CSCI 3022

intro to data science with probability & statistics

Lecture 28
April 27, 2018

Solution Techniques for
Linear and Logistic Regression

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

Dan Larremore

Practicum & Final Exam

- The **Practicum** is due at 11:55pm next Wednesday.
  - Three edits posted for clarification. See the latest file on Github, colored text.
  - Problem 1 can be done entirely with simulation if you wish.
  - What do we assume when we do a t-test for the mean? "hint"

- **Final Exam** Sunday May 6 from 1:30 -4 pm in FLMG 155.
  - If you emailed me w/ a 4:30 final or 1.5X, your final starts at 12:00. Look for email.
  - Cumulative but will emphasize material since midterm
  - Bring a calculator
  - Two-page note-sheet. Handwritten. No magnifying glasses.

mult choice + free response

- **Review** in-class next Wednesday.
  - Come with questions!
  - If there are no questions, I'll lock the doors and Kyle & Sofie 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: $SSE = \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_i)]^2$

p = 1

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

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

$$\text{sigm}(z) = \frac{1}{1 + e^{-z}}$$

where the values of y are 0 or 1.

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

$$\text{odds} = \frac{p}{1-p}$$

$$\text{log odds} = \log\left(\frac{p}{1-p}\right)$$

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

IRL

- 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 guess

one criterion
many criteria

from $x^{(0)}$, get $x^{(0)} \longrightarrow x^{(1)}$

repeat $x^{(1)} \longrightarrow x^{(2)}$

repeat until stopping criterion.

rule for when we stop repeating.

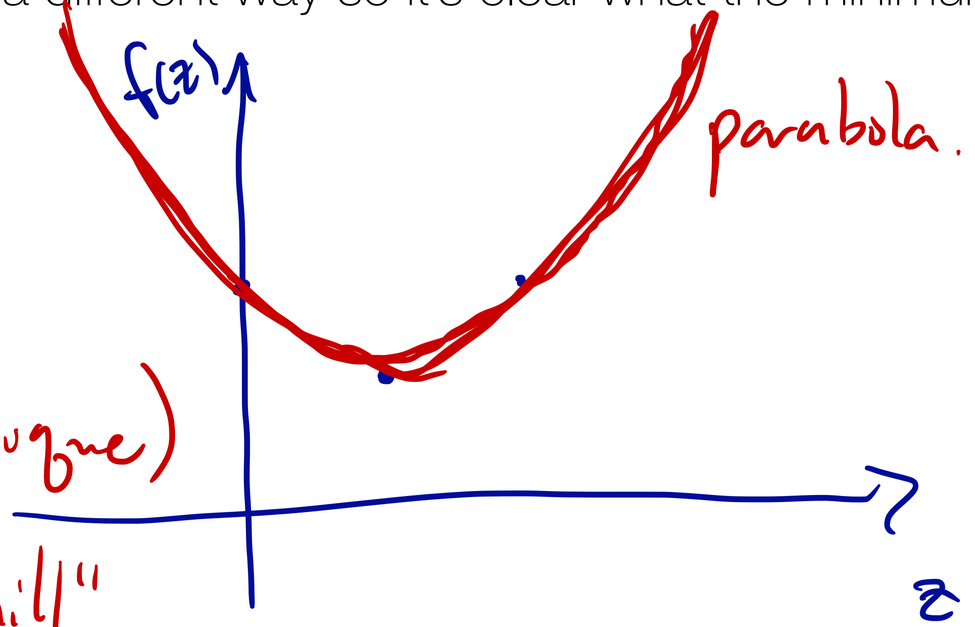
For example . . .

- 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?

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

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



1 minimum (unique)

everything is "downhill" to that min.

For example . . .

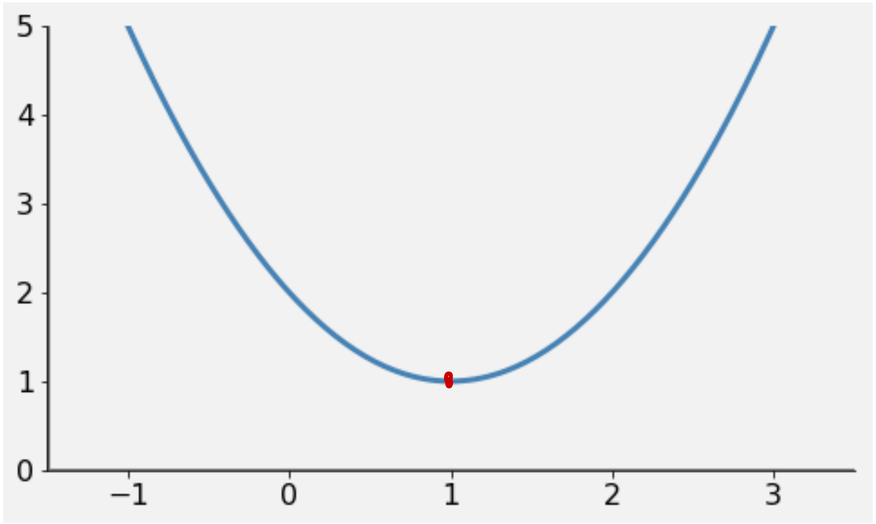
- Suppose you want to find the minimum of the function $f(z) = z^2 - 2z + 2$
- Can we rewrite this function in a different way so it's clear what the minimum is?

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

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

one minimum

many minima



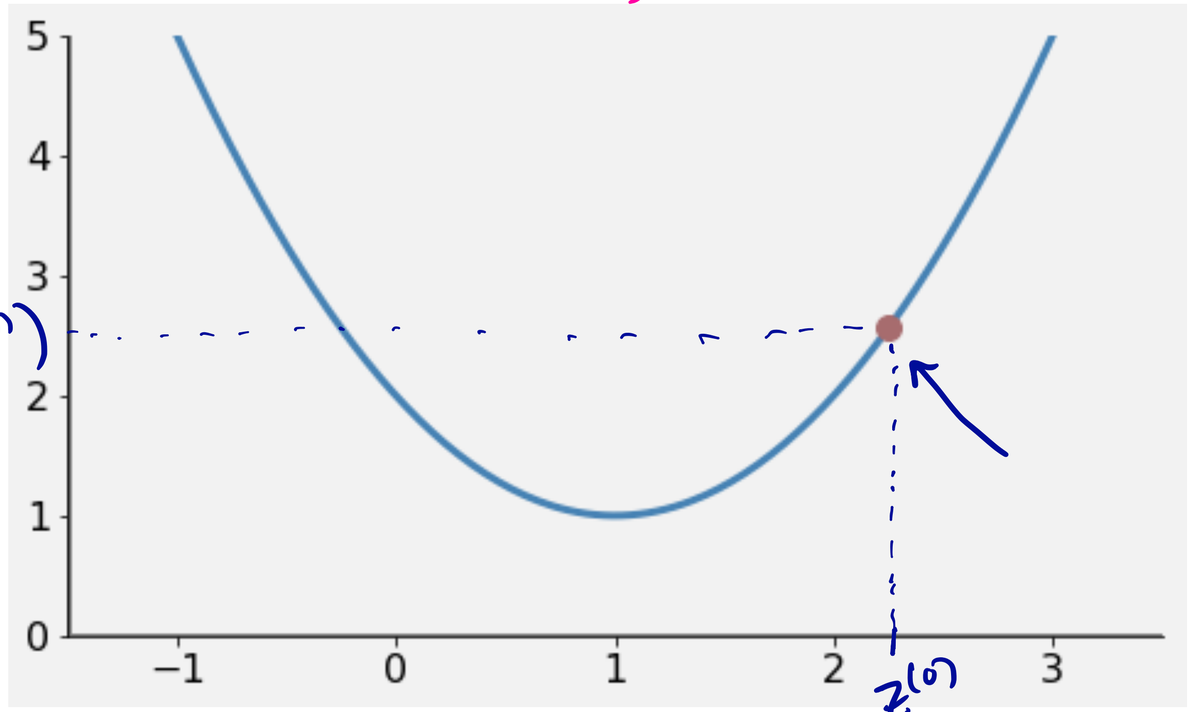
Concave upward everywhere

or convex

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? left, toward a smaller f(z) value



Reason:

Go Downhill.

