Notes

June 23, 2018

1 Applied Predictive Modeling

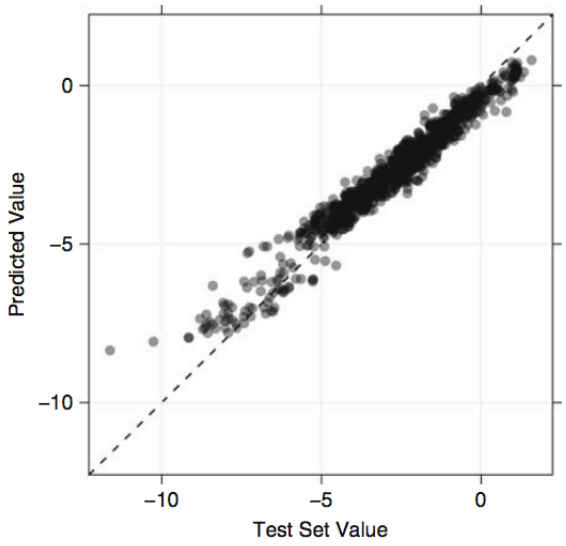
---

2 Measuring Performance in Regression Models

2.1 Quantitative Measures of Performance.

RMSE: When the outcome is a number, the most common method for characterizing a model's predictive capabilities is to use the *root mean squared error (RMSE)*. This metric is a function of the model residuals, which are the observed values minus the model predictions. The mean squared error (MSE) is calculated by squaring the residuals, summing them and dividing by the number of samples. The RMSE is then calculated by taking the square root of the MSE so that it is in the same units as the original data. The value is usually interpreted as either how far (on average) the residuals are from zero or as the average distance between the observed values and the model predictions.

R^2 : Another common metric is the coefficient of determination, commonly written as R^2. This value can be interpreted as the proportion of the variance in the data that can be explained by the model. Thus, an R^2 value of 0.75 implies that the model can explain three-quarters of the variation in the outcome. There are multiple formulas for calculating this quantity, although the simplest version finds the correlation coefficient between the observed and predicted values (usually denoted by R) and squares it.



The figure above plots the observed and predicted outcomes where the R^2 is moderate (51%), but predictions are not uniformly accurate. The diagonal grey line indicates where the observed and predicted values would be equal

While this is an easily, interpretable statistic, the practitioner must remember that R^2 is a measure of correlation, and not accuracy. The figure shows an example where the R^2 between the observed and predicted values is high (51%), but the model has a tendency to overpredict low values and underpredict high ones. This phenomenon can be common to some of the tree-based-regression models. Depending on the content, this systematic bias in the predictions may be acceptable if the model otherwise works well.

It is also important to realize that R^2 is dependent on the variation in the outcome. Using the interpretation that this statistic measures the proportion of variance explained by the model, one must remember that the denominator of that proportion is calculated using the sample variance of the outcome. For example, suppose a test set outcome has a variance of 4.2. If the RMSE of a predicted model were 1, the R^2 would be roughly 76%. If we had another test set with the same RMSE, but the test outcomes were less variable, the results would look worse. For example, if the test set variance were 3, the R^2 would be 67%.

Practically speaking, this dependence on the outcome variance can also have a drastic effect on how the model is viewed. For example, suppose we were building a model to predict the sale price of houses using predictors such as a house characteristic, as well as a lot size and location. If the range of the houses in the test set were large, say from $60K$ to $2M$, the variance of the sale price would also be very large. One might view a model with a 90%R^2 positively, but the RMSE may be in tens of thousands of dollars - poor predictive accuracy for anyone selling a moderately priced property.

In some cases, the goal of the model is to simply rank new samples. As previously dicussed, pharmaceutical scientists may screen a large numbers of compounds for their activity in an effort to find "hits." The scientist will then follow up on the compounds predicted to be the biologically active. Here, the focus is on the ranking ability of the model rather than the predictive accuracy. In this situtation, determining the *rank correlation* between the observed and predicted values might be more appropriate metric. The rank correlation takes the ranks of the observed outcome values,

and evaluates how close they rank to the model predictions. To calculate this value the rank of the observed and predicted outcomes are obtained and the correlation coefficient between these ranks is calculated. This metric is commonly known as *Spearman's Rank Correlation*.

2.2 The Variance-Bias Trade-Off

The MSE can be decomposed into more specific pieces. Formally, the MSE of a *model is*

$$MSE = \frac{1}{n}\sum(y_i - \hat{y_i})$$

Where y_i is the outcome and $\hat{y_i}$ is the model prediction of that samples outcome. If we assume that the data points are statistically independent and that the residuals havve a theortical mean of zero and a constant variance of σ^2, then :

$$E[MSE] = \sigma + (\text{Model Bias})^2 + \text{Model Variance}$$

Where E is the expected value. The first part σ^2 is usually called "irreducible noise" and cannot be eliminated by modeling. The second term is the squared *bias* of the model. This reflects how close the functional form of the model can get to the true relationship between the predictors and the outcomes.

As a general rule, the more complex the model is, the higher the chances that the model will over-fit. Conversly, for models that are too simple, it is often likely to underfit, as they are not flexible enough to model the true relationship (thus high bias).Also, highly correlated predictors can lead to *colinearity* issues and this can greatly increase the model variance.

---

3 Linear Regression and It's Cousin

In this chapter we will discuss several models, all of which are akin to linear regression in that each can directly or indirectly be written in the form

$$y_i = \beta_0 + \beta x_{i1} + \beta x_{i2} + ... + \beta x_{iP} + \epsilon_i$$

where y_i represents the numeric response for the ith sample, b_o represents the estimated intercept, b_j represents the estimated coefficient for the jth predictor, X_{ij} represents the value of the jth predictor for the ith sample, and e_i represents random error that cannot be explained by the model. When a model can be written in the form, we say that it is *linear in the parameters.* In addition to ordinary linear regression, these types of models include partial least squares and penalized models such as ridge regression, the lasso, and the elastic net.

