Outline

Cover Page

Summary / abstract

Previous research

Protein information

Peptides

MS2 data

-----------

Embeddings

Embeddings are used to create vectors of “sentences”. These sentences can then be used to group similar sentences. We are trying to use these sentences as input to a neural network. The network will be designed to take in the x-values of MS2 data which is the M/Z values as a sentence. The embeddings will group similar sentences and therefore help the neural network to decide which protein label is the best match

Our Model

Model 1:

Embeddings layer

Dense connected layer

Dense connected layer

Dense output layer

Model 2:

Embeddings layer

Convolution1d layer

Pooling layer

Convolution1d layer

Pooling layer

Flatten layer

Dense Layer

Dense output layer

Mass Spectrum Protein Matching with Neural Networks

Group 2

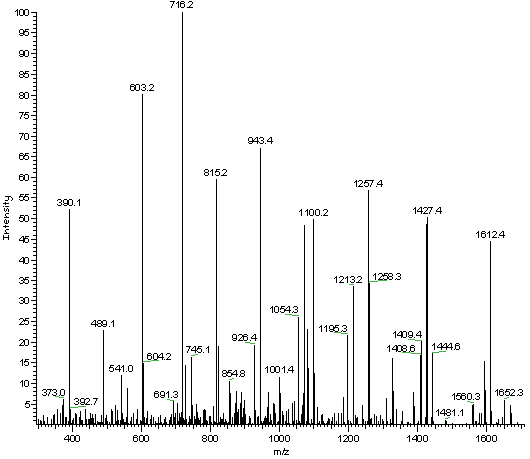
Jonah Kubath

Matt Peter

Abstract

Increasing the rate at which we can identify proteins via their peptide sequences is one of the many computational problems present in the world today. Over the past few decades, protein mass spectrometry has become the primary way of identifying proteins. However, due to the complexity and quantity of proteins, as well as the large amount of “noise” that comes with gathering data, identifying one with an extremely high degree of certainty quickly becomes a daunting task. To solve this problem, large amounts of research have been put into finding ways to decrease the amount of time it takes to identify proteins, while still maintaining a decent level of accuracy. For our project, we decided to specifically focus on using mass-to-charge ratios to assist a neural network in labeling mass spectrometry data with the given protein. Unfortunately, due to the scale and complexity of proteins and the limited time and resources we had, our results did not provide any supporting evidence that neural networks are indeed a viable path to protein identification.

In the field of proteomics, “the study of all proteins in a biological system”, mass spectrometry is a technique that is used “to detect, identify and quantitate molecules based on their mass-to-charge (m/z) ratio” [1]. In order to do this, a protein must first be broken down into a set of peptides using enzymes. After this is done, the first stage of mass spectrometry (MS1) takes place, where peptides are further broken down into ions using an ion source and are separated by m/z ratio. In the second stage of mass spectrometry (MS2), ions of a particular m/z ratio are selected and fragmented, creating fragment ions which are then separated and detected [2]. This collection of fragment ions can then be quantified in MS2 data as “spectra”, a collection of “peaks” that shows the ions’ corresponding intensities at various m/z ratios.



MS2 Data

This MS2 data is then analyzed in order to determine what the original peptides and proteins were. Unfortunately, only a small percentage of the peaks in a spectrum are useful in determining its peptide, as lot of the peaks are either noise or simply not helpful for the given spectrum [3]. This means that a large portion of the time spent analyzing the spectra does not actually improve the end result.

The main layer of the neural network chosen in this project is an embedding layer. Embedding layers work to represent the input “words” in a vector space. As more examples are learned, the layer will move the words in the vector space to group similar words closer together. This technique of word embeddings has been used in other areas of research such as document classification or finding similar words. As an example, when embeddings layers are used in regular text documents, the words “king” and “queen” will be close to each other in the vector space as they are commonly used in similar sentences / situations. The Keras API is built upon TensorFlow which offers an Embedding layer. The inputs to this layer must be integer values. The string or words in sentences can be tokenized with the Keras Tokenize class and then fed into the embedding layer.

The data in this report was modeled around the use of mass spectrometry data (MS2). The yeast protein database was chosen as ground truth training examples. This database is widely selected and used in all areas of proteomics as a standard. In order to generate MS2 data for the proteins, the peptides, which are sub-strings of the protein, must be also be generated. In other areas, the peptides are limited to strings that are six to fifty characters in length. These substrings were generated and labeled with the original protein identification. The peptides were then fed through a mass simulator that would create the theoretical MS2 data for the given peptide. Once this step was complete, the input data was cleaned by binning all the m/z values to the nearest integer value.

Example MS2 m/z data: 10.005, 13.02, 20.08, 47.09

Cleaned MS2 m/z data: 10, 13, 20, 47

The binned input data was then read in and concatenated to form the sentences for a given protein sequence. A given sentence would be similar to the cleaned MS2 m/z data listed above.

The designed neural network started with an embedding layer. This embedding layer would accept tokenized versions of the cleaned MS2 data. This layer would then learn the similarity of the vectors and group them. These weights would then become the input to the remaining layers of the network. Two main methods were tested for the remaining portion of the network. The first being densely connected layers. These layers create “n” number of nodes and each node is connected to every weight of the previous layer. In our implementation, the range of nodes in a dense layer was varied from 50 to 5000. The layer depth was also varied depending on the number of nodes in the layers due to computation time. When layers had 1000 to 5000 nodes, the number of layers was kept low in the ranges of two to four. When layers had 50 to 1000 nodes, the number of layers were varied from two to twelve. To put it into perspective, some of the highest computational tests had over 120 million parameters with most of the smaller tests containing roughly 20 million parameters. The second method that was used to build upon the embedding layer was convolutional layers. Convolutional layers were used to drastically reduce the computation time as the number of parameters is dropped. The convolutional layers were stacked on top of each other with a pooling layer in between. The number of pooling layers was also varied from a single pooling layer to a pooling layer after every convolutional layer. The convolutional layers allow for a smaller window of the previous data to be used as input to the given nodes. Convolutional neural networks are commonly used in other areas were small sections of the data can be analyzed. Ackermann says, “This applies well to the analysis of time sequences of sensor data (such as gyroscope or accelerometer data). It also applies to the analysis of any kind of signal data over a fixed-length period (such as audio signals). Another application is NLP (although here LSTM networks are more promising since the proximity of words might not always be a good indicator for a trainable pattern).” [1]

[1] <https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf>