Predictive Modeling

Department of Mathematical Sciences
Qiuying Sha, Professor

Michigan Tech

Chapter 7. Nonlinear Regression Models

- The previous chapter discussed regression models that were intrinsically linear.
- Many of these models can be adapted to nonlinear trends in the data by manually adding model terms (e.g., squared terms).
- There are numerous regression models that are inherently nonlinear in nature.
- When using these models, the exact form of the nonlinearity does not need to be known explicitly or specified prior to model training.
- This chapter looks at several models:
  - Neural Networks,
  - Multivariate Adaptive Regression Splines (MARS),
  - Support Vector Machines (SVMs), and
  - *K*-Nearest Neighbors (*K*NNs).

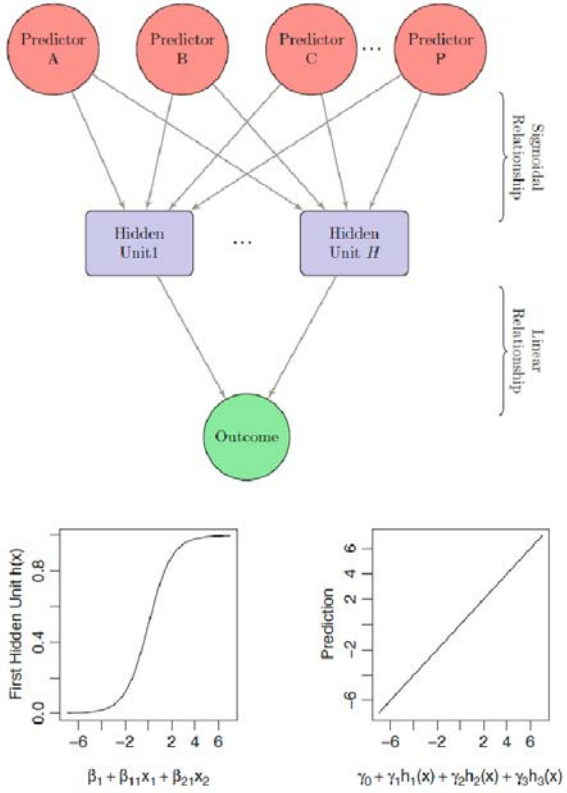


Fig. 7.1: A diagram of a neural network with a single hidden layer. The hidden units are linear combinations of the predictors that have been transformed by a sigmoidal function. The output is modeled by a linear combination of the hidden units

- Tree-based models are also nonlinear. These models are covered in Chapter 8.

7.1 Neural Networks

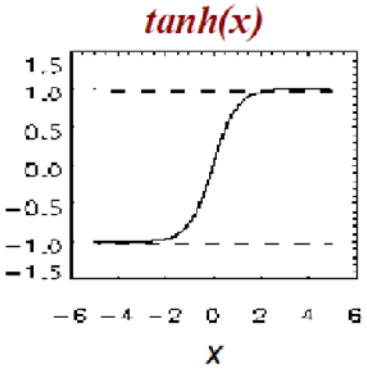
- Like PLS, the outcome is modeled by an intermediary set of unobserved variables (called *hidden variables* or *hidden units*)
- Each hidden unit is a linear combination of some or all of the predictor variables. Then the linear combination is transformed by a nonlinear function $g(\cdot)$, called activation function.

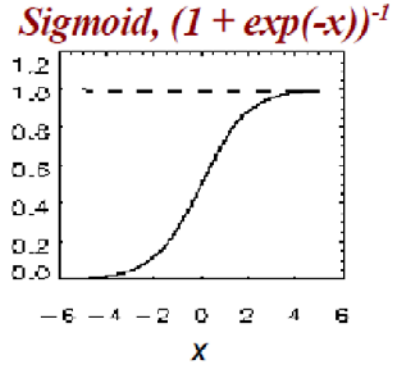
$$h_k(\mathbf{x}) = g\left(\beta_{0k} + \sum_{i=1}^{P} x_j \beta_{jk}\right),$$

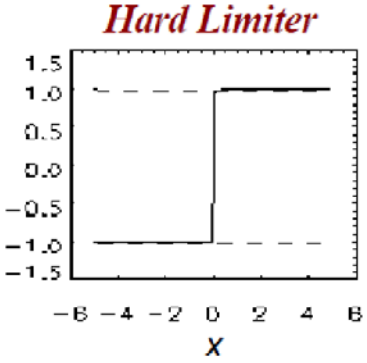
Some popular activation functions are pictured on the right: tanh, sigmoid, hard limiter, and ramp function

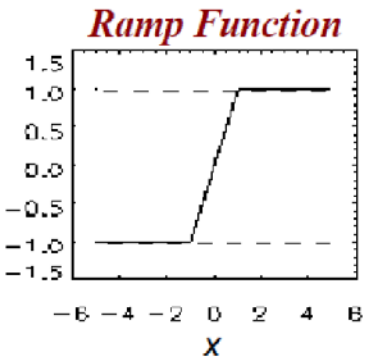
- Once the number of hidden units is defined, each unit must be related to the outcome.

$$f(\mathbf{x}) = \gamma_0 + \sum_{k=1}^{H} \gamma_k h_k.$$









- There are a total of $H(P+1)+H+1$ total parameters being estimated in the model.

