Data Visualization Towards An Understanding of the Benefits or Detriments of Pipeline Diversity

Data Machines Corporation

August 13, 2020

We develop pipeline visualization methods with an eye to addressing questions like the following ones:

1. Looking at a single team for a particular problem/dataset, to what extent has the team built diverse pipelines, with many different TA1 primitives in different orders? Or alternatively, to what extent has the team submitted many similar pipelines, but most with just small differences in primitives—possibly differing largely or solely in hyperparameters?

2. Comparing teams, which teams tend to submit more diverse pipelines, in terms of primitives, and which tend to submit less diverse pipelines?

3. Assessing the corpus of pipelines for all teams and for a particular problem/dataset, to what extent are teams' collections of pipelines different from other teams'? Or alternatively, do the collections of pipelines for different teams repeat variations on one or a few themes?

4. Examining numerous problems of a single type (e.g. image classification), does a given team resubmit similar pipelines/primitives for all problems of that type? (E.g. Do classify MNIST, CIFAR-10, and CIFAR-100 the the same way?) Or does their approach vary highly with the particulars of the problem?

Our visualizations provide a method for approaching the above questions and will soon help us know if diversity, as defined above, tends to build "better" pipelines. That said, our analysis will not show whether for a given problem, a lack of diversity across all teams indicates an appropriate convergence to the best solution, or alternatively, whether gaping holes exist in the explored part of the solution space occluding better solutions. However, our visualizations together with available pipeline-success metrics will soon measure the success or (lack of success) of *a team's strategy* of submitting diverse pipelines, versus an alternative homogeneous strategy of submitting many small variations to the same one or a few theme(s).

Our data comes from the Winter 2020 Evaluation of the D3M project.

1 Contents of Document

This document consists of three parts. The first part (these first few pages) is the textual introduction you are now reading. The second part (comprising most of the document) has 103 sections, one each visualizing the pipelines of a single problem/dataset under consideration. The last part (comprising the last several pages of the document) contains a table of codes for primitives, and their descriptions. These codes are used throughout the second part of the document.

Within each section, the top of the page labels the problem/dataset under consideration (out of 103 possibilities). Beyond this label, there are two pieces to each section. The first piece is an image showing, as best as possible, "distances" between pipelines, specifically edit distances, or minimal number of substitutions, deletions, and insertions, needed to transform one sequence to the other. The second piece of each section shows pipelines as coded sequences of primitives in a multiple global alignment, including all performers and all pipelines for this problem.

To elaborate, each image was generated by first computing the edit distances (Levenshtein distances) between all pairs of pipelines (within and across performers) for the specific problem under consideration. These edit distances were then transformed and passed to a spring layout procedure (Fruchterman-Reingold), to determine the $x-$ and $y-$ location of the pipelines on the two-dimensional plane of the figure. The F-R procedure model nodes as point masses under virtual spring forces. Every node repels every other node, but attractive forces exist as well that pull nodes toward each other: the closer nodes are in edit space, the more they attract. The code I used is described at `https://networkx.github.io/documentation/stable/reference/generated/networkx.drawing.layout.spring_layout.html`. Each circle in the scatter plot represents one pipeline evaluated for that problem. The color of the corresponding circle categorizes the 9 performers (tamu, mit, nyu, sri, uncharted, etc).

The axes of the plot are meaningless, but pipelines with greater edit distances between them should appear further apart on the graph. Technically, we pass the reciprocal of the edit distance to the F-R code, but for identical pipelines, the software does not accept infinity—and enforces a minimum distance between nodes. As a result, identical pipelines appear distinct at a nonzero distance. That said, it is often the case that scatter circles partially overlap when corresponding pipelines are identical, and lie wholly separated when not identical.

Note that there is some loss of information in moving the sequences to two dimensions, as only three points can be represented equidistant in the plane. As a result, one can not conclude that points that are not equidistant on the plot are not equidistant in edit space. For example, there can be a swarm of points in the plane for which the pipelines all differ by the substitution of a single primitive in the same position. Nearest neighbors in this swam will tend to be equidistant, but the diameter of the swam in the planar scatter plot will exceed many times the distance between nearest neighbors.

2 The First Problem/Dataset (CIFAR-10)

Let's look at an example: the first section found below in the second part of the document. Here "first" derives from an alphabetical ordering of the problem titles, shown at the top of each page. This first problem has the title "124_174_cifar10_MIN_METADATA." The keywords for this problem are "classification," "multiClass," and "image" (not currently displayed).

To me, the first thing that jumps out from the image are the patches of yellow dots. The table of multiple sequence alignment, second piece of each section, gives a more precise picture, of what is happening.

The alignments were creating with a program called mafft which was originally intended for biological sequences, by minimizing an objective function penalizing gaps and substitutions.

Colors of aligned pipelines correspond to colors of scatter circles (and both indicate the performer) The performer is also labeled on the multiple alignment, as is the pipeline number, which also labels the scatter circle). Labels of identical scatter circles can overlap rendering them illegible, however the redundancy of the information usually resolves the ambiguity.