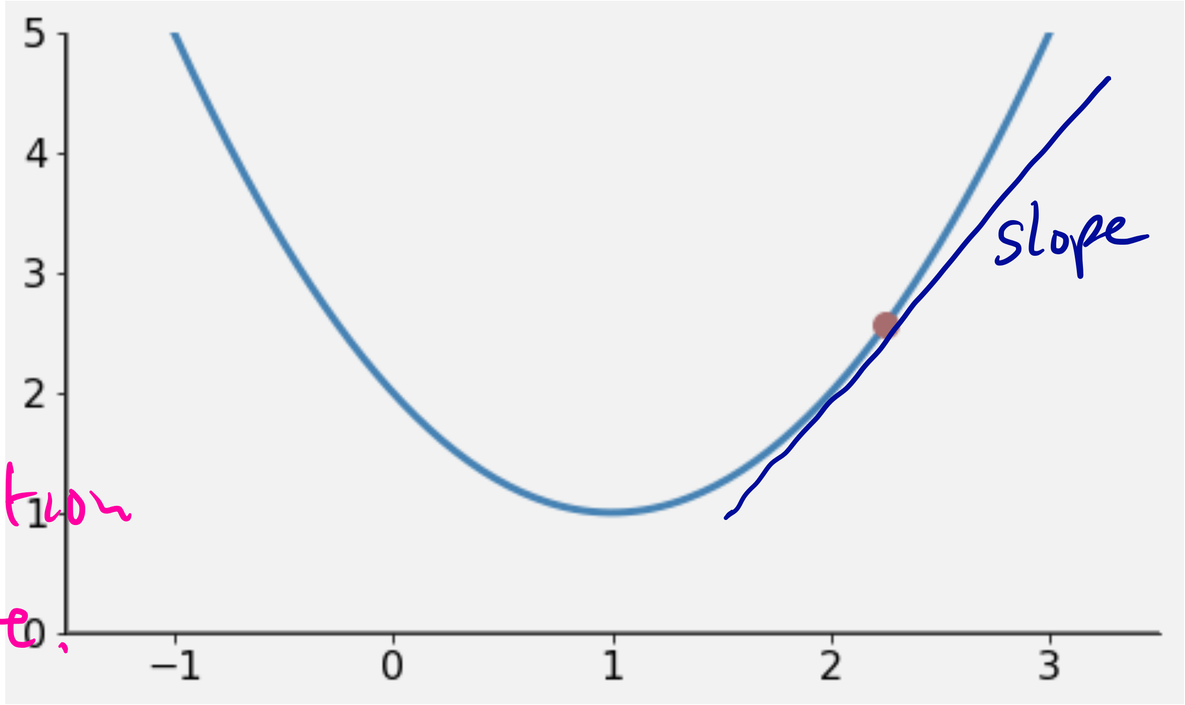
For example . . .

- 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 + ⇒ move left

If slope is − ⇒ move right.

Move in opp. direction of the derivative.

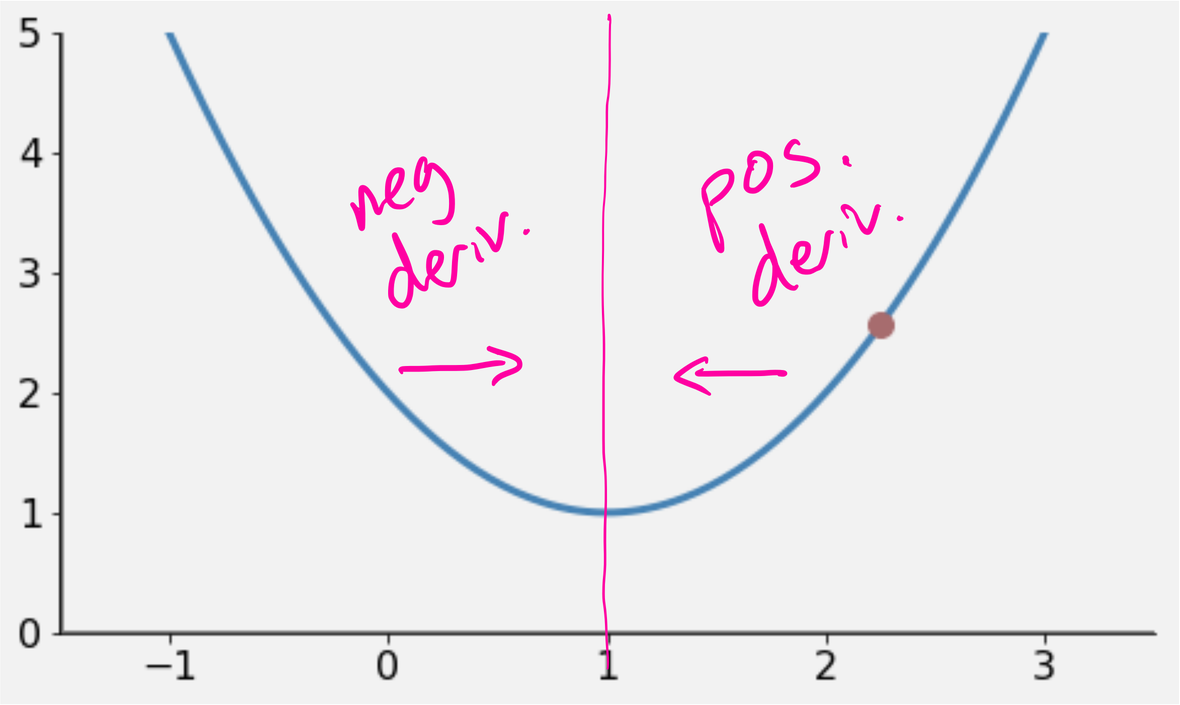


For example . . .

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

- 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 . . .

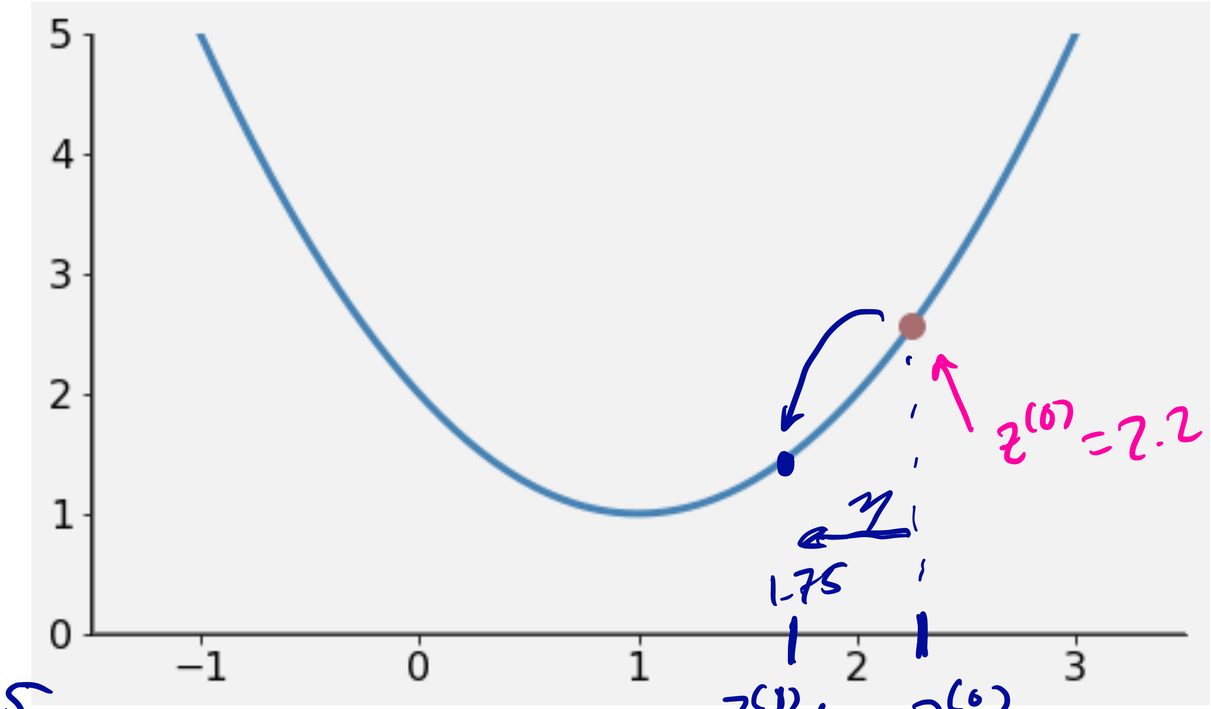
- Suppose you want to find the minimum of the function $f(z) = z^2 - 2z + 2$
- **Question:** But which way is down? **Answer:** Derivative tells you uphill. Go opposite direction
- **Question**: How far should we move? **Answer**: Hmmmm, just pick a small step size like $\eta = 0.5$

