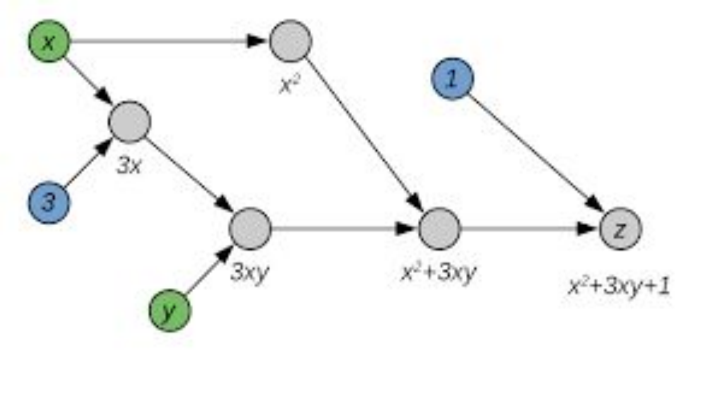
Building ML Models

Week 4 – Linear Regression

Recap From Work So Far

The Automatic Differentiation Library

- **Concept**
  - **Breaking math expressions down recursively**
  - Atomic level = variables, constants
- **Evaluation**
  - Single call from function
  - Input is {‘varname’ : value}
- **Differentiation**
  - Single call, now gradient
  - Done **with respect** to vars



Purpose

ML often relies on gradient descent performed on loss functions (find minimum error)... very complex.

Automatic differentiation lets us determine gradients with a *single function call.*

Let’s apply it to our first model: Multivariable Linear Regression

Multivariable Linear Regression

Linear Regression Basics

$$\hat{Y} = mx + b$$

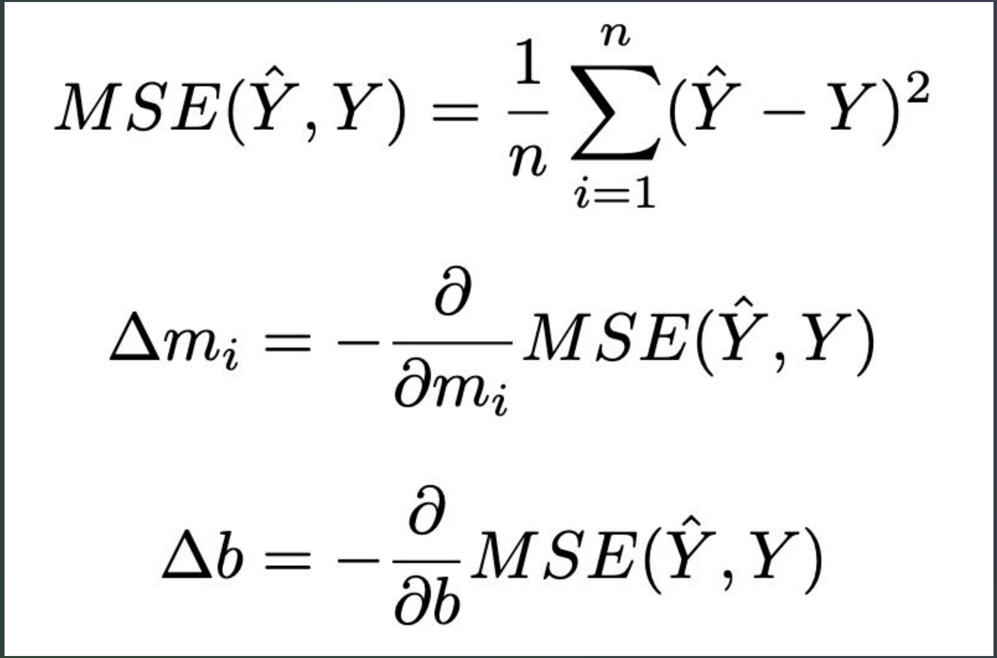
- Single-variable LR
  - **Change in Y as a function of change in x**
  - Y-intercept offsets
- Predict: **what (m, b) minimizes incorrectness of Y?**

More Dimensions

$$\hat{Y} = m_1x_1 + m_2x_2 + ... + b$$

- More predictors?
  - **New weight for each x, called a “feature”**
  - Still a single y-intercept

Applying Gradient Descent

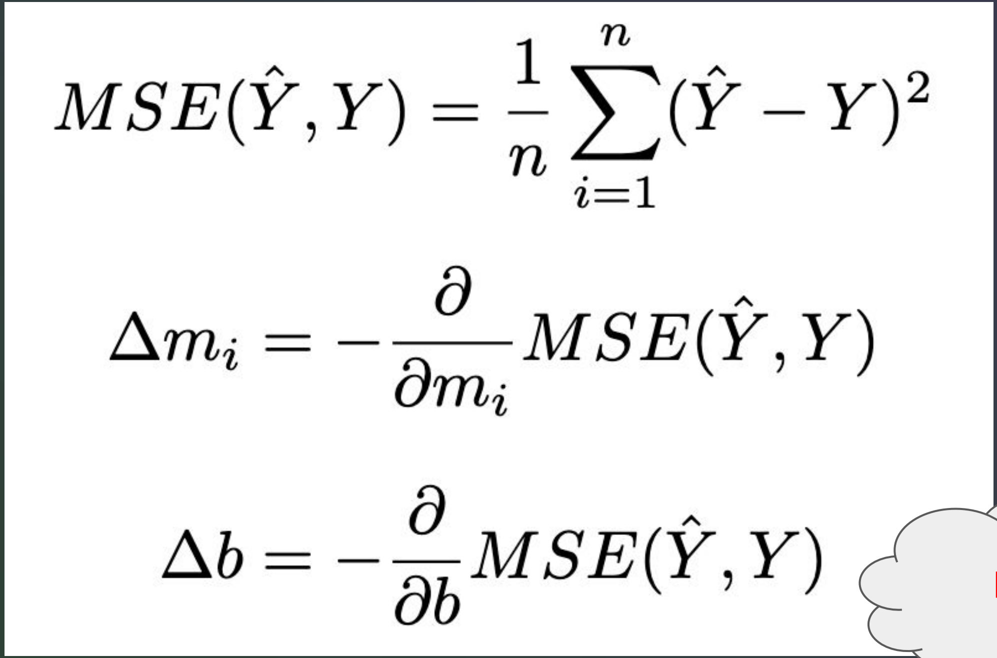


$$MSE(\hat{Y}, Y) = \frac{1}{n} \sum_{i=1}^{n} (\hat{Y} - Y)^2$$

$$\Delta m_i = -\frac{\partial}{\partial m_i} MSE(\hat{Y}, Y)$$

$$\Delta b = -\frac{\partial}{\partial b} MSE(\hat{Y}, Y)$$

Applying Gradient Descent



$$MSE(\hat{Y}, Y) = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y} - Y)^2$$

$$\Delta m_i = -\frac{\partial}{\partial m_i} MSE(\hat{Y}, Y)$$

$$\Delta b = -\frac{\partial}{\partial b} MSE(\hat{Y}, Y)$$

Lazy!

Let’s Code It!

Let’s Code It!

- GitHub
- Jupyter Notebook
- Notable changes from last week:
  - The full autodiff library, completed, is there for your use
  - Feel free to compare it to what you’ve already done!
- Quick Explanation of TODOs:
  - `fit`: computing gradient descent and updating parameters
  - `predict`: given some data, predict the output