Tina Han
Classification and Regression Algorithms

In supervised learning problems, we use data that has and input and output, and give this data to an algorithm with the goal of creating a model that can either predict an outcome or classify a new data point. Given a dataset of N input output pairs, $\{(\mathbf{x}_i, y_i)\}$, where x_i is the ith input vector and y_i is the ith output, we split the data into testing and training sets. We give the algorithm the training set to create a model, and the test set to estimate how correct we expect the model to be. Using the error estimation, and depending on the algorithm, we may go back to training and tweak parameters then rerun the algorithm over the training data to find the best possible model. We will use the R^2 score for regression, and accuracy score for classification to help us decide which model and model parameters are best. Our goal is to find a model that generalizes well to data by not underfitting (a model that is too simple) or overfitting (a model that is too complex).

There are two types of supervised learning problems we approach here: regression and classification. Regression is used for prediction, where the outputs are real numbers. In our experiment, we want to predict the quality (as a numeric representation) of wine based on various factors, such as acidity and alcohol content. For our purposes, we will be considering four regression algorithms: K nearest neighbors (KNN), ordinary least squares regression (OLS), ridge regression, and lasso regression. Classification is used to classify a data point in a discrete category. For example, in our experiment, we want to classify an email as spam or not spam based on a variety of inputs. In our experiments we will use three different algorithms: K nearest neighbors (KNN), logistic regression, and linear support vector machines (SVM).

For our regression experiment we used the wine quality data set which has the inputs: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol; and one output variable: quality (score between 0 and 10). The training data had 4,872 observations and testing had 1,625. The first algorithm we used was k nearest neighbors (KNN) for regression. The KNN algorithm memorizes the training data, and for predictions the algorithm finds the nearest training point(s) to the new data point, then averages the outputs of those nearest neighbors. The algorithm has one parameter: $k =$ the number of neighbors the algorithm uses to average to make a prediction on a new data point. Additionally, the user must also determine a distance metric for the algorithm to use. In our case, we used the default distance metric minkowski. To evaluate the models we used the R^2 score, or coefficient of determination defined by

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{1}$$

where the values of the observed data are $y_1, ..., y_n$, and the predicted values are $\hat{y}_1, ..., \hat{y}_n$ and

$$SS_{res} = \sum_{i=1}^{n} \left(y_i - \frac{1}{n} \sum_{i=1}^{n} y_i \right)^2$$
$$SS_{tot} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

The R^2 is a value between 0 and 1 where 1 is perfect prediction and 0 is a constant model that predicts the mean of the training set.

We built a KNN regressor model for $k = 1, 2, ..., 20$ and compared the R^2 score for the training and testing data. Figure 1 presents the information visually. We can see that at $k = 1$ the training data is predicted perfectly (with and R^2 of 1) and the testing data is predicted poorly (an R^2 close to 0). This is an example of overfitting. The model with $k = 1$ is overly complex, and there is a large discrepancy between the training and testing R^2 scores with the training perfectly predicted and the testing poorly predicted. Looking at figure 1 again, we can see that as k increases the model becomes more simple, and the training and testing R^2 scores both tend towards 0.2. There are no obvious points of underfitting, but the testing R^2 score begins to drop at $k = 19$, so more obvious signs of underfitting may occur at larger k. Since the testing a training R^2 scores are close and the testing R^2 score is greatest at $k = 19$ we would pick this to be the best parameter for prediction with the KNN regressor algorithm.

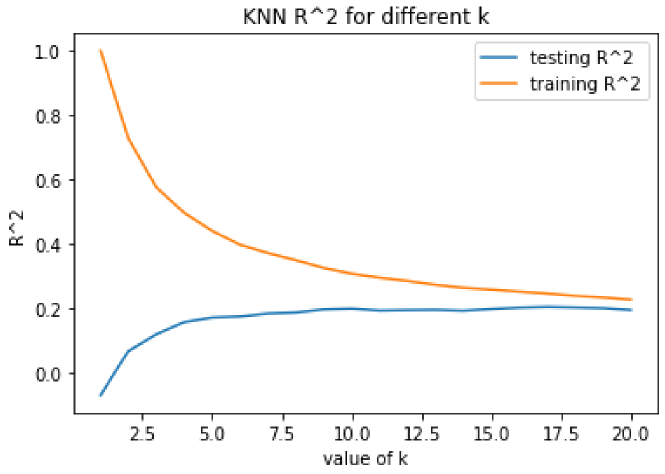

Figure 1: KNN regression on wine quality data

The next algorithm we used was ordinary least squares regression (OLS). OLS is the simplest method for linear regression and the model produced by this algorithm is of the form

