CS6024: Computational Biology Course Project

Disease detection using metagenomic data

Bersilin C (CS20B013) & Yuva Sai Tej A (CS20B002)
Computer Science and Engineering, IIT Madras

May 15, 2024

Abstract

A diverse array of microbiomes exists independently or in symbiosis throughout our surroundings. Identifying the various microbes within a microbiome is crucial for comprehending, altering, or addressing the microbiome. Analyzing the soil microbiome allowed us to identify biomarkers for plant diseases and has facilitated the development of medications for prevention and treatment. The **human gut microbiota** represents a complex consortium of microorganisms cohabiting symbiotically within the human gastrointestinal tract. The varying prevalence of numerous species within it aids in discerning metabolic disparities between healthy individuals and those affected. We intend to utilize insights gleaned from analyzing metagenomic data, particularly the proportional presence of various species within a microbiome, to identify diseases by applying deep learning models such as feed-forward neural networks. Implementing deep learning models instils a strong optimism regarding the potential for successfully detecting diseases from metagenomic data.

1 Introduction

Let's cover some key terms relevant to the project, the rationale behind applying deep learning for disease detection, and outline the specific problem we aim to address. We'll also introduce the dataset utilized and review prior research in the field of disease detection employing metagenomic data.

1.1 Metagenomic Data

Metagenomics is a revolutionary approach in the field of molecular biology that involves the study of genetic material recovered directly from environmental samples. It enables researchers to explore the genetic diversity, structure, and function of entire microbial communities without isolating and culturing individual organisms. It emerged as a response to the limitations of the traditional microbiology techniques, which heavily relied on culturing microbes in the laboratory. Metagenomics allows researchers to access and study these unculturable microorganisms by directly analyzing their genetic material.

It starts with collecting environmental samples ranging from soil and water to human gut or extreme environments like deep-sea vents. Next is the extraction of DNA from the sample. This DNA represents the collective genetic material of all the organisms in the sample, providing a snapshot of the microbial community. Next, we sequence the data using high-throughput sequencing technologies like nextgeneration sequencing (NGS) or shotgun sequencing, generating vast amounts of sequencing data, requiring sophisticated bioinformatics analysis tools.

Metagenomic data serves multiple purposes, one of which is acquiring the relative abundances of various genes and species present in a sample. This information is instrumental for conducting diverse analyses and studies. MetaPhlAn [1] is one tool utilized for acquiring such data. It is a computational tool for profiling the composition of microbial communities (Bacteria, Archaea, and Eukaryotes) from metagenomic shotgun sequencing data at the species level. In this work, we use the species-level abundance information generated by MetaPhlAn for the samples collected from the human gut and skin. We use this to study the difference in abundance levels of various microbes between a healthy and unhealthy individual.

1.2 Human Gut Microbiome

The human gut microbiome generally refers to the diverse community of microorganisms, including bacteria, viruses, fungi and archaea, that inhabit the gastrointestinal tract. These microbes play a crucial role in human health and are involved in various physiological processes, including digestion, nutrient metabolism, immune function and even neurological signalling. This microbiome is incredibly diverse, comprising trillions of microbial cells belonging to thousands of species. The composition of the gut microbiome can vary significantly between individuals and is influenced by factors such as diet, genetics, age, geography, and lifestyle.

As expected, gut microbes are crucial in breaking down complex carbohydrates, fibres, and other indigestible diet components that human enzymes cannot process. Certain gut bacteria also synthesize vitamins (e.g., vitamin K, some B vitamins) and short-chain fatty acids (SCFAs) that provide energy and other beneficial effects on host health. The gut microbiome interacts closely with the immune system, influencing its development, maturation, and function. A diverse and stable gut microbiome can prevent the colonization and overgrowth of harmful pathogens by occupying ecological niches, producing antimicrobial compounds, and modulating the host's immune response.

