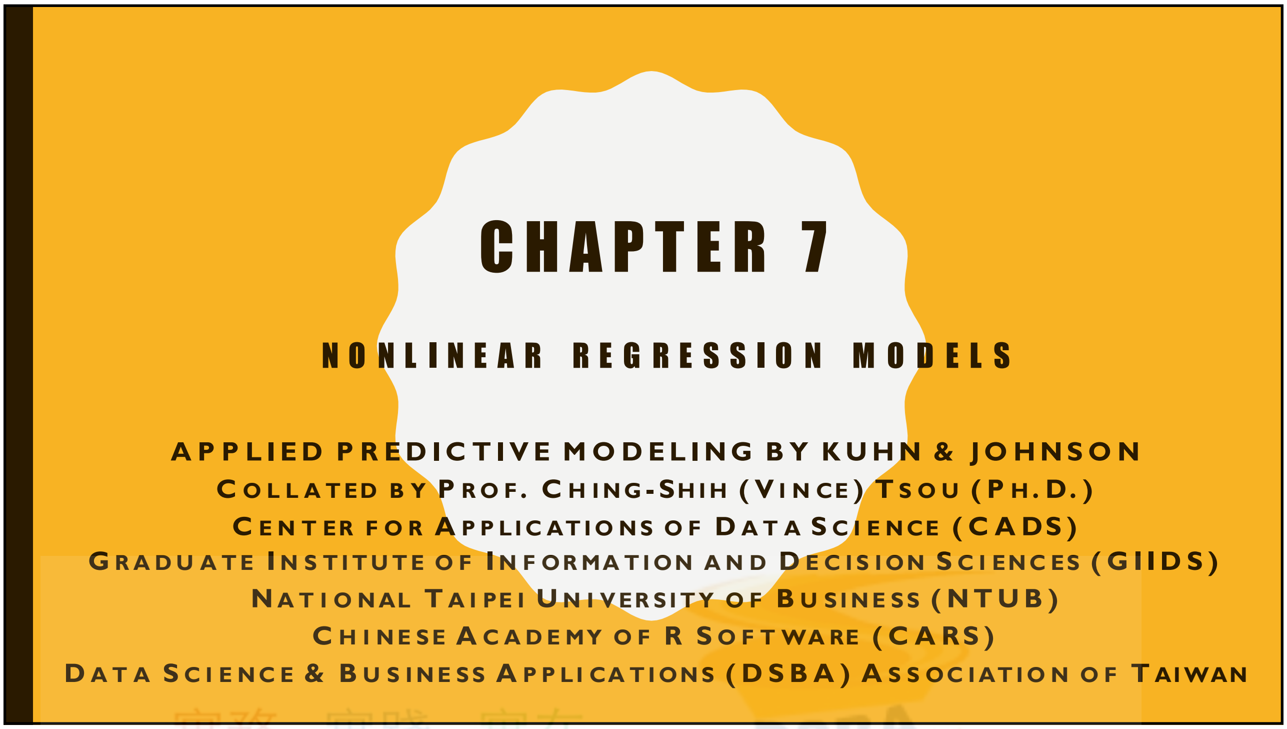


CHAPTER 7

NONLINEAR REGRESSION MODELS

APPLIED PREDICTIVE MODELING BY KUHN & JOHNSON
COLLATED BY PROF. CHING-SHIH (VINCE) TSOU (PH.D.)
CENTER FOR APPLICATIONS OF DATA SCIENCE (CADS)
GRADUATE INSTITUTE OF INFORMATION AND DECISION SCIENCES (GIIDS)
NATIONAL TAIPEI UNIVERSITY OF BUSINESS (NTUB)
CHINESE ACADEMY OF R SOFTWARE (CARS)
DATA SCIENCE & BUSINESS APPLICATIONS (DSBA) ASSOCIATION OF TAIWAN

AGENDA

- Introduction
- Neural Networks
- Multivariate Adaptive Regression Splines
- Support Vector Machines
- K-Nearest Neighbors
- Computing

INTRODUCTION – NONLINEARITY KNOWN OR UNKNOWN

- 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.
- To do this, one must *know the specific nature of the nonlinearity* in the data.
- There are numerous regression models that are *inherently nonlinear in nature*.

INTRODUCTION – NONLINEARITY KNOWN OR UNKNOWN

- 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 (NN), multivariate adaptive regression splines (MARS), support vector machines (SVMs), and K-nearest neighbors (KNNs)*.
- Tree-based models are also nonlinear. Due to their *popularity and use in ensemble models*, we have devoted the *next chapter* to those methods.

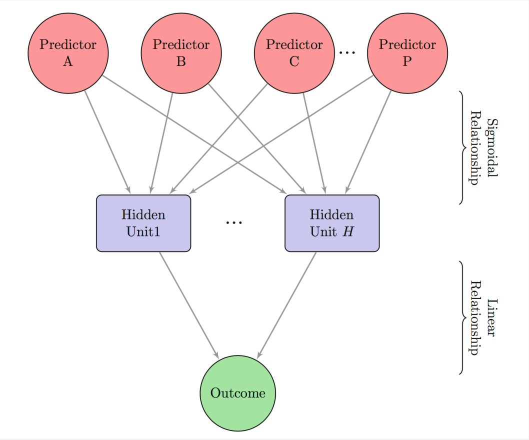
NEURAL NETWORKS – TOPOLOGY AND COMPUTATION

- Neural networks are powerful nonlinear regression techniques inspired by theories about *how the brain works*. Like partial least squares, the outcome is modeled by an *intermediary* set of *unobserved* variables (called hidden variables or *hidden units* here).
- As previously stated, each hidden unit is *a linear combination* of some or all of the predictor variables. However, this linear combination is typically *transformed by a nonlinear function* g(·), such as the *logistic function*: (i -> j)

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

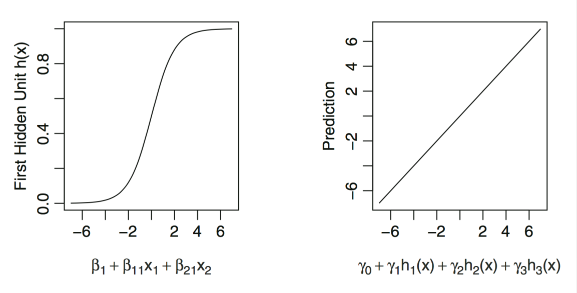
$$g(u) = \frac{1}{1+e^{-u}}.$$

NEURAL NETWORKS – TOPOLOGY AND COMPUTATION



- These hidden units are linear combinations of the original predictors, but, unlike PLS models, they are NOT estimated in a hierarchical fashion. (sigmoidal transform)
- A neural network model usually involves *multiple hidden units* to model the outcome.

