Need a sample size really large in high dimensions to represent the original dataset (original data distribution).

e.g. in one dimension, maybe just 1000 data points (as a sample) would be sufficient to represent the original data distribution.

But in higher dimensions (2, 3 or more), need to have many more data points as a sample to represent the original data distribution in a relatively nice manner.

Each bin has equal number of points in it.

Entropy of this sample

Entropy: a more general quantification of uncertainty, a measure of ignorance.

by thinking of a data set as actual observed “microstates” of the system (assuming total accuracy in measurement); entropy-based analysis is then somewhat of a reversal of classical entropic analysis in that a set of “macrostates” is sought to represent the underlying phenomenon as a coarse-grain model, and the concept of entropy then quantifies the uncertainty in using this model to predict the unknown phenomenon.

we will assume that measurements of some macroscopic phenomenon can be considered truly continuous.

We define geometric partition entropy as a measure of ignorance in prediction of large-scale outcomes, given a limited sample of observations.

The geometric partition itself provides a coarse grain representation of the phenomenon as a set of equally probable Macrostates.

3. Traditional Entropy Estimators

we will see that our estimator often agrees with traditional approaches, but offers certain advantages for more complex data sets.

A sample of data 🡪 determine the quantiles based on my sample 🡪 difference in quantile divided by the overall range 🡪 distribution 🡪 compute the entropy of the distribution

60000 pictures that we want to train our model on, want to select a subsample from it in an intelligent way so that the ML models trained on this subsample is as good as the model is trained with the original dataset

Use PyTorch etc. framework

Extract some features of the picture and represent it as a feature vector and this vector goes into the NN and gets classified.

Take a picture of the images and map it into a feature vector. Just to be consistent, no need to care what exactly the feature space we want.

Train the NN with PyTorch existing frameworks

So start with some sample NN architecture, some sample model that performs well. This model is

So 60000 feature vectors in the end.

When entropies between the sample and overall set are more similar, then get better result with when training on this sample.

Take a bunch of random samples, compare the difference in entropy between the samples and the overall set and difference in performance. Train the model more efficiently.

This sample is good in terms of entropy, let’s use that to train the model. Likely to be successful.

60000 feature vectors, each column is one image, form a matrix representing all of your data. SVD is a generalization of eigenvalue decomposition. Break this feature space U S V three matrices and we can multiply them together to get A back. Left and right matrices are like eigenvectors associated with certain directions in the data. The direction with the most variance like PCA

The middle diagonal matrix are just like the singular values, they are essentially the variance in each of the singular vector directions. The first singular value gives you a measure of what the variance in the first vector direction is. Etc. they let you know how the data is distributed.

The first singular vector has the highest variance. The second singular vector is orthogonal to that one and has the second highest variance.

Then BSEI (unb

Compute SVD of data. So we get a set of variance and a set of corresponding directions.

And how is this variance compared with the variance of the original set. However, our subsample has the variance as the original sample, but it is in a different direction, so it might be an indication that this is not a good representation of the original set.

Use BSIE to figure out are those distribution of singular values similar between the sample and the original set. Then check if the corresponding eigenvectors are the same directions. Use the built-in function in different libraries.

Why 5000 is not working? CNN models are data thirsty and tend to overfit? Try to increase the sample size to 30000 for instance or even more.

Try to see how to use PCA to do data efficient training? Only select essential features to do the training. See the regression as well. Or you can even start with that model to train. Right, if CNN is not working, what about other ML models? You can change the datasets we used as well like Iris

Maybe they took less time to train.

Maybe we don’t use CNN to extract the feature space of images. Other ways to do it?