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

$f'(2.25) = 4.5 - 2$
$= 2.5$

2.5 is positive
⇒ move left.

$z^{(1)} = z^{(0)} - \eta$
$= 2.25 - 0.5 = 1.75$



η eta

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 $\eta = 0.5$

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

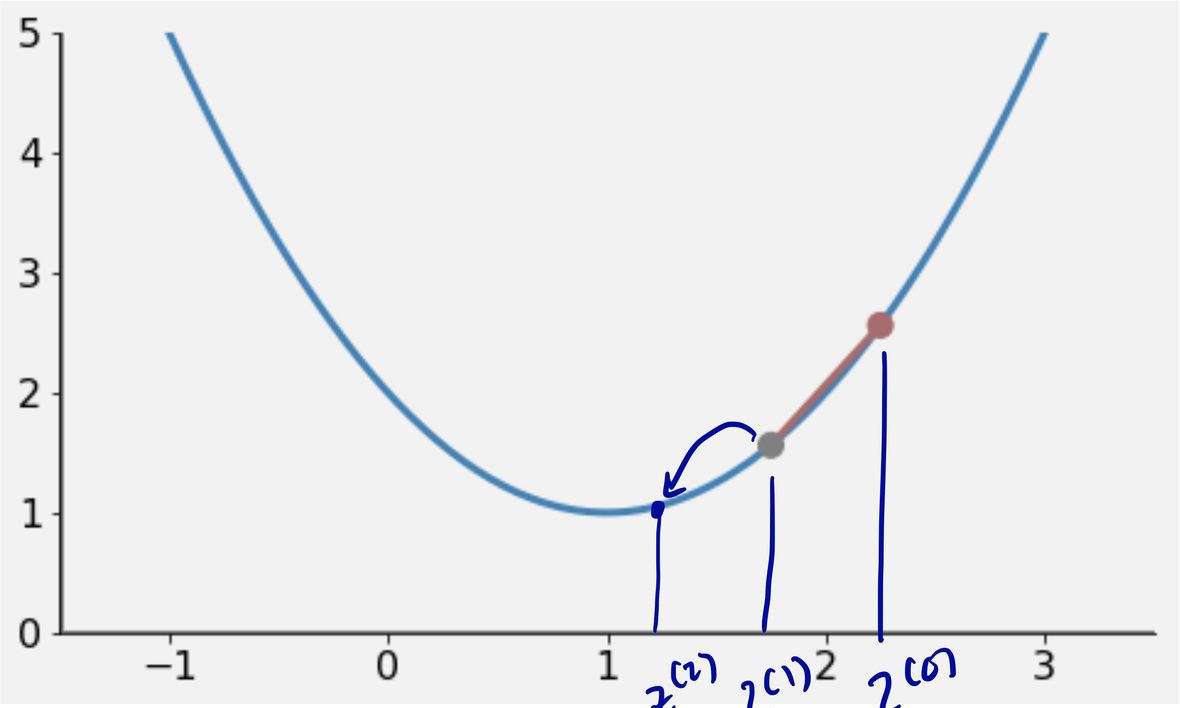
$f'(1.75) = 2 \cdot 1.75 - 2$

$= 1.5 > 0$

go left

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

$1.75 - 0.5 = 1.25$



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 $\eta = 0.5$

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

$f'(1.25) = 2.5 - 2$

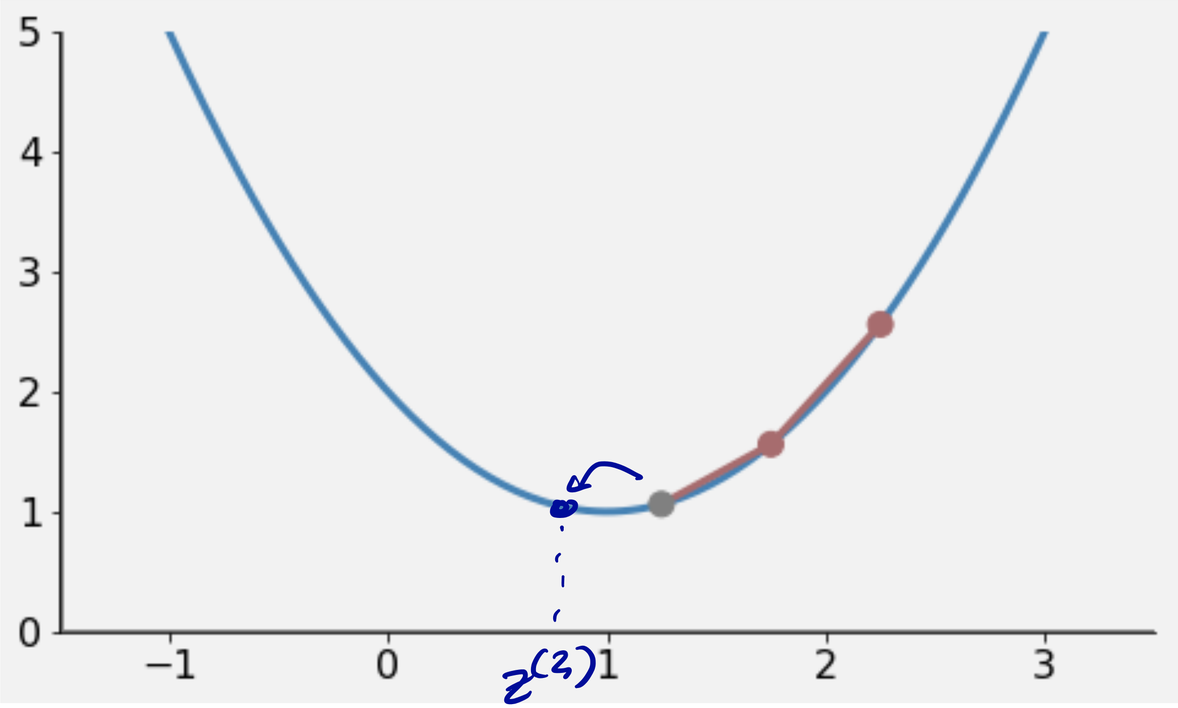
$= 0.5 > 0$

$\Rightarrow$ left.

$z^{(3)} = z^{(2)} - \eta$

$= 1.25 - 0.5$

$= 0.75$



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 $\eta = 0.5$

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

$f'(0.75) = 1.5 - 2$

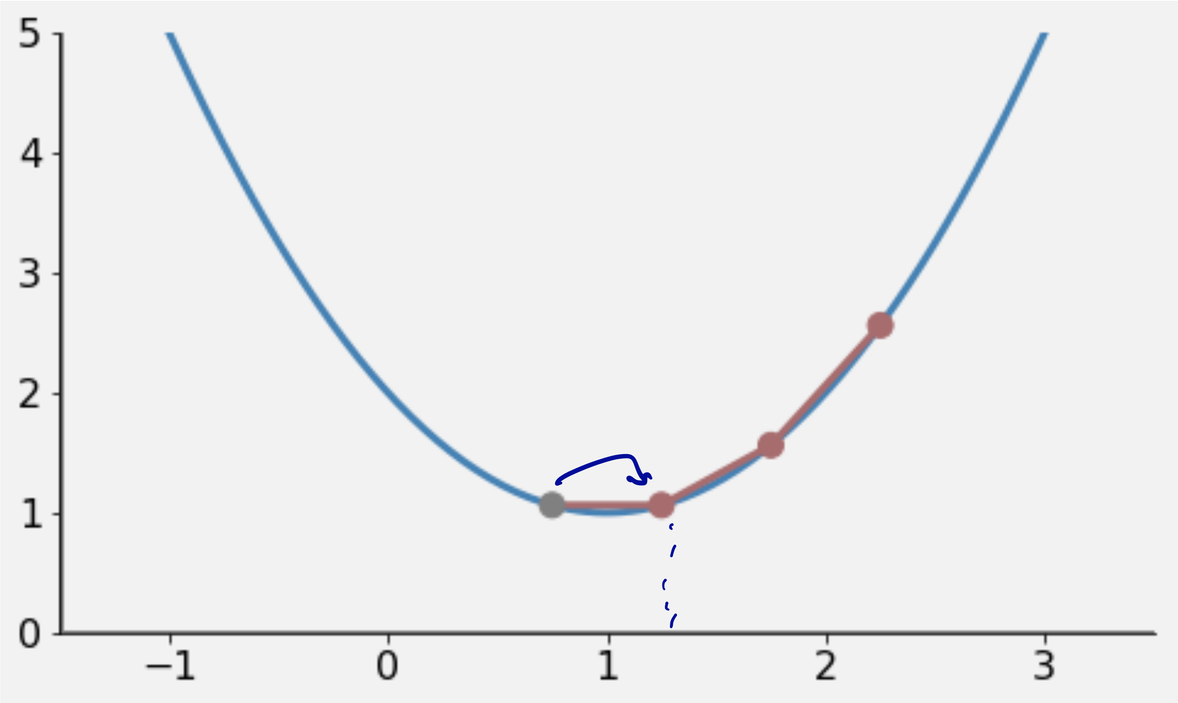
$= -0.5 < 0$

$\Rightarrow$ go right!

$z^{(4)} = z^{(3)} + \eta$

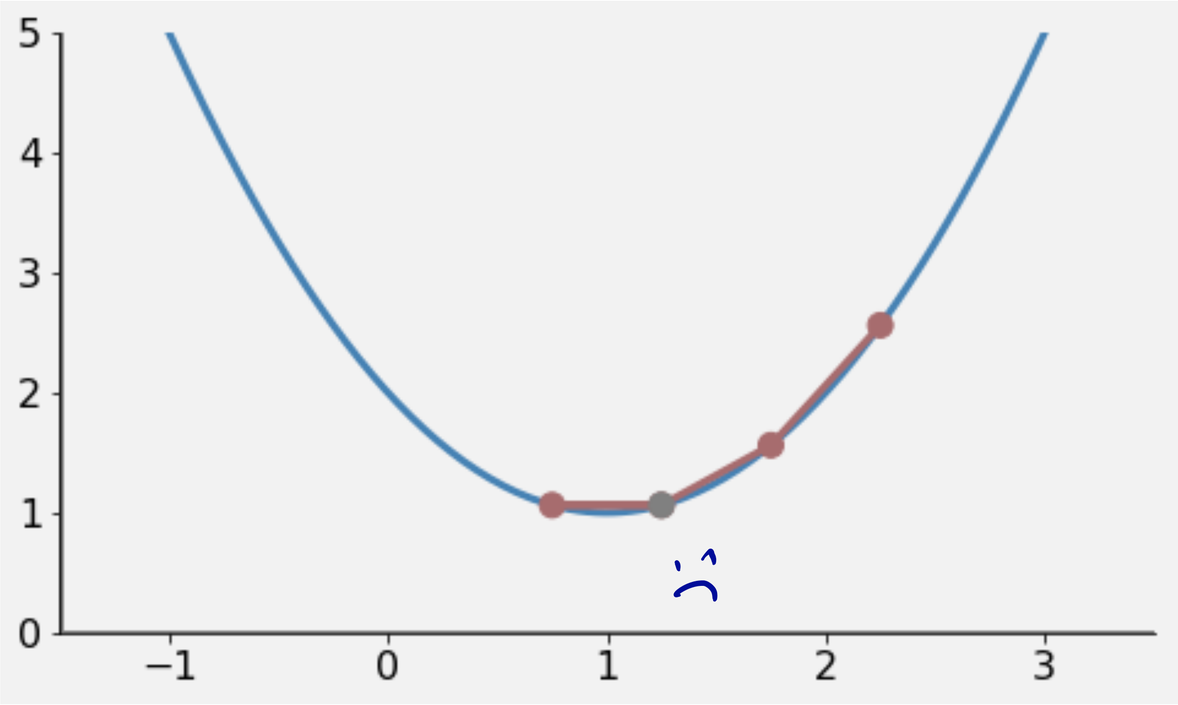
$= 0.75 + 0.5$

$= 1.25$



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 . . .

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

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

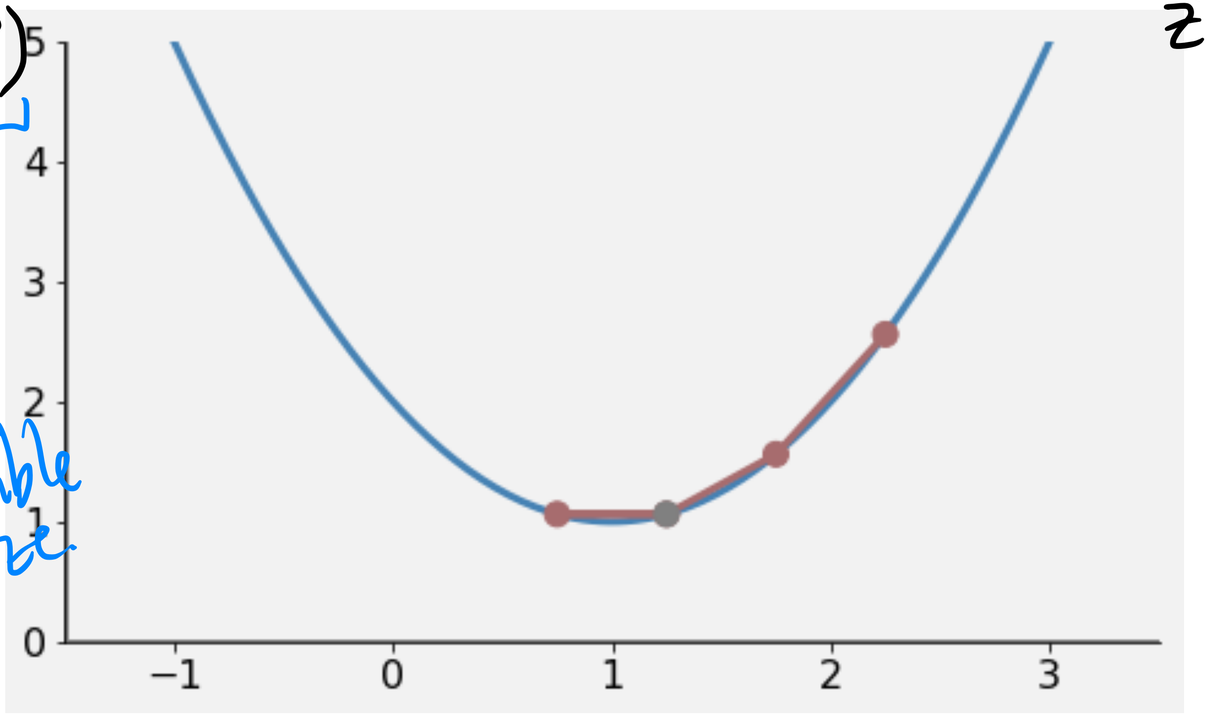
$$z^{(k+1)} = z^{(k)} - \eta(2z^{(k)} - 2)$$

updating from prev. point

learning rate.