NEURAL NETWORKS – ACTIVATION FUNCTIONS



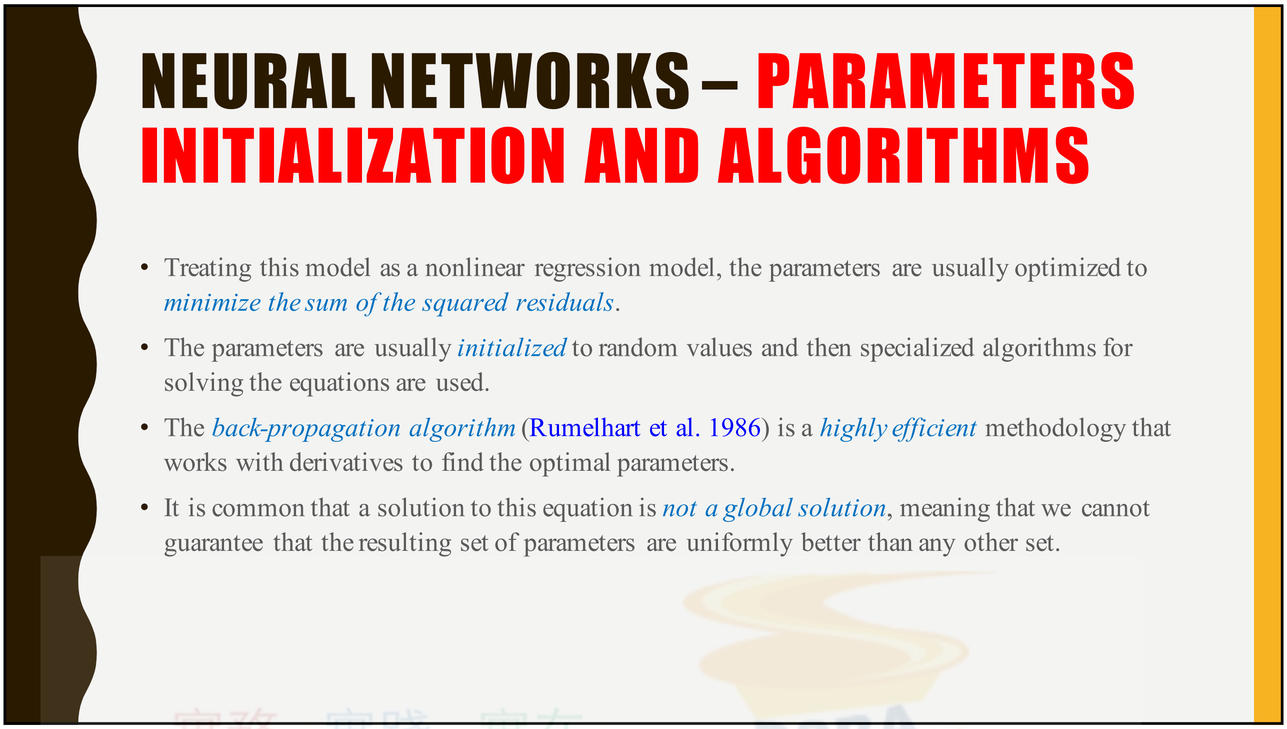
- The β coefficients are similar to *regression coefficients*; coefficient β_{jk} is the *effect* of the *jth predictor* on the *kth hidden unit*.
- Unlike the linear combinations in PLS, there are *NO constraints* that help define these linear combinations.
- Because of this, there is *little likelihood* that the coefficients in each unit represent some *coherent piece of information*.

NEURAL NETWORKS – OVERWHELMING PARAMETERS ESTIMATION

- Once the number of hidden units is defined, each unit must be related to the outcome. *Another linear combination* connects the *hidden* units *to* the *outcome*:

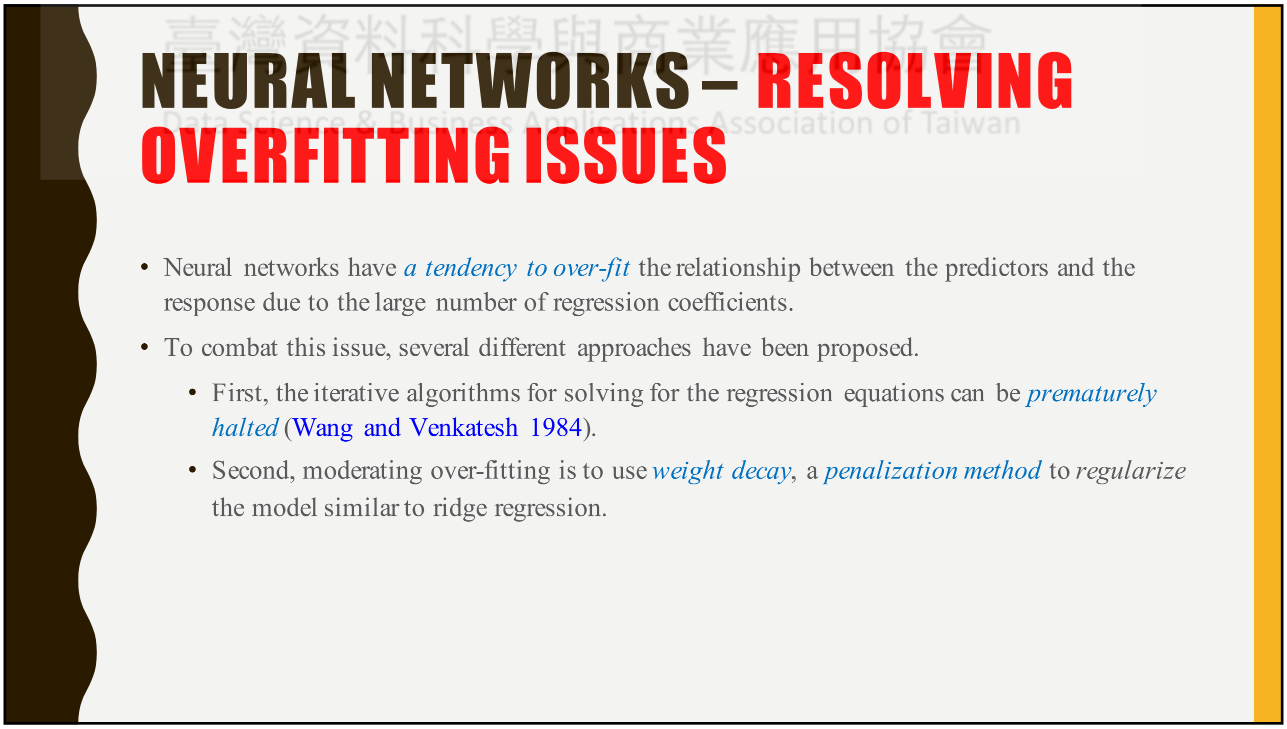
$$f(\mathbf{x}) = \gamma_0 + \sum_{k=1}^{H} \gamma_k h_k$$