Each of these models seek to find estimates of the parameters so that the sum of the squared errors or a function of the sum of the squared errors is minimized. Ordinary linear regression, at one extreme, finds parrameters estimates that have minimum bias, whereas ridge regression, the lasso, and the elastic net find estimates that have lower variance. The impact of this trade-off on the predictive ability of these models will be illustrated later on. A distinct advantage of these

models is that they are highly interpretable. For example, if the estimated coefficient of a predictor is 2.5, then a 1 unit increase in that predictor's value would, on average, increase the response by 2.5 units. Furthermore, relationships amond predictors can be further interpreted through the estimated coefficients. In understanding whether there is significance between-predictor correlations, one can look to employ principal component analysis (PCA) on the full set of transformed predictors, and the percent of variance accounted by for each componenet is determined. Profiling the variance accounded for by each component is commonly known as a scree plot. Notice that there is no one component acconting for more than 13% of the variance. Thus this indicates that the structure of the data is contained in a much smaller number of dimensions than the number of dimensions of the original space, this is often due to a large number of collinearities among predictors.

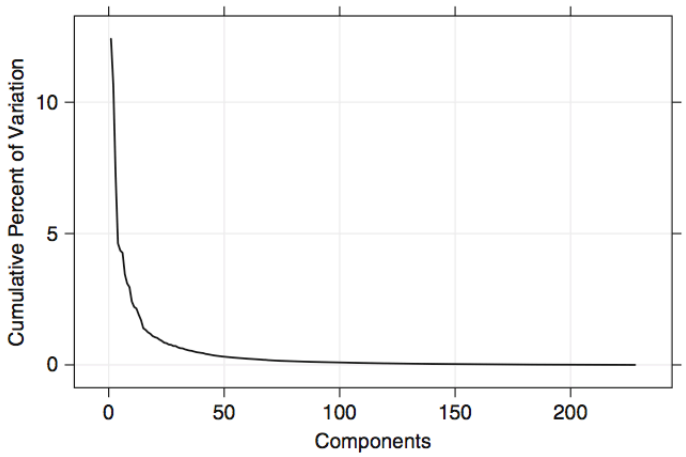


Fig. 6.4: A scree plot from a PCA analysis of the solubility predictors

3.1 Linear Regression.

The objective of ordinary least squares linear regression is to find the plane that minimizes the sum-of-squared errors (SSE) between the observed and predicted response:

$$SSE = \sum (y_i - \hat{y}_i)^2$$

where y_i is the outcome and $\hat{y}_i$ is the model prediction of that sample's outcome. Mathematically, the optimal plane can be shown to be

$$(X^T X)^{-1} X^T y,$$

Where X is the matrix of predictors and y is the response vector. The equation is also known as $\hat{\beta}$ in statistical texts and is a vector that contains the parameter estimates or coefficients for each predictor. This quantity is easy to compute, and the coefficients are directly interpretable. Making some minimal assumtpions about the distribution of the residuals, it is straightforward to show that the parameter estimates that minimize SSE are the ones that have the least bias of all possible parameter estimates. Hence these estimates minimize the bias component of the bias-variance trade-off.

The interpretability of coefficients makes it very attractive as a modeling tool. At the same time, the characteristics that make it interpretable also make it prone to potentially fatal flaws.

Notice that embedded in the equation is the term $(X^T X)^{-1}$, which is proportional to the covariance matrix of the predictors.

A unique inverse of this matrix exists when

1. no predictors can be determined from a combination of one or more of the other predictors
2. the number of samples is greater than the number of predictors

if the data falls under either of these conditions, then a unique set of regression coefficients does not exist.

However, a unique set of predicted values can still be obtained for data that falls under the first category by removing predictors that are collinear. By default, when fitting a linear model with R and collinearity exists among predictors 'R fits the largest identifiable model by removing variables in the reverse order of appearance in the model formuka'. The upshot of these facts is that linear regression can still be used for predictions when collinearity exists within the data. But since the regression coefficients to determine these predictions are not unique, we lose out ability to meaningfully interpret the coefficients.

When condition (2) is true for a data set, the practitioner can take several steps to attempt to build a regression model. As a first step we suggest using pre-processing techniques presented, to remove pairwise correlated predictors, which will reduce the number of overall predictors. However, this pre-processing step may not completely eliminate collinearity, since one or more of the predictors may be functions of *two* or more of the other predictors. To diagnose multicollinearity in the context of linear regression, the *variance inflation factor* can be used. This statistic is computed for each predictor and a function of the correlation between the selected predictor and all of the other predictors.

After pre-processing the data, if the number of predictors still outnumbers the number of observations, then we will need to take other measures to reduce the dimension of the predictor space. PCA pre-processing is one remedy.

3.2 Partial Least Squares

For many real-life data sets, predictors can be correlated and contain similar predictive information. If the correlation among predictors is high, then the ordinary least squares solution for mulitple linear regression will have high variability and will become unstable.

For other data sets, the number of predictors may be greater than the numner of observations. In this case, too, ordinary least squares in its usual form will be unable to find a unique set of regression coefficients that minize the SSE. A couple of common solutions to the regression problem under these conditions include pre-processing the predictors by either removing highly correlated predictors through techniques previously descibed or conducting PCA on the predictors. Removing highly correlated predictors ensure that pairwise correlation among predictors are below a pre-specified threshold. However, this process does not necessarily ensure that linear combinations of predictors are uncorrelated with other predictors. if this is the case, then the ordinary least squares solution will still be unstable. Therefore it is important to understand that the removal of highly correlated pairwise predictors may not guarantee a stable least squares

solution. Alternatively, using PCA for pre-processing guaratees that the resulting predictors, or combination thereof, will be uncorrelated. The trade-of in using PCA is that the new predictors are linear combinations of the original predictors, and thus, the practical understanding of the new predictors can become murky.

Pre-processing predictors via PCA prior to performing regression is known as principal component regression (PCR); this technique has been widely applied in the context of problems with inherently highly correlated predictors or problems with more predictors than observations. While this two-step regression approach (dimension reductio, then regression) has been successfully used to develop predictive models under these conditions , it can easily be misled.

