Assignment 3: Classification, Logistic Regression, and Gradient Descent

Machine Learning

Fall 2019

Learning Objectives

- Learn about the framing of the classification problem in machine learning.
- Learn about the logistic regression algorithm.
- Learn about gradient descent for optimization.
- Some C&E topic.

Prior Knowledge Utilized

- Supervised learning problem framing.
- Training / testing splits.

Recall: Supervised Learning Problem Setup

We are given a training set, $(\mathbf{x_1}, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$ 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.

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

1 The Classification Problem

So far in this class we've looked at supervised learning problems where the responses y_i are continuous-valued and the loss function is quadratic ($\ell(y, \hat{y}) = (y - \hat{y})^2$). This is an example of a regression problem. There are many times, however, where it is unnatural to frame a problem as a regression. For instance, it may be the case that y_i does not come from a continuous range but can only take on a few different values. This sort of problem is known as a classification problem. For instance, you might want to have a system that can take in an image of a person and predict their identity. The identity

could be thought of as the output, y_i, and it could only take on one of several values (each value might represent a particular person the system was trained to recognize). In this assignment you'll learn about a special case of the classification problem known as binary classification (where y_i is either 0 or 1, e.g., a Paul versus Sam detector).

In this assignment will formalize the binary classification problem and see a very useful algorithm for solving it called *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

2 Formalizing the Classification Problem

Let's start by making the binary classification problem more formal. Suppose, we are given a training set, $(\mathbf{x_1}, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$, where each $\mathbf{x_i}$ is an element of the input space (e.g., a vector) and each y_i is a binary number (either 1 or 0). In this setting we will attempt to use the training data to determine a function, $\hat{f}^\star$, that predicts the corresponding output, y, for any possible input, $\mathbf{x}$. To make this concrete with an example, $\mathbf{x}$ could be an image and y could be a binary number that takes on value 1 when the picture contains a puppy.

Exercise 1 (10 minutes)

(a) Given this setup of the binary classification problem, the only thing left to specify is the loss function, ℓ. Recall that ℓ takes as input the actual output y, and the predicted output $\hat{y}$. What function could you use for ℓ that would result in the learning algorithm choosing a good model. If the choice of ℓ depends on the application, how so?

☆ Solution

An easy choice is to output a 1 if the values don't match and a 0 otherwise (essentially counting the number of mistakes the model makes). Alternatively, you could have different penalties for a false positive (the model says $\hat{y} = 1$, but the actual value is $y = 0$) or false negatives (the model says $\hat{y} = 0$, but the actual value is $y = 1$).

(b) One natural choice, that you may have already come up with in the previous question, is to define our loss function as $\ell(y, \hat{y}) = \mathbb{I}[y \neq \hat{y}]$ (the funny looking $\mathbb{I}$ is the indicator function that takes on value 1 when the condition inside is true

and 0 otherwise. Given these choices, the supervised learning problem becomes:

$$\hat{f}^{\star} = \arg\min_{\hat{f}} \sum_{i=1}^{n} \mathbb{I}\left[\hat{f}(\mathbf{x_i}) \neq, y_i\right] \quad . \tag{2}$$

Convert Equation 2 to English to make sure you understand it.

☆ Solution

The equation says that $\hat{f}^{\star}$ is the function $\hat{f}^{\star}$ that minimizes the number of mistakes that the function makes on the training set.

The loss function given in Exercise 2(b) is a totally reasonable choice since it will result in choosing the model that makes the fewest mistakes on the training. It turns out, however, that it has the following drawbacks.

- It is all or nothing. Either we are completely right or completely wrong.
- It is not a particularly easy function to work with mathematically. In fact, for many common classes of models, it will be difficult for the learning algorithm to find the best possible model[1].

[1] One of the key challenges that must be in met in machine learning, and modeling in general, is balancing computational considerations (e.g., how long does it take to find the best possible model) with the realism of the model (e.g., how directly does the task you pose to the learning algorithm match the problem you are solving. Sometimes these things are in conflict and you must make tradeoffs.

It turns out that we can create a much more natural loss function by thinking about predictions in terms of probabilities.

3 Probability and the log loss

Imagine that instead of our model, $\hat{f}$, spitting out either 0 or 1, it outputs a confidence that the input x_i has an output, y_i, that was 1. In other words, rather than giving us its best guess, the classifier would indicate to us its degree of certainty regarding its prediction. This notion of "certainty" can be formalized using the concept of a probability. That is, the model can output a probability that the output for a particular input is 1.

