Thesis Formulation Report
Meta-Data Prediction with DNA Sequences

Jordan Taylor

September 2019

Chapter 1

Introduction

The focus of the Thesis Formulation Report (TFR) is the exploration and classification of a severely underdetermined count dataset in the context of DNA sequencing. This TFR gives context to the problem, a review of *traditional* mathematical techniques, motivation for the use of machine learning, followed by preliminary results obtained over the summer period.

DNA can be thought of as a set of instructions for a particular organism. In the DNA double helix, the four chemical bases bond to form "base pairs". Adrenaline (A) always pairs with thymine (T) and cytosine (C) always with guanine (G). This pairing is the basis for the mechanism by which DNA molecules are copied when cells divide. The human genome contains approximately 3 billion base pairs that specify the instructions for making and maintaining a human being.

Consider the case: a human is infected with a bacterial virus, which is often referred to as phage. The full name "bacteriophage" means "bacteria eater" because bacteriophages destroy the host cells. A phage attaches itself to a bacterium and infects the host cell. Following infection, the phage hijacks the bacterium to prevent it from producing bacterial components and instead, forces the cell to produce viral components. After some time, new phages assemble and burst out of the bacterium in a process called lysis [Nature News, 2019].

In conjunction with a contracting bacteria such as Escherichia coli (E. coli), the phages will cause the spread of bacteria for both the infected human and possibly their surroundings depending on the type of E. coli. Though E. coli is commonly found in human and animal intestines, and forms the normal gut, most are actually harmless. The strains that are harmful such as O157:H7, however can cause nausea, vomiting, diarrhoea, and among the young and elderly, anaemia and dehydration. E. coli is normally contracted through contaminated food or water, especially raw vegetables and undercooked beef. Healthy adults normally recover from O157:H7 within a week, but the young and elderly have a greater risk of it developing into life-threatening situations such hemolytic uremic syndrome, a form a kidney failure [Mayo Clinic, 2019].

The Shiga toxin (Stx) is one of the most potent bacterial toxins known and has various subtypes. Stx is found in Shigella dysenteriae 1 and in some serogroups of E. coli (called Stx 1 in E. coli). In addition or instead of Stx1, some E. coli strains produce the second type of Stx, Stx2, that has a a similar mode of action as Stx1 but antigenically distinct (antibody produced against one strain will react with the other) [Melton-Celsa, 2014].

From a DNA sequence, could one infer what sort of phages or Shiga toxins are present or any other meta-information. This report considers this statement from a (statistical) machine learning viewpoint. It aims to not only "classify" the meta-information, but to give some interpretability through feature importance to explain which DNA sequences should be present given a specific meta class. Since the human genome has far too many permutations to consider or correctly sequence in totality, we consider counts of much smaller lengthed base pairs ranging from length 9 to 100. Currently provided from the University of Bath's Department of Biology & Biochemistry, are datasets from the years 2017 and 2018. In the 2017 dataset, there are approximately 5 million base pair count features and 570 independent observations, and in the 2018 dataset, over 7 million base pair features and 660 observations.

Chapter 2

Literature Review

Mathematically phrased, the problem we are looking to solve is: given a dataset $\mathbf{X} \in \mathbb{R}^{n,m}$ has n observations each having a vector of m features, we wish to classify each of the n observations into one of the p classes in $\mathbf{Y} \in \mathbb{R}^{n,p}$. Each row of $\mathbf{Y}$ will only contain the values 0 or 1 to indicate which of the p classes that row belongs to. The biggest challenge is to do so for when $m \gg n$ i.e. a severely under-determined problem. This chapter considers methods that fit a classification model to the dataset or reduce the number of dimensions first before model fitting from a non-machine learning point of view.

2.1 Logistic Regression

In [James et al., 2013], the author defines the popular *logistic regression* model to be:

$$p(X) = \frac{\exp(\beta_0 + \beta_1 X)}{1 + \exp(\beta_0 + \beta_1 X)} \tag{2.1}$$

where X is a single value. In the case we want to apply this to our dataset for all the observations at once, we can write this as:

$$\begin{aligned} \phi(\mathbf{X}) &= \frac{\exp(\mathbf{w_0} + \mathbf{XW})}{1 + \exp(\mathbf{w_0} + \mathbf{XW})} \\ &= \frac{1}{1 + \exp(-\mathbf{w_0} - \mathbf{XW})} \\ &= \frac{1}{1 + \exp(-\tilde{\mathbf{X}}\tilde{\mathbf{W}})} \end{aligned} \tag{2.2}$$