- For this type of network model and *P* predictors, there are a total of H(P + 1) + H + 1 total parameters being estimated, which *quickly becomes large as P increases*. *(Overwhelming as P increases)*
- For the solubility data, recall that there are 228 predictors. A neural network model with three hidden units would estimate 691 parameters (3*(228+1)+3+1) while a model with five hidden units would have 1,151 coefficients (5*(228+1)+5+1).



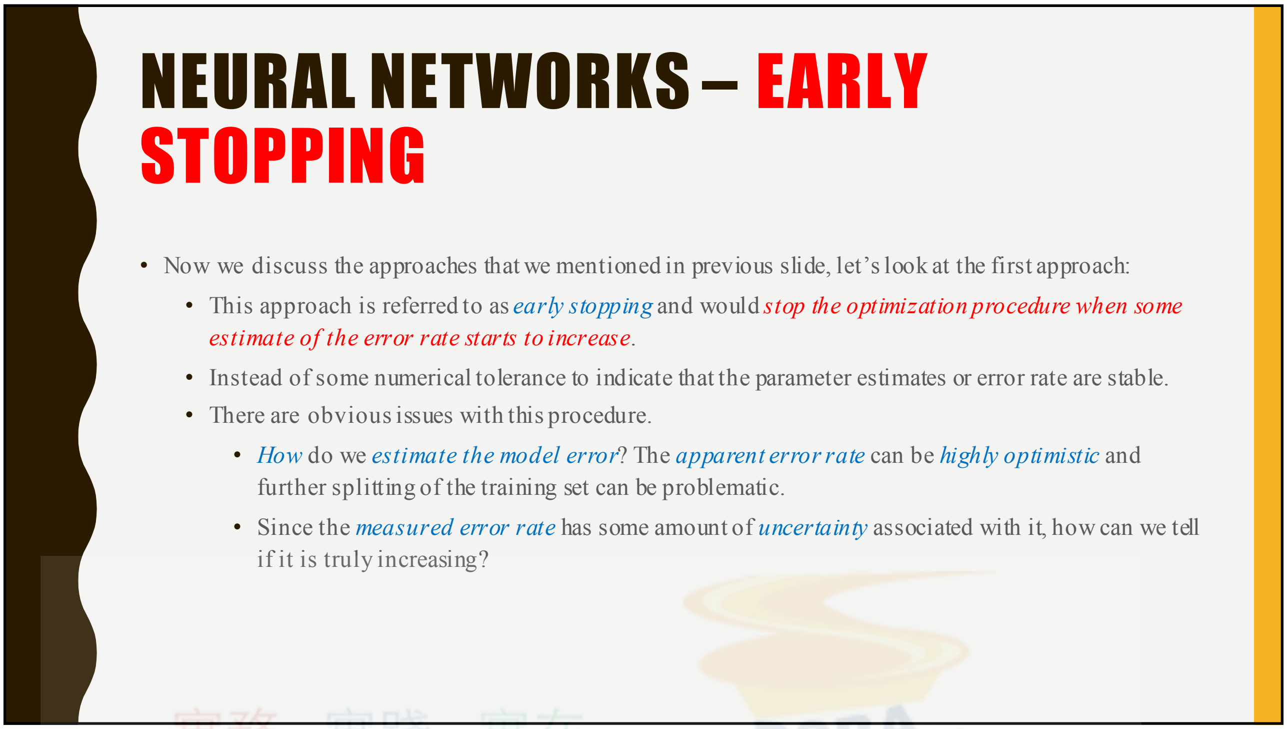
NEURAL NETWORKS – PARAMETERS INITIALIZATION AND ALGORITHMS

- Treating this model as a nonlinear regression model, the parameters are usually optimized to *minimize the sum of the squared residuals*.
- The parameters are usually *initialized* to random values and then specialized algorithms for solving the equations are used.
- The *back-propagation algorithm* (Rumelhart et al. 1986) is a *highly efficient* methodology that works with derivatives to find the optimal parameters.
- 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.



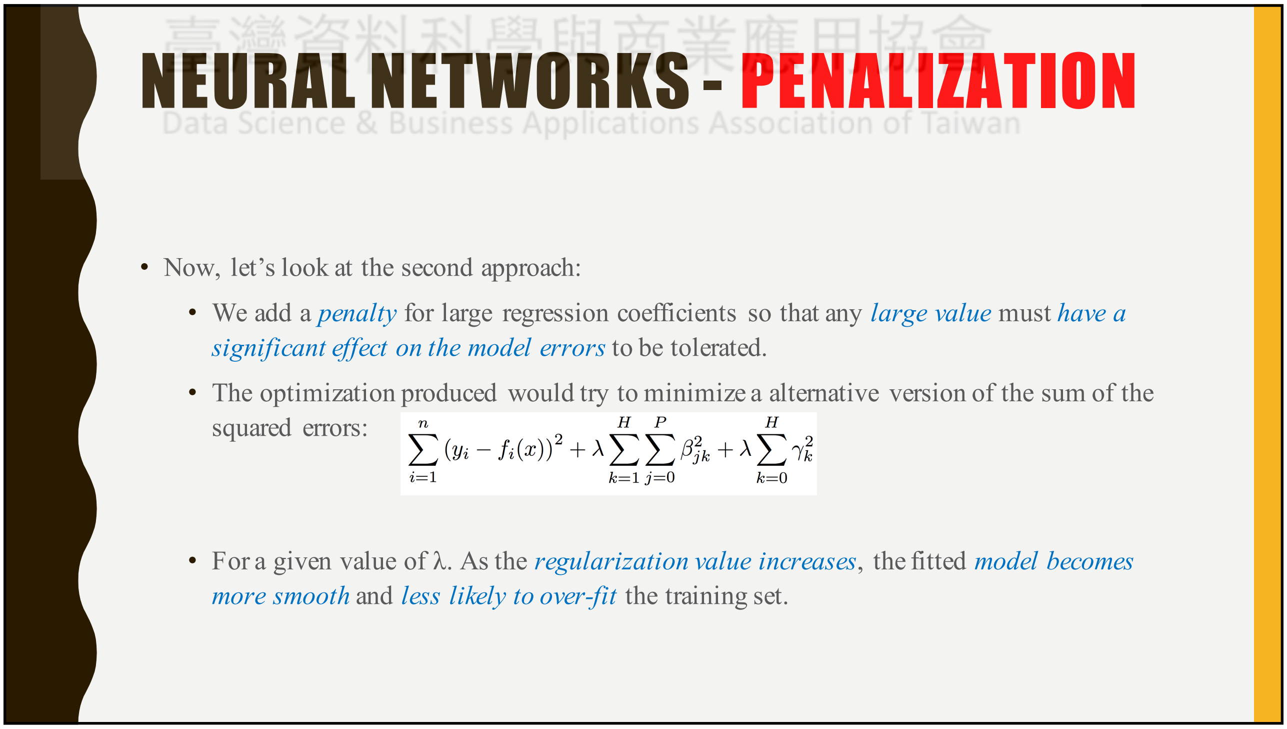
NEURAL NETWORKS – RESOLVING OVERFITTING ISSUES