Alterations in the composition and function of the gut microbiome have been linked to metabolic disorders such as obesity, type 2 diabetes, and metabolic syndrome. Emerging research suggests that the gut microbiome can communicate bidirectionally with the central nervous system through various pathways, collectively referred to as the gut-brain axis. **Dysbiosis** or an imbalance in the gut microbiome composition, has been associated with various diseases and health conditions, including inflammatory bowel diseases (e.g., Crohn's disease, ulcerative colitis), irritable bowel syndrome, allergies, autoimmune disorders, and even neurological disorders.

1.3 Motivation

Sequencing technology advancements and the development of associated analytical tools have catalyzed research in metagenomic data. This progress enhances our capacity to comprehend diverse microbiomes and leverage them to address diseases and subsequent complications stemming from these microbiomes. The association of the imbalance in the gut microbiome composition to the presence of the disease shows the potential of detecting diseases from the species-level abundance information. Several related studies have employed the same concept to detect diseases. One such study is outlined below.

1.4 Previous Work

One of the major works that uses species-level abundance information to detect diseases is Machine Learning Meta-analysis of Large Metagenomic Datasets: Tools and Biological Insights `MetAML`. [2] This paper introduces and analyses classification models for disease prediction from metagenomic data. Specifically, it covers six metagenomic datasets linked to five major diseases: Liver cirrhosis, Colorectal cancer, Inflammatory bowel disease (IBD), Obesity and Type 2 diabetes.

`MetAML` employs cross-study validation, a technique that assesses machine learning models' capability to discern the distribution of diseased samples from that of healthy ones. Initially, the method entails training a model with data from one study to categorize samples as diseased or healthy. Subsequently, this trained model is utilized to classify samples from a distinct study. This process provides insights into the effectiveness of machine learning models in classifying diseased samples, regardless of their study origin, thereby mitigating bias stemming from variations in sampling techniques. The dataset used by the paper is as follows:

Dataset name	Body site	Disease	#case samples	#control sample
Cirrhosis	Gut	Liver Cirrhosis	118	114
Colorectal	Gut	Colorectal Cancer	48	73
HMP	Several	None	-	981
IBD	Gut	Inflammatory Bowel Diseases	25	85
Obesity	Gut	Obesity	164	89
Skin	Skin	None	-	287
T2D	Gut	Type 2 Diabetes	170	174
WT2D	Gut	Type 2 Diabetes	53	43

Table 1: Information about the different datasets used in `MetAML`

`MetAML` used several approaches to detect diseases and biomarkers in various disease datasets and explored machine-learning approaches like Support Vector Machines (SVMs) and Random Forests (RFs) to learn the distribution of species-level abundances. The paper suggests trying out more complex machine-learning models and deep learning for detecting diseases. The paper also doesn't classify samples in a multi-class setting trying to find the particular disease from any new sample. We want to address these two as the problem statement in this project.

1.5 Dataset

The dataset utilized for our project experiments is sourced directly from the GitHub repository referenced in the `MetAML` paper [2]. It comprises processed metagenomic sequencing data provided by `MetaPhlAn` [1], presenting species-level abundance details of microbes found in the human gut microbiome. This dataset encompasses over 3600 samples, each associated with approximately 210 metadata points and 3300 abundance features. Importantly, the data originates from various studies. More information about the dataset is as follows:

Dataset name	Positive	Negative	Total	Positive classes	Negative classes
Zeller Fecal Colorectal Cancer	87	47	134	3	1
WT2D	102	43	145	2	1
Vertical Transmission Pilot	0	17	17	0	1
T2D 2 (Short)	35	38	73	1	1
T2D 1 (Long)	135	155	290	1	2
Obesity	16	35	51	3	1
Human Skin	0	291	291	0	1
Quin Gut Liver Cirrhosis	118	114	232	1	1
Psoriasis	36	34	70	1	1
IBD 1	148	234	382	2	2
IBD 2	25	85	110	2	1
E-Coli Stec2 1	9	0	9	1	0
E-Coli Stec2 2	43	1	44	1	1
Hmpii	0	219	219	0	1
Hmp	0	762	762	0	1
Chatelier Gut Obesity	164	114	278	1	2
Candela Africa	0	38	38	0	1

Table 2: Information about the dataset

1.6 Problem Statement and Approach

The problem statement for this project is: **Identification of diseases from species-level abundance information using multi-layered perceptrons (MLP) / feed-forward neural networks (FFNN)**. We plan to tackle this problem on three levels.

