CSCI 3022

intro to data science with probability & statistics

December 10, 2018

Solution Techniques for
Linear and Logistic Regression

Arkaine!

Department of Computer Science
UNIVERSITY OF COLORADO **BOULDER**

Dan Larremore

Practicum & Final Exam

Office Hrs.
ECES 118B

- The **Practicum** is due at 11:55pm this Wednesday.

- **Final Exam** Tuesday Dec 18 from 7:30 - 10 pm in FLMG 154.
  - Cumulative but will emphasize material since midterm
  - Bring a calculator
  - One 3” x 5” notecard. Handwritten. No magnifying glasses.

- **Review** in-class on Wednesday.
  - Come with questions!
  - If there are no questions, I’ll lock the doors and Katie, Claire, Michael & Other Michael will sing karaoke. Disaster!

Last times on CSCI 3022:

- Given data $(x_{i1}, x_{i2}, \ldots, x_{ip}, y_i)$ for $i = 1, 2, \ldots, n$ fit a MLR model of the form
  $y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon_i$ where $\epsilon_i \sim N(0, \sigma^2)$

- by minimizing the sum of squared errors: $SSE = \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_i)]^2$

- Given data $(x_{i1}, x_{i2}, \ldots, x_{ip}, y_i)$ for $i = 1, 2, \ldots, n$ fit a LogReg model of the form

$$p(y = 1 \mid x) = \text{sigm}(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p)$$

where the values of y are 0 or 1.

- Recall that we determined logistic regression is MLR on the *log odds.*

Finding Parameters in Linear Regression

- Whether doing simple linear regression or multiple linear regression, parameters are estimated by minimizing the SSE

$$SSE = \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_i)]^2 \qquad SSE = \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip})]^2$$

- When you lots of data and a model with many features, this becomes a difficult problem
- While direct methods (based on linear algebra) exist, they are far too memory and computationally expensive to perform in real life

- Instead, we use an **iterative method**

Iterative Solution Methods

Iterative methods can be thought of as very intelligent guess and check

- Make a guess at the parameters
- Update your guess in a smart way, based on the problem specs, to get a better guess
- Repeat until guess converges to something very close to the correct answer

guess $x^{(0)}$ — x guess, and (0) says first/initial guess.

from $x^{(0)}$, ? get to $x^{(1)}$

repeat: from $x^{(1)}$, ? get to $x^{(2)}$

repeat this until some stopping criterion — rule for when to stop

PSA:
1 criterion
many criteria

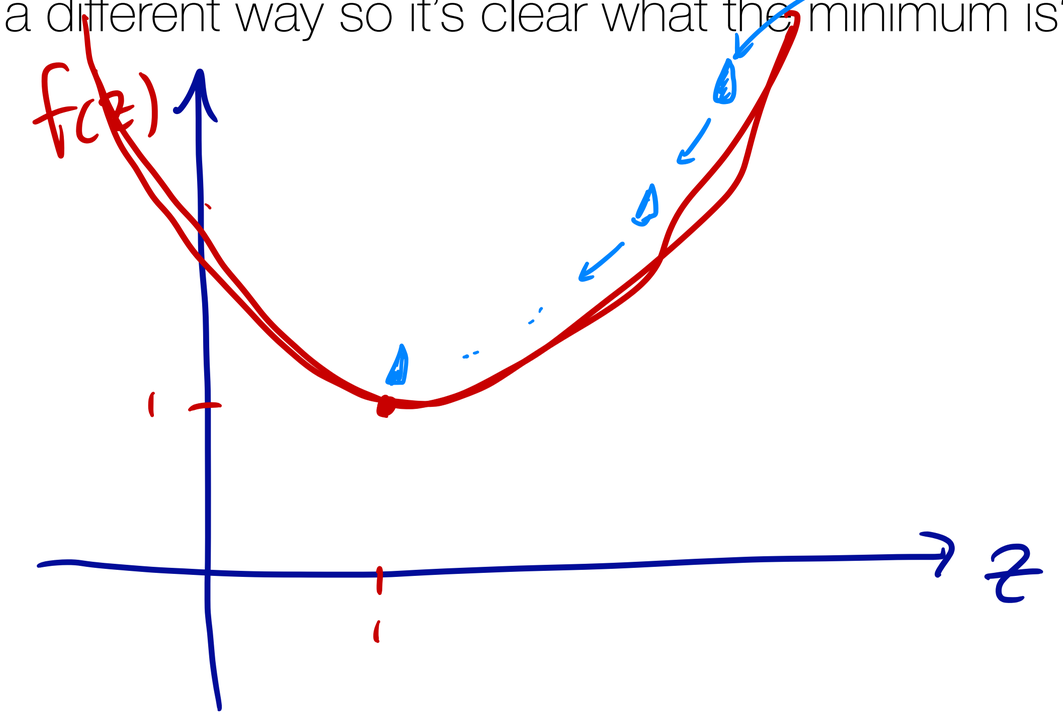
For example . . .

- Suppose you want to find the minimum of the function $f(z) = z^2 - 2z + 2$
- Can we rewrite this function in a different way so it's clear what the minimum is?

Note:

$(z-1)^2 = z^2 - 2z + 1$

$\Rightarrow f(z) = (z-1)^2 + 1$



- 1 minimum. Unique.
- everything goes "downhill" to that minimum.

For example . . .

- Suppose you want to find the minimum of the function $f(z) = z^2 - 2z + 2$
- Can we rewrite this function in a different way so it's clear what the minimum is?

$$f(z) = (z-1)^2 + 1$$

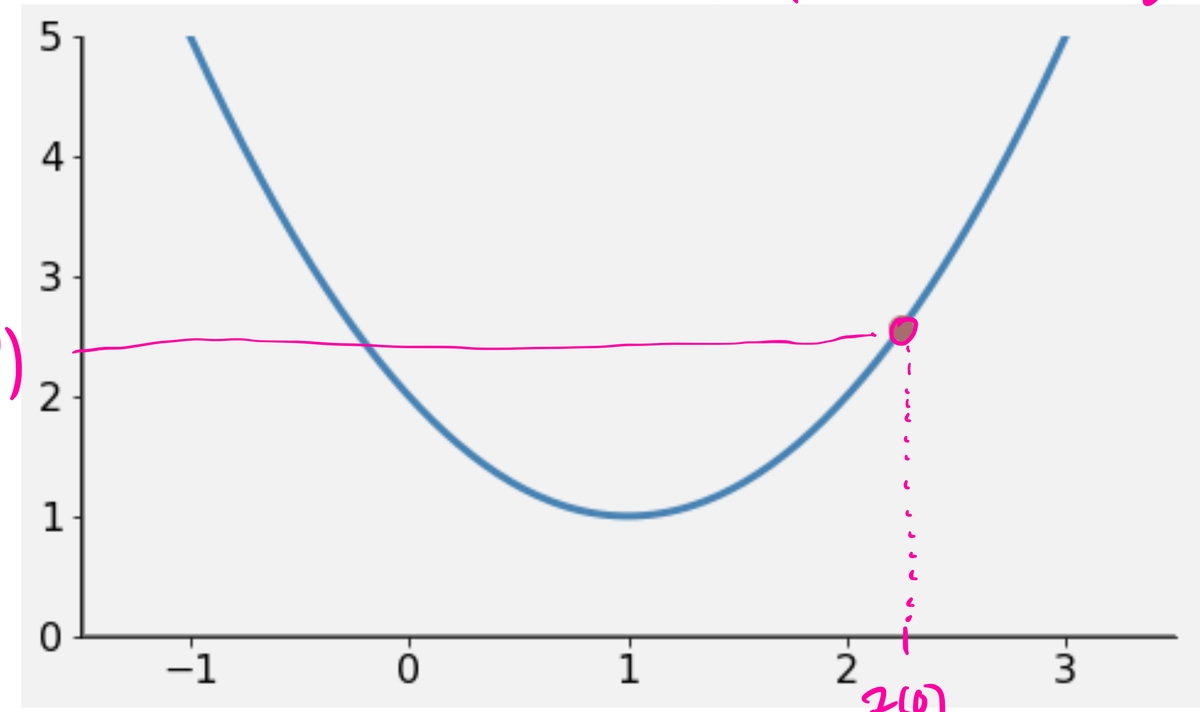
- **Question**: What nice properties for minimization does this function have?

convex
-or-
concave upward
everywhere

For example . . .

- Suppose you want to find the minimum of the function $f(z) = z^2 - 2z + 2$

- Suppose that I guess that the minimizer is $z^{(0)} = 2.25$
- **Question**: Which way should I move? to the left! towards smaller value of f(z).



Reason:

Go Downhill.