variable step size

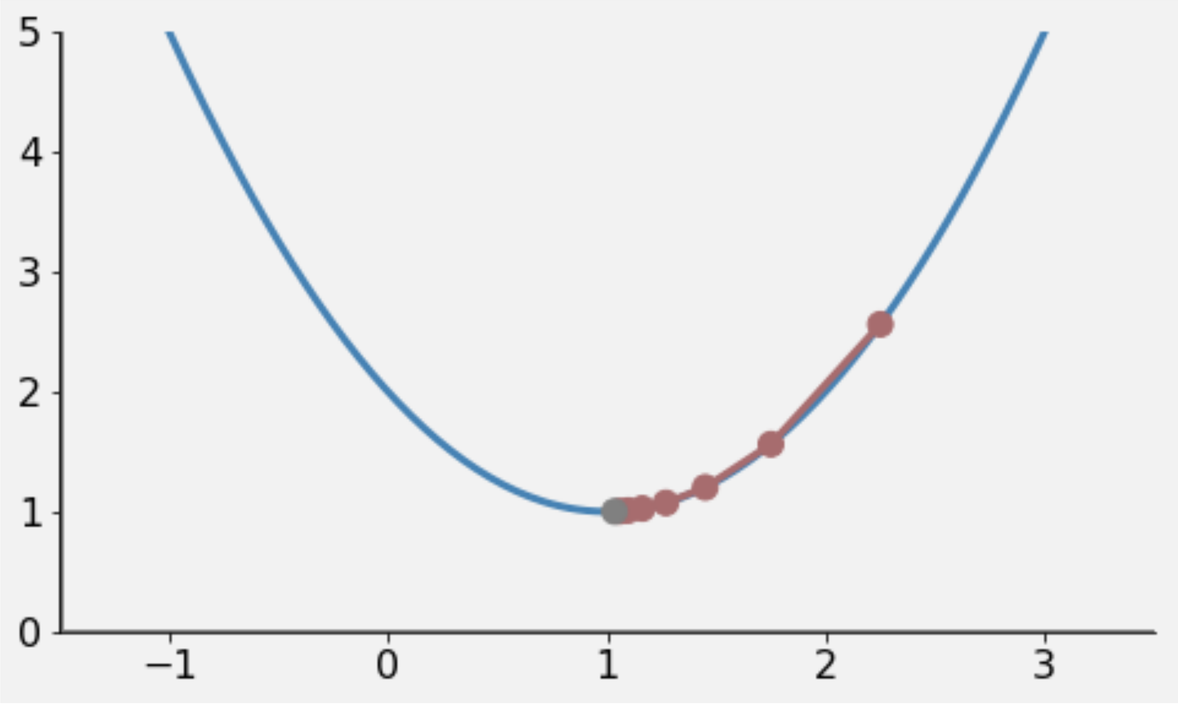
move opp. the sign deriv.



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.

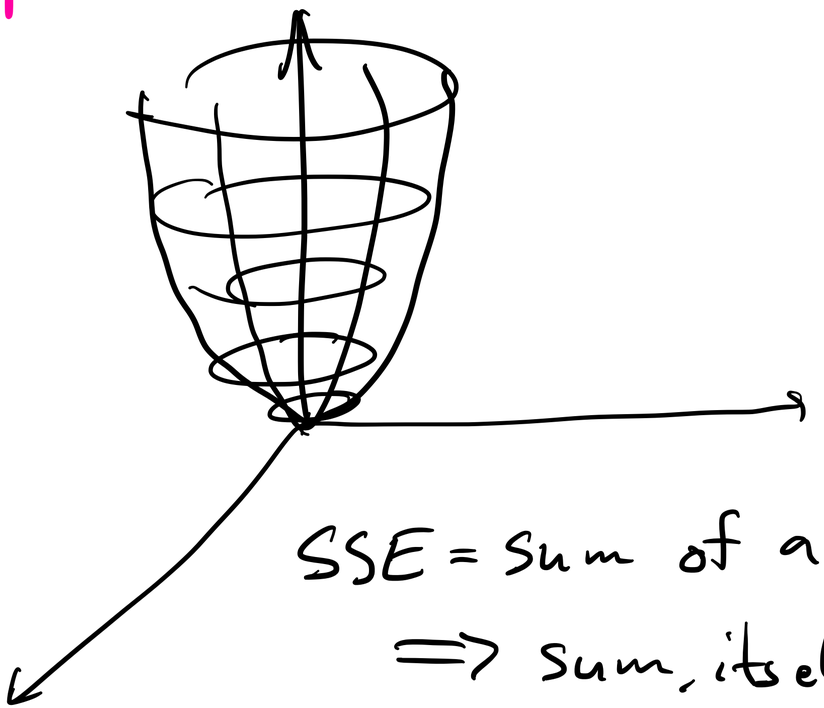
- The difficulty is that this is a function of two variables, which is not quite a simple parabola, like our previous example.

