Homework [1] - Machine Learning 6316

Jacob Dineen / JD5ED

September 10, 2019

1 Linear Algebra Review (QA type)

Let $\mathbf{x} = (x_1, x_2, x_3)^T$ and:

$$\begin{cases} 2x_1 + 2x_2 + 3x_3 = 1 \\ x_1 - x_2 = -1 \\ -x_1 + 2x_2 + x_3 = 2 \end{cases}$$

Please answer the following questions:

1.1:

Solve the linear equations

$$\begin{aligned} 2x_1 + 2x_2 + 3x_3 &= 1 \\ -2x_1 - 2x_2 + 0x_3 &= -2 \\ 4x_2 + 3x_3 &= 3 \end{aligned}$$

We can solve the above linear equations using substitution and elimination, or via Gauss-Jordan Reduction. I've chosen the former here.

$$\begin{aligned} |x_1 - 1x_2 &= -1 \\ -1x_1 + 2x_2 + 1x_3 &= 2 \end{aligned}$$

$$1x_2 + 1x_3 = 1$$

$$1x_2 = -1x_3 + 1$$

$$\begin{aligned} 4(-x_3 + 1) + 3x_3 &= 3 \\ -4x_3 + 4 + 3x_3 &= 3 \\ -x_3 &= -1 \\ x_3 &= 1 \end{aligned}$$

$$x_2 = -x_3 + 1$$
$$x_2 = -1 + 1$$
$$x_2 = 0$$

$$x_1 - x_2 = -1$$
$$x_1 - 0 = -1$$
$$x_1 = -1$$

$$\begin{bmatrix} x_1 \\ x_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

1.2:
Write it into matrix form(i.e. Ax = b) (we will use the same A and b in the following questions.)

$$\begin{bmatrix} 2 & 2 & 3 \\ 1 & -1 & 0 \\ -1 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix}$$

1.3:
The Rank of A is ?
The rank of A is 3, which means that A is full rank. All rows are linearly independent - No linear combination of rows equal another.

1.4:
$Calculate A^{-1} and \det(A)$

$$\det(A) = 2\begin{bmatrix} -1 & 0 \\ 2 & 1 \end{bmatrix} - 2\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} + 3\begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}$$

$$= -2 - 2 + 3 = -1$$

$$A^{-1} = \frac{1}{-1}\begin{bmatrix} -1 & -1 & 1 \\ 4 & 5 & -6 \\ -3 & 3 & -4 \end{bmatrix}^{\top} = -1\begin{bmatrix} -1 & 4 & -3 \\ -1 & 5 & 3 \\ 1 & -6 & -4 \end{bmatrix} = \begin{bmatrix} 1 & -4 & 3 \\ 1 & -5 & -3 \\ -1 & +6 & 4 \end{bmatrix}$$

1.5:
Use (1.4) to solve the linear equations
We can use the property of matrix inversion $(A^T A)^{-1} = (A^{-1}A^{-1T})$ and use the normal equation $(A^T A)^{-1}A^T B$

$$\begin{bmatrix} 1 & -4 & 3 \\ 1 & -5 & -3 \\ -1 & 6 & 4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ -4 & -5 & 6 \\ 3 & -3 & 4 \end{bmatrix} = \begin{bmatrix} 26 & 30 & -37 \\ 30 & 35 & -43 \\ -37 & -43 & 53 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -4 & 3 \\ 1 & -5 & -3 \\ -1 & 6 & 4 \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 \\ -4 & -5 & 6 \\ 3 & -3 & 4 \end{bmatrix} = \begin{bmatrix} 26 & 30 & -37 \\ 30 & 35 & -43 \\ -37 & -43 & 53 \end{bmatrix}$$

$$A^\top B = \begin{bmatrix} -1 \\ 7 \\ 5 \end{bmatrix}$$

$$\theta = \begin{bmatrix} 26 & 30 & -37 \\ 30 & 35 & -43 \\ -37 & -43 & 53 \end{bmatrix}\begin{bmatrix} -1 \\ 7 \\ 5 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

$$A = \left(A^{-1}A^{T^{-1}}\right) \quad B = A^T B$$

1.6:

Calculate the inner product and outer product of x and b.(i.e.$\langle$x, b$\rangle$ and x $\otimes$ b)|

$$\langle x, b\rangle = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix} = -1(1) + 0(-1) + 1(2) = 1$$

$$x \otimes b = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}\begin{bmatrix} 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 1 & -2 \\ 0 & 0 & 0 \\ 1 & -1 & 2 \end{bmatrix}$$