Specifically, dimension reduction via PCA does not necessarily produce new predictors that explain the response. As an example of this scenario, consider the data which contains two predictors and one response using the direction of the maximal variability. The right-hand plot of this figure, however, illustrates that the first PCA direction contains no predictive information about the response.

As this simple example illustrates, PCA does not consider any aspect of the response when it selects its components. Instead, it simply chases the varaibility present throughout the predictor space. If that variability happens to be related to the response variability, then PCR has a good chance to identify a predictive relationship. If, however, the variability in the predictor space is not related to the variability of the response, then PCR can have difficulty identifying a predtice relationship when one might actually exist. Because of this inherent problem with PCR, we recommend using PLS when there there are correlated predictors and a linear regression-type solution is desired. PLS originated with Herman Wold's nonlinear iterative partial least squared (NIPALS) algorithm, which linearized models that were nonlinear in the parameters. Subsequently, Wold adapted the NIPALS method for the regression setting with correlated predictors and and called this adaptation "PLS".

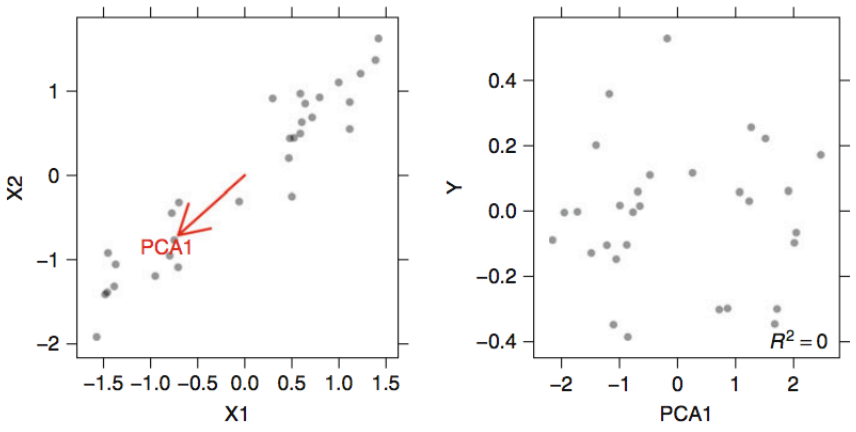


Fig. 6.8: An example of principal component regression for a simple data set with two predictors and one response. *Left*: A scatter plot of the two predictors shows the direction of the first principal component. *Right*: The first PCA direction contains no predictive information for the response

Briefly, the NIPALS algorithm iteratively seeks to find underlying, or latent, relationships among the predictors which are highly correlated with the reponse. For a univariate response,

each iteration of the algorithm assesses the relationship between the predictors (X) and response (y) and numerically summarizes this relationship with a vector of weights (w); this vector is also known as a direction. The predictor data are then orthogonally projected onto the direction to generate scores (t). The scores are then used to generate loading (p), which measure the correlation of the score vector to the original predictors. At the end of each iteration the predictors and the response are "deflated" by dubstracting the current estiamte of the predictor and response structure, respectively. The new deflated predictor and response information are then used to generate the next set of weights, scores, and loading. These quantities are sequentially stored in matrices **W**, **T** and **P**, respectively and are used for predicting new samples and computing predictor importance. A schematic of the PLS relationship between predictors and the response can be seen, and through the figure below. **Simply**, PLS finds linear combinations of the predictors. These linear combinations are commonly called *components* or latent variables.**While the PCA linear combination are chosen to maximally summarize predictor space variability, the PLS linear combination of predictors are chosen to maximally summarize the covariance with the response**. **This means that PLS finds components that maximally summarize the variation of the predictors while simultaneously requiring these components to have maximum correlation with the response**. PLS therefore strikes a compromise between the objective of predictor space dimension reduction and a predictive relatioship with the response. In other words, PLS can be viewed as a *supervised* dimension reduction procedure; PCR is an *unsupervised* procedure.

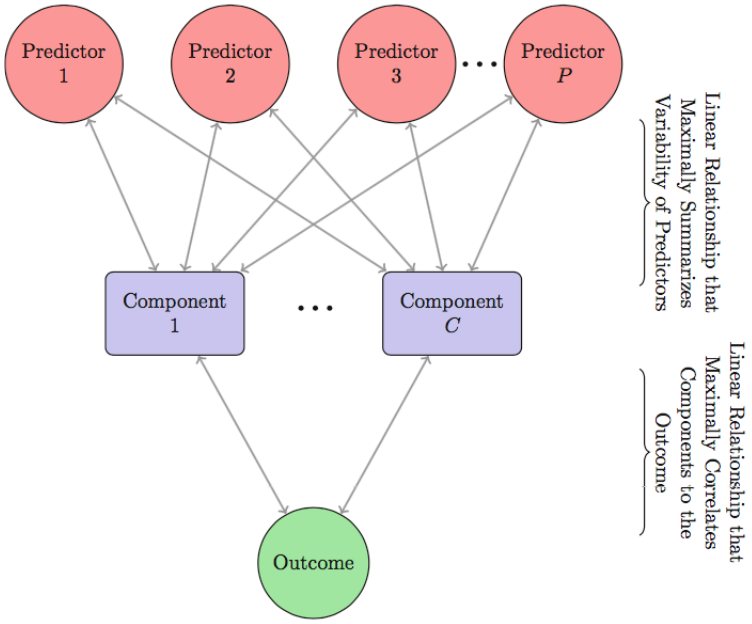
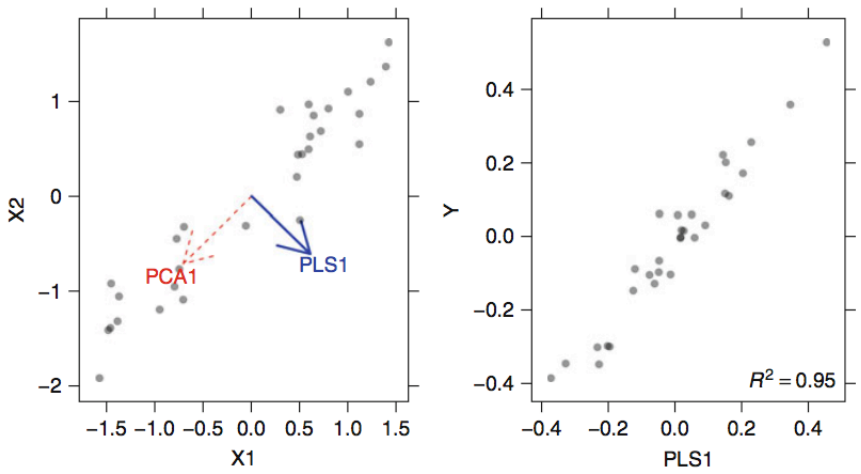