where $\tilde{\mathbf{X}} = [\mathbf{1}, \mathbf{X}]$ and $\tilde{\mathbf{W}} = [\mathbf{w_0}, \mathbf{W}^{\mathrm{T}}]^{\mathrm{T}}$ with $\mathbf{W} \in \mathbb{R}^{m,p}$. The notation has been switched where ϕ is used to represent the logistic function previously denoted by p and $\mathbf{W}$ for $[\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_m]^{\mathrm{T}}$ to be more consistent with later literature. Additionally, for notational purposes, we will assume the bias $\mathbf{w}_0$ is implicit where possible and use the notation $\mathbf{X}$ and $\mathbf{W}$ to mean $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{W}}$ from this point on.

From a statistical angle, the objective function to be maximised is the likelihood. Considering the instance we are only predicting if an observation belongs or not belongs to the j-th class, we can compute the likelihood as:

$$L_j(\mathbf{W}) = \Pi_{i=1}^{i=n} \phi(\mathbf{x}_i \mathbf{W})^{y_{ij}} \left(1 - \phi(\mathbf{x}_i \mathbf{W})\right)^{1 - y_{ij}}$$

or for all classes:

$$L(\mathbf{W}) = \Pi_{j=1}^{j=p} L_j(\mathbf{W})$$

The term $L(\mathbf{W})$ quantifies (in probability space) how well $\mathbf{W}$ in conjunction with ϕ estimates $\mathbf{Y}$. This can be phrased as maximising $\Pr(\mathbf{Y} = \mathbf{y}_i | \mathbf{X} = \mathbf{x}_i, \mathbf{W})$ by changing $\mathbf{W}$. Often the negative log-likelihood is

minimised as this not only the same as maximising the likelihood, but the mathematics becomes easier to derive gradients for iterative methods to update $\mathbf{W}$.

Though the logistic model can be explained nicely using various statistical theories, the main drawback is that it does not model interactions between the m features. Rather it takes some linear sum and applies the logistic function. For data that have dependencies, the logistic model may under perform compared to other models that consider these interactions. In the case the data is linearly independent, or at least modelled to be, then by examining the values in $\mathbf{W}$, we can quantify how important (or not) a feature is for each class.

2.2 Decision Tree

The decision tree algorithm creates a tree-like model with **decision** and **leaf** nodes. The decision node is a query, and depending on the answer, the model traverses through one of it's branches. A query is simple a rule and if a sequence of rules are obeyed, we arrive at the terminal leaf node which tells us a classification. See below for an illustration.

Figure 2.1: Image courtousy of https://www.saedsayad.com/decision_tree.htm

The challenge for the decision tree is how to determine the correct queries. The three main criterion to choose from are:

- gini impurity
- information gain
- variance reduction

Gini Impurity is a measure of how often a randomly chosen observation from the current branch would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the proposed new branch. By defining the proportion of the data classified class i as z_i the gini impurity is computed by:

$$\mathrm{I}_g(z) = \sum_{i=1}^{i=p} z_i(1 - z_i) = \sum_{i=1}^{i=p} z_i - z_i^2 = 1 - \sum_{i=1}^{i=p} z_i^2 \tag{2.3}$$

The first part of the equation is quantifying the probability of the true class being class i whilst the predicted class is not i. Through some simple manipulation, we arrive at the tidied right hand side.

Information Gain quantifies how much information is gained by a proposed decision node. Information gain is quantified by the difference between the current entropy and the weighted proposed entropy. For an observation $\mathbf{X}$ and the associated classification in the form of a one-hot encoded labels $\mathbf{Y}$, entropy is computed by:

$$H(\mathbf{x}) = \mathrm{I}_e(\mathbf{Y}) = -\sum_{i=1}^{i=p} z_i \log_2 z_i \tag{2.4}$$

where z_i is still defined as the proportion of the data that is classified as class i. If we imagine the decision tree decision node to query the data at the current point and split it into a left and right branches, information gain is then quantified by the difference of the current entropy, $H(\mathbf{X})$, and the weighted sum of the left and right conditional entropies, $H(\mathbf{X}_L|z_L)$ and $H(\mathbf{X}_R|z_R)$ where z_L and z_R are the proportions of the data being split into the left (true) and right (false) branches. Formally, information gain is computed by:

$$\begin{aligned}\mathrm{IG}(\mathbf{X}, z_L, z_R) &= H(\mathbf{X}) - H(\mathbf{X}_L|z_L) - H(\mathbf{X}_R|z_R) \\ &= -\sum_{i=1}^{i=p} z_i \log_2 z_i - \sum_{z\in\{z_L,z_R\}} z \sum_{i=1}^{i=p} -\Pr(i|z_i)\log_2 \Pr(i|z_i)\end{aligned} \tag{2.5}$$

By choosing queries with the highest information gain, we keep the decision tree relatively *shallow* to reduce class impurity efficiently.