- Single-disease binary classification: Learn one distribution from one subset of the dataset and classify the samples into positive and negative. If the subset has multiple positive or negative classes, they are combined into single positive and negative classes. In this, we aim to determine if the samples collected in a single study are coherent with each other and recognize how a model can learn the difference between a positive and negative sample. We also want to find which specific species are present in varying abundances between a positive and negative sample.

- Multi-disease binary classification: Performing binary classification after combining all the positive and negative samples from all the different studies. We want to try a simple model architecture (a neural network) with varying complexities in terms of the depth of the model and the count of neurons present in each layer to understand the actual complexity of the distribution.

- Multi-disease multi-class classification: We want to classify all the samples into their respective disease classes without combining all the diseased samples into a single positive class. All the negative samples were combined into a single negative class. This way, we can identify if the different diseases have different separable distributions. This also gives us some idea of whether a relatively complex model like a neural network can learn all the different distributions from skewed prior data.

2 Method and Results

2.1 Single-disease binary classification

We analyzed all 17 datasets from different studies separately. Some datasets did not have separate positive and negative samples; some only had case (positive) samples, while some had only control (negative) cases. We removed those for this analysis. Finally, we were left with 11 datasets. We did some data analysis to find if some particular species were present in different abundances in the positive and negative samples and checked the ones that had the maximum difference. We did the following for each subset of the dataset from various studies:

1. Separate the positive and negative samples
2. Take the average abundance of all the samples in each class across all the features; let them be positive and negative averages (each is a list of numbers with length: number of features)
3. Find the features that have the most difference in abundance between the positive and negative averages
4. Plot the top 20 features that had the maximum difference

