Assignment 4: Logistic Regression and Gradient Descent

Machine Learning

Fall 2021

Learning Objectives

- Learn about the logistic regression algorithm.
- Learn about gradient descent for optimization.
- Contemplate an application of machine learning for home loans.

Prior Knowledge Utilized

- Supervised learning problem framing.
- Calculating gradients.
- Log loss.

1 Logistic Regression (top-down)

In the last part of the notebook that you started in class, you saw a quick implementation of logistic regression to classify if a person was looking to the left or to the right.

In this assignment we will formalize the binary classification problem and dig the theory behind *logistic regression*. You will also see that the logistic regression algorithm is a very natural extension of linear regression. Our plan for getting there is going to be pretty similar to what we did for linear regression.

- Build some mathematical foundations
- Introduce logistic regression from a top-down perspective
- Learn about logistic regression from a bottom-up perspective

Recall:

In the last assignment, you were introduced to the idea of binary classification, which based on some input $\mathbf{x}$ has a corresponding output y that is $y = 0$ or $y = 1$. In logistic regression, this model, $\hat{f}$, instead of spitting out either 0 or 1, outputs a confidence that the input $\mathbf{x}$ has an output $y = 1$. In other words, rather than giving us its best guess (0 or 1), the classifier indicates to us its degree of certainty regarding its prediction as a probability.

We also explored three possible loss functions for a model that outputs a probability p when supplied with an input $\mathbf{x}$ (i.e., $\hat{f}(\mathbf{x}) = p$). The loss function is used to quantify how bad a prediction p is given the actual output y (for binary classification the output is either 0 or 1).

1. **0-1 loss:** This is an all-or-nothing approach. If the prediction is correct, the loss is zero; if the prediction is incorrect, the loss is 1. This does not take into account the level certainty expressed by the probability (the model gets the same loss if $y = 1$ and it predicted $p = 0.51$ or $p = 1$).

2. **squared loss:** For squared loss we compute the difference between the outcome and p and square it to arrive at the loss. For example, if $y = 1$ and the model predicts $p = 0.51$, the loss is $(1 - 0.51)^2$. If instead $y = 0$, the loss is $(0 - 0.51)^2$.

3. **log loss:** The log loss also penalizes based on the difference between the outcome and p, using the formula below.

$$logloss = -\frac{1}{N}\sum_{i=1}^{n}((y_i)\ln(p_i) + (1 - y_i)\ln(1 - p_i)) \tag{1}$$

Since y_i is always 0 or 1, we will essentially switch between the two chunks of this equation based on the true value of y_i. As the predicted probability, p_i (which is constrained between 0 an 1) gets farther from y_i, the log-loss value increases.

Now that you have refreshed on how probabilities can be used as a way of quantifying confidence in predictions, you are ready to learn about the logistic regression algorithm.

As always, we assume we are given a training set of inputs and outputs. As in linear regression we will assume that each of our inputs is a d-dimensional vector $\mathbf{x_i}$ and since we are dealing with binary classification, the outputs, y_i, will be binary numbers (indicating whether the input belongs to class 0 or 1). Our hypothesis functions, $\hat{f}$, output the probability that a given input has an output of 1. What's cool is that we can borrow a lot of what we did in the last couple of assignments when we learned about linear regression. In fact, all we're going to do in order to make sure that the output of $\hat{f}$ is between 0 and 1 is pass $\mathbf{w}^\top\mathbf{x}$ through a function that "squashes" its input so that it outputs a value between 0 and 1. This idea is shown graphically in Figure 1.

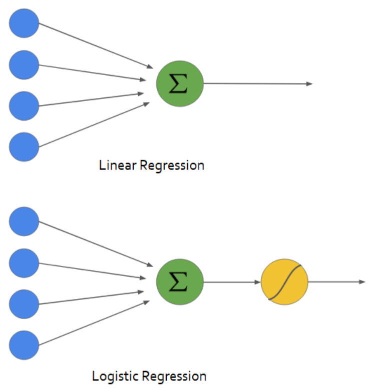


Figure 1: Graphical representation of both linear and logistic regression. The key difference is the application of the squashing function shown in yellow. Original source.

To make this intuition concrete, we define each $\hat{f}$ as having the following form (note: this equation looks daunting. We have some tips for interpreting it below).

$$\begin{aligned}\hat{f}(\mathbf{x}) &= \text{probability that output, } y\text{, is 1} \\ &= \frac{1}{1 + e^{-\mathbf{w}^\top\mathbf{x}}}\end{aligned} \tag{2}$$

Here are a few things to notice about this equation:

1. The weight vector that we saw in linear regression, $\mathbf{w}$, has made a comeback. We are using the dot product between $\mathbf{x}$ and $\mathbf{w}$ (which creates a weighted sum of the x_i's), just as we did in linear regression!

