COMP 642 — Machine Learning Jan 25, 2022

Lecture 5: Deep Learning: Logistic Regression

Lecturer: Anshumali Shrivastava
Scribe By: Kristina Sanclemente, James Kafer, Tess Houlette, and Sarah McDonnell

Disclaimer: *These lecture notes are intended to develop the thought process and intuition in machine learning. The materials are not thoroughly reviewed and can contain errors.*

1 Multi-Class Classification: One-vs-All

A multi-class classification is a classification technique that allows us to categorize data with more than two class labels. Trained multi-class classifiers are able to predict labels for test data based on those that are present in training data. One-Vs-All Classification is a method of multi-class classification. It can be broken down by splitting up the multi-class classification problem into multiple binary classifier models. For k class labels present in the dataset, k binary classifiers are needed in One-vs-All multi-class classification.

Since binary classification is the foundation of One-vs-All classification, here is a quick review of binary classification before we explore One-vs-All classification further.

1.1 Review of Binary Classification Model

In binary classification, the given data $D = \{x_i, y_i\}_{i=1}^n$ is classified into two discrete classes:

$$y_i = \begin{cases} 0 & \text{class 1} \\ 1 & \text{class 2} \end{cases}$$

Binary classification problems requires only one classifier and its effectiveness is easily visualized and understood using a confusion matrix.[1].

		Predicted Class	
		Normal	Attack
Actual Class	Normal	True Negative (TN)	False Positive (FP)
	Attack	False Negative (FN)	True Positive (TP)

If you can solve a binary classification problem, you can solve a multi-class classification problem.

[1] https://towardsdatascience.com/demystifying-confusion-matrix-confusion-9e82201592fd

1.2 One-vs-All Classification

Again, one-vs-all classification breaks down k classes present in our dataset D into k binary classifier models that aims to classify a data point as either part of the current class k_i or as part of all other classes. Each model can discriminate the i^{th} class with everything else.

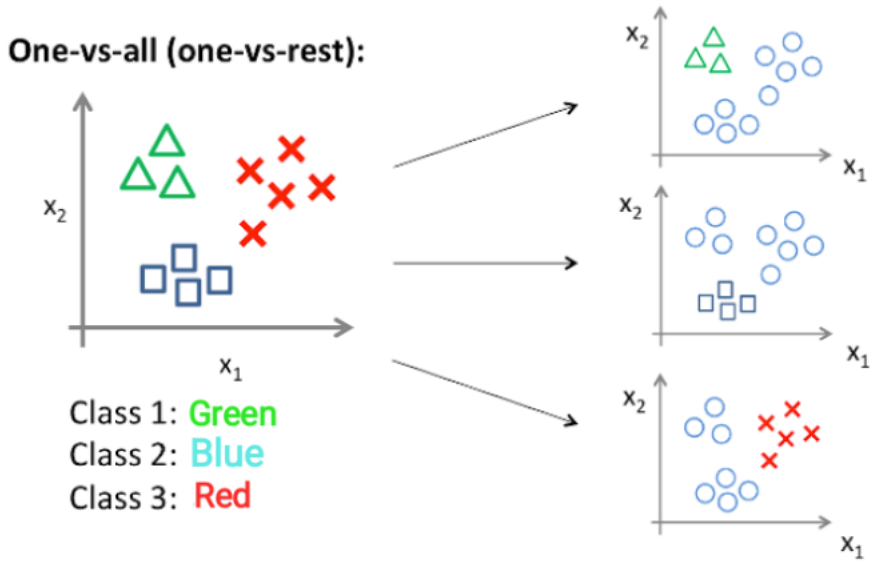
Example: Suppose you have classes A, B, and C. We will build one model for each class:

Model 1: A or BC
Model 2: B or AC
Model 3: C or AB

Another way to think about the models is each class vs everything else (hence the name):

Model 1: A or not A
Model 2: B or not B
Model 3: C or not C

A visual representation of One-vs-All classification can be seen below[2].



1.2.1 Confidence (or score) about a Prediction

The Problem: Suppose your confidence that the chances a data point belongs to Class 1 is very low. Logically, we know this should increase our confidence that this particular data point belongs to all the other classes. However, in One-vs-All classification, each binary classifier (Perceptron) is completely independent from all other $k-1$ classifiers that have been built to model the dataset at hand, meaning the probabilities output from each binary classifier need not sum to 1.

