) and iterate until the model loss reaches a predefined **threshold** or a certain **number of iterations** are executed.



- **Optimization/Loss Function:**

$$E_\Theta = \frac{1}{2}\sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)^2$$

- **Gradient w.r.t.** β_0**:**

$$\frac{\partial E_\Theta}{\partial \beta_0} = \sum_{i=1}^{\boxed{N}}(\beta_0 + \beta_1 x_i - y_i)$$

- **Gradient w.r.t.** β_1**:**

$$\frac{\partial E_\Theta}{\partial \beta_1} = \sum_{i=1}^{\boxed{N}}(\beta_0 + \beta_1 x_i - y_i)\, x_i$$

Stochastic Gradient descent: *N=1 (random)*

Notebook presentation

- Notebook github
  - ➜ [Gradient descent training (Linear Regression)]

GPT

Our model today

What we have discussed today

- ~~General Concepts: ***Straight Line** to **Linear Regression***~~
- Gradient Descent Algorithm
  - ➜ A simple two parameter **Linear Regression** model
  - ➜ Hands on **Notebook implementation**
- **QA …….**