Optimization Theory
Machine Learning Project

Mirela Desa
Connor Pitchford

November 26, 2019

1 Introduction

Machine learning is defined in the Journal of Machine Learning Research as "the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead". Machine learning is programming computers to optimize a performance, using a function that minimizes an error or one that maximizes reward over punishment. Optimization is an essential part of machine learning. First procedure is choosing the model. Then it is trained, by solving a core optimization problem, that selects the best variables and parameters for the model, with respect to the loss function. The last step includes the calculation of the model's accuracy using test data. New models employed in machine learning are developed with advances in mathematical theory, proving one more time that are both intertwined.

This report looks to examine machine learning, focusing on the article *Supervised Machine Learning: A Review of Classification Techniques*, by S. B. Kotsiantis. The report also examines a quadratic optimization application in a real-world problem (see Appendix A) and provides the code used (see Appendix B).

While there are several uses for machine learning, Kotsiantis focuses on data mining. Every instance of which is represented as a continuous, categorical, or binary algorithms. These three cases are then split into two further cases: supervised and unsupervised.

2 Description of Supervised Learning Methodology

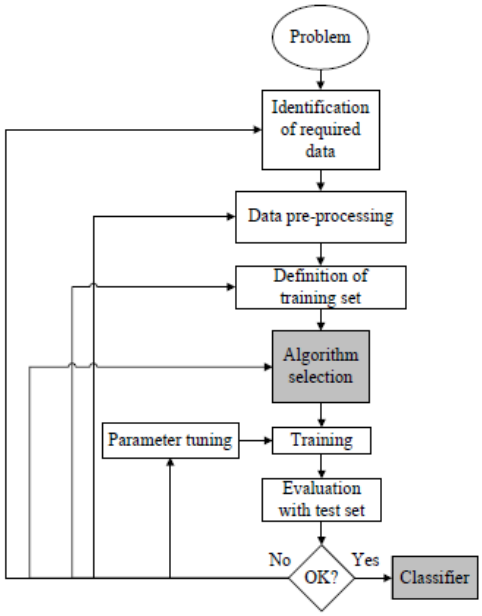
Most machine learning problems are supervised or unsupervised. Supervised learning builds on a model for predicting, or estimating an output based on one or more inputs. It is broadly used in business, medicine, astrophysics and public policy. For unsupervised learning, we have inputs, but the algorithm acts without guidance with unsupervised outputs.

Machine learning algorithms predict variables, which can be continuous or quantitative; this is often referred to as a regression problem. With features that are categorical or qualitative, it is known as a classification problem.

First step in solving a real-world problem is to define it and collect the data set. We will use this data set to solve it. The data set in most cases needs to be pre-processed, to be cleaned of missing values and have duplicates and errors removed. Narrowing down the selection of instances can also take an infeasible problem and create a feasible one. The next step is selecting the relevant features and getting rid of the noise that commonly accompanies large data sets. Variable selection helps the algorithm to work faster and more efficiently, by removing the irrelevant and redundant features.

Choosing the learning algorithm is a crucial step. No one method dominates all others over all possible data sets. Some method may work best on a specific set, and another method may work better on a similar, but different data set. Selecting the best algorithm for the given setup is one of the most challenging part in machine learning. One must choose a balance between accuracy and computational time. Inductive machine learning is where the program learns a set of rules by being given instances (scenarios provided in a training set by an external source). This creates what is called a classifier. The classifier allows the program to generalize (or predict) results for new instances/ scenarios.

Kotsiantis provides the following illustration which shows the generic algorithm:



3 Introduction to Algorithm Accuracy

The choice of algorithm is crucial. No one method dominates all others over all possible data sets. Some methods may work best on a specific set, and another method may work better on a similar, but different data set. Ideally, one knows the types of fields or areas one can gather valuable information from the data. Otherwise, if this is not known, then a method called "brute-force" may be the best option. The brute-force method tries every single combination/ possibility. However, it has potential for a big downfall: noise and missing values. Due to this, the data set produced by the brute-force method requires significant pre-processing. Beyond brute-force, each algorithm creates a classifier. The classifier can be evaluated for accuracy by comparing its accuracy against the known accuracy of a test set. This can be done by splitting the training data into two parts, the first to be used to train the program, from which the algorithm creates the classifier. The program then uses the classifier on the remaining training data to test the performance and accuracy. For example, assessing accuracy of the classifier can be done in the regression setting, by calculating the mean squared error (MSE).

$$MSE = \frac{1}{n} \sum_i (y_i \quad \tilde{y})_i^2$$

Where $\tilde{y}_i$ is the prediction for the i^{th} observation. If the MSE is small, then the predicted response is close to the true response. Another way to minimize the expected test error is to use a method that achieves low bias and low variance. This is a hard balance to achieve, as the the relationship between the two is described by a bias-variance trade-off (low variance implies high variance and vice-versa).

