# Solution Techniques for Linear and Logistic Regression

# Practicum and Final Exam

- The **Practicum** is due at 11:59pm on Wednesday December 13th.
- **Final Exam** Monday December 18th from 7:30-10pm in MUEN E050
  - Cumulative but will emphasize material since midterm
  - Bring a calculator
  - Two-page note-sheet. Handwritten. No magnifying glasses.
- **Review** in class on Wednesday. Come with questions!

# Previously 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

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon_i \qquad \text{where} \qquad \epsilon_i \sim N(0, \sigma^2)$$

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

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

by minimizing a similar functional.

# Finding Parameters in Linear Regression

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

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

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 i**terative 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

$$\beta_0^{(0)} = 0 \rightarrow \beta_0^{(1)} = 0.25 \rightarrow \beta_0^{(2)} \rightarrow \cdots$$

$$\beta_1^{(1)} = 0 \rightarrow \beta_1^{(1)} = 0.75 \rightarrow \beta_1^{(2)} \rightarrow \cdots$$

# A Silly 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?

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

minimize $z = 1$ minimum $f(1) = 1$

# A Silly 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?


CONCAVE UP
⇒ MINIMIZER
IS UNIQUE

# A Silly Example

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

OK, suppose that I guess that the minimizer is $z^{(0)} = 2.25$

**Question**: Which way should I move?

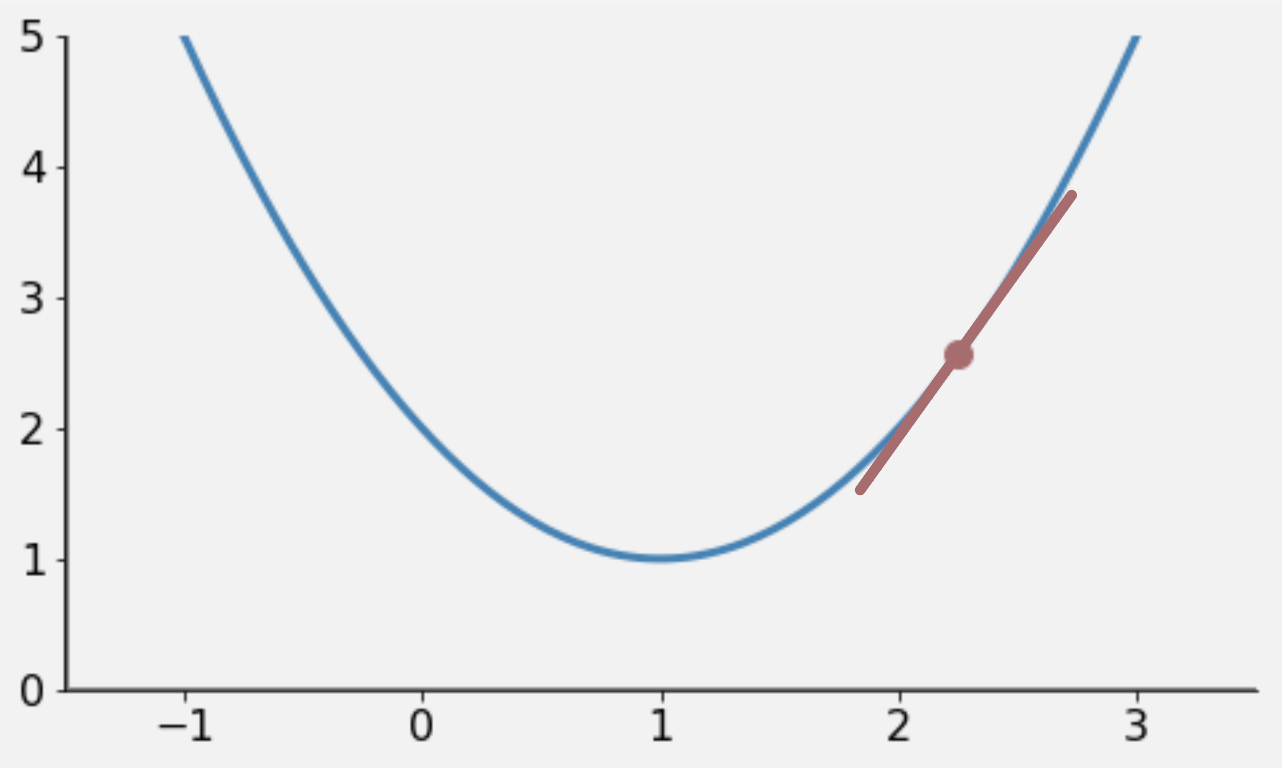
# A Silly Example

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

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

OK, 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?




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

# A Silly Example

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

OK, 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

# A Silly 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

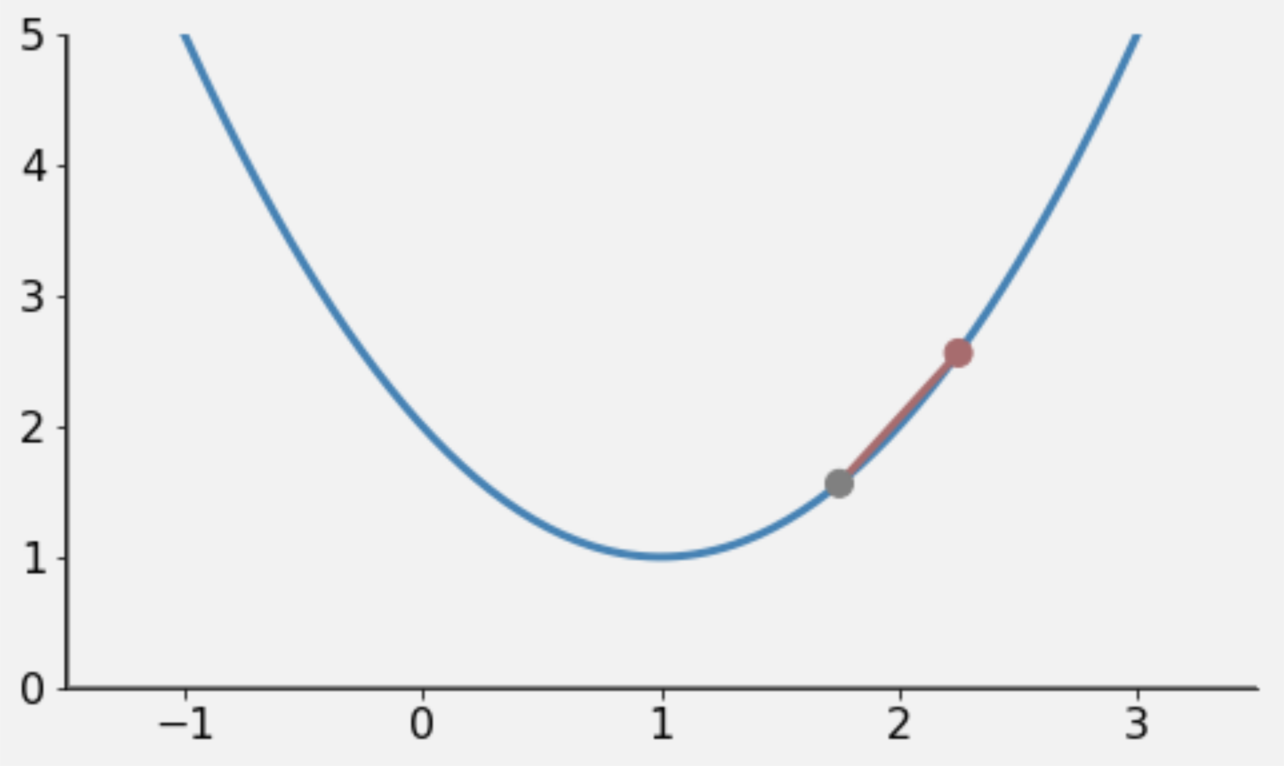
**Question**: How far should we move? **Answer**: Dunno, just pick a small step size like $\eta = 0.5$