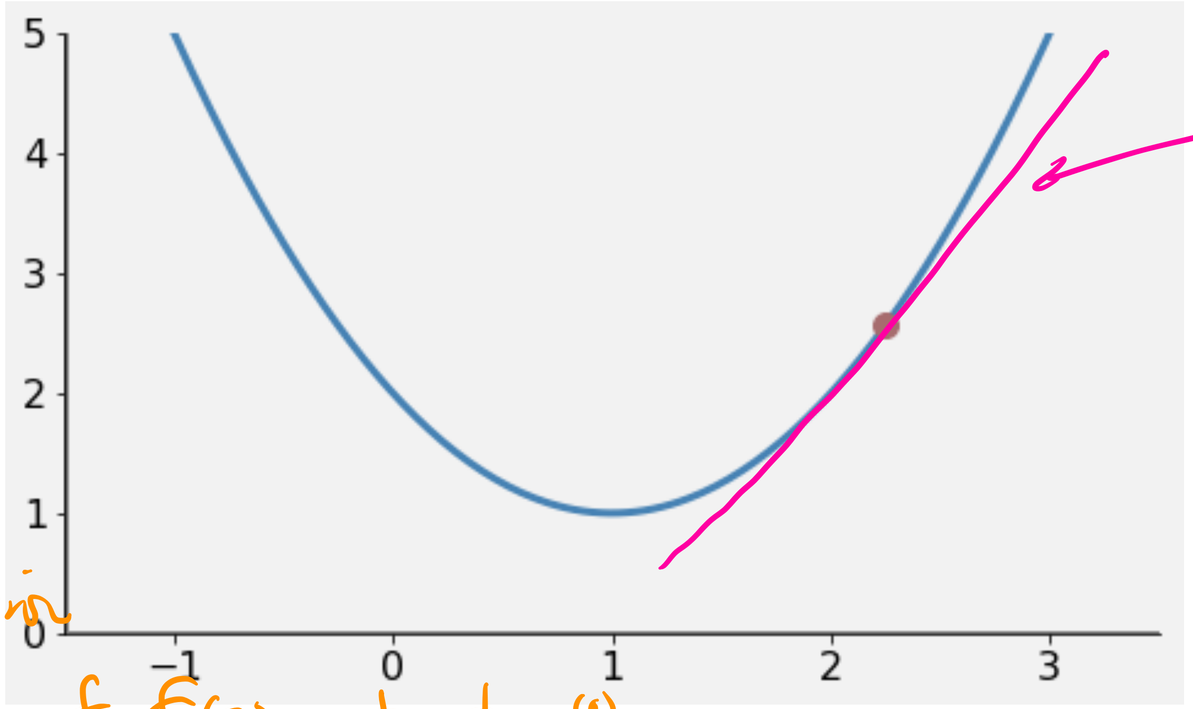
For example . . .

- Suppose you want to find the minimum of the function $f(z) = z^2 - 2z + 2$

- Suppose that I guess that the minimizer is $z^{(0)} = 2.25$
- **Question**: Which way should I move? **Answer**: Downhill! But which way is down?

If slope is +, go left (−z)

If slope is −, go right (+z)



Move opposite direction of the derivative of $f(z)$ at at $z^{(0)}$

For example . . .

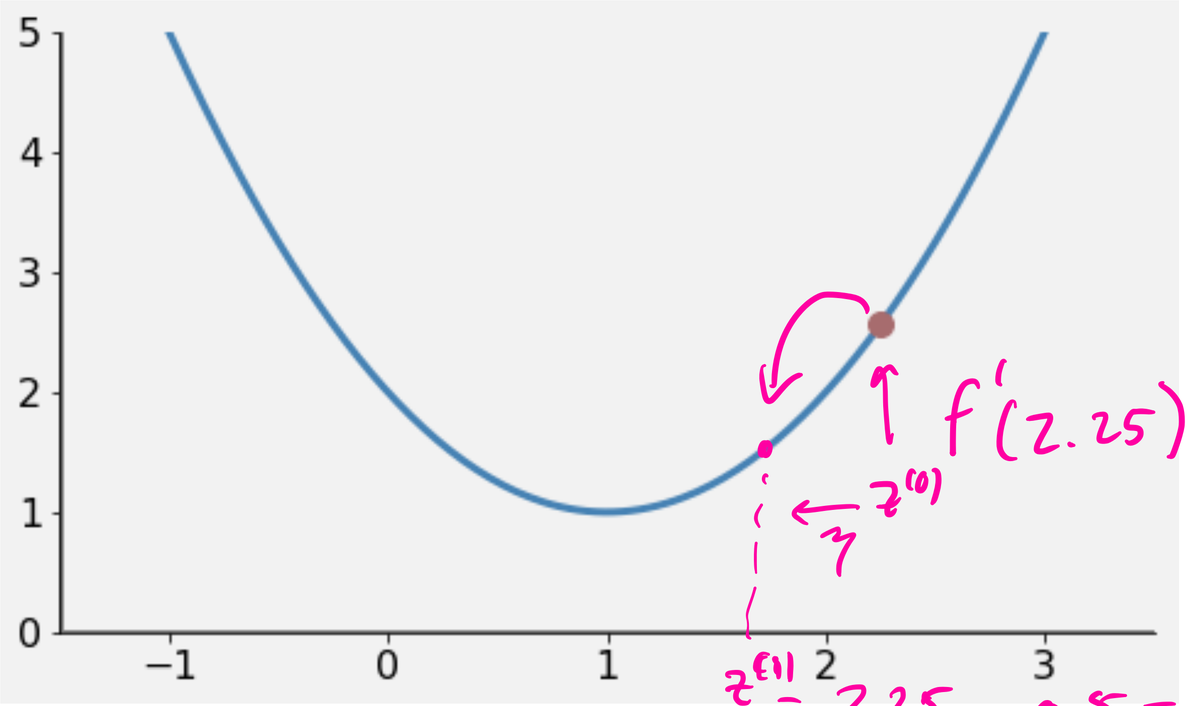
- Suppose you want to find the minimum of the function $f(z) = z^2 - 2z + 2$

- Suppose that I guess that the minimizer is $z^{(0)} = 2.25$
- **Question:** But which way is down? **Answer:** Derivative tells you uphill. Go opposite direction

For example . . .

$f'(z) = 2z - 2$

- Suppose you want to find the minimum of the function $f(z) = z^2 - 2z + 2$
- **Question:** But which way is down? **Answer:** Derivative tells you uphill. Go opposite direction
- **Question**: How far should we move? **Answer**: Hmmmm, just pick a small step size like $\eta = 0.5$



$f'(2.25) = 2 \times 2.25 - 2$

$= 4.5 - 2 = 2.5 > 0$

$z^{(0)}$

η

$z^{(1)} = z^{(0)} - \eta$

$z^{(1)} = 2.25 - 0.5 = 1.75$

For example . . .

$f'(z) = 2z - 2$

- Suppose you want to find the minimum of the function $f(z) = z^2 - 2z + 2$
- **Question:** But which way is down? **Answer:** Derivative tells you uphill. Go opposite direction
- **Question**: How far should we move? **Answer**: Hmmmm, just pick a small step size like $\eta = 0.5$