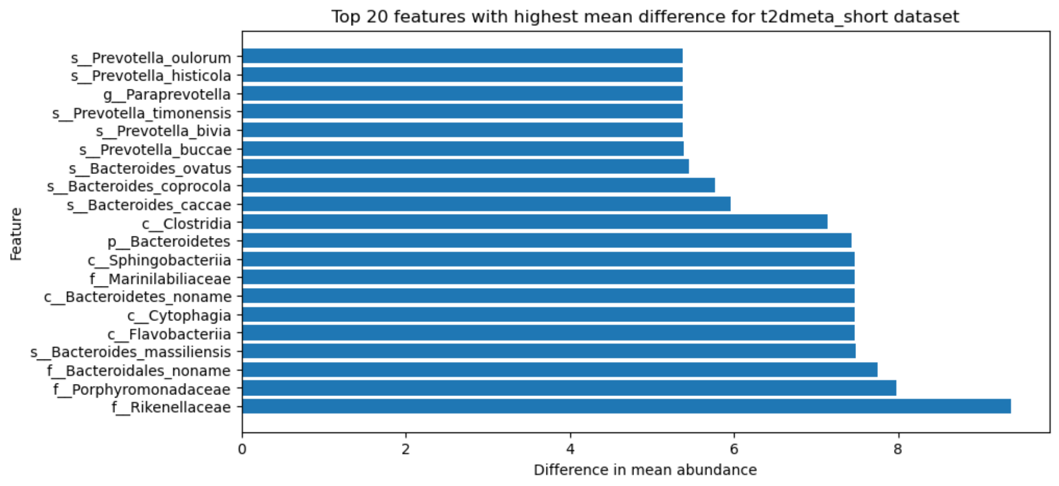

Figure 1: Absolute difference in mean abundances

We build a simple neural network using PyTorch in Python to learn the binary classification model. We used Adam as the optimizer and Cross Entropy Loss as the loss function to calculate the loss against the generated probabilities. We then performed an extensive hyperparameter tuning to find the best hyperparameters for each of the dataset subsets. We considered the following hyperparameters to explore:

```
hyper_parameters = {
    'lr': [1e-5, 1e-5, 1e-4, 5e-4, 1e-3],
    'batch_size': [4, 8, 16, 32],
    'layers': [[128, 64], [256, 128, 64], [512, 256, 128, 64]],
    'n_epochs': [200, 300, 400]
}
```

Dataset name	Layers	Learning Rate	Train Accuracy	Test Accuracy
Zeller Fecal Colorectal Cancer	[512, 256, 128, 64]	1e-05	1.000	0.778
WT2D	[128, 64]	1e-05	0.983	**0.828**
Quin Gut Liver Cirrhosis	[256, 128, 64]	1e-05	0.935	**0.894**
Psoriasis	[512, 256, 128, 64]	0.001	0.946	1.000
IBD 1	[128, 64]	5e-05	1.000	**0.922**
IBD 2	[256, 128, 64]	5e-05	1.000	**1.000**
E-Coli Stec2 2	[128, 64]	1e-05	1.000	1.000
Chatelier Gut Obesity	[128, 64]	1e-05	0.977	**0.696**
T2D 2 (Short)	[128, 64]	1e-05	1.000	**0.933**
T2D 1 (Long)	[128, 64]	0.0001	0.991	**0.759**
Obesity	[256, 128, 64]	5e-05	1.000	**0.818**

Table 3: Performance on the subsets

Dataset name	MetAML accuracy	Our test accuracy
Zeller Fecal Colorectal Cancer	0.805	0.778
WT2D	0.703	**0.828**
Quin Gut Liver Cirrhosis	0.877	**0.894**
IBD	0.809	**0.922, 1.000**
Obesity	0.644	**0.818, 0.696**
T2D	0.664	**0.933, 0.759**

Table 4: Comparison of performance with the paper

2.1.1 Inferences

Lower learning rates performed better than higher values of learning rates. In 9 out of 11 subsets, $1e-5$ and $5e-5$ were the best learning rates. In 2 cases, the best learning rates were $1e-3$ and $1e-4$. There was no particular trend in the values of batch size and the number of epochs needed to get the best validation accuracy.

When we look at the number of layers and the depth of the neural network, we can see that most of the subsets perform better with a shallow (less number of hidden layers) and a narrow network (less number of neurons per layer). This could result from the dataset having an easily separable distribution and the model being able to learn it without over-fitting the samples. Deeper and broader models usually overfit the training samples, lowering test accuracy.

2.2 Multi-disease binary classification

We wanted to learn the ability of a neural network to identify, given a sample, whether an individual is diseased or not. All the positive and negative samples were combined into a single dataset for this experiment. Finally, we had 2700 control (negative) and 900 case (positive) samples. As discussed in the dataset section, the given data has the problem of having many dimensions (3300 per sample), making the learning process for the model and interpretation harder. We tried training the model with all the features as input. The model did not learn with 200-300 epochs and classified most samples as negative. This is generally considered bad (false negatives) in a biological setting where the positive samples are usually far fewer than negative samples and usually also has many features.

To tackle the issue of high dimensionality, we decided to use feature extraction methods. One of the most popular dimensionality reduction techniques is the Principal Component Analysis (PCA) technique.

2.2.1 Principal Component Analysis (PCA)

Principal Component Analysis is a technique used for dimensionality reduction and data visualization. It transforms high-dimensional data into a lower-dimensional space while preserving the most essential information. The main objective of PCA is to find a new set of orthogonal axes, called principal components, along which the data varies the most. By projecting the data onto these principal components, PCA effectively reorients the data in a way that captures the maximum variance in fewer dimensions. Each principal component is a linear combination of the original features. There are also some limitations of PCA. PCA assumes linear relationships between variables and may perform poorly with nonlinear data. Outliers can significantly affect the results.