2. As indicated in Figure 1, the dot product $\mathbf{w}^\top \mathbf{x}$ has been passed through a squashing function known as the sigmoid function. The graph of $\sigma(u) = \frac{1}{1+e^{-u}}$ is shown in Figure 2. $\sigma(\mathbf{w}^\top \mathbf{x})$ is exactly what we have in Equation 2.

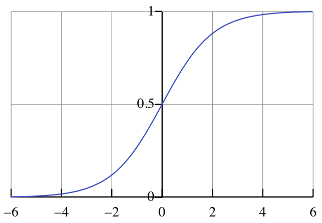


Figure 2: a graph of the sigmoid function $\frac{1}{1+e^{-x}}$.

2 Deriving the Logistic Regression Learning Rule

Now we will formalize the logistic regression problem and derive a learning rule to solve it (i.e., compute the optimal weights). The formalization of logistic regression will combine Equation 2 with the selection of ℓ to be log loss (Equation 1). This choice of ℓ results in the following objective function.

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} \left(-y_i \ln \sigma(\mathbf{w}^\top \mathbf{x_i}) - (1-y_i)\ln(1-\sigma(\mathbf{w}^\top \mathbf{x_i}))\right) \tag{3}$$

$$= \arg\min_{\mathbf{w}} \sum_{i=1}^{n} \left(-y_i \ln \left(\frac{1}{1+e^{-\mathbf{w}^\top \mathbf{x_i}}} \right) - (1-y_i) \ln \left(1 - \frac{1}{1+e^{-\mathbf{w}^\top \mathbf{x_i}}} \right) \right) \quad \text{expanded out if you prefer this form} \tag{4}$$

While this looks a bit intense, since y_i is either 0 or 1, the multiplication of the expressions in the summation by either y_i or $1 - y_i$ are essentially acting like a switch—depending on the value of y_i we either get one term or the other. Our typical recipe for finding $\mathbf{w}^\star$ has been to take the gradient of the expression inside the arg min, set it to 0, and solve for $\mathbf{w}^\star$ (which will be a critical point and hopefully a minimum). The last two steps will be a bit different for reasons that will become clear soon, but we will need to find the gradient. We will focus on finding the gradient in the next couple of parts.

2.1 Useful Properties of the Sigmoid Function

Looking at Equation 4 it looks really, really hairy! We see that in order to compute the gradient we will have to compute the gradient of $\mathbf{x}^\top \mathbf{w}$ with respect to $\mathbf{w}$ (we just wrapped our minds around this last assignment). Additionally, we will have to take into account how the application of the sigmoid function and the log function changes this gradient. In this section we'll learn some properties for manipulating the sigmoid function and computing its derivative.

Exercise 1 (60 minutes)

The sigmoid function, σ, is defined as

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad . \tag{5}$$

(a) Show that $\sigma(-x) = 1 - \sigma(x)$.

(b) Show that the derivative of the logistic function $\frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$

2.2 Chain Rule for Gradients

We now know how to take derivatives of each of the major pieces of Equation 4. What we need is a way to put these derivatives together. You probably remember that in the case of single variable calculus you have just such a tool. This tool is known as the chain rule. The chain rule tells us how to compute the derivative of the composition of two single variable functions f and g.

$$\begin{aligned} h(x) &= g(f(x)) && \text{h(x) is the composition of } f \text{ with } g \\ h'(x) &= g'(f(x))f'(x) && \text{this is the chain rule!} \end{aligned} \tag{6}$$

Suppose that instead of the input being a scalar x, the input is now a vector, $\mathbf{w}$. In this case h takes a vector input and returns a scalar, f takes a vector input and returns a scalar, and g takes a scalar input and returns a scalar.

$$\begin{aligned} h(\mathbf{w}) &= g(f(\mathbf{w})) && \text{h}(\mathbf{w}) \text{ is the composition of } f \text{ with } g \\ \nabla h(\mathbf{w}) &= g'(f(\mathbf{w}))\nabla f(\mathbf{w}) && \text{this is the multivariable chain rule} \end{aligned} \tag{7}$$

Exercise 2 (60 minutes)

(a) Suppose $h(x) = \sin(x^2)$, compute $h'(x)$ (x is a scalar so you can apply the single-variable chain rule).

(b) Define $h(\mathbf{v}) = (\mathbf{c}^\top \mathbf{v})^2$. Compute $\nabla_{\mathbf{v}} h(\mathbf{v})$ (the gradient of the function with respect to $\mathbf{v}$).

(c) Compute the gradient of the expression from Equation 4 (reproduced below for your convenience).

$$\sum_{i=1}^{n} -y_i \ln \sigma(\mathbf{w}^\top \mathbf{x_i}) - (1-y_i)\ln\left(1-\sigma(\mathbf{w}^\top \mathbf{x_i})\right) \ . \tag{8}$$

You can either use the chain rule and the identities you learned about sigmoid, or expand everything out and work from that.