- 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, several different approaches have been proposed.
  - First, the iterative algorithms for solving for the regression equations can be *prematurely halted* (Wang and Venkatesh 1984).
  - Second, moderating over-fitting is to use *weight decay*, a *penalization method* to *regularize* the model similar to ridge regression.



NEURAL NETWORKS – EARLY STOPPING

- Now we discuss the approaches that we mentioned in previous slide, let's look at the first approach:
  - This approach is referred to as *early stopping* and would *stop the optimization procedure when some estimate of the error rate starts to increase*.
  - Instead of some numerical tolerance to indicate that the parameter estimates or error rate are stable.
  - There are obvious issues with this procedure.
    - *How* do we *estimate the model error*? The *apparent error rate* can be *highly optimistic* and further splitting of the training set can be problematic.
    - Since the *measured error rate* has some amount of *uncertainty* associated with it, how can we tell if it is truly increasing?

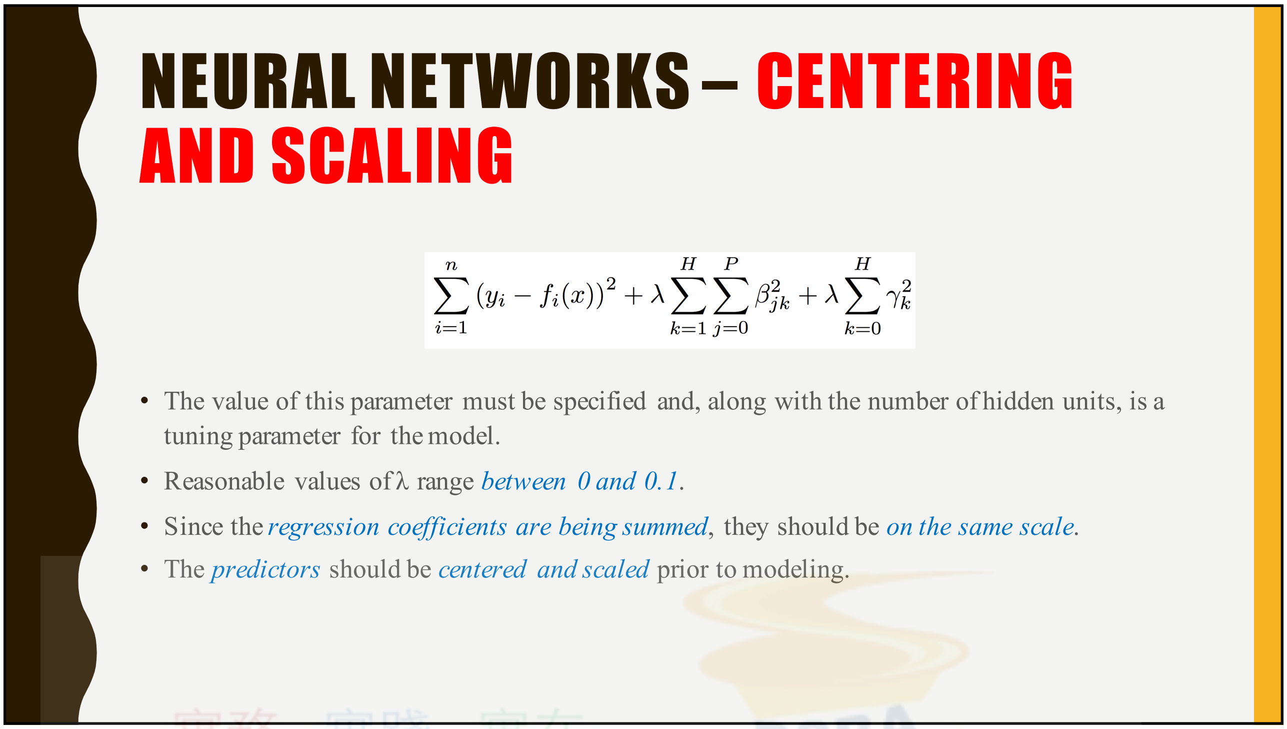


NEURAL NETWORKS - PENALIZATION

- Now, let's look at the second approach:
  - We add a *penalty* for large regression coefficients so that any *large value* must *have a significant effect on the model errors* to be tolerated.
  - The optimization produced would try to minimize a alternative version of the sum of the squared errors:

