towards
data science

Using Complex Networks to improve Machine Learning methods

Using the High-Level Data Classification Algorithm to leverage topological characteristics of the data

Tiago Toledo Jr. 2 days ago · 6 min read

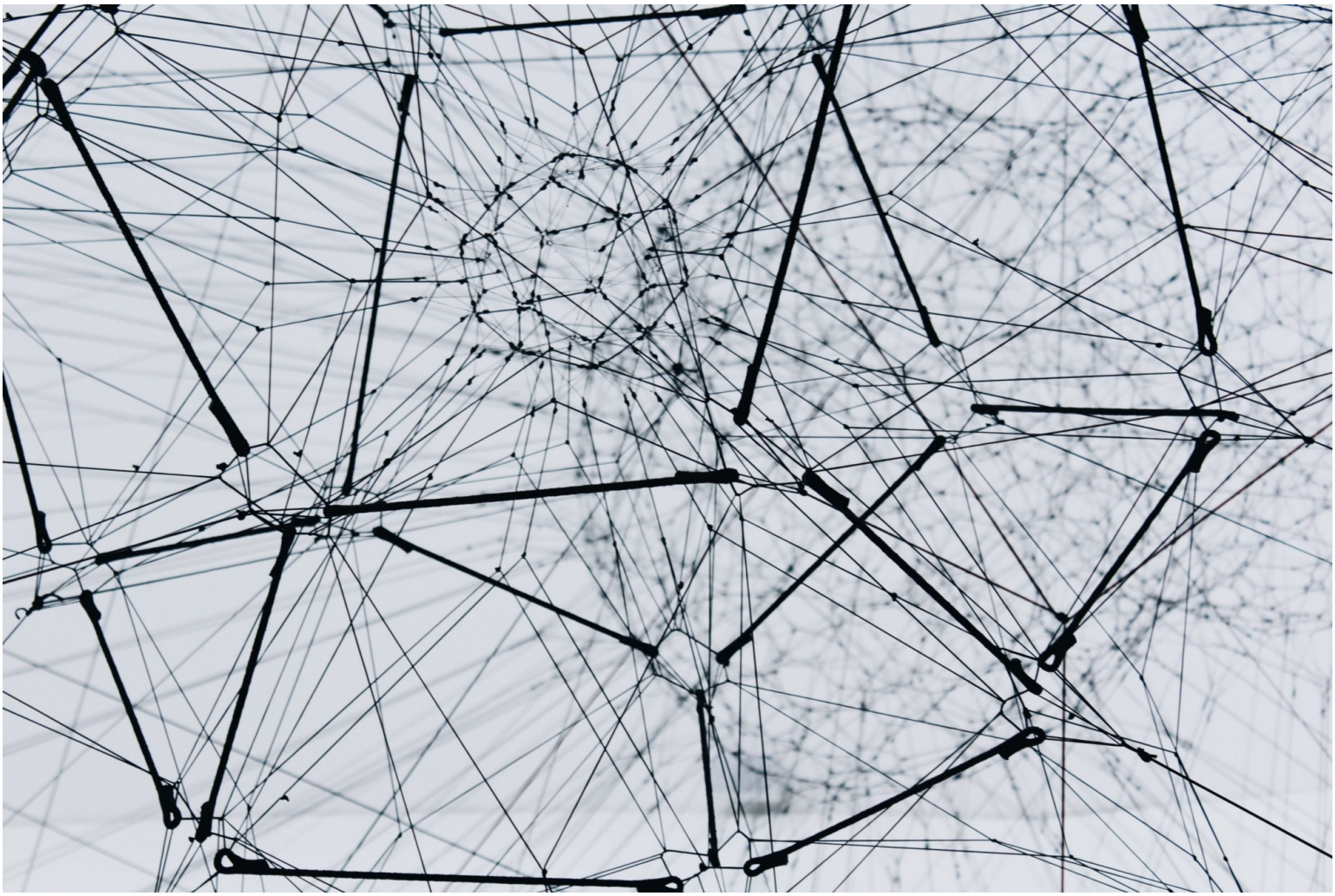

Photo by Alina Grubnyak on Unsplash

Data Science provides lots of different paradigms to a data scientist toolbox such as supervised learning, unsupervised learning, time series, reinforcement learning, and so on. It is impossible to specialize in all of these paradigms, however, having a basic and workable knowledge of each one may help us all become more proficient in problem-solving.

One recent area that has been providing insightful tools for data scientists is complex

networks, an area focused on studying complex phenomena based on relationships between entities.

In this post, I will tell you a bit about complex networks and how you can use them together with machine learning algorithms to improve the performance of your predictions using the sknet library.

Complex Networks

Let's start by defining what a complex network is: a collection of entities called nodes connected between themselves by edges that represent some kind of relationship.

If you're thinking: this is a graph! Well, you are correct, most complex networks can be considered a graph. However, complex networks usually scale up to thousands or millions of nodes and edges, which can make them pretty hard to analyze with standard graph algorithms.

There is a lot of synergy between complex networks and the data science field because we have tools to try and understand how the network is built and what behavior we can expect from the entire system. Because of that, if you can model your data as a complex network, you have a new set of tools to apply to it.