2.3 Gradient Descent for Optimization

If we were to follow our derivation of linear regression we would set our expression for the gradient to 0 and solve for $\mathbf{w}$. It turns out this equation will be difficult to solve due to the σ function. Instead, we can use an iterative approach where we start with some initial value for $\mathbf{w}$ (we'll call the initial value $\mathbf{w^0}$, where the superscript corresponds to the iteration number) and iteratively adjust it by moving down the gradient (the gradient represents the direction of fastest increase for our function, therefore, moving along the negative gradient is the direction where the loss is decreasing the fastest).

External Resource(s) (45 minutes)

There are tons of great resources that explain gradient descent with both math and compelling visuals.

- Recommended: Gradient descent, how neural networks learn | Deep learning, chapter 2 (start at 5:20)
- An Introduction to Gradient Descent
- The Wikipedia page on Gradient Descent
- Ahmet Sacan's video on gradient descent (this one has some extra stuff, but it's pretty clearly explained).
- There are quite a few resources out there, do you have some suggestions? (suggest so on Discord)

Exercise 3 (10 minutes)

To test your understanding of these resources, here are a few diagnostic questions.

(a) When minimizing a function with gradient descent, which direction should you step along in order to arrive at the next value for your parameters?

(b) What is the learning rate and what role does it serve in gradient descent?

(c) How do you know when an optimization performed using gradient descent has converged?

(d) True or false: provided you tune the learning rate properly, gradient descent guarantees that you will find the global minimum of a function.

If we take the logic of gradient descent and apply it to the logistic regression problem, we arrive at the following learning rule. Given some initial weights $\mathbf{w^0}$, and a learning rate η, we can iteratively update our weights using the formula below.

$$\mathbf{w^{n+1}} = \mathbf{w^n} - \eta \sum_{i=1}^{n} -(y_i - \sigma(\mathbf{w}^\top \mathbf{x_i}))\mathbf{x_i} \quad \text{applying the result from exercise 2} \tag{9}$$

$$= \mathbf{w^n} + \eta \sum_{i=1}^{n} (y_i - \sigma(\mathbf{w}^\top \mathbf{x_i}))\mathbf{x_i} \quad \text{distribute the negative} \tag{10}$$

This beautiful equation turns out to be the recipe for logistic regression.

Notice

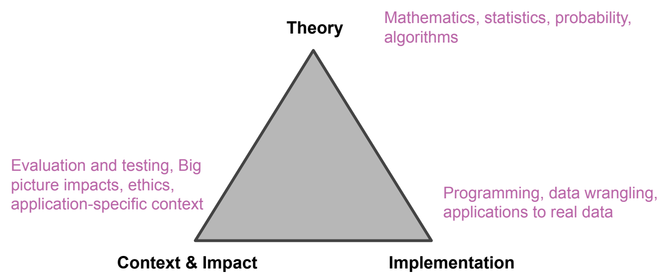
We won't be assigning a full implementation of logistic regression from scratch. In future assignments, we will spend more time applying logistic regression and gradient descent.

If it's helpful for your learning to see a worked example with code now (to help the math make sense), you can optionally check out this example of binary classification for admission to college, noting that some of the math notation is slightly different than ours.

You are also welcome to implement logistic regression using gradient descent if it's helpful for your learning and/or if you already have significant experience with machine learning and want a challenge. This is completely optional, and we assume that most of you will not choose to do this. If you do decide to implement logistic regression using gradient descent, you will need to search for a good learning rate or you may consider implementing some strategies for automatically tuning the learning rate.

3 *Touchpoint to context, impact, and ethics*

As we mentioned in the introduction to the course, we'll be exploring machine learning from three different perspectives: the theory, the implementation, and the context, impact, and ethics.


Theory
Mathematics, statistics, probability, algorithms
Evaluation and testing, Big picture impacts, ethics, application-specific context
Programming, data wrangling, applications to real data
Context & Impact
Implementation

5

Lately, we've been diving deeply into the mathematical theory with some touch points in implementation and context. Here, we want to take a moment to zoom out (with a lowercase z), and think about how algorithms interact with society, individuals, and institutions.

Machine learning governs how people can access wealth-building tools, like loans and mortgages. Please read this article on algorithms preventing bias in home loans. Then watch this TED talk about a way to build credit history.

Exercise 4

Consider the following questions.

1. Would you consider the NY Times story objective? What about the TED talk?
2. Consider the pros and cons of using Tala, Shivani's company. What are the users gaining? What are they giving up? Is what they give up different than

what users of a credit report provide?

3. Here's Tala's Statement on Data Ethics. How does it compare to the FAT-ML framework we looked at on Day 1?