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

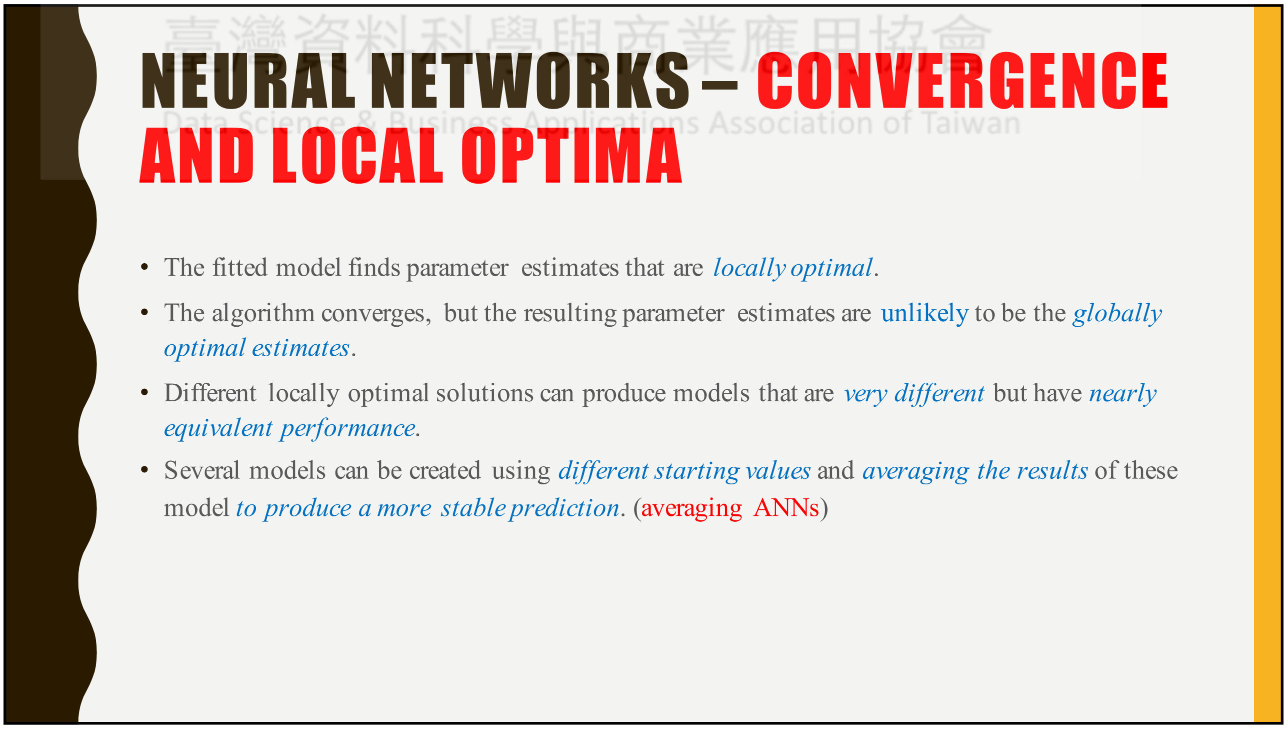
  - For a given value of λ. As the *regularization value increases*, the fitted *model becomes more smooth* and *less likely to over-fit* the training set.



NEURAL NETWORKS – CENTERING AND SCALING

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

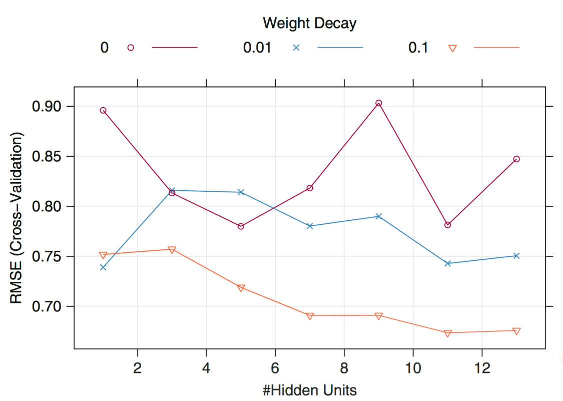
- The value of this parameter must be specified and, along with the number of hidden units, is a tuning parameter for the model.
- Reasonable values of λ range *between 0 and 0.1*.
- Since the *regression coefficients are being summed*, they should be *on the same scale*.
- The *predictors* should be *centered and scaled* prior to modeling.



NEURAL NETWORKS – CONVERGENCE AND LOCAL OPTIMA

- The fitted model finds parameter estimates that are *locally optimal*.
- The algorithm converges, but the resulting parameter estimates are unlikely to be the *globally optimal estimates*.
- Different locally optimal solutions can produce models that are *very different* but have *nearly equivalent performance*.
- Several models can be created using *different starting values* and *averaging the results* of these model *to produce a more stable prediction*. (averaging ANNs)

NEURAL NETWORKS – PARAMETERS TUNING

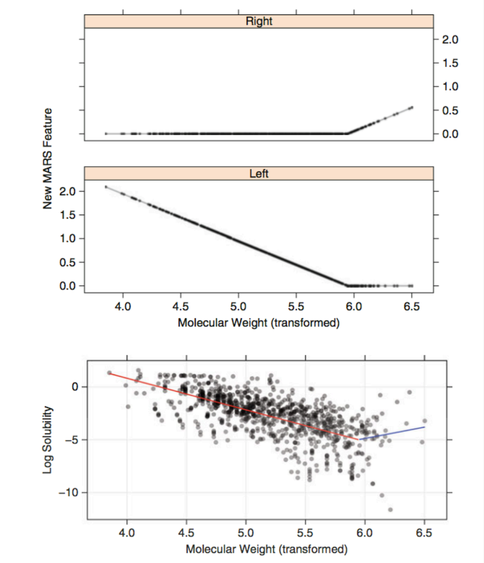


- Three different weight decay values were evaluated (λ = 0.00, 0.01, 0.10) along with a *single hidden layer* with *sizes ranging between 1 and 13* hidden units.
- The cross-validated RMSE profiles of these models are displayed in this figure.
- The optimal model used *11* hidden units with a total of *2,531* coefficients.
- The performance of the model is fairly stable for a *high degree of regularization* (i.e., λ = 0.1), so *smaller models* could also be effective for these data.

MULTIVARIATE ADAPTIVE REGRESSION SPLINES – SURROGATE FEATURES AND HINGE FUNCTION

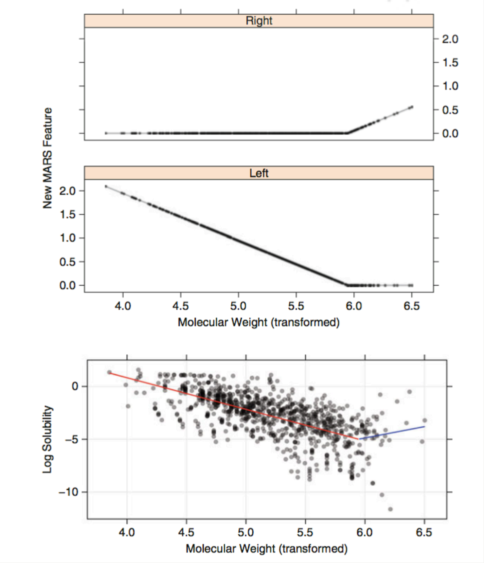
- MARS (Multivariate Adaptive Regression Splines, Friedman 1991) uses *surrogate* features instead of the original predictors.
- MARS creates two contrasted versions of a predictor to enter the model.
- Specifically, given a cut point for a predictor, two new features are “hinge” or “hockey stick” functions of the original.
- This scheme creates a piecewise linear model where each new feature models an isolated portion of the original data.