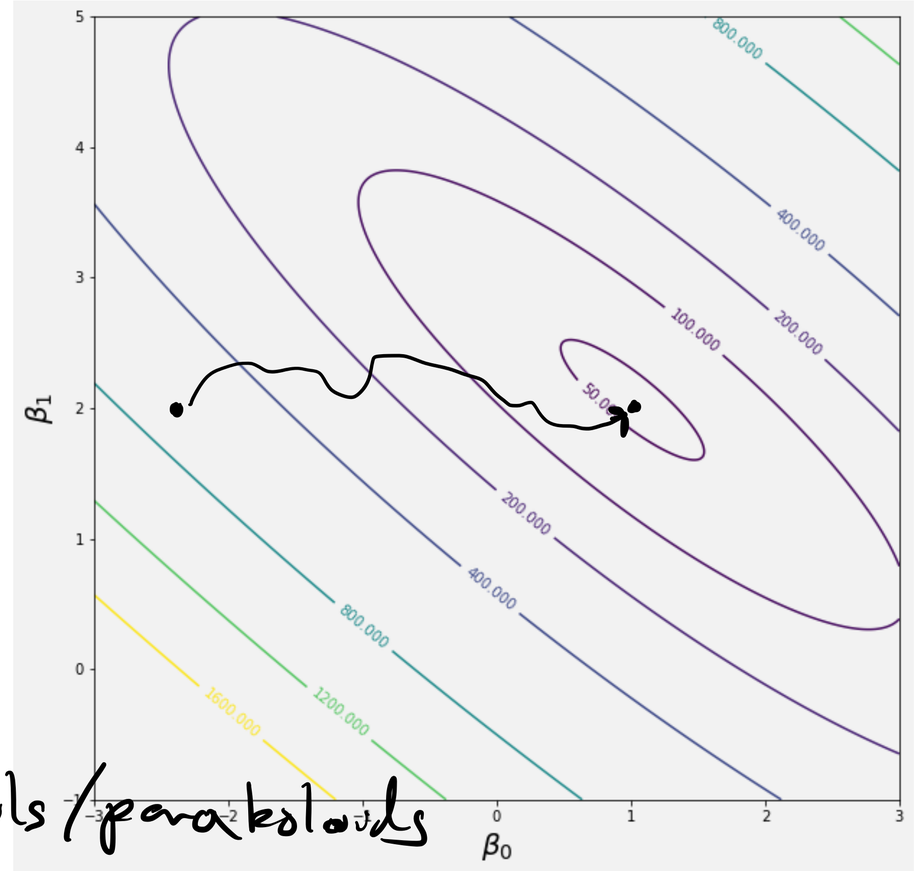
Simple Linear Regression

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

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

~ paraboloid





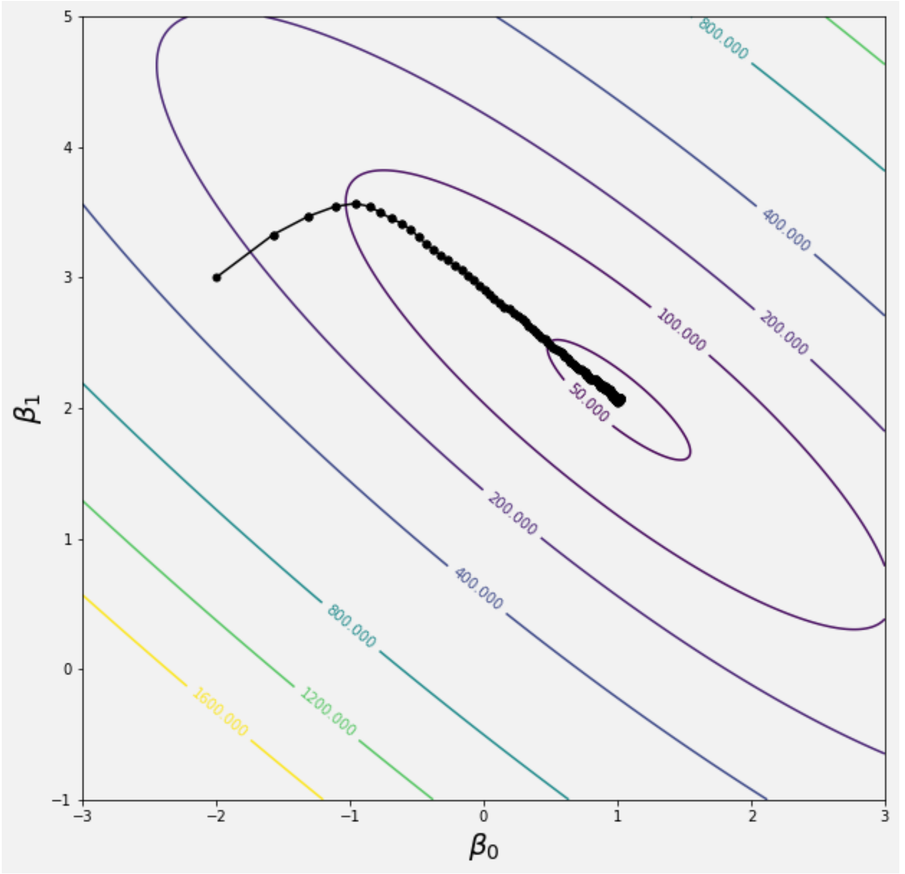
SSE = sum of a set of bowls / paraboloids

⇒ sum, itself, is a bowl.

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

?? What is "downhill" in 2D ?

Simple Linear Regression

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

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

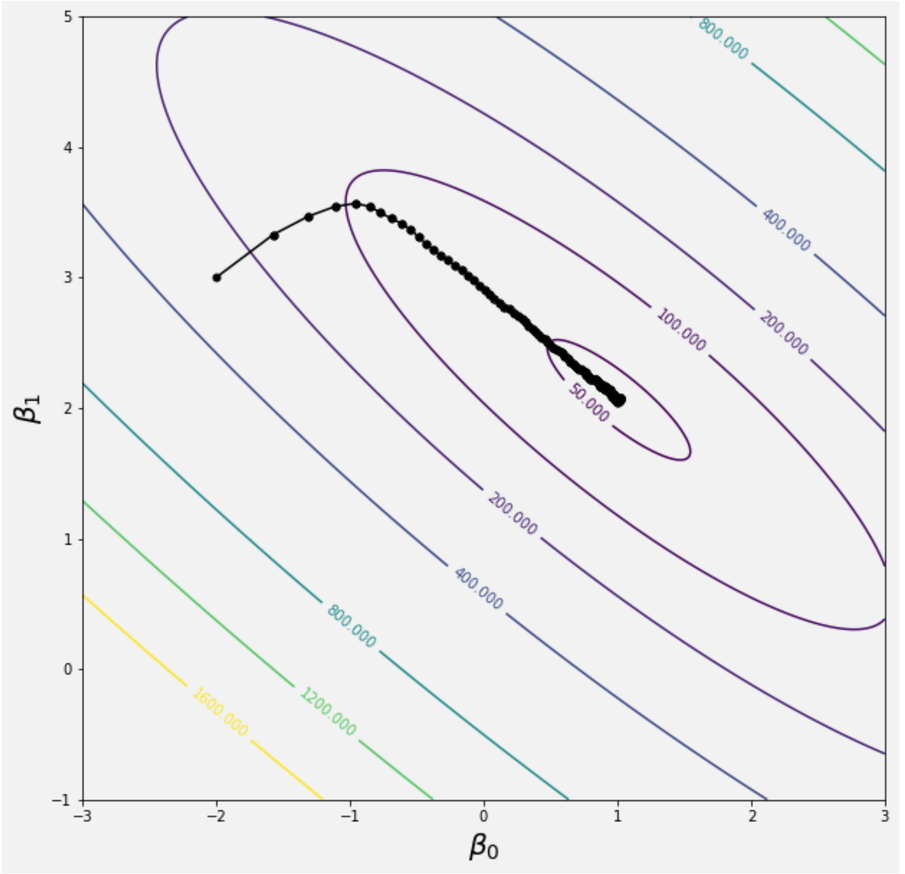
- But how do we move **downhill in 2D?**

Negative of the Gradient.

Negative of each partial derivative.

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

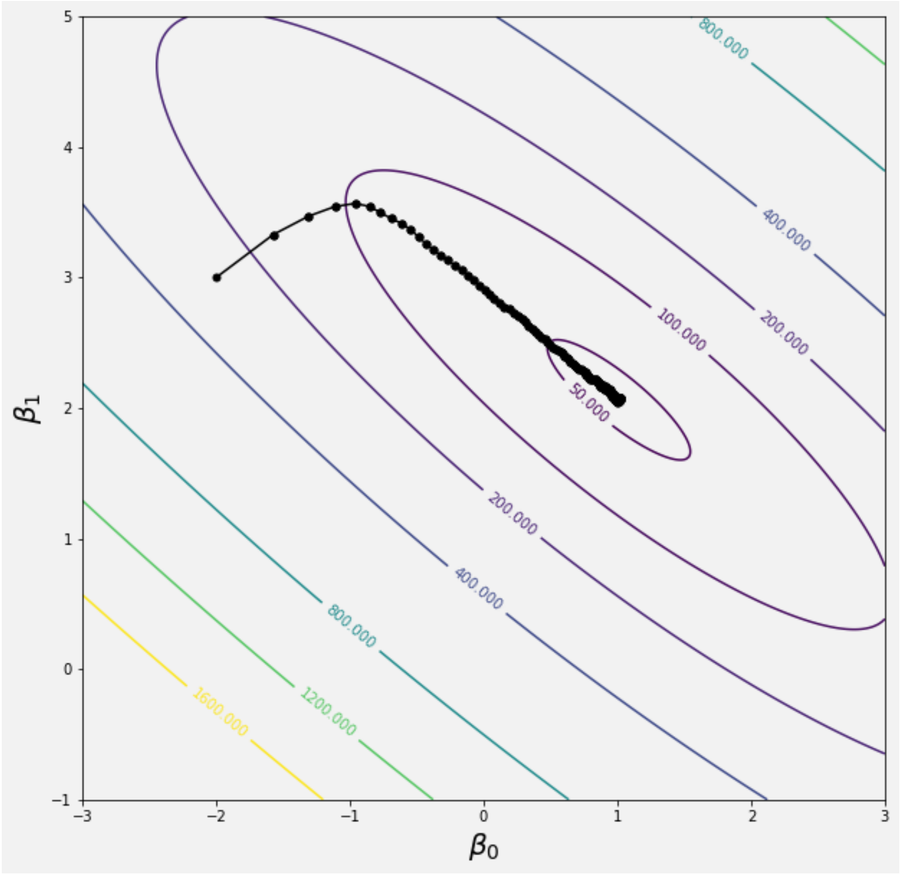
in β_1 direction, $-\frac{\partial SSE}{\partial \beta_1}$