- It is common that a solution to this equation is not a global solution, meaning that we cannot guarantee that the resulting set of parameters are uniformly better than any other set.

- Also, neural networks have a tendency to **over-fit the relationship** between the predictors and the response due to the large number of regression coefficients.

- To combat this issue, we can use ***weight decay***, a penalization method to *regularize* the model similar to ridge regression.

Minimize an alternative version of the sum of the squared errors:

$$\sum_{i=1}^{n} (y_i - f_i(x))^2 + \lambda \sum_{k=1}^{H} \sum_{j=0}^{P} \beta_{jk}^2 + \lambda \sum_{k=0}^{H} \gamma_k^2$$

for a given value of λ. Reasonable values of λ range between 0 and 0.1.

Also note that since the regression coefficients are being summed, they should be on the same scale; hence the predictors should be **centered and scaled prior to modeling**.

- Extension of the basic network:
  - We introduce a **single-layer** network. There are many other kinds, such as models with more than one layer of hidden units.
- The fitted model finds parameter estimates that are locally optimal;
- As an alternative, several models can be created using different starting values and averaging the results of these models to produce a more stable prediction (average the predicted values).

Such ***model averaging*** often has a significantly positive effect on neural networks.

When optimizing the model parameters, **high correlation among the predictor variables** often adversely affect these models.

Two approaches for mitigating this issue

- Pre-filter the predictors to **remove the predictors** that are associated with high correlations.
- A feature extraction technique, such as principal component analysis, can be used prior to modeling to eliminate correlations.

One positive side effect of both these approaches is that fewer model terms need to be optimized, thus improving computation time

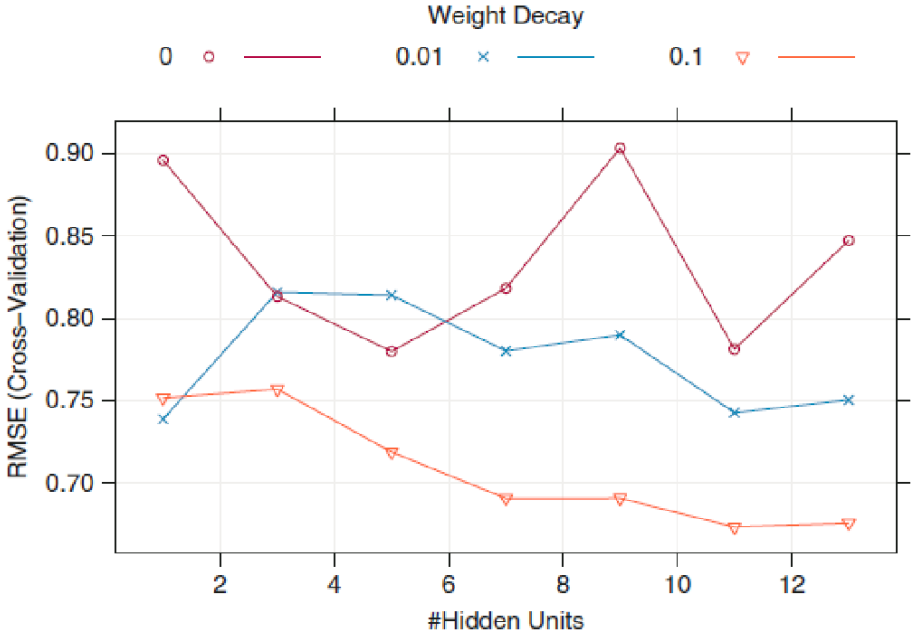


Fig. 7.2: RMSE profiles for the neural network model. The optimal model used $\lambda = 0.1$ and 11 hidden units

For the solubility data,

- Model averaged neural networks were used.
- Three different weight decay values, λ = 0.00, 0.01, 0.10, and with a single hidden layer with sizes ranging between 1 and 13 hidden units were evaluated.
- The final predictions are the averages of five different neural networks created using different initial parameter values.
- The cross-validated RMSE of these models are displayed in Fig. 7.2.
- The optimal model used 11 hidden units with a total of 2,531 coefficients.