We haven't formally defined probability in this class, and we won't do so here (we'll be working with probabilities extensively in module 2 and we don't need a formal treatment to derive the classification algorithms we will study in this module). Here are a few things to keep a few things in mind about probabilities.

- A probability, p, specifies the chance that some event occurs ($p = 0$ means that the even will definitely not occur and $p = 1$ means that it will definitely occur.
- A probability, p, must be between 0 and 1 ($0 \leq p \leq 1$).
- If the probability an event occurs is p, then the probability that the event doesn't occur is $1 - p$.

✔ Understanding Check

Note: these sorts of boxes are here to help you test your understanding of a concept you have just read. We are trying to use these to breakup long blocks of reading. You do not need to submit these.

TODO: Quick check of understanding

3.1 Log loss

Thinking back to the binary classification problem, if we think in terms of probability, then our model output a probability p when supplied with an input $\mathbf{x}$ (i.e., $\hat{f}(\mathbf{x}) = p$). We might then ask ourselves the question of what would be a reasonable loss function to quantify how good a prediction p is given the actual output y (recall that for the binary classification the output is either 0 or 1). To make to this more intuitive, consider the task of quantifying the quality of a weatherperson's predictions. Further, let's assume that on any given day the weather is either sunny (call this output 1) or rainy (call this output 0). Suppose that each night the weather person provides a probability that indicates the chance of it being sunny the next day. In order to quantify the loss of each prediction we need to define a loss function. Here are two potential choices.

1. **0-1 loss:** we will extract from the weatherperson's prediction the most likely outcome (e.g., if $p = 0.75$, that would be sunny, if $p = 0.4$, that would be rainy). If this most likely outcome matches the actual outcome we give a loss of 0, otherwise we give a loss of 1 (this is similar to Equation 2).

2. **squared loss:** one downside of *0-1 loss* is that it doesn't take into account the certainty expressed by the weatherperson. The weatherperson gets the same loss if it is rainy and they predicted $p = 0.51$ or $p = 1$. For squared loss we can compute the difference between the outcome and p and square it to arrive at the loss. So if we predict $p = 0.51$ and it is sunny we get a loss of $(1 - 0.51)^2$. If it was rainy in this same example, we get a loss of $(0 - 0.51)^2$.

As an example, here are hypothetical predictions from two forecasters, the actual weather, and the resulting loss with either *0-1* loss or *squared loss*.

actual weather	forecast 1	0-1 loss	squared loss	forecast 2 1	0-1 loss	squared loss
sunny (y = 1)	$p = 0.3$	1	$(1 - 0.2)^2 = 0.64$	$p = 0.9$	0	$(1 - 0.9)^2 = 0.01$
rainy (y = 0)	$p = 0.6$	1	$(0 - 0.6)^2 = 0.36$	$p = 0.999$	1	$(0 - 0.999)^2 = 0.998$
sunny (y = 1)	$p = 0.8$	0	$(1 - 0.8)^2 = 0.16$	$p = 0.99$	0	$(1 - 0.99)^2 = 0.0001$
average		0.667	0.347		0.333	0.336

✔ Understanding Check

According to the table above, which forecaster is better with regards to *0-1 loss*? Which forecaster is better with regards to *squared loss*?

One entry in the table above is particularly interesting. In the third row, the second forecaster assigned a probability of 0.999 to the fact that it was going to be sunny. It turned out to rain (boo!!!). That is, the forecaster was almost certain it would be sunny and it wasn't. The 0-1 loss of course doesn't capture this at all. The squared loss seems to assign a fairly large loss. One might argue, though, that this loss does not fully capture how bad the prediction was. This last observation motivates a third loss function that we can use to evaluate probabilistic predictions: the log loss.

External Resource(s)

`wiki.fast.ai` has some really nice resources on a number of topics. They have a nice concise writeup that explains the concept of log loss. We ask that you read about log loss on NB so you can take advantage of some notes from us to help guide your reading and add your own notes as well. If you want the original page (e.g., to click on the links), you can access the log loss page on wiki.fast.ai.

Exercise 2

TODO Here are some questions to test your understanding of the log loss reading.