# A Silly 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**: Dunno, just pick a small step size like $\eta = 0.5$

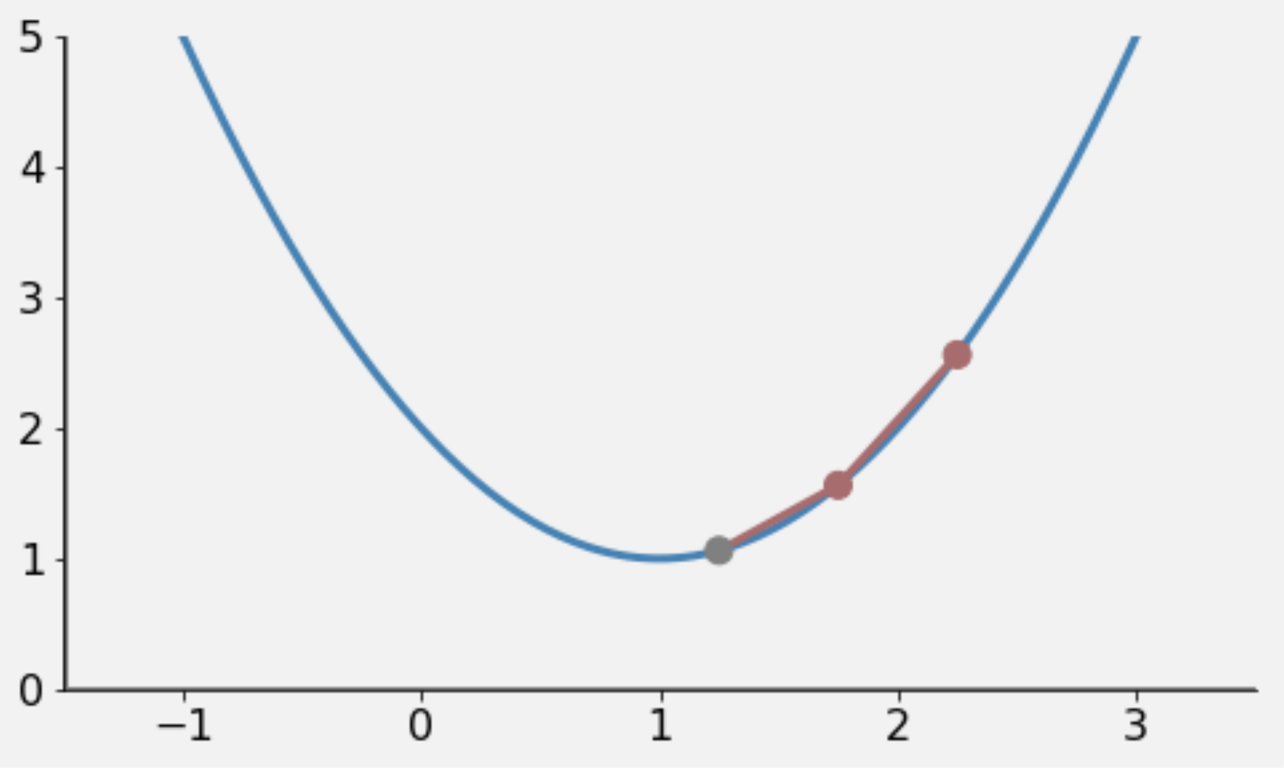


# A Silly Example

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**: Dunno, just pick a small step size like $\eta = 0.5$

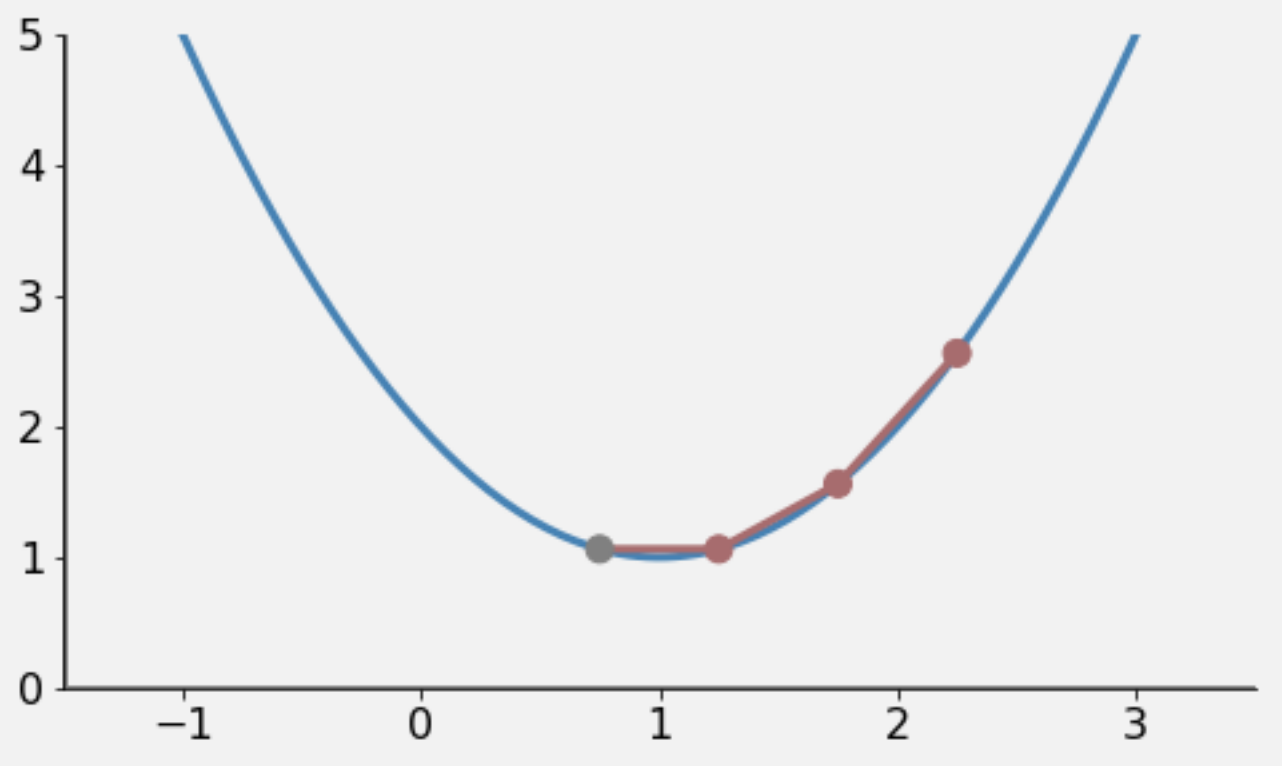


# A Silly Example

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**: Dunno, just pick a small step size like $\eta = 0.5$