Cross-Validation is another method of evaluating the model's performance. This technique also evaluates the predictive model by splitting the original data set into a training set and a test set to asset its accuracy. Cross-Validation includes Validation (data set is randomly split into a training set and a test set). The idea is to create divided subsets of equal sizes. The program is trained on the union of all the subsets except one. It is then tested on the remaining subset. This is repeated until the program has been tested on every subset. The error for the classifier as a whole is estimated as the average of the error for each subset. Obviously, the more subsets the better potential to reduce the error. Which leads us into the Leave-One-Out. The Leave-One-Out method works exactly the same as Cross-Validation except each subset is only a single instance. I.e. it divides the data set into n 1 training observations and one left for predictions (where n is the number of instances). This process is repeated n times. The increased accuracy does come at the expense of computational time and cost.

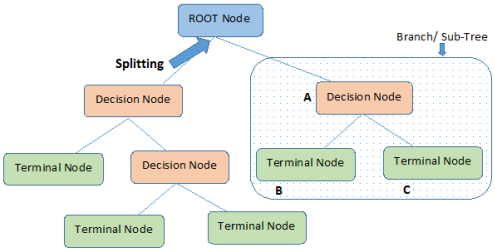
Once the error is determined, the user can decide if it is at an acceptable level. If yes, the algorithm continues. If not, then we must return to a previous stage in the algorithm and make changes. The issue of accuracy can derive from problems such as incorrect selection of features, a different size training set could be needed, the problem can have a high dimensionality, or an imbalanced data set. Of course, the problem may not lie in the data but in the algorithm. It could be an entirely new algorithm needs to be chosen (or created).

4 Logic Based Algorithms

We will present two groups of logical learning methods: decision trees and rule-based classifiers.

Decision trees stratify and segment the predictor space into more simple regions. To make a prediction for an observation, that is in the branches, we go to the conclusion about it in the leaves. Decision trees are used for both regression and classification problems.

An example of a decision tree (Le 2018):



Before we continue, we first wish to identify some terms. A root is the beginning of a tree (e.g. "Root Node" in the example). A branch is a directional path (i.e. an arrow). A leaf is a result or end (e.g. "Terminal Node" in the example).

A Root Node is the graphical representation of the entire sample. It then gets divided into two or more sets. Dividing the path is called splitting and this forms sub-nodes. When the tree splits at a sub-node, forming two or more paths, that node is called a decision node.

Decision trees are in general drawn upside down, the leaves are at the bottom of the tree, the points on the tree, where the independent variable split are called internal nodes. In building a tree, first step is to divide the regressor space, into distinct regions, for every observation that falls into a specific region, we make a prediction. The prediction will be based on the mean or mode of the dependent variable, calculated for the training data. The goal is to find the regions that minimize the residual sum of squares (once again, we see an optimization problem). The residual sum of squares (*RSS*) is given by

$$RSS = \sum_{j} \sum_{i} (y_i \quad \tilde{y}_j)^2$$

Where $\tilde{y}_j$ is the mean response for the training observation in j^{th} region).

What can occur is overfitting and underfitting of data. Given a learned hypothesis h we can determine if it is overfitted. If there exist another hypothesis h' that has a larger error when tested on the training data, but a smaller error on the entire data set (compared to h), then h is overfitted. Overfitting the data occurs when the model is too complex, and results in poor test performance.

To overcome overfitting a smaller tree (less regions and splits) with lower variance (but a bit more bias) we can draw a tree as long as a certain threshold. This will cause a certain decrease in the *RSS*. In other words, to prevent overfitting we simply stop training before it reaches that point.

Pruning is a more popular and better strategy, grow a very large tree and then prune it back to obtain a subtree. That is, we reduce the number of leaves and branches on the tree, which is refered to as pruning. Given two different trees which produce (or are expected to produce) similar results, the smaller tree is preferred. The best way to prune the tree is the one with the lowest error rate. It is easier to prune the tree as one goes and not at the end.

