# Gradient Descent

Recall Linear Regression: the prediction from a linear regression model is

$$\hat{y} = x\beta = xw$$

The error, or residual is calculated by

$$\varepsilon = y - \hat{y}$$

To measure the sum of squared errors:

$$SSE = (y - X\beta)^T(y - x\beta) = \sum_i (y - \hat{y}_i)^2$$

Remember, the goal is to *minimize* the sum of squared error.

But when can we not derive $\hat{\beta}$ directly? There are two cases where it's not possible.

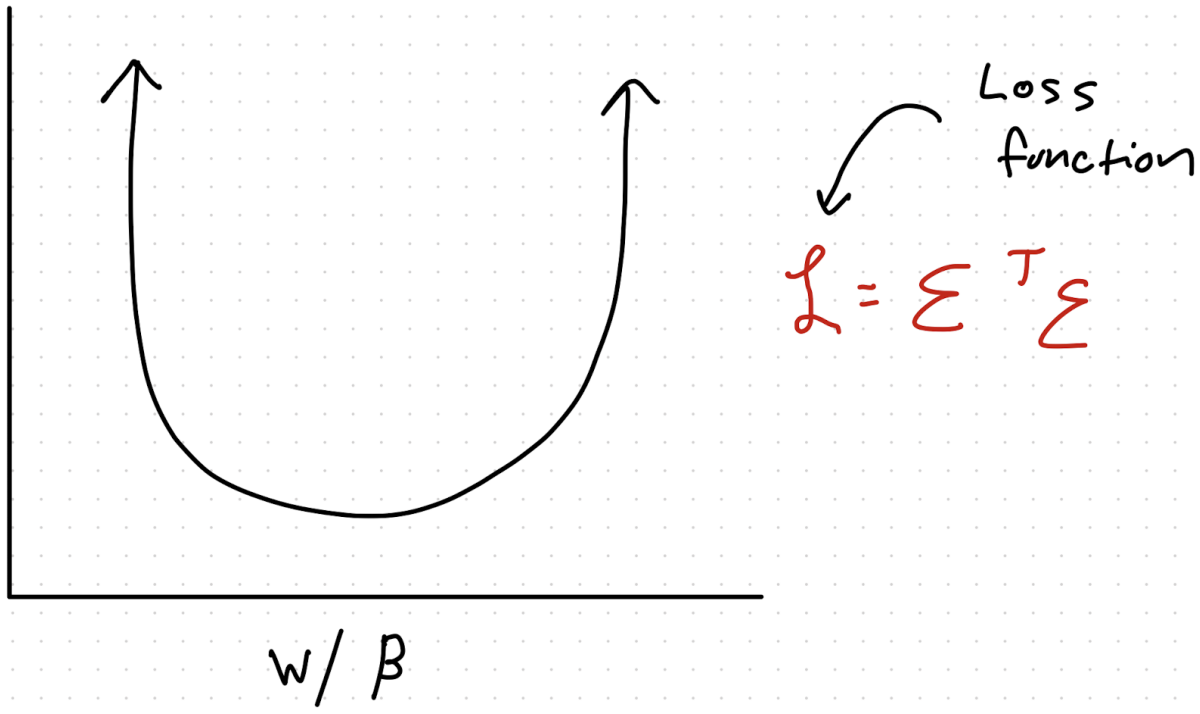
- For matrices, if you have more features than rows, i**t's not possible to transpose**. $(X^TX)^{-1}$ may not always exist.
- Transposing a matrix may be computationally expensive to compute. It might not be practical to directly derive the weights of$(X^TX)^{-1}$

However, this doesn't mean that there isn't some set of optimal weights for an impossible or expensive $(X^TX)^{-1}$. This is where gradient descent comes in

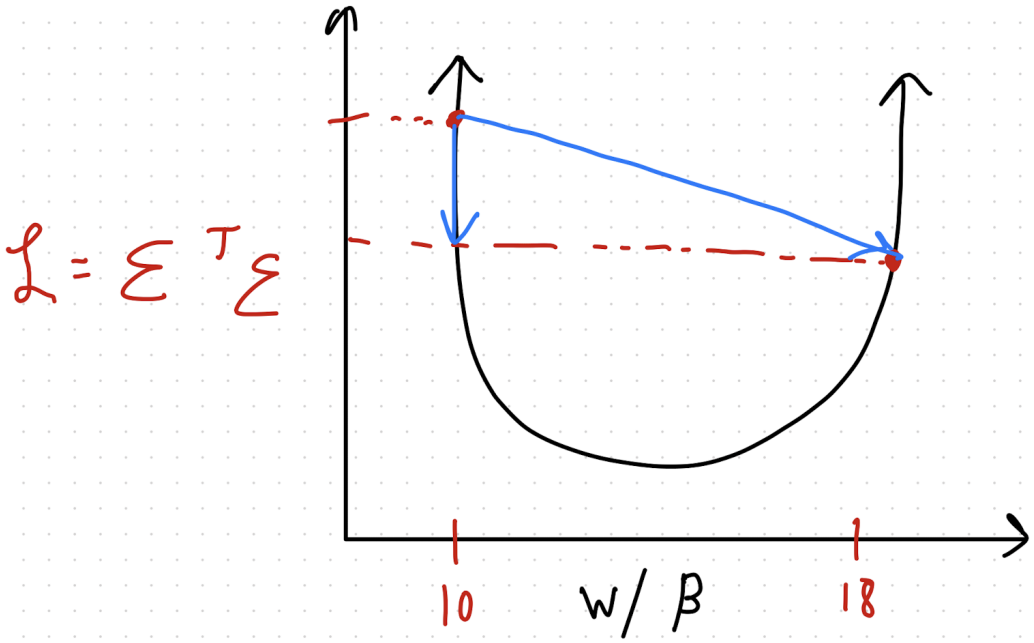
# Gradient Descent

---

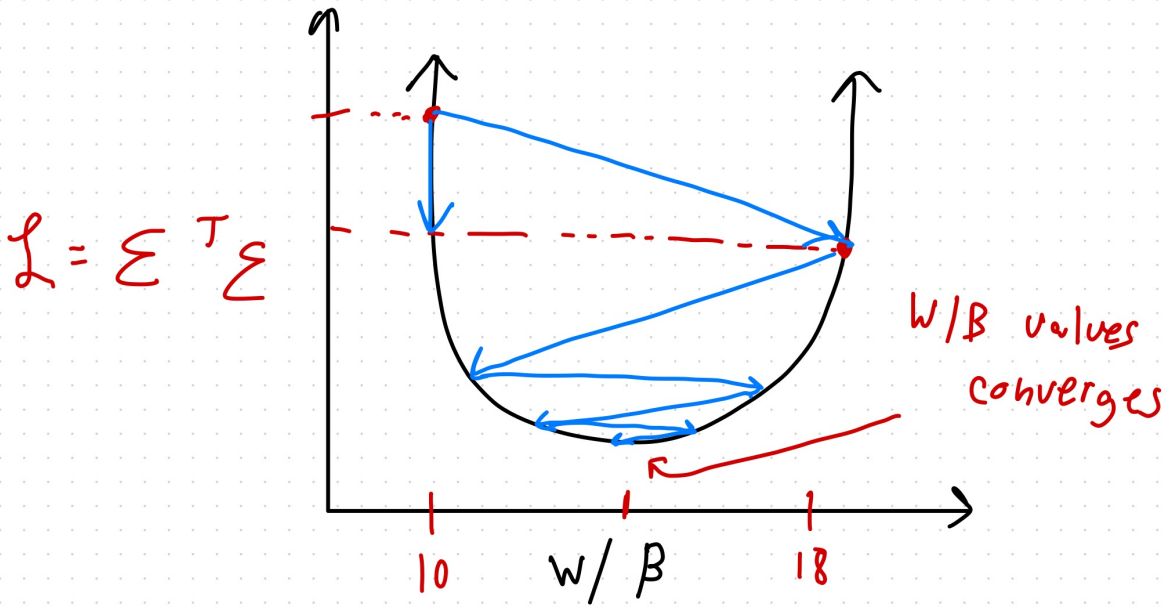
In mathematics, gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. What does it really mean to derive something?




If we plot the SSE (or loss function) on cartesian plane, we can visualize what we are trying to do when we minimize error.




Moving the weight from 10 to 18 for this function shows a noticeable decrease in loss. But this is not adequate. The error can be reduced again and again until it reaches a convergence point.