7.2 Multivariate Adaptive Regression Splines (MARS)

- See supplement MARS from http://en.wikipedia.org/wiki/Multivariate_adaptive_regression_splines

MARS

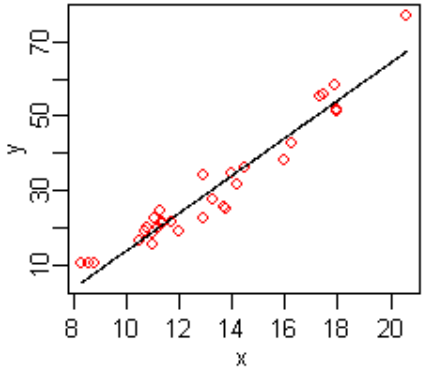
- was introduced by Jerome H. Friedmanin in 1991.
- can be seen as an extension of linear models that automatically models non-linearities and interactions between variables.

We introduce MARS using two examples.

Example 1. We start with one independent variable X and responding variable Y.

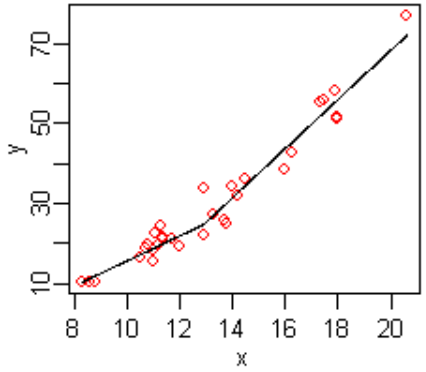
- If I build a linear model which predicts the expected *y* for a given *x* for the data set on the right, the fitted linear model is

$$\hat{y} = -37 + 5.1x$$



- If I use MARS to build a model taking into account non-linearities, the fitted model is

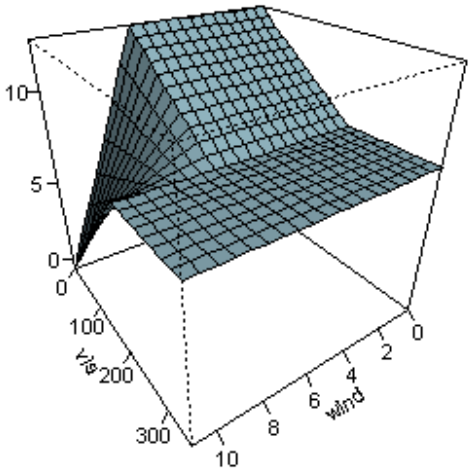
$$\hat{y} = 25 + 6.1\max(0, x - 13) - 3.1\max(0, 13 - x)$$



Michigan Tech

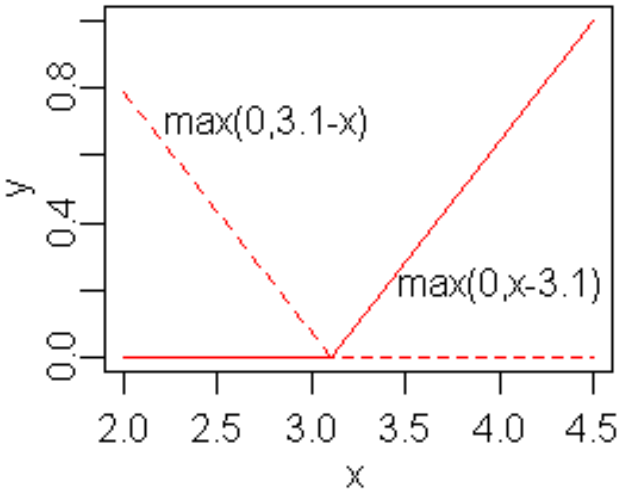
Example 2. A MARS model with multiple variables. The fitted model is

$$
\begin{aligned}
ozone = {} & 5.2 \\
& + 0.93 \max(0, \text{temp} - 58) \\
& - 0.64 \max(0, \text{temp-68}) \\
& - 0.046 \max(0, 234 - ibt) \\
& - 0.016 \max(0, wind - 7) \max(0, 200 - vis)
\end{aligned}
$$



- This expression models air pollution (the ozone level) as a function of the temperature and a few other variables.
- Note that the last term in the formula incorporates **an interaction** between wind and visibility.
- The figure shows that wind does not affect the ozone level unless visibility is low.
- We see that MARS can build quite flexible regression surfaces by combining hinge functions.

Mars uses hinge functions.