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \cdots + w[p] * x[p] + b \tag{2}$$

where the parameters $w[i]$ and b are found by minimizing the mean squared error between the predictions, $\hat{y}$ and target values, y. There are no parameters to adjust, making the algorithm relatively simple but there is no way of controlling the model complexity. Using the OLS algorithm, we have the model

$$\begin{aligned}\hat{y} = 0.0656 * x[0] - 1.318 * x[1] - 0.128x[2] + 0.0413 * x[3] - 0.163 * x[4] + 0.006 * x[5]\\ - 0.002 * x[6] - 0.486 * x[7] + 0.401 * x[8] + 0.677 * x[9] + 0.273 * x[10] + 49.523\end{aligned}$$

which gives a training set R^2 score of 0.28 and test set R^2 score of 0.34. The R^2 scores are relatively low, and the discrepancy between the test and training R^2 scores may indicate underfitting since the model predicts better on test data than training data.

Ridge and lasso regression are both linear regression models (equation (2)), but are both methods of regularization, meaning there are imposed restrictions on the model to avoid overfitting. The goal of both algorithms is to push the magnitude of the coefficients as close to zero as possible (and in the case of lasso some coefficients are exactly zero resulting in automatic feature selection). The difference between ridge and lasso regression is in which norm of the coefficients is penalized by tuning a penalty parameter α. In ridge regression, the L2 norm of the coefficients is penalized as follows in equation (3)

$$J = \frac{1}{2N}\sum_{i=1}^{N}(h_w(x^{(i)}) - y^{(i)})^2 + \alpha\|w\|^2 \tag{3}$$

Whereas lasso regression imposes restrictions on the L1 norm as follows in equation (4)

$$J = \frac{1}{2N}\sum_{i=1}^{N}(h_w(x^{(i)}) - y^{(i)})^2 + \alpha\sum_{j=0}^{p}|w_j| \tag{4}$$

The default α is 1 but needs to be adjusted by the user for the data. A larger α pushes more coefficients towards zero, decreasing the test set performance with the goal of making the model more generalized.

For ridge regression we varied α between 0.1 and 10 and created a plot of the R^2 scores of the training and testing data against the different α. This is shown in figure 2. There is a rather large discrepancy between the training and testing R^2 score despite the different α. Thus, the based on the best R^2 scores, we would choose $\alpha = 0.1$. There is no obvious sign of over or underfitting, just a generally bad fit for the data.

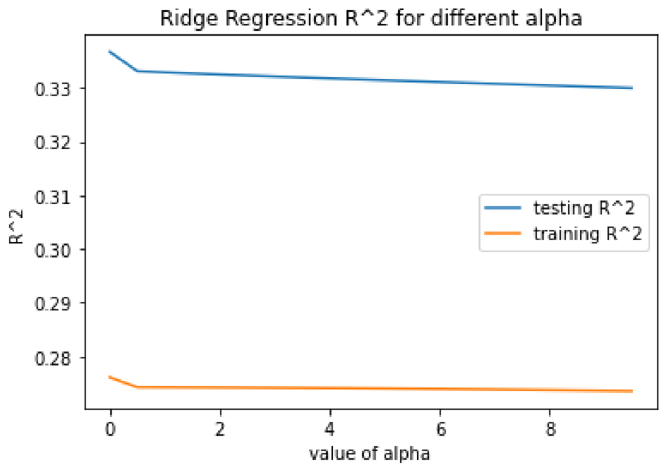

Figure 2: Ridge regression on wine quality data

For lasso regression we varied α between 0.1 and 1 and plotted the R^2 scores of the training and testing data as shown in figure 3. Lasso performed similarly to ridge regression in terms of the R^2 scores, but overall the testing and training R^2 scores were closer together. The best R^2 values were around $\alpha = 0.1$ (which selected 4 features to use, i.e. only 4 coefficients were nonzero), but similar to the other regression methods, the model overall appears to be a poor fit for the data.

