Similarity Modeling 1

Gabriel Breiner
Technical University
Vienna, Austria
gabrielbrbr@gmail.com

***Abstract*—This document is a summary of each of the 5 lectures of the "Similarity Modeling 1" course at TU Wien. Each chapter corresponds to one of the following lectures: 1. Setting, 2. Similarity measurement, 3. Feature engineering, 4. Classification, 5. Evaluation.**

***Index Terms*—similarity, modeling, convolutions**

I. The Setting of Similarity Modeling

In order to understand similarity modeling, we have to take a closer look at the setting, in which in takes place: There is a stimulus, a signal of any kind that is perceived by a sensor. The qualities of this stimulus need to be modeled in way that allow us to infer its similarity to things we have already perceived. We can then compare the modeled stimulus to our knowledge base of similarly modeled data and therefore infer if something is of similar or different kind.
Note that in this brief description it is not specified whether we are talking about a human being or a machine. This is because the perception processes of machines and humans do not differ by a lot structurally. The main difference here is that humans are able to derive deep and layered forms of semantic meaning out of these stimuli, that machines currently lack to understand. A machine is perfectly capable of detecting certain patterns and correctly classifying some meaning to them, which is already some sort of semantic meaning, but there are infinitely more layers of semantic information, that are only accessible to humans. The deeper we go down this semantic cascade, the less our algorithms are able to understand.
As hinted from above both machine and human perception can be modelled with these 5 components: stimulus, model, similarity, knowledge and category. First of all there is the stimulus, which is what is received from sensors (e.g. eyes or camera). This stimuli need to be modelled, so we can later calculate the distance to the known stimuli (knowledge). The modelling step also enables us to generalize the information that we perceive, so that we don't have to have every single sample "in memory" at any given point in time. The third component: Similarity measurement makes use of our model to calculate the distance between samples (e.g. unknown sample vs knowledge). Lastly, this distance measurement allows us to categorize a given sample, cluster many samples and many other tasks.
There still remain a few problems with this model and in similarity model in general: "The Semantic Gap", that was already hinted at, means that, while we can model similarity and use it to astonishing benefits, we can barely involve semantic understanding, specially on deeper levels. Other problems that remain are huge parameter spaces, which makes similarity modeling essentially a parameter optimization problem and imperfection of sensors.

II. Similarity Measurement

This lecture was about similarity measurement for humans and especially for machines. In order to understand how similarity is measured, we need to know about feature spaces. A feature space is a multi-dimensional space in which every feature (=every relevant quality of a sample) is represented by a dimension and every sample is described as a vector, that is an element of this feature space. This again, like with the perception process as a whole is similar between humans and computers: In the human brain this feature space is analogous to the weights, that are associated to the connections between neurons. While the human brain's feature space may be infinitely more complex and the features are more semantically loaded, the ones we model in computer feature models can be found to some degree in the human brain.
In this feature space, where every sample is a vector, a great number of functions exist, that can be used to either measure the similarity or the distance between two samples. While this may seem like 2 distinct categories of measurements, it can be shown that all distance functions can be represented as similarity functions. This means that both similarity and distance functions fall into a spectrum: On the lower end are 'thematic' or 'integral' methods with L1norm being the most extreme, on the higher end are 'taxonomic' or 'integral' functions, with cosine similarity being the most extreme (and known) example. 'Taxonomic' functions are called positive convolutions and 'Thematic' functions are called negative convolutions. While negative convolutions are more forgiving, positive convolutions are more discriminative, so if two samples are very different positive convolutions would rather give a small similarity measure, while negative convolutions would give a relatively smaller amount of distance. This could be because positive convolutions put two samples in a multiplicative, while negative convolutions put them in a subtractive relation. This is best shown with binary features, which tend to carry high semantic meaning regularly. Only if the samples are 'overlapping' taxonomically (both having =1 in the respective feature) will the function not output 0. This means that there is also a correlation between 'seman-

ticness' and the aforementioned similarity function spectrum: lower semantic meaning is better calculated with 'thematic' distance functions, higher semantic meaning with 'taxonomic' functions. It shall be noted, that there are also dual process models, which use a split function, where both taxonomic and thematic elements are combined together. Interestingly enough psychologists independently concurred with this idea.

III. Feature Engineering

Feature Engineering is the process of 'dealing' with all the different data (often encountered in waveform; e.g. audio/image data) and processing it, so we only deal with the most relevant points and patterns or are able to extract them. In comparison to deep learning methods, this requires more expertise and knowledge of the data and may not be the best in all use cases. The advantages of classical feature engineering compared to deep learning are, that it requires less data to work, less computational power and is better in modelling high semantics.

