

CS 331

Computer Vision

10 / 20 / 2026

Instructor: Michael Eckmann

Today's Topics

- Questions / Comments?
- Linear Regression examples
- Linear Regression for higher-order polynomials

When to stop learning

- Convergence could mean:
 - Training loss not changing very much between epochs
 - Validation accuracy change is negligible or going down (in which case go back to state of best validation accuracy --- early stopping)
 - Validation loss stops decreasing (in which case go back to state of best validation accuracy --- early stopping)
- Let me show you output of a network that did early stopping.

Linear Regression

- Remind us of what linear regression is, and how it is done

Gradient Descent

- Regression – vector of real inputs, a scalar real output
- Linear Regression – learns a linear equation of the inputs
- Use gradient descent to minimize a loss function (we used MSE as the loss function)
 - The linear function we're trying to learn has two learnable parameters (aka weights) w_0 is the y-intercept and w_1 is the slope
 - Gradient of MSE is made up of two partial derivatives (one with respect to w_0 and the other with respect to w_1)

Gradient Descent

- MSE loss function

$$\begin{aligned} J &= \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2 \\ &= \frac{1}{2m} \sum_{i=1}^m (w_0 + w_1 x_1^{(i)} - y^{(i)})^2 \\ &= \frac{1}{2m} \sum_{i=1}^m ((\hat{y}^{(i)})^2 - 2\hat{y}^{(i)}y^{(i)} + (y^{(i)})^2) \\ &= \frac{1}{2m} \sum_{i=1}^m \left((w_0 + w_1 x_1^{(i)})^2 - 2y^{(i)}(w_0 + w_1 x_1^{(i)}) + (y^{(i)})^2 \right) \\ &= \frac{1}{2m} \sum_{i=1}^m \left(\left(w_0^2 + 2w_0w_1x_1^{(i)} + w_1^2 (x_1^{(i)})^2 \right) - 2y^{(i)}(w_0 + w_1 x_1^{(i)}) + (y^{(i)})^2 \right) \end{aligned}$$

Gradient Descent

- Partial Derivative of MSE w/r/t w_0 and
- Partial Derivative of MSE w/r/t w_1

$$\frac{\partial J}{\partial w_0} = \frac{1}{m} \sum_{i=1}^m \left(w_0 + w_1 x_1^{(i)} - y^{(i)} \right)$$

$$\frac{\partial J}{\partial w_1} = \frac{1}{m} \sum_{i=1}^m \left(w_0 x_1^{(i)} + w_1 \left(x_1^{(i)} \right)^2 - y^{(i)} x_1^{(i)} \right)$$

- Gradient is a vector of those two partials

Gradient Descent

- More common and consistent forms of the Partial Derivatives:

$$\frac{\partial J}{\partial w_0} = \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)}) x_0^{(i)}$$

$$\frac{\partial J}{\partial w_1} = \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)}) x_1^{(i)}$$

where $x_0^{(i)} = 1$ and $\hat{y}^{(i)} = w_0 x_0^{(i)} + w_1 x_1^{(i)}$

Gradient Descent

- The gradient points in the direction of steepest ascent. We want to travel down, not up in the loss function, so we go in the opposite direction of the gradient.
- To go in the opposite direction of the gradient, when we update the weights, we subtract (see next slide.)

Gradient Descent

- The gradient descent algorithm is this:

Loop until convergence

Use all data points to compute the partial derivatives

Loop for all j 's (number of weights)

$$w_j = w_j - \text{learnRate} * \text{partialDerivOfLoss w/r/t } w_j$$

- In our case of linear regression we have 2 weights so j goes from 0 to 1
- The learning rate is a hyperparameter.

Example of Linear Regression

- I suggest you revisit the spreadsheet for homework. Read the text file I posted and make sure you can understand why the formulas are what they are in certain cells.
- Let me run a python program version of two of the sheets in that spreadsheet with information printed at each epoch.

Gradient Descent

- Good practice:
- Plot iterations of the algorithm (the outer loop) on horizontal axis and loss on vertical axis
- We expect the loss to decrease (since we are trying to minimize the loss) as iterations increase
- We expect the loss to decrease exponentially
- What might the plot look like if learning rate is too high and it diverges?
- What might the plot look like if learning rate is too low and it take a long time to converge?
- What might it look like if it is “just about right”?

