

October 25, 2022

TUTORIAL • MACHINE-LEARNING



Realtime classification in Pathway

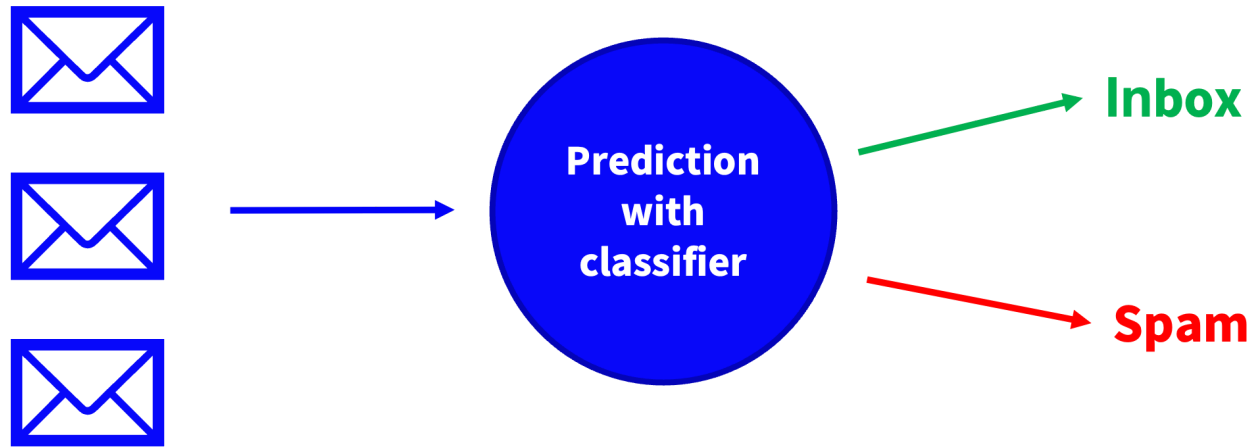
Part 1: Using the Nearest-Neighbors Classifier



Classification - what is it all about?

What is this handwritten digit? Is this new movie an action movie or a drama movie? Is this email a spam or an email from a real [foreign prince reaching out for help](#)?

All these questions have one thing in common: they are forms of **classification problems**.



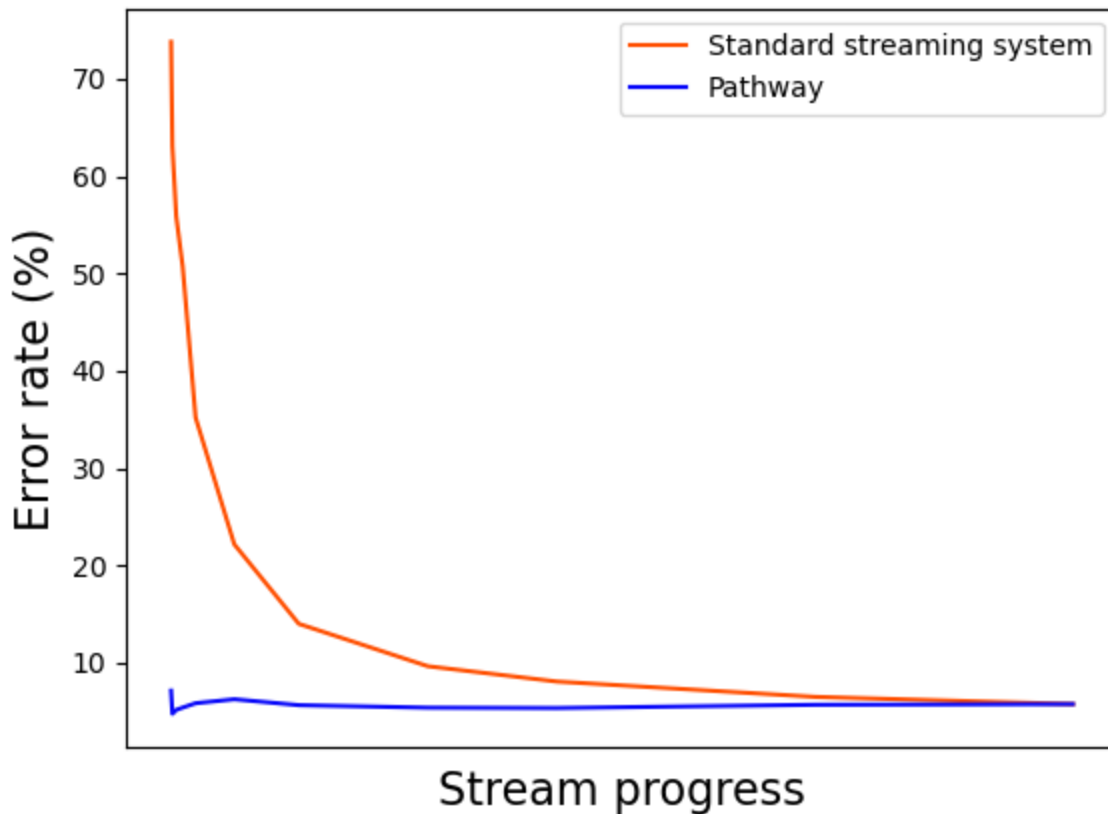
Classification is the process of giving a label to an unknown element. It is used to label or tag new content and has various areas of applications such as marketing personalization. For example, tagging new content on an e-commerce website or a streaming service will ease browsing and allow for better content recommendations.