MULTIVARIATE ADAPTIVE REGRESSION SPLINES

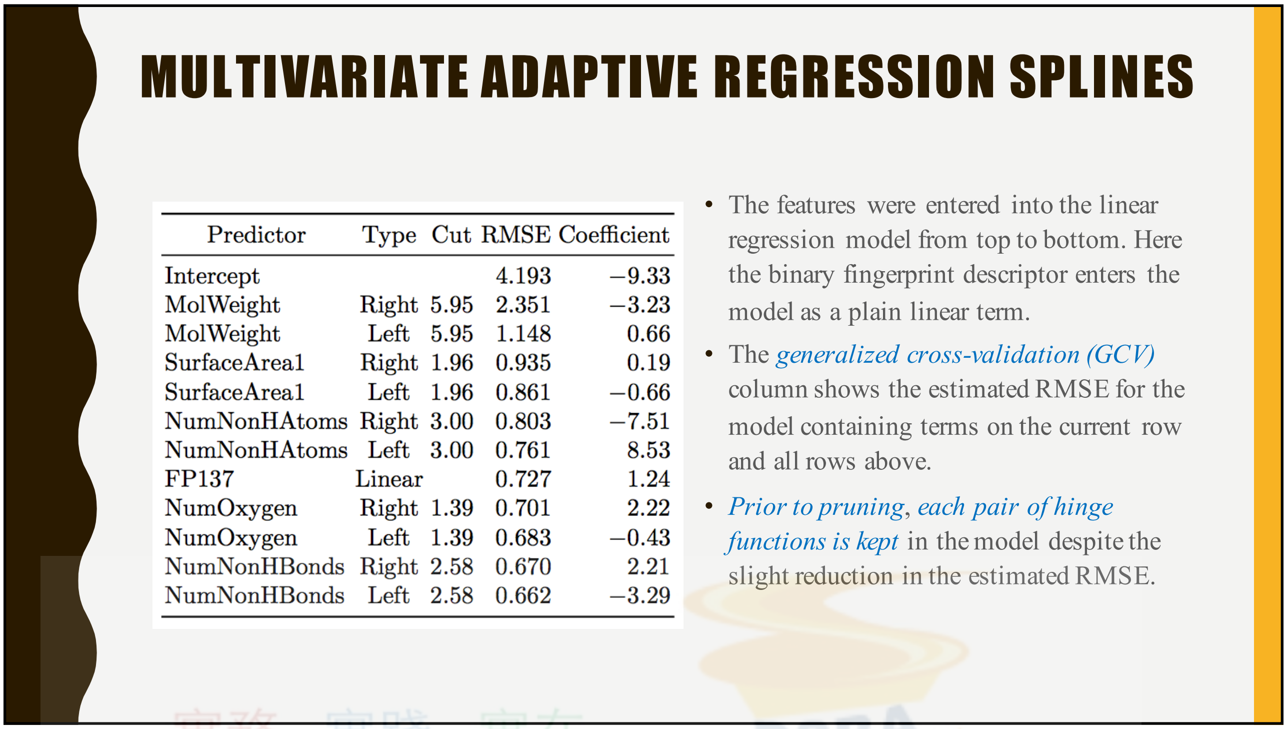


- In the initial search for features in the solubility data, a cut point of 5.9 for molecular weight had the *smallest error rate*.
- The resulting artificial predictors are shown in the top two panels of this figure.
- One predictor has all values less than the cut point set to zero and values greater than the cut point are left unchanged.
- The second feature is the mirror image of the first. Instead of the original data, these two new predictors are used to predict the outcome in a linear regression model.

MULTIVARIATE ADAPTIVE REGRESSION SPLINES



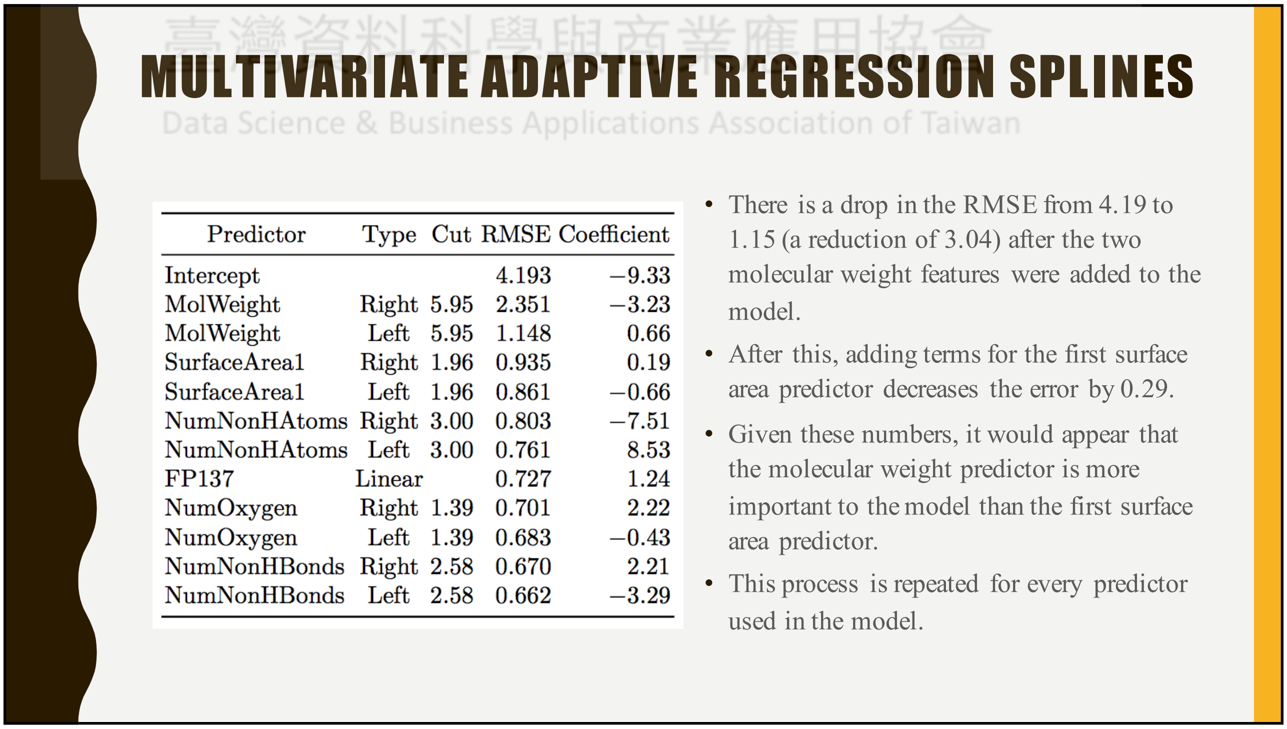
- The bottom panel of this figure shows the result of the linear regression with the two new features and the piecewise nature of the relationship.
- The “left-hand” feature is associated with a negative slope when the molecular weight is less than 5.9 while the “right-hand” feature estimates a positive slope for larger values of the predictor.
- The second term in this equation is associated with the right-hand feature shown in this figure while the last component of the equation is the left-hand feature.