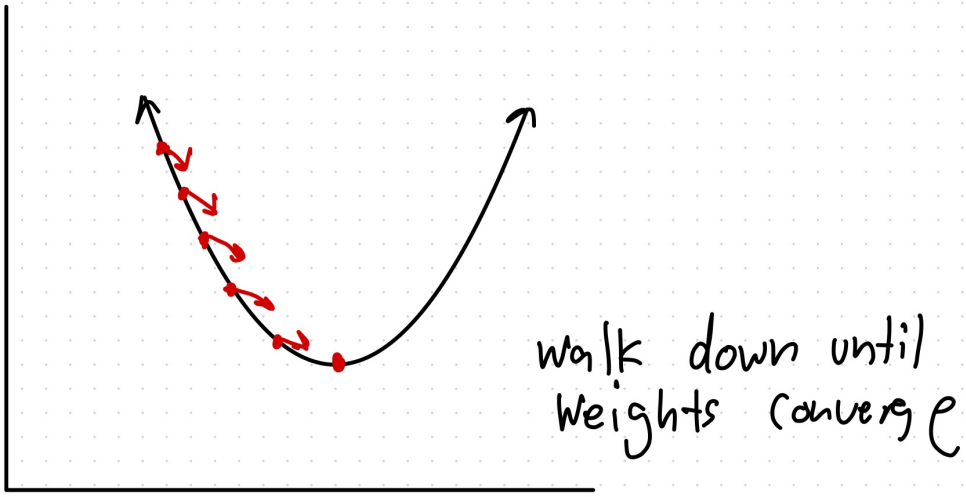



Notice how we can adjust the points in which the error is determined. We call this the **learning rate**, or $\alpha$. This controls how large of a step we take when adjusting weights ($w/\beta$).

- $\alpha$ can be some constant.
- $\alpha$ can also be some variable rate with some decay

Also notice how the learning rate makes smaller and smaller jumps as it converges.

We can also converge in steps downwards instead of going back and forth.




What controls the changes in weights are $\Delta$s. When they are small, you converge slower. If they are large, converging my be more difficult. $\alpha$ controls the step size. We call these changes **epochs**, or the amount of jumps until the curve converges. The larges gains are typically in the first couple of epochs.

If the learning curve ever starts increasing, it runs into a problem called "overfitting". That is an indicator for you to stop training the model. A variable learning rate is ideal for large jumps at the beginning and smaller jumps towards convergence.

# Notation for Gradient Descent

Recall:

$$\hat{y} = \overrightarrow{x} \cdot \overrightarrow{w}$$

The hypothesis function is denoted as

$$h(\overrightarrow{x})$$

which means that given some input data, it produces a prediction.

$$h(\overrightarrow{x}) = \hat{y}$$

It might also be represented as:

$$h_w(\overrightarrow{x})$$

which basically means "predict response/target for $\overrightarrow{x}$ for current $\overrightarrow{w}$.

A loss function is denoted as:

$$\mathcal{L}()$$

You can think about it as some function to minimize for the model. From a research based perspective, we can add a "penalty term" for the weights it selects to help train the model. For input, we usually accept the data, some true value, and a predicted value.

$$\mathcal{L}(X, y, \hat{y}) = \mathcal{L}(X, y, h_w(\overrightarrow{x}))$$

Once again, all this means is to calculate error. More rigorously, we can view this as:

$$\mathcal{L}(X, y, h_w(\overrightarrow{x})) = \frac{1}{2n}[\sum_{i}^{n}(y_i - h_w(x^i))^2]$$

using the *mean squared error*.

# The Partial Derivative

---

How can we take a partial derivative of a function?

The partial derivative of our loss function for any given weight is going to equal to the partial derivative with respect of that weight times the summation from $i$ to $n$ of $y_i - h_w(\overrightarrow{x})$ squared. So for a given $w_3$:

$$\frac{\partial}{\partial w_2} = \frac{1}{2w_3}[\sum_{i}^{n}(y_i - (w_0 + w_1x_1 + w_2x_2 + w_3x_3))^2]$$

or

$$\frac{\partial}{\partial w_3}[\frac{1}{2n}[\sum_{i}^{n}(y_i - \hat{y}_i)^2]]$$

Now recall that

$$\frac{d}{dx}[x^2] = 2x$$

and the chain rule states:

$$\frac{d}{dx}[(-)^2] = 2 \cdot () \cdot \frac{d}{dx}[()]$$

this is essential what we will do when we take the partial derivative of the loss function.

It simplifies to:

$$\frac{1}{n}[\sum_{i}^{\hat{n}}(y_i - \hat{y}_i) \cdot \frac{\partial}{\partial w_3}[y_i - \hat{y}_i]]$$

where

$$\frac{\partial}{\partial w_3}[y_i - \hat{y}_i]$$

simplifies down to $-x_3$. Giving us the total result of:

$$\frac{1}{n}[\sum_{i}^{n}(y_i - \hat{y}_i) \cdot (-x_3)]$$

$$\frac{\partial}{\partial w_i}[\mathcal{L}()] = \frac{1}{n}[\sum_{i}^{n}(y_i - \hat{y}_i) \cdot (-x_3)]$$

Essentially for every single value in the vector of weights:

Update $w_i$ : $w_i = w_i - \alpha \cdot \frac{\partial}{\partial w_i}[\mathcal{L}]$

$$\nabla_w[\mathcal{L}()] = \prod_{i}^{n} \frac{\partial}{\partial w_i}$$