RIPPER is an example of such an algorithm, growing, pruning, and repeating. It grows strictly to the data and then prunes to avoid overfitting. Stopping or reducing the growth is easier than removing the branches and leaves afterwards. By easier, we mean computationally-wise, as there is substantial effort that is thrown out when a built portion of a tree is pruned. There is no identified method that is agreed upon to be the best pruning method.

Another example program is PUBLIC. This program determines if it is likely for a node to be pruned and if so, does not build out that section of the tree.

As a tree can be thought as a set of rules determining the path from root to leaf and given the above, the ideal situation is as few rules as possible. As a tree with many rules is likely a program trying to remember or brute-force all the information instead of trying to determine general assumptions and guidelines.

Classification Trees are a type of machine learning algorithms that are used to classify and to predict a qualitative response. We cannot use *RSS*, as a criterion for binary split, as in the quantitative problems, so it is used the

classification error rate (the fraction of training observations in a region that do not belong to the most common class).

Decision trees are primarily univariate as their branches split based upon single features at each node. This leads to the problem that decision trees have a low performance with diagonal partitioning. There are decision trees which are multivariate.

One significant problem with decision trees is the potential for duplicate nodes as the program could discover the same areas more than once. Thus, an algorithm needs a way to ensure it combines duplicated nodes.

One of the advantages of decision trees is their ease of understanding. One can easily follow the logic, the path, of the tree. Also, they can be easily represented graphically and interpreted by non-experts.

5 Perceptron-based Techniques

We will present single layered and multi-layered perceptrons. A single layered percepton is defined by Kotsiantis as:

> If x_1 through x_n are input feature values and w_1 through w_n are connection weights/prediction vector (typically real numbers in the interval $[-1, 1]$), then perceptron computes the sum of weighted inputs: $\sum_i x_i w_i$ and output goes through an adjustable threshold: if the sum is above threshold, output is 1; else it is 0.

The largest advantage for the perceptron algorithm is it is "anytime online". That is to say, there is no minimum amount of time required to produce results (however longer times do lead to better results). Thus, useful information can be discovered even for short run times. Perceptron algorithms are commonly used for batches of training instances and to run the algorithm through the training set again and again until the algorithm finds a prediction vector which meets the required standards. This prediction is then used on the test set.

Like many algorithms in mathematics, there is also a version using weighted arguments. WINNOW is an example of such an argument for as Kotsiantis states "The weights of the relevant features grow exponentially but the weights of the irrelevant features shrink exponentially." For this reason, WINNOW is capable of quickly changing to user preferences and requests for different target functions.

The single layered perceptrons are also superior with dealing with irrelevant features. It significantly reduces the required time, particularly if there is only a few relevant features needed.

Multilayered perceptrons are a collection of single layered perceptrons. Perceptron methods are binary. The method for multilayered perceptrons is to reduce it to multiple single (binary) problems. We find a hyperplane which separates instances into categories. If this can be done, then the instances are linearly separable and the problem is solvable. It is much more realistic that the instances are not linearly separable and no such hyperplane exists which can properly classify all the cases. This problem brought about Artificial Neural Networks (a type of multilayered perceptron). A neural network contains units, called neurons, joined together by connections. It usually consists of a great number of neurons and any number of connections (or paths). These paths have direction which leads to the different types of units. There are input units which receive information, and output units where the results are found, and lastly units which are called hidden.

The input-output mapping is first created by training the network on a set of paired data. Each connection has a designated weight. The weight is fixed to a set value. The trained network is then used to determine the classifications of a new set of data. The signal from the input units is transmitted throughout the entire network. This determines the activation value for each output unit. Each input unit also has an activation value and the unit sends this value to each hidden unit it is connected to. The hidden unit calculates its value based on the activation values it receives and sends this value to the output units. Every unit that calculates its own activation value does so by a defined function which accounts for all incoming activation values.

One of the major problems with ANN methods is determining the size of the unknown

layer (the depth of the hidden units). Estimating the layer to be smaller than it actually is leads to poor approximation and the program will struggle with generic capabilities. On the other side, overestimating the layer can result in overfitting, the dangers of which have already been discussed above.

