Stochastic Gradient Descent Part II

Logistic Regression

Previously on CSCI 4622

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 $\mathrm{RSS}_\lambda = \sum_{i=1}^{n} \left[(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) - y_i \right]^2 + \lambda \sum_{k=1}^{p} \beta_k^2$

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

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

$y_i = \{0, 1\}$

Today we'll see that we can learn the parameters of LogReg by minimizing a loss function

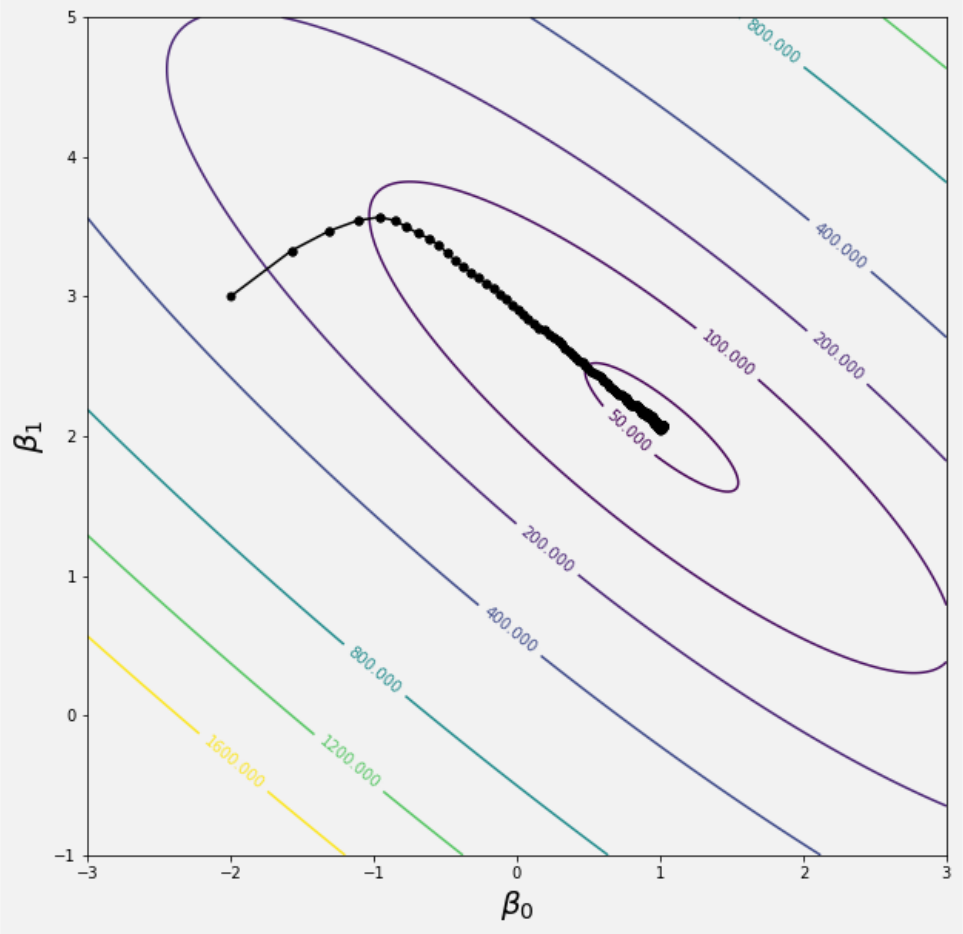
Simple Linear Regression

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

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

In 2D the process is still the same:

- Make a guess at the minimum
- Move iteratively downhill



Vectorizing Gradient Descent

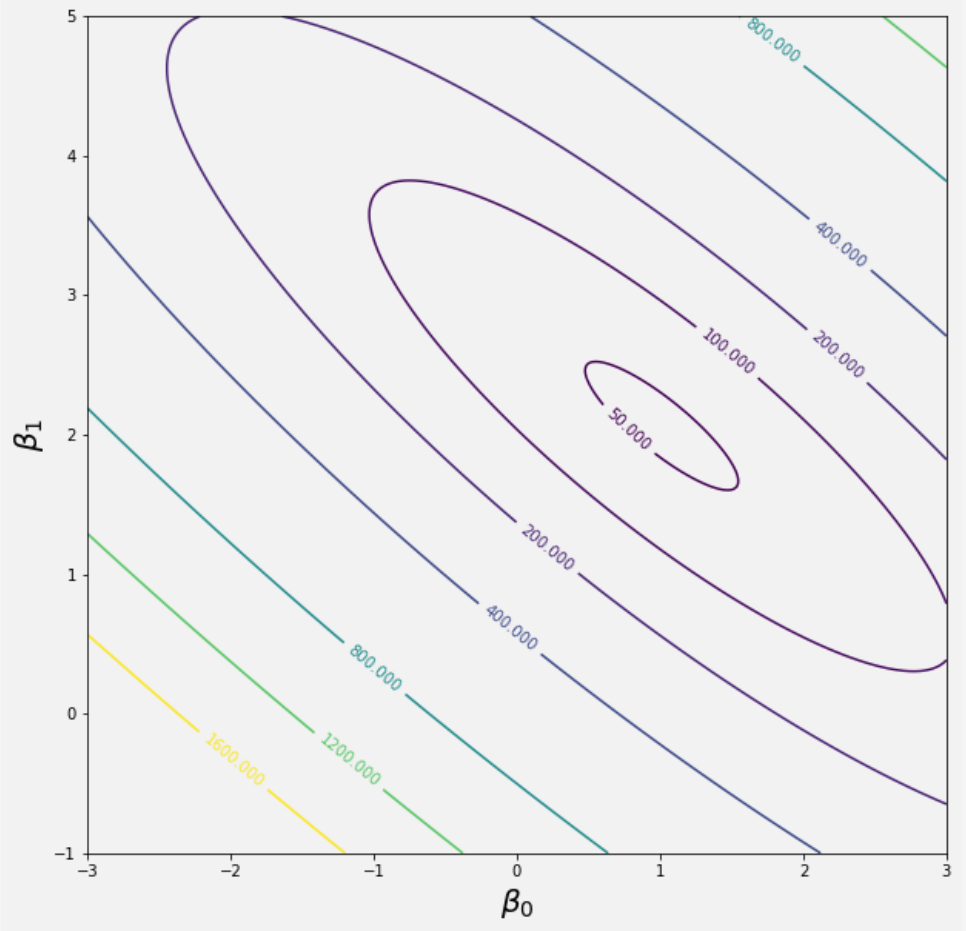
Let $\boldsymbol{\beta} = [\beta_0, \beta_1]^T$ be vector of the parameters. Need to **vectorize** derivatives too

$$\nabla \mathrm{RSS} = \begin{bmatrix} \sum_{i=1}^{n} 2\left[(\beta_0 + \beta_1 x_i) - y_i\right] \\ \\ \sum_{i=1}^{n} 2\left[(\beta_0 + \beta_1 x_i) - y_i\right] x_i \end{bmatrix}$$

$$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \eta \nabla \mathrm{RSS}$$

Vector of derivatives is called the **Gradient**

Points in direction that loss function increases the **fastest** (i.e. steepest direction)



Stochastic Gradient Descent

Update based on one training example at a time. Faster, but less accurate updates

$$\begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} \leftarrow \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} - \eta \begin{bmatrix} 2((\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) - y_i) \\ 2((\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) - y_i) x_{i1} \\ \vdots \\ 2((\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) - y_i) x_{ip} \end{bmatrix}$$

Important: Just like when training a Perceptron:

- Loop through training examples in random order to avoid bias
- Go through all training examples before starting over

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:

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

This can be written more compactly as

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

$p(y=0|x) = \text{sigm}(\beta_0+\beta_1 x)^0 (1-\text{sigm}(\beta_0+\beta_1 x))^{1-0}$

1

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

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

This can be written more compactly as

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

So, how do we estimate the parameters? We don't have a Loss function to minimize ...

$p(H) = p$

Maximum Likelihood Estimation

- An alternate method for estimating model parameters is to create a likelihood function involving the model parameters and the data, and choose the value of the parameter that maximizes it

- You've probably done this before, but haven't called it Maximum Likelihood Estimation

Example: Suppose you have a biased coin with an unknown probability p of landing heads, you flip it 6 times and it comes up HHHHHT. Estimate the parameter p for the coin.