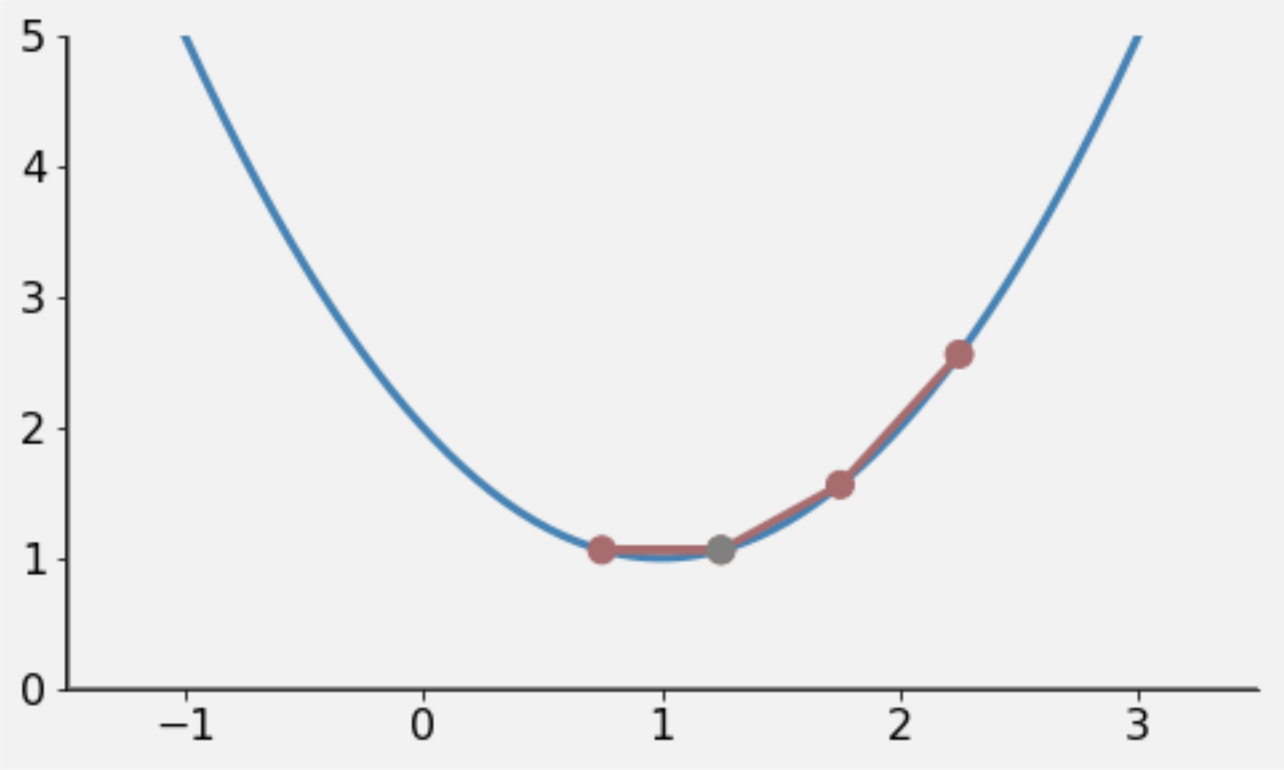


# A Silly Example

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

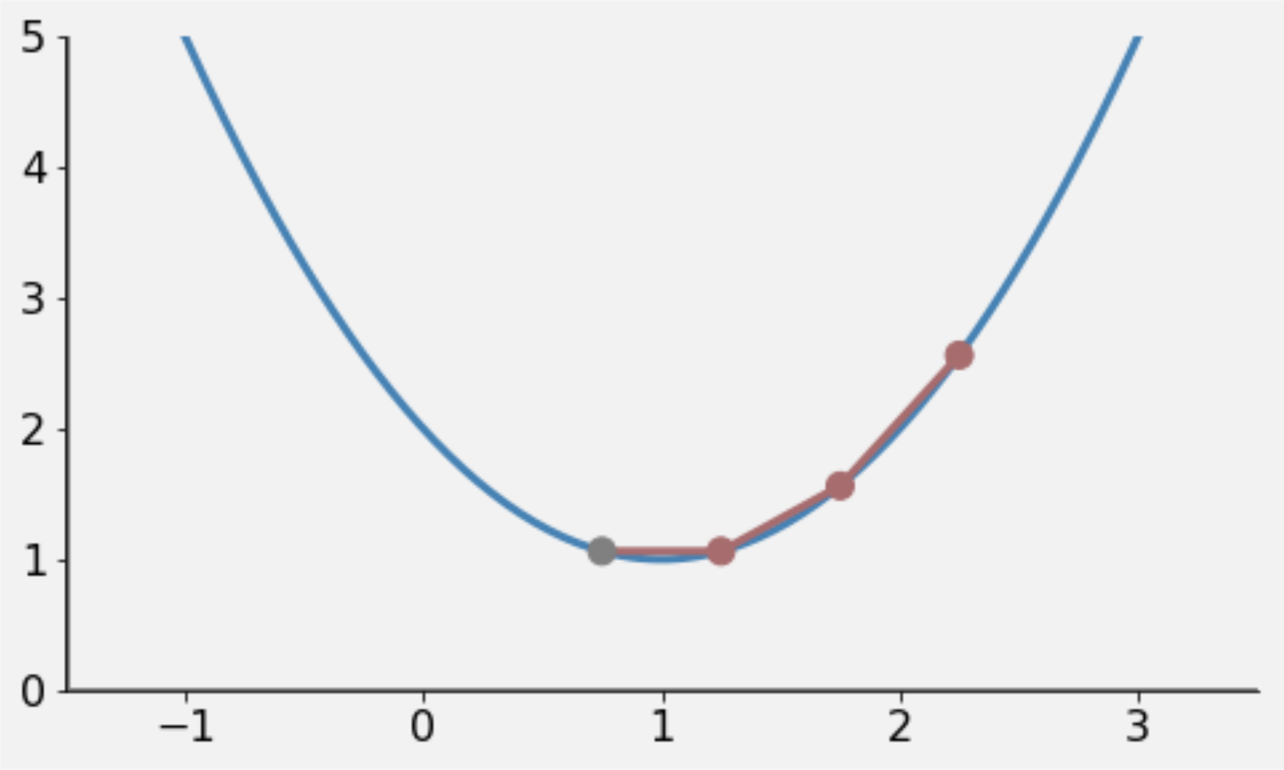
**Question**: How far should we move? **Answer**: Dunno, just pick a small step size like $\eta = 0.5$