4 Logistic Regression (top-down)

Now that we have built up some understanding of 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 each be binary numbers (indicating whether the corresponding input is of class 0 or 1). Our hypothesis functions, $\hat{f}$, output the probability that a given input has a corresponding output of 1. Each $\hat{f}$ has the following form (note: this equation will look daunting, so we have some tips for interpreting it below).

$$\begin{aligned}\hat{f}(\mathbf{x}) &= \text{probability that } y \text{ is 1 for } \mathbf{x} \\ &= \frac{1}{1+e^{-\mathbf{w}^\top \mathbf{x}}}\end{aligned} \tag{3}$$

Here are a few things to notice about this equation:

1. The weight vector that we saw in linear regression, $\mathbf{w}$, is making a comeback! In fact, we have a 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. Despite the similarities with linear regression, we now have this super weird term involving $\frac{1}{1+e^{-\text{stuff}}}$. It turns out that this function is known as the logistic function, σ.

The graph of $\sigma(u) = \frac{1}{1+e^{-u}}$ is shown in Figure 1. The function acts to map any real number to a number between 0 and 1. Thus it is a perfect "squashing" function to take numbers that might not necessarily be valid probabilities (e.g., $\mathbf{w}^\top \mathbf{x}$) and squash them to the range of valid probabilities $[0, 1]$.

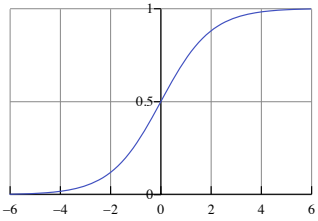

Figure 1: fasf

As motivating example, consider what would happen if we were faced with learning the following task.

The dataset can be found at the UCI Machine Learning repository. was presented in the paper Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models

2 | 2

TODO: state the objective function.

4.1 Example

Todo: link to notebook

5 Gradient Descent

TODO: Link out to a resource here.

6 Chain Rule for Gradients

It turns out that when we are fitting the parameters of our logistic regression model we are going to use the technique of gradient descent in order to compute the optimal weights. Looking back at Equation REFTOOBJECTIVE, we see that in order to compute a gradient we will have to compute the gradient of $\mathbf{x}^\top \mathbf{w}$ with respect to $\mathbf{w}$ (which we know how to do from the last assignment). Additionally, we will have to understand how the application of the sigmoid function and the log function change this gradient. In this section we'll learn some rules for easily computing gradients in such situations.

In the past assignment you dusted off some old tricks from single variable calculus. You may have applied the chain rule for derivatives. The chain rule tells us how to compute the derivative of the composition of two single variable functions f and g.

$$\begin{aligned} h(x) &= g(f(x)) && \text{h(x) is the composition of } f \text{ with } g \\ h'(x) &= g'(f(x))f'(x) && \text{this is the chain rule!} \end{aligned} \tag{4}$$

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.

$$\begin{aligned} h(\mathbf{w}) &= g(f(\mathbf{w})) && \text{h}(\mathbf{w}) \text{ is the composition of } f \text{ with } g \\ \nabla_{\mathbf{w}} h(\mathbf{w}) &= g'(f(\mathbf{w}))\nabla_w f(\mathbf{w}) && \text{this is the multivariable chain rule} \end{aligned} \tag{5}$$

Exercise 3

(a) Compute the gradient of $h(\mathbf{v}) = (\mathbf{c}^\top \mathbf{v})^2$.

☆ Solution

We can see that $h(\mathbf{v}) = g(f(\mathbf{v}))$ with $g(x) = x^2$ and $f(\mathbf{v}) = \mathbf{c}^\top \mathbf{v}$ The gradient can now easily be found by applying the chain rule.

$$\nabla h(\mathbf{v}) = 2(\mathbf{c}^\top \mathbf{v})\mathbf{c} \tag{6}$$

(b) This should be punted until the known the derivative of sigmoid function: Compute the gradient of $h(\mathbf{w}) = \frac{1}{1+e^{-\mathbf{w}^\top \mathbf{x_i}}}$ (hint: this is $\sigma(\mathbf{w}^\top \mathbf{x_i})$).

☆ Solution

We can see that $h(\mathbf{w}) = g(f(\mathbf{w}))$ with $g(x) = \sigma(x)$ and $f(\mathbf{w}) = \mathbf{w}^\top \mathbf{x_i}$ The gradient can now easily be found by applying the chain rule.

$$\nabla h(\mathbf{v}) = \sigma(\mathbf{w}^\top \mathbf{x_i})(1 - \sigma(\mathbf{w}^\top \mathbf{x_i}))\mathbf{x_i} \tag{7}$$

7 Deriving the Learning Rule for Logistic Regression

7.1 Useful Properties of the Sigmoid Function

The sigmoid function[2] known as a logistic function) turns out to be very useful for modeling the probability that some event occurs. TODO.