$L(p) = p(HHHHHT \mid p)$ ← LIKELIHOOD FUNCTION

$= p(H \mid p)^5 p(T \mid p)^1 = p^5(1-p)^1$

TAKE $\log L(p) = \log(p^5) + \log(1-p) = 5\log p + \log(1-p)$

Maximum Likelihood Estimation

- An alternate method for estimating model parameters is to create a likelihood function involving the model parameters and the data, and choose the value of the parameter that maximizes it

- You've probably done this before, but haven't called it Maximum Likelihood Estimation

Example: Suppose you have a biased coin with an unknown probability p of landing heads, you flip it 6 times and it comes up HHHHHT. Estimate the parameter p for the coin.

$$LL(p) = 5\log p + \log(1-p) \quad \left(\text{TAKE } \frac{\partial}{\partial p} = 0\right)$$

$$\frac{dLL}{dp} = \frac{5}{p} + \frac{-1}{1-p} = 0 \Rightarrow 5(1-p) - p = 0$$

$$\Rightarrow 5 - 5p - p = 0 \Rightarrow 5 = 6p \Rightarrow \hat{p} = 5/6$$

Maximum Likelihood Estimation

Maximum Likelihood Estimation asks: *What are the parameters that best explain the observed data?*

In practice we compute the MLE in three steps:

1. Write down the probability of observing the data, given the probability distribution and the parameter(s) of interest.

2. Take the negative log to get the Negative Log-Likelihood (NLL)

3. Minimize the NLL to get the optimal value of the parameters given the data

MLE for Logistic Regression

The distribution for Logistic Regression looks just like a coin flip

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

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

$$L(\beta_0, \beta_1) \quad = \quad p(y_1, y_2, \ldots, y_n \mid x_1, x_2, \ldots, x_n, \beta_0, \beta_1)$$

MLE for Logistic Regression

The distribution for Logistic Regression looks just like a coin flip

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

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

$$\begin{aligned} L(\beta_0, \beta_1) &= p(y_1, y_2, \ldots, y_n \mid x_1, x_2, \ldots, x_n, \beta_0, \beta_1) \\ &= \prod_{i=1}^n p(y_i \mid x_i, \beta_0, \beta_1) \\ &= \prod_{i=1}^n \text{sigm}(\beta_0 + \beta_1 x_i)^{y_i} (1 - \text{sigm}(\beta_0 + \beta_1 x_i))^{1-y_i} \end{aligned}$$

MLE for Logistic Regression

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

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 get the so-called Negative Log-Likelihood

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

MLE for Logistic Regression

For Logistic Regression, this Negative Log-Likelihood is our **Loss function**

We minimize it to find the optimal parameters

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

Do Stochastic Gradient Descent same way as before. Just need to compute new gradient

SGD for Logistic Regression

For Logistic Regression, this Negative Log-Likelihood is our **Loss function**

We minimize it to find the optimal parameters

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

Do Stochastic Gradient Descent same way as before. Just need to compute new gradient

It turns out:

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

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

Mathematical Details

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

Simpler notation: Let $s(z) = \text{sigm}(z) = \frac{1}{1 + e^{-z}}$ and $\text{sigm}(\beta_0 + \beta_1 x_i) = s(\boldsymbol{\beta}^T \mathbf{x}_i)$

Fact: $s'(z) = s(z)(1 - s(z))$

$$\frac{d}{dz}(1+e^{-z})^{-1} = -1(1+e^{-z})^{-2}(+e^{-z})$$

$$= \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}} = s(z) \cdot \left(\frac{(1+e^{-z})-1}{1+e^{-z}}\right)$$

$$= s(z)(1 - s(z))$$

Mathematical Details

$$NLL(\beta_0, \beta_1) \quad = \quad -\sum_{i=1}^{n} \Big\{ y_i \;\; \log s(\boldsymbol{\beta}^T \mathbf{x}_i) + (1 - y_i) \log(1 - s(\boldsymbol{\beta}^T \mathbf{x}_i)) \Big\}$$

Simpler notation: Let $s(z) = \text{sigm}(z) = \dfrac{1}{1 + e^{-z}}$ and $\text{sigm}(\beta_0 + \beta_1 x_i) = s(\boldsymbol{\beta}^T \mathbf{x}_i)$

Fact: $s'(z) = s(z)(1 - s(z))$

Proof:

Mathematical Details

$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots$

$$\frac{\partial NLL}{\partial \beta_k} = -\sum_{i=1}^{n} \left\{ y_i \frac{\partial}{\partial \beta_k} \log s(\beta^T \mathbf{x}_i) + (1 - y_i) \frac{\partial}{\partial \beta_k} \log(1 - s(\beta^T \mathbf{x}_i)) \right\}$$