# A Silly Example

**Question:** How do we fix this so we can get closer to the true minimum?

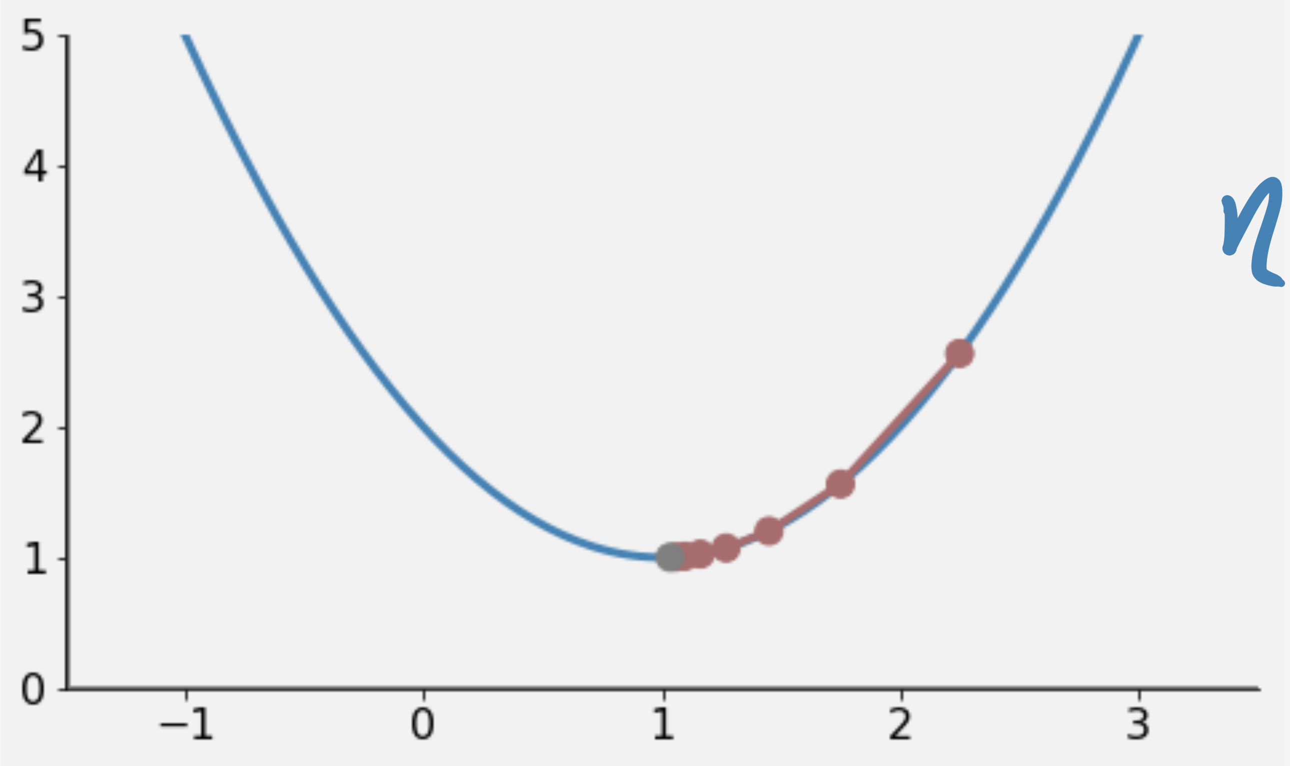
- DECREASE STEP SIZE
- CHOOSE TO MOVE, PROPORTIONAL to DERIVATIVE



# A Silly Example

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

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




$\eta = 0.2$

# Simple Linear Regression

OK, so how do we use the idea of Gradient Descent to estimate the parameters in SLR

Recall, the estimated parameters are the value of $\beta_0$ and $\beta_1$ that minimize the SSE

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

DATA - FIXED
#'s

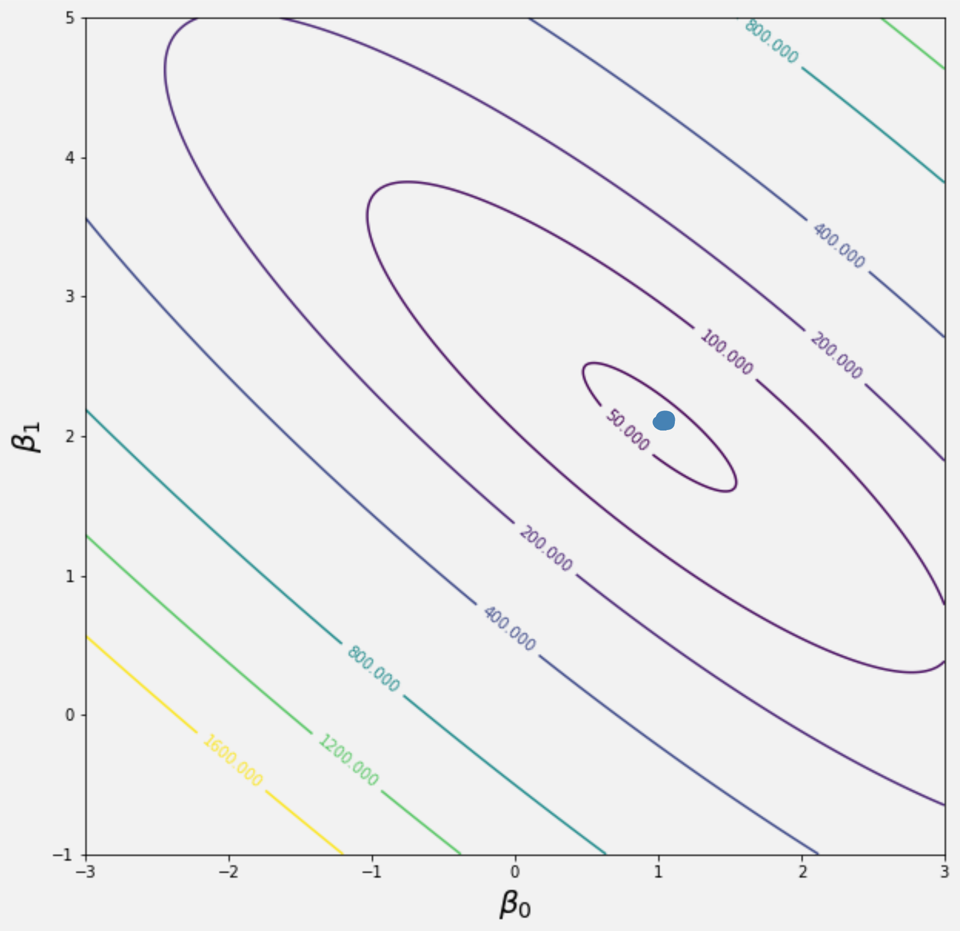
**Important**: We're minimizing over the $\beta$'s . The $x$'s and $y$'s are the values from the data.

