https://www.udacity.com/course/deep-learning--ud730

[730] (Deep Learning)

Albert Garcia

Vincent Vanhoucke • (Winter) 2016 • Udacity – Google Research

Last Revision: January 23, 2016

Table of Contents

Abstract

These notes are intended as a resource for myself; past, present, or future students of this course, and anyone interested in the material. The goal is to provide an end-to-end resource that covers all material discussed in the course displayed in an organized manner. If you spot any errors or would like to contribute, please contact me directly.

1 Machine Learning to Deep Learning

1.1 Linear Model

$$wx + b = y$$

1.2 Softmax

The way to turn scores (also known as logits) into probabilities is to use a softmax function S and apply it to all the elements of the classifier output vector Y. Assuming that the output of the classifier has n dimensions, the softmax function is defined as follows

$$S(y_i) = \frac{e^{y_i}}{\displaystyle\sum_{j=0}^{n} e^{y_j}}$$

There are two important remarks about this softmax function. On the one hand, if the magnitude of the scores increases significantly, the probabilities get close to either 0 or 1. On the other hand, if the magnitude of the scores decreases, the probabilities tend to get close to the uniform distribution. In other words, increasing the magnitude of the classifier output makes the system more confident about the decision once softmax is applied and vice-versa. For machine learning systems, we would like to progressively turn the system from unsure to confident.

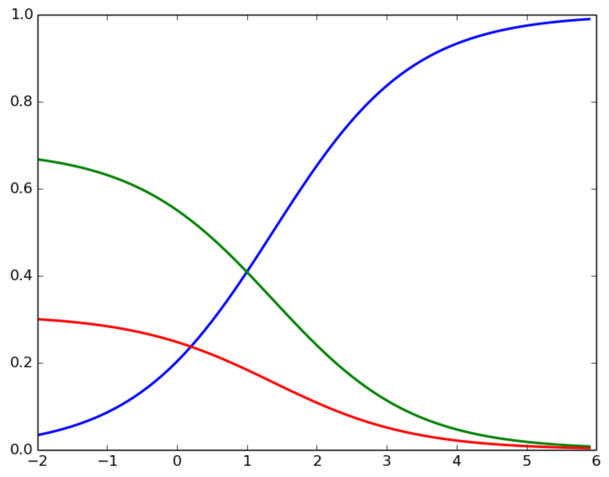

Figure 1.1

1.3 One-Hot Encoding

We need a way to represent class labels mathematically. We can use some sort of one-hot encoding for that, representing each label with a vector that has as many elements as class labels our system is able to classify. Each one-hot class vector will have a 1 at the position of the corresponding class, and the rest of the elements will be 0.

a	0	0	0	1
b	0	0	1	0
c	1	0	0	0
d	0	1	0	0

Table 1: One-hot encoding example vectors for four classes a, b, c, and d.

By doing this, we can map the classifier output probability provided by softmax to the appropriate class label following a likeness criteria. For instance, for the output probability vector $S(y) = \begin{bmatrix} 0.7 & 0.0 & 0.1 & 0.2 \end{bmatrix}$, and the one-hot encoding example shown previously, the most likely class is c.
This approach works well for most of the cases, but in some situations we can have a vast number of classes so that we end up having huge class vectors with almost all positions filled up with zeroes. This turns out to be quite inefficient. However, using one-hot encoding provides us a way to determine how well our classifier is doing by comparing the probabilities vector provided by softmax and the correct class label one-hot encoded vector.

1.4 Cross Entropy

The natural way to measure the difference between two probability vectors is the cross-entropy D. Given a probability vector from softmax $S(y)$, and a class label one-hot encoded vector L, their difference can be expressed as follows

$$D(S, L) = -\sum_{i=0}^{n} L_i \log(S_i)$$

Be wary, $D(S, L) \neq D(L, S)$**.**

1.5 Multinomial Logistic Classification

$$D(S(wx + b), L)$$

1.6 Minimizing Cross Entropy