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

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

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

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

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:

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

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

partial

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

think of it like

$\frac{d\,SSE}{d\beta_0}$

while assuming everything else is const.

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:

plugged in.

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

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 sum over all the 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

Saying same thing

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

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

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

new ↑ old ↑

- removed indices $k+1$, k, etc.
- We have update equations. Use 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

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

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

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

Instead of $i = 1, 2, 3, \ldots$

Use shuffled indices.

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

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

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

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**".

$\beta^{\text{after epoch}}$ $\beta^{\text{before epoch}}$

we choose this!

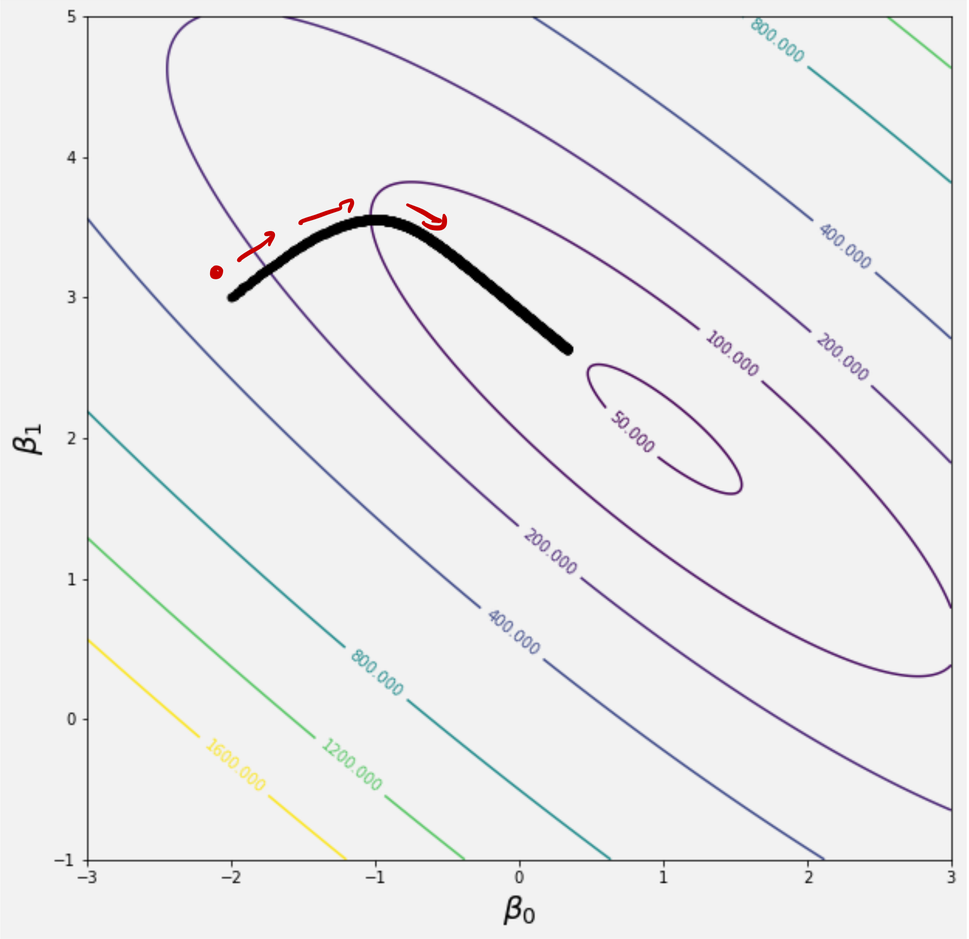
Stopping Criterion: $\| \beta^{new} - \beta^{old} \|$ = small then stop!

NB: Cauchy Sequence!

Euclidean distance aka Euclidean Norm Sometimes l_1 norm. l_∞ norm.