The difficulty is that this is a function of two variables, which is not quite a simple parabola

# Simple Linear Regression

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

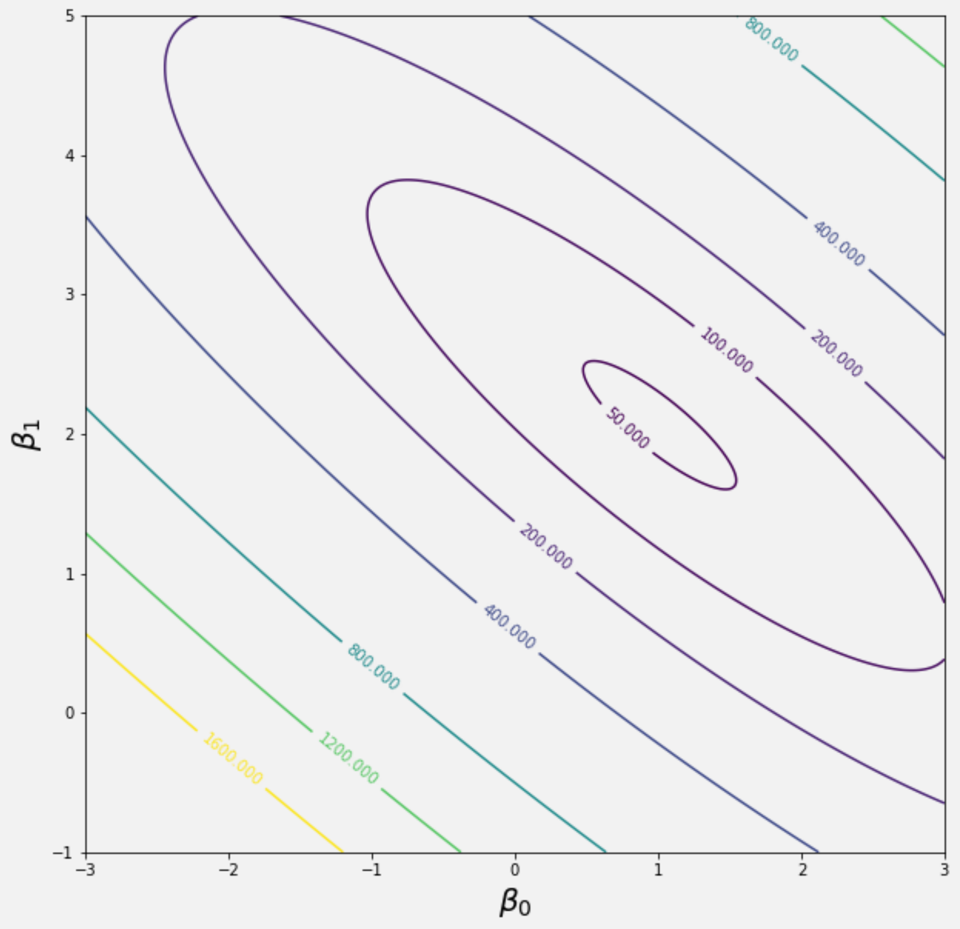
$$SSE = \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_i)]^2$$



# 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:

- Make a guess at the minimum
- Move iteratively downhill

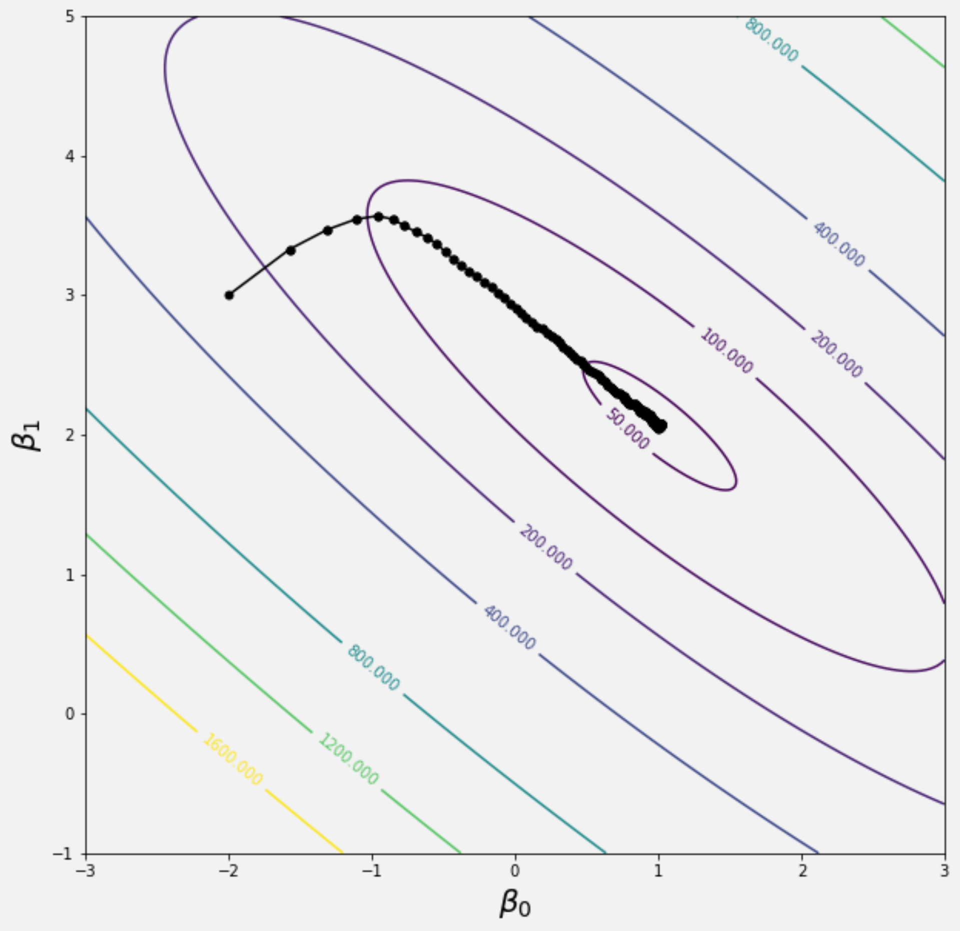
# Simple Linear Regression

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:

- Make a guess at the minimum
- Move iteratively downhill

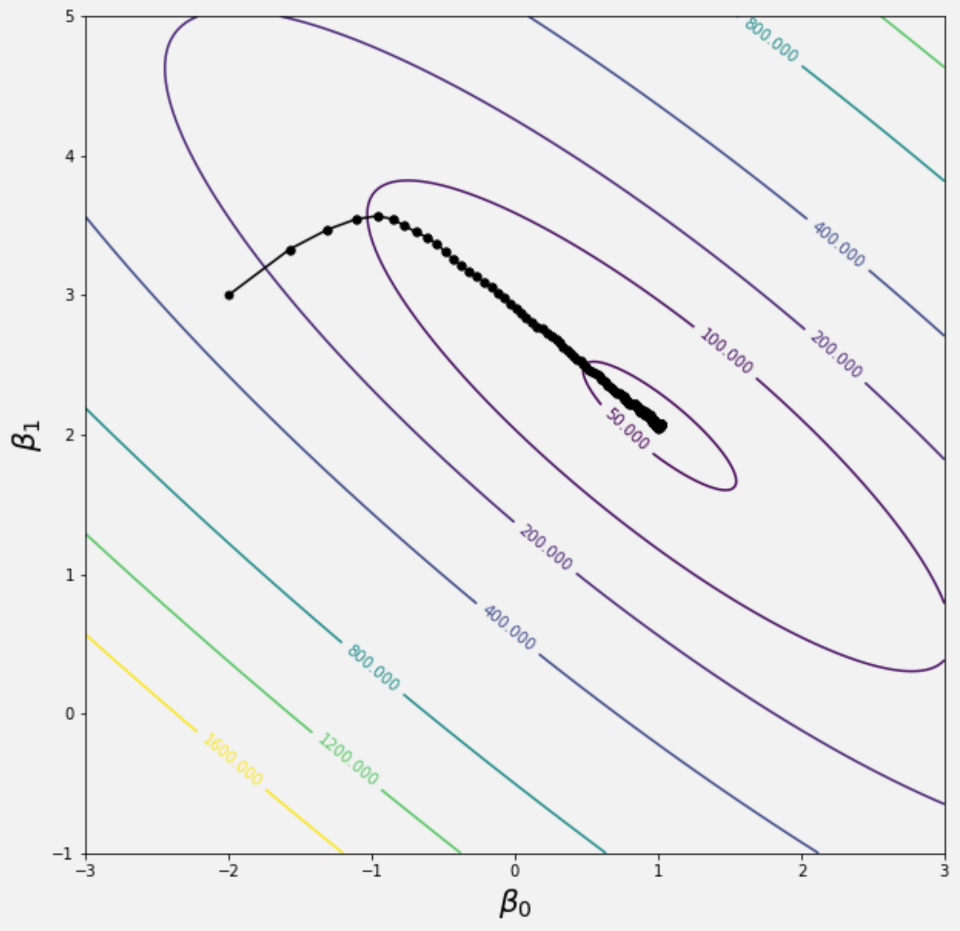


# Simple Linear Regression

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

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

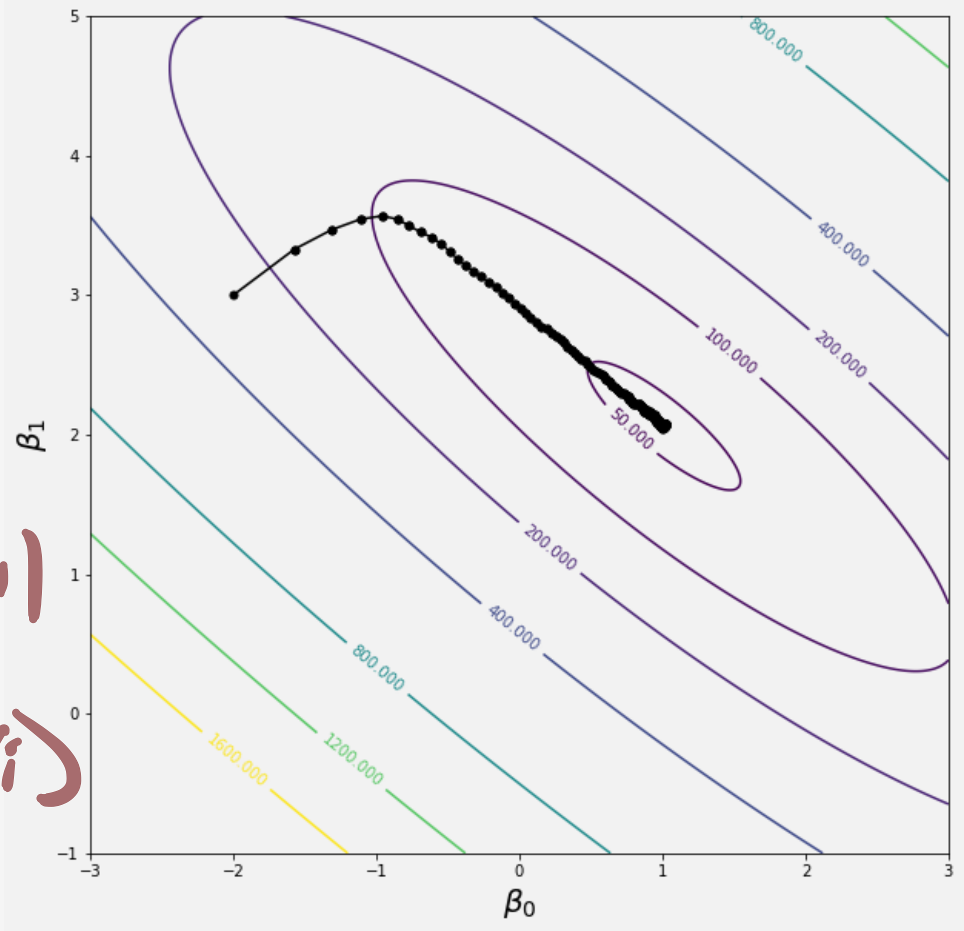
But how do we move downhill in 2d?



# 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?

- Move downhill in each coordinate direction
- Use partial derivatives

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

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

# 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:

$$\begin{aligned} \beta_0^{(k+1)} &= \beta_0^{(k)} - \eta \frac{\partial SSE}{\partial \beta_0^{(k)}} \\ \beta_1^{(k+1)} &= \beta_1^{(k)} - \eta \frac{\partial SSE}{\partial \beta_1^{(k)}} \end{aligned}$$

