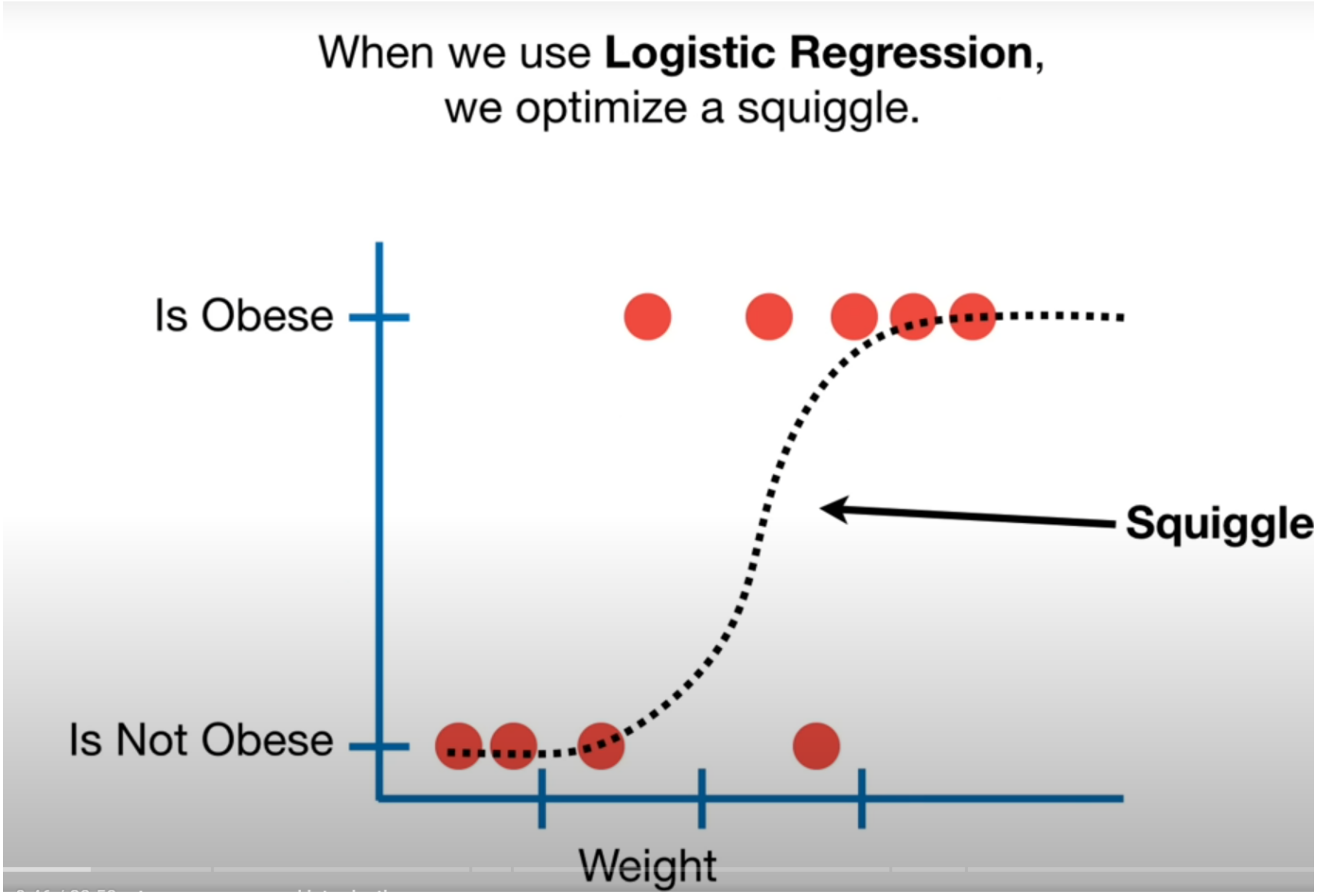
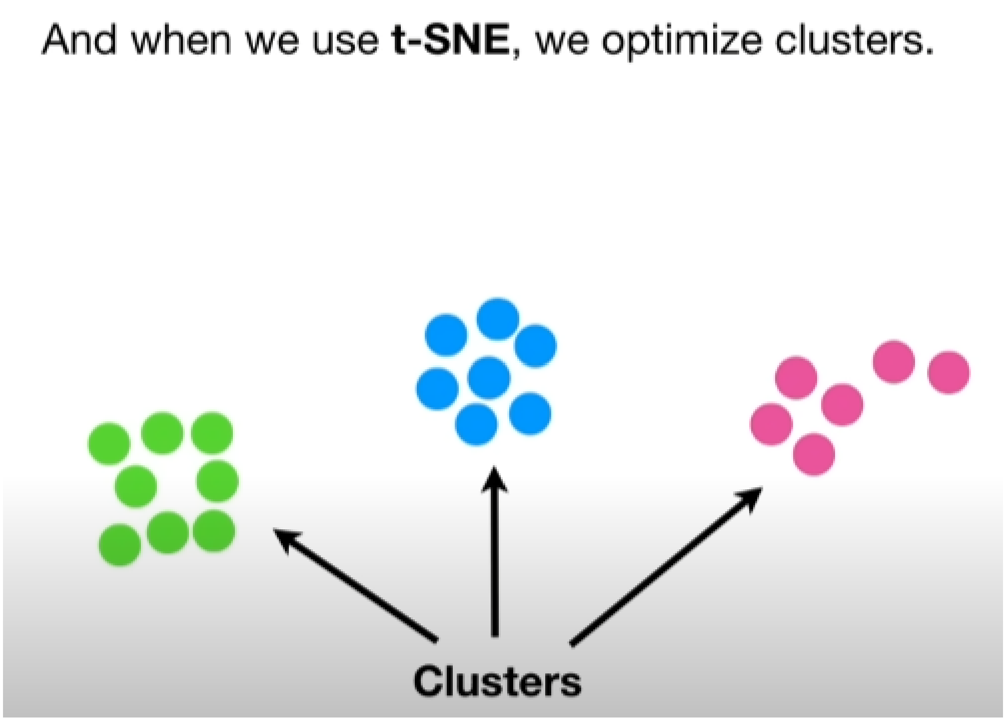