1.7:

Calculate the L_1, L_2 and L_∞ norm of **b**

$$b = \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix}$$

$$L_1 = |1| + |-1| + |2| = 4$$
$$L_2 = \sqrt{(1)^2 + (-1)^2 + (2)^2} = \sqrt{6}$$
$$L_\infty = 2$$

1.8:

Now we add one more linear equation $-x_1 + 2x_2 + x_3 = 1$ into linear equations above. Write it into matrix form(i.e. $A_1 x = b$)

$$\begin{bmatrix} 2 & 2 & 3 \\ 1 & -1 & 0 \\ -1 & 2 & 1 \\ -1 & 2 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 2 \\ 1 \end{bmatrix}$$

1.9:

The rank of A_1 is?

$Rank(A_1) = 3$

1.10:
Could these linear equations be solved? Why ?
No, they can not be solved because A, of size 4X3, is not a square matrix and is not invertible.
A is a singular matrix.

1.11:

Calculate the Pseudo-inverse of $\mathbf{A}_1$

$$\text{Suppose } \mathbf{B} = \begin{bmatrix} -\frac{2}{3} & \frac{1}{3} & \frac{2}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} & \frac{1}{3} \end{bmatrix}$$

(You can use a tool to solve it.)

$A^+ = \left(A^T A\right)^{-1} A^T$

$$A^+ = \begin{bmatrix} 1 & -4 & -1.5 & -1.5 \\ 1 & -5 & -1.5 & -1.5 \\ -1 & 6 & 2 & 2 \end{bmatrix}$$

1.12:
Is B an orthogonal matrix? Why ?

A matrix is orthogonal if $B(B^T) = B^T(B) = I$, where I repesents an identity matrix.

$$\begin{bmatrix} -\frac{2}{3} & \frac{1}{3} & \frac{2}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} -\frac{2}{3} & -\frac{1}{3} & \frac{2}{3} \\ \frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{2}{3} & \frac{1}{3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1.13:

$$\text{Suppose } \mathbf{y} = (y_1, y_2, y_3)^T, \text{ calculate } \nabla_{\mathbf{y}} \mathbf{y}^T \mathbf{A}\mathbf{y} = \frac{\partial \left(\mathbf{y}^T \mathbf{A}\mathbf{y}\right)}{\partial \mathbf{y}} =$$

We can use the product rule to solve for the gradient.

$$\begin{aligned} \frac{\partial}{\partial y} &= y^\top \frac{dAy}{dy} + Ay\frac{dy}{dy} \\ &= y^\top A + Ay \\ &= y^\top A + (Ay)^\top \\ &= y^\top A + A^\top y^\top \\ &= y^\top \left(A + A^\top\right) \end{aligned}$$

2 Linear Regression Model Fitting (Programming)

2.1: Coding Portion

All code submitted separately in .py.

2.2: Written Portion

We begin by reading the given file in. The file has two columns, $x_0 and x_1$. At first glance, I had thought that the first column was reflective of some binary flag, but after receiving clarification, it was determined to just be a bias vector of size n. As such, we have a single variable linear regression problem of the form y=mx+b, which is extended in the case of matrix math to f(X) = XW + B, where X represents the inputs, W represents the feature map/ weights, and B represents our bias terms. $w \in W$ and $b \in B$ are learnable parameters, meaning their values are dynamic subject to the chosen optimization method, which, in this case, are variants of gradient descent (stochastic, batch, minibatch). SGD variants will iteratively optimize our learnable parameters so as to minimize the accrued loss $J(\theta)$. In this case, our loss function is sum of square errors, which has a nice property when you are taking the partial w.r.t. x_i.