The ANN methods depend on two fixed aspects and one flexible aspect. The methods require input and activation functions and a network architecture. While the user does have some control over the data input, these two things are primarily fixed. It is only over the third aspect, the weight of each input connection, that the user may define the behavior of the ANN. In general, the program is exposed to random weights and then improves itself by finding better weights while being exposed to the data again and again. These algorithms will have to go through multiple iterations. In general, it is a very efficient algorithm. Given n training instances and also given W weights, each repetition takes $O(nW)$ time. It can occur that the number of epochs is exponential to the number of inputs. Given that a data set can be very large and than one epoch is passing the entire data set through the network once, having an exponentially growing number of epochs can be very computationally costly. Without proper procedures in place, the end user may not realize what is happening when the program doesn't end in a timely matter. For this reason, we introduce fail safes. Rules that tell the algorithm to stop if the process is taking to long. Common rules include 1) stopping after a predetermined number of epochs, 2) stopping when a predetermined level of error is reached, 3) stopping when the error level has not changed for x number of epochs, 4) stopping when the error of the training data is more than the error on the training set (i.e. overfitting). But even then, these types of algorithms are too slow. One method to improve this is to try to guess the accurate starting weights (and thus reducing the number of iterations to reach an acceptable level of error). There are also Weight-elimination algorithms which derive topology that avoids overfitting. Another method is to simply remove weights. Pruning weights and nodes can greatly reduce a problem. A little counter-intuitive, but there is also a technique called constructive algorithms which add extra nodes to also achieve quicker run times.

In summary, Artificial Neural Networks are found by synaptic weight modification, modifications of the network structure, use of suitable stable points, and proper choice of activation functions.

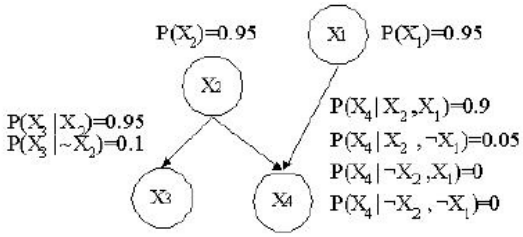
6 Statistical Learning Algorithms

Statistical learning algorithms have an explicit underlying probability model. They predict and assign the probability of an observation to a certain class.

An example is Linear Discriminant Analysis (LDA). This method is used in statistics, pattern recognition, and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events. The assumptions of discriminant analysis are multivariate normality (independent variables are normal for each level of the grouping variable), homoscedasticity, multicollinearity, and independence. It uses Bayes' Theorem to classify and it is trying to approximate the Bayes classifier. The LDA algorithm starts by finding directions that maximize the separation between classes, and then uses these directions to predict the class of individuals. These directions, called linear discriminants, are a linear combinations of predictor variables. The Bayes Classifiers are algorithms based on the Bayes' Theorem. The Bayes Classifier is a very simple algorithm that assigns each observation to the most likely class, given its treatment value. And by doing so, it is also minimizing the test error rate. A test observation belongs to a class j if the probability $Y = j$, given the response variable $X = x_i$ is the biggest. The Bayes Classifier has the lowest possible test error rate, named the Bayes Error Rate. One issue with the real-world problem is we don't know the conditional distribution of Y, given X, and that makes the calculation of the classifier improbable. The Bayesian Networks are statistical graphical models that attach a set of variables with their conditional probabilities. It takes an event that took place and predicts the likelihood any of the known causes was a contributing factor. It is important to have prior knowledge of data, especially when data

is scarce or expensive to produce. A Bayesian Network for a set of variables $X = X_1; X_2; ::X_n$ has a network structure, that encodes a set of conditional independence assertions about X and a set of local probability distributions associated with it.

From the article, the next figure represents a Bayesian Network, where X_1 is a variable conditionally independent from X_2 given X_3, if $P(X_1 j X_2 X_3) = P(X_1 j X_3)$ for all possible values of X_1, X_2, X_3.



The structure can be known or unknown. For Bayes Networks that are structured, finding the parameters in the Conditional Probability Table is done by estimating a locally exponential number of parameters from the data provided. If the structure is unknown and we have n features, we may have to choose more than n exponential variable orderings to find the best one. To simplify the process, we introduce a scoring function to evaluate the model with respect to the training data; the function is added to the search method to find the networks with high scores (we can observe the learning task is a combinatorial optimization problem).

We also can construct Bayes Networks using Chi-squared and mutual information tests to find the conditional independence relationships and use them to build the Bayes Network. The Chi-squared algorithm is preferred for large networks to the scoring algorithm, that has lower prediction accuracy in this case. Also, the Chi-squared algorithm is used for sparse networks with more efficiency.