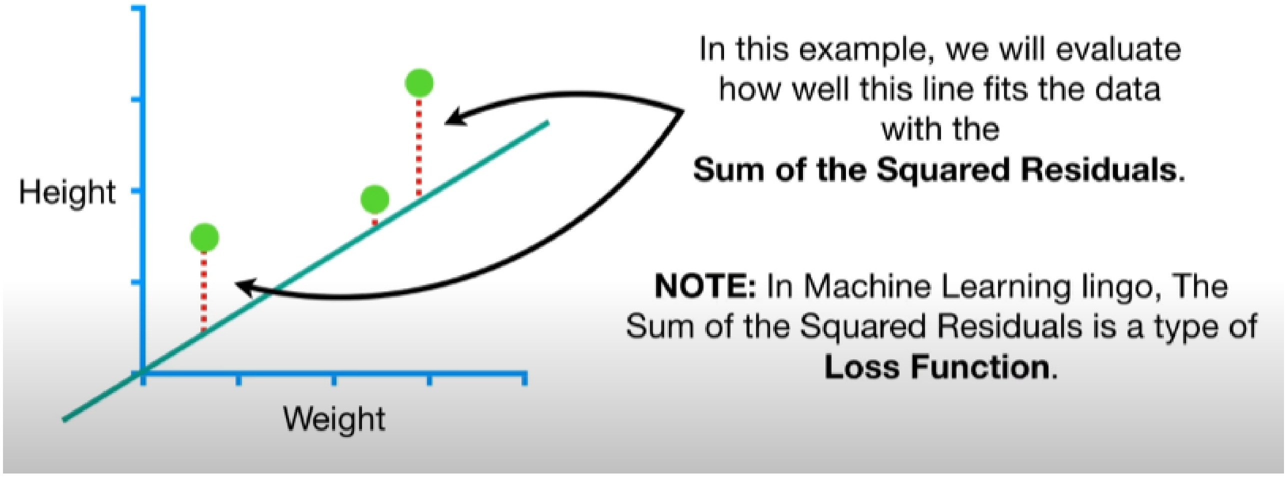
Gradient **DESCENT**