Let's do one term in the derivative at a time

$$\frac{\partial}{\partial \beta_k} \log s(\beta^T \mathbf{x}_i) = \frac{1}{s(\beta^T x_i)} \cdot \frac{\partial}{\partial \beta_k} s(\beta^T x_i) =$$

$$= x_{ik}$$

$$\frac{1}{s(\beta^T x_i)} \; s(\beta^T x_i)(1 - s(\beta^T x_i)) \frac{\partial}{\partial \beta_k} (\beta^T x_i)$$

$$= [1 - s(\beta^T x_i)] x_{ik}$$

Mathematical Details

$$\frac{\partial NLL}{\partial \beta_k} = -\sum_{i=1}^{n} \left\{ y_i \left[1 - s(\boldsymbol{\beta}^T \mathbf{x}_i)\right] x_{ik} + (1 - y_i) \frac{\partial}{\partial \beta_k} \log(1 - s(\boldsymbol{\beta}^T \mathbf{x}_i)) \right\}$$

Let's do one term in the derivative at a time

$$\frac{\partial}{\partial \beta_k} \log(1 - s(\boldsymbol{\beta}^T \mathbf{x}_i)) =$$

$s(\beta^T x_i) = s(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p)$

$$= \frac{1}{1 - s(\beta^T x_i)} \cdot \frac{\partial}{\partial \beta_k} \left(1 - s(\beta^T x_i)\right)$$

$$= \frac{1}{1 - s(\beta^T x_i)} \cdot -s(\beta^T x_i)\left(1 - s(\beta^T x_i)\right) x_{ik}$$

$$= -s(\beta^T x_i)\, x_{ik}$$

Mathematical Details

$$\frac{\partial NLL}{\partial \beta_k} = -\sum_{i=1}^{n} \left\{ y_i \left[1 - s(\boldsymbol{\beta}^T \mathbf{x}_i)\right] x_{ik} - (1 - y_i) s(\boldsymbol{\beta}^T \mathbf{x}_i) x_{ik} \right\}$$

And simplify, and simplify, and simplify, and simplify ...

$$-\sum_{i=1} \{ y_i [1 - s(\beta^T x_i)] - (1 - y_i) s(\beta^T x_i) \} x_{ik}$$

$$-y_i s(\ )$$

$$-\sum_i \{ y_i - \text{sigm}(\beta^T x_i) \} x_{ik}$$

$$\Rightarrow \sum_i [\text{sigm}(\beta^T x_i) - y_i] x_{ik}$$

Mathematical Details

$$\frac{\partial NLL}{\partial \beta_k} = -\sum_{i=1}^{n} \left\{ y_i \left[1 - s(\boldsymbol{\beta}^T \mathbf{x}_i) \right] x_{ik} - (1 - y_i) s(\boldsymbol{\beta}^T \mathbf{x}_i) x_{ik} \right\}$$

And simplify, and simplify, and simplify, and simplify ...

Mathematical Details

And finally we arrive at the Gradient Descent Update

$$\beta_k \leftarrow \beta_k - \eta \frac{\partial NLL(\boldsymbol{\beta})}{\partial \beta_k} = \beta_k - \eta \sum_{i=1}^{n} \left[\text{sigm}(\boldsymbol{\beta}^T \mathbf{x}_i) - y_i \right] x_{ik} \text{ for } k = 0, \ldots, p$$

Which we immediately turn into a Stochastic Gradient Descent Update

For (x_i, y_i)

$$\beta_k \leftarrow \beta_k - \eta \left[\text{sigm}(\boldsymbol{\beta}^T \mathbf{x}_i) - y_i \right] x_{ik} \text{ for } k = 0, \ldots, p$$

Just like before:

- Loop through training examples in random order to avoid bias
- Go through all training examples before starting over

Geometry and Unique Solutions

The RSS Loss function for regression was a bowl, so we had exactly one minimizing solution