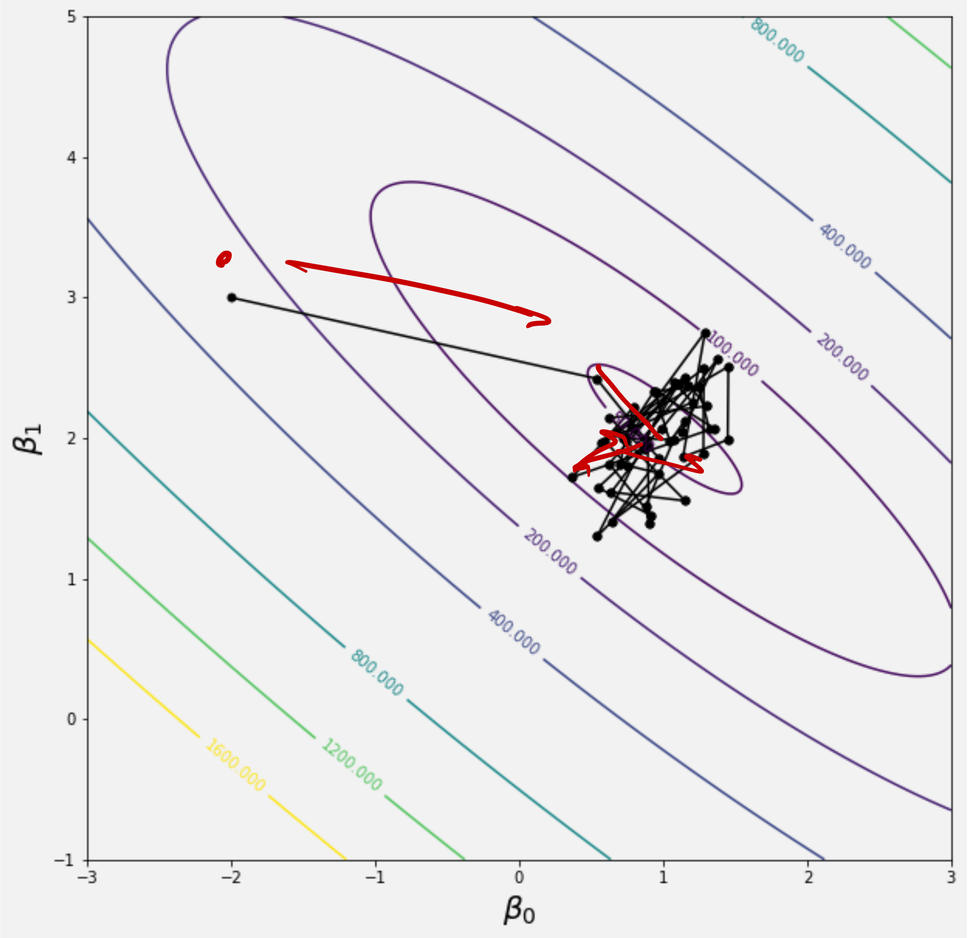
The Importance of the Learning Rate

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



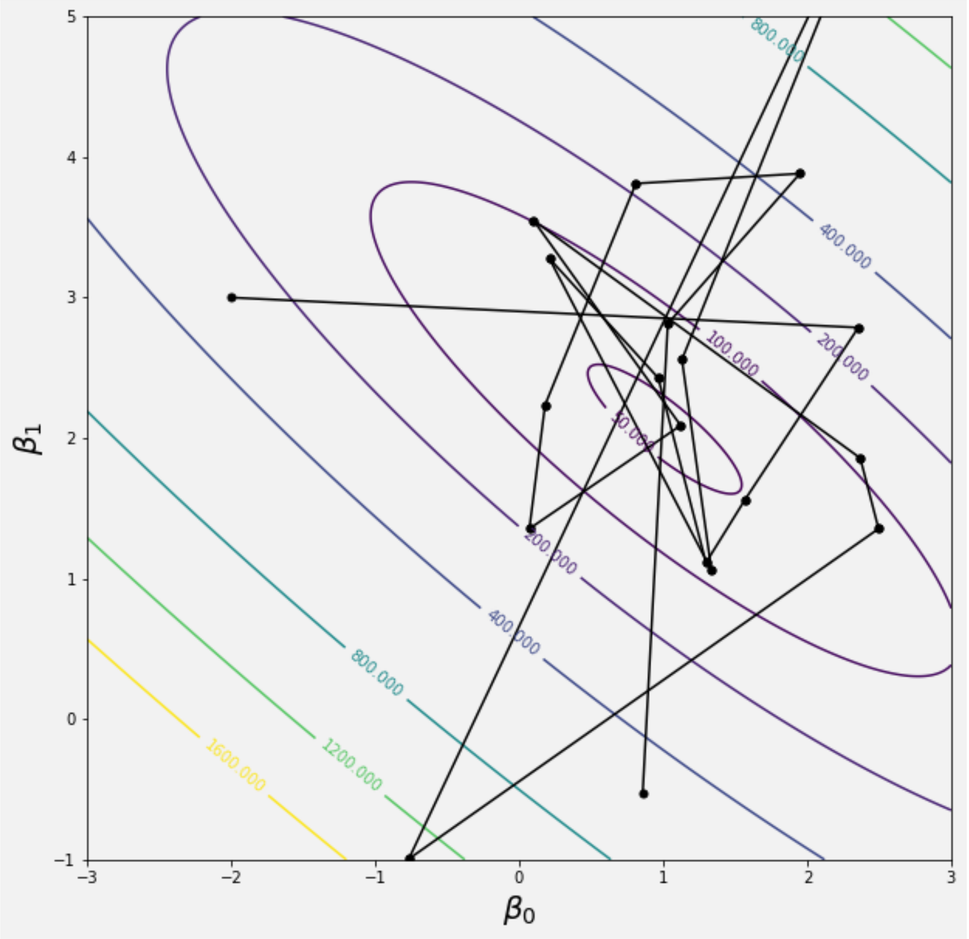
The Importance of the Learning Rate

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



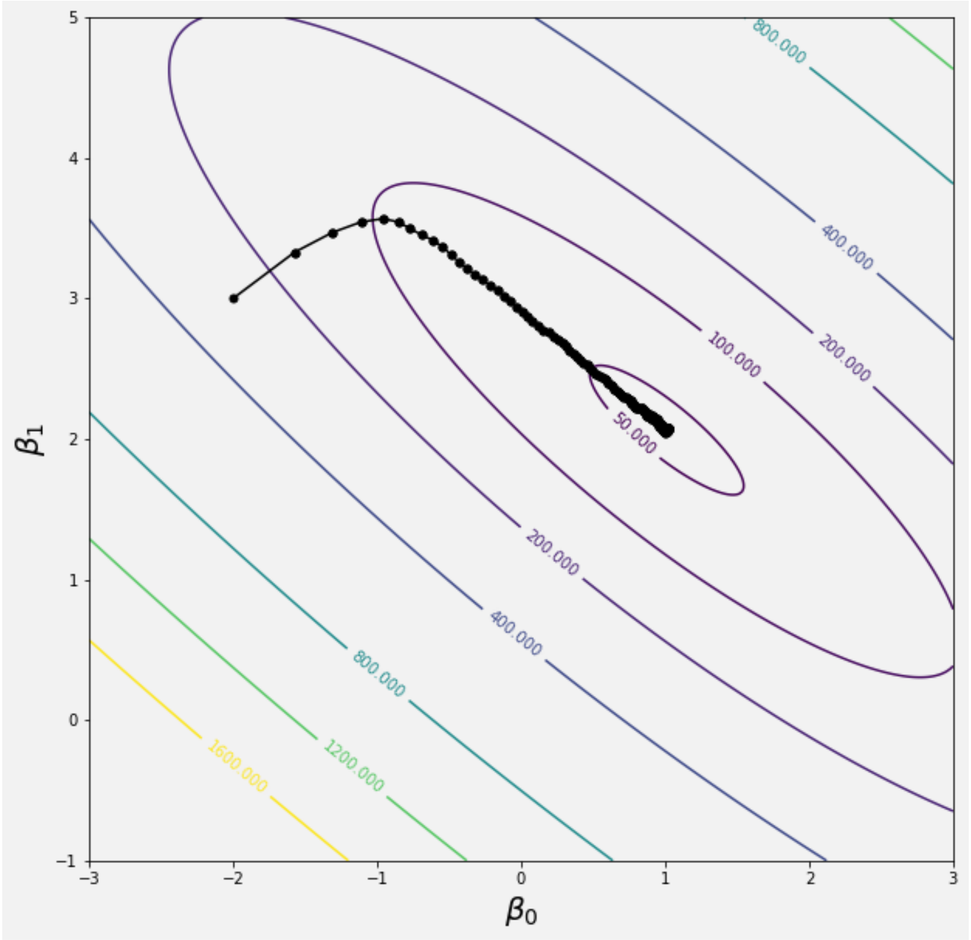
The Importance of the Learning Rate

- 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

$$L(\beta_0, \beta_1) = \Pi_{i=1}^n p(y_i \mid x_i; \beta_0, \beta_1)$$

$$= \Pi_{i=1}^n \text{sigm}(\beta_0 + \beta_1 x_i)^{y_i} (1 - \text{sigm}(\beta_0 + \beta_1 x_i))^{1-y_i}$$

Prob. of getting/observing my data, given my model and parameters.

SGD for Logistic Regression

$\log(ab) = \log a + \log b$

$\log a^b = b \log a$

Likelihood:

$$L(\beta_0, \beta_1) = \Pi_{i=1}^{n} p(y_i \mid x_i; \beta_0, \beta_1)$$

$$= \Pi_{i=1}^{n} \text{sigm}(\beta_0 + \beta_1 x_i)^{y_i} (1 - \text{sigm}(\beta_0 + \beta_1 x_i))^{1-y_i}$$

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

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

$$\log p(y|x) = \log \left(\text{sigm}(\beta_0+\beta_1 x)^y (1-\text{sigm}(\beta_0+\beta_1 x))^{1-y} \right)$$

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

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

$$NLL(\beta_0, \beta_1) = -\log \left(\Pi_{i=1}^{n} p(y_i \mid x_i; \beta_0, \beta_1)\right)$$

$$NLL(\beta_0, \beta_1) = -\sum_{i=1}^{n} y_i \log \text{sigm}(\beta_0 + \beta_1 x_i) + (1 - y_i) \log(1 - \text{sigm}(\beta_0 + \beta_1 x_i))$$

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

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

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

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

Repeat over epochs until stopping criterion.

SGD for Logistic Regression

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

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

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