$f'(z^{(1)}) = 2 \cdot 1.75 - 2$

$= 3.5 - 2$

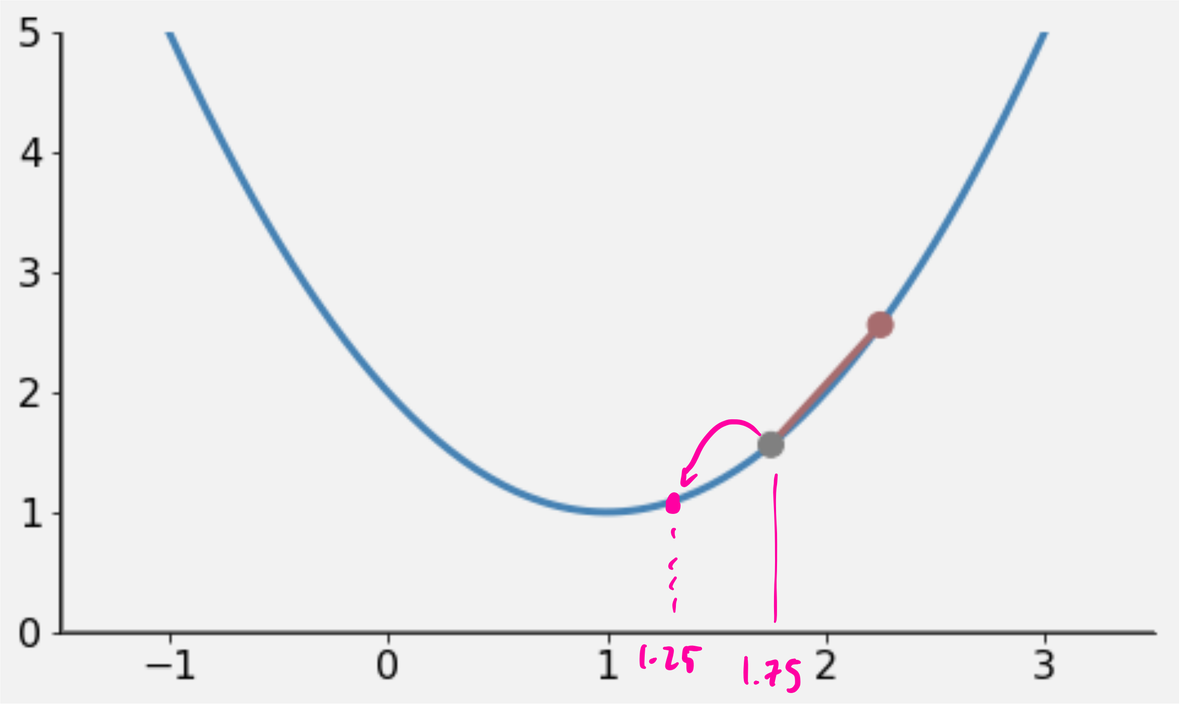
$= 1.5 > 0$

=> go left

$z^{(2)} = z^{(1)} - \eta$

$= 1.75 - 0.5$

$= 1.25$



For example . . .

$f'(z) = 2z - 2$

- Suppose you want to find the minimum of the function $f(z) = z^2 - 2z + 2$
- **Question:** But which way is down? **Answer:** Derivative tells you uphill. Go opposite direction
- **Question**: How far should we move? **Answer**: Hmmmm, just pick a small step size like $\eta = 0.5$

$f'(z^{(1)}) = 2 \cdot 1.25 - 2$

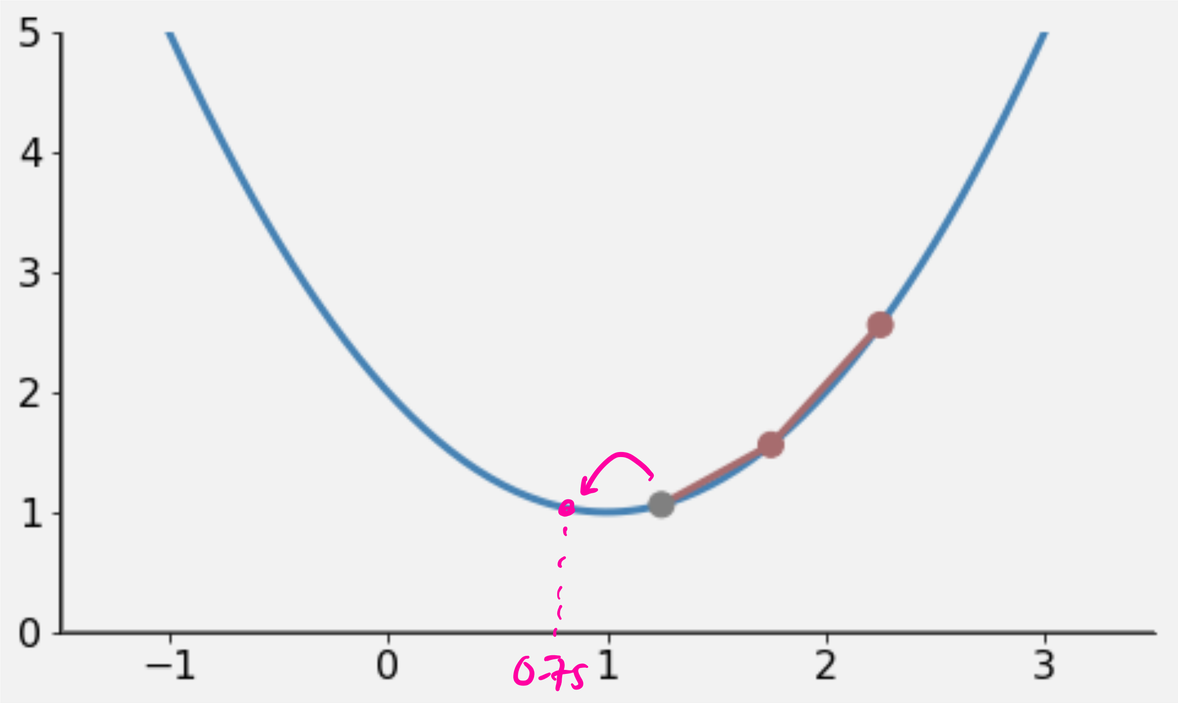
$= 2.5 - 2$

$= 0.5 > 0$

$\Rightarrow$ go left

$z^{(3)} = 1.25 - 0.5$

$= 0.75$



For example . . .

$f'(z) = 2z - 2$

- Suppose you want to find the minimum of the function $f(z) = z^2 - 2z + 2$
- **Question:** But which way is down? **Answer:** Derivative tells you uphill. Go opposite direction
- **Question**: How far should we move? **Answer**: Hmmmm, just pick a small step size like $\eta = 0.5$