Applying PCA to our gave rise to some interesting observations. The data had a total of around 3300 features. After PCA, we observed that 95% of all the data variance could be explained using only 705 features, which is 21% of the total features. The figure 2 on page 7 explains this. We also assume that removing some variance from the data removes outliers and stabilizes the learning process for the model.

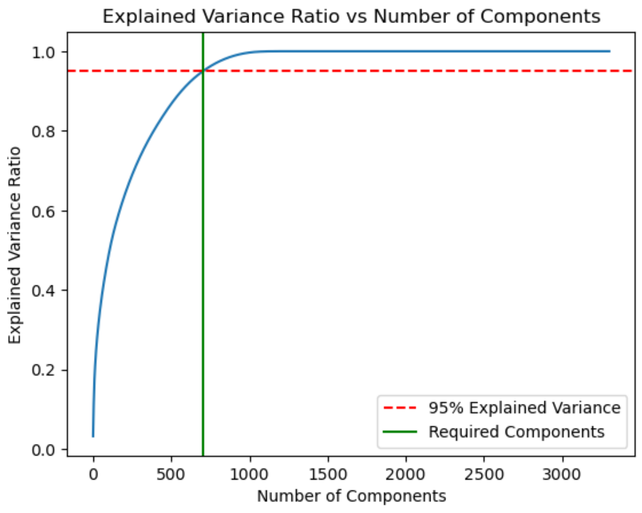

Figure 2: Retention of variance by PCA

We train a neural network to perform binary classification using this newly obtained low-dimension data. We also perform hyperparameter tuning with the following set of hyperparameters. A total of 510 models were trained.

```
hyperparameters = {
    "layers": [
        [1024, 512, 256, 128, 64, 64, 32, 32],
        [1024, 1024, 512, 512, 128, 64, 32],
        [1024, 512, 256, 128],
        [512, 256, 128, 64, 32],
        [256, 128, 64, 32],
        [128, 64, 32],
        [64, 64, 32],
        [32, 32],
    ],
    "batch_size": [8, 16, 32, 64, 128],
    "learning_rate": [1e-4, 5e-4, 1e-05, 5e-05, 1e-06],
    "epochs": [100, 200, 300, 400, 500],
}
```

2.2.2 Results and inferences

The best hyperparameters we obtained for this task were: learning rate: $1e-5$, epochs: 200, layers: $[256, 128, 64, 32]$, batch size: 64. We obtained a test accuracy of **82.96%**.

When we analyze the best set of hyperparameters, we see that a relatively shallow and narrow network can better understand the data distribution. The effect of the learning rate is significantly related to other parameters like model layers and the batch size used. Plateau in the validation accuracy is reached at around 200 epochs.

2.3 Multi-disease multi-class classification

In this section of our experiments, we wanted to analyze the ability of the neural network to learn the distribution of all the different diseases in a very skewed distribution. The distribution of the disease classes is as follows:

Disease	Count	Percentage
healthy	2692	74.57
obesity	164	4.54
stec2 positive	52	1.44
ibd ulcerative colitis	148	4.10
ibd crohn disease	25	0.69
psoriasis	36	1.00
cirrhosis	118	3.27
obese	5	0.14
overweight	10	0.28
underweight	1	0.03
t2d	223	6.18
impaired glucose tolerance	49	1.36
cancer	48	1.33
small adenoma	26	0.72
large adenoma	13	0.36

Table 5: Performance on the subsets

A total of 180 models were trained. The same set of hyperparameters were used to tune the models with the following values:

```
hyperparameters = {
    'batch_size': [32, 16, 8],
    'lr': [0.0001, 0.00005, 0.00001],
    'layers':[
        [1024, 512, 256, 128],
        [1024, 512, 256, 128, 64, 64, 32, 32],
        [1024, 512, 256, 128, 64, 64, 32],
        [1024, 512, 256, 128, 64, 64],
        [1024, 512, 256, 128, 64]
    ],
    'epochs': [200, 300, 400, 500]
}
```

2.3.1 Results and inferences

The best hyperparameters we obtained for this task were: learning rate: $5e-5$, epochs: 500, layers: $[1024, 512, 256, 128]$, batch size: 32. We obtained a test accuracy of **80%**. As in the other two tasks, the performance is better in shallower models, denoting that the distribution of each class is not very complex.

