Assignment 2: Classification and Linear Regression

Machine Learning

Fall 2021

Learning Objectives

- Implement a simple classification algorithm in Python
- Learn mathematical tricks for manipulating matrices and vectors.
- Derive the linear regression algorithm
- Implement linear regression in Python

Prior Knowledge Utilized

- Supervised learning problem framing
- Setup for the linear regression problem
- Partial derivatives and gradients, matrix-vector multiplication, and vector-vector multiplication

1 Supervised Learning Framing

In the last assignment, you saw a high level overview of machine learning broken into the categories of supervised, unsupervised, and reinforcement learning (see figure below). Supervised learning can be split into regression (predicting a quantity) and classification (predicting a label). In this assignment, we will look at an example of classification, and then dig deeper into linear regression.

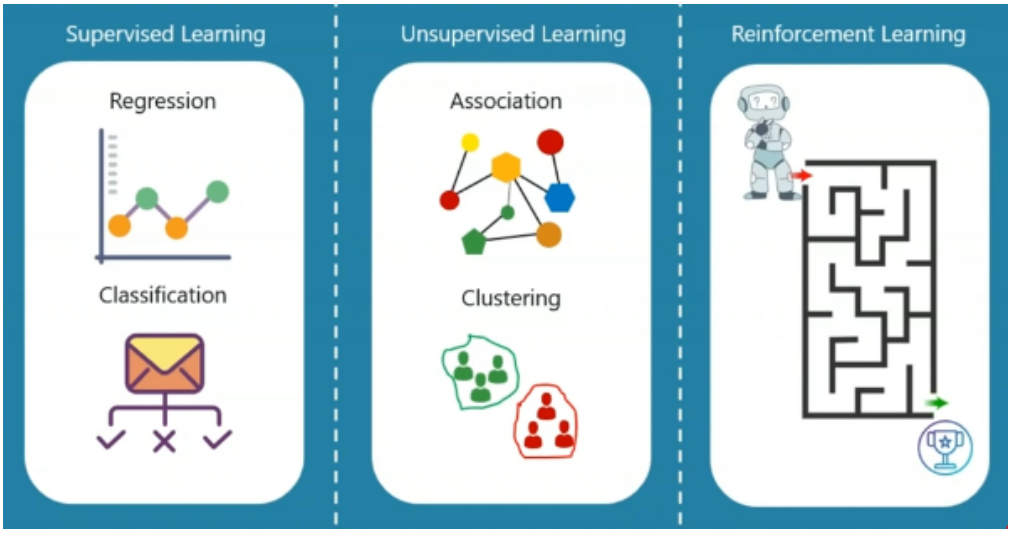

Image from Esma Bozkurt article on medium.com

2 *Implement classification with a train/test split*

As we (hopefully) learned in the class activity, overfitting our model to our data can lead to diminished results when we apply our model to a new set of data. One of the ways we try to avoid overfitting is by splitting our data into a training and testing set. (In the future, we will talk about another split of the training data called cross-validation, but for now, we won't worry about that.)

External Resource(s) (60 minutes)

Work through the Assignment 2a Companion Notebook to practice implementing a simple classification algorithm called a decision tree in Python. You can place your answers directly in the Jupyter notebook so that you have them for your records.

3 *A Quick Refresher on Linear Regression*

In the last assignment you worked with the linear regression (ordinary least-squares) algorithm from a top-down perspective. We focused on what problem the algorithm is trying to solve (minimizing squared error) and then showed some of the algorithm's behavior when applied to a toy problem.

In this part of the assignment we are going to focus on linear regression from the bottom up. We will be working through the mathematics (and eventually the implementation in code) of the solution to the linear regression problem. Before diving in, we thought it would be helpful to include some reminders of the setup for the generic supervised learning problem and the setup for linear regression. If you already feel good about these topics, you can safely skip to the next section.

Recall: Supervised Learning Problem Setup

We are given a training set of data instances, $(\mathbf{x_1}, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$ where each $\mathbf{x_i}$ represents an element of an input space (e.g., a d-dimensional feature vector) and each y_i represents an element of an output space (e.g., a scalar target value). Our goal is to determine a function $\hat{f}$ that maps from the input space to the output space.

We assume there is a loss function, ℓ, that determines the amount of loss that a particular prediction $\hat{y}_i$ incurs due to a mismatch with the actual output y_i. The best possible model, $\hat{f}^\star$, is the one that minimizes these losses over the training set. This notion can be expressed with the following equation.

$$\hat{f}^\star = \arg\min_{\hat{f}} \sum_{i=1}^{n} \ell\left(\hat{f}(\mathbf{x_i}), y_i\right) \tag{1}$$

Recall: The Linear Regression Model

Our input points, $\mathbf{x_i}$, are d-dimensional vectors (each entry of these vectors can be thought of as a feature), our output points, y_i, are scalars, and our prediction functions, $\hat{f}$, are all of the form $\hat{f}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} = \sum_{i=1}^{d} w_i x_i$ for some weights $\mathbf{w}$.