# 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:

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

# 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:

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

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

# 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

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

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

# 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 for SLR

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

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

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

# Stochastic Gradient Descent for SLR

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

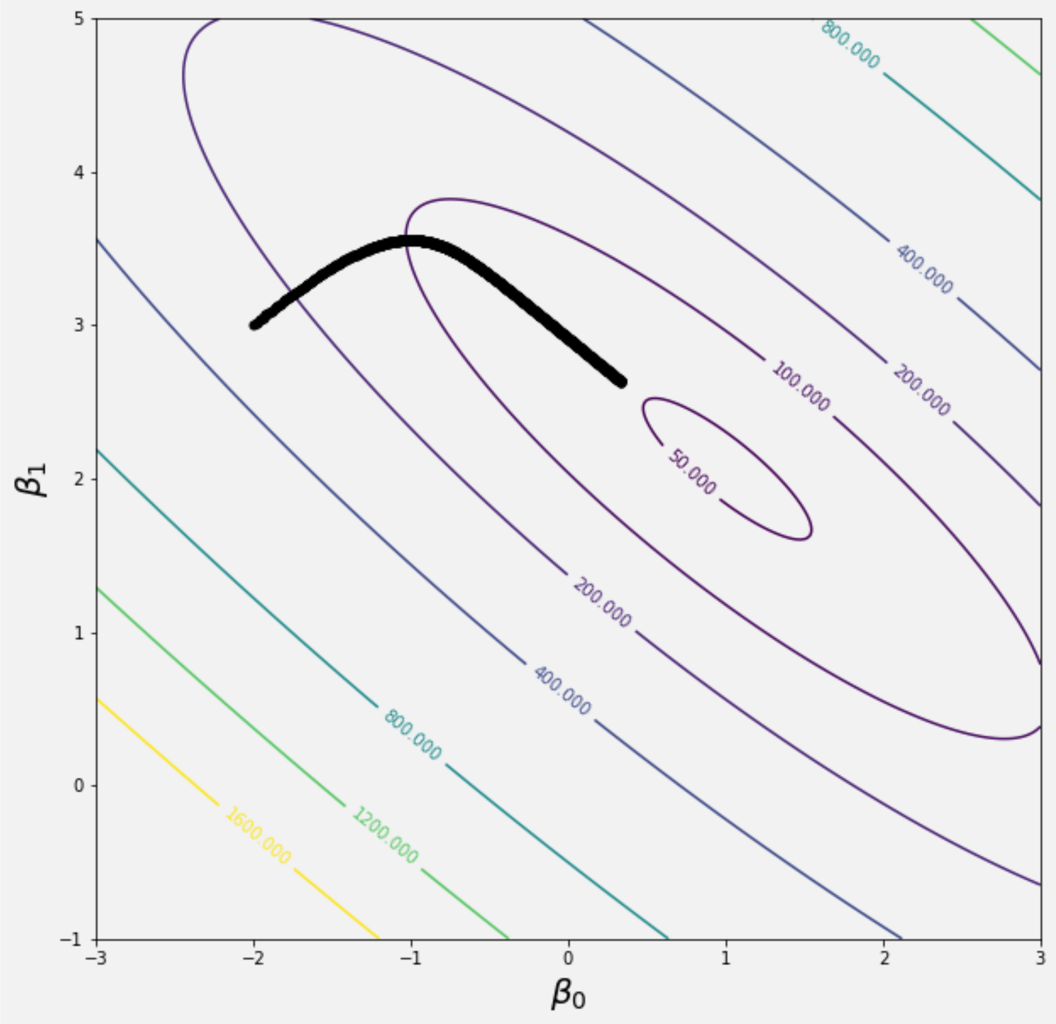
For each point $(x_i, y_i)$ in **RANDOMLY SHUFFLED** dataset, update parameters