Despite choosing the best parameters all four of our models have testing and training R^2 scores around 0.2-0.3, which is not very good. A summary of the scores is presented in table 1. No model stands out above the others, but if we had to choose one as is, the OLS method has the largest R^2 values. With R^2 values this poor, to find a good model, we would want to go back to the data and do some preprocessing like scaling the data, removing insignificant features, or dimensionality reduction with a technique like PCA.

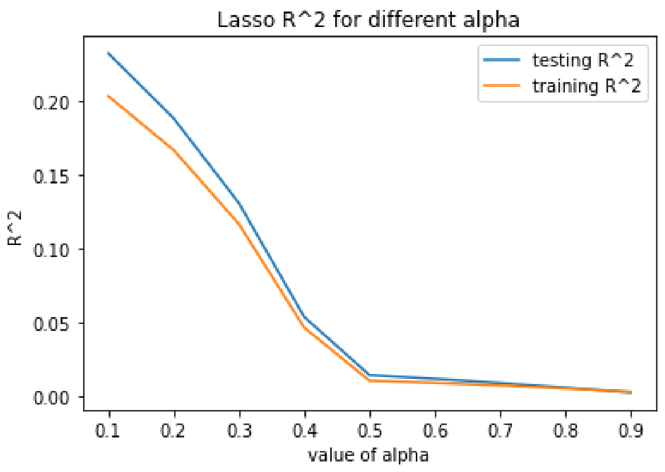

Figure 3: Lasso regression on wine quality data

	KNN ($k = 19$)	OLS	ridge ($\alpha = 0.1$)	lasso ($\alpha = 0.1$)
training R^2	0.234	0.28	0.274	0.203
testing R^2	0.201	0.34	0.333	0.233

Table 1: Best R^2 scores for each regression model

For classification, we used a spam email data set with 48 attributes and a class label of spam (1) or not spam (0). Most of the input features identify frequently used characters or words, some other attributes determine the length of sequences of consecutive capital letters. The first classification algorithm we used was KNN for classification. Like KNN for regression, it stores the training data and uses the k nearest neighbors to give a label to a new data point, but instead of averaging there is a majority vote to classify. Figure 4 shows the testing and training accuracy for different values of k. We used $k = 1, 2, ..., 20$ to train different models. We can see in figure 4 that for $k < 5$ we have an over complicated model which is overfit to the training data. For $5 \leq k \leq 11$ we can see we get a decent fit with maybe a little overfitting, but the accuracy scores for the testing and training are closer and quite high. For $k > 11$ we can see the training accuracy decline and have a case of underfitting; a model that is too simple. Here we would pick $k = 5$ as the best parameter since it gives the best overall accuracy.

Logistic regression is another classification algorithm which labels a data point x using a logistic function to compute the log odds of an output based on an input. Explicitly the formula is given in (5).

$$\hat{y} = h_w(x) = \sigma(w^\top x) = \frac{1}{1 + e^{-w^\top x}} \tag{5}$$

So for $y = 1$ we want $h_w(x)$ to be close to 1, corresponding to a large $w^\top x$, and for $y = 0$ we want $h_w(x)$ to be close to 0, corresponding to a small $w^\top x$. Logistic regression finds the decision boundary by minimizing the cost function

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} -(y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})) \tag{6}$$

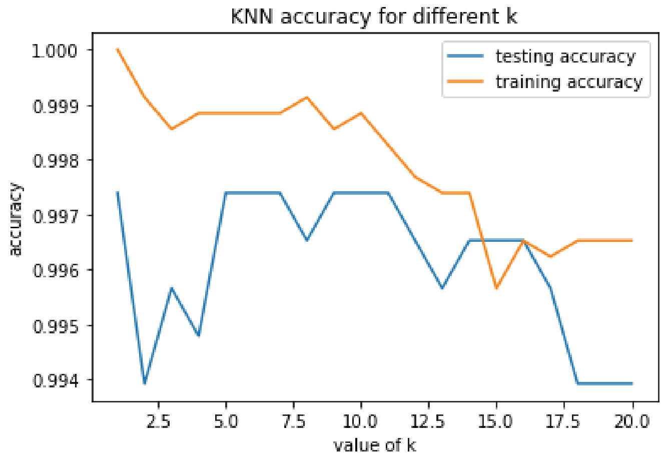

Figure 4: KNN classifier with spam email data

Like logistic regression, linear support vector machines (SVM), seek to find a boundary between classes but by maximizing a margin between classes. We assume that larger margins between classes would generalize better than small margins which can overfit the data. If classes are not linearly separable, we may need to introduce a slack variable which allows for the algorithm to converge by letting some of the training data be on the ”wrong side” of the boundary.