Variance Reduction though normally used for regression trees, attempts to find the best split such that the weighted sum of variances is minimised. The computation for this criteria is defined as:

$$\mathrm{I}_v(\mathbf{y}) = \sum_{\mathbf{z}\in\{\mathbf{z}_L,\mathbf{z}_R\}} |\mathbf{z}|^{-2} \sum_{i\in\mathbf{z}} \sum_{j\in\mathbf{z}} (y_i - y_j)^2 \tag{2.6}$$

where $\mathbf{z}_L$ and $\mathbf{z}_R$ are defined to be the indicies of observations that go left or right at the decision node [StatSoft Inc., 2013].

Overall, the decision tree is an easy to interpret intuitive model with clear decision boundaries. The downside is that the first few decision nodes will target the most impure classes in order to *sift* through the data efficiently. This results in a few class balancing decision nodes before

2.3 Dimensionality Reduction

The main non-machine learning methods for dimensionality reduction are the Principle Component Analysis (PCA) and Non-negative Matrix Factorisation (NMF). Since both methods can be used to visualise high dimensional data in 2 or 3 dimensions, we can inspect scatter plots of data in the attempts to visualise clustering. The main limitation of clustering using these methods, are the methods transform the data linearly, and so non-linearly related data will typically be clustered poorly.

Principle Component Analysis is method to linearly transform our data into an orthogonal space such that the order of dimensions is determined by the variance in each projected dimension. There are two different views on how to generate such a transformation, and both considers a "centered" matrix. This matrix can be obtained by first subtracting the mean of each column which yields $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ to be the covariance matrix. The first view uses eigenvalues and eigenvectors, and to maximise the variance in the first projected dimension, we seek some $\mathbf{v}_1$ such that:

$$\begin{aligned}\mathbf{v}_1 &= \arg\max_{\mathbf{v}} ||\mathbf{X}\mathbf{v}||_2^2, \quad ||\mathbf{v}||_2^2 = 1 \\ &= \arg\max_{\mathbf{v}} \frac{\mathbf{v}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{v}}{\mathbf{v}^{\mathrm{T}}\mathbf{v}}\end{aligned} \tag{2.7}$$

which is known as the Rayleigh quotient. By considering the eigenvalue problem, $\mathbf{Xv} = \lambda\mathbf{v}$ we see that $\mathbf{v}_1$ must be the eigenvector associated with the highest absolute eigenvalue. Further components can be shown to be the remainder of the eigenvectors and eigenvalues ordered by absolute eigenvalues. At this point, we can transform $\mathbf{X}$ into some orthogonal space:

$$\mathbf{T} = \mathbf{XV}, \quad \mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_m]$$

By truncating $\mathbf{V}$ to be the first k columns instead of the entire m columns, we can transform $\mathbf{X}$ into some lower dimensional space:

$$\mathbf{T}_k = \mathbf{XV}_k, \quad \mathbf{V}_k = [\mathbf{v}_1, ..., \mathbf{v}_k], \quad k < m$$

The second way of performing PCA is to use the Singular Value Decomposition (SVD). The SVD factors $\mathbf{X} = \mathbf{U\Sigma V}^{\mathrm{T}}$ where $\mathbf{U} \in \mathbb{R}^{n,\min(n,m)} = [\mathbf{u}_1, ..., \mathbf{u}_{\min(n,m)}]$ are referred to as left-singular vectors with $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$, $\mathbf{V} \in \mathbb{R}^{\min(n,m),m} = [\mathbf{v}_1, ..., \mathbf{v}_m]$ as right-singular vectors with $\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}$, and $\mathbf{\Sigma} = \mathrm{diag}(\sigma_1, ..., \sigma_{min(n,m)})$ where $\sigma_1 \geq ... \geq \sigma_{min(n,m)} \geq 0$ as singular values. The orthogonal transform is then the same as the method of using eigenvectors and eigenvalues.