Hinge functions:

- Hinge functions are a key part of MARS models.
- A hinge function takes the form

$$\max(0, x - c) \text{ or } \max(0, c - x)$$

where c is a constant, called *knot*

- The figure on the right shows a mirrored pair of hinge functions with a knot at 3.1.
- A mirrored pair of hinge functions in the expression creates the piecewise linear graph.

$$6.1\max(0, x - 13) - 3.1\max(0, 13 - x)$$

The MARS model

MARS builds models of the form

$$\hat{f}(x) = \sum_{i=1}^{k} c_i B_i(x)$$

The model **is a weighted sum of basis functions $B_i(x)$** . Each c_i is a constant.

Each basis function $B_i(x)$ takes one of the following three forms:

1) a constant 1. There is just one such term, the intercept.
2) a *hinge* function.
   - A hinge function has the form $\max(0, x - c)$ or $\max(0, c - x)$
   - MARS automatically selects variables and values of those variables for knots of the hinge functions.

3) a product of two or more hinge functions.
   - These basis functions can model interaction between two or more variables

The model building process

MARS builds a model in two phases: **the forward and the backward pass**

The forward pass:

- MARS starts with a model which consists of just the intercept term (which is the mean of the response values).

- MARS then repeatedly adds basis function in pairs to the model.

  - At each step it finds the pair of basis functions that **gives the maximum reduction in sum-of-squares residual error**.
  - The two basis functions in the pair are identical except that a different side of a mirrored hinge function is used for each function.

- This process of adding terms continues **until the change in residual error is too small** to continue or until **the maximum number of terms** is reached.

The maximum number of terms is specified by the user before model building starts.

The backward pass:

The forward pass usually builds an overfit model.

- The backward pass prunes the model.
- It removes terms one by one, **deleting the least effective term** at each step until it finds the best submodel.
- Model subsets are compared using the **GCV criterion**.
- The backward pass has an advantage over the forward pass: at any step it can choose any term to delete.

Generalized Cross Validation (GCV)

- The backward pass uses GCV to compare the performance of model subsets in order to choose the best subset**: lower values of GCV are better**.

- The GCV is a form of regularization: it trades off goodness-of-fit against model complexity.

- We want to estimate how well a model performs on *new* data, not on the training data.
  - Such new data is usually not available at the time of model building, so instead we use GCV to estimate what performance would be on new data.
  - The raw residual sum-of-squares (RSS) on the training data is inadequate for comparing models, because the RSS always increases as MARS terms are dropped.

The formula for the GCV is

$$GCV = \frac{RSS}{n(1 - ENOP / n)^2}$$

- RSS is the residual sum-of-squares measured on the training data
- n is the number of observations.
- EffectiveNumberOfParameters (ENOP)
  = NumberOfMarsTerms + Penalty * (NumberOfMarsTerms - 1) / 2

  Penalty is about 2 or 3 (the MARS software allows the user to preset Penalty).

  (NumberOfMarsTerms - 1) / 2 is the number of hinge-function knots, so the formula penalizes the addition of knots

- GCV formula adjusts the training RSS to take into account the flexibility of the model.

- GCV is so named because it uses a formula to approximate the error that would be determined by leave-one-out validation.

Constraints in MARS

- The maximum number of terms in the forward pass.
- A maximum allowable degree of interaction in the forward pass.

Typically only one or two degrees of interaction are allowed, but higher degrees can be used when the data warrants it.

Pros and Cons of MARS

- No regression modeling technique is best for all situations.
- The guidelines below are intended to give an idea of the pros and cons of MARS, but there will be exceptions to the guidelines

- MARS models are more flexible than linear regression models.
- MARS models are simple to understand and interpret.
- MARS can handle both continuous and categorical data.
- Building MARS models often requires little or no data preparation??

(Nevertheless, as with most statistical modeling techniques, known **outliers** should be considered for removal before training a MARS model.)

- MARS does ***automatic variable selection*** (meaning it includes important variables in the model and excludes unimportant ones).

Notes: there is usually some arbitrariness in the selection, especially in the presence of **collinearity**.

- MARS models tend to have a good bias-variance trade-off.
- MARS models have fairly low variance.
- MARS is suitable for handling fairly large datasets.
- With MARS models, as with any non-parametric regression, parameter confidence intervals on the model cannot be calculated directly.
- The earth, mda, and polspline implementations **do not allow missing values** in predictors.
- MARS models can make predictions quickly.

Results of solubility data.