$f'(0.75) = 2 \cdot 0.75 - 2$

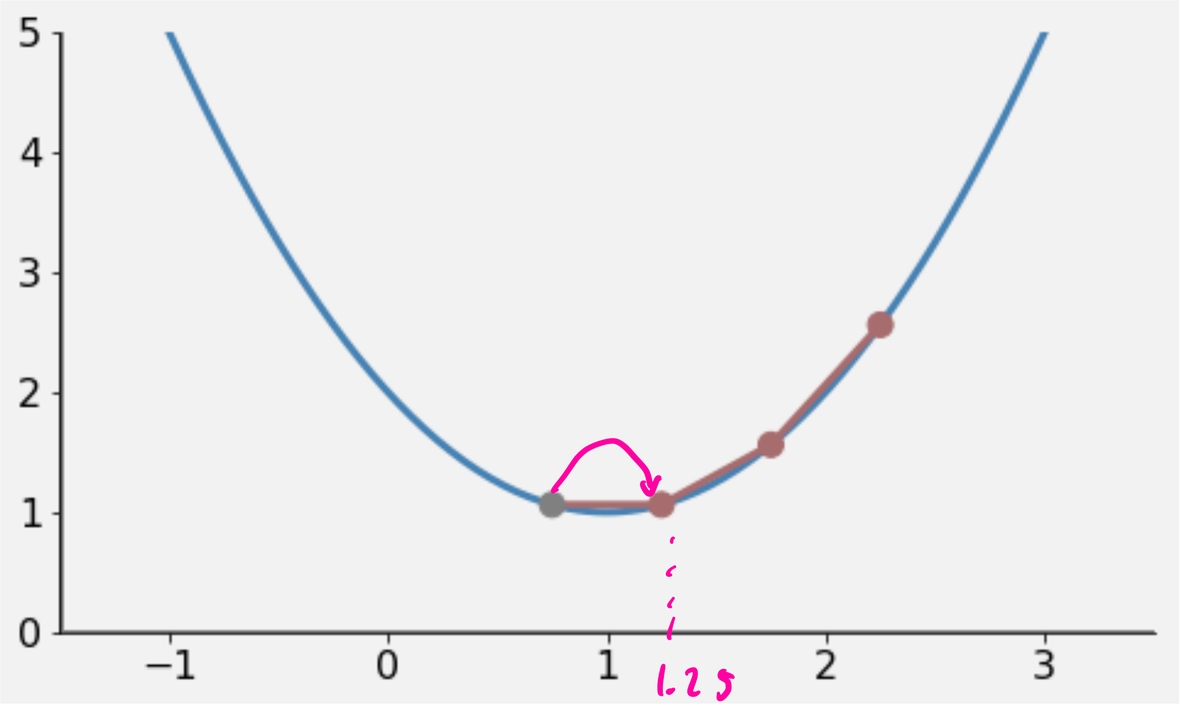
$= 1.5 - 2$

$= -0.5 < 0$

$\Rightarrow$ go right

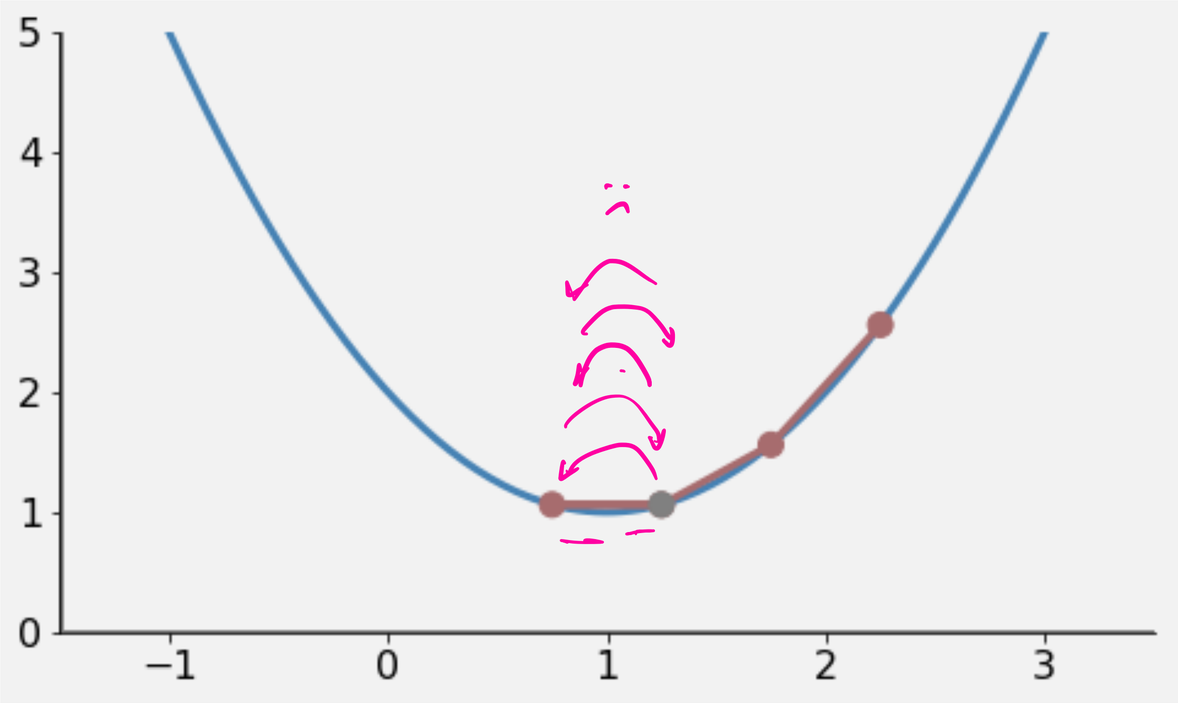
$z^{(4)} = 0.75 + 0.5$

$= 1.25$



For example . . .

- Suppose you want to find the minimum of the function $f(z) = z^2 - 2z + 2$
- **Question:** But which way is down? **Answer:** Derivative tells you uphill. Go opposite direction
- **Question**: How far should we move? **Answer**: Hmmmm, just pick a small step size like $\eta = 0.5$



For example . . .

$f'(z) = 2z - 2$

$\Rightarrow z^{(k+1)} = z^{(k)} - \eta(2z^{(k)} - 2)$

- **Question**: How do we fix this so we can get closer to the true minimum?
  - Bad Idea: make η smaller. Yes, closer. But slow.
  - Better Idea: use magnitude of derivative to adjust how far we move!

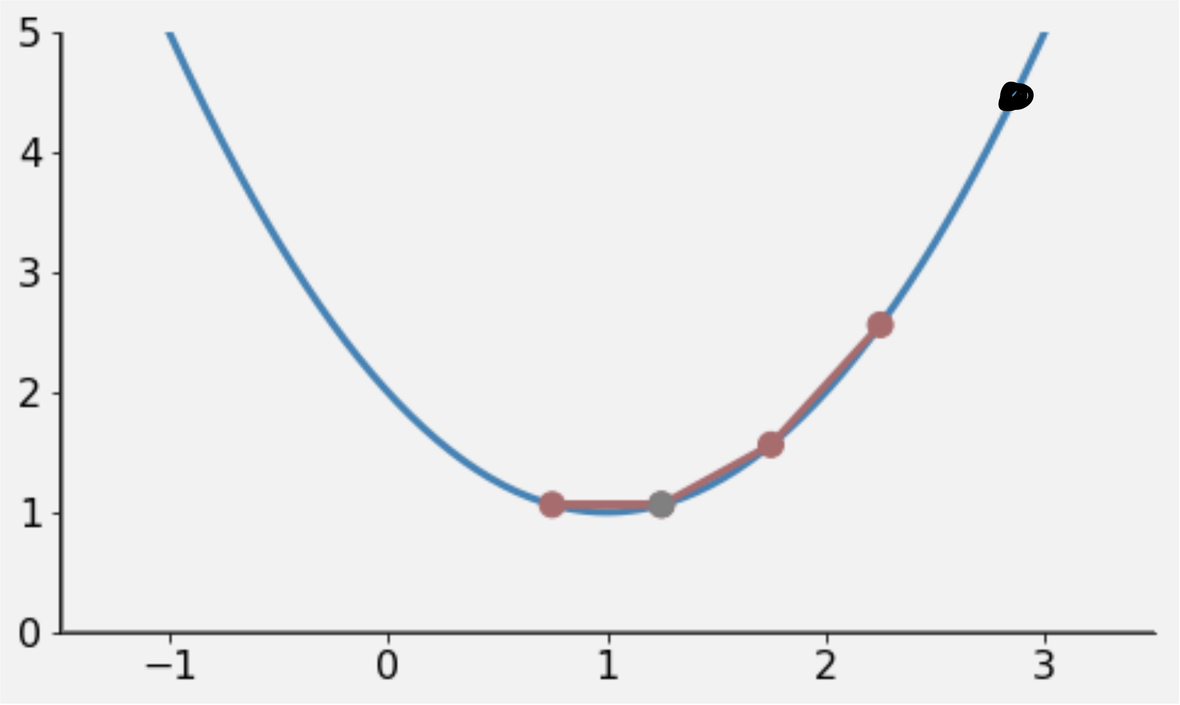
$$z^{(k+1)} = z^{(k)} - \eta f'(z^{(k)})$$

variable step size

update from prev point.

move opposite direction of derivative

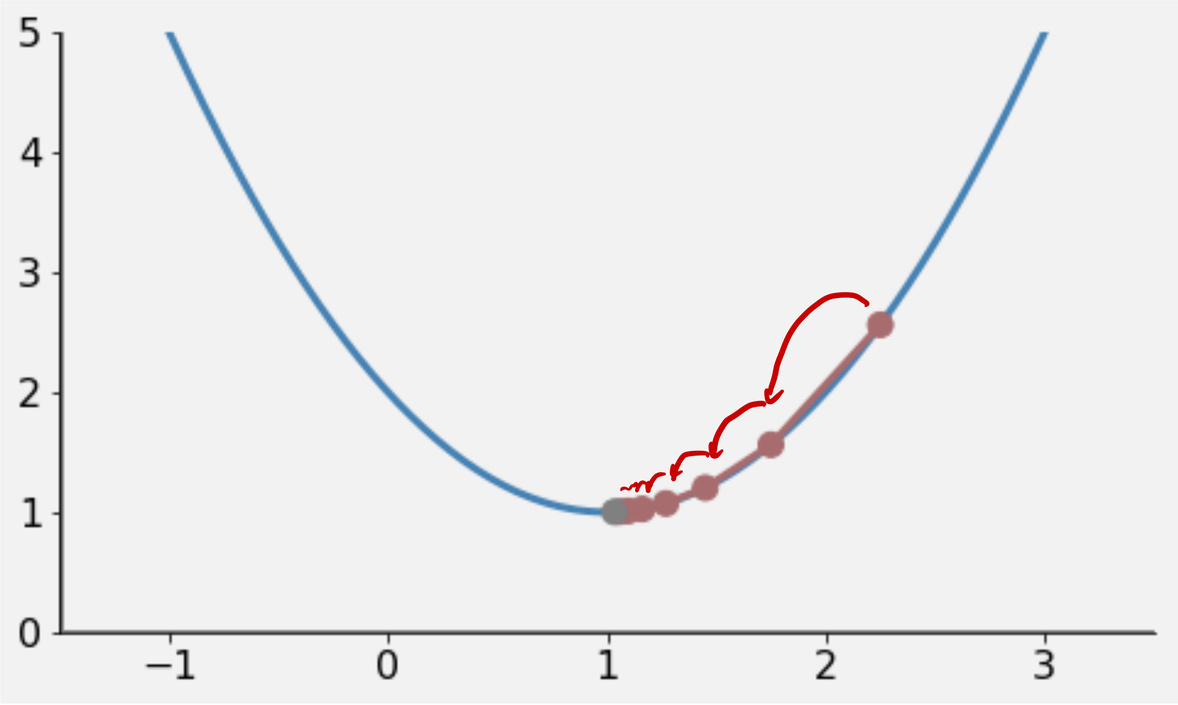
learning rate



Voila!