The yellow pipelines, from MIT, have numbers consecutive numbers from 5435 to 5454. A performer should always have consecutive numbers for the a single problem.

As suggested in the scatter plot and confirmed in the alignment, MIT submitted only two distinct pipeline sequences. Pipelines 5435 and 5442 have sequence SN-IB-XD-RS-EO-MM, the other 17 have sequence SN-IB-UD-XD-ZK-ZK-UJ-MM.

According to the translator in the last part of this document, the shorter sequence translates to:

SN	'Extract a DataFrame from a Dataset'
IB	'Determine missing semantic types for columns automatically'
XD	'Parses strings into their types'
RS	'Single Table Deep Feature Synthesis'
EO	'XGBoost GBTree classifier'
MM	'Construct pipeline predictions output'

As slightly visible on the image, the brown pipeline 879 (from UCB) also has this sequence of primitives. On the other hand, the other 17 yellow pipelines (from MIT) stand alone with a slightly longer sequence. Indeed, the first, second, and last primitives are the same. The other five primitives, having codes starting with UD and ending with UJ are

SN	
IB	
UD	'Remove semantic types from columns'
XD	'Parses strings into their types'
ZK	'Extracts columns by semantic type'
ZK	'Extracts columns by semantic type'
UJ	'Tree-Augmented Naive Bayes Classifier'
MM	

The multiple sequence alignment aligns primitives SN and IB, then aligns the pipelines with substitutions to the end of the sequence, ignoring the coincidence of MM. To reiterate, surprisingly, the common primitives MM are not aligned by mafft. The algorithm prefers not to add gaps toward the end of the sequence unless the gaps fall *after* the very end of one of the sequences. My guess is that gaps at the very tail of a sequence are not penalized, whereas gaps in the middle are penalized more than substitutions. Thus a gap will be assigned thereby accepting the larger gap penalty (larger than the substitution penalty) only if it is counterbalanced by preventing a subsequent sufficient number of substitutions. Presumably, the large gap penalty prevents an undesirable number of gaps in the alignment, which would render the alignment hard to read, and may have other undesirable features in certain problem domains.

There are a few artifacts here. A primitive denoted ** indicates a "bug" that appears occasionally in the output of mafft. If it is a "feature", I'm not sure what it's for. This "bug" reared its ugly head twice in this problem, in pipelines 880 and 881 (UCB) with sequences KV-KV-**-E0-MM and KV-**-DD-LB-MM. The missing primitive was in both cases AB. It is possible that the string encoding I am using assigns a forbidden value to the primitive AB for use in the multiple alignment. In any event, the code that computes Levenshtein distances and FR positioning is completely independent from the alignment, and will not have this problem. As a result, the Levenstein distance between these sequence is 3, and their distance on the plot is visible. The other UCB (brown) circle representing pipeline 879, overlaps (and is identical to) the shorter yellow pipelines, but should have a larger edit distance to the other UCB pipelines. This larger distance is not reflected in the graph.

I should point out that the F-R positioning algorithm has a random element to it and is dependent on initial condition. Because nodes are added in order, keeping pipelines from the same performer together, I believe that the initial condition systematically places pipelines from the same performer near each other, assuming edit distances are equal. It would be interesting to see if a random ordering of nodes breaks this effect, but in any event, I would choose to keep the effect because it is pleasing to the eye and leads to images which are easier to read. The UCB (brown) circles are an exception to this good behavior, at least in the first image shown. Still, it would be interesting to see if the same ordering of nodes but different random seed leads to a more reasonable positioning of the UCB (brown) pipelines. I have noted that different random seeds produce differently looking images, but usually edit distances are faithfully represented, and they are here for most pairs of circles.

The red circles representing the pipelines from cmu fall at distance one (one substitution) from each other. They all differ in the second to last primitive of their respective pipelines.

3 Future Work

We will work on answering the questions that opened this document. To this end, it is important to develop quantitative measures to go along with the visualizations. How do you quantify diversity of pipelines from a single team? One idea is to sum or average the edit distances between pairs of pipelines across all pipelines submitted by the team.

In addition to quantative measures, some improvements could be made to the images. The images were created with the networkx code that produces the F-R positioning, however other graphics engines may be desirable, such as matplotlib, seaborn, or ggplot2.

With any of these choices we could represent using symbol size the accuracy of the pipeline (or whatever metric is used for the problem), relative to the same metric applied to other pipelines. The diameter of the circles could map to a z-score range of -3 to 3, with all pipelines below the range having some fixed minimal diameter.

To compare problems/datasets within a whole problem type (e.g. classification), we could use different symbols other than circles, one symbol for each different problem within the problem type. The size of symbols will still indicate the relative value of the metric used for the problem, and colors could still indicate performer, even if only one performer is considered in the plot.

Additionally we could add some sort of legend showing the relationship between edit distances and distances on the F-R plot. One idea is a scatter plot between these two quantities across all pairs of symbols on the original graph.

Perhaps other graphs should be considered, such as side-by-side box plots of edit distances between pairs of pipelines, grouped by performer, with one additional box for pairs of pipelines across performers.