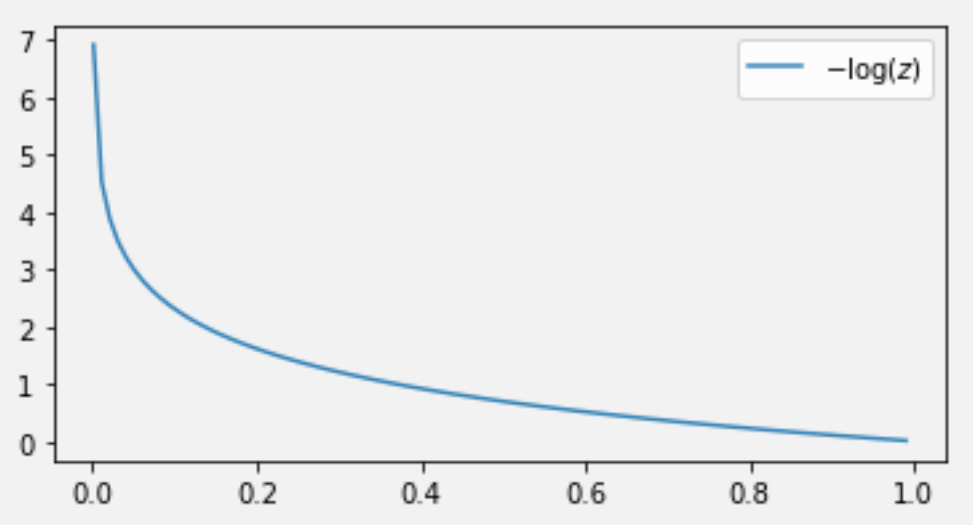
$$NLL(\beta_0, \beta_1) \quad = \quad -\sum_{i=1}^{n} \left\{ y_i \ \log s(\boldsymbol{\beta}^T \mathbf{x}_i) + (1 - y_i) \log(1 - s(\boldsymbol{\beta}^T \mathbf{x}_i)) \right\}$$

$-\log(z_i) \quad -\log(z_j)$

$-\log(1-z_k)$

$-\log(1-z_m)$

Do we have that for Logistic Regression as well?

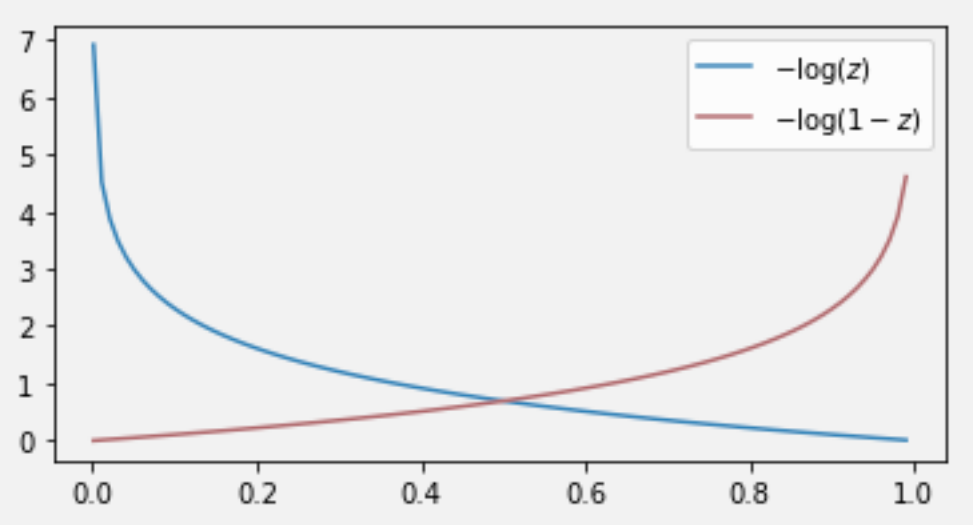


Geometry and Unique Solutions

The RSS Loss function for regression was a bowl, so we had exactly one minimizing solution

$$NLL(\beta_0, \beta_1) \quad = \quad -\sum_{i=1}^{n} \left\{ y_i \ \log s(\boldsymbol{\beta}^T \mathbf{x}_i) + (1 - y_i) \log(1 - s(\boldsymbol{\beta}^T \mathbf{x}_i)) \right\}$$

Do we have that for Logistic Regression as well?