This showcase explains how to achieve high quality classification using Pathway.

Why use Pathway for classification?

In a traditional streaming system, the classification of a query is done based on the available data at the time of the query. However, as time goes on, the available data grows, providing a better insight on the classification tasks done on previous queries. The prediction which made sense with a partial view of a data can be seen as wrong with more complete data.

Pathway guarantees classifications with the most up-to-date-model. Under the hood, the system does this by automatically revisiting the classifications of past queries in the stream as new training data becomes available.



Unlike a classic streaming system, Pathway updates the previous query as data arrive, resulting in a lower error rate.

The source code in this article is completely self-contained. With Pathway installed, you can run it directly!

How does Pathway perform classification?

Classifiers are just regular Pathway library functions. In Pathway's standard library, we provide you with a choice of neat classifiers for your convenience - but if you prefer to create your own, be our guest: the logic will only take a couple dozen lines of Python in our framework. As with all code written in Pathway, Pathway takes care of making sure classifiers work correctly on top of data streams.

In this showcase, we will show you how to use Pathway to make your own classification app. We will be using the kNN+LSH classifier from Pathway's standard library: if you are interested about how it works, you can find out more about those topics in our [article](#) about it.

And here comes: Your real-time classification Data App in Pathway

In Pathway, everything you need to perform efficient real-time classification is already implemented for you.

All you need is to load your data, and use our provided classifier functions to train the model and label your queries.

Let's take a look how Pathway performs on a real-time classification problem. The kNN+LSH classifier we will use in this case is available with several metrics, such as cosine or Euclidean distance - we stick to the defaults.

Connecting to streaming data

To illustrate how Pathway performs on real-time classification, we use Pathway to classify handwritten images fed into the system in streaming mode.

We will use the well-known [MNIST](#) as an example. MNIST is composed of 70,000 handwritten digits, each image has a 28x28 resolution and is labeled.

However, we work here with a **streaming data set**: we suppose that the MNIST data and the queries are arriving in a streaming fashion. In streaming, the data is incomplete and the stream progresses over time, until the full data is received.

As MNIST is so standard, we provide a standard loader which simulates just such a data stream. Both the data and the queries are fed in at the same rate, with a 6:1 ratio of data to queries. (For production deployment of your application, you would normally use Pathway's input connectors instead of the simulator.)

```
import pathway as pw
```

```
X_train_table, y_train_table, X_test_table, y_test_table = pw.ml.datasets.class
```

Setting up classification

Here comes the actual training and classification source code, in Pathway.

```
lsh_index = pw.ml.classifiers.knn_lsh_train(  
    X_train_table, d=28 * 28, L=10, M=10, A=0.5, metric="euclidean"  
)  
predicted_labels = pw.ml.classifiers.knn_lsh_classify(  
    lsh_index, y_train_table, X_test_table, k=3  
)
```

What does this code do? We first show how to prepare the classifier with the function `knn_lsh_classifier_train` . (Under the hood, this logic computes the LSH index, i.e. the LSH projector and the buckets of the training data.)