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^{n} (f(\mathbf{x}_i) - y_i)^2 \tag{1}$$

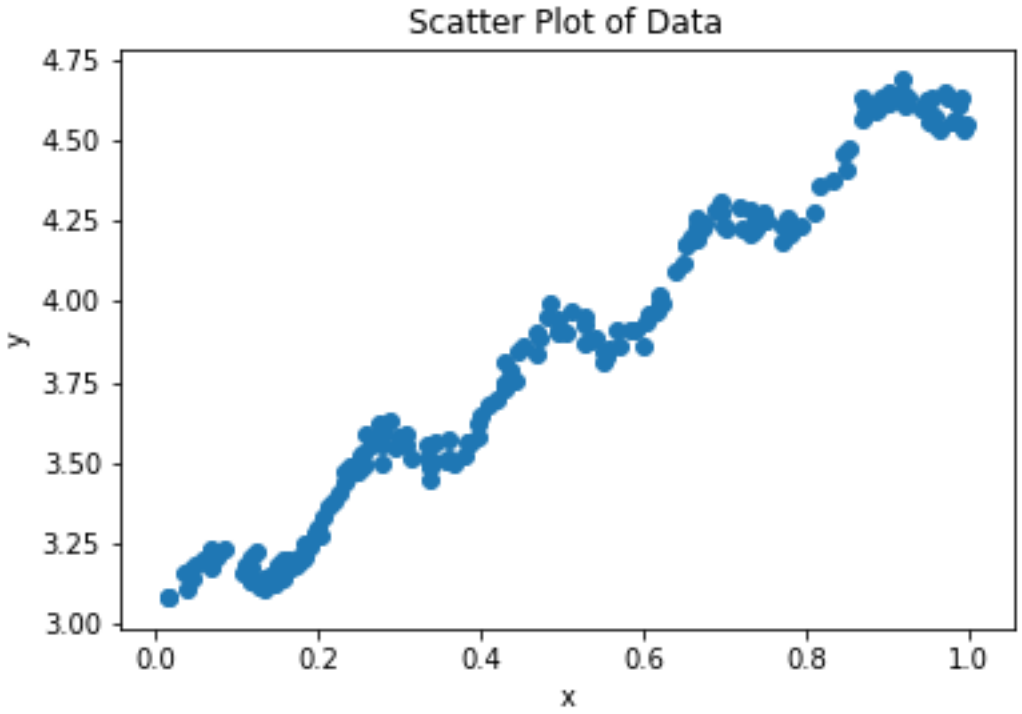
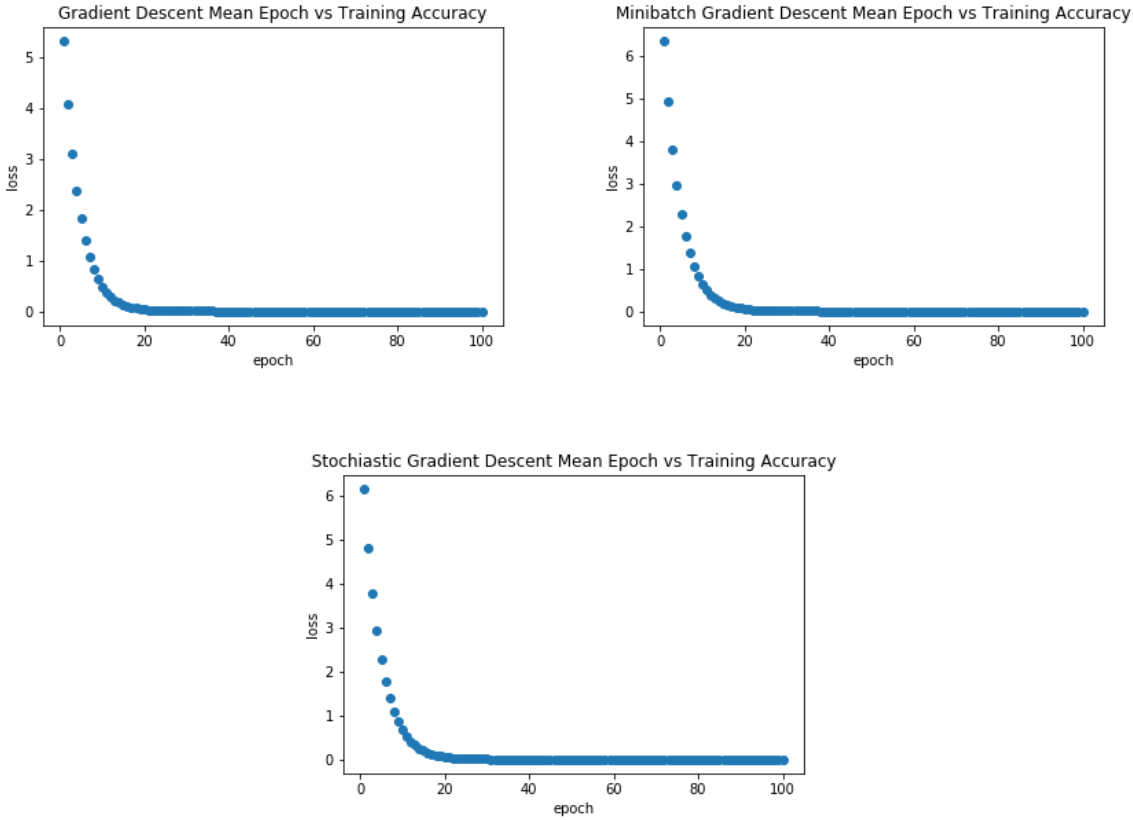


