ROCKET - RandOm Convolutional KErnel Transform Paper review

Paper link: https://arxiv.org/pdf/1910.13051.pdf

Reviewers: Omri Kazelnik.

Abstract

Time series classification is the task of categorizing time bounded segments and labeling them according to prior knowledge.
State of the art implementations and solutions conclude couple of straight forward features from the time streams, mainly series shape and frequency (derived from frequency domain, Fourier transform family) and as a result potentially missing additional meaningful information.
Moreover, the aforementioned solutions suffer from high computational complexity, requiring long training time even for small datasets.
ROCKET brings to the shelf a compact multivariate features solution aimed to reduce the training complexity and improve the classification generalization.

Paper Summary

ROCKET, stands for random convolutional kernel transformation, is a mixture of different kernels targets to classify time series streams.
Inspired by ensembling methods, the large amount of "significantly" different kernels helps to extract meaningful features from the time series segments, find patterns and improve the classification accuracy even on small datasets.
State of the art methods typically focus on univariate classification feature such as shape, frequency or variance where convolutional kernels constitute a mechanism which can find multivariate features relationships.
Indeed for some problems the frequency domain might provide enough information in order to classify correctly the streams, however we should remember that many of the problems compose of noisy data with potentially many crossing frequencies and different phases.
The randomness of the kernels selection and the kernels nature for capturing complex features information mixed together for a robust and effective classification method.

Drawbacks and Weaknesses

While the usage of many kernels has many advantages, the correct selection of the kernels is not trivial at all.
The common cases in deep learning where convolutional kernels are in use is in learning procedures where their weights optimize during the training. In the paper, the authors mention

that it is “well established” that random convolutional kernels can be effective - the question is if they indeed well established it.
Empirically it seems they are right, but maybe it is worth adding some regularization mechanisms to avoid overfitting issues.
In addition, a single layer convolutional indeed decreases the model complexity and its training time, but what about non linear combination between deep feature maps. One of the benefits of deep convolutional networks is the ability to find deep nested features (deep feature maps interpolation), which is strongly hard to achieve in single layer feature maps extraction (the amount of maps and their extracted features may be very large and complex).

Setup and Domain Considerations

ROCKET aims to classify time series streams by extracting features from convolutional kernels and apply a classifier to label them.
The setup of it is pretty straightforward:

1. K kernels random generation.
2. Classifier for the kernels features (softmax layer/ regression classifier).

There is no specific domain for ROCKET classification tasks, and it tested on 85 “bake off” datasets from UCR archive and additional 43 later datasets from 2018.

Paper Extensions

The main extension of paper is breaking the “rule” of deep convolutional neural networks architectures for classification tasks.
Instead of finding consecutive convolved features to extract information from the data, the paper proposes a more “ensemble oriented” solution with a single random wide convolutional layer.

Comparison to Prior Work

Compared to previous work, because the architecture is built with fewer variables and the gradients calculation is easier, the train time and it’s complexity decreased dramatically.
The mentioned decrease didn't harm the classification accuracy in comparison to the state of the art methods, and on some datasets even improved it’s mean.
From the paper's results it seems like ROCKET found almost the same characteristic function as deep convolutional networks, while keeping the same accuracy and improving the training time convergence.

Technical Aspects

The main innovation of the paper from the technical point of use is reducing the method complexing while gaining the same classification accuracy as state of the art methods.

The authors proposed two effectively equivalent architectures for ROCKET:

1. Single convolutional layer with softmax/sigmoid layer.
2. Single convolutional layer with conjugate regression model.

It is pretty trivial to prove that the regression coefficients represent the same function as the weights in the softmax/sigmoid function and therefore the two results should converge to the same optimal point.
However, in large datasets the second option holds an advantage because the gradient optimization is looser (less weights to optimize in single SGD iteration), and also the models cross validation is faster for hyperparameters regularization.
For the following parts we will deal mainly with the second option.

Features - pooling and ppv

A convolutional layer is a mathematical operator on the input and controlled by filtering window function. The operator is convolution between the input data and the filter's kernels, where each convolution results in receptive fields that yield feature maps.
In the paper, the kernels selected randomly and the feature space derived from two aggregations on the convolved features maps:

1. The maximum value of feature map channels (e.g. global max pooling).
2. The proportion of positive values in feature map channels.

So, if we have K kernels the amount of features is 2K.
The first feature type is responsible for detecting invariantly in successive stream time slots, where the second is responsible for capturing "strong steady" continuous patterns in the input.
The paper said that empirically these features are enough to gain the same results as other state of the arts methods.
That said, ppv showed the biggest effect on accuracy over all parameters. The combination of global max pooling and ppv increase the accuracy but not significantly.

Training Procedure

The authors use for the training all datasets which are in UCR archive.
In the beginning of the training phase, the kernels are already fixed (K sampled randomly), so the model **shouldn't** learn and optimize them.
For the classification, the authors chose ridge regression with L_2 regularization. Regularization is critically very important where the number of features is bigger than the amount of data samples, what might happen in our case. That's why, when we increase the number of kernels to improve our feature selection pool, we basically require our method to fit more potentially information that it holds (e.g. overfitting tendency). With good regularization, we "punish" large expectations in our loss function so the model can converge accurately and faster.
The ridge regression classifier can exploit generalised cross validation to determine the regularization amplitude.
In conclusion, the training phase gets the features from a fixed convolved feature maps, fits a ridge regression model with cross validation and produces a classification ridge regression model.