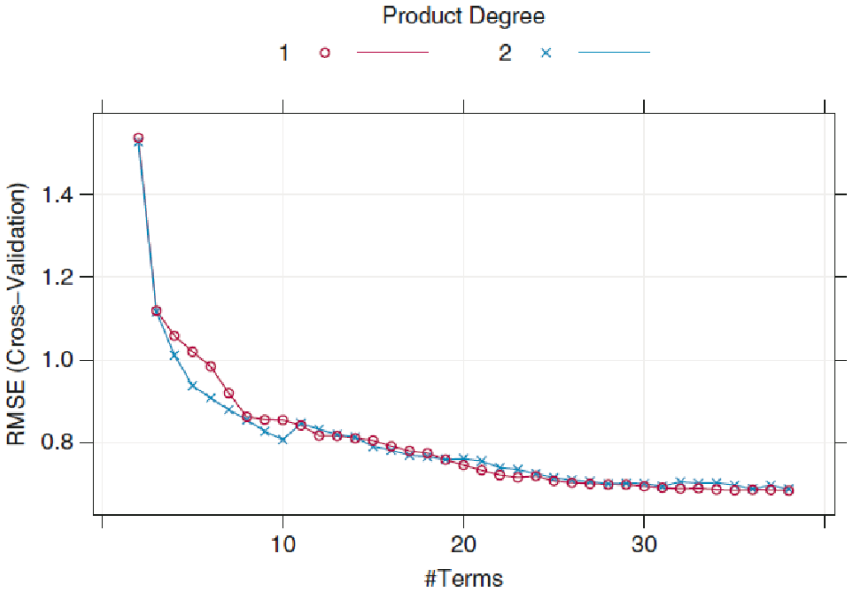


Fig. 7.4: RMSE profiles for the MARS model. The cross-validation procedure picked a second-degree model with 38 terms, although there is little difference between the first- and second-degree models. Given this equivalence, the more simplistic first-order model was chosen as the final model

7.3 Support Vector Machines (SVM)

There are several flavors of support vector regression.

We focus on one particular technique called ***ε-insensitive regression***.

Linear regression seeks to find parameter estimates that minimize SSE

- One drawback of minimizing SSE -- parameter estimates can be influenced by **outliers**.

SVM uses a function similar to the Huber function.

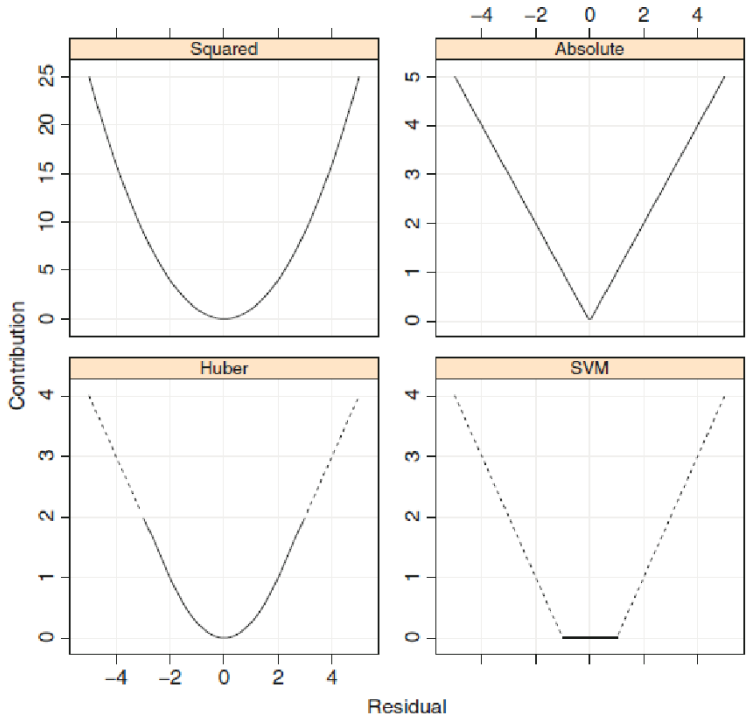


Fig. 7.6: The relationship between a model residual and its contribution to the regression line for several techniques. For the Huber approach, a threshold of 2 was used while for the support vector machine, a value of $\epsilon = 1$ was used. Note that the y-axis scales are different to make the figures easier to read

Given a threshold ε,

- Data points with residuals within the threshold do not contribute to the regression fit.
- Data points with an absolute difference greater than the threshold contribute a linear-scale amount.

ε- *insensitive* function is shown in Fig.7.6.

Notes about SVMs:

- First, the squared residuals are not used, large outliers have a limited effect on the regression equation.
- Second, samples that the model fits well (i.e., the residuals are small) have *no* effect on the regression equation.