Non-negative Matrix Factorisation decomposes $\mathbf{X} = \mathbf{WH}$ assuming all three matrices do not contain any negative values. Though it was introduced in [Paatero and Tapper, 1994], it was properly motivated and made popular by [Lee and Seung, 1999] and [Lee and Seung, 2000] through uses such as image processing, text mining, and hyperspectral imaging. The idea is we want to decompose $\mathbf{X}$ into a low rank approximation where $\mathbf{W} \in \mathbb{R}^{n,k}$ has k basis dimensions which can be thought of as the important features, and $\mathbf{H} \in \mathbb{R}^{k,m}$ are the sets of weights to to linearly transform the features back into the original data.

The initialisation of $\mathbf{W}$ and $\mathbf{H}$ can be arbirary though there question of what objective function to minimise comes to mind. The authors consider two main objectives depending on the data and/or task context: the Frobenius norm and the Kullback-Leibler (KL) divergence. The NMF has the interesting property of clustering $\mathbf{X}$ if the Frobenius norm objective function is used. Furthermore, if we additionally add the constraint of orthogonality for $\mathbf{H}$ i.e. $\mathbf{HH}^{\mathrm{T}} = \mathbf{I}$ then the minimisation task is similar to that of the K-means clustering algorithm which is covered later.

In the case of minimising the Frobenius norm, [Lee and Seung, 2000] shows the *multiplicative update rule*:

$$\mathbf{H}_{ij}^{n+1} \leftarrow \mathbf{H}_{ij}^{n} \frac{\left((\mathbf{W}^n)^{\mathrm{T}}\mathbf{X}\right)_{ij}}{\left((\mathbf{W}^n)^{\mathrm{T}}\mathbf{W}^n\mathbf{H}^n\right)_{ij}}$$

and:

$$\mathbf{W}_{ij}^{n+1} \leftarrow \mathbf{W}_{ij}^{n} \frac{\left(\mathbf{X}\left(\mathbf{H}^{n+1}\right)^{\mathrm{T}}\right)_{ij}}{\left(\mathbf{W}^n\mathbf{H}^{n+1}\left(\mathbf{H}^{n+1}\right)^{\mathrm{T}}\right)_{ij}}$$

with the stopping condition as both $\mathbf{H}^n$ and $\mathbf{W}^n$ becoming stable.

2.4 Discussion

When examining the data for correlations between the features and one of the meta data classifications, the correlations seem low. This tells us the features either is unrelated to the classifications or may have non-linear relations as correlation is a linear statistic.

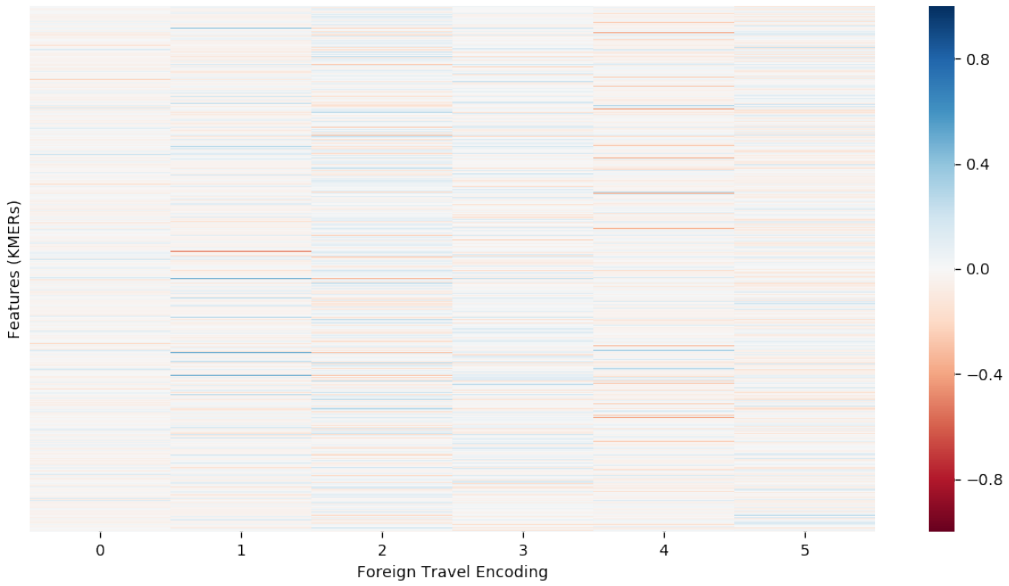

Figure 2.2: Cross correlation heatmap between features and one of the sets of meta classifications

From Figure 2.2, we see that most of the colors seem very discoloured and so, it is expected that logistic regression and PCA will have weak performance for the classification of *foreign travel*. We therefore desire methods that can take into consideration non-linearity.