[2] the sigmoid function is a special case of the logistic function. It is common to see it referred to in be either name

Exercise 4

In this exercise you will be working to better understand some of the properties of the logistic function. Remember, the logistic function, σ, is defined as:

$$\sigma(x) = \frac{1}{1+e^{-x}} \ . \tag{8}$$

(a) Do some thought exercises on the logistic function. Limiting cases, etc. TODO.

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

☆ Solution

$$\sigma(-x) = \frac{1}{1+e^{x}} \tag{9}$$
$$= \frac{e^{-x}}{e^{-x}+1} \quad \text{multiply by top and bottom by } e^{-x} \tag{10}$$
$$\sigma(-x) - 1 = \frac{e^{-x}}{e^{-x}+1} - \frac{1+e^{-x}}{1+e^{-x}} \quad \text{subtract } -1 \text{ on both sides} \tag{11}$$
$$= \frac{-1}{1+e^{-x}} \tag{12}$$
$$= -\sigma(x) \tag{13}$$
$$\sigma(-x) = 1 - \sigma(x) \tag{14}$$

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

☆ Solution

Two solutions for the price of 1!

Solution 1:

$$\frac{d}{dx}\sigma(x) = -e^{-x}\sigma(x)^2 \quad \text{apply quotient rule} \tag{15}$$
$$= \sigma(x)\left(\frac{-e^{-x}}{1+e^{-x}}\right) \quad \text{expand out one of the } \sigma(x)\text{'s} \tag{16}$$
$$= \sigma(x)\left(\frac{-1}{e^{x}+1}\right) \quad \text{multiply top and bottom by } e^{x} \tag{17}$$
$$= \sigma(x)(-\sigma(-x)) \quad \text{substitute for } \sigma(-x) \tag{18}$$
$$= \sigma(x)(\sigma(x)-1) \quad \text{apply } \sigma(-x) = 1-\sigma(x) \tag{19}$$

Solution 2:

$$\frac{d}{dx}\sigma(x) = \frac{-e^{-x}}{(1+e^{-x})^2} \quad \text{apply quotient rule} \tag{20}$$
$$= \frac{-e^{-x}}{1+2e^{-x}+e^{-2x}} \quad \text{expand the bottom} \tag{21}$$
$$= \frac{-1}{e^{x}+2+e^{-x}} \quad \text{multiply top and bottom by } e^{x} \tag{22}$$
$$= \frac{-1}{(1+e^{x})(1+e^{-x})} \quad \text{factor} \tag{23}$$
$$= -\sigma(x)\sigma(-x) \quad \text{decompose using definition of } \sigma(x) \tag{24}$$
$$= -\sigma(x)(1-\sigma(x)) \quad \text{apply } \sigma(-x) = 1-\sigma(x) \tag{25}$$
$$= \sigma(x)(\sigma(x)-1) \quad \text{distribute the } -1 \tag{26}$$

Todo: this is easier with the identities of the derivative of a logistic function.

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} e(\mathbf{w}) \tag{27}$$

$$e(\mathbf{w}) = \sum_{i=1}^{n} y_i \log \frac{1}{1+e^{-\mathbf{w}^\top \mathbf{x_i}}} + (1-y_i) \log \frac{1}{1+e^{\mathbf{w}^\top \mathbf{x_i}}} \tag{28}$$

$$= \arg\min_{\mathbf{w}} \sum_{i=1}^{n} -y_i \log \left(1+e^{-\mathbf{w}^\top \mathbf{x_i}}\right) - (1-y_i) \log \left(1+e^{\mathbf{w}^\top \mathbf{x_i}}\right) \tag{29}$$

$$\nabla e(\mathbf{w}) = \sum_{i=1}^{n} \frac{y_i \mathbf{x_i}}{1+e^{-\mathbf{w}^\top \mathbf{x_i}}} - \frac{(1-y_i)\,\mathbf{x_i}}{1+e^{\mathbf{w}^\top \mathbf{x_i}}} \tag{30}$$

$$= \sum_{i=1}^{n} \mathbf{x_i} \left(\frac{y_i}{1+e^{-\mathbf{w}^\top \mathbf{x_i}}} - \frac{(1-y_i)}{1+e^{\mathbf{w}^\top \mathbf{x_i}}} \right) \tag{31}$$

8 Implementation?

Todo