Fig. 6.9: A diagram depicting the structure of a PLS model. PLS finds components that simultaneously summarize variation of the predictors while being optimally correlated with the outcome

To better understand how PLS works and to relate it to PCR, we will revisit the data presented. This time we seek the first PLS component. The left-hand scatter plot, constrasts the first PLS direction with the first PCA direction. For this illustration the two directions was not related to maximal variation in the predictor space. Instead PLS, identified the optimal predictor space dimension reduction for the purpose of regression with the response.



Clearly this example is designed to show an important flaw with PCR. However, in most casts the predictability of PCR and PLS are similar. Based on our experience, the number of components retained via cross-validation using PCR is always equal to or greater than the number of comonents retained by PLS. This is due to the fact that dimensions retained by PLS have been chosen to be optimally related to the response, while those chosen with PCR are not.

Prior to performing PLS, the predictors should be centered and scaled, especially if the predictors are not scales of differing magnitude. As descrived above, PLS will seek directions of maximum variation while simultaneously considering correlation with the response. Even with the constraint of correlation with the response, it will be more naturally drawn towards predictors with large variation. Therefore, predictors should be adequately preprocessed prior to performing PLS.

Once the predictors have been preprocessed, the practitioner can model the response with PLS. PLS has one tuning parameter: the number of components to retain. Potential resampling techniques can be used to determine the optimal number of bins.

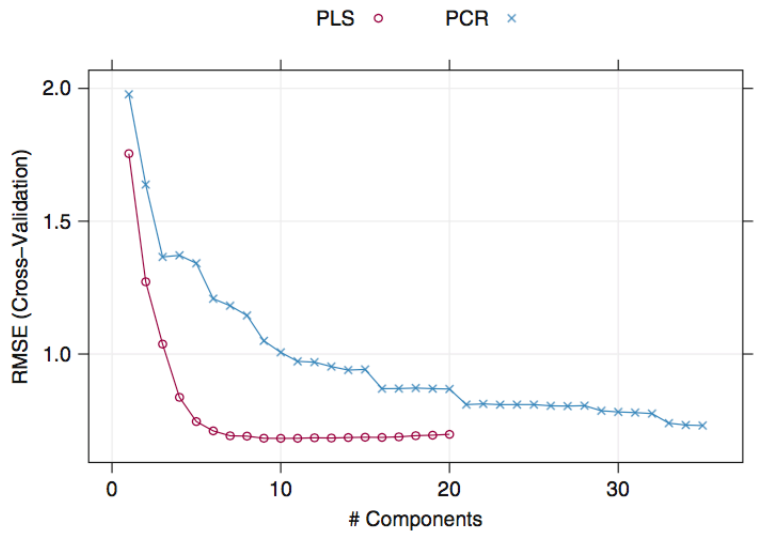


Fig. 6.11: Cross-validated RMSE by component for PLS and PCR. RMSE is minimized with ten PLS components and 35 PCR components

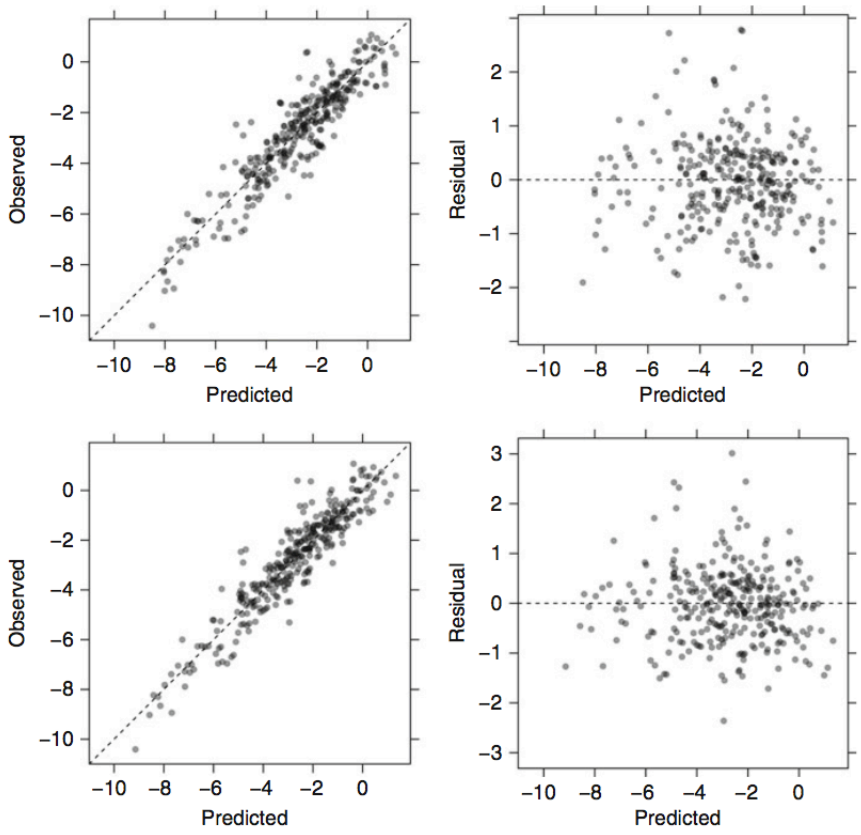


Fig. 6.13: *Left side*: Observed versus predicted values for the solubility test set for PCR (*upper*) and PLS (*lower*). *Right side*: Residuals versus the predicted values for PCR and PLS. The residuals appear to be randomly scattered about 0 with respect to the predicted values. Both methods have similar predictive ability, but PLS does so with far fewer components

