Introduction to Machine Learning for Newbies

Part II: Practical Considerations for Input Data Preparation

Kyungmin Kim, Ph.D.

Created Apr 8, 2016
Last updated Apr 10, 2016

Preface

In the part I, I've summarized the conceptual definitions, basic principles, and kinds of ML. (If you didn't see the part I yet, you can find it from https://goo.gl/gC7KrY).

In this part II, I'd like to spend a bit more slides about some practical considerations which are needful in the preparation of input data because I think the related statements in the part I are not sufficient for practical implementation of machine learning to your problem.

Let's see how we can prepare input data properly through following slides.

Size of Training Input Data

The size of training input data can be determined by both

- Number of training samples and
- Number of features.

If you want to surely get an advantage about using ML compared to writing a full process of computation by yourself, you may guess that it should be better when you have bigger the size of input data as much as possible.

But I'd like to say either it's right or it's wrong for your guess. Then let's see when it's right and when it's wrong in next slide for the side of number of training samples.

Number of Training Samples

- Answer 1: It’s right.
  - If you have more bigger size of data, particularly for training input data, your machine can learn from more various examples.
  - Then you may avoid that the machine does underestimation for your training input data.
- Answer 2: It’s wrong.
  - If the variety of training input data reaches a certain limit, training a machine with more data over the limit may be redundant, that is, you may spend more time which you don’t need to consume.
  - Whether you have too much data or not, you can naturally expect that training many examples will need more time than smaller.

Then, now you can ask what an appropriate answer is. In short, prepare it properly. Am I kidding? No! I mean you need to spend some time to get a proper upper limit for the size of input data in sense of both the number of samples and the number of features and decide the upper limit depends on your problem and your MLA.

It’s also known that even number of samples for each class is more efficient. But, unfortunately, I don’t have an exact reference about this statement. Sorry about that.

Number of Features

For a proper number of features, the situation is similar to the number of samples case, that is, if you have too many features, it will surely need more time to train a machine fully.

But, a difference in this case to the number of samples is not just related to the computation time but related to the performance efficiency or accuracy. Then let's see when this issue can make trouble in your problem.

- ➔ Some of your features may be redundant.
  - ◆ If one of your features shows very similar distribution to other(s), it is redundant.
- ➔ Some of your features may be irrelevant.
  - ◆ If one of your features doesn't show any differences between different classes, this feature is irrelevant because it may cause some confusion to your machine.

So, (1) you need to choose proper features based on your problem or (2) you can use other statistical (or computational) algorithms like Principal Component Analysis which picks more relevant features based on a statistical point of view about the distributions of your input data itself.

Size of Test Input Data

You may not need to worry about the size of test input data if you don't need a real-time processing because you have much smaller size of test input data than that of training input data, usually.

But if you consider a real-time processing with a MLA, it may be a trouble.

In my experience with my Macbook Air (1.7 GHz Intel Core i5 and 4 GB DDR3), the order of evaluation time for test input data on a single core was about milliseconds (surely it can be varied by the size of your test input data) in the wall clock time.

So, I think that whether you have very powerful computer or not, you may perform a pseudo real-time processing at least with your test input data and your MLA. You can trust me!

Update Trained Result with New Input Data

Let’s suppose some unknown test input data are found that they are outliers compared to the distributions of existing training input data. Then, do we need to train our machine again by adding those outliers to existing train input samples for future evaluations? Well, I think it can be varied by your situation.

1. The most natural treatment is train our machine with a new train input data which includes both existing train input data and the new input data.
2. But, if it is hard to add new input data to existing input data, for example, if you don’t have a permission to modify the existing train input data, you can just update the trained results for parameters of your MLA. The principle is this: Set the initial parameters of the training process with the existing trained results and then train more with new input data.

Update Trained Result with New Input Data (cont’d)

If your situation corresponds to the case 2, you’re lucky because most of MLA packages contains an option for updating trained result!

But you should be careful about using the update option because it may shift the parameters of your MLA from the global minima to a local minima which is not the true goal of the training process on your train input data.

So, if you are sure the new input data is not much different with the existing train input data, just update the trained result like case 2. But, if you are not so urgent, I recommend to train your MLA again like case 1 for safety.

Have fun
with
Machine Learning! ;)