Figure 1: Scatter Plot showing the independent variable against the output

Note that in equation 1, I've seen this written as 1/2 * the sum of the squared differences, but my previous understanding was that we are looking at the average cost (1/2n) over an epoch

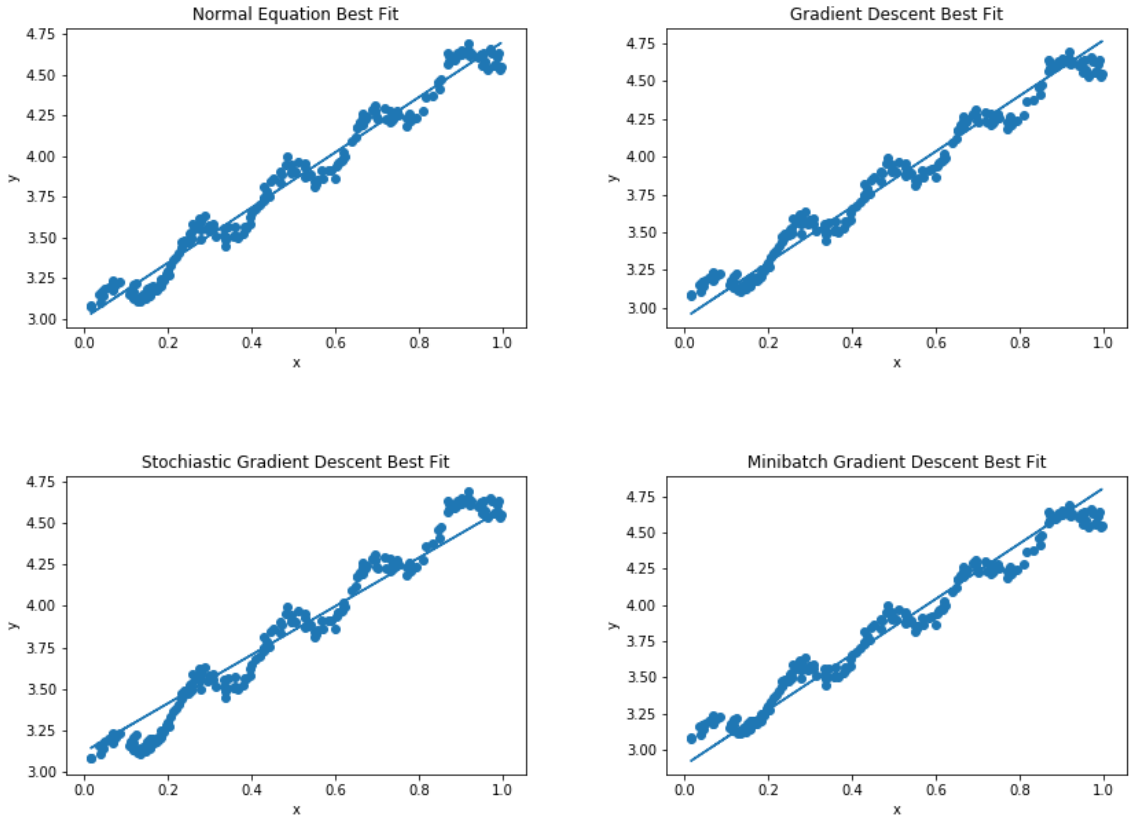
and backpropogating that error signal throughout the model, where n is the number of rows in the training set. My implementation proceeded as such.