Generally, the predictive models for classification described earlier in this chapter maximise the conditional log likelihood of the classifications given the observations and model parameters. Though this is exactly what we want to achieve, we more importantly desire this property for unseen datapoints. By simply maximising the log likelihood of some training set of datapoints, how can we expect a fully trained model will be able to yield similar accuracy results to some unseen test dataset? Assuming we have a large amount of training datapoints that sufficiently represent the distribution of the total dataset, it may be better to maximise the *expectation* of the log likeihood. Given that we sample from the data uniformly the cost, C, can be computed by:

$$\begin{aligned}
C &= -\mathbb{E}\left[\log \Pr(\mathbf{y}|\mathbf{x},\theta)\right] \qquad \text{(negative as to consider the negative log-likelihood)} \\
&= -\sum_{\mathbf{x}\in\mathbf{X},\mathbf{y}\in\mathbf{Y}} \Pr(\mathbf{x},\mathbf{y}) \log \Pr(\mathbf{y}|\mathbf{x},\theta) \\
&= -\sum_{\mathbf{x}\in\mathbf{X},\mathbf{y}\in\mathbf{Y}} \Pr(\mathbf{x},\mathbf{y}) \left[-\log \frac{\Pr(\mathbf{y}|\mathbf{x})}{\Pr(\mathbf{y}|\mathbf{x},\theta)} + \log \Pr(\mathbf{y}|\mathbf{x})\right] \\
&= \mathbb{E}\left[\log \frac{\Pr(\mathbf{y}|\mathbf{x})}{\Pr(\mathbf{y}|\mathbf{x},\theta)}\right] - \mathbb{E}\left[\log \Pr(\mathbf{y}|\mathbf{x})\right] \\
&= D_{\mathrm{KL}}(\mathbf{Y}||\hat{\mathbf{Y}}) + H(\mathbf{Y}|\mathbf{X}) \qquad (2.8)
\end{aligned}$$

where θ are the model parameters, $D_{\mathrm{KL}}(\mathbf{P}||\mathbf{Q})$ the KL divergence, and $H(\mathbf{Y}|\mathbf{X})$ the conditional entropy. Note that the conditional entropy is not a function of θ and can be ignored in the optimisation process. The

big assumption is that the training datapoints sufficiently describe the distribution of all datapoints. The cost to be minimised is then simply the distance measure between $\mathbf{Y}$ and our model estimate $\hat{\mathbf{Y}} = f(\mathbf{X})$ in expected distribution space. From an implementation point of view, this justifies the use of uniformly subsampling the data. For the logistic model, this means using stochastic or batched gradient descent methods over total gradient descent, and in the case of the decision tree, make many trees out of subsamples of the data to form the *Random Forest* model discussed in the next chapter.

As most algorithms have computational complexity as functions of the number of features (usually beyond linear complexity), it is desired to reduce the dimensionality of high dimensional datasets into something easier to model. Looking at Figure 2.2 again, we see that our data may have non-linear relationships such that correlation fails to identify this correctly. How do we determine what non-linear function to consider and in what proportions? This drives the need for a way to perform feature extraction automatically in the case of a high dimensional problem.

Chapter 3

Methodology

The applications of machine learning are extremely broad and, in many cases, are shown to be the state-of-the-art models excelling in areas such as imaging and natural language processing. In context with our DNA application problem, we consider the use of machine learning for: a predictive model for classification, and for dimensionality reduction. Namely, the models used for classification are the **Random Forest** and **Gradient Boosting** models, and for dimensionality reduction: **t-distributed Stochastic Neighbour Embedding** (t-SNE) and the **Autoencoder**.

3.1 Ensemble Methods

Note that even though we consider the use of machine learning, we still desire to preserve a sense of interpretability. Starting with the use of the Random Forest and Gradient Boosting models, they are both ensemble methods using many decision trees. Since the decision tree was intuitive as it is a tree based method with a series of decision nodes, a collection of decision trees should be just as interpretable. Wrong. An ensemble of decision nodes can conflict with each other and from that view, a boundary function may be difficult to extract where a single decision tree's boundary function is completely determined by the collection of decision nodes. Though we cannot extract a deterministic boundary function, we can examine the frequency of the features used in every decision node. If the feature comes up often, we can attempt to quantify feature importance and rank the features accordingly.