Prediction Procedure

The prediction procedure is pretty straightforward.
In inference, the model obtains the fixed K kernels, extracts the 2K features and predicts a probability from the fitted model.
There is no detailed description in the paper regarding the prediction phase, but I believe that it also improves dramatically the prediction time compared to the other methods.

Hyperparameters Experimental Tunings

As already mentioned before, the hyperparameters are summarized under the kernels setup. From that statement several questions were raised, how to choose them correctly? How to choose them random “enough” in order to generalize the input space widely and correctly? The following section describes in details the hyperparameters and their creation process.

Kernels Amount

The number of kernels playing the rule of “no free lunch” in the classification problem. Intuitively, more kennels will cause more accurate evaluations. However, large numbers of kernels cause more dependent parameters and therefore more complex classifiers.
Empirically (and supriosly), the actual difference between 5000 and 10000 kernels is relatively small. More precisely, the authors mentioned that it wasn't **statistically significant.**
In summary, even though ROCKET is non-deterministic method, the variability in accuracy is reasonably low for large numbers of kernels.

Kernels Length

The length of the kernels helps to catch locality connections between observations. As more accurate the kernel length compared to the “real” connections between the observations`s features the more accuracy gained by the method.
The paper varies kernel length by two methods:

1. Selecting randomly from the following groups - {3,5,7}, {5,7,9}, {9,11,13}, and {11,13,15}.
2. Selecting fixed lengths sampled uniformly from group {3, 5, 7, 9, 11, 13, 15}.

The two methods showed almost the same results, but naturally short kernels struggle to extract meaningful enough autocorrelated features.

Kernels Centering and Bias

Because the time series streams are normalized and standardized, cenerized kernels may perform better. Another way to state it is via cross-entropy between the kernel and steams distributions - centralized kernels will gain lower cross-entropy than spread ones.
The paper adds more randomness to ROCKET with centering the kernels with Binomial distribution where N is number of kernels and p is the probability of centralizing the kernel.
For bias, as other state of the arts methods the biases sampled from normal Gaussian distribution (in some cases the biases set to 0).

The authors found that non biased/centralized kernels achieve almost the same accuracy as the offsite on a large number of datasets, therefore they considered them to be statistically **insignificant**.

Kernels Weights

The kernel weights setup is a crucial part of deriving better features. Not just the baseline, where the weights initialized from normal distributions, ROCKET initializes them with two methods:

1. Sample from uniform distribution in the internal (-1,1).
2. Sample randomly from the group {-1,0,1}.

That's a very interesting insight. From first glance it seems that most of the features from these kernels should be pretty the same in option 2 or too versatile in option 1.
However, the kernels are still mean centered by default, have random bias and substantial variety in terms of length and dilation.
Moreover, one more important point is that the second option resembles derivative kernels which might be helpful to find local gradient changes in the streams.

Streams Dilation

Streams dilations is a key point to increase the performances. Different dilations capture a wider range of possible stream lengths and as a result enables the extraction of different autocorrelated relationships between stream's events.
Due to kernels usage, the streams length must take the kernel's length into consideration because out of kernel bounds streams may produce biased features.
The papers selected two methods to set streams dilations:

1. No dilations (e.g. fixed dilation of one).
2. Sample dilations uniformly form interval (1, len(input) -1 / len(kernel) - 1)

Intuitively the second option gained much more accuracy because of the mentioned above results.

Prototype

The prototype will follow the second implementation option - kernels random creation + train regression classifier model.

Moving Parts

The implementation will consider the following components:

1. Data loader for classification (tuples of steams and target).
2. Kernels generator (including features extraction).
3. Classifier.

Training

The training part will be composed of picking up random kernels with their hyperparameters setup. After the kernels are fixed, convolve the inputs with them, extract the features and fit the classifier.
The classifier training will also use a cross validation part to select the best regression weights and regulazier factor.

Prediction

The predictor part will be composed of loading the fixed kernels, convolving the inputs, extracting the features and sending them to the classifier's probability prediction.

New Ideas

More Features vs More Kernels

More kernels make the ROCKET method more accurate and generalized, however it influences the model complexity (more learning parameters).
Extracting more features from the feature maps **doesn't** increase the model complexity, and with correct extraction it affects the model performances.
For example, let's consider adding one more feature extraction in addition to the 2 existing, average pooling (notice isn't global pooling). The average pooling involves calculating the average for each stream's dilation.
If we will think of the feature map as effectively univariate streams average, the average pooling is exactly the same as the **MA** (moving average) part in **ARMA** models - it represents the PACF of each kernel receptive field.

Kernels "Smarter" Selection

The kernels are randomly selected in order to insert uncertainty generalization to the ROCKET method. The authors mentioned in the paper that the initial high amount of kernels required to gain accuracy as other state of the art methods.
If we will cluster our kernels to "families", we can sample a single/couple of kernels from a family and as a result save "similar" kernels selection.
For example, families can be "derivative", "autocorrelation" and "autoregressive".