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

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

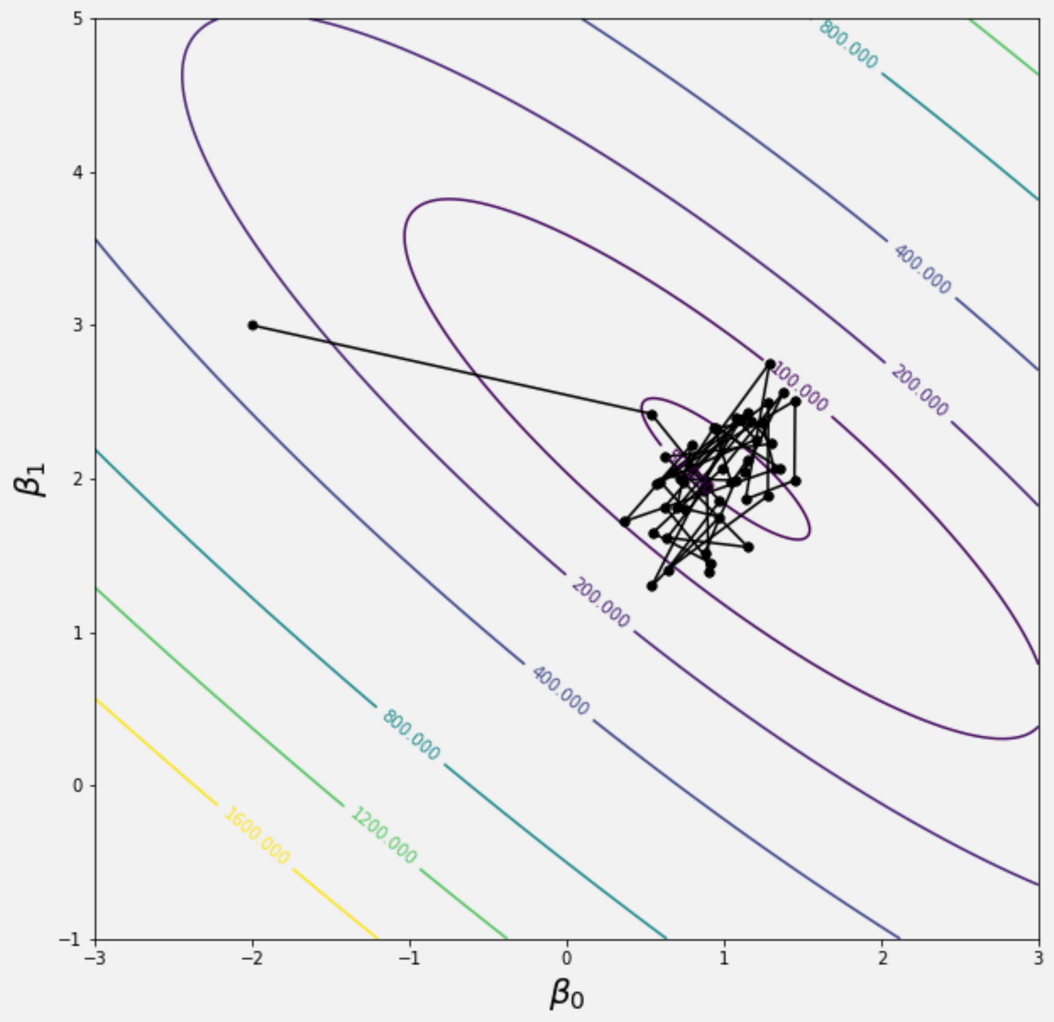
# The Importance of the Learning Rate

Too small of a learning rate and it takes forever to converge



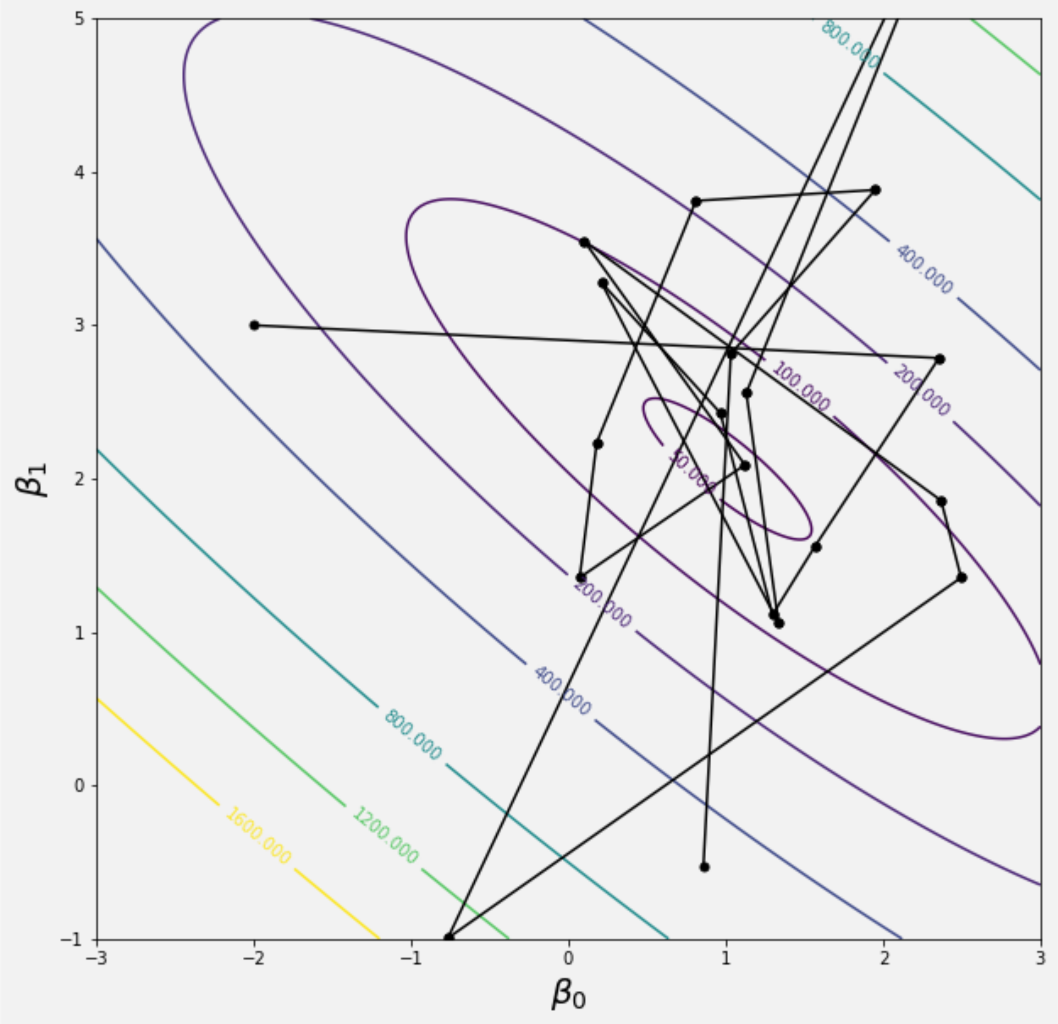
# The Importance of the Learning Rate

Too large a learning rate and you can get oscillations that bounce around



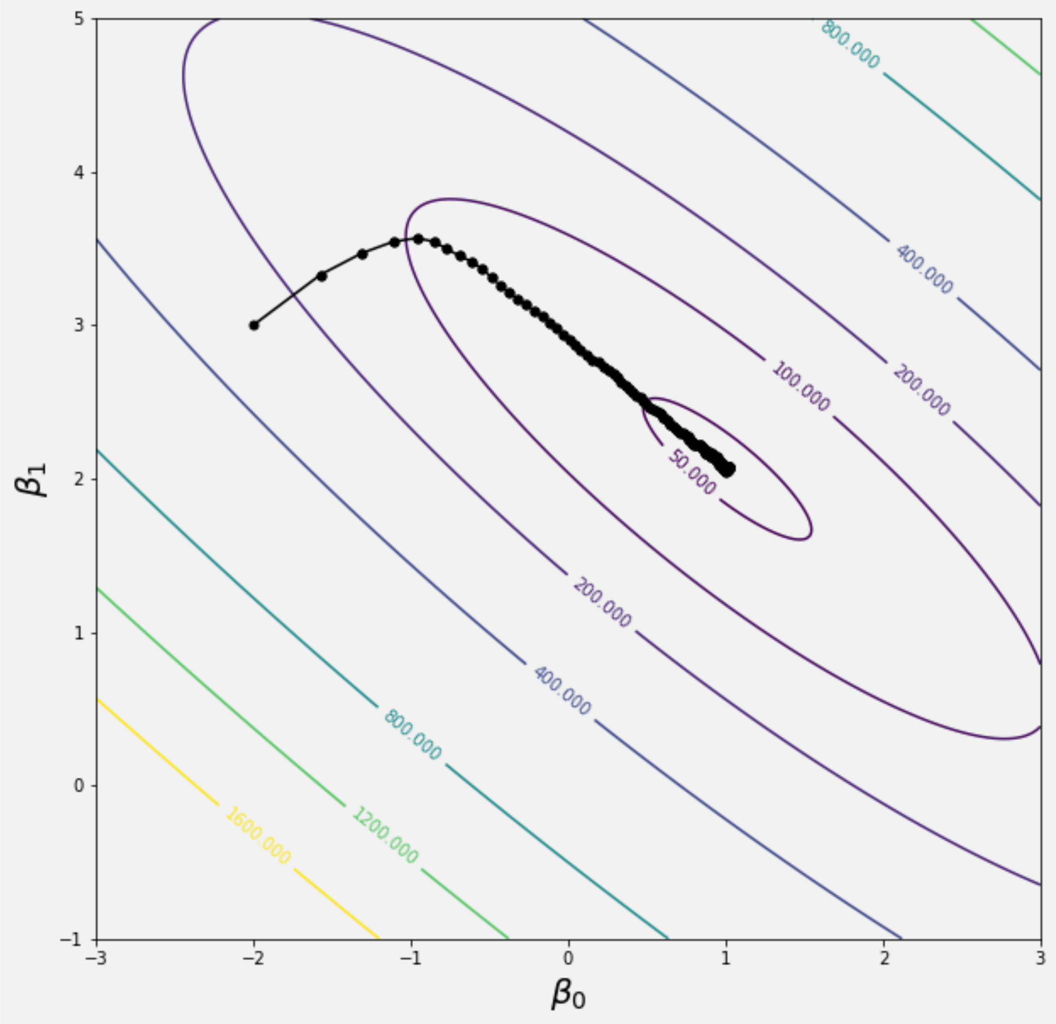
# The Importance of the Learning Rate

Way too large a learning rate and you can diverge



# The Importance of the Learning Rate

Generally have to tune learning rate to get it just right



# SGD for Logistic Regression

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

Our model was $p = p(y=1 \mid x) = \mathrm{sigm}(\beta_0 + \beta_1 x) \;\; \Rightarrow p(y=0 \mid x) = 1 - \mathrm{sigm}(\beta_0 + \beta_1 x)$

This can be written more compactly as

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

Notice that this looks like a Bernoulli random variable with mean $\mathrm{sigm}(\beta_0 + \beta_1 x)$

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

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

# SGD for Logistic Regression

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

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

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

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

# 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!)

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

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

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

# SGD for Logistic Regression

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

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

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

# OK! Let's Go to Work!

Get in groups, get out laptop, and open the Lecture 26 In-Class Notebook

**Let's**:

- Actually implement SGD for simple linear regression
- Actually implement SGD for logistic regression

# Faculty Course Questionnaires

Available at the following link:

http://colorado.campuslabs.com/courseeval

- Open now until December 15th at 11:59pm