The SVM regression coefficients minimize

$$Cost \sum_{i=1}^{n} L_\epsilon(y_i - \hat{y}_i) + \sum_{i=1}^{P} \beta_j^2,$$

where $L_\varepsilon(\cdot)$ is the ε-insensitive function. The *Cost* parameter is the *cost* penalty set by the user, which penalizes large residuals.

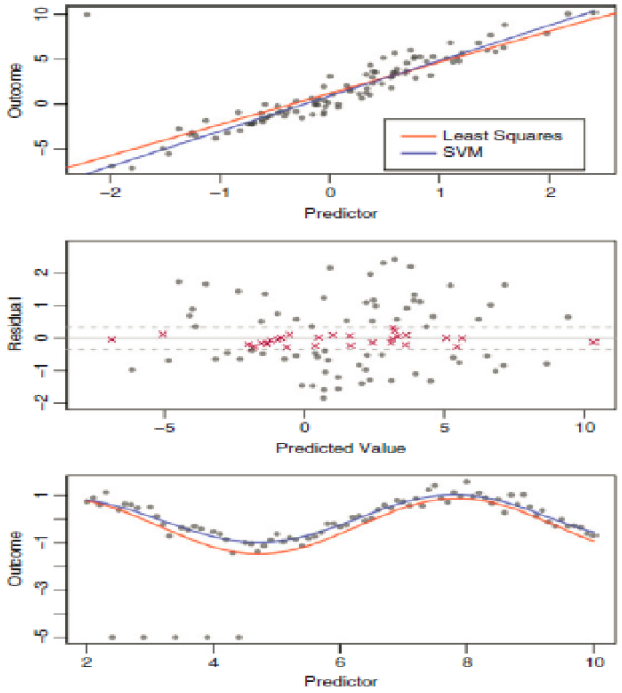


Fig. 7.7: The robustness qualities of SVM models. *Top*: a small simulated data set with a single large outlier is used to show the difference between an ordinary regression line (*red*) and the linear SVM model (*blue*). *Middle*: the SVM residuals versus the predicted values (the upper end of the y-axis scale was reduced to make the plot more readable). The plot symbols indicate the support vectors (shown as *grey colored circles*) and the other samples (*red crosses*). The *horizontal lines* are $\pm\varepsilon = 0.01$. *Bottom*: A simulated sin wave with several outliers. The *red line* is an ordinary regression line (intercept and a term for $sin(x)$) and the *blue line* is a radial basis function SVM model

- True function: $y = 1 + 4x$ with one extreme outlier.
- Linear model: $\hat{y} = 1.2 + 3.5x$
- SVM: $\hat{y} = 0.9 + 3.9x$ with 70 support vectors.

Support vectors support the regression line.

- True model: $y = \sin(x)$
- Linear regression: fit

$$y = \beta_0 + \beta_1 \sin x + \varepsilon$$

- SVM: without specifying the sin function.

Michigan Tech

Simple linear regression predicts a new sample, *u*, by

$$\begin{aligned}\hat{y} &= \beta_0 + \beta_1 u_1 + \ldots + \beta_P u_P \\ &= \beta_0 + \sum_{j=1}^{P} \beta_j u_j\end{aligned}$$

Linear SVM predicts *u* by

$$\begin{aligned}\hat{y} &= \beta_0 + \beta_1 u_1 + \ldots + \beta_P u_P \\ &= \beta_0 + \sum_{j=1}^{P} \beta_j u_j \\ &= \beta_0 + \sum_{j=1}^{P} \sum_{i=1}^{n} \alpha_i x_{ij} u_j \\ &= \beta_0 + \sum_{i=1}^{n} \alpha_i \left(\sum_{j=1}^{P} x_{ij} u_j \right).\end{aligned} \tag{7.2}$$

α_i are unknown parameters and n is the number of data points in the training set.

The regression equation in (7.2) can be rewritten as

$$f(\mathbf{u}) = \beta_0 + \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{u}),$$

K(·) is called the *kernel function*.

$$K(\mathbf{x}_i, \mathbf{u}) = \sum_{j=1}^{P} x_{ij} u_j = \mathbf{x}_i' \mathbf{u}.$$

Other types of kernel functions:

$$\begin{aligned} \text{polynomial} &= \left(\phi\left(\mathbf{x}'\mathbf{u}\right) + 1\right)^{degree} \\ \text{radial basis function} &= \exp(-\sigma\|\mathbf{x} - \mathbf{u}\|^2) \\ \text{hyperbolic tangent} &= \tanh\left(\phi\left(\mathbf{x}'\mathbf{u}\right) + 1\right), \end{aligned}$$

where Φ and σ are scaling parameters.

Which kernel function should be used?