- This method is called **Gradient Descent** (think "Derivative" Descent)

$$z^{(k+1)} = z^{(k)} - \eta \cdot f'(z^{(k)})$$



Simple Linear Regression

- Right! So, how do we use the idea of Gradient Descent to estimate the parameters in SLR?
- **Recall**, the estimated parameters are the value of β_0 and β_1 that minimize the SSE

$$SSE = \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_i)]^2$$

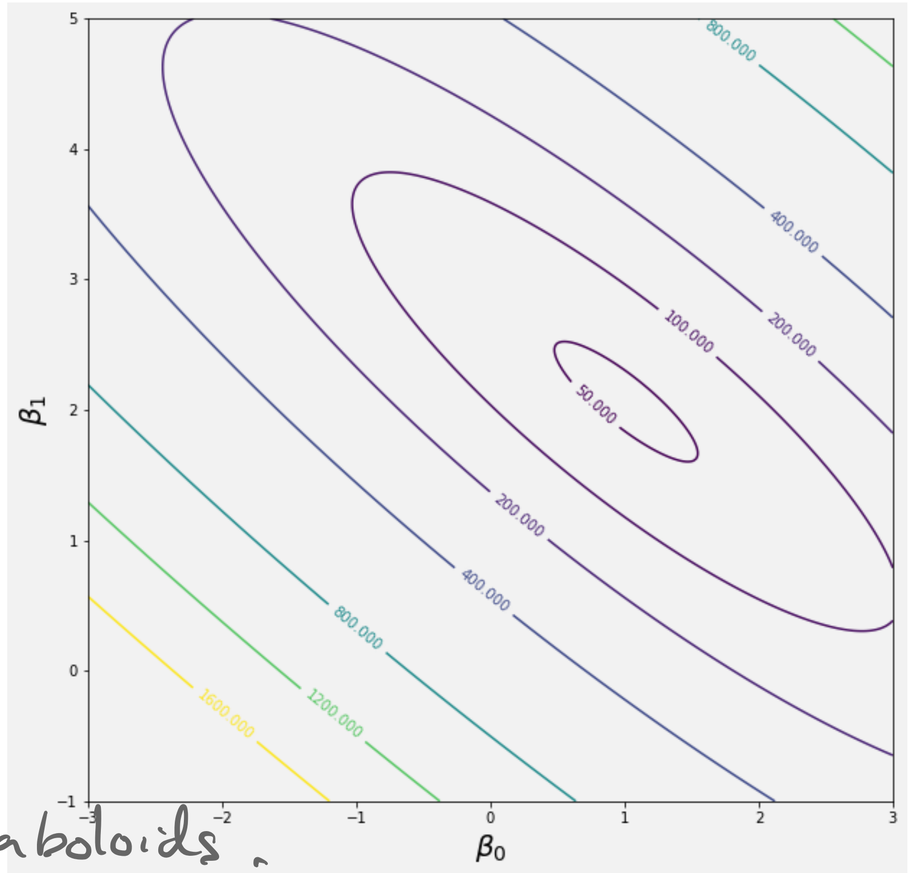
- **Important**: We're minimizing over the β's. The x's and y's are the values from the data.

- The difficulty is that this is a function of two variables, which is not quite a simple parabola, like our previous example.

Simple Linear Regression

- In 1-dimension the SSE is a parabola. In 2-dimensions it's a bowl-like surface.

$$SSE = \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_i)]^2$$



SSE = sum of a set of bowls / paraboloids.
=> sum, itself, has a bowl shape.

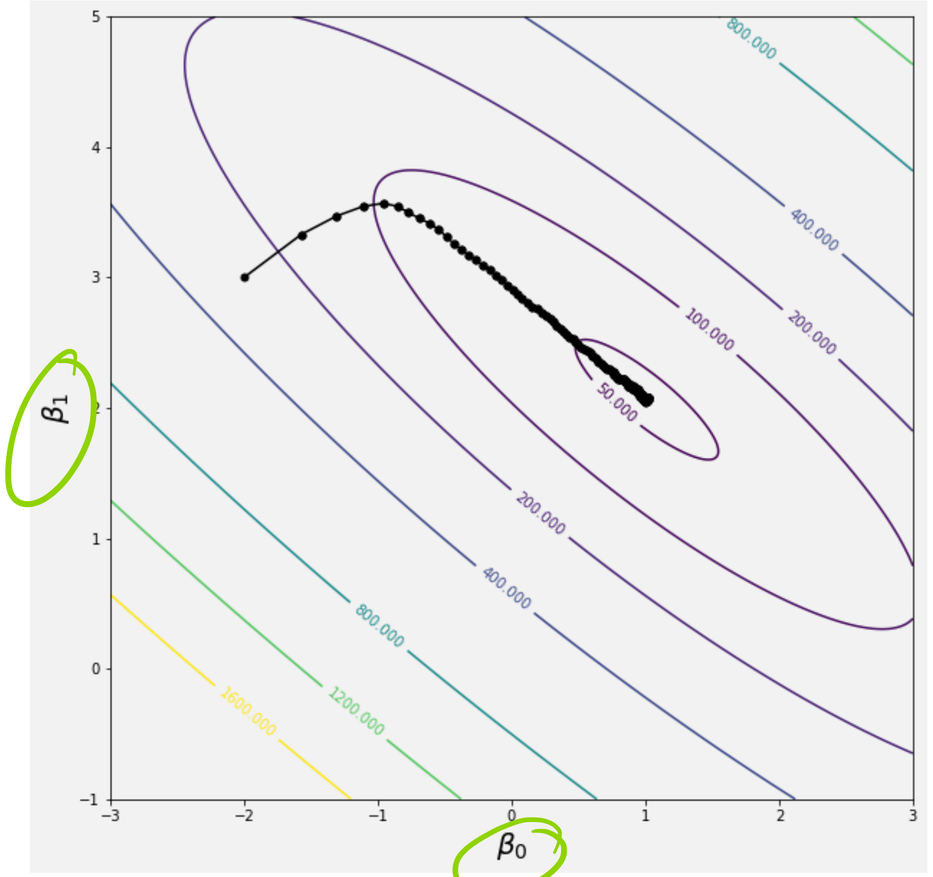
Simple Linear Regression

- In 1-dimension the SSE is a parabola. In 2-dimensions it's a bowl-like surface.

$$SSE = \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_i)]^2$$

- In 2D the process is still the same:
  1. Make a guess at the minimum
  2. Move iteratively downhill

$(\beta_0^{(0)}, \beta_1^{(0)})$



"Downhill" in higher dimension is

$-\nabla SSE$

gradient. nabla

Simple Linear Regression

- In 1-dimension the SSE is a parabola. In 2-dimensions it's a bowl-like surface.

$$SSE = \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_i)]^2$$

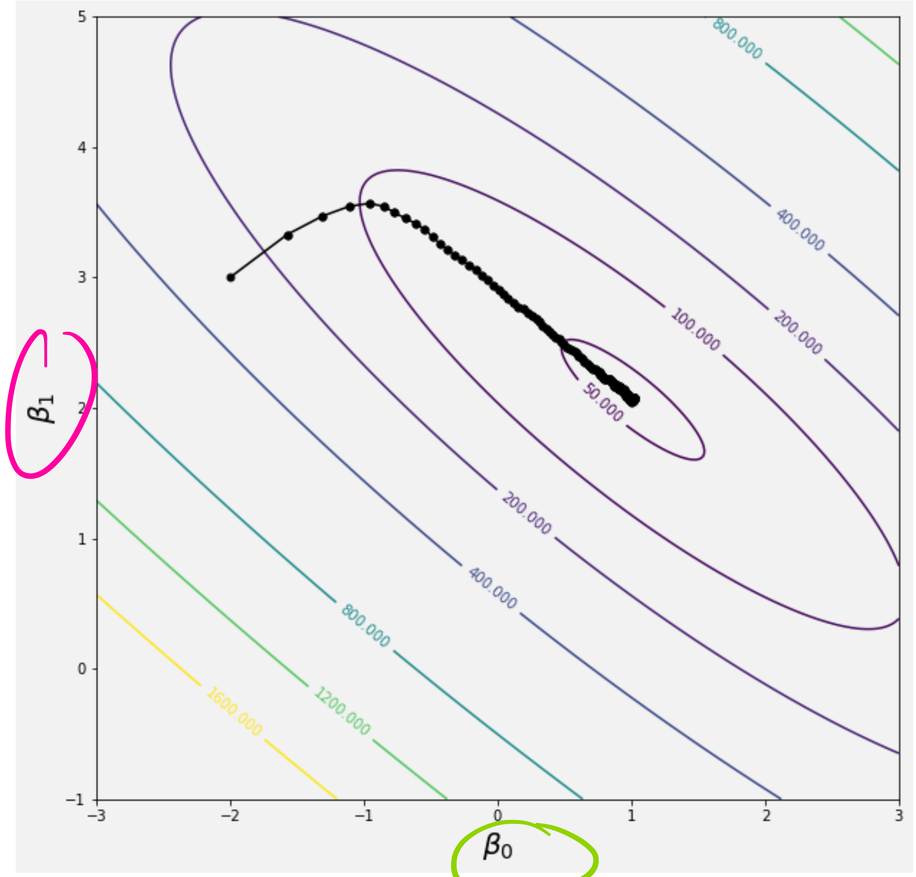
- But how do we move **downhill in 2D?**

Neg. of gradient.