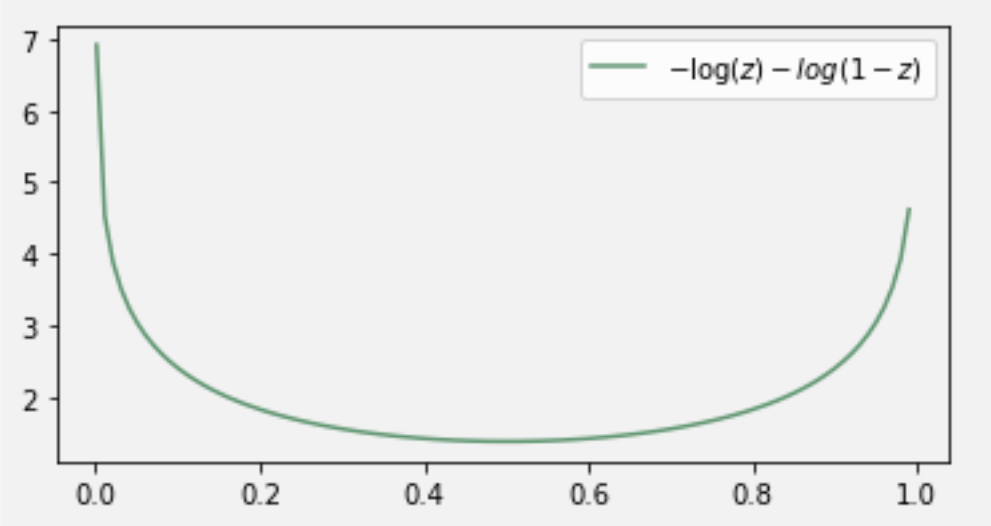


Geometry and Unique Solutions

The RSS Loss function for regression was a bowl, so we had exactly one minimizing solution

$$NLL(\beta_0, \beta_1) \quad = \quad -\sum_{i=1}^{n} \left\{ y_i \ \log s(\boldsymbol{\beta}^T \mathbf{x}_i) + (1 - y_i) \log(1 - s(\boldsymbol{\beta}^T \mathbf{x}_i)) \right\}$$

Do we have that for Logistic Regression as well?