- In Stats, ML and other Data Science field, We optimize a lot of stuff.
- In Linear Regression, we optimize intercept and slope. Eq. : $\text(y = mx + c)$
- In Logistic one, we optimize a squiggle.

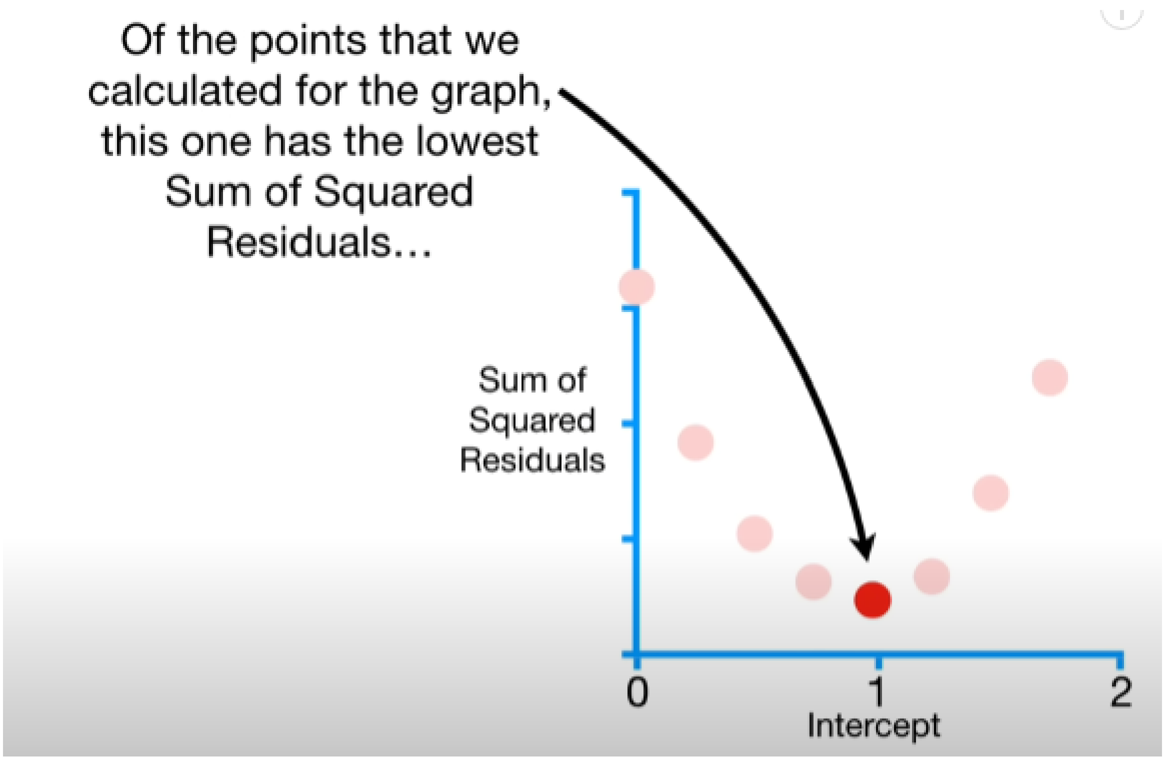


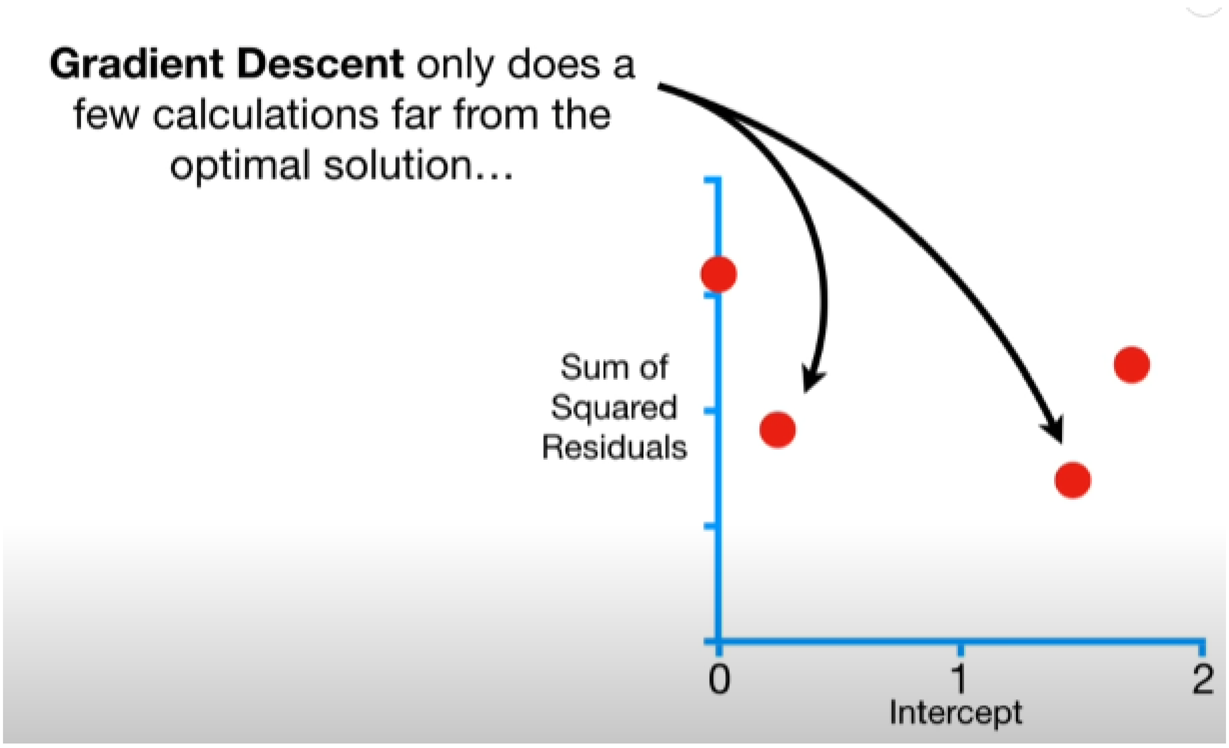