3.1.1 Random Forest

A random forest is a collection of decision trees and where both the observations and features are subsampled. The method was founded by [Tin Kam Ho, 1995] then extended by [Breiman, 2001]'s *bagging* technique. Bagging, a term given to *Bootstrap aggregating*, is to subsample (with replacement) the observations in $\mathbf{X}$. This helps to extract useful features through simpler decision node queries as the focus is not on the overall training set class impurities. Consider the case whereby, a subsample only contained some of the classes, any features seen in a decision tree only considering this subsample will compute efficient decision queries for these classes. This may prove useful for when analysing features to distinguish class behaviours.

Bagging combined with [Tin Kam Ho, 1995]'s main idea of sampling features in $\mathbf{X}$ introduces a lot of variability in model predictions. One way of reducing this is to have a large amount of decision trees in the forest. As each decision tree is constructed independently, training can be executed in parallel for computational efficiency.

As there are sampling methods both in the observations and features, the Random Forest model overcomes the natural overfitting problem present in the single decision tree model. A way of regularising further the random forest model is to limit the maximum number of decision nodes from the tree base. This improves generalised performance on unseen datapoints as the queries do not become too specific to the training set topology.

3.1.2 Gradient Boosting

Boosting is a meta algorithm, that is, it takes another algorithm and takes an ensemble. By iteratively fitting a series of weak learners together, we arrive at the gradient boosting algorithm. [Breiman, 1997] spurred the idea of boosting but it was [Friedman, 2002] that led the development of it. Boosting often takes, as it's weak learner algorithm, decision trees to form a boosting forest. Since both the Random Forest and Gradient Boosting models take in an ensemble of decision trees how do they differ?

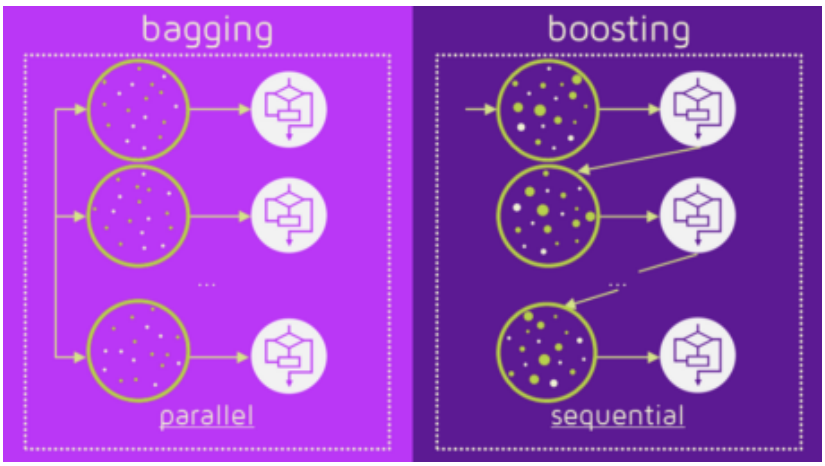

Figure 3.1: Image courtesy of `https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/`

From Figure 3.1, we see that the boosting model dependently builds itself. By measuring the residuals after constructing each decision tree, the boosting model looks to fit a new tree in areas where the residuals indicate need improvement. As this model requires a sequential build, it cannot be parallelised and so, is significantly more expensive in time cost. Without the proper regularisation, a boosting model is more prone to overfit.

3.2 Dimensionality Reduction

As our DNA data is extremely high dimensional, it makes training any predictive model a slow process even on today's top of the range computers. Too reduce this computational burden, we consider the use of dimensionality reduction techniques within machine learning. The two techniques we consider are the t-distributed Stochastic Neighbour Embedding (t-SNE) and the Autoencoder. Unlike PCA which is a deterministic process, both of these models are stochastic dimensionality reduction techniques. As both models are minimising an objective function in a non-linear functional space, the training process may reach different local optimas depending on initialisation and / or random batches used in gradient descent. That being said, the learnt projections in lower dimensional space is often still useful.

3.2.1 t-distributed Stochastic Neighbour Embedding

The t-SNE algorithm was founded and developed by [van der Maaten and Hinton, 2008] and further developed by [van der Maaten, 2009], [van der Maaten and Hinton, 2012], and [van der Maaten, 2014]. The t-SNE algorithm starts by a probabilistic conditional similarity measure as:

$$p_{j|i} = \frac{\exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/\sigma_i^2)}{\sum_{k \neq i} \exp(-||\mathbf{x}_i - \mathbf{x}_k||^2/\sigma_i^2)} \tag{3.1}$$