Then `lsh_index` is used to predict the label for the queries (`X_test_table`) using the function `knn_lsh_classify` .

What about parameter choices? The dimension $d = 28 * 28$ is simply the pixel size of the classified images - something specific to MNIST. The other parameter choices ($M = 10, L = 10, A = 0.5$) are set to provide the sweet spot between quality and efficiency: the number of comparisons LSH does for computing the kNN of a query over the full dataset is 830 on average instead of 60,000! Don't hesitate to play with the settings, you'll get the hang of the right "strings to pull" in a couple of minutes.

Now, the resulting labels are then compared to the real ones to measure the accuracy of our approach with our accuracy function `classifier_accuracy` .

```
accuracy = pw.ml.utils.classifier_accuracy(predicted_labels, y_test_table)
```

That's it! You can now preview the `accuracy` table during stream execution, and see how the classification outcomes improve as the stream progresses:

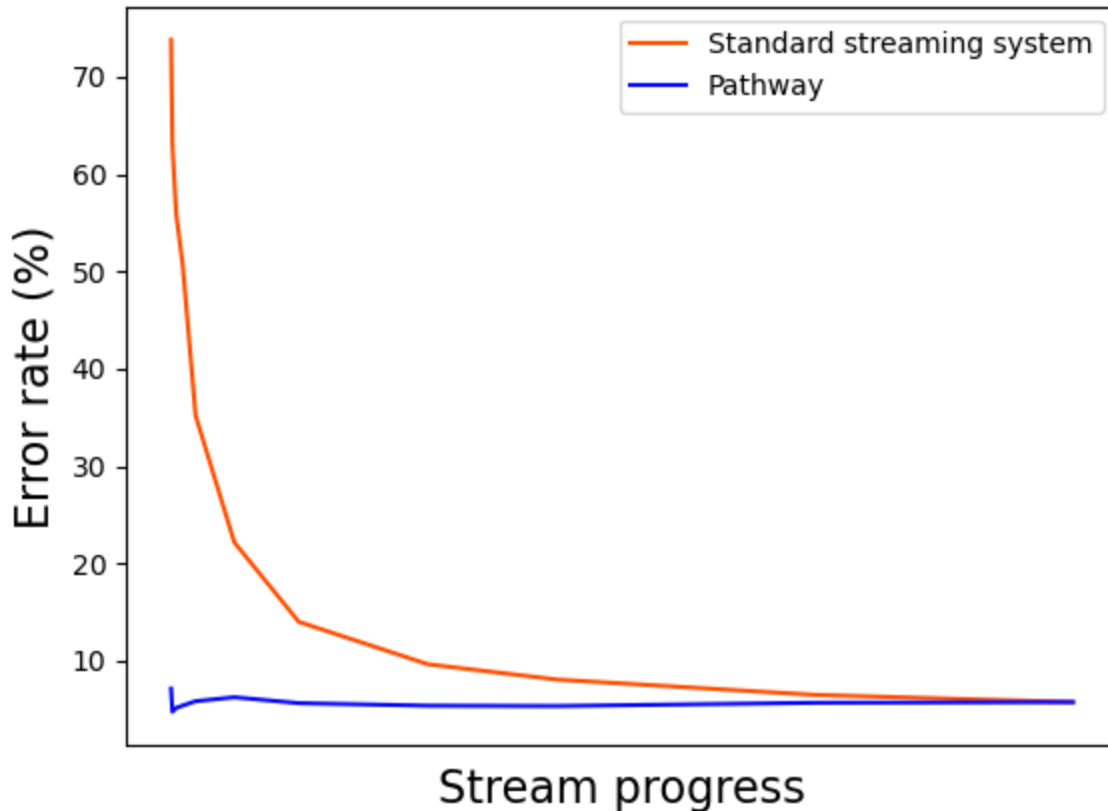
```
pw.debug.compute_and_print(accuracy)
```

```
[2024-01-20T18:44:25]:INFO:Preparing Pathway computation
```