In the function, $\hat{f}$, the elements of the vector $\mathbf{w}$ represent weights that multiply various entries of the input. For

instance, if an element of $\mathbf{w}$ is high, that means that as the corresponding element of $\mathbf{x}$ increases, the prediction that $\hat{f}$ generates for $\mathbf{x}$ would also increase. The products of the weights and the features are then summed to arrive at an overall prediction.

Given this model, we can now define the ordinary least squares (OLS) algorithm! In the ordinary least squares algorithm, we use our training set to select the $\mathbf{w}$ that minimizes the sum of squared differences between the model's predictions and the training outputs. This corresponds to choosing $\ell(y, \hat{y}) = (y - \hat{y})^2$.

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} \ell\left(\hat{f}(\mathbf{x_i}, \mathbf{w}), y_i\right) \tag{2}$$

$$= \arg\min_{\mathbf{w}} \sum_{i=1}^{n} \left(\hat{f}(\mathbf{x_i}, \mathbf{w}) - y_i\right)^2 \tag{3}$$

$$= \arg\min_{\mathbf{w}} \sum_{i=1}^{n} \left(\mathbf{w}^\top \mathbf{x_i} - y_i\right)^2 \tag{4}$$

Once we have $\mathbf{w}^\star$, we can predict the unknown output, y, for a new input point, $\mathbf{x}$, as $\hat{y} = \mathbf{w}^{\star\top}\mathbf{x}$.

4 *Linear Regression from the Bottom-up*

Now that we've refreshed ourselves on the basic framing of linear regression, we'll be diving into the mathematics of how to find the vector $\mathbf{w}^\star$ that best fits a particular training set. The outline of the steps we are going to take to learn this are:

1. Solve the special case of linear regression with a single input ($d = 1$, meaning a 1-dimensional feature vector).
2. Learn some mathematical tricks for manipulating matrices and vectors and computing gradients of functions involving matrices and vectors (these will be useful for solving the general case of linear regression).
3. Solve the general case of linear regression (where d can be any positive, integer value).

4.1 *Linear regression with one variable*

Exercise 1 (20 minutes)

(a) Given a dataset $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ (where each x_i and each y_i is a scalar) and a potential value of w (note that w is a scalar in the case where $d = 1$), write an expression for the sum of squared errors between the model predictions, $\hat{f}$, and the targets, y_i. **Note:** In contrast to the line of best fit we saw in the last assignment, here we are not computing a y-intercept (effectively forcing the y-intercept to be 0). This choice may result in a worse fit, but it is easier to work out and helps build mathematical intuition.

(b) Compute the derivative of the expression for the sum of squared errors from part (a).

(c) Set the derivative to 0, and solve for $w^\star$. $w^\star$ corresponds to a critical point of your sum of squared errors function. Is this critical point a minimum, maximum, or neither? (here is a refresher on classifying critical points).

4.2 Reminder of mathematical ideas from the last assignment

In the previous assignment, we asked you to solidify your knowledge of three different mathematical concepts. The box below summarizes what you were supposed to learn and provides the resources we provided to help you.

🔁 Recall: Mathematical background

In order to engage with this assignment, you'll want to make sure you are familiar with the following concepts (links to resources embedded below):

- vector-vector multiplication
  - Section 2.1 of Zico Kolter's Linear Algebra Review and Reference
- Matrix-vector multiplication
  - Section 2.2 of Zico Kolter's Linear Algebra Review and Reference
  - The first bits of the Khan academy video on Linear Transformations
- partial derivatives and gradients
  - Khan Academy videos on partial derivatives: intro, graphical understanding, and formal definition
  - Khan Academy video on Gradient

4.3 Building our bag of mathematical tricks

The derivation of linear regression for the single variable case made use of your background from single variable calculus, and you used some rules for manipulating such functions. When approaching linear regression with multiple variables, you have two choices.

1. You can apply the same bag of tricks you used for the single variable problem and only at the end convert things (necessarily) to a multivariable representation.
2. You can approach the whole problem from a multivariable perspective.

This second approach requires that you learn some additional mathematical tricks, but once you learn these tricks, the derivation of linear regression is very straightforward. The secondary benefit of this approach is that the new mathematical tricks you learn will apply to all sorts of other problems.