- If the regression line is truly linear, the linear kernel function will be a better choice.
- Otherwise, the **radial basis function** has been shown to be very effective.

The radial basis function has a parameter (σ) that controls the scale, is a tuning parameter.

- Tune this parameter over a grid of candidate values.
- Use a computational shortcut to estimating the kernel parameter (Caputo et al. 2002)
  - Calculate the distribution of $\|x - x'\|^2$ for the training set points.
  - Use the 10th and 90th percentiles as a range for σ.
  - Use the midpoint of these two percentiles to estimate σ.

The cost parameter is used to adjust the complexity of the model.

- When the cost is large, the model becomes very flexible (likely to over-fit).
- When the cost is small, the model will “stiffen” (more likely to under-fit).

The authors suggest

- fixing a value for ε and tuning over the other kernel parameters.
  - There is a relationship between ε and the cost parameter. In authors’ experience, the cost parameter provides more flexibility for tuning the model.

Centering and scaling the predictors prior to building an SVM model.

SVMs were applied to the solubility data.

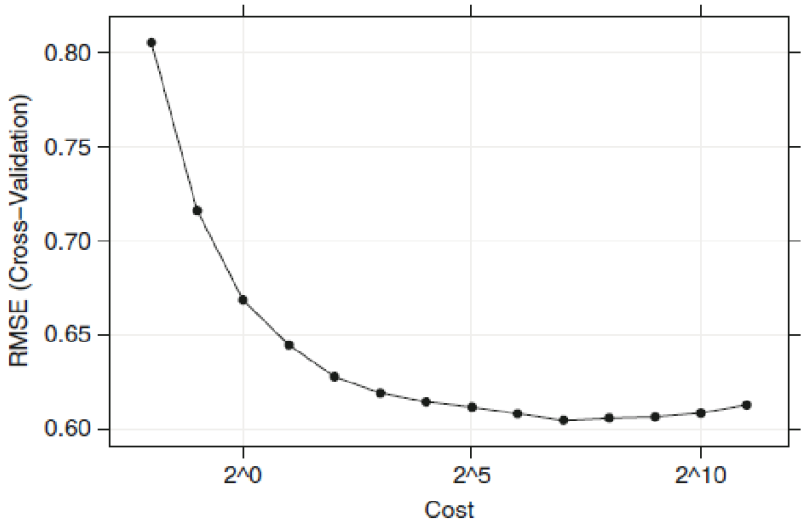


Fig. 7.8: The cross-validation profiles for a radial basis function SVM model applied to the solubility data. The kernel parameter was estimated analytically to be $\sigma = 0.0039$

- A radial basis kernel function was used.
- The kernel parameter was estimated analytically to be $\sigma = 0.0039$.
- The model was tuned over 14 cost values between 0.25 and 2048.

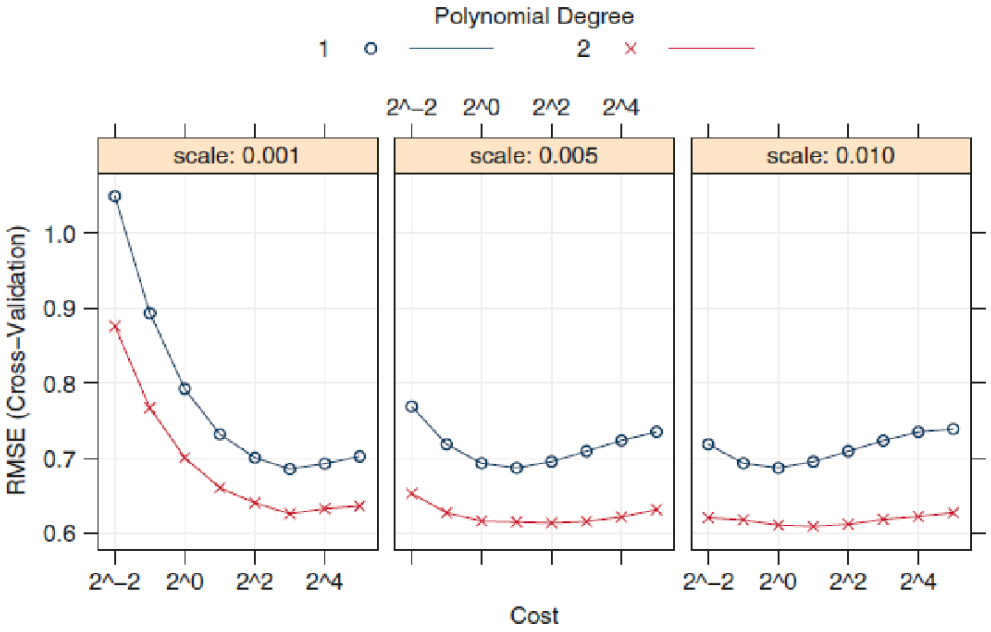


Fig. 7.9: Cross-validation results for the polynomial SVM model for the solubility data. The final model was fit using a quadratic model with a scale factor of 0.01 and a cost value of 2