Logistic regression and Linear SVC apply regularization, and in our case this corresponds to the parameter C. For larger C, there is less regularization, and fits the model well to the training data. A lower C balances the fitting error and complexity. In figures 5 and 6 we vary C and plot it against the training and testing accuracy for logistic regression and linear SVM, respectively.

Looking at figure 5, we see that at $C = 8$ both the training and testing accuracy is the greatest for logistic regression. We also see that the training and testing accuracy remain close together despite varying values of C, making it difficult to determine where we may have underfitting and overfitting.

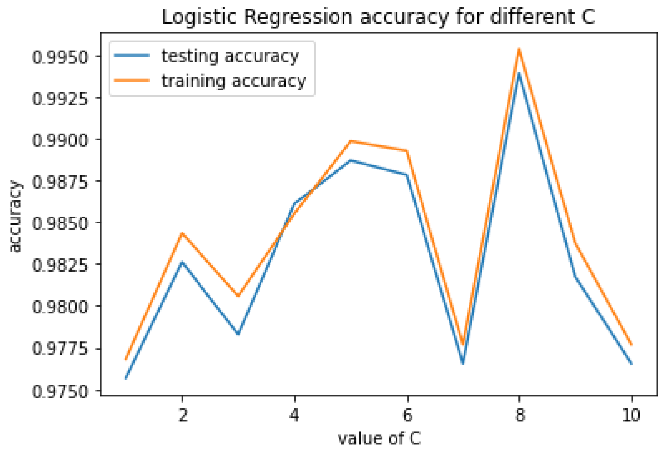

Figure 5: Logistic Regression with spam email data

Similarly, figure 6 illustrates the testing and training accuracy based on choice of C for the linear SVM models. Like the logistic regression models, the accuracy for testing and training stay close together despite the varying C values. We also note that linear SVM changed drastically each time we ran the algorithm indicating that perhaps this data set is not large enough to be used well with this algorithm.

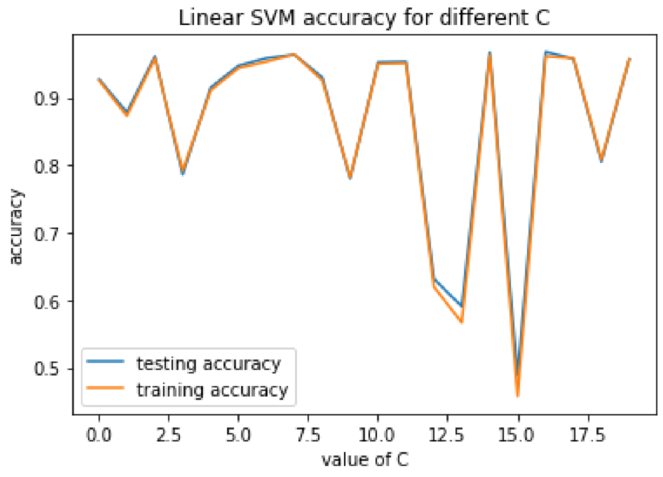

Figure 6: Linear SVM with spam email data

	KNN ($k = 5$)	Logistic Regression ($C = 8$)	Linear SVM ($C = 7$)
training accuracy	0.999	0.995	0.964
testing accuracy	0.997	0.994	0.964

Table 2: Best accuracy scores for each classification model

Based on table 2, we see that KNN with $k = 5$ gives the best overall accuracy, so for classification, this is the model we would want to use for this data set. However, all classifiers perform well, so one might consider other factors such as computational complexity when choosing a model. For example, while linear SVM performed well, it took the computer the most time to run.