The Solution: Logistic Regression

[2] https://towardsdatascience.com/multi-class-classification-one-vs-all-one-vs-one-94daed32a87b

2 Logistic Regression

Note: Although named Logistic "Regression", it is used for classification.

2.1 What is Logistic Regression?

Logistic Regression is a classification algorithm used when the dependent (target) variables are categorical in nature- meaning the data can be grouped into discrete outputs $\{0, 1, ..., k-1\}$. Since we are dealing with categorical variables, logistical models must be used to map probabilities to predicted labels of the data. Examples of Logistic Regression classification include spam detection in email, cancer detection, and credit fraud detection.

There are three types of Logistic Regression:

1) Binomial: Where target variable is one of two classes
2) Multinomial: Where the target variable has three or more possible classes
3) Ordinal: Where the target variables have ordered categories

Out of the three types, logistic regression is most commonly used for predicting binary target variables. This lecture scribe will focus on Multinomial classification and briefly touch on Binomial classification.

2.2 Activation Functions used in Classification

Classification activation functions map probabilities of an outcome to categorical values. We will explore the Softmax Function and briefly the Sigmoid Function, which is a special case of the Softmax Function.

2.2.1 Softmax Activation Function

The Softmax function can be used in multi-class classification problems where the goal is to predict a single label from multiple classes. Let us make the assumption that the probability that x belongs to a certain class i is proportional to an exponential function as follows:

$$\mathbb{P}(x \text{ belongs to class } i) \propto e^{(xw_i)}$$

There are many reasons why people like e:

- It is easy to take the derivative of and you can take n number of derivatives with it.
- It is very sensitive. (or spiking)
- It has been proven to work really well in practice over time by many people.

In order solve the independence problem described in All-vs-One classification, we have to enforce proportionality and dependence in the outcomes of each class. This is done by normalizing $e^{(xw_i)}$ from our previous proportionality assumption:

$$a(z_i) = softmax(z_i) = \frac{e^{(xw_i)}}{\sum_{j=0}^{k} e^{(xw_j)}}, \quad z_i = x \cdot w_i \tag{1}$$

Note: xw_i is exactly what was used before: $F_i(x) = w^T x$ (expression can be written in different orientations depending on shape of input matrices W and X). xw_i could be used, and it might give us a very good classifier.

Whatever score the numerator was, it has now been forced into a probability. This can be seen as a probability distribution because sum of all the probabilities of each class z_i will equal 1. Because of this, if a classifier has a high confidence on one of the classes, it automatically implies that it cannot have a very high confidence on other classes.

For example, for a 3 class classification, if the probability of one class is 0.6, this implies the sum of the probabilities of the other two classes must equal 0.4. The output probability of each class from softmax function is translated into a prediction of the label and then compared to the true label of the data. There is a full example in Section 2.3.

2.2.2 Sigmoid Activation Function

The Sigmoid function is commonly used for classification of binary responses in Logistic Regression and is a special case of the Softmax where $k = 2$. It is a mathematical function that takes any real number and maps it to a probability between 1 and 0.

Note: In cases of multi-class classification (such as One-vs-All) that use a concatenation of binary logistic sigmoid functions, the sum of the probabilities of each model does not necessarily equal 1. This is different from the Softmax function.

Here is the equation for the Sigmoid Function:

$$f(x) = sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

As shown in Figure 1[3], the Sigmoid function forms an S shape curve. As $x \to \infty$, the probability becomes 1. As $x \to -\infty$, the probability becomes 0. The model determines what range of probability is mapped to which binary variable. For example, in Figure 1, the threshold was set at 0.5. So a probability greater than 0.5 would map to one of the outcomes whereas a probability less than 0.5 would map to the other outcome.

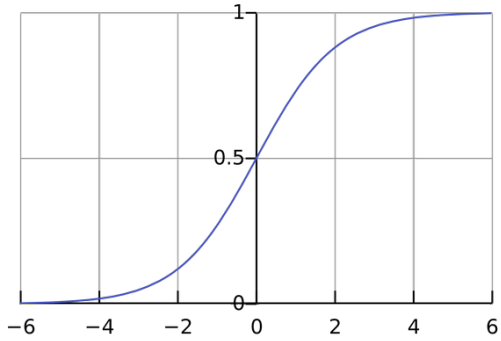


Figure 1: .

Additional reference material for the Sigmoid function can be found in this blog[3] which gives a simple overview of the Sigmoid function in relation to Logistic Regression. A comparison of the Sigmoid and Softmax functions can be found in this blog[4].