When the structure is unknown and there is missing data, Friedman and Koller (2003) presented how to efficiently compute a sum over the exponential number of networks, that are consistent with a fixed order over networks. Comparing to Decision trees, where the decisions are straightforward evaluations based on the frequency distribution of likely events, Bayesian Networks take into consideration the "evidence" that a certain event occurred, and calculate its probability distribution.

7 Instance-based Learning

Instance-based learning algorithms are called lazy-learning. These algorithms require less training computation time than other non-lazy algorithms (e.g. decision trees).

One of the most used algorithms is the nearest neighbors algorithm. The K-Nearest Neighbors (KNN) is a non-parametric algorithm, that classifies an observation to the class with highest estimated probability. We have a test observation x_0, an integer predetermined K, then the KNN algorithm finds the K points in the training data that are closest to x_0, named M, then it estimates the conditional probability for class j as the fraction of point in M with the response value equal j:

$$\text{Probability}(Y = jjX = x_0) = \frac{1}{K} \text{å } \mathbb{1}(y_i = j)$$

Where $\mathbb{1}$ is the indicator function. To calculate the minimum distance from the new observation to the training sample, to determine the KNN algorithm different formula can be used, the most significant are Euclidean, Minkowsky, Manhattan, Chebyshev, Camberra, Kendall's Rank Correlation.

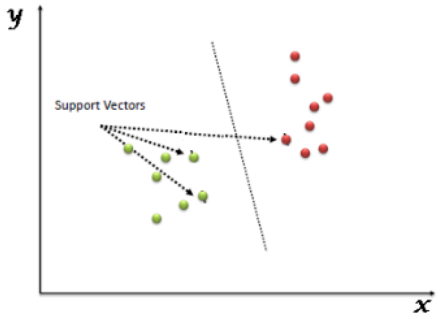
Minkowsky: $D(x,y)=\left(\sum_{i=1}^{m}\lvert x_i - y_i\rvert^r\right)^{1/r}$
Manhattan: $D(x,y)=\sum_{i=1}^{m}\lvert x_i - y_i\rvert$
Chebychev: $D(x,y)=\max_{i=1}^{m}\lvert x_i - y_i\rvert$
Euclidean: $D(x,y)=\left(\sum_{i=1}^{m}\lvert x_i - y_i\rvert^2\right)^{1/2}$
Camberra: $D(x,y)=\sum_{i=1}^{m}\frac{\lvert x_i - y_i\rvert}{\lvert x_i + y_i\rvert}$
Kendall's Rank Correlation: $D(x,y)=1-\frac{2}{m(m-1)}\sum_{i=j}^{m}\sum_{j=1}^{i-1} sign(x_i - x_j)sign(y_i - y_j)$

KNN classifier unfortunately has its drawbacks, it does tell us which predictors are important, we must pay a computational cost and

storage cost. We carefully select K the number of neighbors. A small number for K provides a flexible fit, with low bias and high variance. A large value of K will remove the noise in the locality of the query instance, and will result in a smoother decision boundary, but will have a high bias and a low variance. To improve its application, we need to determine first the input features, through model selection, then this process will improve the computational time and classification accuracy, and to find the best distance metric for the given data set. KNN is easy to understand, to implement, to work with multiclass data sets, also is a stable algorithm.

8 Support Vector Machines

Support Vector Machine (SVM) is an algorithm used for classification and regression analysis. SVM has a significant accuracy and needs less computational power. It works on smaller data sets. The SVM is an extension of the support vector classifier (this classifier cannot be use for most data sets, because it requires that classes should be separable by linear boundaries). For the support vector classifier, we plot each observation as a point in n dimensional space, where n is the number of variables, with each variable, being a coordinate. We find a hyperplane to divide the two classes. The process is shown in the figure below (Ray 2017).



Choosing the hyperplane, that separates the training observation into two classes is a optimization problem.

$$\begin{aligned} \underset{b_0,b_1,\ldots,b_p,e_1,\ldots,e_n}{\text{maximize}} \quad & M \\ \text{s.t.} \quad & \sum_{j=1}^{p} b_j^2 = 1 \\ y_i(b_0 + b_1 x_{i1} + b_2 x_{i2} + \cdots + b_p x_{ip}) \quad & M(1 \quad e_i) \\ e_i \quad & 0 \\ \sum_{i=1}^{n} e_i \quad & C \end{aligned}$$

Where C is a nonnegative tuning parameter, M is the width of the margin, e_i are slack variables. After we solve the equations, we find classify the test observation x, based on what side of the hyperplane is situated (James 2017).