MULTIVARIATE ADAPTIVE REGRESSION SPLINES

Predictor	Type	Cut	RMSE	Coefficient
Intercept			4.193	−9.33
MolWeight	Right	5.95	2.351	−3.23
MolWeight	Left	5.95	1.148	0.66
SurfaceArea1	Right	1.96	0.935	0.19
SurfaceArea1	Left	1.96	0.861	−0.66
NumNonHAtoms	Right	3.00	0.803	−7.51
NumNonHAtoms	Left	3.00	0.761	8.53
FP137	Linear		0.727	1.24
NumOxygen	Right	1.39	0.701	2.22
NumOxygen	Left	1.39	0.683	−0.43
NumNonHBonds	Right	2.58	0.670	2.21
NumNonHBonds	Left	2.58	0.662	−3.29

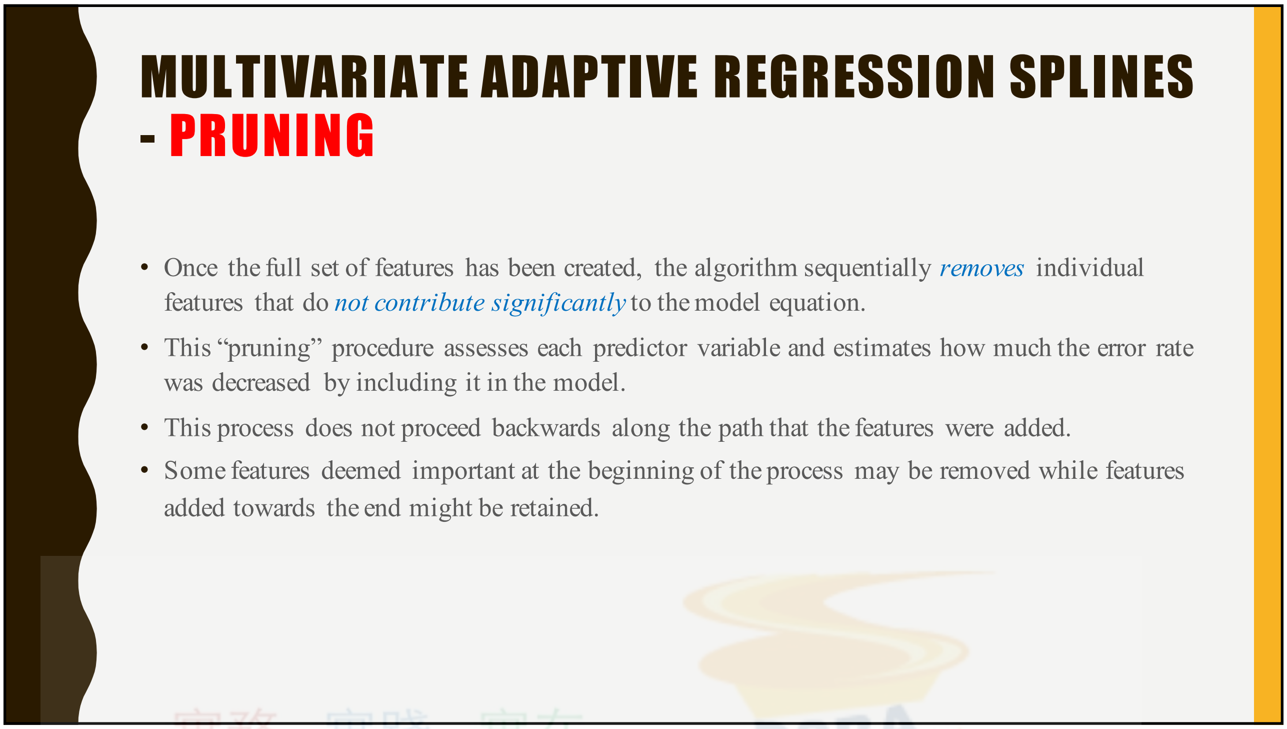
- The features were entered into the linear regression model from top to bottom. Here the binary fingerprint descriptor enters the model as a plain linear term.
- The *generalized cross-validation (GCV)* column shows the estimated RMSE for the model containing terms on the current row and all rows above.
- *Prior to pruning, each pair of hinge functions is kept* in the model despite the slight reduction in the estimated RMSE.