3.3 Penalized Models

Under standard assumptions, the coefficients produce by ordinary least squares regression are unbiased, and, of all unbiased linear techniques, this model also has the lowest variance. However, given that the MSE is a combination of variance and bias, it is very possible to produce models with smaller MSEs by allowing the parameter estimates to be biased. it is common that a small increase in bias can produce a substantial drop in the variance and thus a smaller MSE than ordinary least squares regression coefficients. One consequence of a large correlation between the predictor variances is that the variance can become very large. Combatting collinearity by using biased models may result in regression models where the overall MSe is competitive. One method of creating biased regression models is to add a penalty to the sum of squared errors. Recall that original least squares regression found parameter estimates to minimize the sum of the squared errors:

$$\text{SSE} = \sum (y_i - \hat{y}_i)^2$$

When the model over-fits the data, or when there are issues with collinearity, the linear regression parameter estimates may become inflated. As such, we may want to control the magnitude of these estimates to reduce the SSE. Controlling (or *regularizing*) the parameter estimates can be accomplished by adding a penalty to the SSE if the estimates become large.

Ridge regression adds a penalty on the sum of the squared regression parameters:

$$\text{SSE}_{L2} = \sum (y_i - \hat{y}_i)^2 + \lambda \sum \beta^2$$

The L_2 signifies that a second-order penalty is being used on the parameter estimates. The effect of this penalty is that the parameter estimates are only allowed to become large if there is a proportional reduction in SSE. In effect, this method shrinks the estimates towards 0 as the λ penalty becomes large. By adding the penalty, we are making a trade-off between the model variance and bias. By sacrificing some bias, we can often reduce the variance enough to make the overall MSE lower than unbiased models.

For example, the figure below shows the *path* of the regression coefficients for the solubility data over different values of λ. Each line corresponds to a model parameter and the predictors were centered and scaled prior to this analysis so that their units are the same. When there is no penalty, many parameters have reasonable values, such as the predictor for the number of multiple bonds. However, some parameter estimates are abnormally large, such as the number of non-hydrogen atoms, and the number of non-hydrogen bonds. These large values are indicative of collinearity issues. As the penalty is increased, the parameter estimates move closer to 0 at different rates.

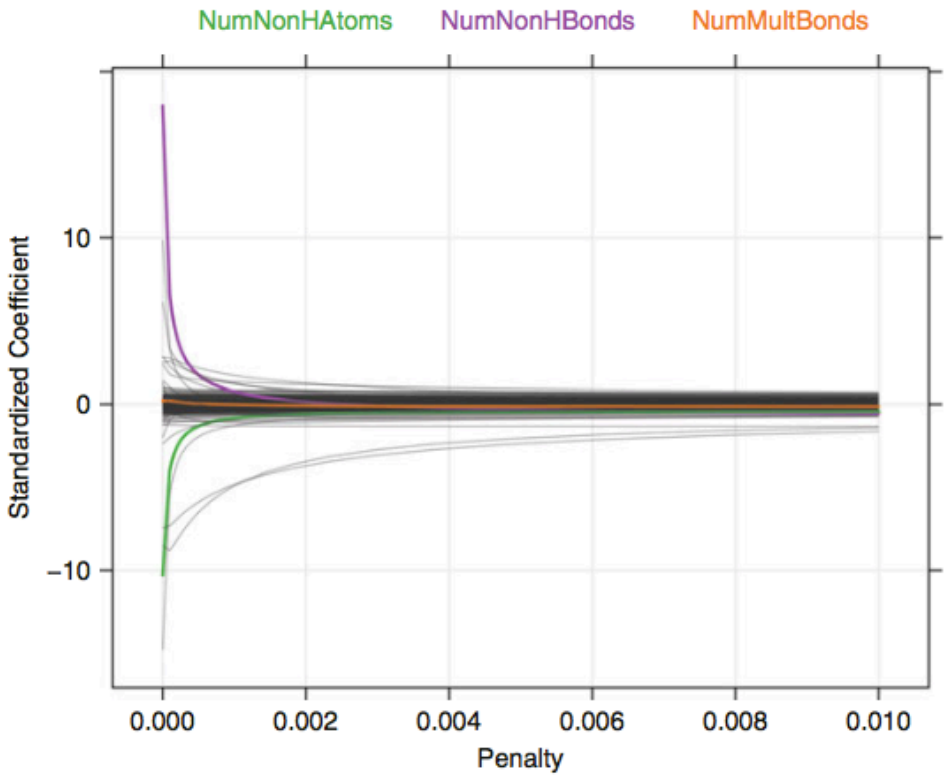


Fig. 6.15: The ridge-regression coefficient path

By the time the penalty has a value of $\lambda = 0.002$, these two predictors are much more well behaved although other coefficient values are still relatively large in magnitude. Using cross-validation, the penality value was optimized. Where there is no penalty, the errors are inflated. As the penalty increases beyond 0.0366, the bias becomes to large and the model starts to under-fit, resulting in an increase in MSE.

While ridge regression shrinks the parameter estimates towards 0, the model does not set the values to absolute 0 for any value of the penalty. Even though some parameter estimates become negligibly small, this model does not conduct *feature selection*.

A popular alternative to ridge regression is the *least absolute shrinkage and selection operator* mode, frequently called the *lasso*. This model uses a similar penalty to ridge regression:

$$\text{SSE}_{L1} = \sum(y_i - \hat{y}_i)^2 + \lambda \sum |\beta_j|$$

While this may seem like a small modification, the practical implications are significant. WHile the regression coefficients are still shrunk towards 0, the consequence of penalizing tha abosulate values is that some parameters are actually set to 0 for some values of λ. Thus the lasso yields models that simultaneously use regularization to improve the model and to conduct feature selection.