*Note that the homework was revised to advise us to use MSE rather than SSE for error propagation, so the above explanation should be correct. We also have an additional constraint where $t_max = 100$, meaning our learning algorithms can only train for a maximum of 100 epochs. An epoch is completed once the entire training set has been iterated through - In Mini Batch Stochastic Gradient Descent, we are really computing the gradients $\frac{lengthofdataset}{batchsize}$ times before an epoch is completed. In Stochastic Gradient Descent, we are performing random sampling over the entire dataset n times during each epoch. Please note that SGD and minibatch gradient descent underwent shuffling prior to inference, as per requirements.



Convergence Rates SGD Variants

Above, we show how the training cost decreases over the finite number of iterations. While you may train for much longer in practice, these graphs show us that all algorithms display elements of learning, as their losses decrease continuously throughout their runs, with SGD converging the quickest here. Being as how this can be solved in closed form via the normal equation, it is not particularly surprising that we can converge to a solution in a smaller amount of updates. Taking this a step further, we display the learned lines of best fit through the data.

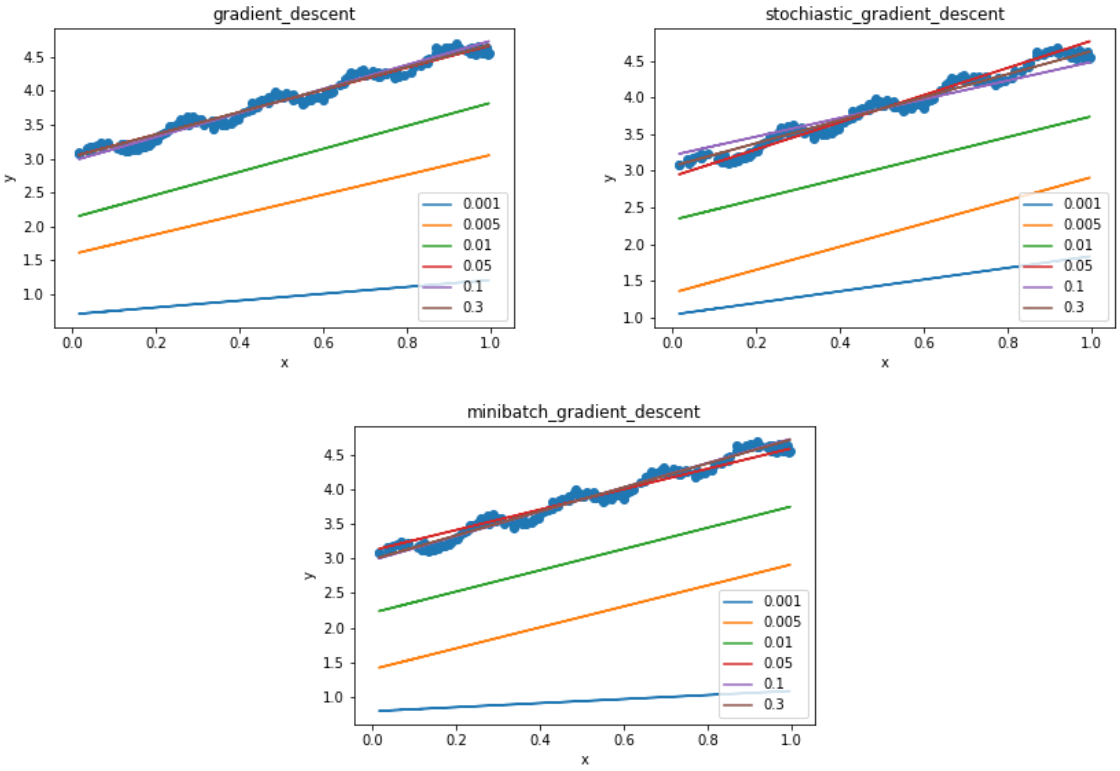


Lines of Best fit through data

The learned parameters are as follows***:
$NormalEquation : f(X) = (3.0077) + (1.6953)x$
$GradientDescent : f(X) = (3.0350) + (1.6434)x$
$StochasticGradientDescent : f(X) = (3.1712) + (1.5346)x$
$MinibatchGradientDescent : f(X) = (2.9472) + (1.8109)xx$

***There was no random seed set for the shuffling, so each run is likely to have a different fit, i.e., these formulas and graphs displayed here are likely to be slightly different with each run of regressiontemplate.py.