The authors then justify:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n} \tag{3.2}$$

by saying *the similarity of datapoint x_j to datapoint x_i is the conditional probability, $p_{i|j}$, that x_i would pick x_j as its neighbour if neighbors were picked in proportion to their probability density under a Gaussian centered at x_i.* The probabilities p_{ii} are set to zero as a datapoint cannot pick itself as a neighbour. What we have just described is the *true* distribution we want to approximate in some lower dimensional space. By introducing the variable $\mathbf{z}_i \in \mathbb{R}^d$ in the lower d-dimensional space, we desire our approximations:

$$q_{ij} = \frac{(1 + ||\mathbf{z}_i - \mathbf{z}_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||\mathbf{z}_i - \mathbf{z}_k||^2)^{-1}} \tag{3.3}$$

The locations $\mathbf{z}_i$ are determined by minimising the KL divergence between $\mathbf{P}$ and $\mathbf{Q}$:

$$D_{\mathrm{KL}}(\mathbf{P}||\mathbf{Q}) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{3.4}$$

Examining some examples of use from `https://lvdmaaten.github.io/tsne/`, we see that the MNIST handwritten digits in 2D is clearly well separated:

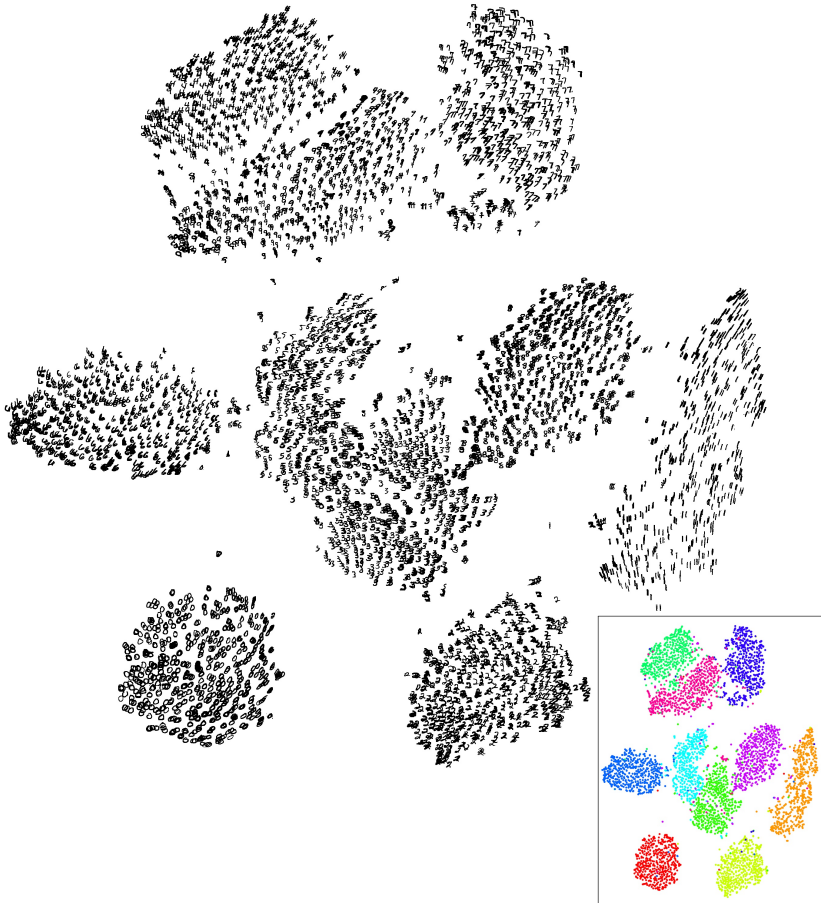

Figure 3.2: Image courtesy of `https://lvdmaaten.github.io/tsne/examples/mnist_tsne.jpg`

and are even better in 3D `https://lvdmaaten.github.io/tsne/examples/mnist_tsne.mov`. An important hyperparameter to be set is perplexity which is defined as a measure for information. Mathematically, it is defined as 2 to the power of entropy. It may be viewed as a knob that sets the number of effective nearest neighbours, where typical values are often in the range between 5 and 50.

3.2.2 Autoencoder

The Autoencoder is a specific form of an (artificial) neural network. It is comprised of two parts, the encoder network, which attempts to encode $\mathbf{X} \in \mathbb{R}^{n,m} \rightarrow \mathbf{Z} \in \mathbb{R}^{n,d}$ where $d \ll m$, and a decoder which approximates $\mathbf{X}$ from the encoded $\mathbf{Z}$.

