Data, Tuples, Features

Alex S.[*]

1 Data

Every process generates data. It comes in many different forms: coordinates, text, images, sound, video, etc. Computers deal with all these different formats by representing everything digitally—so for our purposes, it is all just bytes.

Raw data is not very useful. It often includes noise, measurement error, irrelevant things, etc. What we want to do is extract relevant *features*, and apply our algorithms (whatever they may be) on those features.

Obviously different data will have different features. For example, for text, features may be occurance of words, or word frequencies. Images may have lists of recognized edges or shapes. Sound may have a list of sequency components. Video may be treated as a stream or as a series of images, etc.

2 Tuples and Features

Whatever the input format, our goal is to organize it into *tuples*, such as $(a_1, \ldots, a_n)$, where each item in the tuple represent the value of some feature.

For example, if we are dealing with text, and our dictionary of words contains say 100000 distinct words, then we could setup a bag-of-words representation of documents by turning each document into a tuple such as $(a_1, \ldots, a_{100000})$, where each item represents word frequency. With this representation, we can compare one document to another by doing an inner product of the document duples.

Feature extraction is not trivial. It depends on data, domain, and often uses heuristics specific for a particular application. Success of algorithms often depends on robust extraction of correct features—without this critical step, none of the later steps would be feasible.

To complicate matters, feature extraction is often layered. For example, for images, you may first extract edges. In second pass, you might extract shapes (using the edges from first step). Then extract scale/rotation invariant features, etc.

[*]alex@theparticle.com