Neg. of every partial derivative

in β_0 direction, $-\frac{\partial SSE}{\partial \beta_0}$

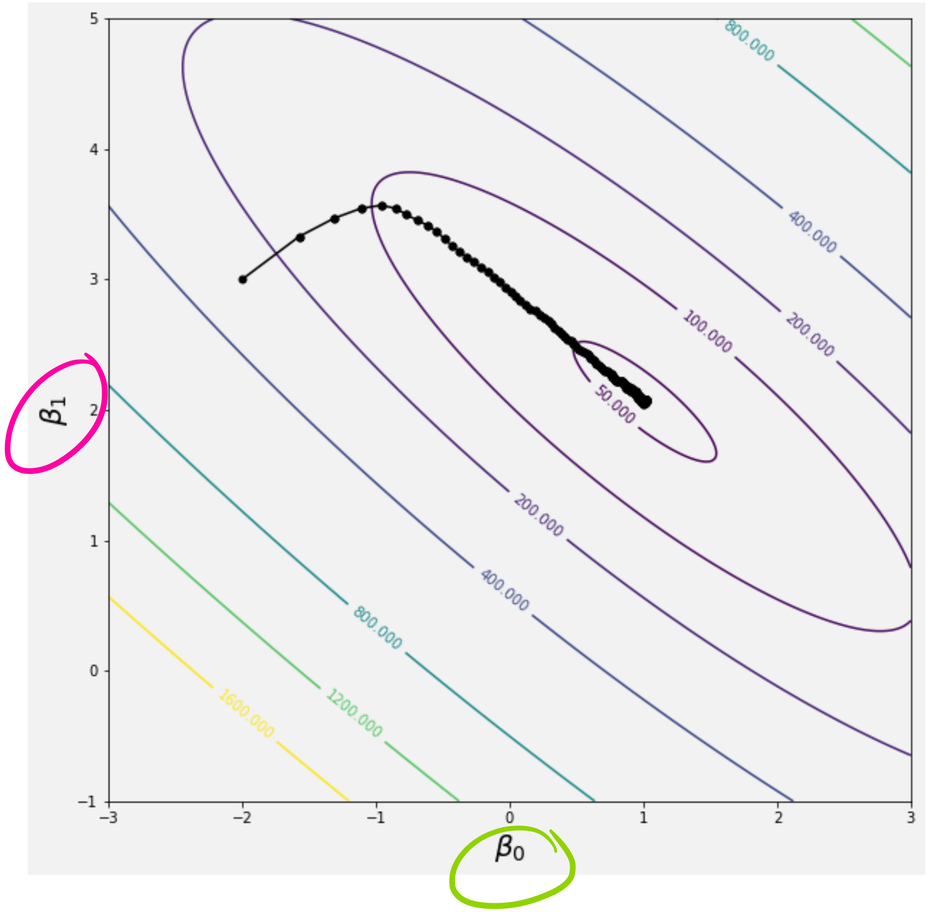
in β_1 direction, $-\frac{\partial SSE}{\partial \beta_1}$



Simple Linear Regression

- In 1-dimension the SSE is a parabola. In 2-dimensions it's a bowl-like surface.

$$SSE = \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_i)]^2$$



- But how do we move **downhill in 2D?**
  - Move downhill in each coordinate direction
  - Use partial derivatives

$$\frac{\partial SSE}{\partial \beta_0} = \sum_{i=1}^{n} \frac{\partial}{\partial \beta_0} [y_i - (\beta_0 + \beta_1 x_i)]^2$$

$$= \sum_{i=1}^{n} 2[y_i - (\beta_0 + \beta_1 x_i)](-1)$$

$$\frac{\partial SSE}{\partial \beta_1} = \sum_{i=1}^{n} 2x_i [y_i - (\beta_0 + \beta_1 x_i)](-1)$$

Gradient Descent for SLR

- In 1-dimension, Gradient Descent was

$$z^{(k+1)} = z^{(k)} - \eta \cdot f'(z^{(k)})$$

- In 2-dimensions, we have the following:

$$\beta_0^{(k+1)} = \beta_0^{(k)} - \eta \frac{\partial SSE}{\partial \beta_0^{(k)}}$$

$$\beta_1^{(k+1)} = \beta_1^{(k)} - \eta \frac{\partial SSE}{\partial \beta_1^{(k)}}$$

Gradient Descent for SLR

- In 1-dimension, Gradient Descent was

$$z^{(k+1)} = z^{(k)} - \eta \cdot f'(z^{(k)})$$

- In 2-dimensions, we have the following:

$$\beta_0^{(k+1)} = \beta_0^{(k)} - \eta \sum_{i=1}^{n} -2 \cdot \left[y_i - (\beta_0^{(k)} + \beta_1^{(k)} x_i) \right]$$

$$\beta_1^{(k+1)} = \beta_1^{(k)} - \eta \sum_{i=1}^{n} -2 \cdot \left[y_i - (\beta_0^{(k)} + \beta_1^{(k)} x_i) \right] x_i$$

$\frac{\partial SSE}{\partial \beta_0}$

$\frac{\partial SSE}{\partial \beta_1}$

Gradient Descent for SLR

- In 1-dimension, Gradient Descent was

$$z^{(k+1)} = z^{(k)} - \eta \cdot f'(z^{(k)})$$

- In 2-dimensions, we have the following:

$$\beta_0^{(k+1)} = \beta_0^{(k)} - \eta \sum_{i=1}^{n} -2 \cdot \left[y_i - (\beta_0^{(k)} + \beta_1^{(k)} x_i) \right]$$

$$\beta_1^{(k+1)} = \beta_1^{(k)} - \eta \sum_{i=1}^{n} -2 \cdot \left[y_i - (\beta_0^{(k)} + \beta_1^{(k)} x_i) \right] x_i$$

- **Question**: Does anything about this seem slow?

Every update requires a summer all our data.
What if we have 500 GB of data?

Gradient Descent for SLR

- To get more rapid updates, update parameters one data point at a time

- For each point (x_i, y_i) in dataset, update parameters

$$\beta_0 \leftarrow \beta_0 - \eta \cdot (-2) \cdot [y_i - (\beta_0 + \beta_1 x_i)]$$

$$\beta_1 \leftarrow \beta_1 - \eta \cdot (-2) \cdot [y_i - (\beta_0 + \beta_1 x_i)]\, x_i$$

Note: ① removed the (k+1) and (k) indices.

② we now have an update for each data point.

Fix $i = 1$. update.
Fix $i = 2$. update.

Gradient Descent for SLR

- To get more rapid updates, update parameters one data point at a time

- For each point (x_i, y_i) in dataset, update parameters

$$\begin{aligned} \beta_0 &\leftarrow \beta_0 - \eta \cdot (-2) \cdot [y_i - (\beta_0 + \beta_1 x_i)] \\ \beta_1 &\leftarrow \beta_1 - \eta \cdot (-2) \cdot [y_i - (\beta_0 + \beta_1 x_i)]\, x_i \end{aligned}$$

- **Important Note**: Better if you loop through dataset randomly. Avoids biases in order of data.