From the diagram on the right, we see that the original feature space is reduced to some encoded space before being reconstructed back into feature space. Denoting the original and reconstructed features to be $\mathbf{X}$ and $\hat{\mathbf{X}}$ respectively, the Autoencoder has the objective to minimise the distance between $\mathbf{X}$ and $\hat{\mathbf{X}}$ in some metric space. Commonly the objective function is defined as the Euclidean distance or if the values of $\mathbf{X}$ are probabilities, the KL divergence.

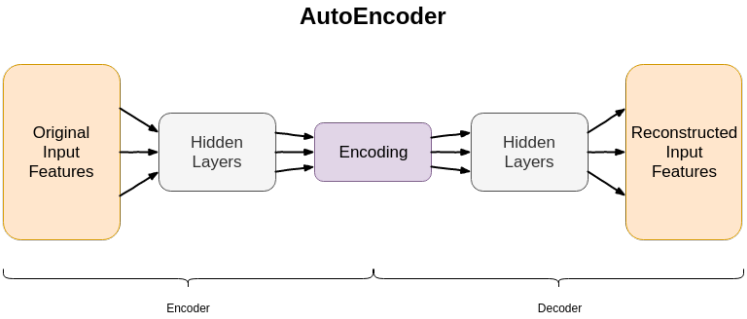

Figure 3.3: Diagram made using `www.draw.io`

Each layer can be defined as some element wise function of a linear operator:

$$\mathbf{L}_{i+1} = \phi_i(\mathbf{L}_i \mathbf{W}_i + \mathbf{b}_i)) \tag{3.5}$$

where ϕ_i is the (non-linear) activation function, $\mathbf{W}_i$ is the projection weights , $\mathbf{b}_i$ is the bias term, and $\mathbf{L}_i$ is the i-th layer in the network. $\mathbf{W}_i$ has less columns than it does rows so long as it is part of the encoder section, and likewise, has more columns than rows when part of the decoder. Gradients can be computed using standard calculus where there will be excessive use of the *chain rule* as the output of the Autoencoder network is defined as many nested functions. Though the gradients are computed using standard calculus, it is known in the community as the *backpropagation* algorithm.

Like with any neural network, the challenges faced here are how to decide the number of hidden layers, the layer sizes, and activation functions. The additional hyperparameter to search over specific to an Autoencoder is the encoding size.

Chapter 4

Findings

4.1 Data

4.2 Methods

4.3 Results

4.4 Analysis

Chapter 5

Future Works

Bibliography

[Breiman, 1997] Breiman, L. (1997). Arcing the edge. Technical report.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[Friedman, 2002] Friedman, J. H. (2002). Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 38(4):367–378.

[James et al., 2013] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning – with Applications in R*, volume 103 of *Springer Texts in Statistics*. Springer, New York.

[Lee and Seung, 1999] Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401:788–791.

[Lee and Seung, 2000] Lee, D. D. and Seung, H. S. (2000). Algorithms for non-negative matrix factorization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, pages 535–541, Cambridge, MA, USA. MIT Press.

[Mayo Clinic, 2019] Mayo Clinic (2019). Symptoms & causes of e. coli. https://www.mayoclinic.org/diseases-conditions/e-coli/symptoms-causes/syc-20372058. [Accessed 20-08-2019].

[Melton-Celsa, 2014] Melton-Celsa, A. (2014). Shiga toxin (stx) classification, structure, and function. *Microbiology spectrum*, 2.

[Nature News, 2019] Nature News (2019). bacteriophage. https://www.nature.com/scitable/definition/bacteriophage-phage-293/. [Accessed 20-08-2019].

[Paatero and Tapper, 1994] Paatero, P. and Tapper, U. (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126.

[StatSoft Inc., 2013] StatSoft Inc. (2013). Popular Decision Tree: Classification and Regression Trees (C&RT). http://www.statsoft.com/Textbook/Classification-and-Regression-Trees. [Accessed 22-08-2019].

[Tin Kam Ho, 1995] Tin Kam Ho (1995). Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1.

[van der Maaten, 2009] van der Maaten, L. (2009). Learning a parametric embedding by preserving local structure. In van Dyk, D. and Welling, M., editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 384–391, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA. PMLR.

[van der Maaten, 2014] van der Maaten, L. (2014). Accelerating t-sne using tree-based algorithms. *Journal of Machine Learning Research*, 15:3221–3245.

[van der Maaten and Hinton, 2008] van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

[van der Maaten and Hinton, 2012] van der Maaten, L. and Hinton, G. (2012). Visualizing non-metric similarities in multiple maps. *Machine Learning*, 87(1):33–55.