Image media is very much concerned with color, light, saturation and hue. To model these features, many attempts at creating feature spaces have been made. The classical RGB model expresses a pixel of an image as a combination of 3 colors. This is a very restrictive model, as it cannot model brightness or saturation. In this regard the HSV model transports this model into a 3D space. The XYZ and YCrCb model are designed to mimic human perception and assign similar priorities to colors, and brightness as the human eye. Even though these are necessary to model data, they do not able us to derive high semantic meaning by abstracting or generalizing the data enough. In this regard features like color histograms, surface properties, edge or fore/background detection may be more useful. In color histograms for example every color is assigned to a bucket. This aggregates the data and also translates easily into vector representations in our feature space. Similarly edge detection enables us to detect the boundaries of objects on the screen and are thus able to derive very high semantic meaning. This is done by applying a convolution between frame areas and predefined matrices that represent different kinds of edges (=cross-correlation).

The Audio signals can be viewed as rather holistic, meaning that assumptions can seldom be made from individual points in the signal, but rather from patterns or the signal as a whole. Loudness, Fundamental Frequency, Rhythm Detection and the Attack/Decay/Sustain/Release Model can serve as semantically highly loaded features to derive information. The fundamental of all most audio methods is the arbitrary definition of a frame size (this can very from 20ms in speech task to 1s in environmental sound tasks). In order to do rhythm detection, we try out different frame sizes and do convolutions on the neighboring frames (=autocorrelation). If the similarity or distance surpasses a threshold on multiple following frames, we can assume, that we have found some sort of rhythm.

Finally we can see a similar pattern in many feature extraction methods: All of them are based on convolutional operations, that are applied between a unit of the signal to either other units of the signal or units, that are predefined. In the first case it is called auto-correlation, the second case is called cross-correlation. If the similarity or distance surpasses a certain threshold, we consider a certain feature detected. This enables us to train a classifier and finally arrive at classifying our samples.

IV. Classification

A classifier receives some input sample and is then able to classify the given sample as a member of one of multiple classes. We can imagine a class as a group of data points, that belong to each other, because they are similar in a certain semantic way, that if we properly model our feature space, should result in a close distant or high similarity in the feature space, meaning that samples that belong a class are situated in clusters or Gaussian blobs. Ideally A) the boundaries by which we define such a cluster should be minimal and B) the distance between them should be maximal. Both of these ideal cases separate the taxonomy of classifiers into two groups: A) Hedging or centroid-based algorithms, they are associated with the classical concept theory and also with negative convolutions and B) Separation algorithms, that are closely associated with prototype theory and positive convolutions. Abstractly speaking, all kinds of classifiers are compression algorithms, since they transform a possibly huge feature space of n dimensions into a single bit through semantic enrichment of our data. Classifiers can further be categorized into supervised and unsupervised algorithms. The first category is dependant on ground truth, while unsupervised algorithms are not.

The latter algorithms are also called clustering algorithm and come in two flavors: bottom-up and top-down. Both aim to create a tree structure, that is able to put the samples in relation: With the bottom-up approach, we pick a sample in question and then measure the distance to the nearest other sample. We then take the distance from another sample to our previous samples. If we continue this process, we get a Dendrogram. The other approach, which is called 'top-down' has a similar approach: We choose an entry point, from which we calculate the distance to the nearest sample. This entry point is the center point for all clusters. We then continue from point to point which also yields us a tree-like graph. Similar methods can be enriched by using annotated data (ground truth). With these methods a sample in question can be classified by looking at similar elements in the feature space. In the most simple case of KNN we look at the k closest neighbors, who then majority vote with their respective class for the class of the sample in question. K-means compares distance to the cluster centroids and self organizing maps is similar, but changes the position of other samples in the process.

Neural Networks have a long tradition, but have gained momentum with the introduction of GPUs to employ more efficient training and thus the ability to store more data. Additionally the vanishing gradient problem has been marginalized by the introduction of ReLU activation functions.

Convolutional Neural Networks introduce convolutions by passing the aggregations of multiple neighboring neurons to the next layer. This way certain neurons can serve very specific highly semantically loaded tasks, just like in the human brain. Regularization Layers were another major improvement to CNNs.

V. Evaluation

The central piece for evaluating a classification algorithm is the confusion matrix. For binary classification tasks, it is a 2x2 Matrix. The first dimension being the output of the classifier and the second being the ground truth. This leaves us with 4 values in the matrix, that signify the number of true positive, false positive, true negative and false negative classifications. By setting those values into diverse formulas we can get the well known evaluation metrics: precision, recall (true positive rate), fallout (false positive rate) and by combining precision and recall we get the f1-score.

There are 2 main visualizations for confusion matrices: The recall-precision-graph (RPG) and the Receiver Operating Characteristics (ROC) curve. The RPG has recall has precision on its x and recall on its y axis. Since both of them have desirable high values, the further out the curve is from the origin, the better the classification algorithm is performing. The second option is the ROC curve plot. It has fallout assigned to its x axis and recall on its y. Having a curve similar to the median in this curve is the worse outcome, since it is essentially a random classifier (e.g. coin toss). Extreme curves are desirable here (remember that an extremely bad classifier can simply be inverted to get an extremely good one).