[Table of Contents](#) >

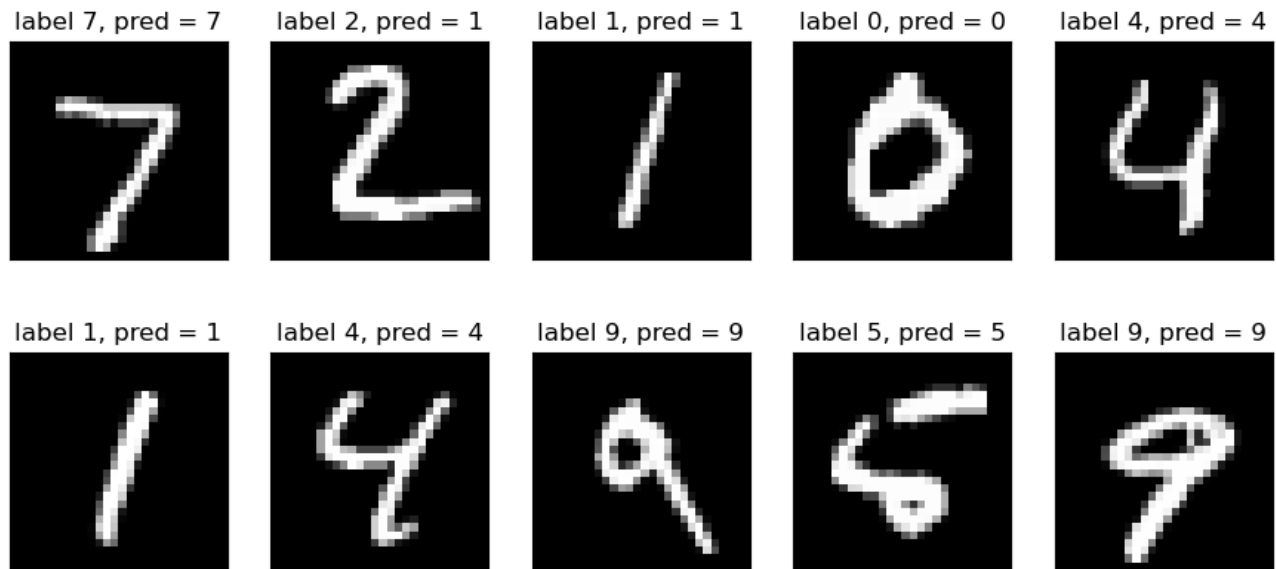
We obtain 9453 correct classifications out of the 10000: an error rate of 5.5%.

Results



As we can see, a normal streaming system exhibits poor performances at first due to an incomplete data set.

Pathway, on the other hand, improves the accuracy of those previous queries by revisiting its predictions at each update. As the data grows, its error rate decreases until it converges to an error close to 5%. Here is a sample of outcomes we get at the end of the stream.



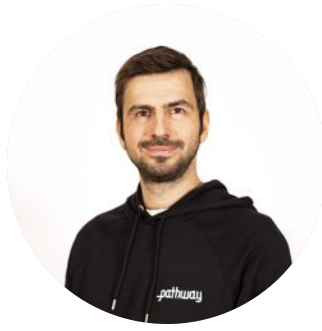
Conclusion

When doing classification on a stream in Pathway, the model is kept up to date automatically. As the model improves over time, results get better and previous classification decisions are also updated to the most up-to-date-model, without having to worry about it.

In many streaming scenarios, the kNN+LSH approach we used provides a sweet-spot between speed and quality. If you want to know more about how all of this works, you can read our [article](#) about it.

In our next articles, we will show you how to use Pathway to build streaming recommender systems and real-time anomaly detection. (Coming soon.)

Related articles



Olivier Ruas

Algorithm and Data Processing Magician



#Classification #KNN #LSH #index #Locality Sensitive Hashing #MNIST
#euclidean distance

C

F

S

C

C

Events

Share this article



Showcases

← Use LLMs for notific...

Lsh

Realtime Classification with Nearest Neighb... →

Developers

Documentation

Tutorials

Showcases

About

Legal & GDPR

Equal opportunity employer

Privacy policy

Licensing

Media kit

Glossary

Contact

Let's talk

Chat with us on Discord

Pathway

96bis Boulevard Raspail

Agoranov

75006 Paris, France

contact@pathway.com

© 2021-2023 Pathway