Stochastic Gradient Descent

- To get more rapid updates, update parameters one data point at a time

- For each point (x_i, y_i) in **RANDOMLY SHUFFLED** dataset, update parameters

$$\begin{aligned} \beta_0 &\leftarrow \beta_0 - \eta \cdot (-2) \cdot [y_i - (\beta_0 + \beta_1 x_i)] \\ \beta_1 &\leftarrow \beta_1 - \eta \cdot (-2) \cdot [y_i - (\beta_0 + \beta_1 x_i)]\, x_i \end{aligned}$$

Stochastic Gradient Descent

- To get more rapid updates, update parameters one data point at a time
- For each point (x_i, y_i) in **RANDOMLY SHUFFLED** dataset, update parameters

$$\beta_0 \leftarrow \beta_0 - \eta \cdot (-2) \cdot [y_i - (\beta_0 + \beta_1 x_i)]$$

$$\beta_1 \leftarrow \beta_1 - \eta \cdot (-2) \cdot [y_i - (\beta_0 + \beta_1 x_i)]\, x_i$$

- One pass over the entire data set is called an **epoch**.

we pick this,

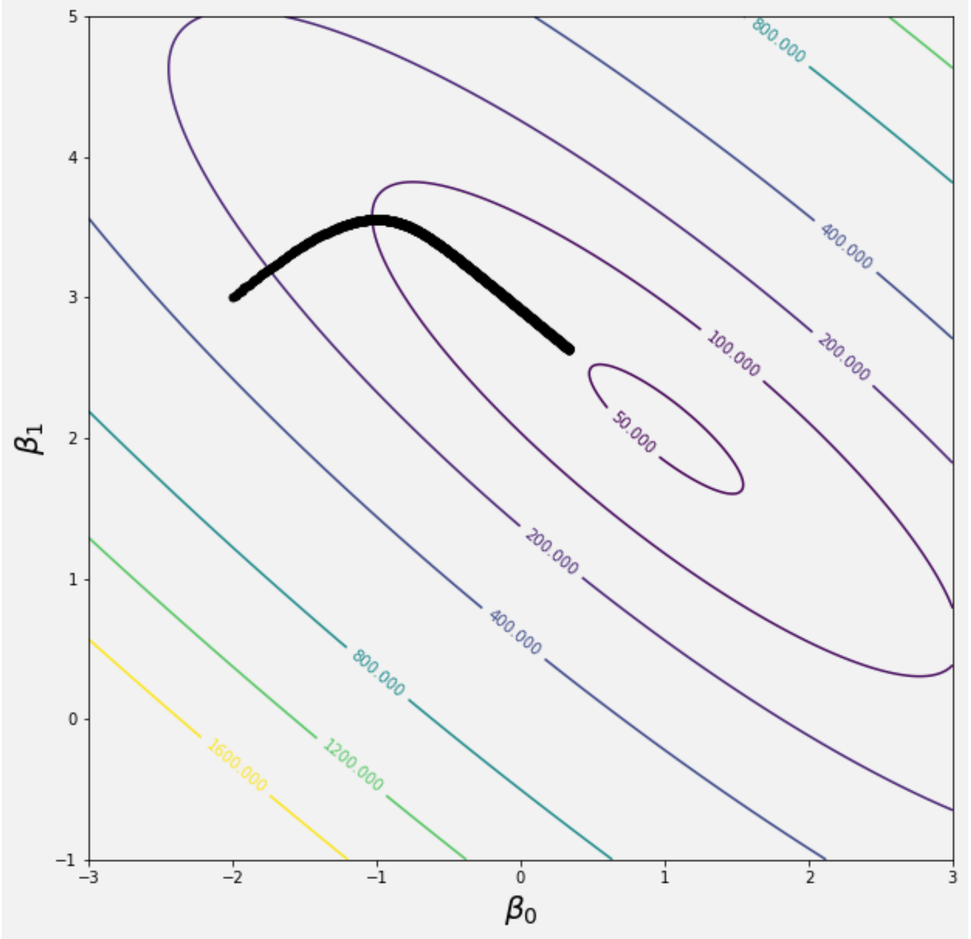
Stopping Criterion: $\|\beta^{new} - \beta^{old}\| \leq$ small , then stop.

NB: cauchy sequence.

Euclidean Distance $\|\cdot\|$ aka Euclidean norm. Sometimes, l_1 norm, l^∞ norm.

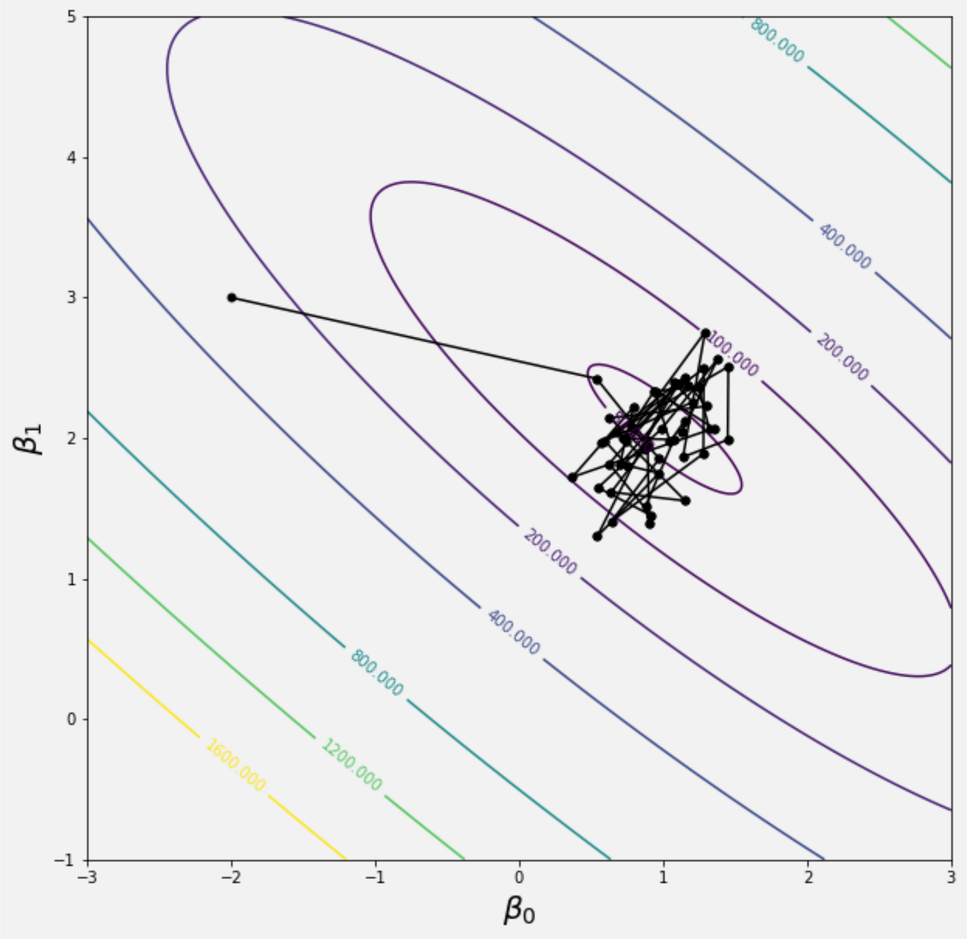
The Importance of the Learning Rate

- If the learning rate is **too small**, this process will take a long time to converge.



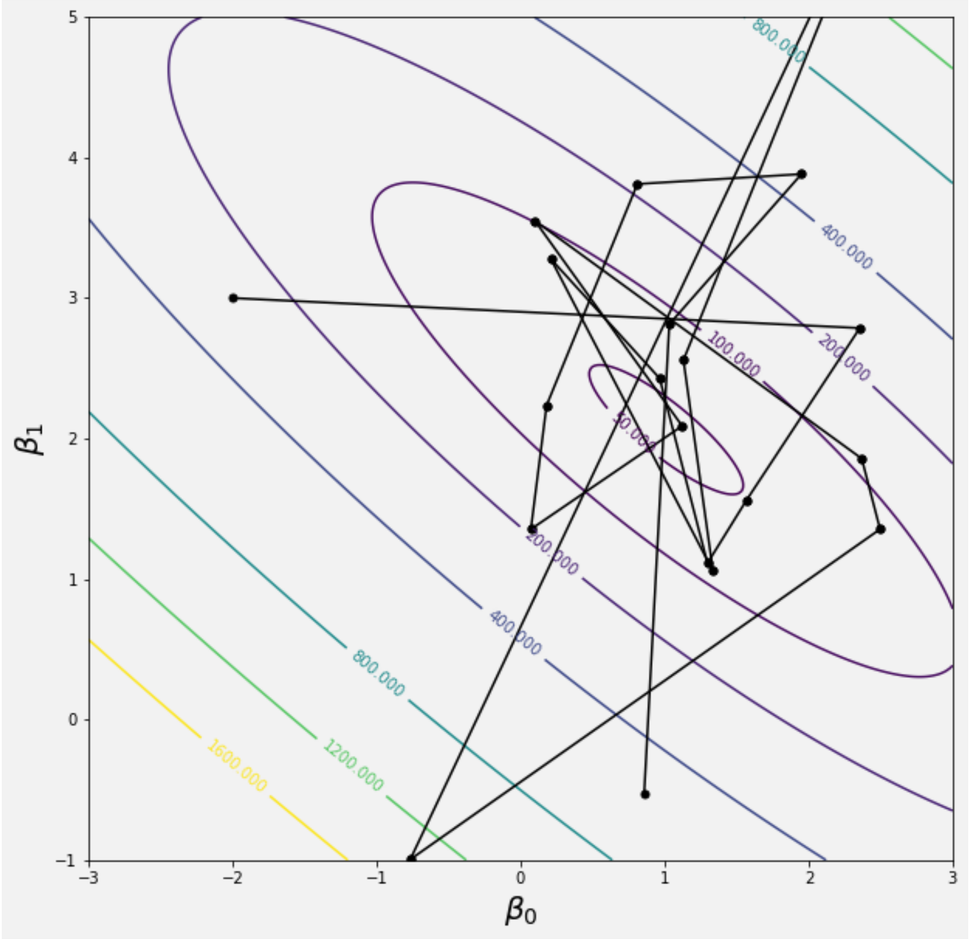
The Importance of the Learning Rate

- If the learning rate is **too large**, the process may oscillate and bounce around.