If the space is not linearly separable, SVM uses a mapping to make the feature space bigger, making the separation easier in that space. The mapping is done by dot products of pairs of input data vectors, defined by a kernel function. SVM doesn't need the actual vectors, it uses the product between them, so it reduces the intensive calculations of adding a new space dimension. Common types of kernels used to separate non-linear data are polynomial kernels, radial basis kernels, and linear kernels (which are the same as support vector classifiers). SVM is effective in high-dimensionality data set, efficient in using the memory resource and it is versatile, used in many non-linear problems.

9 Comparison of Algorithms

A wide range of supervised learning algorithms are available, each having its strengths and weakness. The fact that no algorithm works best for all data set or all circumstances is called in statistics "No free lunch theorem". For accuracy in general, SVM, Neural Networks work better, followed by Decision trees and Rule learners. For speed of classification KNN is the poorest choice, KNN is a slow algorithm, to compute when we have a lot of data. A better selection would Decision trees, Neural Networks, Naïve Bayes, SVM and Rule-learners. With respect to redundancy in variables, a solution would be regularization of data. SVM handles better, the redundancy than the other methods. With respect of dimensionality of the input variables,

high number of predictors typically will make the model have low variance and high bias. The best algorithm for dealing with high number of attributes and the number of instances is Naïve Bayes and KNN. Another issue is how much noise has the response variable. One example of too much noise is overfitting the data. Too handle a lot of noise, it is preferred an algorithm that has a higher bias and lower variance, such as Naïve Bayes. The worst choice would be KNN. Heterogeneity of data is best modeled by SVM, KNN and Neural Networks. The amount of data need it is another important issue, and it is related to the complexity of the classifier function. A learning algorithm with high bias and low variance with a simple function can learn from a small amount of data. For a learning algorithm, with low bias and high variance, with a complex function we will need a very large amount of training data.

10 Conclusion

Machine learning technology is improving in efficiency and accuracy with the advances in the computer technology and mathematical theory. There are thousands of algorithms and hundreds are added every year. Machine learning has many applications in the object recognition field, in computer vision, pattern recognition, spam detection, bioinformatics, database marketing and many more fields. Every Machine Learning algorithm has three components. First is representation, we need to define a good design matrix X, from raw data, that we are going to use in the next step, optimization, where we find, hopefully, a convex function to help us solve the problem. The last step is evaluation, estimate how accurate our calculations are. Machine learning's newest developments are in multi-task and lifelong learning. In multi-task learning, the learner optimizes the performance across all of the n performed tasks through some shared knowledge. The lifelong learning uses the relevant knowledge gained in the past n 1 tasks to help learning for the n^{th} task.

References

[1] Frees, Edward W. *Regression Modelling with Actuarial and Financial Applications*. University Press, 2010.

[2] James, Gareth, et al. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2017.

[3] Le, James. "Decision Trees in R." DataCamp Community, 19 June 2018, https://www.datacamp.com/community/tutorials/decision-trees-R.

[4] Kotsiantis, S. B. "Supervised Machine Learning: A Review of Classification Techniques." *Informatica*, vol. 31-249-268, 16 July 2007, https://datajobs.com/data-science-repo/Supervised-Learning-[SB-Kotsiantis].pdf.

[5] Ray, Sunil, and Business Analytics and Intelligence. "Understanding Support Vector Machines Algorithm (along with Code)." Analytics Vidhya, 13 Sept. 2017, https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/.

Appendix A: Analysis of Maze Data Set

1 Introduction

This paper accompanies our Data Mining Project and presents an analysis of the male flowering time, using Ridge regression. The data set was provided by (Buckler 2009). A short description of the data is given by Buckler:

> There were 25 crosses (also called families or populations), each with about 200 recombinant inbred lines (RILs). A number of phenotypes were measured along with marker data. In the data set above, there are 4981 observations on 7393 variables with some missing values. The phenotype in this data set is time to male flowering (DtoA), and there are 7389 independent variables representing the SNP markers.