3.4 Multivariate Adaptive Regression Splines

Like neural networks and partial least squares, MARS uses surrogate features instead of the original predictors. However, whereas PLS and neural networks are based on linear combinations

of the predictors, MARS creates two contrasted versions of a predictor to enter the model. Also, surrogate features in MARS are usually a function of only one or two predictors at a time. The nature of the MARS features breaks predictors into two groups and models linear relationships between the predictor and the outcome in each group. Specifically, given a cut point for a predictor, two new features are "hinge" or hocket stick functions of the original. The "left-hand" feature has values of zero greater than the cut point, while the second feature is zero less than the cut point. The new features are added to a basic linear regression model to estimate the slopes and intercepts. In effect, this scheme creates a *piecewise linear model* where each new feature model an isolated portion of the original data.

How was the cut point determined? Each data point for each predictor is evaluated as a candidate cut point by creating a linear regression model with the candidate features, and the corresponding model error is calculated. The predictor/cut point comvination that achieves the smallest error is then used for the model. The nature of the predictor transformation makes such a large number of linear regressions computationally feasible. In some MARS implementation, including the one used here, the utility of simple linear terms for each predictor (i.e, no hinge function) is also evaluated.

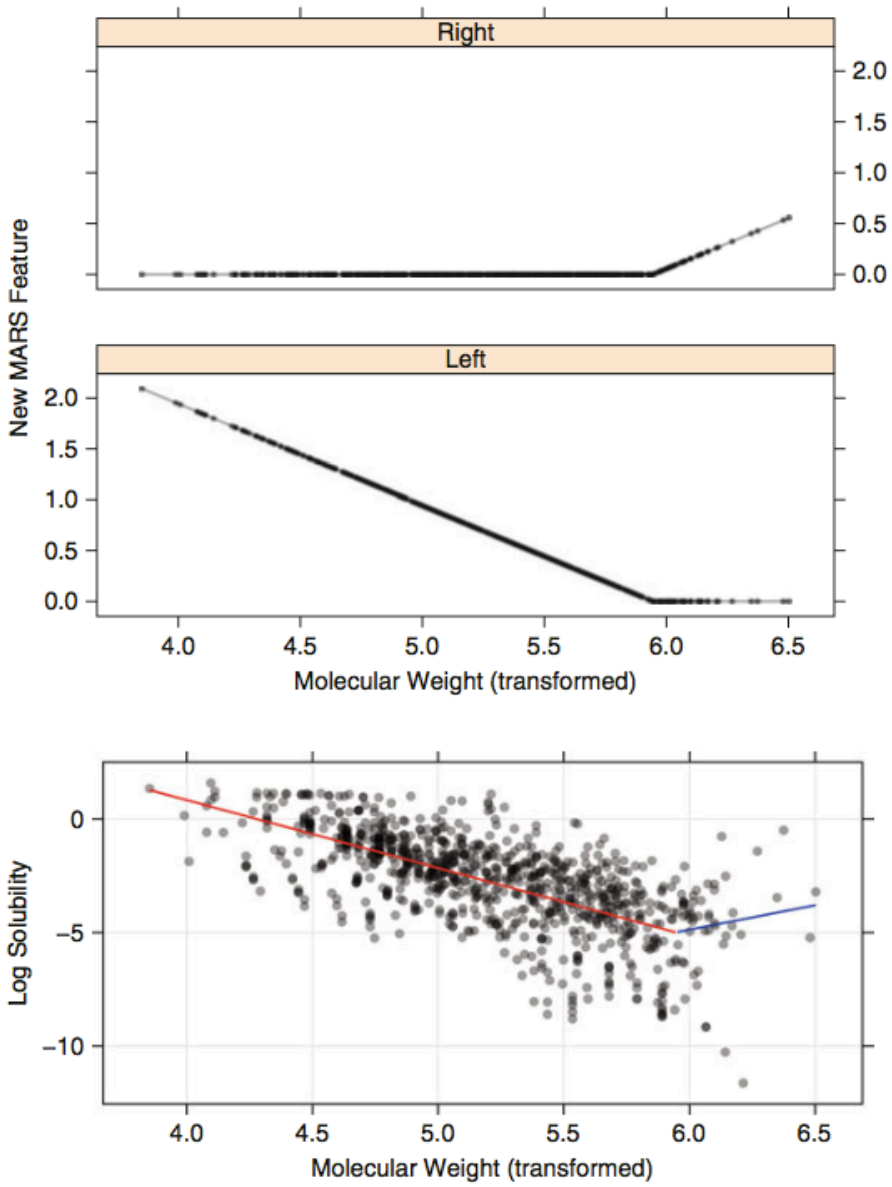
After the initial model is created with the first two features, the model conducts another exhaustive search to find the next set of features, that given the initial set, yield the best model fit. This process continues until a stopping point is reached. Mathematically, the hinge function for new features can be written as:

$$h(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

A pair of hinge funtions is usually written as $h(x - a)$ and $h(a - x)$. The first is nonzero when $x > a$, while the second is nonzero when $x < a$. Note that when this is true the value of the function is actually $-x$. For the MARS model, the actual equation would be.

$$-5 + 2.1 * h(\text{MolWeight - 5.94516}) + 3 * h(5.94516 - \text{MolWeight})$$

The first term in this equation (-5) is the intercept. The second term is associated with the right-hand feature, while the third term is associated with the left-hand feature. The figure below shows a few steps of the linear regression model from top to bottom. Here the binary fingerprint descriptor enters the model as a plain linear term. The generalized cross-validation (GCV) column shows the estimates RMSE for the model containing terms on the current row and all rows above.



Once the full set of features has been created, the algorithm sequentially removes individual features that do not contribute significantly to the model equation. This *pruning* procedure assesses each predictor variable and estimates how much the error rate was decreased by including it in the model. This process does not proceed backwards along the path that the features were added; some features deemed important at the beginning of the process may be removed while features added towards the end might be retained. To determine the ontribution of each feature to the mode, the GCV statistic is used. This value is a computational shortcut for linear regression models that produce an error value that approximates leave-one-out cross-validation. GCV produces better esimates than the apparent error rate for determining the importance of each feature in the model. The number of terms to remove can be manually set or treated as tuning parameter and determined using some other form of resampling. The process above is a description of an additive MARS model where each surrogate feature involves a single predictor. However, MARS can build models where the features involve multiple predictors at once.