Typical learning rates

- If all data is scaled to a small range like -1 to 1
- Some typical learning rates to try would be like:
- 1, 0.1, 0.01, 0.001, 0.0001 etc.
- Can try in between those too once you hone in on a pair where one is too high and the other too low

Linear regression python example

- Let me show you a program I wrote that does linear regression on some other data and displays plots

Linear regression to learn higher order polynomials

- The linear regression algorithm can be used to learn higher order polynomials, not just linear ones.
- Example of how the coefficient of a square term can be learned

Linear Regression

- Let me show you now how to use the linear regression algorithm to learn higher order polynomials (a quadratic to start).
- A quadratic has this form:

$$w_0 + w_1 * x + w_2 * x^2$$

- Differences between this and the linear function we know how to learn are:
 - 3 weights
 - 1 of the weights is multiplied by x squared term

Linear Regression

- A quadratic has this form:

$$w_0 + w_1 * x + w_2 * x^2$$

- We will only have values for x in our dataset. We can create another column of our design matrix to be the values of x squared
- Loss function will be similar, just substitute the quadratic equation for the linear equation
- We will need the partial derivative with respect to each of w_0 w_1 and w_2

Linear Regression

Loss function for learning a quadratic:

$$J = \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2$$
$$\hat{y}^{(i)} = w_0 x_0^{(i)} + w_1 x_1^{(i)} + w_2 x_2^{(i)}$$
$$x_2^{(i)} = (x_1^{(i)})^2$$

We will need the partial derivative with respect to each of w_0 , w_1 and w_2

- Luckily these are similar to the partials we took for the function of only two weights

Partial Derivatives for quadratic MSE

$$\frac{\partial J}{\partial w_0} = \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)}) x_0^{(i)}$$

$$\frac{\partial J}{\partial w_1} = \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)}) x_1^{(i)}$$

$$\frac{\partial J}{\partial w_2} = \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)}) x_2^{(i)}$$

where $x_0^{(i)} = 1$, $x_2^{(i)} = \left(x_1^{(i)}\right)^2$, and $\hat{y}^{(i)} = w_0 x_0^{(i)} + w_1 x_1^{(i)} + w_2 x_2^{(i)}$

Loss function multiplied out

$$\begin{aligned} J &= \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2 \\ &= \frac{1}{2m} \sum_{i=1}^m \left(w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} - y^{(i)} \right)^2 \\ &= \frac{1}{2m} \sum_{i=1}^m \left((\hat{y}^{(i)})^2 - 2\hat{y}^{(i)}y^{(i)} + (y^{(i)})^2 \right) \\ &= \frac{1}{2m} \sum_{i=1}^m \left(\left(w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} \right)^2 - 2y^{(i)} \left(w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} \right) + (y^{(i)})^2 \right) \\ &= \frac{1}{2m} \sum_{i=1}^m \left(\left(w_0^2 + 2w_0w_1x_1^{(i)} + 2w_0w_2x_2^{(i)} + w_1^2x_1^{(i)} + 2w_1w_2x_1^{(i)}x_2^{(i)} + w_2^2 \left(x_2^{(i)} \right)^2 \right) \right. \\ &\quad \left. - 2y^{(i)} \left(w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} \right) + (y^{(i)})^2 \right) \end{aligned}$$

Compute the partial derivatives

- For homework, using the final equation on the last slide, compute the partial derivatives of the loss function with respect to w_0 , w_1 , and w_2 .
- Noting the equation for $\hat{y}$, see if you can get them into the form of the equations as listed on slide 19.

Gradient Descent

- The gradient descent algorithm is this:

Loop until convergence

Loop for all j 's (number of weights)

$$w_j = w_j - \text{learnRate} * \text{partialDerivOfCost w/r/t } w_j$$

- If we are trying learn a quadratic function, we have 3 weights to learn so j goes from 0 to 2
- The three weights (aka parameters) are the constant term, the coefficient of x and the coefficient of x^2

Linear Regression

- Let me show you the original dataset that I used to learn a linear function but let's have it learn a quadratic
- I also made a dataset that has data that is more quadratic in nature and I'll run it on that too

Linear Regression

- Let's take a look at code I will provide for your next assignment.
- Your assignment will be to write code that is missing for linear regression.
- You'll need to understand the shapes and contents of each of the variables to be able to write the missing code.