[3] https://www.educative.io/edpresso/what-is-sigmoid-and-its-role-in-logistic-regression

[4] https://medium.com/arteos-ai/the-differences-between-sigmoid-and-softmax-activation-function-12adee8cf322

2.3 Cross-Entropy Loss Function for multi-class classification

Any loss function is a "measure of goodness" between two functions: a predicted and expected target. This is usually in the form of an average distance between the two.

For multi-class classification with logistic regression, both the predicted and expected targets are probability distributions. Therefore, the distance used in the loss function can be derived from Kullback-Leibler (KL) Divergence. KL Divergence is a measure of distance between two probability distributions and takes the form:

$$= \sum P(x) log(\frac{P(x)}{Q(x)}) = \sum P(x) log(P(x)) - \sum P(x) log(Q(x))$$

where P(x) represents the expected target and Q(x) represents the predicted target.

When computing the loss function for different models, we will always have P(x) and the only difference is Q(x). Therefore, we can ignore any "constants" that do not rely on Q(x).

$$= - \sum P(x) log(Q(x))$$

Replacing with our notation we obtain the **Cross-Entropy Loss Function**:

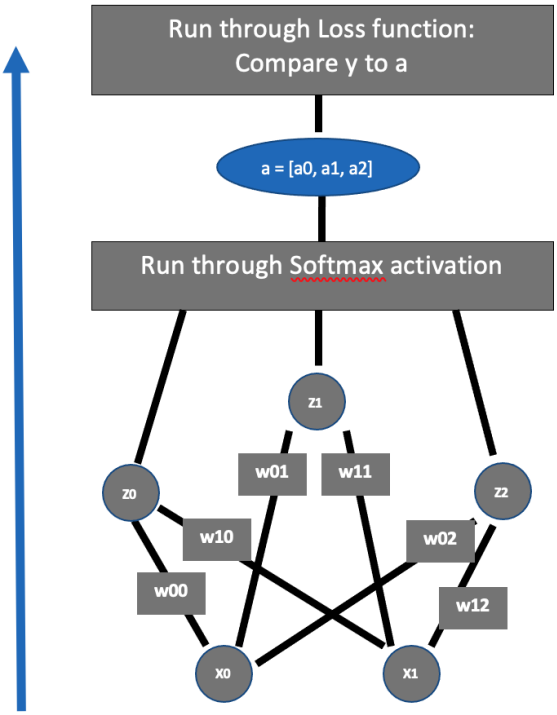
$$\mathcal{L}(a, y) = \sum_{n=0}^{N} \sum_{k=0}^{K} -y_k^n * log(a_k^n)$$

where

- Loss function ($\mathcal{L}$)
- Number of features (D), indexed by (d)
- Number of samples (N), indexed by (n)
- Features ($\mathbf{x}$) as a (N by D) matrix
- Number of target classes (K), indexed by (k), with specific target (t) $\in$ (T)
- True Values ($\mathbf{y}$) hot-encoded as a (N by K) matrix where, for a given n, $\sum_{k=0}^{K}(y_k^n) = 1$
- Weights ($\mathbf{w}$) as a (D by K) matrix, initialized to random numbers between 0 and 1.
- Net features (Logits) ($\mathbf{z}$): $\mathbf{x} \cdot \mathbf{w}$ as a (N by K) matrix
- activation ($\mathbf{a}$) for multi-class classifier uses softmax:

$$a(z)_t = \frac{exp(z_t)}{\sum_{k=0}^{K} exp(z_k)},$$

2.3.1 Example with Visualization: multi-class Classification with softmax activation



Example:

Given:

One sample, N = 1
T = {0, 1, 2}, so K = 3
x = [x0, x1], so D = 2
y = [y0, y1, y2], one-hot coded
w = [[w00 w01 w02],
[w10 w11 w12]]

Net features:

(This is NOT another layer; this is just an intermediate calculation)

Z = [Z0, Z1, Z2]
Z0 = (x0*w00) + (x1*w10)
Z1 = (x0*w01) + (x1*w11)
Z2 = (x0*W02) + (x1*w12)

Softmax activation:

a = [a0, a1, a2]
b = exp(Z0) + exp(Z1) + exp(Z2)
a0 = exp(Z0) / b
a1 = exp(Z1) / b
a2 = exp(Z2) / b

3 Note

The end of the lecture recaps XOR problem (see notes from lecture 3 or 4) and introduces deep learning (see notes from lecture 6)