3.5 Support Vector Machines

SVMs are a class of powerful, highly flexible modeling techniques. For regression the motivation of this technique in the framework of *robust regression* where we seek to minimize the effect of outliers on the regression equation. Also, there are several falvors of support vector regression and we focus on one partifuclar technique called *e-insensitive regression*.

Recall that linear regression seeks to find parameter estimates that minimize SSE. One drawback of minimizing SSE is that the parameter estimate can be influenced by just one observation that falls far from the overall trend in the data. When data may contain influential observations, an alternative minimization metric that is less senseitive, such as the Huber function, can be used to find the best parameter estimates. This function uses the squared residuals when they are "small" and uses the absolute residuals when the residuals are large. SVMs for regression use a function siilar to the Huber function, with an important difference. Given a threshold set by the user (denoted as ϵ) data points with residuals within the threshold do not contribute to the regression fit while data points with an absolute difference greater than the threshold contribute a linear-scale amount. There are several consequences to this approach. First, since the squared residuals are not used, large outliers have a limited effect on the regression equation.

4 Classification Models

4.1 Support Vector Machines

Support vector machines are a class of statistical models first developed in the mid-1960's by Vladimir Vapnik. Suppose we have a two-class problem and we code the class #1 samples with the value of 1 and the class #2 samples with -1. Also, let the vector x_i contain the predictor data for the training set samples. The maximum margin classifier creates a decision value $D(x)$ that classifies samples such that if $D(x) > 0$, we would predict a sample to be class #1, otherwise class #2. For an unknown sample $\mathbf{u}$, the decision equation can be written in a similar form as a linear discriminant function that is parameterized in terms of an intercept and slope as

$$D(u) = \beta_o + \beta u$$

$$= \beta_0 + \sum \beta_j u_j$$

Notice that this equation works from the viewpoint of the predictors. This equation can be transformed so that the maximum margin classifier can be written in terms of each data point in the sample. This changes the equation to

$$D(u) = \beta_0 + \sum \beta_j u_j$$

$$= \beta_0 + \sum y_i a_i x_i u$$

where u_i is for the unknown sample, and x_i contain the predictor data for a training set sample, and D(u) is the decision value for the classification of the unknown sample, where $a_i \geq 0$.

It turns out that, in the complete separable case, the a parameters are exactly zero for all samples that are not on the margin. Conversely, the set of nonzero a values are the points that

fall on the boundary of the margin. Because of this, the predictor equation is a function of only a subset of the training set points and these are referred to as the *support vectors*. Interestingly, the prediction function is only a function of the training set samples that are closest to the boundary and are predicted with the least amount of certainty.

Since the prediction equation is *supported* solely by these data points, the maximum margin classifier is usuallly called the *support vector machine*. On first examination, it may appear someone arcane. However, it can shed some light on how we support vector machines classify new samples.

Consider the following figure; Where a new sample, shown as a solid grey circle, is predicted by the model. The distance between each of the support ectors and the new sample are as grey dotted lines.

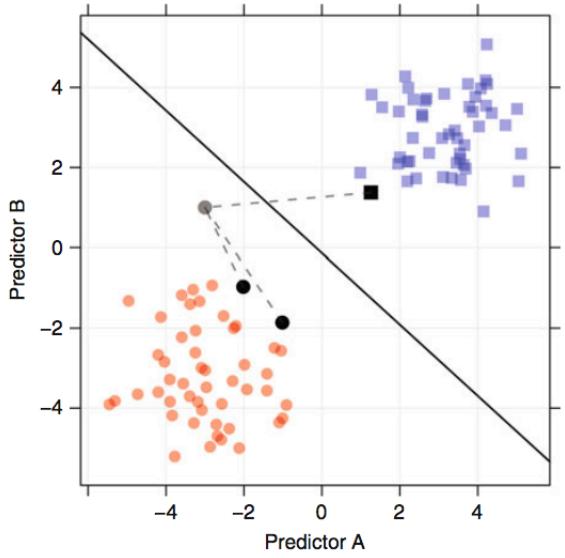


Fig. 13.10: Prediction of a new sample using a support vector machine. The final value of the decision equation is $D(u) = 0.583$. The grey lines indicate the distance of the new sample to the support vectors

For these data, there are three support vectors, and therefore contain the only information necessary for classifying the new sample. The meat of the equation for SVM is the summation of the product of the sign of the class, the model parameter, and the dot product between the new sample and the support vector predictor values. The following table shows the components of this sum, broken down for each of these three support vectors.

	True class	Dot product	y_i	α_i	Product
SV 1	Class 2	−2.4	−1	1.00	2.40
SV 2	Class 1	5.1	1	0.34	1.72
SV 3	Class 1	1.2	1	0.66	0.79

The dot product, $x_i u$ can be written as a product of the distance of x_i from the origin, the distance of u from the origin, and the cosine of the angle between x_i and u.

$$x \cdot u = ||x|| \cdot ||u|| \cdot \cos(\theta)$$

Based on the parameter estimates a_i, the first support vector has the largest single effect on

the prediction equation (all other things being equal) and it has a negative slope. For out new sample, the dot product is negative, so the total contribution of ths point is positive and pushes the predicton towards the first class (.e., a positive value of the decision function $D(u)$). The remaining two support vectors have positive dot products and an overall product that increases the decision function for this new sample. For this model, the intercept is -4.372; $D(u)$ for the new sample is therefore 0.583. Snce this value is greater than zero, the new sample has the highest association with the first class.

What happens when the classes are not completely seperable? there is a developed extension to the early maximum margin classifier to accommodate this situation. Their formulation puts a cost on the sum of the training set points that are on the boundary, or on the wrong side of the boundary. When determining the estimates of the a values, the margin is penalized when data points are on the wrong side of the class boundary or inside the margin. The cost value would be a tuning parameter for the model and is the primary mechanism to control the complexity of the boundary. For example, as the cost of errors increases, the classification boundary will shift and contort itself so that it correctly classifies as many of the training set points as possible.

