3-1-2017 12:00 AM

Using Machine Learning to Predict Chemotherapy Response in Cell Lines and Patients Based on Genetic Expression

Dimo Angelov, *The University of Western Ontario*

Supervisor: Dr. Lucian Ilie, *The University of Western Ontario*
Co-Supervisor: Dr. Peter Rogan, *The University of Western Ontario*
A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science

Abstract

The goal of this thesis was to examine different machine learning techniques for predicting chemotherapy response in cell lines and patients based on genetic expression. After trying regression, multi-class classification techniques and binary classification it was concluded that binary classification was the best method for training models due to the limited size of available cell line data. We found support vector machine classifiers trained on cell line data were easier to use and produced better results compared to neural networks. Sequential backward feature selection was able to select genes for the models that produced good results, however the greedy algorithm has limitations. We found that genetic algorithms and simulated annealing were able to select genes that produced better results on both cell lines and patients. We found that combining cell line data sets from different types of cancers produced models that performed well at predicting outcome in cell lines and in patients, indicating that the method of action of chemotherapy drugs is similar across different types of cancer. The use of cell line trained machine learning models to predict patient chemotherapy response shows great promise, however future studies need to acquire larger cell line data sets and find better ways of evaluating the transfer of predictive ability of cell line trained models to patients.

Keywords: Machine learning, Cancer, Drug sensitivity, Feature selection, SVM, Kernels, Genomic profiles

Contents

List of Figures

Chapter 1

Background

1.1 Introduction

With the availability of more data and knowledge about diseases and people's life styles there is a movement towards personalized medicine, also referred to as precision medicine. This is an emerging approach for the prevention and treatment of diseases that takes into account the difference individuals have in their genetics, their environment and lifestyle. This is possible as a result of improved genome sequencing technologies and improved technologies for biomedical data analysis.

There is a rapidly increasing understanding of cancer biology as well as availability and affordability of high throughput technologies for individual genetic analysis and molecular characterization of tumours. This is facilitating an explosion in research focusing on personalized medicine for cancer patients [25]. Cancer is recognized as being a disease of the genome [9]. Data that shows the abnormalities in genes, proteins and other factors that cause cancer can be used to form machine learning models. They can be used to identify types of cancers, predict drug resistance, and hopefully to identify best treatment strategies.

Being able to predict the response of a patient to a specific therapy is essential in the era of personalized medicine. There are hundreds of different chemotherapy drugs; according to the U.S. National Cancer Institute, for breast cancer alone there are 63 different chemotherapy drugs approved for treatment [17]. Accurate computational prediction of therapy response would assist clinicians in choosing most effective and least toxic treatments [2].

There are many research challenges involved in creating computational prediction models for cancer drug response. The biological challenge is that cancer is a highly heterogeneous disease. The data challenge is the volume, complexity and noise of available data sets. The technological challenge is finding and applying the most appropriate machine learning techniques to the data.

There is publicly available data for thousands of cell lines available through on-line sources such as the Cancer Cell Line Encyclopaedia. One limitation of the available data is that it does not offer extensive cell line collections for all cancer types. Additionally there are limited numbers of cell lines that have all been treated with a specific drug, making it difficult to train, test and evaluate drug specific models. A further challenge is finding data to verify efficacy of cell-line-trained models on patients. One of the most common approaches is train-

ing models on public cell-line-derived data and testing on publicly available patient data [2]. Another approach is training models on privately owned cell-line data and also testing on publicly available patient data. Additionally, there are models that are both trained and tested on patient-derived data.

Once a data set has been selected to be used for building a prediction model, the relevant features of the data must be separated from the irrelevant features. The first step is to choose the relevant data types. Cell-line data usually contains single-nucleotide mutations, gene copy numbers and gene expression. Comparative analyses suggest that gene expression data has the most predictive power [10, 20]. The next step is to perform the feature selection, in the case of gene expression based models the most relevant genes must be selected. There are different methods for doing this, some involve looking at the correlation between gene expression and drug sensitivity measurements, other methods build feature selection into the model training phase [2]. A very common approach is using all of the features available in the data set [11]. There currently isn't a universal solution to the problem of feature selection [2].

There are many different computational models but most commonly derived models are built to be able to predict drug sensitivity for a single drug [2]. Pan-cancer models are models that predict drug sensitivity across multiple types of cancer. Where as cancer-specific models predict drug sensitivity for a specific type of cancer. Recently there have been efforts to form models that predict synergistic effects of several drugs. The most common approach for building machine learning models is supervised learning although unsupervised learning is frequently used for data visualization, data selection, and as an aid to building supervised learning models. There is no general solution for building regression and classification supervised models [20]. Some approaches work better than others, depending on the training data set and measure of drug sensitivity used. No single approach can consistently outperform others on different data sets and across different drugs [2].

There are still many challenges in the development of computational models for predicting chemotherapy response. Personalized medicine will greatly benefit from advances in this area, as cancer continues to be one of the leading causes of death. There is currently no universal solution to feature selection, nor is there a single machine learning approach that outperforms all others; examining which methods work best is essential to making progress towards a universal solution. This thesis focuses on building single drug cancer and pan-cancer chemotherapy machine-learning prediction models. Models were trained on publicly available cell-line data and tested on publicly available patient data. We examined classification and regression techniques for predicting chemotherapy response. We compared support vector machine classification to neural network classification. We also examined different feature selection techniques. This thesis explores the advantages of using support vector machine classification models compared to neural network classification, several types of multi-class classification and regression techniques for limited size cell-line data. We examine the advantages of simulated-annealing and genetic algorithms for feature selection, compared to the very commonly used sequential backward feature selection algorithm. We also examine combining cell line data sets from different types of cancers and training models on them produced models that performed well at predicting cell-line and patient chemotherapy response.

1.2 About Cancer

Cancer is one the leading causes of death according the the World Health Organization [24]. In 2012 there were 14 million new cases and 8.2 million cancer related deaths. According to the 2014 World Cancer Report [24] annual cancer rates will increase from 14 million in 2012 to 22 million by 2032, which is a 70% increase. Cancer puts a huge toll on people who are affected by it. Those with cancer require therapies such as surgery, chemotherapy, radiotherapy, and hormonal therapies. In some parts of the world these therapies have immense costs associated with them, and put a huge burden on people and their families. Many people have trouble coping with the acceptance of having cancer, and develop mental health problems such as depression and anxiety as a result. Cancer and its treatments also take a toll on the body of patients, and leave them feeling weak and in pain. For many it results in death, and leaves family members to cope with the loss.

Cancer is the disease where certain cells in the body start to divide without stopping and grow out of control. Normally the cells in the human body only divide when the body needs to replace them, and there are mechanisms that control their division. Cancer is a genetic disease; it occurs when there are anomalies in the genes of a cell, that regulate its functions. These genetic anomalies are usually the result of mutations that can be caused by many different factors such as sun exposure, use of tobacco, consuming carcinogens, obesity, and even viruses. When a mutation occurs in a cell, the body usually destroys these damaged cells, but sometimes it does not, that is when cancer occurs. Mutations in the part of the DNA that are responsible for regulating cell growth and division are ones that lead to cancerous cells. Once a cell becomes cancerous it can grow into a malignant tumour, which is a collection of cancerous cells. Figure 1.1 shows the formation of a cancerous tumour. Metastasis is another danger of cancer, it occurs when cancerous cells can also split off from the tumour and travel to different areas of the body and infect them.

1.3 Chemotherapy

1.3.1 What is Chemotherapy?

One of the most common types of cancer treatment is chemotherapy. Chemotherapy is the use of anticancer drugs that kill cancer cells by disrupting their ability to grow and undergo cell division. Chemotherapy destroys cells that are undergoing rapid division; since that is a property of cancer cells, they are targeted. However, they are not the only target. Chemotherapy also kills other normal cells that divide rapidly, such as skin cells, bone marrow cells, hair follicles, and the lining of the digestive system. Thus there are quite many side effects from the use of chemotherapy, such as: fatigue, pain, loss of appetite, hair loss, changes in thinking and memory, and nausea.

1.3.2 Chemotherapy Drugs - Paclitaxel

There are hundreds of different chemotherapy drugs; for breast cancer alone there are over 60 different types. One of the most common and effective chemotherapy drugs is called Paclitaxel.

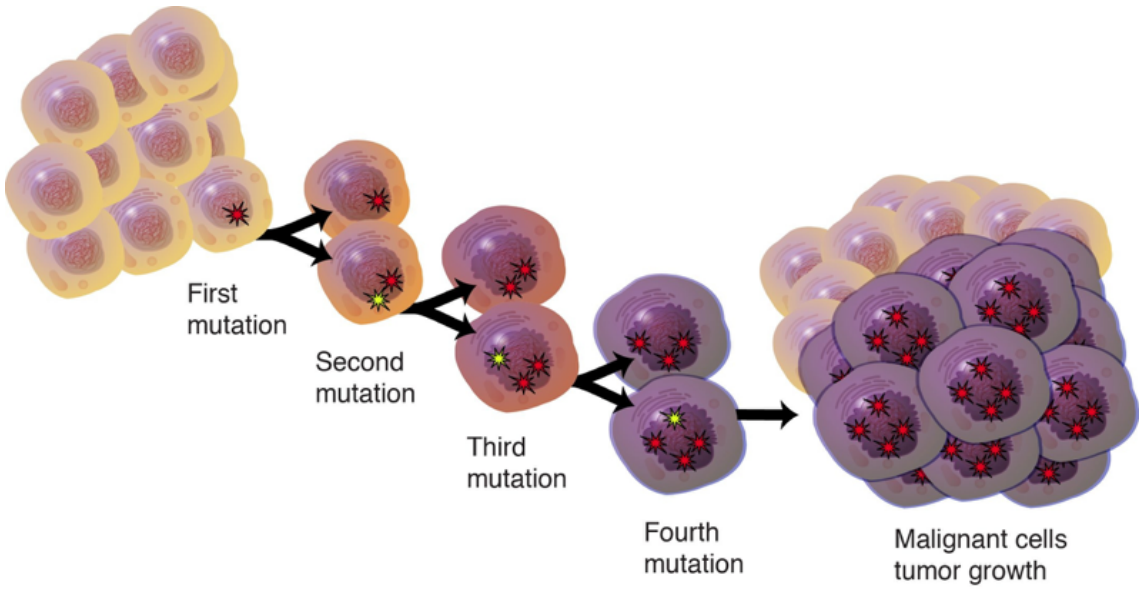

Figure 1.1: Cancer cell growth. (National Human Genome Research Institute.)

It is used to treat ovarian, breast, lung, pancreatic, and other types of cancers. Paclitaxel was discovered in a very interesting way: The U.S. National Cancer Institute funded a screening program where random plants extracted from the wild were tested for their abilities to treat cancer; the bark of the pacific yew tree turned out to be quite effective [33]. The L atin name of the Pacific yew tree is Taxus brevifolia, thus the name of the drug became Taxol, later branded as Paclitaxel. What is interesting is that Paclitaxel is on the list of the World Health Organization's list of essential medicines, which consists of the most important medication needed in a basic health system.

Every chemotherapy drug has a different chemical composition, way it is taken, cancer type it is most effective on and set of side effects [29]. Chemotherapy drugs can be grouped by their method of action, chemical structure, and relationship to other drugs [29]. Some drugs may fall into several categories since they work in more than one way. Knowing the method of action of a drug is very important for predicting its efficacy and side effects. Paclitaxel falls in the category of mitotic inhibitors, which are derived from natural products and they work by stopping the cell from dividing [29]. Paclitaxel is a cytoskeletal drug, which means it is a small molecule that interacts with the protein tubulin [33]. Every cell has microtubules within it, which are small tubes which have an important role in cell division, cell movements, and preservation of cell shape. These microtubules are made up of the protein tubulin. Microtubules are important in many cell processes, but one of the most relevant functions with respect to Paclitaxel is their part in cell division where microtubule spindles are involved in chromosome separation. Figure 1.2 shows a cell undergoing cell division, with the microtubule spindles pulling the chromosomes. Additionally, the cell's skeleton, called the cytoskeleton, is also made up of microtubules. Most other drugs that target tubulin work by inhibiting microtubule assembly, but paclitaxel stabilizes the microtubule polymer. Paclitaxel boosts the polymerization of tubulin and leads to the over production of microtubules. This leads to the cell not being able to use its

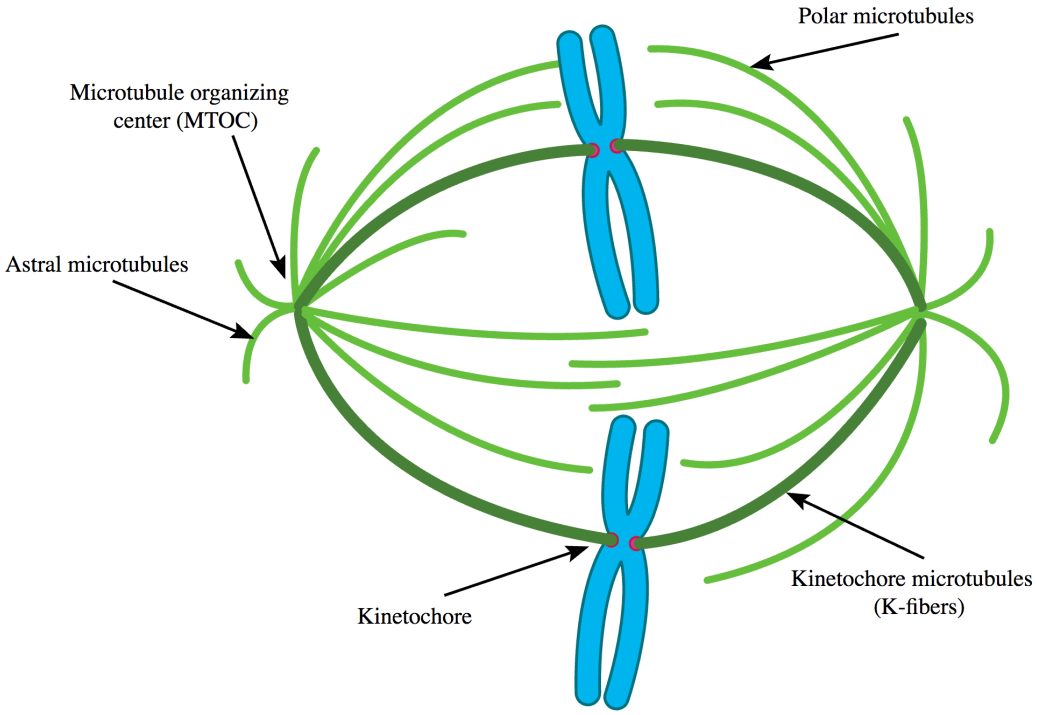

Figure 1.2: Microtubules during cell division. (Wikimedia Commons)

cytoskeleton properly, as it becomes too rigid. The spindles required during cell division are also affected by the overproduction of microtubules and the cell is unable to divide, leading to cell death [33].

Since cancer cells are rapidly dividing, paclitaxel is able to kill off these cells; as they are killed when they attempt to divide. Normal cells in the body do not divide and multiply unless they need to repair, which is why they are not affected as much by the drug. However, chemotherapy also kills other normal cells that divide rapidly, such as skin cells, bone marrow cells, hair follicles, and the lining of the digestive system. This is why taking chemotherapy is very taxing on the patient; it ends up killing not only cancer cells but some healthy cells.

1.3.3 Chemotherapy Drug Pathways

Knowing the method of action of a chemotherapy drug is very important for predicting its efficacy and side effects. The method of action is the specific biochemical interaction through which the drug produces its intended effect. When the drug enters the cell and is metabolized, it interacts with many different proteins, some of which have an effect on its efficacy [24]. Figure 1.3 shows the genes that encode some of the proteins paclitaxel interacts with as it enters and leaves the cell. SLCO1B3 is a gene that encodes an important transporter protein that has an effect on how well paclitaxel enters the cell. This is one of the most important genes in the pathway, as it determines how much of the drug will enter the cell. The genes NR1I2, CYP3A4, CYP2C8, and CYP1B1 all encode proteins which are responsible for metabolizing paclitaxel once it enters the cell [26]. This is very important since the drug is only effective

once it is metabolized. Since this drug works by stabilizing microtubules in the cell, the genes MAPT, MAP2, and MAP4 are very important as they encode the microtubule proteins, which have a direct effect on whether the drug is able to kill the cell. The genes ABCB1, ABCC2, ABCG2, and ABCC1 encode transporter proteins which determine how much of the drug exits the cell. Understanding the mechanism of action of a drug and the relationship that genes have on the effect of the drug is essential to being able to predict drug response.

1.4 Predicting Patient Outcome

1.4.1 Genetic Expression

Almost every cell in the body contains the same set of genes, however only a fraction of them are actively expressed. Generally, a different, but overlapping set of genes is expressed in each type of tissue in the body. What genes are expressed is determined by the type of cell and its current state. In order for a gene to be active, the cell needs to copy the desired DNA sequence of that gene into a piece of messenger RNA. Messenger RNA has the role of communicating the genetic information from the DNA of the cell, located in the nucleus, to the ribosomes, which build proteins. The messenger RNA is what the ribosome of the cell uses as a template to build proteins, out of amino acids. Genetic expression is a measure of the quantity of messenger RNA for a specific gene. Genetic expression is important because it gives information about the cell, for example if there is no messenger RNA for a specific gene, it likely means that gene is inactive; different levels of expression tell us about how active a gene is.

1.4.2 The Idea

According to the U.S. National Cancer Institute there are 63 different chemotherapy drugs approved to treat breast cancer [17]. One of the commonly used chemotherapy drugs paclitaxel has a response rate range from 30% to 60% [28]. There are many genes that affect the outcome of the drug and each individual patient has different gene expression levels for those genes. So the goal of many researchers is to be able to use machine learning techniques to predict the response a patient will have to a specific drug, based on things like genetic expression, mutations, and gene copy number. To be able to predict what drug would work on a patient would be revolutionary, as currently patients are given drug cocktails that are most likely to work on the average person. Drugs are currently administered in an experimental way; the patient receives their chemotherapy cocktail and months later, if there is no progress, they are given a different set of drugs. Being able to predict what drugs work on patients would eliminate that step and would increase the chance of survival for the patient, since the time they are on the wrong drug allows the cancer to grow and spread. Additionally, patients could be given cocktails of the drugs that best work on them based on their genetics. The current response rate for drugs is quite low and has a wide range. Being able to predict drug outcome would improve the drug response rate for patients.

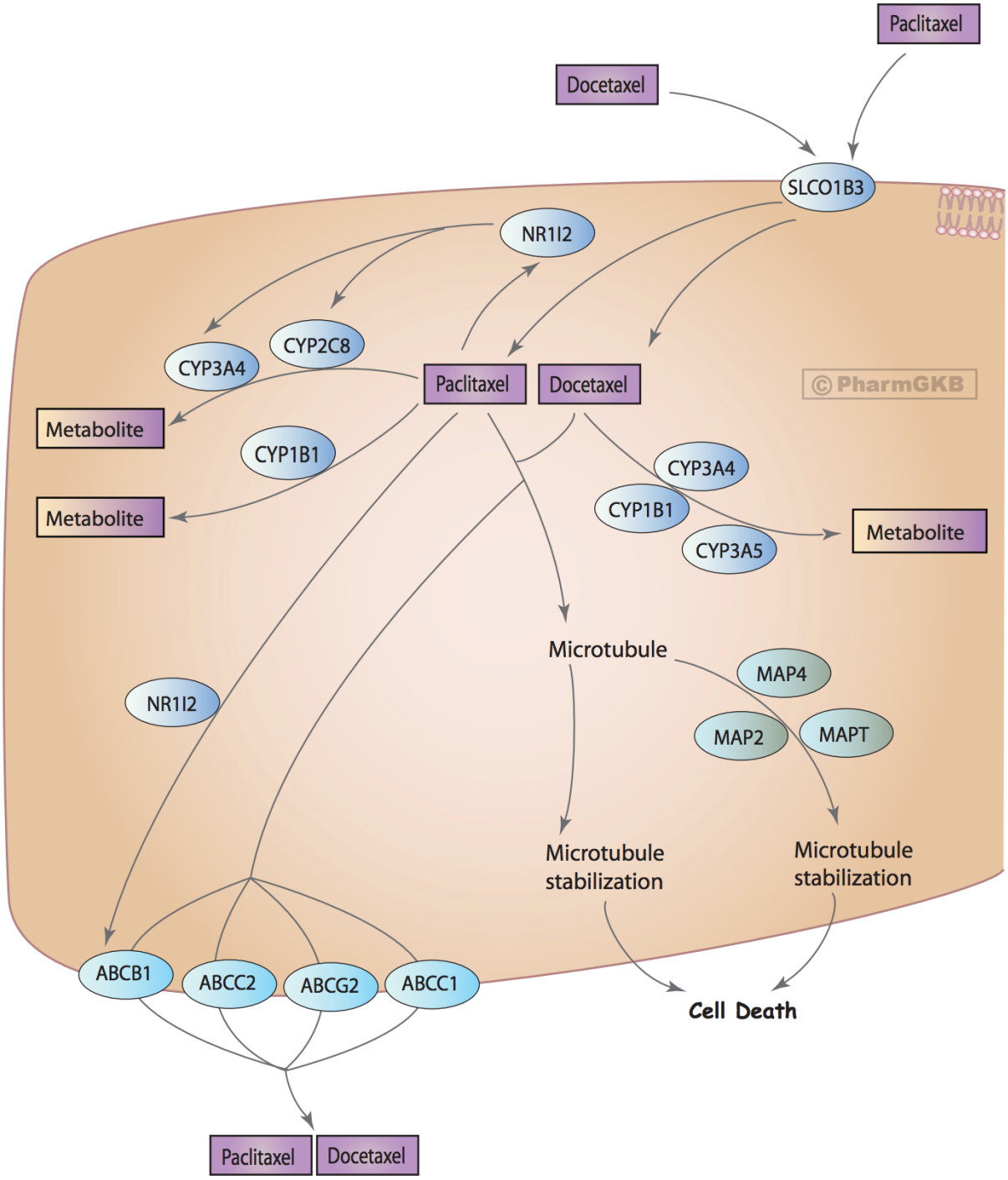

Figure 1.3: Paclitaxel pathways. (PharmGKB and Stanford University.)

1.4.3 Cancer Cell Lines

In order to derive a gene signature that can predict chemotherapy drug response in patients based on their genetics, data sets showing the relationship between gene expression values and drug response must be found. Patient data sets showing drug response are readily available, however they are very noisy data sets, since patient drug outcome is determined by many extraneous variables beyond the genetic expression of the patient. Patient outcome is affected by things like age, health, lifestyle, diet, and many other factors. This makes patient data sets unsuitable for training machine learning models, as there is too much noise in the data.

Cancer cell lines are cancer cells extracted from a human cancerous tumour, and then grown in a Petri dish. For breast cancer, cell lines have very similar characteristics to the tumours they are derived from [11]. This makes cell lines ideal candidates for training machine learning models, since cell lines do not have the number of extraneous variables affecting their response to the drug as patients do.

Cancer cell line data consist of information about gene expression values for many genes for each cell line and the response of each cell line to a specific drug. The data sets include cell lines from different patients, thus each cell line has unique genetic expression values. Genetic expression values are measured and appear as numerical values in the data set; the numeric value indicates the concentration of the messenger RNA for a specific gene. Each of the cell lines in the data set are treated with a chemotherapy drug, and their response to the drug is measured and recorded in the data set with a metric called GI_{50}. The GI_{50} value is the drug concentration required to inhibit cell line growth by 50%. The values of drug concentration are recorded as $-log_{10}M$.

1.5 Machine Learning

1.5.1 What is Machine Learning?

Machine learning is a part of computer science that deals with the study of algorithms that can learn from data and make predictions on it [1]. Machine learning models are not explicitly programmed how to behave, instead they are trained on data from which they learn. Its easy to write algorithms for certain tasks, such as deciding if an email contains a specific word. The input to the algorithm is the email text and the output will be a yes or no response indicating whether the email contains the specific word. The algorithm would be very simple to design. It just has to match up every word in the email against the specific word being checked for, if any of the words in the email match the word, the algorithm outputs yes, otherwise it outputs no. There are other tasks for which it is much harder to design the algorithm, for example classifying emails as being either spam or not spam. We know the input, the contents of the email; we know the output, the classification as either spam or not spam. However, coming up with an algorithm that decides whether an email is spam or not is something very difficult, as each spam email is different. So we do not know how to get the desired output from the input that we have. This is where data and machine learning become necessary. We can get lots of spam emails, lots of regular emails, and give them to a machine learning algorithm which will learn from the data how to classify the emails; rather than having to explicitly design an algorithm that can do this classification.

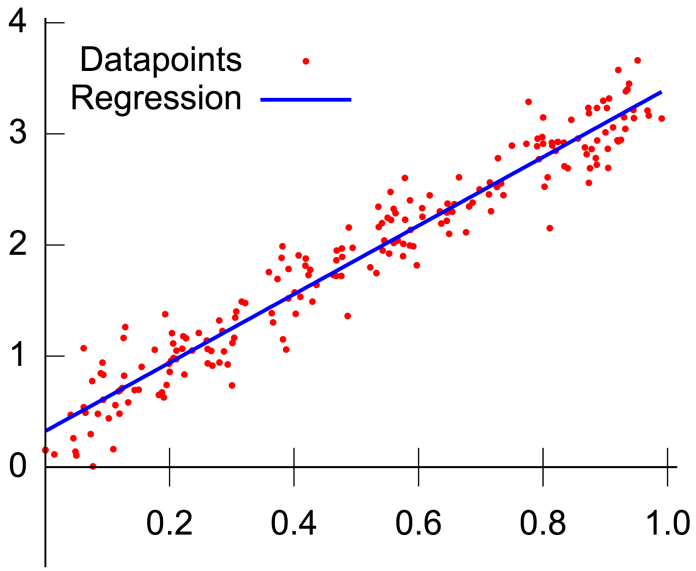

Figure 1.4: Linear Regression. (Wikimedia Commons.)

Machine learning is a type of artificial intelligence that finds patterns in data and adjusts the model behaviour to be able to recognize those same patterns in unseen data sets. Machine learning models can perform regression, which is being able to predict continuous values for a certain data member. Figure 1.4 shows data points which have been learned by some model, the red line is the regression line. The regression line is the learned relationship. An example would be predicting someone's weight given information about their diet and exercise habits. Machine learning models can also perform classification, which is being able to learn and predict what category a certain member of data belongs to. Figure 1.5 shows the difference between classification and regression. Regression tries to find the best line of fit of the data. Classification tries to find a line that separates the data into two classes, plus signs and circles. The learned relationship is this line, as it represents the models understanding of the difference of the two classes.

There are three main categories of machine learning: supervised learning, unsupervised learning, and semi-supervised learning. In supervised learning the model is built by training on data with known labels. A label classifies the data into belonging to a specific category; in the case of drug outcome that label is binary: either the drug works on a patient or it does not. To build a model that predicts drug response, the model would be trained by being given data for a number of cell lines, where each cell line would have a label indicating if the drug worked on it or not, and it would have the genetic expression for a number of genes. In machine learning the genetic expression for the genes are called the features of the data. The features are the values of that data which we use to learn about the label, and once trained, to predict it. Once the model is trained on this data set, it will be able to predict the label on data sets with just genetic expression values. In unsupervised learning the model is trained by being given data without labels, and it finds structures present in the data. One example of unsupervised learning is clustering, where the model will create clusters of data it thinks belong to the same class. Semi-supervised learning involves training the model on data that is a mix of labelled

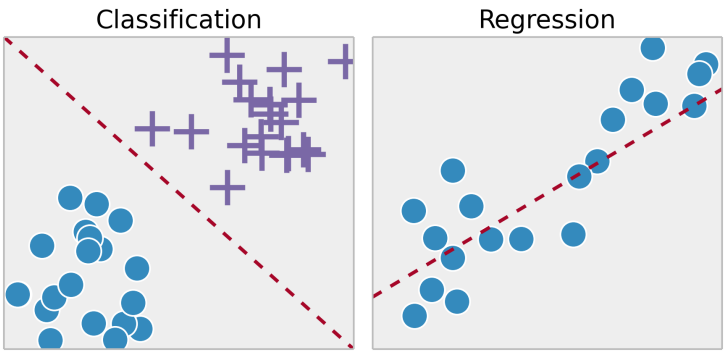

Figure 1.5: Regression vs Classification. (Packt Publishing, Cyrille Rossant, IPython Cookbook.)

and unlabelled members. Figure 1.6 shows an example of k-means clustering, where the goal of the algorithm is to partition the input data into k clusters.

1.5.2 Why Machine Learning is necessary?

The relationships between gene expression and drug response are very complex and not linear. In order to be able to predict the drug response in patients this relationship between genetic expression and drug outcome must be understood and modelled. However, the complexity of those relationships are not something humans can easily understand, as many of these relationships are synergistic between multiple genes, and would require someone to be able to understand a relationship in multidimensional space. Humans are very good at understanding two dimensional relationships. In two dimensional space one variable affects the outcome of the other; it is easy to visualize a graph where such a relationship is modelled, such as $y = x$, which is just a diagonal line. Figure 1.7 shows this relationship. We can also think about and understand three dimensional space, which is because we can also visualize it. However, three dimensional space is more complex and harder to understand. Figure 1.8 shows a sphere, which is a three dimensional relationship modelled by $xdx + ydy + zdz = 0$. Once we move on to four dimensional space, we can no longer visualize the relationship; we can describe and understand it through equations and symbols, however it becomes harder to understand. This applies to all dimensional spaces greater than three dimensions. When trying to find the relationship between gene expression and drug outcome, the number of genes is almost always greater than three, thus the dimensionality of the relationship is very high and thus very hard to visualize and understand. This is one of the main reasons why machine learning is a necessity for building these predictive models. Additionally, the data sets are often very large and not very much is known about their structure, making them very challenging to work with and try to extract information from. Although a lot is known about biochemical pathways and about which genes interact with each other to affect chemotherapy response, the numerical relationship of their genetic expression and drug response is not known. This multi-dimensional relationship is what machine learning techniques aim to learn.

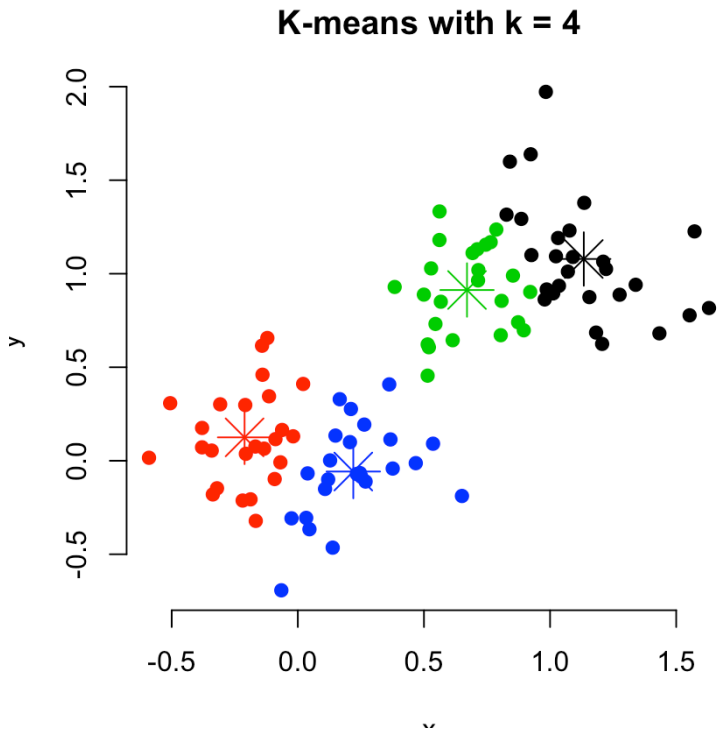

Figure 1.6: K-means Clustering, with k=4. (Statistical tools for high-throughput data analysis.)

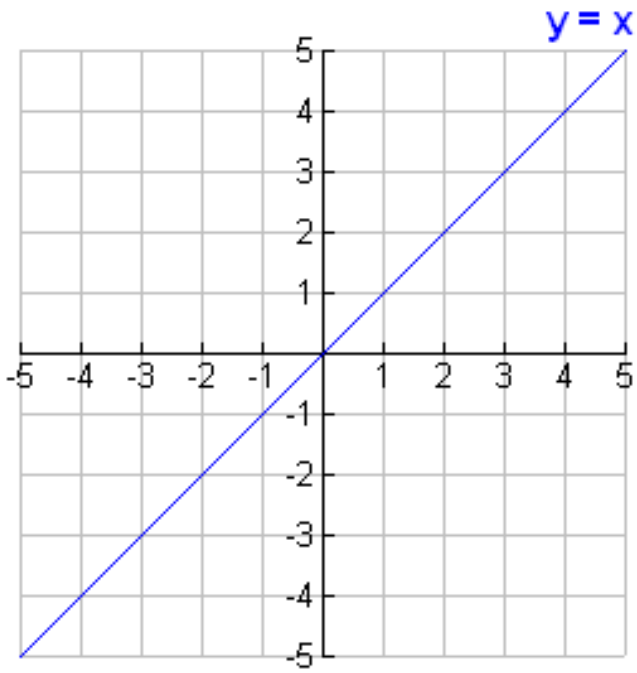

Figure 1.7: 2 dimensional relationship, $y = x$. (Ajay Halthor, Quora.)

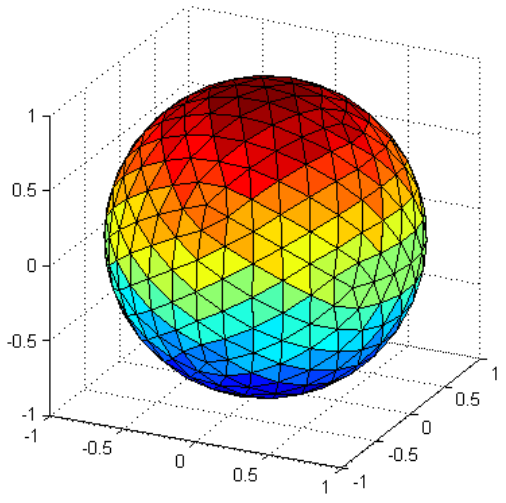

Figure 1.8: 3 dimensional relationship, $xdx + ydy + zdz = 0$. (Larry Cinnabar, Stack Overflow.)

1.5.3 Neural Networks

Neural networks(NNs) are very versatile artificial intelligence models that are inspired from the way the brain works, consisting of many interconnected neurons [1]. In a neural network each neuron is interconnected with a set of other neurons. Each neuron performs some operation on the value it receives from another neuron. The combination of these operations is what dictates the output of the neural network. NNs are capable of regression, classification and other types predictions. NNs are capable of supervised learning, unsupervised learning and semi-supervised learning. A NN takes features values as input, those values then go through the hidden layers of the NN, which are just a series of interconnected neurons, and the output layer are the neurons that give the result. Figure 1.9 shows a NN with one hidden layer, NNs can have any number of hidden layers.

The NN takes in numeric values for each feature, and outputs a numeric value that either indicates some scalar value prediction in the case of regression, or a value indicating the label of the input for classification. NNs can also have multiple output values. In supervised learning the NN is given a data set with labels, the goal of NN training is to adjust the weights of each neuron so that the input values are transformed into the correct label or scalar prediction. There are many different algorithms that can be used for finding the correct weights; the most common for NNs are back-propagation combined with gradient descent.

Neural networks can be used to build deep-learning computational models which are composed of multiple processing layers that can learn data with multiple levels of abstraction [22]. Conventional machine learning techniques require domain expertise to extract relevant features from data and to transform the raw data into a meaningful format. Representation-learning is a set of methods that allows the models to automatically figure out the representation of the raw data that is needed for detection or classification. Deep-learning methods are composed of multiple levels of representation-learning. They are composed of non-linear modules that each transform the representation of one level into a representation into a slightly more abstract

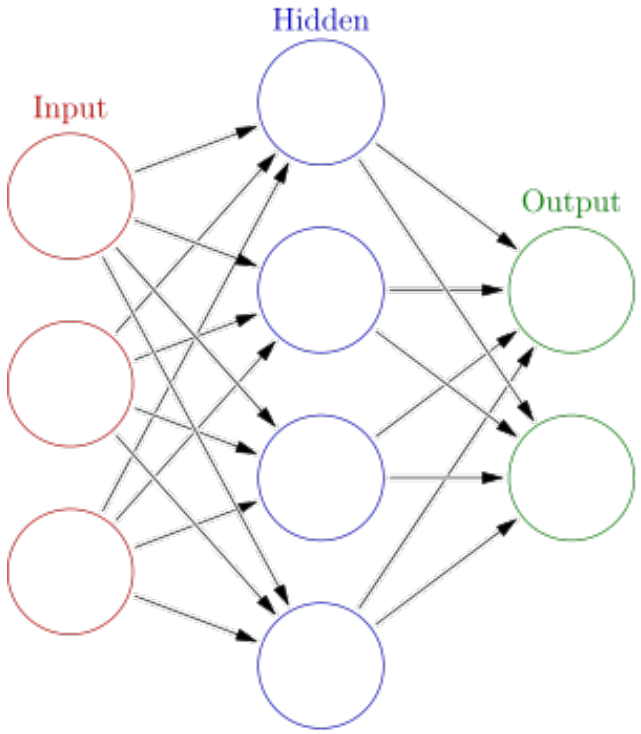

Figure 1.9: A neural network with one hidden layer. (Wikimedia Commons.)

level. The composition of these transformations allows very complex functions to be learned. These methods have dramatically improved speech recognition, visual object recognition, object detection and many other domains.

What makes NNs difficult to work with is that you have to decide how many hidden layers to use and how many neurons to have in each layer. Additionally, you must choose what training algorithms to use and there are many options. The difficulty is that all of these choices are made based on trial and error, which can be very time consuming; one might not choose the optimal combination of parameters, as the number of possibilities are very large. The largest drawback of NNs is that they can get stuck in local optima and are not guaranteed to find the best solution, which makes training them more difficult and time consuming compared to support vector machines.

1.5.4 Support Vector Machines

Support vector machines (SVMs) are supervised learning models that can perform both classification and regression. They are binary classifiers, meaning they can be trained to classify data members into one of two categories. SVMs are linear classifiers, so the way they learn to distinguish between the two classes of data is by forming a hyperplane which separates the two classes. SVMs are large margin classifiers, so the hyperplane will be chosen such that its distance from the data points is maximized. Figure 1.10 shows different hyperplane orientations. H_3 does not separate the classes properly, H_1 does so adequately but with a small margin, and H_2 separates the classes with a maximum margin. Several SVMs can be combined together to

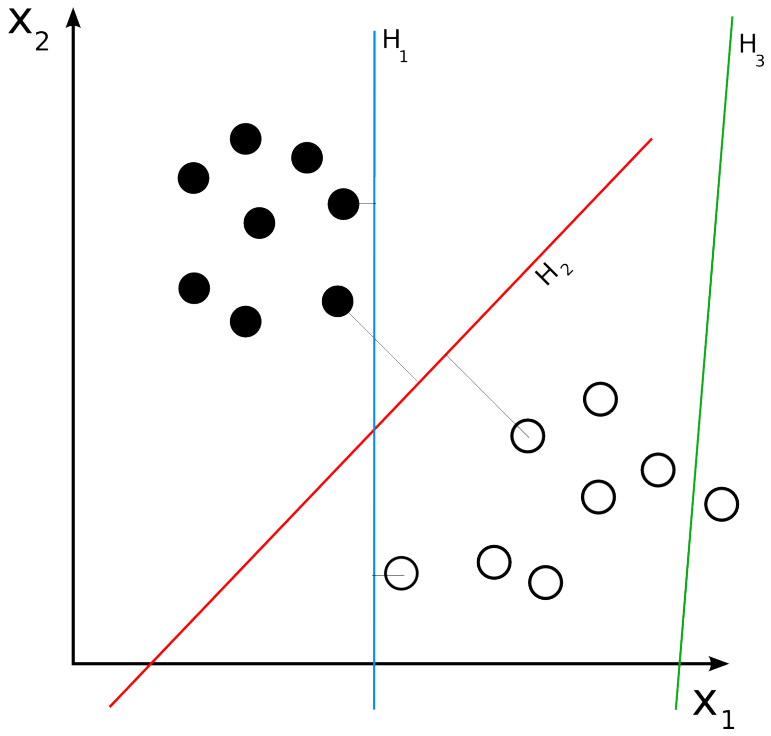

Figure 1.10: Support Vector Machine Hyperplane. (Wikimedia Commons.)

perform multi-class classification.

The challenge with SVMs being a linear classifier is that most data sets are not linearly separable, so there is not any hyperplane that can separate the data points. In many data sets some non-linear boundary is required to accurately separate the two classes. In biology most data sets are very complex and are infrequently linearly separable. The solution to this challenge is the use of kernel function, which maps the original dimension space into a higher dimension space, where the data is linearly separable. Figure 1.11 shows this transformation. When using kernels there are parameters which must be tuned through trial and error, increasing the time it takes to train a model. However, using kernels is what gives SVMs their ability to learn very complex data and the added time for tuning the parameters is not significant.

In many cases the data is still not perfectly linearly separable despite the use of a kernel. When the data is not linearly separable another method that is used with SVMs is something called a soft-margin; which allows some data points to be on the wrong side of the hyperplane. The soft-margin works using the hinge loss function, which tries to put as few points on the wrong side of the hyperplane as possible. When using a soft-margin SVM there is a C parameter which must be tuned. The C parameter relates to the cost function which determines how many data points are placed on the wrong side of the hyperplane. Smaller C values penalize the SVM less for putting data points on the wrong side, where as large C values penalize the SVM more for misclassifying points. Thus the larger the C value the less data points that will be misclassified; which may lead to the problem of over-fitting which will be discussed in detail later on in the paper.

What makes the SVM a good choice for building predictor models is that it does not require understanding anything about the interaction of the features, the genetic expression and their

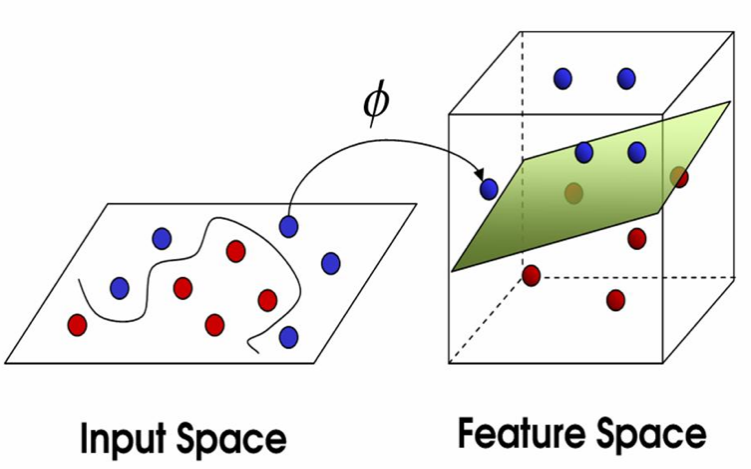

Figure 1.11: Kernel function. Mapping input space to higher dimensional space, to make it linearly separable. (Wikimedia Commons.)

relationship to the label or the drug response. SVMs also reach the global optimum solution and do not have many of the problems that neural networks have. Additionally, SVMs have very few parameters to tune, in some cases none, depending on the data set. An SVM with a Gaussian kernel can fit most types of data, and is thus very easy to use in situations where you do not understand the data very well, which is the case very often in biological problems.

1.5.5 Feature Selection

The more features a data set has, the larger the amount of data that is required to provide enough information for machine learning techniques to capture the relationships within it [15]. Due to this reason it is very important to only use the features that are necessary for capturing the relationship in the data; feature selection is one of the most important steps in building a machine learning model. Since in many cases the data that is to be learned is not very well understood, it is not clear which features are more important than others, thus selecting the most valuable features requires heuristic algorithms. In the domain of genetics, the problem of too many features is very prevalent, since there are many genes that affect things like chemotherapy response and current knowledge of biology does not know all genes that are involved in the outcome and what their role in the process really is. Additionally, if there are too many features, the time it takes to train models increases greatly, another reason why it is important to select only the most important features.

Feature subset selection is the process of removing features from an original set that are not relevant or are redundant. The problem with feature selection is the time it requires, you cannot try all combinations of features and see which ones create the best model; brute force does not work for feature selection. If you had an initial set of 30 features and wanted to select the best 15, there are 155,117,520 combinations to consider. In most cases, there are more than 30 features, and it is not known what number of them are needed, drastically increasing the number of combinations that would have to be considered. Therefore, heuristic algorithms must be used, as it is not practical to consider all the combinations of features.

Feature selection algorithms are divided into three categories: filters, wrappers, and embedded techniques [15]. Filter algorithms perform feature selection without training any machine learning models to evaluate the feature subset. This property makes them very efficient, as they do not require intensive computation. Wrapper algorithms use machine learning techniques as part of the process of feature selection. This allows them to perform better in selecting features since they take into account the performance of the models with the features. However training models on specific feature subsets is computationally expensive. Wrappers are not as efficient as filters as a result. Embedded algorithms construct the machine learning model as the features are being selected. They perform better computationally than wrappers, however they choose features that are classifier dependent, that may not work with other types of classifiers than the one trained on.

1.5.6 The Problem of Over-fitting

One of the challenges of training machine learning models on data sets is the risk of over-fitting the data. Over-fitting is when the model learns random error, or noise in the data instead of the actual relationship. Such models perform very well on the training data, but perform very poorly on new data, as they are not able to generalize. Under-fitting is when the model does not learn the underlying relationship, it is on the opposite side of the spectrum of over-fitting. Figure 1.12 shows this difference for a classifier that is trying to find a hyperplane between two different classes. In the under-fitting example, the hyperplane does not separate the two classes very well. In the over-fitting example, the hyperplane is trying to fit data points that are outliers and not representative of the general relationship. In the appropriate fitting example, the hyperplane does not try and fit the outliers, instead it separates the two classes well and captures the general relationship of the data. Over-fitting occurs when the model tries to fit the training data perfectly, rather than find the general trend.

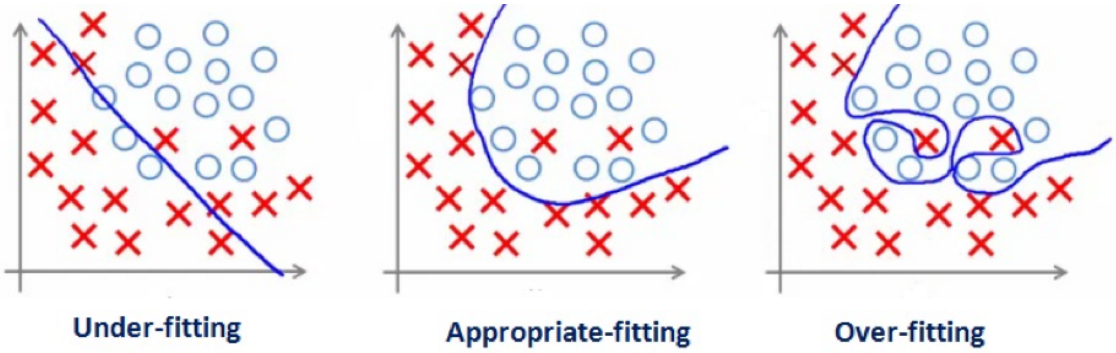

Figure 1.12: Comparison of underfitting, appropriatefitting, and overfitting. (Victor Lavrenko, University of Edinburgh.)

A common approach for preventing over-fitting is a technique called cross-validation. In cross-validation the training data is divided into partitions, so that some of it is used for training and the remainder for validation. This way the model is trained on a subset of the data and its ability to generalize is tested on the remainder of data that it has not been trained on. In k-fold cross-validation the data is divided into k partitions of equal size. The model is trained on all

but one of the groups; the remaining group is used for validation. This is repeated so all groups have been used as the validation group. This ensures the model is able to generalize to all parts of the data. Additionally, to reduce variance this process can be performed multiples times, so that cross-validation is performed with different partitions containing different data elements.

1.6 Related Works

1.6.1 Best Data Types for Cancer Predictive Models

Breast cancer is a disease that is clinically and genomically heterogeneous [11]. Molecular profiles for human breast cancer have enabled the use of features to predict therapeutic response. Little is known about how to best combine the various available data types to form predictors so that they yield optimal results. Cancer cell lines are well-suited for identification of predictive molecular features sets because they mirror many aspects of real breast cancer cell biological behaviour and therapeutic response.

Daemen et al. [11] analysed the response of 70 well characterized breast cancer cell lines to 90 therapeutic agents and used two independent machine learning approaches to identify pretreatment molecular features that are indicative of response in the cell lines. The machine learning methods used to identify the molecular features were support vector machines and random forest algorithms. The molecular feature data types were: pretreatment measurements of mRNA expression, genome copy number, protein expression, promotor methylation, gene mutation, and transcriptome sequence (RNAseq).

The cell lines were treated with nine different concentrations of each compound. The concentration required to inhibit growth in the cells by 50% (GI_{50}) was used to measure the response for each compound. This data along with the molecular feature data was made public and was used in the research for this thesis. The therapeutic compounds that were used to treat the cell lines include cytotoxic agents such as taxanes, platinols and anthracyclines, as well as targeted agents such as hormone and kinase inhibitors. This data was then used to train machine learning models. The cell lines were divided into a sensitive and resistant group for each compound, based on the mean GI_{50}, the assumption being that the larger the dose of the compound required to inhibit growth, the more likely the cell line is to be resistant. Grid search was used for feature selection. The models trained were support vector machines and random forests.

The results of the trained models were analysed to find out if particular machine learning methods or molecular data types were superior in predicting drug response. RNAseq outperformed all other data types across the complete list of 90 compounds. The gene expression profile was found to be the most effective way to model response to therapy.

Although cell lines mirror many of the characteristics of actual cancer tumours, they still have several drawbacks [11]. Actual cancer tumours have a micro-environment that has additional cell types that contribute to the growth of the tumour. Additionally, the micro-environment has varying oxygen content, which has been shown to influence therapeutic response. Despite the limitations of cell lines, Daemen et al. [11] concluded that their predictive signatures trained on cell lines are able to capture many of the key elements involved in predicting response; validating the use of cell lines for training predictive models.

1.6.2 Using Cell Line Data to Train SVM Predictors for Breast Cancer

Patients with breast cancer are prescribed chemotherapeutic agents such as paclitaxel and gemcitabine [12]. However, no standard chemotherapy regimen currently exists for patients. With current diagnostic techniques, surgery, and chemotherapy 30% to 70% of patients have recurrent disease or metastasis and die from the disease [5]. Resistance to the chemotherapy drugs is one of the major barriers to therapy. According to a study, breast cancer patient response rates to paclitaxel and gemcitabine after 6 cycles of chemotherapy were found to be only between 50% and 79% [23]. This is the motivation for developing gene signatures that predict response to chemotherapy drugs.

Deamen et al. [11] showed that the gene expression profile is the most effective way to model therapy. They used a genome wide approach to build their models; each model was derived from thousands of genes. Forming models from that many genes makes feature selection very difficult and time consuming. Additionally, using that many genes that are unknown to be related to chemotherapy response risks fitting the noise in the data rather than the underlying relationship [12].

Dorman et al. [12] proposed building SVM predictive models for paclitaxel and gemcitabine by selecting genes based on known drug pathways, metabolism, and genes implicated in drug resistance as opposed to a genome wide approach. They used gene expression, copy number, and mutation to train SVM models on cell line data. They acquired this data from the supplementary data of Daemen et al. [11]. Their hypothesis was that genomic differences in genotypes, expression and copy number of the selected genes account for the variable response to drugs like paclitaxcel and gemcitabine. Multiple factor analysis (MFA) was performed to see the correlation between gene expression, copy number, and mutation to GI_{50} for the available cell line data. Data types that did not correlate with GI_{50} were excluded. Figure 1.13 shows the selected genes and the results of the MFA: part A is for paclitaxel and part B is for gemcitabine. Genes with an asterisk are stable genes, genes highlighted in red show positive correlation with GI_{50} and genes highlighted in blue show negative correlation.

In order to train SVMs on the cell line data, they classified the cell line data into sensitive and resistant groups. They separated the cell lines based on median GI_{50} as opposed to mean GI_{50} used by Deamen et al. [11], as it was less affected by outlier cell lines [12]. The gene expression, copy number and mutation data for the chosen genes were used together with the binary sensitive and resistant labels to train the SVMs. The SVMs were trained with the Statistics Toolbox in Matlab, using a linear kernel function. In order to prevent over-fitting, leave-one-out cross-validation was used. Leave-one-out cross-validation is a type of k-fold cross-validation, where the model is trained on the whole data set, except one data member, which is used for validation. The process is repeated for each member; thus each data member is used as the validation set.

The SVM was initially trained on all 31 genes that were selected and verified by the MFA. Sequential backward features selection was performed to improve the classification error of the SVM. Each gene was left out one at a time, then an SVM model was trained on that subset of genes. The cross-validation misclassification score was calculated for each SVM. The gene that decreased the misclassification the most when left out from the model was removed from the SVM. This process was repeated until further gene removal led to higher misclassification score. Figure 1.14 shows the effect the gene removal had on the misclassification score for the

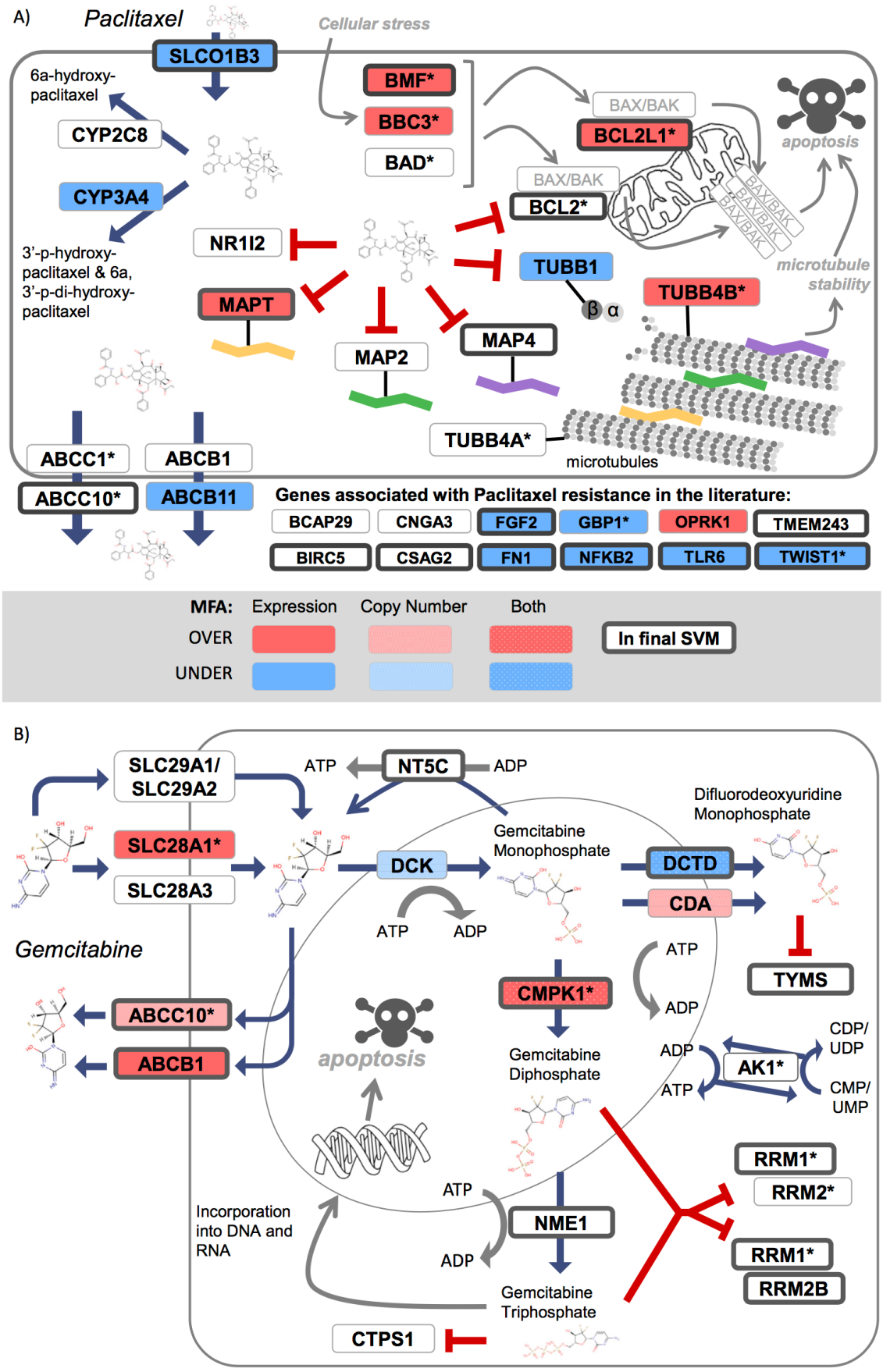

Figure 1.13: Genes Selected for Paclitaxel and Gemcitabine Predictive Models. (Elsevier, Molecular Oncology, Dorman et al. [12])

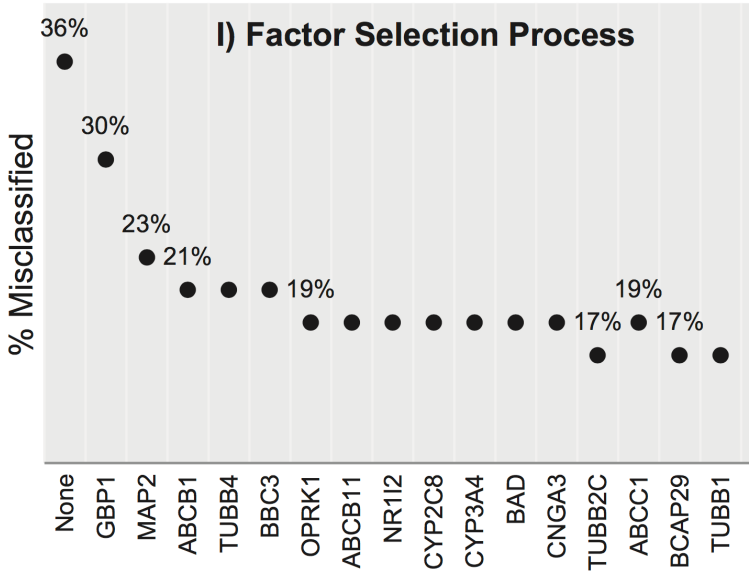

Figure 1.14: Paclitaxel SVM Feature Selection - Sequential Backward Feature Selection (Elsevier, Molecular Oncology, Dorman et al. [12].)

paclitaxel SVM. The initial set of 31 genes had a misclassification score of 36%, after the gene removal the misclassification score was reduced to 17%. Figure 1.13 shows the genes that were left after the gene removal process, they are indicated by a dark grey outline.

The SVMs were trained on the cell line data with the idea that they will capture the relationship between genetic expression and drug response. The goal for the SVM is to be able to predict chemotherapy response on patients. Clinical data for 319 patients was obtained, along with genetic expression measurements for each patient [14]. Some of the genes in the SVM model were not present in the patient data so they were removed from the SVM. The SVM was then used to predict the therapeutic response of the patients. The SVM predicitons were compared with the clinical outcome of the patients; which was either the patient had recurrent disease or complete pathological response. The SVM correctly predicted complete pathological response for 84% of patients. The SVM only correctly predicted 53% of patients who had recurrent disease. The overall accuracy on all the patients was only 58%. Additionally, tissue blocks were obtained from 17 patients whose response to paclitaxel and gemcitabine is known [12]. The genetic expression measurements for only 11 genes out of the 15 in the SVM were obtained. The SVM was trained on those 11 genes and were used to predict the outcome of the patients. Its accuracy was 71%, which is similar to the SVM performance on cell lines, which was 70%. These results show that SVMs trained on cell line data are able to transfer their ability to predict to patients.

Chapter 2

Methods

2.1 Selection of Initial Gene Sets

2.1.1 Data Dimensionality

In machine learning, as the number of features of the data set increases, the amount of data required to train an accurate model increases exponentially [15]. Additionally, data sets with a high number of features and low number of samples are prone to over-fitting. An SVM will try and find a relationship in any data it is given, and it will often find some relationship, regardless of whether or not that relationship actually means anything. The goal is to be able to train an SVM on cell line data and to capture the relationship between genetic expression and chemotherapy response. However, if the SVMs are trained on data with genes that are not involved in determining drug response, then there is the possibility that the SVM will just fit the noise in the data. The SVM trained on these irrelevant genes may learn to predict the cell line response for the cell line training data, but then when used to predict patient outcome it will fail to make accurate predictions, as it has fit a relationship that does not exist in patients. It is very important that the initial set of genes that are used for feature selection are chosen carefully and are known to be genes that are related to chemotherapy response.

2.1.2 Informed Gene Selection

We chose to use the methods that Dorman et al. [12] performed for initial feature selection. Which is using scientific literature and knowledge of drug pathways to select an initial set of genes that are known to be associated with chemotherapy response. Also including genes that are know to affect chemotherapy resistance in general, not necessarily for the specific drug the model is being built for.

2.1.3 Multiple Factorial Analysis and Epistasis

One modification in the initial gene selection process is the elimination of the multiple factorial analysis, which eliminated genes that did not correlate with the GI_{50} of the treated cell lines. The goal of this elimination was to get rid of genes that were deemed to not have predictive value for drug response. However, this elimination relies on the assumption that if a gene does

not correlate with GI_{50} it does not have any predictive value for the model. This implies that the gene should have a linear relationship with GI_{50}. However, genes often have synergistic effects. This phenomenon is called epistasis, which is where one gene's effect depends on one or more other genes. For example, two genes might be dependent on each other because they both encode for different proteins that interact together to carry out some function. Therefore the combination of expression values of multiple genes might become meaningful, due to their combined non-linear relationship, even if individually they did not correlate with GI_{50}. The idea is that when training the SVM it may find these non-linear gene interactions. So the features that were normally eliminated during the MFA step will be put into the feature selection process, with the assumption that the feature selection process will keep genes that have these synergistic relationships, and eliminate those who do not, as they will not improve the predictive ability. In order to verify this assumption, the feature selection process was carried out with the MFA eliminated genes, and without them. SVMs were then trained on the final set of genes from both feature selection processes to compare which SVM is better able to predict response.

2.1.4 Challenges of Available Data Size

Sometimes it is not known what genes are related to chemotherapy response, or there is still a lack of information about their involvement in chemotherapy response. Ideally if there were cell line data sets that contained more samples compared to the number of features, then feature selection techniques could be used to more accurately select the genes that are likely to be involved in predicting chemotherapy outcome. Feature selection techniques could be used to find which genes have the greatest predictive ability on the training data. This would allow for new genes to be discovered as having predictive value. The problem is that most cell line data sets are small; they usually have less then a hundred cell line types for a specific type of cancer. So it is not possible to use this strategy for selecting initial genes, as the data sets are too small to be able to capture the relationships of the thousands of potential genes.

The size of the cell line data sets presents a challenge even with the few carefully selected genes, as the initial number of genes is generally larger than the number of cell lines. This makes feature selection very important to building a good model, as it is important to use as few genes as possible, in order to prevent fitting noise and ensuring the model is able to capture the relationships present in the data. As previously mentioned the more features that are present in the model, the more data that is required to be able to learn the relationship between the features and the label that is to be predicted.

2.2 Automated Feature Selection and Parameter Tuning

2.2.1 Automating Sequential Backward Feature Selection

In Dorman et al. [12] the feature selection method used was sequential backward feature selection. This method starts with a full set of features, then leaves out each feature and trains a model. Each model's performance is then evaluated on the training data. The feature that decreased the misclassification the most when left out is then eliminated from the feature set.

This is repeated until the stopping criterion is reached. The stopping criterion for their method was when misclassification started to increase. This process was performed manually so they had to train each SVM with each gene left out. With their initial set of 31 genes they reached the stopping criterion after eliminating 16 genes. This means 376 SVMs had to be manually trained.

In order to speed up this process we wrote a Matlab program that automates this process. We used the Statistics and Machine Learning Toolbox SVM library. The program takes as input a data set with a full set of features. It performs sequential backward feature selection and outputs the subset of features that produce the lowest misclassification score on the provided test data. It also outputs the misclassification score for the subset of features that were found to be optimal. The automated feature selection method repeats the gene elimination process until there is only one gene left instead of stopping when misclassification starts to increase, as was done in the manual process. The reason this change was made is that the misclassification might increase in a certain step, but later decrease when more genes are removed; the old method was more prone to getting stuck in local optima.

The function that we used for evaluating gene sets is leave-one-out cross-validation. The model is trained on the whole cell line data set except one cell line, which is used for validation. The process is repeated for each cell line; thus each one is used for validation. Each prediction the SVM makes is on a cell line that was not included in its training set. The misclassification score is the percent of cell lines that were misclassified during the leave-one-out cross-validation. The goal of cross-validation is to see how well the trained model generalizes to unseen data. Leave-one-out cross-validation generally gives an optimistic misclassification score since the model is trained on the whole data set less one member; so the model sees most of the data. The reason why leave-one-out cross-validation was chosen is because the cell line data sets are very small, thus smaller values of k for the k-fold cross-validation would lead to the model not seeing enough data to learn the relationship. We also used 9-fold cross-validation, and other smaller values of k-fold validation when larger data sets were available and to more rigorously test successful models trained on smaller data sets.

The benefit of leave-one-out cross-validation is that it considers each combination of training and validation sets, since the validation set is just one member. The drawback is that it is computationally expensive, since the number of iterations required to compute it is the number of data members, so it can take a very long time to compute for large data sets. The benefit of smaller values of k-fold cross-validation is that less models have to be trained, since the data is split into fewer groups, thus it is less computationally expensive than leave-one-out cross-validation. The drawback is that each group is usually randomly generated, so each time it is performed the misclassification score varies, depending on how the data elements were grouped into the folds. In order to reduce the variance in the misclassification score cross-validation must be performed several times to combine the results of different group combinations.

2.2.2 Change of Kernel and Parameter Tuning

Dorman et al [12] used a linear kernel for their support vector machine models. A linear kernel uses a hyperplane to separate the cell lines into resistant and sensitive classes. In order to try and improve the SVM model we introduced a soft-margin linear kernel. The soft-margin kernel allows the SVM to place the hyperplane in such a way that some cell lines end up in the wrong

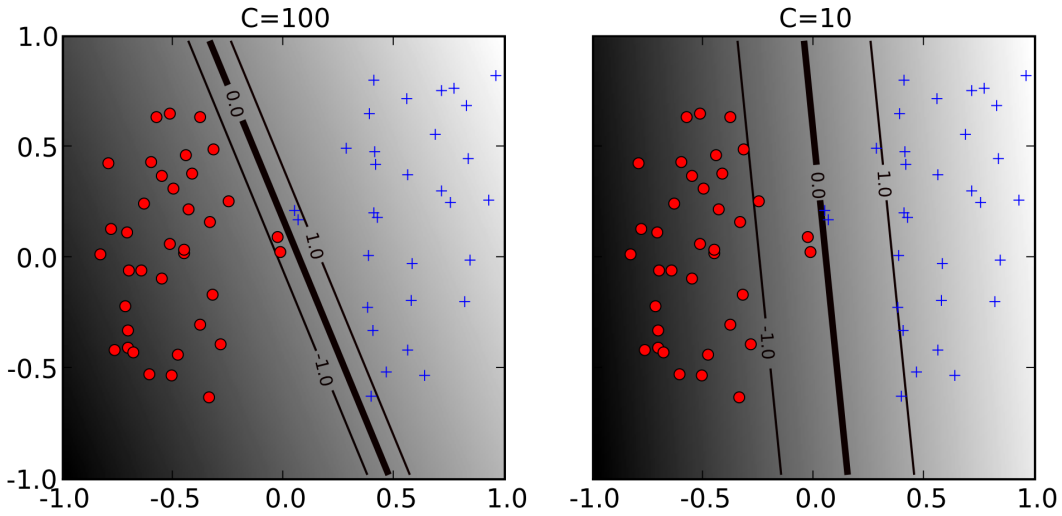

Figure 2.1: How C affects SVM Hyperplane. (Springer, A Users Guide to Support Vector Machines, Asa Ben-Hur.)

class or allows the hyperplane to come very close to data points. This is something an SVM normally does not do, as it is a large margin classifier; it normally places the hyperplane so that it is as far away from data points of both classes. When using the soft-margin the SVM is better able to get more of the cell lines into the right class. When using the soft-margin there is a parameter C that needs to be tuned. The value of C determines how the hyperplane will be placed. Large values of C penalize the SVM for misclassifying cell lines, thus it tries to place the hyperplane in such a way as to get as many cell lines into the right classes. Smaller values of C penalize the SVM less for misclassifying cell lines, so it tries to find a larger margin hyperplane. Figure 2.1 shows how a smaller C allows the hyperplane to get closer to data points.

The reason we do not just use a very large value for C, so that the SVM just puts all the cell lines into the right classes is that this can lead to over-fitting. In order to find the optimal value of C we try values from 10^{-10} to 10^{10}. In order to prevent over-fitting we perform cross-validation for each C value. An SVM is trained for each C value. The SVM that has the lowest misclassification score during cross-validation is the one with the correct C value, since it is best able to generalize to unseen data.

This parameter tuning was integrated into the feature selection program. The way this was done is the sequential feature selection algorithm was run for each value of C. The program still takes as input a data set with a full set of features. However it outputs not just the optimal subset of features and corresponding misclassification score but also the optimal C value. Algorithm 1 shows the pseudo code for the C optimization program.

Even with the use of a soft margin, if the data is not linearly separable there will be cell lines that the hyperplane cannot separate properly. There is no way to directly tell if a data set is linearly separable. However, if a linear kernel is able to achieve a low misclassification percentage then the data is probably linearly separable. This would be true if the misclassifi-

Input : Data set with full set of features
Output: Optimal Subset of Features, Misclassification Score, Optimal C Value
C_{best};
$Missclassification_{best}$;
$Features_{best}$;
for $C \leftarrow 10^{-10}$ **to** 10^{10} **do**
    $M_{current}, F_{current} \leftarrow SequntialBackwardSelection(Features, C)$;
    if $M_{current} < Missclassification_{best}$ **then**
        $C_{best} = C$;
        $Missclassification_{best} = M_{current}$;
        $Features_{best} = F_{current}$;
    end
end
return $C_{best}, Missclassification_{best}, Features_{best}$;

Algorithm 1: C Optimization

cation percentage was close to 0%, since that would be the case where the two classes can be perfectly separated by a linear hyperplane. Another way to know if the data is linearly separable is to compare the use of kernels that allow the model to fit non-linear data and compare the results with the use of a linear kernel. If the use of a non-linear kernel does not improve the misclassification percentage then the data is likely linearly separable since a linear hyperplane is able to separate the classes as well as a non-linear hyperplane. Since the misclassification score could be improved we decided to try using a Gaussian kernel. Additionally, since it is not explicitly known if the cell line data is linearly separable it is better to use the Gaussian kernel since it performs at least as well as a linear kernel. The linear kernel is a special case of the Gaussian kernel, since a Gaussian kernel with a very small σ value can be made to have the same performance as the linear kernel [21]. The Gaussian kernel can handle data where the relationship between gene expression and response is non-linear by mapping the data to higher dimensional space where it is linearly separable. Therefore, using the Gaussian kernel only increases the ability for the model to learn the data, whether the data is linearly separable or not.

When using the Gaussian kernel there is still the C parameter that needs to be tuned, but there is also another parameter σ. C still affects how close the hyperplane can get to data points. The value of parameter *sigma* determines how much curvature the decision boundary can have. When *sigma* is small the hyperplane is closer to being linear, when *sigma* is larger the hyperplane has more curvature. Therefore, large values of *sigma* can lead to over-fitting, as the hyperplane can perfectly fit the training data, which likely wont allow it to generalize to unseen data. Figure 2.2 shows how as *sigma* increases so does the curvature of the hyperplane. It is apparent that when $sigma = 100$ the hyperplane is over-fitting the data.

In order to prevent over-fitting with the Gaussian kernel both parameters C and *sigma* need to be tuned. The way we choose *sigma* will be the same as C, we try values from 10^{-10} to 10^{10}. In order to prevent over-fitting we perform cross-validation for each *sigma* value. However, we also have to consider C, so we decided to do a grid search on all values of C and *sigma*. We chose to do this because there are only 121 combinations to try and it ensures we consider

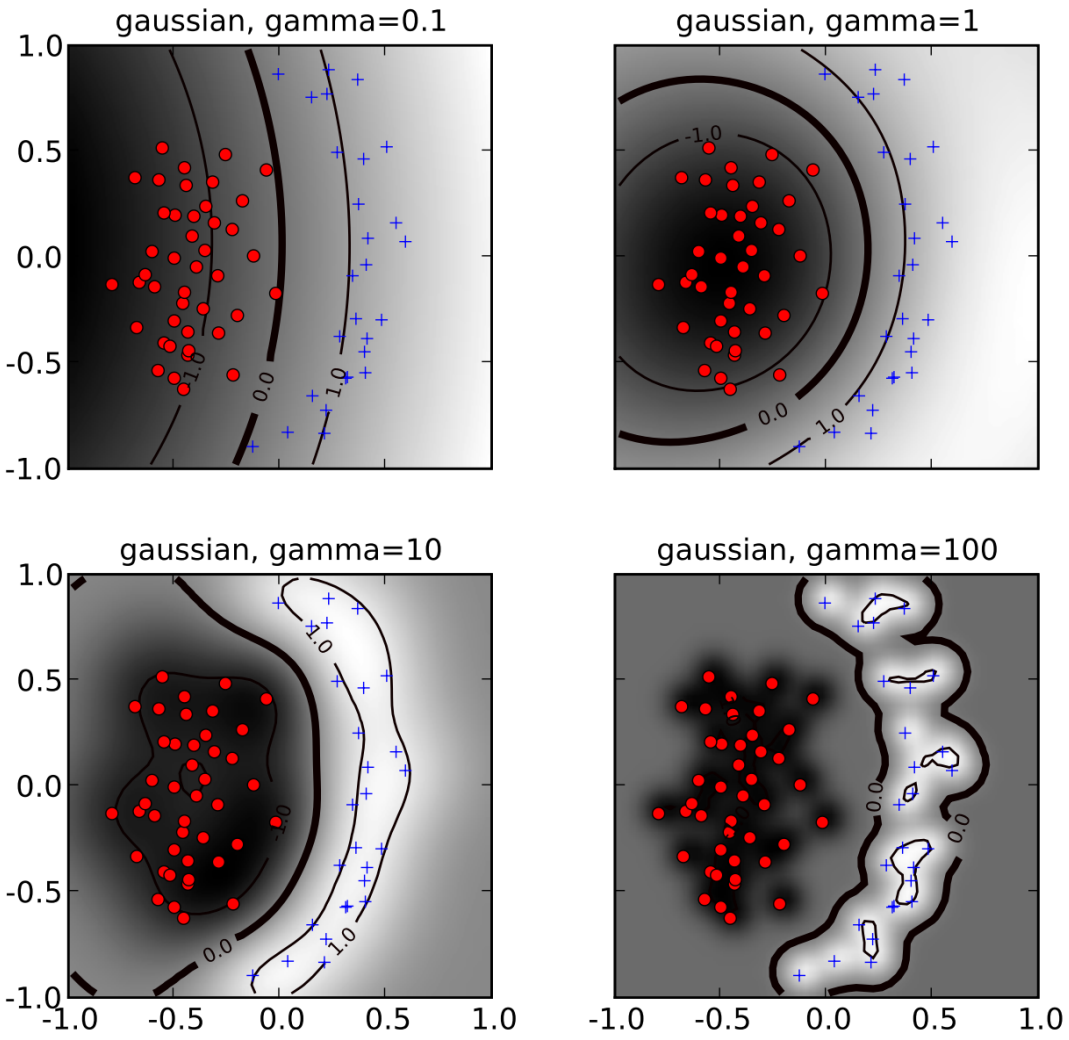

Figure 2.2: How σ(gamma in this diagram) affects SVM Hyperplane. (Springer, A Users Guide to Support Vector Machines, Asa Ben-Hur.)

a wide range of C and *sigma*; the computational cost of this is not significant. The program still takes as input a data set with a full set of features. However, it outputs not just the optimal subset of features and corresponding misclassification score but also the optimal C and *sigma* values. Algorithm 2 shows the pseudocode for the C and *sigma* optimization program. Inside the nested loops of the algorithm, the sequential feature selection algorithm is run, it does not rely on previous iterations of C and *sigma*. We decided to parallelize this program, since the sequential backward features selection could be run simultaneously for different values of C and *sigma*. This significantly improved the run time of the program.

Input : Data set with full set of features
Output: Optimal Subset of Features, Misclassification Score, Optimal C Value
$sigma_{best}$;
C_{best};
$Missclassification_{best}$;
$Features_{best}$;
for $\sigma \leftarrow 10^{-10}$ **to** 10^{10} **do**
  for $C \leftarrow 10^{-10}$ **to** 10^{10} **do**
    $M_{current}, F_{current} \leftarrow SequntialBackwardSelection(Features, C, \sigma)$;
    if $M_{current} < Missclassification_{best}$ **then**
      $\sigma_{best} = \sigma$;
      $C_{best} = C$;
      $Missclassification_{best} = M_{current}$;
      $Features_{best} = F_{current}$;
    end
  end
end
return $C_{best}, Missclassification_{best}, Features_{best}, \sigma_{best}$;

Algorithm 2: C and σ Optimization

2.3 Regression

The goal of the SVM trained on cell lines is to be able to predict based on genetic expression if the cell lines will be resistant or sensitive to chemotherapy. The resistant and sensitive labels for the cell line data are generated based on the GI_{50} values, which represent the amount of drug concentration required to inhibit their growth. This problem is actually a simplification of the harder problem of being able to predict the continuous variable GI_{50} based on the genetic expression of the cell lines. Being able to predict GI_{50} would be a lot more useful as it is more precise, that way an expert in biology could decide exactly what the predicted GI_{50} value means for resistance or sensitivity. We decided to try and solve this harder problem by using regression.

We first tried performing linear regression, which tries to fit the relationship of the data with a linear hyperplane. We initially tried doing this with the full set of features. We then performed sequential backward feature selection in order to improve results by reducing the number of

features. Since we do not know if the relationship in the data is linear we also tried using polynomial kernels to perform non-linear regression. Polynomial kernels can be of different degrees, the higher their degree the more curves it will have. However, when using a high degree polynomial over-fitting can occur. We tried different degrees of polynomial kernels, along with sequential backward feature selection. We evaluated the models based on their mean-squared-error on leave-one-out cross-validation, along with looking at the predictions of GI_{50} for leave-one-out cross-validation.

In order to try and reduce the dimensionality of the data without losing the information from the features, we decided to design a custom kernel for regression. We did this specifically for the drug paclitaxel since it is one of the most widely used chemotherapy drugs and there is lots of information about its mechanism of action and the genes that affect its efficacy. The idea behind the custom kernel is that some of the genes in the feature set belong to the same drug pathway. We combined those genes and multiplied their expression values together to generate a new feature which represents that specific pathway. We did research to find the main pathways for the method of action of paclitaxel. We found three pathways to be most significant. The pathway that affects microtubules, the pathway that affects the drug metabolism, and the pathway that affects transport of the drug. In the microtubule pathway there were three genes: MAPT, MAP2, and MAP4. In the metabolism pathway there were four genes: CYP3A4, CYP2C8, CYP1B1, and NR1I2. In the transport pathway there were seven genes: ABCB1, ABCG2, ABCC1, ABCC2, NR1I2, SLC22A7 and SLCO1B3. During the research we also found that a specific gene TLR4, correlates with chemoresistance and metastasis for breast cancer being treated with paclitaxel [30]. We decided that this gene should also be included, however it was not combined with other genes as it did not belong to any of the other pathways. There is also a group of five genes BMF, BBC3, BAD, BCL2L1, BCL2 that are associated with cellular stress [12]. We used those genes all multiplied by each-other as another feature. The kernel has weight parameters for each feature that even out the values during training, since some features have more genes than others; thus when multiplied together will give larger values than other features.

2.4 Using Multi-class SVM

The quantity of cell line data was insufficient for regression, so we tried to solve a simpler problem. Instead of predicting GI_{50}, we decided to split the cell lines into three categories rather than just two. Initially the cell line data was split into resistant and sensitive categories, with the split happening based on the mean GI_{50} value. The assumption is that if a cell line requires a higher concentration of drug for its growth to be inhibited then it is more likely that it will be resistant. The issue with having just two labels is that cell lines that are really close the the mean GI_{50} will not be represented properly, as they are right on the border. In order to prevent this we proposed three labels, low, medium and high concentration of drug required. This would be a more accurate predictor since it will identify the cell lines close to the mean GI_{50} as medium. This would be a more useful predictor as it gives more information about how likely a cell line is to being resistant.

SVMs are binary classifiers, so they can learn to distinguish only between two different classes. In order to make a multi-class SVM several binary SVMs can be combined together;

there are two main strategies for doing this. The one-verses-rest method trains an SVM for each class; it learns to distinguish between that class and the remaining classes. Each SVM outputs a confidence score rather than a label. When predicting a class the SVM for each class will make a prediction, the SVM with the highest score will determine which class is predicted. The other approach is called one-versus-one, where if there are K classes $K(K-1)/2$ binary classifiers are trained to form the multi-class SVM. In the one-versus-one approach an SVM is trained for each combination of pairs of classes, instead of just an SVM that distinguishes from one class and the rest. When making a prediction the class that gets the most votes from the binary classifiers gets chosen as the predicted class.

The Matlab multi-class SVM function we used employed the one-versus-one approach. We divided the cell line data set into three equal parts with the low, medium and high drug concentration labels. We then used the feature selection and parameter tuning program to train our model. The results were not as good as the SVM we trained on just two labels. This was to be expected since we are splitting an already small data set into even smaller groups. We concluded that the data set is also too small for multi-class classification. The loss of accuracy was not worth the added ability to be able to predict low, medium and high concentrations of drug. Thus we concluded that these small data sets will be best used with just sensitive and resistant labels.

2.5 Using Neural Networks

Artificial neural networks (NNs) are a supervised learning classifier that is frequently used in machine learning. We decided to try using NNs for classifying the cell line data, due to their ability to learn both linear and non-linear relationships. The goal was to find out if NNs are better at classifying the cell line data set than SVMs. Additionally, we wanted to compare the complexity of training a NN to that of training an SVM, as well as their computational requirements.

NNs consist of several layers which contain interconnected neurons. Each connection between two neurons is a weight function; it multiplies the value it receives from the previous neuron which it then passes on to the other connected neuron. Neurons themselves also perform functions to the values they receive before passing them on. A neural network has an input layer which is the features that it takes as input. That layer has a neuron for each feature, which for the cell line data would take the gene expression values. It then has hidden layers of neurons, which perform operations to the initial expression values and which then get passed on to the other hidden layers. The final layer is the output layer where the neural network outputs its prediction. NNs are a supervised learning model, so they are trained by being given an input and the corresponding desired output. There are different algorithms that adjust the weights of the neurons to achieve the desired output from the input.

When creating a NN one of the challenges is choosing the number of hidden layers and the number of neurons in each layer. There is a lot of opinions on how many hidden layers to use, but for many problems one hidden layer is sufficient [27]. Adding additional layers to a NN improves its ability to fit more complex functions, however with two layers a NN can represent a function of any shape [27]. The number of neurons in each hidden layer also improves the ability of the NN to fit more complex functions. It is important to choose the correct amount

of neurons since too few neurons can lead to under-fitting and too many neurons can lead to over-fitting. A guideline for choosing the correct number of neurons in the hidden layer is that there should be an amount between the size of the input layer and the output layer. The correct amount is different for each data set since there a number of factors that determine the optimal number of neurons in the hidden layer. The number of features, the size of the data set, the amount of noise in the data, the complexity of the relationship to be learned, and the training algorithm all affect the selection of the optimal amount of neurons. Therefore, trial and error is required to find the correct amount.

We used the Matlab Statistic and Machine Learning Toolbox which provides a NN library. When designing our neural network we initially started with one hidden layer and with ten neurons. We experimented with increasing the number of neurons up to the number of input features. The results were not significantly affected by increasing the number of neurons, ten performed just as well as greater values. Additionally, we tried adding an additional layer, this actually decreased the performance of the classifier, so we concluded one hidden layer was optimal for our problem.

During training, backpropagation is the algorithm that updates the weights of the NN. It propagates the error of the output back to each neuron, figuring out the contribution of each neuron to the error; based on that it updates the weights of the neuron. There are many different types of modifications of the backpropagation algorithm, all aimed at improving neural network training. Initially we used Bayesian regularization back-propagation which is a type of backpropagation that reduces over-fitting. We also tried using Levenberg-Marquardt back-propagation, which is optimized for fast training. Finally, we tried using scaled conjugate gradient back-propagation. All of these back-propagation algorithms were a part of the Matlab Statistics and Machine Learning Toolbox. We used sequential backward feature selection and cross validation to train and test the NNs. We got the best results using Bayesian regularization back-propagation.

From the many variations of NN architectures we tried, none of the NNs were able to achieve results that surpassed those of the SVM models. Each time a NN model is trained it forms a different set of weights; sometimes the models do not find the optimal weights for the neurons due to the training algorithm getting stuck in a local optimum. It is therefore necessary to re-train the model multiple times until it reaches an acceptable misclassification score. Choosing the correct number of hidden layers and hidden neurons is another challenge that complicates training as the trial and error process is time consuming. In contrast to NNs which can get stuck in local optima during training, SVMs always find a global minimum during training [6]. SVMs also do not require the selection of the internal architecture, they automatically determine their internal structure by selecting support vectors [19]. We concluded that SVMs are preferable to NNs for classifying our cell line data due to their ability to produce better results and their simplicity.

2.6 Limitations of Sequential Backward Feature Selection

The major challenge of feature selection is that the size of the search space is too large for any algorithm to be able to fully explore, since the number of possible feature subsets for n features is 2^n. Feature selection methods are heuristics that explore some of the search space. They all

aim to reach the global optimum, which is the optimal subset of genes.

Greedy algorithms are algorithms that are short sighted, they only consider the locally optimal choice at each step. They are not guaranteed to reach the global optimum and in most cases they do not [32] because that is not their goal. Their goal is to make the optimal choice at each step with no regard for how that will affect future steps. The sequential backward feature selection algorithm is a greedy algorithm. Its only consideration is what feature is bringing the misclassification down the most in each iteration. Its greedy because it just eliminates the gene that improves the the misclassification the most at each step, without considering how eliminating that feature will affect future steps. The main limitation of sequential backward feature selection is that it can not re-evaluate the usefulness of a feature after it has been removed. There could be combinations of features that could benefit from a feature that has been removed. Additionally, backward feature selection explores mostly large subsets of features. As the backward feature selection algorithm removes genes, each following iteration has fewer combinations to consider. When the algorithm starts it explores the greatest number of combinations of genes. When the algorithm has removed most of the genes it considers much fewer combinations. Thus when the algorithm is working with smaller subsets it has a lot fewer genes to consider. A similar algorithm, sequential forward selection works in the reverse order. It begins with an empty set of genes and adds genes one by one, according to which gene maximized performance when added. This algorithm mostly explores small subsets of data; since it only has a few genes left to add near the end of its operation. Therefore, with sequential backward feature selection, if the optimal subset of features is smaller it is less likely to be found. The sequential backward feature selection algorithm explores a very narrow section of the search space.

2.7 Genetic Algorithm for Feature Selection

Genetic algorithms are a type of artificial intelligence algorithms that are inspired by evolution. They are used to solve a variety of optimization problems. They are based on the principle of ”survival of the fittest” [31]. This is the principle that organisms that are best able to meet the demands of their environment are the ones that will live on and spread their genes to the following generation. Organisms that are not able to cope with their environment as well will have few or no offspring, and thus will not spread their genes. Since organisms that are more fit spread their genes more than ones who are less fit, with time the whole population evolves to be more like the fit organisms, since their genes become more widespread.

Genetic algorithms use chromosomes to encode possible solutions to the problem to be solved. These chromosomes are strings, where each position in the string represents an element of the solution. The chromosome strings are normally bit strings; thus the solution must be encoded as a bit string. There are also other string types such as hexadecimal that can be used, however binary is the simplest. A genetic algorithm starts with a population of randomly generated bit strings, representing random solutions to the problem that is to be optimized. This is referred to as the first generation. The techniques of evolution are then applied to the first generation to generate the following generations. This process is repeated until either a predefined number of generations have been generated or a satisfactory solution is found in one of the generations.

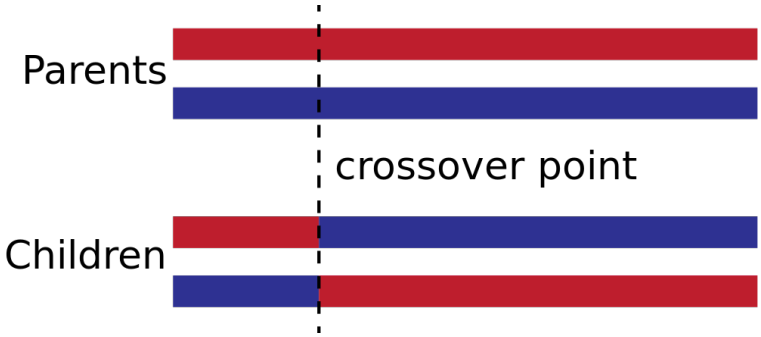

Figure 2.3: Single-Point Crossover. (Wikimedia Commons.)

During each generation a subset of the population is selected to breed the new generation. Chromosomes that have a higher fitness are more likely to be chosen for breeding; according to a fitness function that ranks the chromosomes solution. There are many different techniques for choosing this subset. The main issue in genetic algorithms is early convergence, which happens when the whole population looses its diversity and it becomes just multiple copies of one solution. This can happen due to many reasons, but one of the undesirable reasons is a poor choice for the method that selects which chromosomes should breed. If the method just chooses the best performing chromosomes of each generation to mate and gives no chance to chromosomes with worse performance, they would be wiped out entirely, when some part of their solution could have been useful. There are methods for avoiding this that will be discussed later.

Crossover is the step that comes after chromosomes are chosen for breeding. Crossover is a genetic operator that combines two chromosomes to form a new one. Usually two parent chromosomes are used to generate the child chromosome, but more than two can be used. The child chromosome is formed by taking a section of the chromosome from one parent and another section from the other parent. There are multiple ways to choose the portion of the chromosomes from the parents. The two common ways are single-point and two-point crossover [31]. In single point crossover a point along the chromosome is randomly chosen, then both parent chromosomes are cut at that point and each child gets one part from each parent; forming two new children. This is demonstrated in Figure 2.3. In two-point crossover two points are randomly chosen, both parent chromosomes are cut at those points, each child then gets the middle section of one parent and the outside sections of the other parent. This is demonstrated in Figure 2.4. Having more crossover points reduces the performance of the genetic algorithm, since the information about each solution is more likely to be scattered [31].

In order to increase diversity and explore more of the search space mutation is applied to chromosomes after they undergo crossover. Additionally, this prevents the algorithm from getting stuck in local optima. The goal of crossover is to find the best solution from the current solutions and the goal of mutation is to find new solutions from the search space [31]. A mutation involves making a random change to some part of the chromosome. There are many different techniques for mutation, the simplest is just choosing a random element of the binary chromosome string and flipping the bit; if it was a zero make it a one and vice versa. Another technique is choosing two random positions on the chromosome and exchanging the bits. Any manipulation that randomly alters the chromosome is a technique that can be used for mutation.

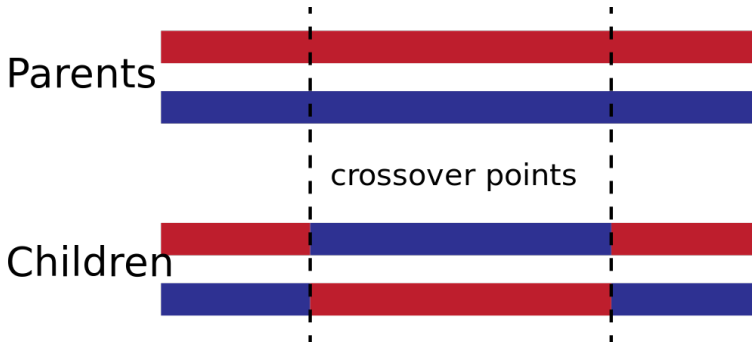

Figure 2.4: Two-Point Crossover. (Wikimedia Commons.)

Once the chromosomes have undergone crossover and mutations have been applied to the children the new population is formed. Another step before the new population is finalized is evaluation. The performance of each chromosome of the new population is measured, using the evaluation function that tests the candidate solutions. There are many techniques for what is done after the evaluation to finalize the new generation. The simplest technique is to just use the entire new population. Other techniques only take the best performing chromosomes from the new generation combined with the best performing chromosomes from the previous generation. There is also a technique called elitism, where the worst chromosome from the new generation is replaced by the best chromosome from the previous generation. This ensures the solution quality will not decrease from one generation to the next, however it can also lead to early convergence. Once these techniques are applied, the new generation is finalized and it undergoes through the whole process over and over until the stopping criterion is reached.

When implementing the genetic algorithm for feature selection the first step was to encode the problem as a string. We choose to use a binary string since it is sufficient for our purposes. Our goal was to find the optimal subset of features that minimize misclassification error on the cell line data. The genetic algorithm tries to find this subset of features, so each chromosome in the population represents a subset of genes. The chromosome was a binary string that has a bit for each feature in the set of chromosomes from which the subset is to be found. A one in the string represents a gene that is included in the subset and a zero represents a gene that is not included in the subset. We decided to start with a population of 10 randomly generated chromosomes. The evaluation function for the genes is training an SVM on the subset of the genes represented by the binary string and performing leave-one-out cross-validation. The misclassification was the score that is assigned to the corresponding chromosome.

Crossover requires some fraction of the genes to be selected for breeding. The percentage of genes that get selected for crossover must be selected for the algorithm. This parameter is very important, since too small a value will lead to little progress in finding new solutions and a value that is too high will lead to early convergence and potentially scatter valuable information that is contained in chromosomes.

In order to avoid early convergence, we decided to use linear rank based selection for selecting the chromosomes that will be used for breeding. Rank based selection assigns probability of being chosen for breeding based on the relative fitness of the chromosome. It assigns probability based on the rank of the chromosome relative the the fitness of other chromosomes, rather than based on the absolute fitness. If the most fit chromosome had a score 10 times

better than the second best, that will not affect the probability that the most fit chromosome would get. In order to assign probabilities, the chromosomes are first ranked from worst to best according to their fitness scores as evaluated by the leave-one-out SVM cross-validation. For a population of size N, rank r would be assigned a new score of $\sum_{n=1}^{r} n/N$. In order to choose a chromosome for breeding a random number is generated between the min and max score, and the chromosome that is the closest to that number is chosen. This randomness ensures that even chromosomes with low performance score get a chance to breed. However, they are less likely to be chosen, ensuring that more high performing chromosomes breed more often.

For breeding we decided to use single-point crossover. To perform single-point crossover we generate a random index, then both parent chromosomes are split at that index. Children receive one half from each parent so two children are formed from the breeding process.

For mutation we decided to use a single point mutation per chromosome since our chromosomes are relatively small; mutating more than one feature per chromosome would introduce too much randomness and we will loose too much valuable information. We used a 60% mutation rate in our algorithm. The mutation probability for genetic algorithms normally represents the percent of bits in each chromosome string that will be flipped. We used a slightly different approach, our percentage represents the percentage of chromosomes that will be mutated. However, each chromosome that is selected for mutation only has one bit that is flipped. Therefore our 60% mutation rate means that 1.9% of the bits in the population will be mutated, since we have a population of 10 and chromosome length of 31.

The genetic feature selection algorithm we implemented took a considerably longer time to run compared to our sequential backward feature selection algorithm. In order to speed up the run time, we parallelized the population evaluations. This is possible since they are all independent of each-other. During evaluation an SVM is trained for each member of the population and its performance evaluated on leave-one-out cross-validation. All these SVMs were trained and evaluated in parallel. There was no other part of the algorithm that could be parallelized since each generation of the algorithm depends on the previous generation. Algorithm 3 shows the pseudocode for the steps we performed in our genetic algorithm.

Input : Data set with full set of features, Number of generations
Output: Optimal subset of features, Misclassification score
$P \leftarrow$Generate random initial population;
for 1 **to** *Number of Generations* **do**
    Evaluate population(P);
    $P_{new} \leftarrow$ Perform crossover(P);
    Perform mutation(P_{new});
    Evaluate population(P_{new});
    Perform elitism(P_{new});
    $P \leftarrow P_{new}$;
end
$Missclassification \leftarrow$ getBestScore(P);
$OptimalGeneSubset \leftarrow$ getBestChromosome(P);
return $OptimalGeneSubset, Missclassification;$

Algorithm 3: Genetic Algorithm

2.8 Simulated Annealing for Feature Selection

"I have never encountered any problem where genetic algorithms seemed to me the right way to attack it. Further, I have never seen any computational results reported using genetic algorithms that have favorably impressed me. Stick to simulated annealing for your heuristic search voodoo needs." [32]. According to Skiena genetic algorithms do not work as well as simulated annealing for combinatorial optimization problems due two main reasons. Modelling the solution with genetic operators like crossover and mutation adds an unnecessary additional layer to the problem that is to be solved. The other reason is that crossover and mutation transitions usually lead to inferior solutions therefore making convergence of the algorithm very slow.

Simulated annealing is a probabilistic technique that is useful for finding the global optimum to an optimization problem in a large search space. Simulated annealing is inspired from the formation of crystals in solids during cooling [31]. When complex physical systems are cooling they naturally converge toward a state of minimal energy. The slower the cooling the more minimal energy state can be reached. During cooling particles move around randomly, the probability to stay in a particular configuration depends on the energy of the system and the temperature. This relationship is modelled by the equation $p = e^{-E/kT}$. Where e is Euler's number, E stands for energy, k is the Boltzmann constant, and T is the temperature. At high temperatures the system behaves more randomly, where as at low temperatures it becomes more stable since the probability to stay in a certain configuration increases.

The idea behind simulated annealing is in some ways similar to genetic algorithms. Usually the algorithm begins by generating a random solution. It then finds the fitness of the solution using an evaluation function. Then the algorithm generates a random neighbouring solution. This is similar to mutation, since these neighbouring solutions are formed by taking the current best solution and making a random change to it. The fitness of the neighbouring solution is then determined with the evaluation function. If the neighbour solution is better than the current best solution it replaces the current best, since it is getting closer to the optimum. If the neighbouring solution is worse then it decides whether or not to replace the current best solution based on the principles behind annealing. If the fitness of the current best solution is s_i and the fitness of the neighbouring solution is s_j the probability that the neighbouring solution will be chosen to replace the current based is modelled by the equation $p = e^{s_i - sj/kT}$. The probability is determined by the temperature and the difference between fitness of the neighbouring and current best solution. If the temperature is higher, the probability that a worse solution will be accepted is higher. The larger the difference between the neighbouring solution and the current best the lower the probability that it will be accepted. Once this probability is calculated, a random number between zero and one is generated and if the probability calculated is greater than this number then the neighbouring solution replaces the current best solution. For some implementations even if the neighbouring solution is better than the current best it is only accepted if the probability determined by the equation $p = e^{s_i - sj/kT}$ is greater than the randomly generated number between one and zero. This whole process is then repeated for many iterations at the same temperature. The temperature is then decreased and the procedure is repeated.

Simulated annealing models the cooling process so it starts at a high temperature and slowly reduces it. When the temperature is high worse neighbouring solutions are more likely to be accepted. This allows the algorithm to thoroughly explore the search space while the temper-

ature is high. As the temperature is decreased worse solutions are less likely to be accepted, however they still can. This prevents the algorithm from getting trapped in local optima. Simulated annealing behaves like a hill climbing algorithm that sometimes permits going downhill to avoid local optima [31]. As the temperature reaches the minimal value the algorithm rarely accepts worse solutions and narrows in on the best solution.

There are several parameters that need to be selected when implementing the simulated annealing algorithm. The initial temperature parameter is usually set to 1, which is then decreased by the temperature decrement function. The function for decrementing the temperature is modelled by the equation $t_k = \alpha * t_{k-1}$. The temperature undergoes exponential decay, the rate of the decay is controlled by the parameter α. This parameter α is generally set between values of 0.8 and 0.99 [32]. The algorithm undergoes a set amount of iterations between each temperature change. The number of iterations is usually set between 100 and 1000 [32]. The stopping criterion for the algorithm can be either achieving a specific classification score, or until the minimum temperature is reached.

Our genetic algorithm was implemented in Matlab and did not use any of the Matlab Statistics and Machine Learning simulated annealing libraries. When implementing our simulated annealing feature selection algorithm, we encoded our solutions as binary strings just as we did for genetic algorithms. We defined the neighbouring solution to be the current best solution with a random bit that is flipped. Our fitness function for evaluation solutions was training an SVM with leave-one-out cross-validation on the cell line data and using the performance of the gene subset as the score. Our initial temperature was set to 1. The minimum temperature was set to 1^{-6}. We experimented with different values of α and found the value that gave the best results to be 0.9 for our data sets. We also experimented with different values for the number of iteration at each temperature and found 100 iterations to be sufficient for producing good results. We considered results to be good when they surpassed the misclassification scores of sequential backward feature selection and were at least as good as the results obtained from the genetic algorithm. Higher values did not produce significantly better results but significantly increased the run time. There is no part of this algorithm that can be parallelized, however several instances of the algorithm can be run in parallel. This was a big advantage over the genetic algorithm, since we need to find optimal C and σ parameters for the Gaussian kernel of the SVM. Simulated annealing allowed us to run several instances of the algorithm for different C and σ values. We also modified the algorithm slightly by keeping track of the best solution encountered while searching, as there is no guarantee the final solution will be better than all solutions that were encountered during the search. Algorithm 4 shows the pseudocode for our simulated annealing feature selection algorithm.

Input : Data set with full set of features
Output: Optimal Subset of Features, Misclassification Score
$T \leftarrow 1$;
$T_{min} = 0.00001$;
$alpha = 0.9$;
$OptimalGeneSubset_{current} \leftarrow$ Generate Random Solution;
$Missclassification_{current} \leftarrow$ Evaluate($OptimalGeneSubset_{current}$);
$OptimalGeneSubset_{best} \leftarrow OptimalGeneSubset_{current}$;
$Missclassification_{best} \leftarrow Missclassification_{current}$;
while $T > T_{min}$ **do**
  for $i \leftarrow 1$ **to** 100 **do**
    $NeighbourGeneSubset \leftarrow$ generateNeighbour($OptimalGeneSubset_{current}$);
    $NeighbourMissclassification \leftarrow$ Evaluate($NeighbourGeneSubset$);
    $\Delta \leftarrow Missclassification_{current} - NeighbourMissclassification$;
    $p \leftarrow e^{\Delta/kT}$;
    if *p random(0,1)* **then**
      $OptimalGeneSubset_{current} \leftarrow NeighbourGeneSubset$;
      $Missclassification_{current} \leftarrow NeighbourMissclassification$;
    end
    if $NeighbourMissclassification < Missclassification_{best}$ **then**
      $OptimalGeneSubset_{best} \leftarrow NeighbourGeneSubset$;
      $Missclassification_{best} \leftarrow NeighbourMissclassification$;
    end
    $T = T * \alpha$;
  end
end
return $OptimalGeneSubset_{best}, Missclassification_{best}$;

Algorithm 4: Simulated Annealing Algorithm

Chapter 3

Results

3.1 Results after Optimizing Parameters and Kernel Change

3.1.1 Cell Line Data

The paclitaxel cell line data we used was the same data set used by Dorman et al. [12]. We obtained the data set from the Daemen et al. [11] supplementary materials. The data set contains 49 cell lines with GI_{50} values for each cell line along with genetic expression values. The GI_{50} is the drug concentration M, that is required to inhibit cell line growth by 50%. The values of drug concentration in the dataset are recorded as $-log_{10}M$. The genetic expression values were log_2 normalized, the data was derived from Affymetrix GeneChip Human Genome U133A and Affymetrix GeneChip Human Exon 1.0 ST arrays [11].

SVMs are sensitive to the way features are scaled, the accuracy of an SVM can be impacted if the data is not normalized [4, 8]. When proper scaling is not performed features with larger numeric ranges dominate features with smaller numeric ranges, which skews results. It is important that all features have values that fall within the same range. It is recommended that this range be [-1,+1] or [0,1], so feature scaling must be used to normalize the feature values to fall within those ranges [16].

In the paclitaxel cell line data, the gene expression values for each gene have slightly different ranges so we decided to normalize the data. To normalize our data, we chose to use standardized z-scores, which is the number of standard deviations each data point is from the mean. The range of gene expressions values were normalized to the range [0,1]. The z-score normalization ensures that the values for each feature have a mean of zero and a standard deviation of 1. The z-score represents the number of standard deviations a specific cell line is from the average of all cell lines for that gene.

3.1.2 Patient Data

Cancer patient data sets contain gene expression measurements and clinical data for patients who were treated with chemotherapy drugs. In order to test our models on patient data, patients must be classified as either resistant or sensitive based on the clinical data. However, there are no GI_{50} measurements available for patients. There is either time to relapse or time to death. We used thresholds of 5 years or less until relapse to divide patients into sensitive and resistant

since we are interested in the immediate response of patients to the drug. We are less interested in the long term effects post drug intake as the number of factors that affect outcome are too many. The thresholds we have chosen are based off clinically defined criteria for relapse, chosen empirically by physicians. For some patients there was only time to death information. For those we also used thresholds of 5 years or less to separate patients into sensitive and resistant groups. For both cases, less than the selected threshold would be labelled resistant and greater than the threshold would be considered sensitive.

3.1.3 Optimizing C parameter for Linear Kernel SVM

In the paper by Dorman et al. [12] the SVM classifier that was trained on the paclitaxel cell line data set used a linear kernel without the C parameter optimization. They used the sequential backward feature selection algorithm to find the optimal subset of features. To evaluate their model, they used leave-one-out cross-validation. Leave-one-out cross validation is optimistic since the model sees all of the data except for one cell line when making a prediction. We trained a model with the same features and kernel, we then performed 8-fold, 10-fold, and 12-fold cross-validation to examine the performance of the classifier. The results are shown in Figure 3.1. Using 12-fold cross validation the misclassification dropped off a significant amount from 17.02% to 25.01%. This indicates that the model isn't able to generalize well to unseen data. Decreasing the number of folds further did not significantly increase the misclassification, using a 8-fold cross-validation gave a misclassification of 27.26%. This demonstrates the optimism of leave-one-out cross-validation and why it is important to also evaluate models with smaller k-fold cross-validation.

The first thing we did to try and improve the performance of the classifier was to introduce the C parameter and try and find the optimal value that will minimize leave-one-out cross-validation. We used the automated sequential backward feature selection algorithm which performs feature selection for every value of C from 10^{-10} to 10^{10}. The value of C that gave the best misclassification score was 100,000. It is a very high value; however we are confident that we are not over-fitting due to the use of cross-validation. The optimized linear kernel SVM performed significantly better than the unoptimized version. The leave-one-out cross-validation was reduced to 6.38% from 17.02%, showing that the model has better accuracy than the unoptimized version. Its superior performance is confirmed by examining the performance of the optimized linear kernel SVM on the less optimistic cross-validation with fewer folds. The results are shown in Figure 3.1. The misclassification for 12-fold cross-validation was 15.81% which is less than the optimistic leave-one-out cross-validation score for the unoptimized linear kernel SVM. Comparing the 12-fold score for the unoptimized linear kernel, 27.26% with the C optimized version, 15.81%, shows that the optimized version missclassifies almost half the amount of cell lines. This indicates that this model is better able to generalize to unseen data. The consistent optimized linear kernel SVM results for different fold sizes confirm that the model is not over-fitting. During cross-validation on smaller sizes of k-fold the model is trained on fewer cell lines, and tested on more cell lines it has not seen, demonstrating its ability to generalize.

3.1.4 Optimizing C and σ Parameters for Gaussian Kernel SVM

After optimizing the linear kernel, we looked to different kernels for improving the results further. Since we are unsure if our data is linearly separable we decided to try using the Gaussian kernel since it is able to fit data that is not linearly separable. We used the automated sequential backward feature selection algorithm which performs a grid search for all values of C and σ between 10^{-10} and 10^{10}. The results for the best two SVMs derived from the algorithm are shown in Figure 3.1. The better performing one was optimized Gaussian SVM #2, which had a C value of 100,000 just like the optimized linear kernel and a σ value of 100. Its leave-one-out cross-validation misclassification was the same as the optimized linear version, 6.38%. For k-fold cross-validation for values of 12, 10 and 8, the optimized Gaussian kernel was only slightly better than the optimized linear version. This suggests that our data may be linearly separable. However, this could also be due to the dimensionality of the data, since we have many features compared to the number of cell lines, and linear kernels work well in these situations. However, since a Gaussian kernel with parameters tuned can be made to have the same performance as the linear kernel [21], we decided to continue using the Gaussian kernel since it will always perform at least as well as the linear kernel.

We also included the results from a second SVM with an optimized Gaussian kernel, it had a C value of 10,000. It performed significantly worse for leave-one-out cross-validation than the other one, with a misclassification of 17.02% compared to 6.38%. For small fold cross-validations it performed only slightly worse. These results show the importance of finding the optimal C value, as it is clear a value of 100,000 instead of 10,000 leads to a better model.

3.1.5 Patient Data Prediction Results for Linear and Gaussian Optimized SVM

The unoptimized linear kernel SVM model developed by Dorman et al. was tested on a patient data set which was obtained from Hatzis et al. [14]. The data set contained gene expression measurements and clinical data for 319 patients who were treated with paclitaxel and anthracycline chemotherapy. The clinical data contained clinical outcome for the patients, whether the patient had recurrent disease or complete pathological response. This clinical outcome was used to compare against the prediction of the SVM model on the patient data set.

Dorman et al. [12] also evaluated their SVM model on a set of tissue samples that were missing four genes that were present in the SVM model. Their original cell line-derived SVM had 15 genes, from which they removed the missing four and retrained the model. However, they did not redo the feature selection process from the full set of genes, so the 11-gene SVM they used did not necessarily contain the optimal gene subset. If they had removed the four missing genes from the full set of genes, then performed the feature selection process then they would have the optimal subset of genes. The Hatzis et al. [14] patient data set is missing two genes, BMF and CSAG2. These same two genes were also missing from the tissue samples, as a result they used the same 11-gene SVM to test on the patient data.

In order to have comparable results we decided to remove the missing genes BMF and CSAG2 from the initial gene set for the paclitaxel cell line data. We then used the automated sequential backward feature selection without tuning the C parameter, to derive the optimal subset of genes that could be obtained with an unoptimized C parameter. We then trained the

Paclitaxel SVM Classifier Linear Kernel vs. Gaussian Kernel

Misclassification Percentage

Cross Validation	Original Linear SVM	Optimized Linear SVM	Optimized Gaussian SVM #1	Optimized Gaussian SVM #2
Leave-one-out	17.02%	06.38%	17.02%	06.38%
12-fold	25.01%	12.16%	15.19%	12.11%
10-fold	25.75%	13.89%	15.37%	13.75%
8-fold	27.26%	15.81%	15.75%	15.74%

The k-fold percentages are averaged over 1000 iterations to account for k-fold generation variance.

Original Linear SVM

C: 1
Features: ABCC10, BCL2, BCL2L1, BIRC5, BMF, C7orf23/TMEM243, CSAG2, FGF2, FN1, MAP4, MAPT, NFKB2, SLCO1B3, TLR6, TWIST1

Optimized Linear SVM

C: 100,000
Features: ABCB1, ABCB11, ABCC10, BAD, BCL2L1, BMF, CNGA3, CYP3A4, FGF2, FN1, MAP4, MAPT, NFKB2, NR1I2, SLCO1B3, TUBB1, TUBB2C/TUBB4B, TWIST1

Optimized Gaussian SVM #1

C: 10,000
Sigma: 100
Features: ABCB11, ABCC1, ABCC10, BAD, BCL2L1, BMF, FGF2, FN1, MAPT NFKB2, SLCO1B3, TUBB4/TUBB4A, TWIST1

Optimized Gaussian SVM #2

C: 100,000
Sigma: 100
Features: ABCB11, BAD, BCL2L1, BMF, CNGA3, FGF2, FN1, MAP2, MAP4, MAPT, NR1I2, SLCO1B3, TUBB1, TWIST1

Figure 3.1: Results for SVM linear and Gaussian kernel optimization.

SVM on the paclitaxel cell line data for the derived subset of features. This model was then used to predict the patient outcome for the Hatzis et al. [14] patient data set.

In order to test the optimized Gaussian kernel SVM we also had to retrain the model due to the two genes BMF and CSAG2 that were missing from the Hatzis et al. [14] patient data set. We removed the two genes from the initial set of genes for the paclitaxel cell line data. We then used the sequential backward feature selection algorithm that finds the optimal subset of features and the optimal C and σ values. We also performed this process for the optimized linear kernel SVM, where the algorithm returns the optimal subset of genes and the optimal C value.

Figure 3.2 shows the results for retraining the linear and Gaussian kernel SVM models with the genes BMF and CSAG2 being left out from initial gene set. The optimized Gaussian kernel SVM has the best results but they are slightly worse then when the genes BMF and CSAG2 were in the model, suggesting that these genes could be important for prediction. The worse results could also be due to the sequential backward feature selection method, since not having the genes changes the order in which it removes genes, therefore changing the combinations of genes that are explored. These results also suggest the data may not be linearly separable since the Gaussian kernel outperformed the linear optimized kernel. Additionally, for these genes the optimized linear kernel had an optimal C value of one, which is equivalent to the unoptimized linear kernel; the two models are equivalent.

The cell-line-trained models were used to predict the patient outcome for the Hatzis et al. [14] patient data. That data set contained genetic expression values for the genes in the models and clinical outcome. There were 63 patients that showed complete pathological response and 257 with residual disease. The results for the linear, optimized linear and optimized Gaussian kernel SVMs are shown in Figure 3.3. The 11-gene SVM Dorman et al. [12] had an accuracy of 58.44%, the recall was 84% and specificity was 52.5%. The optimized linear kernel SVM that was derived from removing the genes BMF and CSAG2 performed slightly worse with an accuracy of 53.29%. The optimized Gaussian kernel SVM performed better than all the other models, with an accuracy of 65.20%. However, it had a very low recall compared to specificity. The Gaussian kernel SVM model was better able to predict which patients paclitaxel would not work for compared to which patients it would work for.

In order to see what effect standardizing the genetic expression values has on the prediction ability of the models we used the sequential backward feature selection algorithm to derive linear, optimized linear and optimized Gaussian kernel SVM models. We derived these models from the paclitaxel cell line data with genes BMF and CSAG2 removed, so that the models can be tested on the Hatzis et al. [14] patient data set. The cell line data prediction results for the SVM models is shown in Figure 3.4. The patient prediction results for the SVM models are shown in Figure 3.5. Not standardizing the genetic expression values significantly impacted the performance of the models. The decrease in predictive ability on the paclitaxel cell line data was very small, however the Gaussian kernel SVM model was mostly affected since it gets the most benefit from standardization. The results on the patient data set without standardization were very poor, all below 50%, which confirmed that standardization is very important for training our models.

Paclitaxel SVM Classifier Linear Kernel vs. Gaussian Kernel
(Excluding genes BMF and CSAG2)
(Standardized)

Misclassification Percentage

Cross Validation	Original Linear SVM	Optimized Linear SVM	Optimized Gaussian SVM
Leave-one-out	12.77%	12.77%	10.64%
12-fold	20.41%	20.59%	15.31%
10-fold	22.02%	22.07%	16.52%
8-fold	23.86%	23.76%	18.59%

The k-fold percentages are averaged over 1000 iterations to account for k-fold generation variance.

Original Linear SVM

C: 1
Features: BCAP29, BCL2, BCL2L1, C7orf23/TMEM243, CNGA3, FGF2, FN1, GBP1, MAPT, NR1I2, OPRK1, SLCO1B3, TLR6, TUBB1, TUBB4/TUBB4A

Optimized Linear SVM

C: 1
Features: BCAP29, BCL2, BCL2L1, C7orf23/TMEM243, CNGA3, FGF2, FN1, GBP1, MAPT, NR1I2, OPRK1, SLCO1B3, TLR6, TUBB1, TUBB4/TUBB4A

Optimized Gaussian SVM

C: 10,000
Sigma: 100
Features: ABCB11, ABCC1, BCAP29, BCL2L1, C7orf23/TMEM243, CNGA3, CYP3A4, FN1, MAP4, MAPT, OPRK1, TLR6, TUBB1

Figure 3.2: Results for SVM linear and Gaussian kernel optimization with BMF and CSAG2 left out from initial gene set.

SVM Linear vs. Gaussian Kernel Patient Results

Kernel	Accuracy	Recall	Specificity
Linear	53.29%	24.19%	60.31%
Linear Optimized	53.29%	24.19%	60.31%
Gaussian Optimized	65.20%	6.45%	79.38%

Figure 3.3: SVM model prediction results on Hatzis et al. [14] patient data set.

3.2 Pan Cancer Model Results

3.2.1 Cell Line Training Results

The paclitaxel breast cancer cell line data set has only 47 cell lines, which makes it tough for the SVM models to fully capture the relationship between genetic expression and chemotherapy response. We hypothesized that we can combine cell line data for multiple types of cancer that were treated with paclitaxel and train SVM models on this data. Paclitaxel has the same method of action for killing all cancer cells, so it works similarly in different types of cancer. We believe that on a larger data set, despite the different cancer types our model will be able to capture the relationship between genetic expression and paclitaxel response in the cell lines. We were able to acquire a data set of 414 cell lines that were treated with paclitaxel from the Cancer Cell Line Encyclopedia [3]. The data contained genetic expression values and IC_{50} measurements for paclitaxel response. IC_{50} is the concentration at which drug response reaches an absolute inhibition of cell grown by 50%. The median IC_{50} value was used to split the cell lines into sensitive and resistant groups.

We used the sequential backward feature selection algorithm to train an optimized Gaussian kernel SVM model on the 414 cell line data set containing cell lines from multiple cancer types. Figure 3.6 shows the results of the model compared to the 47 breast cancer cell line model. The results indicate that the 414 cell line SVM model was able to capture a relationship that is present in all the different cancer types, since it is able to predict their outcome well in k-Fold cross-validation. The lower C and σ values also indicate that the model has found a simpler relationship than the 47 cell line model. The C value of the 414 cell line model is significantly lower, indicating that the decision boundary is much further away from the data points. The σ value of the 414 cell line model is also lower, indicating that the decision boundary has less curvature. The consistent misclassification score for different sizes of k-Fold cross-validation also indicates that the model is better able to generalize. This is to be expected since the data set is a lot larger, however this likely confirms our hypothesis that the relationship between genetic expression and paclitaxel outcome is similar in different cancer types.

Paclitaxel SVM Classifier Linear Kernel vs. Gaussian Kernel
(Excluding genes BMF and CSAG2)
(Non-standardized)

Misclassification Percentage

Cross Validation	Original Linear SVM	Optimized Linear SVM	Optimized Gaussian SVM
Leave-one-out	12.77%	12.77%	12.77%
12-fold	18.23%	17.97%	17.48%
10-fold	19.33%	19.36%	18.45%
8-fold	21.05%	21.16%	20.0%

The k-fold percentages are averaged over 1000 iterations to account for k-fold generation variance.

Original Linear SVM

C: 1
Features: ABCC1, ABCC10, BAD, BCAP29, BCL2L1, FGF2, FN1, GBP1, MAP4, MAPT, NR1I2, OPRK1, SLCO1B3, TLR6, TUBB1, TWIST1

Optimized Linear SVM

C: 1
Features: ABCC1, ABCC10, BAD, BCAP29, BCL2L1, FGF2, FN1, GBP1, MAP4, MAPT, NR1I2, OPRK1, SLCO1B3. TLR6, TUBB1, TWIST1

Optimized Gaussian SVM

C: 100,000
Sigma: 100
Features: ABCB11, ABCC10, BCL2L1, BIRC5, C7orf23/TMEM243, CNGA3, FN1, GBP1, MAP4, MAPT, NR1I2, OPRK1, SLCO1B3, TLR6, TUBB1, TUBB2C/TUBB4B, TUBB4/TUBB4A

Figure 3.4: Results for non-standardized SVM linear and Gaussian kernel optimization with BMF and CSAG2 left out from initial gene set.

SVM Linear vs. Gaussian Kernel Patient Results
(non-standardized gene expression values)

Kernel	Accuracy	Recall	Specificity
Linear	44.51%	45.16%	44.36%
Linear Optimized	44.51%	45.16%	44.36%
Gaussian Optimized	44.83%	56.45%	42.02%

Figure 3.5: Non-standardized SVM model prediction results on Hatzis et al. [14] patient data set.

3.2.2 Patient Prediction Results

The Cancer Genome Atlas (TCGA) Data

In order to see if the predictive ability of the 414 cell line trained SVM transfers to patients we acquired several data sets for lung, ovary and breast cancer patients who were treated with paclitaxel. We acquired the patient data from The Cancer Genome Atlas (TCGA) [7, 13]. The data set contained genetic expression values for patients and clinical data. The clinical data contained information about whether or not a patient had a relapse of cancer and if they did how long after paclitaxel treatment. We used this information to form labels for the patients, splitting them into patients that are sensitive and resistant to the drug. Patients who did not at all relapse were classified as sensitive. Patients who relapsed within a year of treatment were classified as resistant. Patients who relapsed after a year from treatment were classified as sensitive.

The SVM model was used to predict the clinical outcome of the TCGA data patients, the results are shown in Figure 3.7. The accuracy is the percentage of the patients that were correctly classified. The recall is the percentage of correctly classified patients who did not relapse. The specificity is the percentage of correctly classified patients who did re-lapse. The ovary and breast patient data set results are very good; however the recall is low. This means the model is not able to predict which patients will not relapse, however it is good at predicting which patients will relapse. The lung data set contained mostly patients that did not relapse, which is why the overall accuracy for that data set was so low. The specificity however is comparable along all three patient data sets, suggesting that the model consistently is able to identify patients that will relapse. This means the model can identify which patients will not benefit from paclitaxel.

Hatzis et al. Data

We wanted to compare the predictive ability of the 47 breast cancer cell line Gaussian SVM model and the 414 multiple cancer cell line Gaussian SVM on patient data, so we used the

47 Breast Cancer Cell Line vs. 414 Multiple Cancer Cell Line Data Sets

Misclassification Percentage

Cross Validation	47 Cell Line Gaussian SVM	414 Cell Line Gaussian SVM
Leave-one-out	10.64%	12.08%
12-fold	15.31%	12.34%
10-fold	16.52%	12.36%
8-fold	18.59%	12.47%

The k-fold percentages are averaged over 1000 iterations to account for k-fold generation variance.

47 Cell Line Optimized Gaussian SVM

C:	10,000
Sigma:	100
Features:	ABCB11, ABCC1, BCAP29, BCL2L1, C7orf23/TMEM243, CNGA3, CYP3A4, FN1, MAP4, MAPT, OPRK1, TLR6, TUBB1

414 Cell Line Optimized Gaussian SVM

C:	100
Sigma:	10
Features:	ABCB1, BBC3, BCAP29, BCL2L1, BMF, FGF2, FN1, GBP1, MAP2, NFKB2, TUBB4B

Figure 3.6: Optimized Gaussian SVMs trained on 47 breast cancer cell lines and 414 multiple cancer type cell lines.

TCGA Patient Data Prediction Results for 414 Cell Line Trained SVM

Patient Data Set	Accuracy	Recall	Specificity
Ovary (219 Patients)	72%	8%	93%
Lung (37 Patients)	49%	0%	86%
Breast (191 Patients)	90%	17%	92%

Figure 3.7: 414 multiple cancer type cell line trained SVM prediction results on TCGA patient data.

Hatzis et al. (2011) Data Set Prediction Results

SVM Model	Accuracy	Recall	Specificity
47 breast cancer cell line SVM	65.20%	6.45%	79.38%
414 multiple cancer type cell line SVM	62.00%	34.00%	68.00%

Figure 3.8: Comparison of prediction results for 414 multiple cancer type cell line SVM and 47 breast cancer cell line SVM on Hatizis et al. [14] data set.

multiple cancer type model to predict the patient outcome for the Hatzis et al. [14] patient data set. The comparison of the results is shown in Figure 3.8. The accuracy of the multiple cancer type cell line SVM was slightly worse, however the recall is slightly better. These two results also show that specificity is continuously higher than recall for these models.

Molecular Taxonomy of Breast Cancer International Consortium (METABRIC) Data

We also obtained an additional two paclitaxel treated breast cancer patient data sets. We obtained these data sets from the Molecular Taxonomy of Breast Cancer International Consortium (METABRIC) [18]. These data sets contained genetic expression and clinical data containing information about relapse and how long after treatment it occurred. We classified the patients into sensitive and resistant classes the same way we did for the TCGA patients.

The patient data set was missing the gene TUBB4B that was present in our 414 cell line trained SVM. We used the sequential backward feature selection algorithm to form a new model that excludes this gene. The model contained following genes: ABCC1, ABCC10, BCL2L1, CNGA3, CYP2C8, CYP3A4, FGF2, FN1, MAP2, MAP4, NFKB2, OPRK1, TUBB1,

METABRIC Patient Data Prediction Results for 414 Cell Line Trained SVM

Patient Data Set	Accuracy	Recall	Specificity
Breast (123 Patients)	74%	0%	74%
Breast (87 Patients)	85%	50%	86%

Figure 3.9: 414 multiple cancer type cell line trained SVM prediction results on METABRIC patient data.

TWIST1. The model had a C value of 100 and σ value of 10. Its leave-one-out cross-validation score was 25.85%, the 12-fold cross-validation score was 27.41%, the 10-fold cross-validation score was 27.65%, and the 27.65 8-fold cross-validation score was 27.94%. This model performed worse on the cell line data set compared to the model that contained the gene TUBB4B. This suggests that the gene might be important for prediction. The decrease in performance could also be due to the limitations of the sequential backward feature selection algorithm.

The SVM model was used to predict the clinical outcome of the METABRIC data patients, the results are shown in Figure 3.9. The accuracy is good on both data sets. Similar to the performance on the TCGA patient data, the specificity is significantly higher than the recall. Very few patients in the 123 patient breast cancer data set did not relapse, which explains the 0% recall, since the model has greater predictive power for patients who do relapse.

3.3 Regression

We performed regression on the 47 cell line breast cancer data set from Daemen et al. [11]. The variable we were trying to fit was GI_{50}. The range of the GI_{50} variable was from 6.64 to 8.38. We initially performed regression with the full set of features, however the results were unsatisfactory. The mean-squared-error of the trained model was 0.1315, which appears to be a good result. However the model would predict almost the same GI_{50} value for each cell line. The leave-one-out cross-validation predictions were all within 8±0.015. These predictions are not useful since the model is not able to distinguish between cell lines with different levels of GI_{50}, instead it is predicting the same value for each cell line. We then performed sequential backward feature selection to reduce the number of features in order to improve results. The mean-squared-error after feature selection was 0.1280, which is slightly better than the result from using all the features. The leave-one-out cross-validation predictions were all within 8±0.0131. The results were still unsatisfactory so we decided to try using polynomial kernels since the relationship of the data is unlikely to be linear. We tried different degrees of polynomial kernels, along with sequential backward feature selection. The best polynomial kernel result had mean-squared-error of 0.1139, which is better than the linear models. The leave-one-out cross-validation predictions were all within 7.6±0.6930. The polynomial model

was able to predict a wider range of values however when compared with the actual cell line GI_{50} values the model was not able to distinguish between cell lines with higher and lower GI_{50} values.

The unsatisfactory results were probably due to the large number of features and the small size of the data set. Regression is a harder problem thus more data is required for the model to learn the relationship between the continuous variable GI_{50} and genetic expression. A common result from training regression models was the model predicting a value close to the mean GI_{50} for every cell line. Since the objective function that the model tries to minimize during training is mean-squared-error, we believe the model was predicting values close to the mean as it does not have enough information to learn the actual relationship in the data; this was the best way the model could minimize the mean-squared-error.

Training the regression model with the custom pathway based kernel did not give significantly better results than the polynomial kernel. We also tried neural network regression and SVM regression, but neither of the techniques produced meaningful results. We concluded that the data sets for cell lines are too small for the number of features in order to be able to accurately train regression models.

3.4 Genetic Feature Selection Algorithm Results

3.4.1 Cell Line Training Results

In order to evaluate the genetic algorithm for feature selection, we wanted to compare its performance to the results from the sequential backward feature selection algorithm. We used the genetic algorithm to perform feature selection on the 47 breast cancer paclitaxel treated cell line data. The results are shown in Figure 3.10. For both the genetic algorithm and the sequential backward feature selection algorithm the evaluation function that is used to evaluate gene subsets is leave-one-out cross-validation. The reason for this is that it always gives the same misclassification score, since it explores all combinations of the data. Smaller fold cross-validation generates the folds randomly, thus there is lots of variance in the misclassification score, depending on how the data was split into the folds. This requires the folds to be generated many times in order to eliminate the variance. The genetic algorithm was able to achieve the lowest leave-one-out cross-validation score of 2.13%. The 12 and 10-fold cross-validation scores are a little bit better than the sequential backward feature selection algorithm. However, the 8-fold cross-validation score is slightly worse. Since the feature selection algorithms aim to minimize the leave-one-out cross-validation score, its value is the one that indicates the performance of the feature selection algorithm.

3.4.2 Patient Prediction Results

In order to test the SVM model formed by using the genetic algorithm on the Hatzis et al. [14] patient data set we needed to redo the feature selection process with the genes BMF and CSAG2 left out from the initial set, since they are not present in the data set. The comparison of the results of the genetic algorithm and sequential backward feature selection algorithms with the genes BMF and CSAG2 left out is shown in Figure 3.11. The genetic algorithm

Feature Selection Algorithm Comparison

Misclassification Percentage

Cross Validation	**SBFS** Original Optimized Gaussian SVM	**GA** Optimized Gaussian SVM	**SA** Optimized Gaussian SVM
Leave-one-out	06.38%	02.13%	4.26%
12-fold	12.11%	11.40%	14.61%
10-fold	13.75%	13.74%	17.40%
8-fold	15.74%	16.92%	20.70%

The k-fold percentages are averaged over 1000 iterations to account for k-fold generation variance.

Sequential Backward Feature Selection (SBFS) Optimized Gaussian SVM

Box Constraint: 100,000
Sigma: 100
Features: ABCB11, BAD, BCL2L1, BMF, CNGA3, FGF2, FN1, MAP2, MAP4, MAPT, NR1I2, SLCO1B3, TUBB1, TWIST1

Genetic Algorithm (GA) Optimized Gaussian SVM

Box Constraint: 100,000
Sigma: 100
Features: ABCB11, ABCC10, BAD, BBC3, BCL2L1, BMF, CYP2C8, FGF2, GBP1, MAP2, MAP4, MAPT, NFKB2, NR1I2, SLCO1B3, TUBB4/TUBB4A, TWIST1

Simulated Annealing (SA) Optimized Gaussian SVM

Box Constraint: 100,000
Sigma: 100
Features: ABCB11, BAD, BCAP29, BCL2L1, BMF, C7orf23/TMEM243, CNGA3, CYP3A4, FGF2, FN1, GBP1, MAP2, MAP4, MAPT, NR1I2, OPRK1, SLCO1B3, TLR6, TUBB1, TUBB2C/TUBB4B, TUBB4/TUBB4A

Figure 3.10: Feature selection method comparison on paclitaxel breast 47 cell line data.

outperformed the sequential backward feature selection algorithm since it was able to achieve a lower leave-one-out cross-validation score.

We used the genetic algorithm-derived SVM Gaussian model to predict the outcome of the patients in the Hatzis et al. [14] patient data set. The comparison of the results of the genetic algorithm and sequential backward feature selection models is shown in Figure 3.12. The genetic algorithm model had a very good accuracy but it classified every patient as resistant. It is not clear why the model has 100% specificity and 0% recall, however all the models that were tested on patient data have had a bias for specificity.

3.5 Simulated Annealing Feature Selection Results

3.5.1 Tuning Algorithm Parameters

We experimented with a range of values for the percentage of genes that get selected for crossover. We tried values between 10% and 95%. We found that 70% prevented early convergence and allowed the algorithm to find gene subsets with low misclassification percentages. This means that only 70% of the chromosomes undergo crossover to generate offspring, the rest just get placed into the new generation unaltered.

For crossover we found that single-point crossover performed better. We also tried two-point crossover but it would cause the populations to not converge to better solutions, the misclassification of each following generation would not improve. This is what is expected since more crossover points cause the information about each solution to get scattered [31].

We experimented with different probabilities for the mutation rate and found that too high values prevent the genetic algorithm from converging, where as values that are too small cause early convergence to suboptimal answers. We found 60% mutation rate to give the best results for our data.

The chromosomes that were generated from crossover along with the chromosomes that were not selected for breeding undergo mutation. We initially used these chromosomes as our new generation. However, we found that when doing this, the new generations would not improve significantly compared to previous generations. This is because the best solution in each generation can undergo crossover and combine with a worse solution. This results in that optimal solution being lost. In order to prevent this, we introduced elitism. We would remove the worst member of the newly generated population and replace it with the best member of the previous generation. This made a great impact on achieving better results.

The number of generations required to reach a satisfactory solution vary for each run. We found that sometimes we reached optimal solutions within 500 generations and other times did not get any good solutions by 2000 generations. Solutions were deemed to be good when the misclassification percentage was lower than that achieved by the sequential backward feature selection algorithm. So we found that running the algorithm several times for 2000 generations would usually find a solution with a good misclassification score.

Feature Selection Algorithm Comparison

(Excluding genes BMF and CSAG2)

Misclassification Percentage

Cross Validation	SBFS Optimized Gaussian SVM	GA Optimized Gaussian SVM	SA Optimized Gaussian SVM
Leave-one-out	10.64%	8.51%	8.51%
12-fold	15.31%	16.16%	16.46%
10-fold	16.52%	17.84%	18.51%
8-fold	18.59%	20.46%	21.20%

The k-fold percentages are averaged over 1000 iterations to account for k-fold generation variance.

Sequential Backward Feature Selection (SBFS) Optimized Gaussian SVM

C: 10,000
Sigma: 100
Features: ABCB11, ABCC1, BCAP29, BCL2L1, C7orf23/TMEM243, CNGA3, CYP3A4, FN1, MAP4, MAPT, OPRK1, TLR6, TUBB1

Genetic Algorithm (GA) Optimized Gaussian SVM

C: 100,000
Sigma: 100
Features: ABCB11, BCAP29, BCL2L1, BIRC5, C7orf23/TMEM243, CYP2C8 FN1 MAP2, MAP4 MAPT SLCO1B3, TWIST1

Simulated Annealing (SA) Optimized Gaussian SVM

C: 100,000
Sigma: 100
Features: ABCB11, BAD, BBC3, BCL2, BCL2L1, CNGA3, FGF2, FN1, MAP4, MAPT, NFKB2, OPRK1, SLCO1B3, TUBB1, TWIST1

Figure 3.11: Feature selection method comparison on paclitaxel breast 47 cell line data with genes BMF and CSAG2 left out from initial gene set.

Hatzis et al. (2011) Feature Selection Comparison Predictions

Feature Selection Method	Accuracy	Recall	Specificity
Sequential Backward Selection	65.20%	6.45%	79.38%
Genetic Algorithm	80.00%	0.00%	100.00%
Simulated Annealing	80.56%	0.00%	100.00%

Figure 3.12: Hatzis et al. [14] patient prediction results for SVM models formed using different feature selection methods.

3.5.2 Cell Line Results

We wanted to see if simulated annealing is a superior method for feature selection compared to sequential backward feature selection. In order to do this, we acquired several cell line data sets that were treated with different drugs from the Daemen et al. [11] supplementary data. We acquired cell line data for three different drugs: doxorubicin, methotrexate, and tamoxifen. We then used both the sequential backward feature selection algorithm and simulated annealing to create optimized Gaussian SVM models. The cell line data sets used contained only genes that correlated with GI_{50}. The results of the models performance are shown in Figure 3.13. The simulated annealing feature selection algorithm was able to outperform the sequential backward feature selection algorithm for doxorubicin and tamoxifen. Methotrexate had only 8 initial genes, where as doxorubicin had 16, and tamoxifen had 24. One of our goals of using simulated annealing was to find models with lower C and σ values, however for methotrexate this was not possible due to the very few GI_{50} correlated genes.

The initial gene selection performed by Dorman et al. [12] involved a step where genes where eliminated if their expression levels did not correlate with GI_{50}. We hypothesized that due to epistasis, there could be non-linear relationships that exist between the genes. We performed feature selection using the simulated annealing algorithm on all three drugs with the initial genes sets containing all the genes that correlate with GI_{50} and those that did not. The results of the Gaussian SVM models are shown in Figure 3.13. The addition of the genes that did not correlate with GI_{50} improved the leave-one-out cross-validation for all three drugs. Additionally, the models derived from sequential backward feature selection call had a C value of 100,000 and σ value of 100, as those values minimized misclassification. When using simulated annealing for all three data sets the C value was 100 and σ was 10. This indicates that the simulated annealing feature selection algorithm was able to select better features, since they contained a simpler relationship.

We also compared the simulated annealing, genetic algorithm and sequential backward feature selection on the 47 breast cancer paclitaxel treated data set from Daemen et al. [11]. We used the simulated annealing to perform feature selection on the data set and derived a Gaussian

Comparing Feature Selection Methods on Cell Line Data

Method	Doxorubicin Leave-one-out Misclassification	Methotrexate Leave-one-out Misclassification	Tamoxifen Leave-one-out Misclassification
SVM trained on subset of genes that correlate with GI50 using sequential backward feature selection	12.5%	15.5%	17.8%
SVM trained on subset of genes that correlate with GI50 using simulated annealing	7.7%	18.4%	11.1%
SVM trained on all available genes using simulated annealing	2.6%	0.0%	0.0%

Figure 3.13: Comparison of SVM models derived using different feature selection methods for doxorubicin, methotrexate and tamoxifen cell line data sets.

SVM model. The comparison of the results is shown in Figure 3.10. The simulated annealing feature selection algorithm had a leave-one-out cross-validation result that was between the genetic algorithm value and the sequential backward feature selection value. If we wanted to achieve a score that surpassed that of the genetic algorithm we would need to run the algorithm more times, however the scores were satisfactory for showing that the simulated annealing is superior to sequential backward feature selection.

3.5.3 Patient Results

In order to test the SVM model formed by using the simulated annealing algorithm on the Hatzis et al. [14] patient data set we needed to redo the feature selection process with the genes BMF and CSAG2 left out from the initial set. The comparison of the results of the simulated annealing algorithm, genetic algorithm and sequential backward feature selection algorithms with the genes BMF and CSAG2 left out is show in Figure 3.11. The simulated annealing algorithm had results that were equivalent to the genetic algorithm.

We used the simulated annealing derived optimized Gaussian SVM model to predict the patient outcome of the Hatzis et al. [14] patient data set. The comparison of the results of the simulated annealing, genetic algorithm, and sequential backward feature selection models is show in Figure 3.12. The simulated annealing algorithm had results that were equivalent to the genetic algorithm.

Chapter 4

Discussion

In this thesis we examined the use of machine learning techniques to predict chemotherapy response in cell lines and in patients. These are two different problems that are linked by the hypothesis that cancer cell lines mirror the behaviour of cancerous tumours in the human body. Under this assumption if we can build models that capture the relationship between genetic expression and cancer cell line chemotherapy response, we can use these models to predict the clinical outcome of patients.

Our first goal was to improve the current methods of learning the relationship between genetic expression and chemotherapy response in cancer cell line data. Previous studies have used binary labels for classifying the chemotherapy response [11, 12]. We wanted to be able to predict a continuous variable such as GI_{50}, instead of a binary label. Models that could predict GI_{50} or a similar continuous variable that measures drug response would be a lot more valuable since they would give more information about the response of the patient. This led us to using machine learning regression techniques to try and form a model that can do this. We were unsuccessful at developing models that give good results due to the limited size of cancer cell line data sets. After our unsuccessful regression techniques, we tried multi-class SVM classification, which also had unsatisfactory results, due to the size of the cancer cell line data. Most cell line data sets that are currently available for specific drugs have under 100 samples. This is one of the biggest challenges in our research as it limits how much information we can learn from the data. We concluded that for the available cancer cell line data sizes, binary classification was the best method to use for predicting chemotherapy response. We compared NNs to SVMs and found that SVMs were a superior machine learning technique for binary classification on these data sets. SVMs have fewer parameters to tune and produce better results. We then compared the use of the linear kernel to the Gaussian kernel for SVMs. We concluded that the Gaussian kernel with optimized C and σ values is superior since it can fit non-linear data as well as fit any data the linear kernel can.

After we concluded that a binary classifier, specifically an optimized Gaussian kernel SVM, is best able to learn the cancer cell line data we looked at other ways to improve prediction results. The feature selection method we initially used was sequential backward feature selection, which is what Dorman et. al (2016) used in their models [12]. This feature selection algorithm is very popular and produces good results. However, it explores a limited amount of the search space of possible feature subsets. We researched alternative feature selection techniques and decided to use genetic algorithms for feature selection. Genetic algorithms explore a lot more

the search space and are able to re-introduce a gene that is eliminated during previous steps of the feature selection process, which is one of the major issues with sequential backward feature selection. Genetic algorithms consistently outperformed sequential backward feature selection. We then looked at simulated annealing for feature selection, as it required less parameter tuning and produced results that were on par with genetic algorithms. The only drawback of simulated annealing is that it is computationally expensive, as a result its run time is a lot longer than the genetic algorithm. Simulated annealing is superior to backward feature selection as it is able to not only find models with better accuracy but also ones with lower values of C and σ values. This means that the features the simulated annealing algorithm selects contain a simpler relationship which requires less curves in the hyperplane and allows the hyperplane to have a further distance from data points. Therefore, these models are more likely to generalize to unseen data, and patient data. We concluded that simulated annealing was the superior method for feature selection for cancer cell line data.

All sequential backward feature selection derived models performed better for specificity than for recall, while simulated annealing and genetic algorithms had 100% specificity and 0% recall. We are not sure why there is a bias for specificity, however this was a recurrent pattern in our study and previous studies [12]. This is something future studies should research as it may uncover something important about the transfer of predictive ability of cell line trained machine learning models to patients.

When we combined several cancer cell line types that were all treated with paclitaxel into one data set we hypothesized that the method of action of the drug is similar across the different cancer types. The results from the optimized Gaussian SVM models confirmed this hypothesis, since the model was able to achieve good results that were consistent among different sizes of k-fold cross-validation. The model trained on this data was the only model which had a consistent score among the decreasing sizes of k-Fold cross-validation, showing that a larger data set is really beneficial. Additionally, the SVM model trained on the 414 multi cancer cell line data set had a C value of 100 and a σ value of 10, compared to the C value of 100,000 and σ value of 100 for the smaller 47 breast cancer cell line SVM. This shows that the larger data set allowed for a simpler relationship to be learned which is more likely to generalize to unseen data and patient data. The patient prediction results of the SVM model trained on the large data set were good, suggesting that the relationship it learned in the cell line data transfers to patients. Therefore, the acquisition of larger cell line data sets is a very important goal for future studies, along with exploring combined cell line data sets from different cancer types.

For the problem of predicting chemotherapy response in cell lines our greatest challenge was that data sets were too small, except for the pan-cancer data set we created by combining several data sets. Our greatest challenge for predicting chemotherapy response in patients was evaluating how well the cell line trained models transfer to patients. The challenge is that when the cell line trained model incorrectly classified a patient, it is not easy to know if the SVM predicted wrong or some external factor other than genetic expression affected the patient outcome. The patient could have had external factors such as other illnesses, unhealthy lifestyle, age, and other things that would affect chemotherapy response. This makes it difficult since patient clinical data is limited and does not contain enough information to decide whether external factors caused the chemotherapy response. Additionally, since we do not have GI_{50} values for patients, we must classify patients as resistant and sensitive based on time to death or relapse clinical data. This adds another element of uncertainty in validating cell-line-trained

models on patients. Another challenge with patients is that they are normally treated with several drugs simultaneously and undergo other treatments. This also adds to the challenge of evaluating the cell line trained model's ability to transfer to patients, since the models only learn the relationship between genetic expression and a single drug's outcome. Future studies need to consider synergistic drug relationships and find ways to deal with the extraneous variables that affect patient chemotherapy response. With the ultimate goal of figuring out ways to more effectively evaluate the ability of cell line trained models to predict patient chemotherapy response.

In this thesis we examined the predictive ability of genetic expression, which has been shown by previous studies to be one of the most effective molecular compounds to use for modelling response to therapy [11, 12]. Our many good results confirm that genetic expression is a very good predictor for chemotherapy response. The reason why genetic expression is good predictor is that it is an indicator of how active a certain gene is in a cell. However, genetic expression does not always exactly correlate with the level of activity of the gene in a specific cell. Genetic expression measures the amount of messenger RNA in the cell, however its not clear how much of the messenger RNA in the cell will actually be used by the ribosomes to create proteins. We think that future studies should also look to alternative means to measure the activity level of a gene other than genetic expression. Additionally, future studies should try combining different predictive features other than just genetic expression in order to have more information about the state of the cell.

There is currently no gold standard for chemotherapy. Personalized drug cocktails could help reduce the number of patients who are resistant to therapy and increase the overall chance of survival for cancer patients. Machine learning predictors trained on cell line data show promising results and should continue to be researched as they could one day be used to select the chemotherapy drugs that are most likely to work for a specific patient based on their genetics.

Bibliography

[1] Ethem Alpaydin. *Introduction to machine learning*. The MIT Press, Cambridge, MA, third edition, 2014.

[2] Francisco Azuaje. Computational models for predicting drug responses in cancer research. *Briefings in Bioinformatics*, page bbw065, 2016.

[3] Jordi Barretina, Giordano Caponigro, Nicolas Stransky, Kavitha Venkatesan, Adam A. Margolin, Sungjoon Kim, Christopher J. Wilson, Joseph Lehr, Gregory V. Kryukov, Dmitriy Sonkin, Anupama Reddy, Manway Liu, Lauren Murray, Michael F. Berger, John E. Monahan, Paula Morais, Jodi Meltzer, Adam Korejwa, Judit Jan-Valbuena, Felipa A. Mapa, Joseph Thibault, Eva Bric-Furlong, Pichai Raman, Aaron Shipway, Ingo H. Engels, Jill Cheng, Guoying K. Yu, Jianjun Yu, Peter Aspesi Jr, Melanie De Silva, Kalpana Jagtap, Michael D. Jones, Li Wang, Charles Hatton, Emanuele Palescandolo, Supriya Gupta, Scott Mahan, Carrie Sougnez, Robert C. Onofrio, Ted Liefeld, Laura MacConaill, Wendy Winckler, Michael Reich, Nanxin Li, Jill P. Mesirov, Stacey B. Gabriel, Gad Getz, Kristin Ardlie, Vivien Chan, Vic E. Myer, Barbara L. Weber, Jeff Porter, Markus Warmuth, Peter Finan, Jennifer L. Harris, Matthew Meyerson, Todd R. Golub, Michael P. Morrissey, William R. Sellers, Robert Schlegel, and Levi A. Garraway. The cancer cell line encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483(7391):603–607, 2012.

[4] Asa Ben-Hur and Jason Weston. A user's guide to support vector machines. *Methods in molecular biology (Clifton, N.J.)*, 609:223–239, 2010.

[5] C. H. G. Beurskens, C. v. Uden, L. J. Strobbe, R. A. B. Oostendorp, and Th Wobbes. The efficacy of physiotherapy upon shoulder function following axillary dissection in breast cancer, a randomized controlled study. *BMC Cancer*, 7(1):166–166, 2007.

[6] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[7] Ethan Cerami, Jianjiong Gao, Ugur Dogrusoz, Benjamin E Gross, Selcuk Onur Sumer, Bülent Arman Aksoy, Anders Jacobsen, Caitlin J Byrne, Michael L Heuer, Erik Larsson, et al. The cbio cancer genomics portal: an open platform for exploring multidimensional cancer genomics data. *Cancer discovery*, 2(5):401–404, 2012.

[8] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):1–27, 2011.

[9] Francis Collins. Cancer: a disease of the genome, 2007.

[10] James C Costello, Laura M Heiser, Elisabeth Georgii, Mehmet Gönen, Michael P Menden, Nicholas J Wang, Mukesh Bansal, Petteri Hintsanen, Suleiman A Khan, John-Patrick Mpindi, et al. A community effort to assess and improve drug sensitivity prediction algorithms. *Nature biotechnology*, 32(12):1202–1212, 2014.

[11] Anneleen Daemen, Obi L. Griffith, Laura M. Heiser, Nicholas J. Wang, Oana M. Enache, Zachary Sanborn, Francois Pepin, Steffen Durinck, James E. Korkola, Malachi Griffith, Joe S. Hur, Nam Huh, Jongsuk Chung, Leslie Cope, Mary J. Fackler, Christopher Umbricht, Saraswati Sukumar, Pankaj Seth, Vikas P. Sukhatme, Lakshmi R. Jakkula, Yiling Lu, Gordon B. Mills, Raymond J. Cho, Eric A. Collisson, Laura J. van't Veer, Paul T. Spellman, and Joe W. Gray. Modeling precision treatment of breast cancer. *Genome biology*, 14(10):R110, 2013.

[12] SN Dorman, K. Baranova, JHM Knoll, BL Urquhart, G. Mariani, ML Carcangiu, and PK Rogan. Genomic signatures for paclitaxel and gemcitabine resistance in breast cancer derived by machine learning. *MOLECULAR ONCOLOGY*, 10(1):85–100, 2016.

[13] Jianjiong Gao, Bülent Arman Aksoy, Ugur Dogrusoz, Gideon Dresdner, Benjamin Gross, S Onur Sumer, Yichao Sun, Anders Jacobsen, Rileen Sinha, Erik Larsson, et al. Integrative analysis of complex cancer genomics and clinical profiles using the cbioportal. *Science signaling*, 6(269):pl1, 2013.

[14] Christos Hatzis, Lajos Pusztai, Vicente Valero, Daniel J. Booser, Laura Esserman, Ana Lluch, Tatiana Vidaurre, Frankie Holmes, Eduardo Souchon, Hongkun Wang, Miguel Martin, Jos Cotrina, Henry Gomez, Rebekah Hubbard, J. I. Chacn, Jaime Ferrer-Lozano, Richard Dyer, Meredith Buxton, Yun Gong, Yun Wu, Nuhad Ibrahim, Eleni Andreopoulou, Naoto T. Ueno, Kelly Hunt, Wei Yang, Arlene Nazario, Angela DeMichele, Joyce OShaughnessy, Gabriel N. Hortobagyi, and W. F. Symmans. A genomic predictor of response and survival following taxane-anthracycline chemotherapy for invasive breast cancer. *JAMA*, 305(18):1873–1881, 2011.

[15] Zena M. Hira and Duncan F. Gillies. A review of feature selection and feature extraction methods applied on microarray data. *Advances in Bioinformatics*, 2015:1–13, 2015.

[16] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[17] National Cancer Institute. Drugs approved for breast cancer. `https://www.cancer.gov/about-cancer/treatment/drugs/breast`, March 2016.

[18] F. Iorio, TA Knijnenburg, DJ Vis, GR Bignell, MP Menden, M. Schubert, N. Aben, E. Goncalves, S. Barthorpe, H. Lightfoot, T. Cokelaer, P. Greninger, E. van Dyk, H. Chang, H. de Silva, H. Heyn, XM Deng, RK Egan, QS Liu, T. Mironenko, X. Mitropoulos, L. Richardson, JH Wang, TH Zhang, S. Moran, S. Sayols, M. Soleimani, D. Tamborero, N. Lopez-Bigas, P. Ross-Macdonald, M. Esteller, NS Gray, DA Haber,

MR Stratton, CH Benes, LFA Wessels, J. Saez-Rodriguez, U. McDermott, and MJ Garnett. A landscape of pharmacogenomic interactions in cancer. *CELL*, 166(3):740–754, 2016.

[19] Reza Jahanbakhshi, Reza Keshavarzi, Mahdi Aliyari Shoorehdeli, Abolqasem Emamzadeh, et al. Intelligent prediction of differential pipe sticking by support vector machine compared with conventional artificial neural networks: An example of iranian offshore oil fields. *SPE Drilling & Completion*, 27(04):586–595, 2012.

[20] In Sock Jang, Elias Chaibub Neto, Justin Guinney, Stephen H Friend, and Adam A Margolin. Systematic assessment of analytical methods for drug sensitivity prediction from cancer cell line data. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, page 63. NIH Public Access, 2014.

[21] S. S. Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computation*, 15(7):1667–1689, 2003.

[22] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *NATURE*, 521(7553):436–444, 2015.

[23] SY Lee, SA Im, YH Park, SY Woo, S. Kim, MK Choi, W. Chang, JS Ahn, and YH Im. Genetic polymorphisms of slc28a3, slc29a1 and rrm1 predict clinical outcome in patients with metastatic breast cancer receiving gemcitabine plus paclitaxel chemotherapy. *EUROPEAN JOURNAL OF CANCER*, 50(4):698–705, 2014.

[24] Shelley McGuire. World cancer report 2014. geneva, switzerland: World health organization, international agency for research on cancer, who press, 2015. *Advances in nutrition (Bethesda, Md.)*, 7(2):418, 2016.

[25] Funda Meric-Bernstam, Carol Farhangfar, John Mendelsohn, and Gordon B. Mills. Building a personalized medicine infrastructure at a major cancer center. *Journal of Clinical Oncology*, 31(15):1849–1857, 2013. PMID: 23589548.

[26] Connie Oshiro, Sharon Marsh, Howard McLeod, Michelle W. Carrillo, Teri Klein, and Russ Altman. Taxane pathway. *Pharmacogenetics and Genomics*, 19(12):979–983, 2009.

[27] Gaurang Panchal, Amit Ganatra, YP Kosta, and Devyani Panchal. Behaviour analysis of multilayer perceptronswith multiple hidden neurons and hidden layers. *International Journal of Computer Theory and Engineering*, 3(2):332, 2011.

[28] Edith A Perez. Paclitaxel in breast cancer. *The Oncologist*, 3(6):373–389, 1998.

[29] Chemotherapy Principles. An in depth discussion of the techniques and its role in cancer treatment. *American Cancer Society*, 2011.

[30] Sandeep Rajput, Lisa D. Volk-Draper, and Sophia Ran. Tlr4 is a novel determinant of the response to paclitaxel in breast cancer. *Molecular Cancer Therapeutics*, 12(8):1676–1687, 2013.

[31] S. N. Sivanandam, S. N. Deepa, SpringerLink (Online service), and Inc Books24x7. *Introduction to Genetic Algorithms*. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 2008;2007;.

[32] Steven S. Skiena and SpringerLink (Online service). The algorithm design manual, 2008.

[33] BA Weaver. How taxol/paclitaxel kills cancer cells. *MOLECULAR BIOLOGY OF THE CELL*, 25(18):2677–2681, 2014.

Curriculum Vitae

Name:	Dimo Angelov
Post-Secondary Education and Degrees:	University of Western Ontario London, ON 2015 - 2017 M.Sc. University of Western Ontario London, ON 2011 - 2015 B.Sc.
Related Work Experience:	Teaching Assistant The University of Western Ontario 2015 - 2017 Research Assistant The University of Western Ontario 2015 - 2017

Publications:

Rezaeian, I., Mucaki, E. J., Baranova, K., Pham, H. Q., Angelov, D., Ngom, A., Rogan, P. K. (2016). Predicting Outcomes of Hormone and Chemotherapy in the Molecular Taxonomy of Breast Cancer International Consortium (METABRIC) Study by Biochemically-inspired Machine Learning. F1000Research, 5.