The variables are as follows: pop (integers 1 to 25 labeling families), $m1$ $m7389$ (independent variables=gene scores from the SNP markers: mostly $0; 1; 2$ but some intermediate values due to imputation), entry (order), geno_code (marker reference: information about inbred line ID and family ID, combining Pop and Entry variables), and DtoA (dependent variable = days to anthesis male flowering time).

2 Analysis of Data

First inspection of the data shows that out of 4981 observations, some have missing values. After removing, the entries with missing values, the data set contains 4494 observations and 7393 variables. Performing linear regression to model the dependent variable DtoA, using all 7389 independent variables, the result reveled over 3000 dependencies. As a shrinkage method, it will be used Ridge regression.

Ridge Regression is a procedure, for analyzing multiple regression data that suffer from multicollinearity. If multicollinearity occurs, least squares estimates are unbiased, but their variances are large.

Ridge regression adds bias, but reduces the standard error. The Ridge regression estimate is defined as the value of b that minimizes

$$\text{å}(y_i - x_i^T b)^2 + l \text{ å} b_j^2$$

Where l is a tuning parameter, that is determined separately. l å b_j^2 is called a shrinking parameter, is small when the b_j are close to zero, having the effect of shrinking the estimates of b_j towards zero. The values of l used range from $l = 10^{10}$ to $l = 10^{-3}$, covering a full range of scenarios from the null model, containing only the intercept, to least square fit. Associated with each value of l is a vector of Ridge regression coefficients. Here, there is a matrix with 7390 rows and 100 columns. The coefficient estimates are smaller, when a large value of l is used, compared with a small value of l. For $l = 3678.38$, here are some coefficients, along with their l_2 norm.

(Intercept)	m1	m2	m3	m4
7.660904e+01	4.909929e-05	4.912605e-05	4.915264e-05	4.943931e-05
m5	m6	m7	m8	m9
4.976872e-05	5.009670e-05	5.042324e-05	5.074831e-05	5.107188e-05
m10	m11	m12	m13	m14
5.062287e-05	4.976912e-05	4.890227e-05	4.927382e-05	4.902819e-05
m15	m16	m17	m18	m19
4.876963e-05	4.851941e-05	4.867536e-05	4.881640e-05	4.903376e-05
m20	m21	m22	m23	m24
4.920920e-05	4.758506e-05	4.049125e-05	3.809171e-05	3.560681e-05
m25	m26	m27	m28	m29
3.986098e-05	4.449768e-05	4.270121e-05	4.029179e-05	4.304214e-05

Table1. Coefficients estimates.

For $l = 178.865$, here are some coefficients, along with their l_2 norm.

(Intercept)	m1	m2	m3	m4
7.489471e+01	4.202733e-04	4.191627e-04	4.180650e-04	4.221605e-04
m5	m6	m7	m8	m9
4.271092e-04	4.320293e-04	4.369206e-04	4.417826e-04	4.466150e-04
m10	m11	m12	m13	m14
4.349927e-04	4.149614e-04	3.949324e-04	3.899328e-04	3.809341e-04
m15	m16	m17	m18	m19
3.718894e-04	3.633293e-04	3.650919e-04	3.668332e-04	3.719934e-04

Table2. Coefficients estimates.

We can predict Ridge regression coefficients for a new value of l, $l = 50$. In the next table, there are some of the predicted values.

(Intercept)	m1	m2	m3	m4	m5
7.460195e+01	1.200044e-03	1.191479e-03	1.183212e-03	1.193502e-03	1.206838e-03
m6	m7	m8	m9	m10	m11
1.220045e-03	1.233091e-03	1.245951e-03	1.258604e-03	1.213301e-03	1.138743e-03
m12	m13	m14	m15	m16	m17
1.064683e-03	1.035995e-03	9.964473e-04	9.570400e-04	9.199156e-04	9.247490e-04

Table3. Predicted coefficients estimates.

Using ten-fold cross-validation, with a random choice of cross-validation fold results in $l = 161{;}217$ as the value of l with the smallest cross-validation error. The MSE, for this l value is 12:09. The model is refitted on the full data set, using $l = 12{:}09$ and we predict the coefficients estimates (some are presented in Table 4).