3 Models and technical details

As mentioned earlier, all the code was written using the PyTorch framework in Python. This uses the `torch.nn.Module` class as the base model. PyTorch's `nn` module provides a high-level abstraction for building neural networks. Internally, it's implemented using a combination of Python classes and C++ backend, leveraging the efficient computation offered by the Torch library. PyTorch's automatic differentiation engine, Autograd, provides the foundation for efficiently computing gradients. PyTorch employs various techniques for optimizing computational graphs, such as operator fusion, to improve performance and reduce memory usage.

The model architecture used for the experiments in this project:

```python
class MLP(nn.Module):
  def __init__(self, idim, odim, layers, batch_norm=True):
    super(MLP, self).__init__()
    self.layers = nn.Sequential()

    for i, layer in enumerate(layers):
      if i == 0:
        self.layers.add_module("fc{}".format(i), nn.Linear(idim, layer))
      else:
        self.layers.add_module("fc{}".format(i), nn.Linear(layers[i-1], layer))

      if batch_norm:
        self.layers.add_module("bn{}".format(i), nn.BatchNorm1d(layer))

    self.layers.add_module("relu{}".format(i), nn.ReLU())
    self.layers.add_module("fc{}".format(len(layers)), nn.Linear(layers[-1], odim))
    self.layers.add_module("softmax", nn.Softmax(dim=1))

  def forward(self, x):
    return self.layers(x)
```

Very general train and test modules were used to improve modularity.

```python
def train(model, data_loader, criterion, optimizer):
    """
    Generic training function
    """
    model.train() # Set model to training mode
    running_loss = 0.0
    for _, data in enumerate(data_loader):
        inputs, labels = data
        inputs, labels = inputs.to(device), labels.to(device)

        optimizer.zero_grad() # Zero the gradients

        outputs = model(inputs) # Forward pass
        loss = criterion(outputs, labels) # Compute loss
        loss.backward() # Backward pass
        optimizer.step() # Update weights

        running_loss += loss.item()
    return running_loss / len(data_loader)

def evaluate(model, data_loader, criterion):
    """
    Generic evaluation function
    """
    model.eval() # Set model to evaluation mode
    running_loss = 0.0
    correct = 0
    total = 0
    true_labels = []
    predicted_labels = []
    with torch.no_grad():
        for _, data in enumerate(data_loader):
            inputs, labels = data
            inputs, labels = inputs.to(device), labels.to(device)

            outputs = model(inputs)
            loss = criterion(outputs, labels)

            running_loss += loss.item()
            _, predicted = torch.max(outputs, 1)

            true_labels.extend(labels.cpu().numpy())
            predicted_labels.extend(predicted.cpu().numpy())

            correct += (predicted == labels).sum().item()
            total += labels.size(0)
    return running_loss / len(data_loader), correct / total, true_labels,
    predicted_labels
```

References

[1] Truong, D. T., Franzosa, E. A., Tickle, T. L., Scholz, M., Weingart, G., Pasolli, E., Tett, A., Huttenhower, C., and Segata, N. Metaphlan2 for enhanced metagenomic taxonomic profiling. *Nature Methods* (2015).

[2] Waldron, L., Edoardo, T., Pasolli, D. T., Malik, F., and Segata, N. Machine learning meta-analysis of large metagenomic datasets: Tools and biological insights. *PLOS Computational Biology 12*, 7 (07 2016), 1–26.