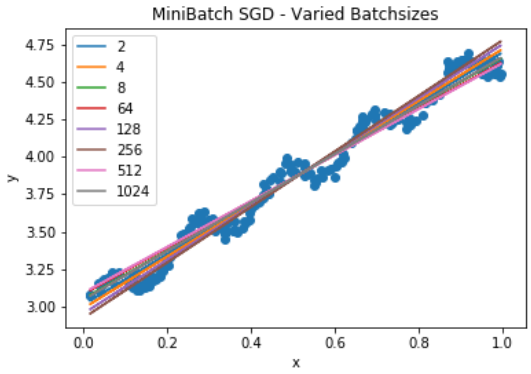
The batch (full batch) GD learned function most closely resembles the closed form solution using the normal equation. It should be noted that running for a $t_{max} > 1000$ produces functions that fit the data nearly perfectly on all variants. All variations of SGD were trained with a learning rate of 0.01, respectively. The batch size for mini batch SGD was set to 64, and the number of epochs was set to 100 across the board.



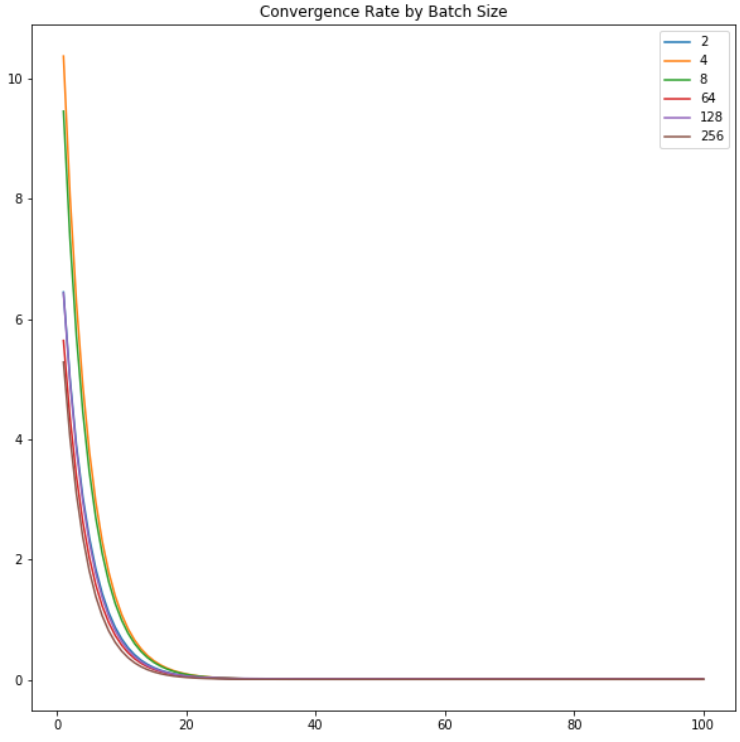
Iterating through Learning Rates

We iterate through through learning rates $lr_i \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.3\}$ to see the impact of alpha on the overall fit of the line through the data. A higher learning rate means that we are taking bigger steps in gradient descent (We're subtracting the gradient times the learning rate from the weight at each timestep - This has the benefit of sometimes converging quicker, but sometimes overstepping local/global optima. The above graphs show that all when the learning rate was greater than 0.01, we were able to converge within the constraint $t_max = 100$. I believe most SotA algorithms these days employ some kind of learning rate scheduling or cosine annealing, rather than a fixed learning rate throughout the training phase. E.g., a less naive implementation may consider momentum to help escape saddle points in highly nonconvex loss surfaces.

We also explore various batch sizes on our minibatch SGD variant with a learning rate of 0.1. In practice, I believe you tune this to be powers of two when hyperparameter searching. The resulting graph doesn't tell us much. *Because of the stochastic nature of the sampling, we get different fits with each run. We can see via the right graph that we appear to converge to a global minima quicker when the batch size increases. I believe this ties in with the fact that we get a more accurate representation of the gradient as the size of the minibatch approaches the size of the training set - Minibatch GD was meant to be a 'best of both worlds' combination of SGD and Full Batch Gradient Descent, so as we increase the size of the batch, the computational cost of approximating the gradient increases.



Varied Batch Sizes - Line of Best Fit



Varied Batch Sizes - Convergence Rates

Batch size: 2 cost: θ.007
Batch size: 4 cost: θ.088
Batch size: 8 cost: θ.088
Batch size: 64 cost: $\theta.\theta$1
Batch size: 128 cost: θ.607
Batch size: 256 cost: θ.687
Batch size: 512 cost: 0.007
Batch size: 1024 cost: θ.017

3 Sample Exam Questions:

3.1. Linear Regression+ Train-Test Split

What is the mean squared training error when running linear regression to fit the data ? (i.e., the model is $y = \beta_0 + \beta_1 x$)). Assuming the rightmost three points are in the test set, and the others are in the training set. (you can eyeball the answers.)