4.2 Non-Linear Support Vector Machines

To extend the Support Vector Machine towards a non-linear decision boundary, we subsitute the kernal function instead of a simple linear cross product.

$$D(u) = \beta_0 + \sum y_i a_i x_i u$$

$$= \beta_0 + \sum y_i a_i K(x_i u)$$

where K(.,.) is a *kernel function* of the two vectors. For the linear case, the kernel function is the same inner product $x_i u$/ However, just as in regression SVMs, other nonlinear transformations can be applied, including:

$$\text{polynomial} = (\text{scale}(xu) + 1)^{\text{degree}}$$

$$\text{Radial Basis Function} = exp(-\sigma||x - u||^2)$$

$$\text{hyperbolic tangent } = tanh(scale(xu) + 1)$$

Note that, due to the dot product, the predictor data should be centered and scaled prior to fitting so that attributes whos values are large in magnititude do not dominate the calculatons.

The *kernel trick* allows the SVM model to produce extremely flexible decison boundaries. The choice of the kernel function parameters and the cost value control the complexity and should be tuned appropriately so that the model does not overfit the training data.

4.3 Basic Classificatons Trees

As with regression trees, the aim of the classification is to partition the data nto homogeneous groups. Homogeneity in this context means that the nodes of the split are more pure. A smple way to define purity in classification is by maximizing accuracy, or equivalently minimizing the misclassification error. Accuracy as a measure of purity, is a bit misleading since the measure's

focus is on partitionng the data in a way that places samples primarily in one class.

Two alternative measures, the Gini Index and the cross entropy, which is also referred to as deviance of information, shift the focus from accuracy to purity. For two-class problems, the Gini index for a given node is defined as:

$$p_1(1-p_1)+p_2(1-p_2)$$

where p_1 and p_2 are the Class 1 and Class 2 probabilities, respectively. Since this is a two-class problem $p_1+p_2=1$, and therefore can be equivalently written as $2p_1p_2$. It is easy to see that the Gini index is minimized when either the class probabilities is driven towards zero, meaning that the node is pure with respect to one of the classes. Conversely, the Gini index is maximized when $p_1=p_2$, the case in which the node is least pure.

When working with a continous predictor and a categorical response, the process for finding the optimal split point is as such:
We first calculate the Gini index prior to the split

$$\text{Gini(prior to split)} = 2(\frac{n_{1+}}{n})(\frac{n_{2+}}{n})$$

and the gini index can be calculated after the split within each of the new nodes with values $2(\frac{n_{11}}{n_{+1}})(\frac{n_{12}}{n_{+1}})$ and $2(\frac{n_{11}}{n_{+1}})(\frac{n_{12}}{n_{+1}})$ for greater than and less than or equal to the split, respectively. These values are combined using the proportion of samples in each part of the split as weights with $(\frac{n_{+1}}{n})$ and $(\frac{n_{2+}}{n})$ representing the respective weights for greater than and less than or equal to the split. After some simplification, the Gini index evaluation would be:

$$\text{Gini(after spit)} = 2[(\frac{n_{11}}{n_{+1}})(\frac{n_{12}}{n_{+1}})+(\frac{n_{21}}{n_{+1}})(\frac{n_{21}}{n_{+2}})]$$

Note: Trees that are constructed to have the maximum depth are notorious for over-fitting the training data. A more generalizable tree is one that is a pruned version of the initial tree and can be determined by cost-complexity tuning, in which the purity criterion is penalized by a factor of the total number of terminal nodes in the tree. The cost-complexity factor is called the complexity parameter and can be incorporated into the tuning process so that an optimal value can be estimated.

Similar to regression trees, classification trees can handle missing data. In tree construction, only samples with non-missing information are considered for creating the split. In prediction, surrogate splits can be used in place of the split for which there are missing data. Likewise, variable importance can be computed for classification trees by assessing the overall improvement in the optimization criteria for each predictor.

4.3.1 Bagged Trees

Bagging for classification is a simple modification of bagging for regression. Specifically, the regression tree is replaced with an unpruned classification tree for modeling C classes. Since each model has equal weight in the ensemble, each model can be thought of as casting a vote for the class it thinks the new sample belongs to. The total number of votes within each class are then divided by the total number of models in the ensemble (M) to produce a predicted probability

vector for the sample. The new sample is then classified into the group that has the most votes, and therefore the highest probability.

4.3.2 Random Forest

Random forest for classification requires a simple tweak to the random forest regression algorithm. As with bagging, each tree in the forest in the forest casts a vote for the classification of a new sample, and the proportion of votes in each class across the ensemble is the predicted probability vector.

While the type of tree changes in the algorithm, the tuning parameter of the number of randomly selected predictors to choose from at each split is the same (denoted as m_{try}). As in regression, the idea behind randomly sampling predictors during training is to de-correlate the trees in the forest.

4.3.3 Boosting

To summarize the algorithm, AdaBoost generates a sequence of weak classifiers, where at each iteration the algorithm finds the best classifier based on the current sample weights. Samples that are incorrectly classified in the *kth* iteration recieve more weight in the (k + 1)st iteration, while samples that are correctly classified recieve less weight in the subsequent iteration. This means that samples that are difficult to classify reieve increasingly larger weights until the algorithm indentifies a model that correctly classifies these samples. Therefore, each iteration of the algorithm is required to learn a different aspect of the data, focusing on regions that contain difficult-to-classify samples. At each iteration, a *stage weight* is computed based on the error rate at that iteration. The nature of the stage weights descrived in Algorithm implies that more accurate models have higher positive values and less accurate models have lower negative values. The overall sequence of weighted classifiers is then combined into an ensemble and has a strong potential to classify better than any of the individual classifiers.

Boosting can be applied to any classification techniques, but classification trees are a popular method for boosting since these can be made into weak learners by restricting the tree depth to create trees wth few splits.