- A polynomial model was also evaluated.
- The cost, the polynomial degree, and a scale factor were tuned.
- Both the optimal radial basis and the polynomial SVM models use a similar number of support vectors, 623 and 627 (out of 951 training samples).
- Tuning the radial basis function kernel parameter was easier than tuning the polynomial model (which has three tuning parameters).

7.4 *K*-Nearest Neighbors (KNN)

The *K*NN approach predicts a new sample using the *K*-closest samples from the training set.

To predict a new sample for regression,

- *K*NN identifies that sample's *K*NNs in the predictor space.
- The predicted response for the new sample is the mean of the *K* neighbors' responses.

Other summary statistics, such as the median, can also be used in place of the mean to predict the new sample.

The basic *K*NN method depends on how the user defines distance:

- The most commonly is the Euclidean distance

$$\left(\sum_{j=1}^{P} (x_{aj} - x_{bj})^2 \right)^{\frac{1}{2}}$$

- A generalization of Euclidean distance is the Minkowski distance

$$\left(\sum_{j=1}^{P} |x_{aj} - x_{bj}|^q \right)^{\frac{1}{q}}$$

where $q > 0$.

***Pre-processing the data*:**

- The *K*NN method fundamentally depends on distance between samples, **the scale of the predictors** can have a dramatic influence on the distances among samples
  - ➢ We recommend that all predictors be **centered and scaled** prior to performing *K*NN.

- Using distances between samples can be problematic if one or more of the predictor values for a sample is **missing**.
  - ➢ The samples or the predictors can be excluded from the analysis.
  - ➢ **Impute the missing data** using a naıve estimator such as the mean of the predictor, or a nearest neighbor approach that uses only the predictors with complete information (see Sect. 3.4).

Find the optimal number of neighbors.

- Like tuning parameters from other models, *K* can be determined by resampling.

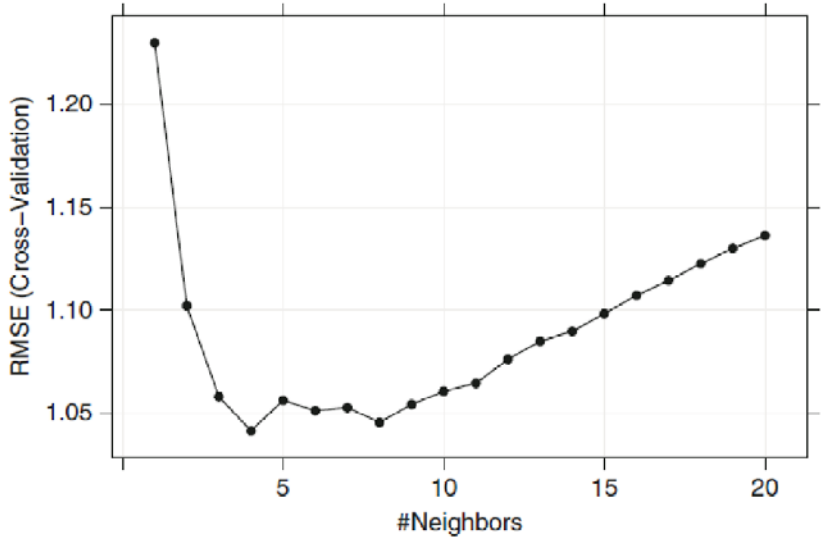


Fig. 7.10: The RMSE cross-validation profile for a KNN model applied to the solubility data. The optimal number of neighbors is 4

For the solubility data, 20 values of *K* were evaluated.

Facts:

- Small values of *K* usually over-fit the model.
- Large values of *K* under-fit the model.

Two commonly noted problems are

- **Computational time**.
- The disconnect between local structure and the predictive ability of *K*NN.
  - The *K*NN method can have poor predictive performance when **local predictor structure is not relevant to the response**.
  - **Irrelevant or noisy predictors** are one culprit.
  - **Removing irrelevant, noise-laden predictors** is a key pre-processing step for *K*NN.

Another approach to enhancing *K*NN predictivity is to **weight the neighbors' contribution** to the prediction of a new sample based on their distance to the new sample.

- Training samples that are **closer** to the new sample contribute more to the predicted response, while those that are **farther away contribute less** to the predicted response.