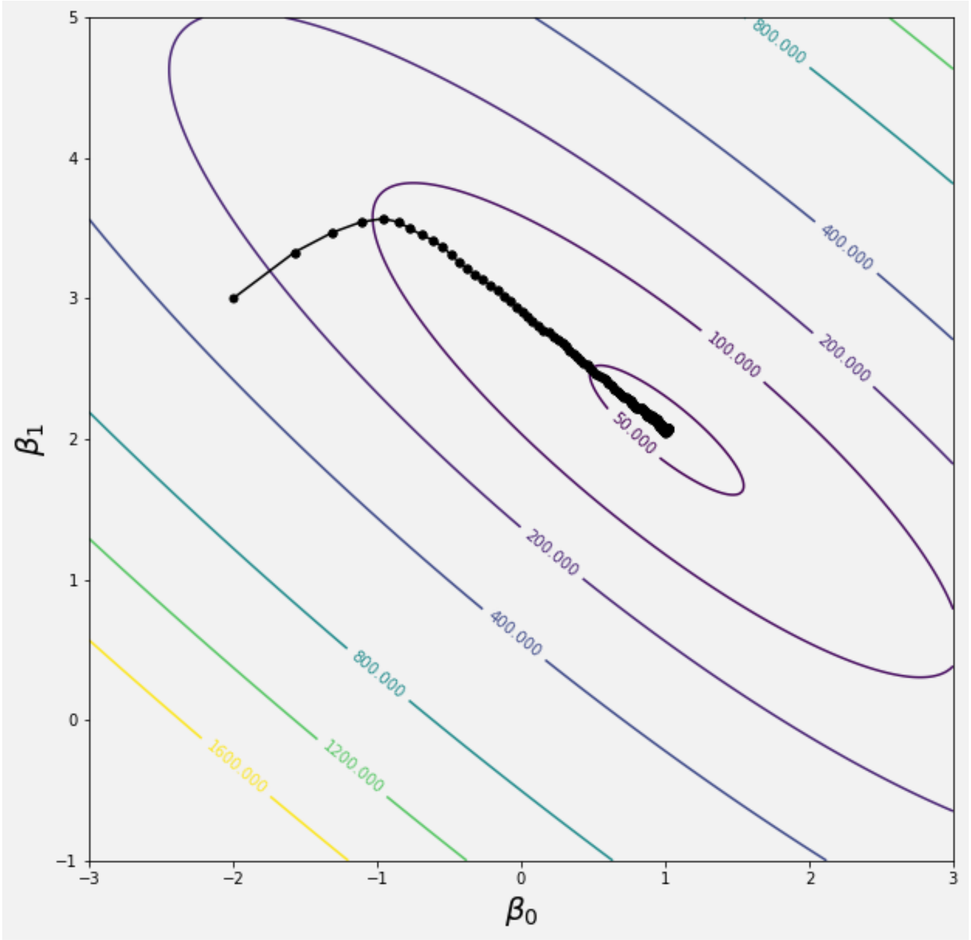
The Importance of the Learning Rate

- If the learning rate is **way too large**, the process may diverge entirely!



The Importance of the Learning Rate

- Generally, we have to **tune** the learning rate to get it just right.



SGD for Logistic Regression

labels.

- Recall that in LogReg we have data of the form (x_i, y_i) where $y_i \in \{0, 1\}$
- Our model was

$$p = p(y = 1 \mid x) = \text{sigm}(\beta_0 + \beta_1 x) \qquad \Rightarrow p(y = 0 \mid x) = 1 - \text{sigm}(\beta_0 + \beta_1 x)$$

- This can be written more compactly as

$$p(y \mid x) = \text{sigm}(\beta_0 + \beta_1 x)^y \ (1 - \text{sigm}(\beta_0 + \beta_1 x))^{1-y}$$

- Notice that this looks like a Bernoulli random variable with mean

$\text{sigm}(\beta_0 + \beta_1 x)$

- The Likelihood tells us how well the parameters fit the data and model

$$\begin{aligned} L(\beta_0, \beta_1) &= \Pi_{i=1}^n p(y_i \mid x_i; \beta_0, \beta_1) \\ &= \Pi_{i=1}^n \text{sigm}(\beta_0 + \beta_1 x_i)^{y_i} (1 - \text{sigm}(\beta_0 + \beta_1 x_i))^{1-y_i} \end{aligned}$$

SGD for Logistic Regression

$$
\begin{aligned}
L(\beta_0, \beta_1) &= \Pi_{i=1}^{n} p(y_i \mid x_i; \beta_0, \beta_1) \\
&= \Pi_{i=1}^{n} \text{sigm}(\beta_0 + \beta_1 x_i)^{y_i} (1 - \text{sigm}(\beta_0 + \beta_1 x_i))^{1-y_i}
\end{aligned}
$$

- This is messy because of all the products. We'll turn them into sums by taking the log

$$p(y \mid x) = \text{sigm}(\beta_0 + \beta_1 x)^{y} \ (1 - \text{sigm}(\beta_0 + \beta_1 x))^{1-y}$$

- Can make it even nicer by taking the negative, to the the so-called Negative Log-Likelihood

$$
\begin{aligned}
NLL(\beta_0, \beta_1) &= -\log \left(\Pi_{i=1}^{n} p(y_i \mid x_i; \beta_0, \beta_1) \right) \\
&= -\sum_{i=1}^{n} y_i \log \text{sigm}(\beta_0 + \beta_1 x_i) + (1 - y_i) \log(1 - \text{sigm}(\beta_0 + \beta_1 x_i))
\end{aligned}
$$

SGD for Logistic Regression

$$
\begin{aligned}
NLL(\beta_0, \beta_1) &= -\log\left(\Pi_{i=1}^{n} p(y_i \mid x_i; \beta_0, \beta_1)\right) \\
&= -\sum_{i=1}^{n} y_i \log \operatorname{sigm}(\beta_0 + \beta_1 x_i) + (1 - y_i) \log(1 - \operatorname{sigm}(\beta_0 + \beta_1 x_i))
\end{aligned}
$$

- Need partial derivatives of $NLL(\beta_0, \beta_1)$ w.r.t. parameters (Try these yourself!)

$$
\frac{\partial NLL(\beta_0, \beta_1)}{\partial \beta_0} = \sum_{i=1}^{n} \left[y_i - \operatorname{sigm}(\beta_0 + \beta_1 x_i)\right]
$$

$$
\frac{\partial NLL(\beta_0, \beta_1)}{\partial \beta_1} = \sum_{i=1}^{n} \left[y_i - \operatorname{sigm}(\beta_0 + \beta_1 x_i)\right] x_i
$$

- Like before, we'll only do one data point at a time!

SGD for Logistic Regression

$$
\begin{aligned}
NLL(\beta_0, \beta_1) &= -\log\left(\Pi_{i=1}^{n} p(y_i \mid x_i; \beta_0, \beta_1)\right) \\
&= -\sum_{i=1}^{n} y_i \log \text{sigm}(\beta_0 + \beta_1 x_i) + (1 - y_i) \log(1 - \text{sigm}(\beta_0 + \beta_1 x_i))
\end{aligned}
$$

- Stochastic Gradient Descent for LogReg: Loop over each shuffled data point and do

$$
\begin{aligned}
\beta_0 &\leftarrow \beta_0 - \eta \cdot [y_i - \text{sigm}(\beta_0 + \beta_1 x_i)] \\
\beta_1 &\leftarrow \beta_1 - \eta \cdot [y_i - \text{sigm}(\beta_0 + \beta_1 x_i)]\, x_i
\end{aligned}
$$