We can see that the first four points (the training set) can have a linear function fit through them with little to no error (eyeballing). The model has an intercept term of zero, as it passes through the origin, and appears to be of the form $f(x) = x$. Computing the Mean Squared Error w.r.t the data points equates to a loss of 0, or an r^2 of 1.00. In this case, as the test set also falls on the same line as the fitted function (f), we should also have a test error of 0.

3.2. Linear Regression+LOOCV

Note: LOOCV means (Leave-One-Out-Cross-Validation) Suppose you are given a data set with three data points (see both table and figure). This dataset includes one real-valued input variable and one real-valued output variable. We are trying to learn a regression function to map from the input to the output. (Key: mostly likely you only need 10 minutes to solve the following three sub-questions.)

x	y
0	2
2	2
3	1

(a) What is the mean squared leave one out cross validation error if using linear regression ?

In this step we will fit a model through x_2 and x_3, x_1 and x_3, x_1 and x_2 . $LOO - CV_i$ where $i \in \{1, 2, 3\}$ represents the model.

$LOO - CV_1 : y = -x + 4 : Loss = (4 - 2)^2$
$LOO - CV_2 : y = -\frac{1}{3}x + 2 Loss = (-\frac{2}{3})^2$
$LOO - CV_3 : y = 2 : Loss = (2 - 1)^2$

Putting it all together we have: $CVError = \frac{49/9}{3} = \frac{49}{27}$

(b) Suppose we use a trivial algorithm of predicting a constant y = β_0 (Att: you can assume any function form to map from input to output ! This is your assumption !). What is the mean squared leave one out error in this case? (Assume β_0 is learned from the non-left-out data points.)

$LOO - CV_1 : y = 1.5 : Loss = (2 - 1.5)^2$
$LOO - CV_2 : y = 1.5 : Loss = (2 - 1.5)^2$
$LOO - CV_3 : y = 2 : Loss = (2 - 1)^2$

we have: $CVError = 0.5$

(c) Based on the LOOCV results, which model category will you pick for this dataset: the category described in (a) or (b)?

While the trivial algorithm looks like an attractive option, predicting a horizontal line for all test set/online data will prove hazardous in the long run. The variance of our training set is small and the category B model would allow for us to accrue a lesser cost in the short term, but at the sacrifice of being able to generalize in the future. This seems like a textbook case of underfitting.

Question 3.3. Error Evaluations for Regression
We are given a dataset with R records in which the ith record has one real-valued input attribute xi and one real-valued output attribute yi. By running a linear regression method on this data, we choose at random some data records to be a training set, and choose at random some of the remaining records to be a test set. Now let us increase the training set size gradually. As the training set size increases, what do you expect will happen with the mean training and mean testing errors? (Please provide 1 sentence of explanation to justify your choice.)

(e) Mean Training Error:
- A. Increase;
- B. Decrease
Your choice:
Looking back to the previous example, as these two questions really concern the bias-variance tradeoff, we look at model complexity. If we take a model with a single parameter, fitting just a horizontal line, we have a very simple model that has low variance. This model will perform well on the three data points, but as we add more data gradually, we'll see our error explode. Our model is biased toward the training set. If we instead fit increase the number of parameters and fit a line with a non-zero slope through the data, we have decreased the bias, but increased the variance of our model. They key is to find a happy medium of the two, which I'm sure will be discussed in future classes.

(f) Mean Testing Error:
- A. Increase;
- B. Decrease

Your choice:
Increasing the size of the training set should improve the ability to generalize come test time, given that we have now seen more examples and fit a model through data that might otherwise seem irregular at inference time. If the data is highly noisy, we might learn a function that doesn't generalize well (Overfitting the training data), however this would likely be more of an issue when dealing with a model with higher complexity / more parameters. This is why we generally have a holdout test during training time that comes from the same distribution as the test set and helps us to understand how our model will perform on unseen data.

All in all, there's not a concrete answer here. Increasing the size of the training set could increase the bias or variance of our model depending on the data. Bias is closely associated with underfitting, and variance is related to overfitting.