[1] 7.490403e+01	4.192769e-04	4.213689e-04	4.234036e-04	4.311220e-04
[6] 4.397500e-04	4.483176e-04	4.568262e-04	4.652767e-04	4.736705e-04
[11] 4.637472e-04	4.443624e-04	4.248209e-04	4.208510e-04	4.129604e-04
[16] 4.049266e-04	3.973615e-04	4.016071e-04	4.058297e-04	4.140216e-04
[21] 4.212596e-04	3.932209e-04	2.437833e-04	1.999513e-04	1.564693e-04
[26] 2.513572e-04	3.276888e-04	2.888426e-04	2.279865e-04	2.466363e-04
[31] 3.999435e-04	4.646708e-04	4.184872e-04	2.419267e-04	1.369482e-04

Table4. Predicted coefficients estimates.

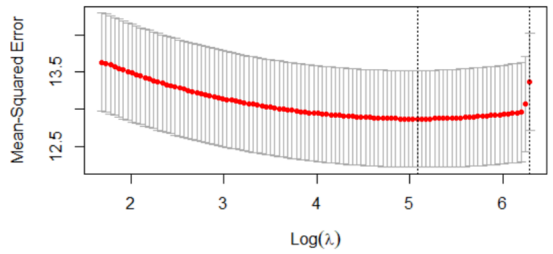


Figure1.MSE Ridge regression

3 Conclusion

Ridge regression is doing a great job analyzing the data and produces a $MSE = 12{:}09$. It shrinks the data set and fits it better than, the full fit linear regression.

4 References

[1] Buckler et al. (2009), "The Genetic Architecture of Maize Flowering Time," Science 325, 714-718

[2] Frees, Edward W. Regression Modeling with Actuarial and Financial Applications. Cambridge University Press, 2010.

[3] James, Gareth, et al. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2017.

Appendix B: Code and Results

```
> dim(x)
[1] 4494 7389
> y=Maize$DtoA
> grid=10
> grid=10^seq(10,-3, length=100)
> ridge.mod=glmnet(x,y, alpha=0, lambda=grid)
> dim(coef(ridge.mod))
[1] 7390  100
> ridge.mod$lambda[50]
[1] 3678.38
> coef(ridge.mod)[,50]
  (Intercept)            m1            m2            m3            m4
 7.660904e+01  4.909929e-05  4.912605e-05  4.915264e-05  4.943931e-05
           m5            m6            m7            m8            m9
 4.976872e-05  5.009670e-05  5.042324e-05  5.074831e-05  5.107188e-05
          m10           m11           m12           m13           m14
 5.062287e-05  4.976912e-05  4.890227e-05  4.927382e-05  4.902819e-05
          m15           m16           m17           m18           m19
 4.876963e-05  4.851941e-05  4.867536e-05  4.881640e-05  4.903376e-05
          m20           m21           m22           m23           m24
 4.920920e-05  4.758506e-05  4.049125e-05  3.809171e-05  3.560681e-05
          m25           m26           m27           m28           m29
 3.986098e-05  4.449768e-05  4.270121e-05  4.029179e-05  4.304214e-05
          m30           m31           m32           m33           m34
 5.236374e-05  5.525719e-05  5.377942e-05  4.692252e-05  4.148750e-05
          m35           m36           m37           m38           m39
 3.769873e-05  2.910107e-05  2.968898e-05  3.793132e-05  3.916577e-05
          m40           m41           m42           m43           m44
 4.038226e-05  4.143843e-05  3.734478e-05  3.973307e-05  3.990233e-05
          m45           m46           m47           m48           m49
 4.640031e-05  5.387568e-05  5.188967e-05  5.573901e-05  5.009278e-05
          m50           m51           m52           m53           m54
 4.336769e-05  3.988182e-05  4.166083e-05  4.853179e-05  5.016383e-05
 > sqrt(sum(coef(ridge.mod)[-1,50]^2))
 [1] 0.01242537

 >

> ridge.mod$lambda[60]
[1] 178.865
> coef(ridge.mod)[,60]
  (Intercept)            m1            m2            m3            m4
 7.489471e+01  4.202733e-04  4.191627e-04  4.180650e-04  4.221605e-04
           m5            m6            m7            m8            m9
 4.271092e-04  4.320293e-04  4.369206e-04  4.417826e-04  4.466150e-04
          m10           m11           m12           m13           m14
 4.349927e-04  4.149614e-04  3.949324e-04  3.899328e-04  3.809341e-04
          m15           m16           m17           m18           m19
 3.718894e-04  3.633293e-04  3.650919e-04  3.668332e-04  3.719934e-04
sqrt(sum(coef(ridge.mod)[-1,60]^2))
[1] 0.08204422
```