Exercise 2 (15 minutes)

A quadratic form can be expressed in matrix-vector form as $\mathbf{x}^\top \mathbf{A}\mathbf{x}$. Written this way, it looks very mysterious, but in this exercise you'll build some intuition about what the expression represents. Further, it turns out that expressions like this show up in all sorts of places in machine learning. To get a better understanding of what a quadratic form *is* (we'll see what it's good for later), watch this Khan Academy video.
After you've watched the Khan academy video, answer these questions.

(a) Multiply out the expression $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}^\top \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$.

(b) Complete the following expression by filling in the part on the righthand side inside the nested summation.

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}^\top \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,d} \\ a_{2,1} & a_{2,2} & \dots & a_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ a_{d,1} & a_{d,2} & \dots & a_{d,d} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} = \sum_{i=1}^{d} \sum_{j=1}^{d} \text{(your answer here)}$$

Spoiler alert: Exercise 4 provides you with this result (or, as always, you can check the solutions).

Exercise 3 (5 minutes)

Matrix multiplication distributes over addition. That is, $(\mathbf{A}+\mathbf{B})(\mathbf{C}+\mathbf{D}) = \mathbf{AC}+\mathbf{AD}+\mathbf{BC}+\mathbf{BD}$. Use this fact coupled with the fact that $(\mathbf{AB})^\top = \mathbf{B}^\top\mathbf{A}^\top$ to expand out the following expression.

$$(\mathbf{Ax}+\mathbf{y})^\top(\mathbf{v}+\mathbf{u})$$

Exercise 4 (25 minutes)

(a) Using the definition of the gradient, show that $\nabla \mathbf{c}^\top \mathbf{x} = \mathbf{c}$ where the gradient is taken with respect to $\mathbf{x}$ and $\mathbf{c}$ is a vector of constants.

(b) Using the definition of the gradient, show that the $\nabla \mathbf{x}^\top \mathbf{A}\mathbf{x} = 2\mathbf{A}\mathbf{x}$ where the gradient is taken with respect to $\mathbf{x}$ and $\mathbf{A}$ is a *symmetric dxd* matrix of constants. Hint: utilize the fact that $\mathbf{x}^\top \mathbf{A}\mathbf{x} = \sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j a_{i,j}$.

5 *Linear Regression with Multiple Variables*

Exercise 5 (40 minutes)

Consider the case where $\mathbf{w}$ is a d-dimensional vector. In this case, it is convenient to represent our n training inputs as an $n \times d$ matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix}$ and our n training outputs as an n-dimensional vector $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$.

In order to solve this problem, you'll be leveraging some of the new mathematical tricks you picked up early in this assignment. As you go through the derivation, make sure to treat vectors as first-class objects (e.g., work with the gradient instead of the individual partial derivatives).

(a) Given $\mathbf{w}$, write an expression for the vector of predictions $\hat{\mathbf{y}} = \begin{bmatrix} \hat{f}(\mathbf{x}_1) \\ \hat{f}(\mathbf{x}_2) \\ \vdots \\ \hat{f}(\mathbf{x}_n) \end{bmatrix}$ in terms of the training input matrix $\mathbf{X}$ (Hint: you should come up with something very simple).

(b) Write an expression for the sum of squared errors for the vector $\mathbf{w}$ on the training set in terms of $\mathbf{X}$, $\mathbf{y}$, and $\mathbf{w}$. Hint: you will want to use the fact that $\sum_{i=1} v_i^2 = \mathbf{v} \cdot \mathbf{v} = \mathbf{v}^\top \mathbf{v}$. Simplify your expression by distributing matrix multiplication over addition (don't leave terms such as $(\mathbf{u} + \mathbf{v})(\mathbf{d} + \mathbf{c})$ in your answer).

(c) Compute the gradient of the sum of squared errors that you found in part (b) with respect to $\mathbf{w}$. Make sure to use the results from the previous exercises to compute the gradients.

(d) Set the gradient to 0, and solve for $\mathbf{w}$ (note: you can assume that $\mathbf{X}^\top \mathbf{X}$ is invertible). This value of $\mathbf{w}$ corresponds to a critical point of your sum of squared errors function. We will show in a later assignment that this critical point corresponds to a global minimum. In other words, this value of $\mathbf{w}$ is guaranteed to drive the sum of squared errors as low as possible.

6 Implementation

Hopefully that wasn't too painful and you learned a few useful tricks along the way. The good news is that even complex machine learning techniques wind up boiling down to following recipes that look very much like what you just did.

External Resource(s) (100 minutes)

At this point, you'll be taking the derivation of linear regression that you just worked out and using it to implement the linear regression algorithm from scratch in Python (well, you will be using `numpy`). Along the way we'll show you some sanity checks that you can perform to verify that your implementation is correct and help you to solidify your understanding of the math. Without further ado, let's transition over to the Assignment 2b Companion Notebook.