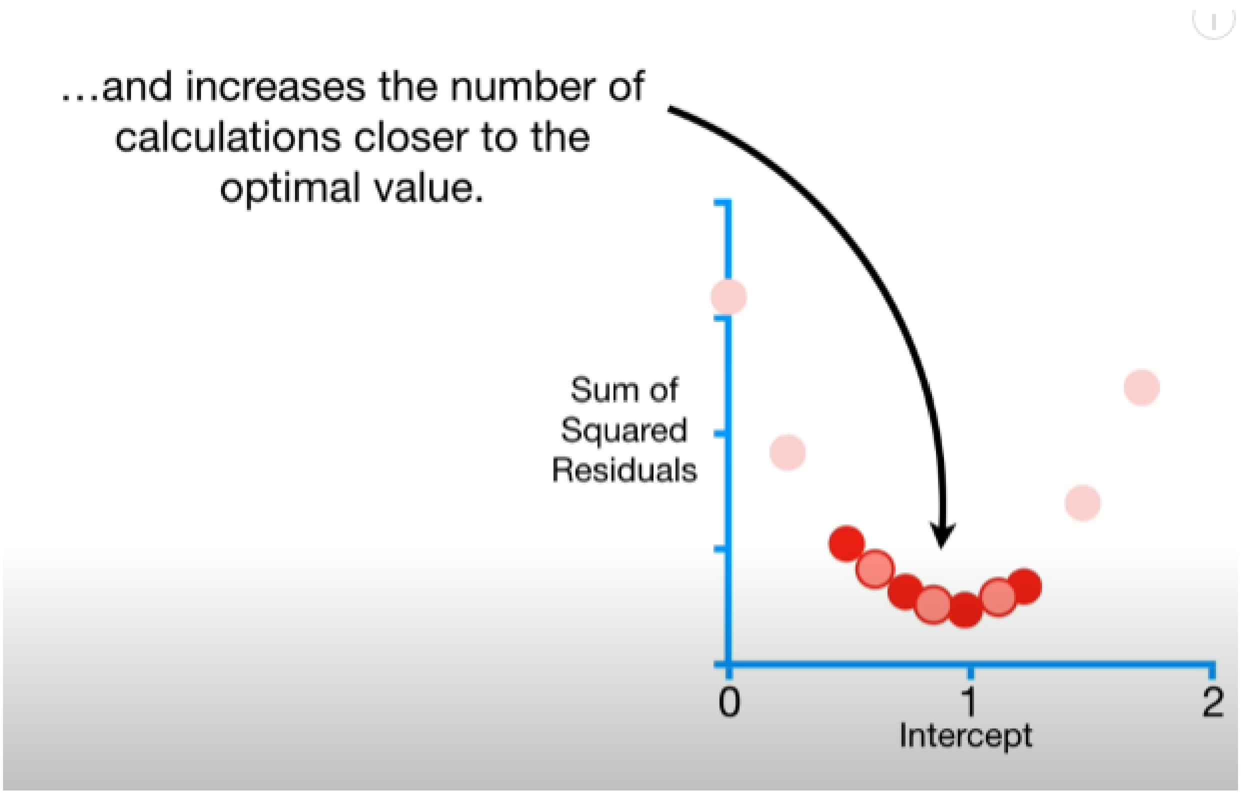
- Gradient Descent can optimize all this stuff and much more.
- Suppose, you have 3 points in a graph for which you have to find optimized intercept and slope.
- First, You assume intercept so that you have something to improve upon. (Also, slope as least square estimate)