In fact, there are many machine learning algorithms that can be applied to complex networks and also algorithms that can leverage network information for prediction. Even though this intersection is relatively new, we can already play around with it a bit.

So, to start our journey towards joining these two areas, let's see now how we can transform some tabular data into a complex network and how we can classify it.

Transforming Data into Complex Networks

There are many ways of constructing a complex network from tabular data. Here, I will describe a KNN Construction method that uses the closest neighbors from each data point to construct the network.

The algorithm is basically as follows given a tabular dataset:

- Select a value K
- Every row of your dataset is a point in the space and therefore will be a node
- Get every point in space and calculate the distance to the other points
- Select the K closest nodes from the node you are evaluating and draw an edge between them

The idea is to connect the closest points. Notice that this method will not produce singletons (nodes without connections) because every node will have a distance to another one.

Side Note: One may require that when transforming data with classes, that each class is on a separate component. This is a requirement for some algorithms.

The fragility of this method happens when you have dense regions, with many nodes close to each other. It is fair to imagine that a dense region should be densely connected too, but the KNN constructor will only create K edges for each node.

There are other algorithms that try to solve this problem. One example is the Epsilon Radius constructor that will work well on dense regions, however, will be bad on sparse regions generating singletons. One can then combine both methods and use the Epsilon-Radius KNN Constructor. All of these algorithms are available on the sknet library.

High-Level Data Classification

Now that we can transform our data into a complex network, what do we do with it? Well, there are many things we can do but here I will focus on one specific algorithm: high-level data classification.

The High-Level Data Classification algorithm tries to incorporate the findings from traditional Machine Learning algorithms, such as SVMs and Random Forests, with the structural pattern recognition promoted by analyzing the metrics of a complex network.

It basically consists of two parts:

- A low-level classifier, a fancy name for the traditional machine learning algorithms we use such as SVMs or Boosting Algorithms
- A high-level classifier, a complex network-based classification mechanism

The model then combines both of the results to generate the predictions. The idea here is that the complex network mechanism is able to identify topological aspects of the data that the low-level classifier will not be able to identify easily.

But how does this complex network mechanism works? Basically, for each prediction point, we add it to each class component of the network and verify how some metrics of that component vary. If little variation happens, then we are confident that the point did not change the structure of the component and maybe from there. However, if the metrics change a lot, the point changed the structure of the component and probably does not belong there.

One can control how much influence the high-level classification will have on the final prediction by setting a parameter called p, which is a value between 0 and 1 defining the high-level impact.

More details about how the High-Level Data Classification works can be found on [1].

Coding

Let's see how we can use the sknet library to transform our tabular data into a complex network and then leverage the structural pattern recognition to try and improve our standard machine learning algorithms.

The first step before going into any coding is to install the sknet library using pip:

```
pip install scikit-net
```

Now, let's take a look at the code:

```
from sklearn.datasets import load_wine
from sklearn.metrics import accuracy_score
from sknet.network_construction import KNNConstructor
from sknet.supervised import HighLevelClassifier
```

```
X, y = load_wine(return_X_y = True)
X_train, X_test, y_train, y_test = train_test_split(X, y,

test_size=0.2)
knn_c = KNNConstructor(k=5)
classifier = HighLevelClassifier()
classifier.fit(X_train, y_train, constructor=knn_c)

pred = classifier.predict(X_test)
print(accuracy_score(y_test, pred))
```

As you can see the code is fairly simple, but let's look at each section to understand what is happening:

```
X, y = load_wine(return_X_y = True)
X_train, X_test, y_train, y_test = train_test_split(X, y,

test_size=0.2)
```

Here we are just using the sklearn library to import the Wine dataset and then to split it into a holdout test set of 20% of the data.

```
knn_c = KNNConstructor(k=5)
classifier = HighLevelClassifier()
classifier.fit(X_train, y_train, constructor=knn_c)
```

In the first row, we are creating a KNN Constructor which will be responsible for transforming the Wine dataset into a complex network. On the next line, we are instantiating the High-Level Classifier class, the same way we would do with a classifier from sklearn.

Finally, we fit the classifier on the training data. Notice that we pass the constructor to the fit function so it will know how to transform the data during the fit. This is a standard approach for every algorithm in the library.

```
pred = classifier.predict(X_test)
print(accuracy_score(y_test, pred))
```

Finally, we just use the fitted classifier to predict the labels of the test set and compare it with the true labels.

Notice here that I did not parametrize the classifier. I used the default parameters for simplicity, however, we are able to control which low-level classifier we are going to use, how much weight we should put onto the high-level prediction, which metrics we should use on the high-level classification, and so on. A list with every parameter can be found here.

More about the sknet library

The sknet library was developed aiming to close the existing gap in the implementation of machine learning algorithms on complex networks. It already provides tools for transforming tabular data into complex networks, as you saw in this post, but also to transform time series into networks. It also implements supervised, unsupervised, and semi-supervised algorithms that can be used to learn from complex networks or to

improve existing methods.

Hope you enjoy it and test the lib!

[1] Silva, T.C., Zhao, L, Network-based high level data classification (2012), IEEE Trans. Neural Netw. Learn. Syst. 23(6)

Thanks to Elliot Gunn.

Complex Networks Machine Learning