

CS 331

Computer Vision

10 / 19 / 2026

Instructor: Michael Eckmann

Today's Topics

- Questions / Comments?
- Continue Linear Regression and gradient descent for linear regression

Linear Regression

As the name implies, linear regression solves a regression problem. In other words, the goal is to build a system that can take a vector $\mathbf{x} \in \mathbb{R}^n$ as input and predict the value of a scalar $y \in \mathbb{R}$ as its output. The output of linear regression is a linear function of the input. Let $\hat{y}$ be the value that our model predicts y should take on. We define the output to be

$$\hat{y} = \mathbf{w}^\top \mathbf{x}, \quad (5.3)$$

where $\mathbf{w} \in \mathbb{R}^n$ is a vector of **parameters**.

Parameters are values that control the behavior of the system. In this case, w_i is the coefficient that we multiply by feature x_i before summing up the contributions from all the features. We can think of $\mathbf{w}$ as a set of **weights** that determine how each feature affects the prediction. If a feature x_i receives a positive weight w_i , then increasing the value of that feature increases the value of our prediction $\hat{y}$. If a feature receives a negative weight, then increasing the value of that feature decreases the value of our prediction. If a feature's weight is large in magnitude, then it has a large effect on the prediction. If a feature's weight is zero, it has no effect on the prediction.

Linear Regression

$$\text{MSE}_{\text{test}} = \frac{1}{m} \sum_i (\hat{\mathbf{y}}^{(\text{test})} - \mathbf{y}^{(\text{test})})_i^2. \quad (5.4)$$

Linear Regression

- A linear equation has this form:

$$w_0 + w_1 * x$$

- Given: x and y values of data points.
- Goal: find the equation of a line that fits the data best.
- The best values for the weights will be when we minimize a loss function (MSE).
- To minimize the loss function we want to find the values of the weights such that the gradient of the loss function is 0.
- We will need the partial derivative with respect to each of w_0 and w_1

Linear Regression

- Conventions
- m is the name we give to the number of training examples
- Input variable(s) are named x
- Output variable is y
- $\hat{y}$ is the predicted value of y during training
- w indicates a weight
- $\hat{y}$ is a linear function of w 's and x 's
- vectors are $1 \times n$ matrices, aka column vectors

Gradient Descent

- Gradient Descent for linear regression
 - 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
 - The loss function for this is the MSE over all the m training examples (formula on next slide)
 - 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

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

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.
- If it is set too high, steps may be too large and might diverge
- If set too small, in this case at least, will eventually converge but after a long number of iterations of the outer loop

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)

Example of Linear Regression

- Let me show you a spreadsheet that performs linear regression
- Once we learn the weights we can use them to predict the y values of some other x 's
- We can alter the learning rate to see how it is affected

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”?

Linear regression python example

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