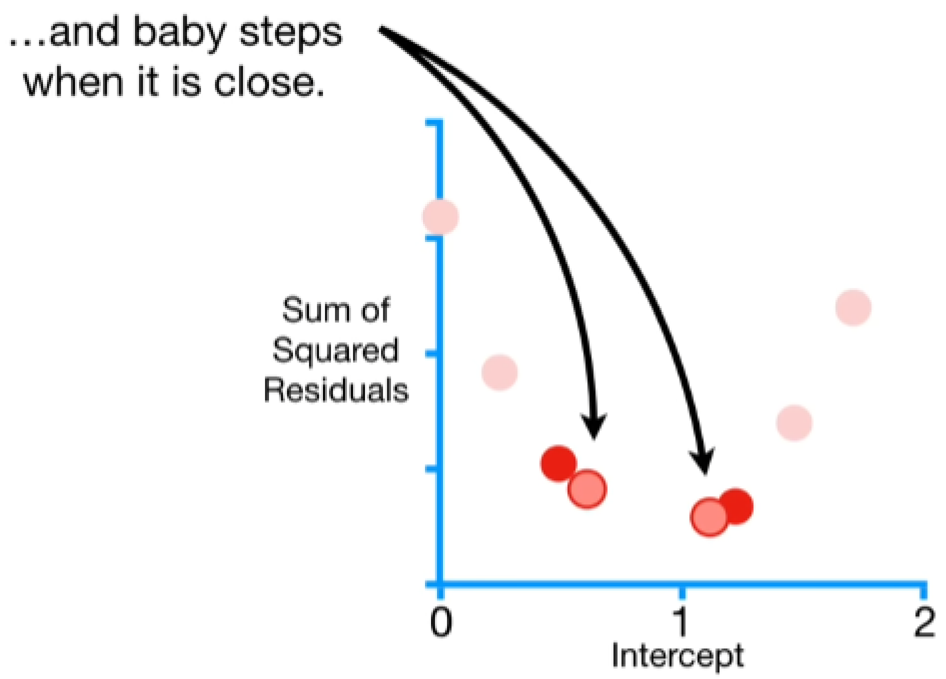


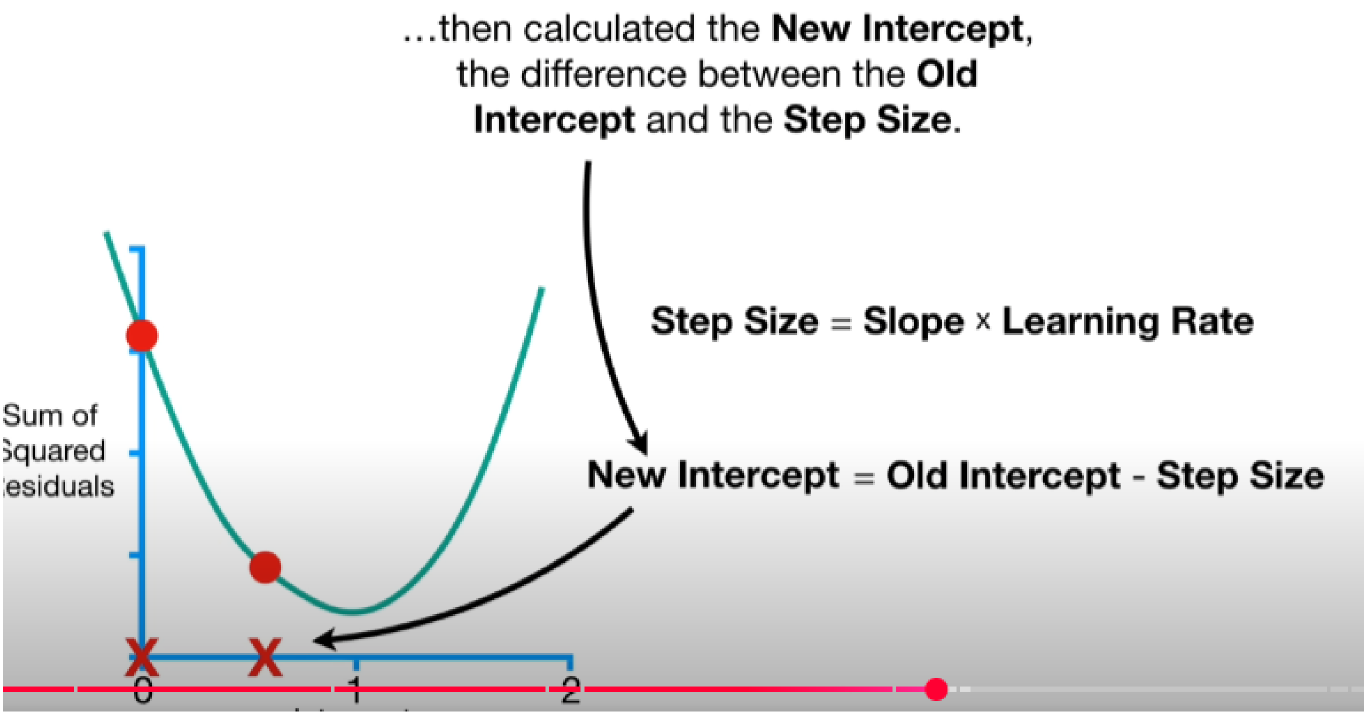
- Find residual between observed and predicted and using that → Sum of Squared Residuals. $\sum_{i=1}^{n} (\text{observed}_i - \text{predicted}_i)$
- Then plot a graph of Intercept(X) - Sum of Squared Residuals(Y)


Of the points that we calculated for the graph, this one has the lowest Sum of Squared Residuals...
Sum of Squared Residuals
0
1
2
Intercept


Gradient Descent only does a few calculations far from the optimal solution...
Sum of Squared Residuals
0
1
2
Intercept


...and increases the number of
calculations closer to the
optimal value.
Sum of
Squared
Residuals
0
1
2
Intercept


...and baby steps
when it is close.
Sum of
Squared
Residuals
0
1
2
Intercept


...then calculated the **New Intercept**, the difference between the **Old Intercept** and the **Step Size**.
Step Size = Slope × Learning Rate
New Intercept = Old Intercept - Step Size
Sum of
Squared
esiduals
0
1
2

Step 1: Take the derivative of the **Loss Function** for each parameter in it.
In fancy Machine Learning Lingo, take the **Gradient** of the **Loss Function**.

Step 2: Pick random values for the parameters.

Step 3: Plug the parameter values into the derivatives (ahem, the **Gradient**).

Step 4: Calculate the Step Sizes: **Step Size** = **Slope** × **Learning Rate**

Step 5: Calculate the New Parameters:

New Parameter = **Old Parameter** - **Step Size**

Now go back to **Step 3** and repeat until
Step Size is very small, or you reach
the **Maximum Number of Steps**.

Step 3: Plug the parameter values into the derivatives (ahem, the **Gradient**).

Step 4: Calculate the Step Sizes: **Step Size** = **Slope** × **Learning Rate**

Step 5: Calculate the New Parameters:

New Parameter = **Old Parameter** - **Step Size**

- For large datasets, this approach could be in-efficient.

So there is a thing called **Stochastic Gradient Descent** that uses a randomly selected subset of the data at every step rather than the full dataset.

This reduces the time spent calculating the derivatives of the **Loss Function**.