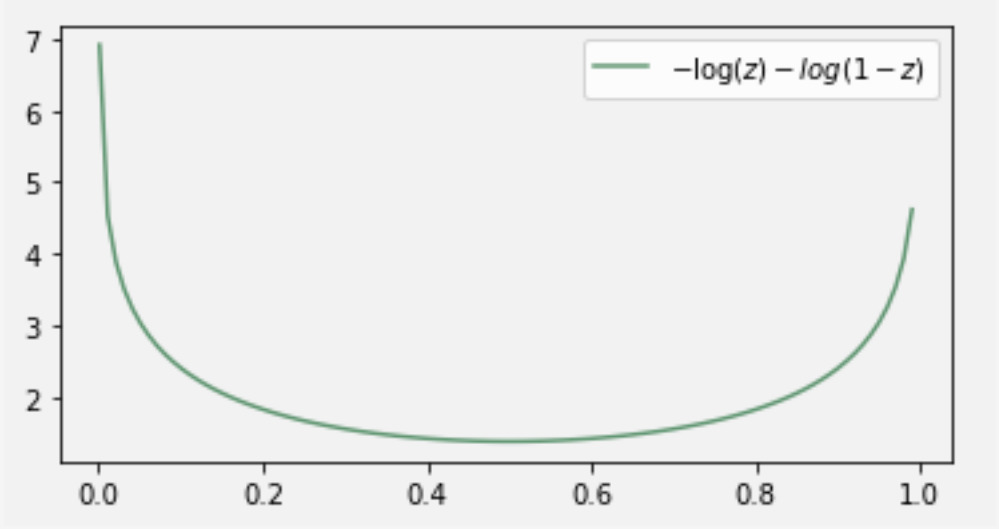


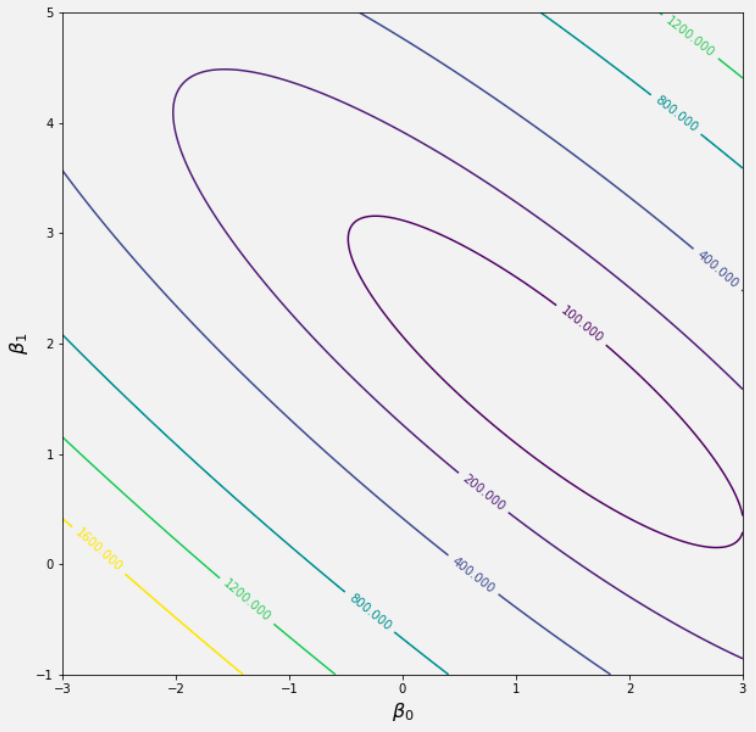
Geometry and Unique Solutions

The RSS Loss function for regression was a bowl, so we had exactly one minimizing solution

$$NLL(\beta_0, \beta_1) \quad = \quad -\sum_{i=1}^{n} \left\{ y_i \ \log s(\boldsymbol{\beta}^T \mathbf{x}_i) + (1 - y_i) \log(1 - s(\boldsymbol{\beta}^T \mathbf{x}_i)) \right\}$$

Do we have that for Logistic Regression as well?





Logistic Regression and Regularization

What if we want to add regularization?

$$NLL(\beta_0, \beta_1) \quad = \quad -\sum_{i=1}^{n} \Big\{ y_i \;\; \log s(\boldsymbol{\beta}^T \mathbf{x}_i) + (1 - y_i) \log(1 - s(\boldsymbol{\beta}^T \mathbf{x}_i)) \Big\}$$

We've got a loss function now, just add regularization!

$$NLL_\lambda \quad = \quad -\sum_{i=1}^{n} \Big\{ y_i \;\; \log s(\boldsymbol{\beta}^T \mathbf{x}_i) + (1 - y_i) \log(1 - s(\boldsymbol{\beta}^T \mathbf{x}_i)) \Big\} + \lambda \sum_{k=1}^{p} \beta_k^2$$

SGD update becomes

$$\beta_0 \leftarrow \beta_0 - \eta \Big[\text{sigm}(\boldsymbol{\beta}^T \mathbf{x}_i) - y_i \Big]$$

$$\beta_k \leftarrow \beta_k - \eta \Big(\Big[\text{sigm}(\boldsymbol{\beta}^T \mathbf{x}_i) - y_i \Big] \, x_{ik} + 2\lambda \beta_k \Big) \;\; \text{for } k = 1, \ldots, p$$

SGD Part II Wrap-Up

- Can go from probability model to Loss function with Maximum Likelihood Est
- SGD is SGD for the most part.
- For a new Loss function, just have to figure out the derivatives

Next Time:

- Hands-On Stochastic Gradient Descent
- Talk about good implementation practices. Actually code the algorithm.

If-Time Bonus: Merging Loss and MLE

Why do we have this ready-made loss function in regression

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

How come we have to use MLE to get a loss function for Logistic Regression?

Remember the assumptions of simple linear regression:

FIXED

$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \qquad \epsilon_i \sim N(0, \sigma^2)$

$\Rightarrow \quad p(y_i \mid \beta_0 + \beta_1 x_i) \sim N(\beta_0 + \beta_1 x_i, \sigma^2)$

$$L(\beta_0, \beta_1) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(\beta_0 + \beta_1 x_i - y_i)^2}{2\sigma^2} \right]$$

$$NLL = -\sum_{i=1}^{n} \left\{ \log\left(\frac{1}{\sqrt{2\pi}} \right) - \frac{(\beta_0 + \beta_1 x_i - y_i)^2}{2\sigma^2} \right\}$$

$$= -n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) + \sum_{n} \frac{(\beta_0 + \beta_1 x_i - y_i)^2}{2\sigma^2}$$

$$= \frac{1}{2\sigma^2} \sum_{i=1} (\beta_0 + \beta_1 x_i - y_i)^2 \propto RSS(\beta_0, \beta_1)$$