MULTIVARIATE ADAPTIVE REGRESSION SPLINES

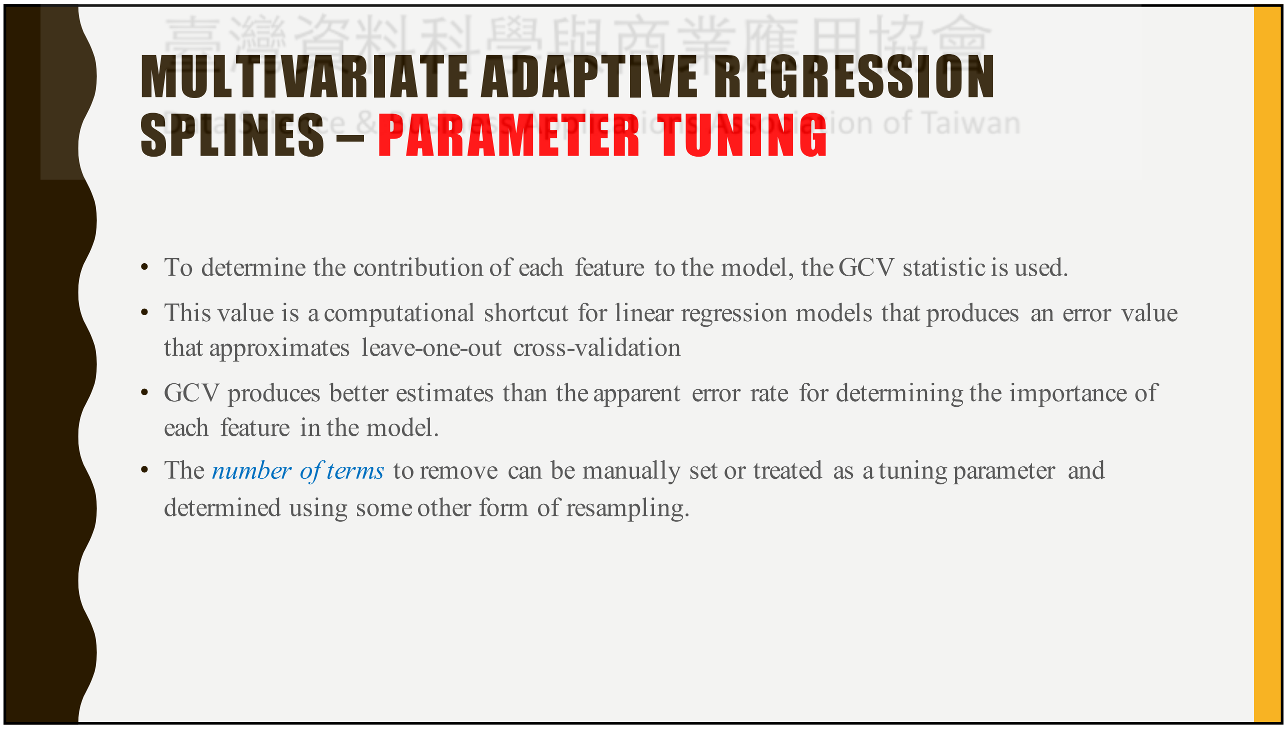
Predictor	Type	Cut	RMSE	Coefficient
Intercept			4.193	−9.33
MolWeight	Right	5.95	2.351	−3.23
MolWeight	Left	5.95	1.148	0.66
SurfaceArea1	Right	1.96	0.935	0.19
SurfaceArea1	Left	1.96	0.861	−0.66
NumNonHAtoms	Right	3.00	0.803	−7.51
NumNonHAtoms	Left	3.00	0.761	8.53
FP137	Linear		0.727	1.24
NumOxygen	Right	1.39	0.701	2.22
NumOxygen	Left	1.39	0.683	−0.43
NumNonHBonds	Right	2.58	0.670	2.21
NumNonHBonds	Left	2.58	0.662	−3.29

- There is a drop in the RMSE from 4.19 to 1.15 (a reduction of 3.04) after the two molecular weight features were added to the model.
- After this, adding terms for the first surface area predictor decreases the error by 0.29.
- Given these numbers, it would appear that the molecular weight predictor is more important to the model than the first surface area predictor.
- This process is repeated for every predictor used in the model.



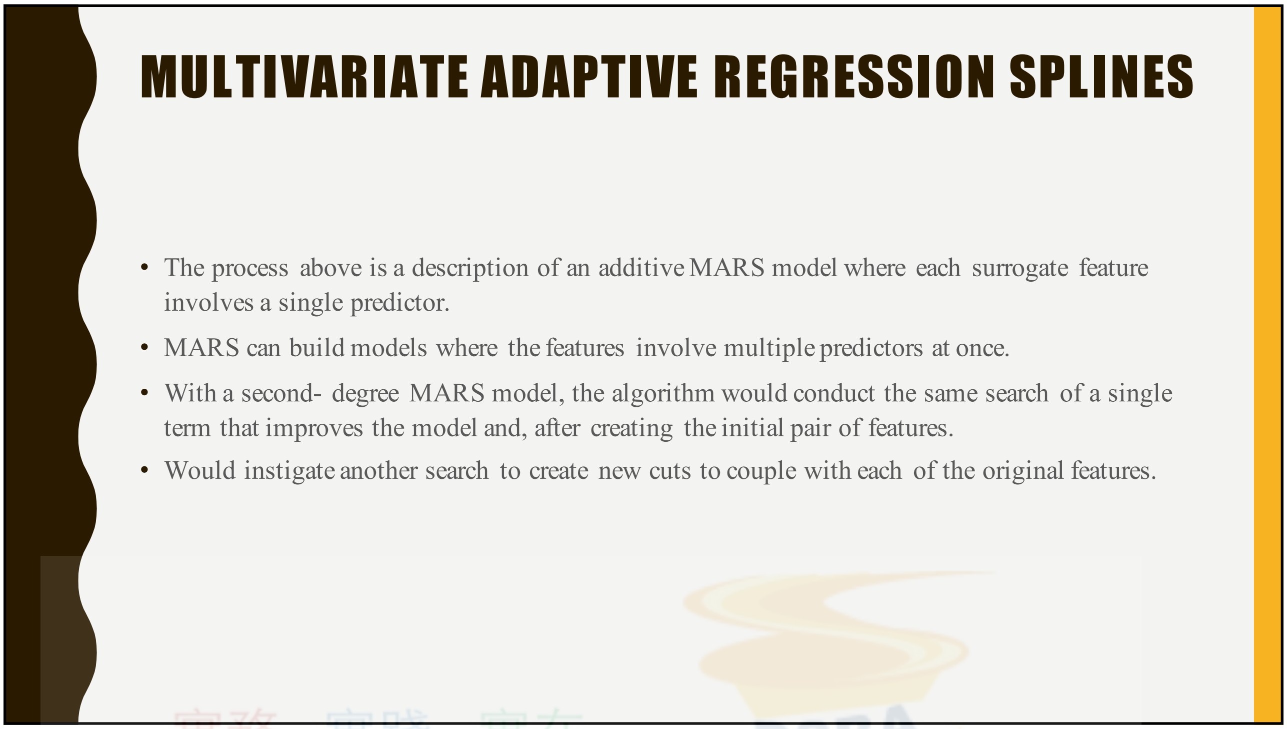
MULTIVARIATE ADAPTIVE REGRESSION SPLINES - PRUNING

- 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.



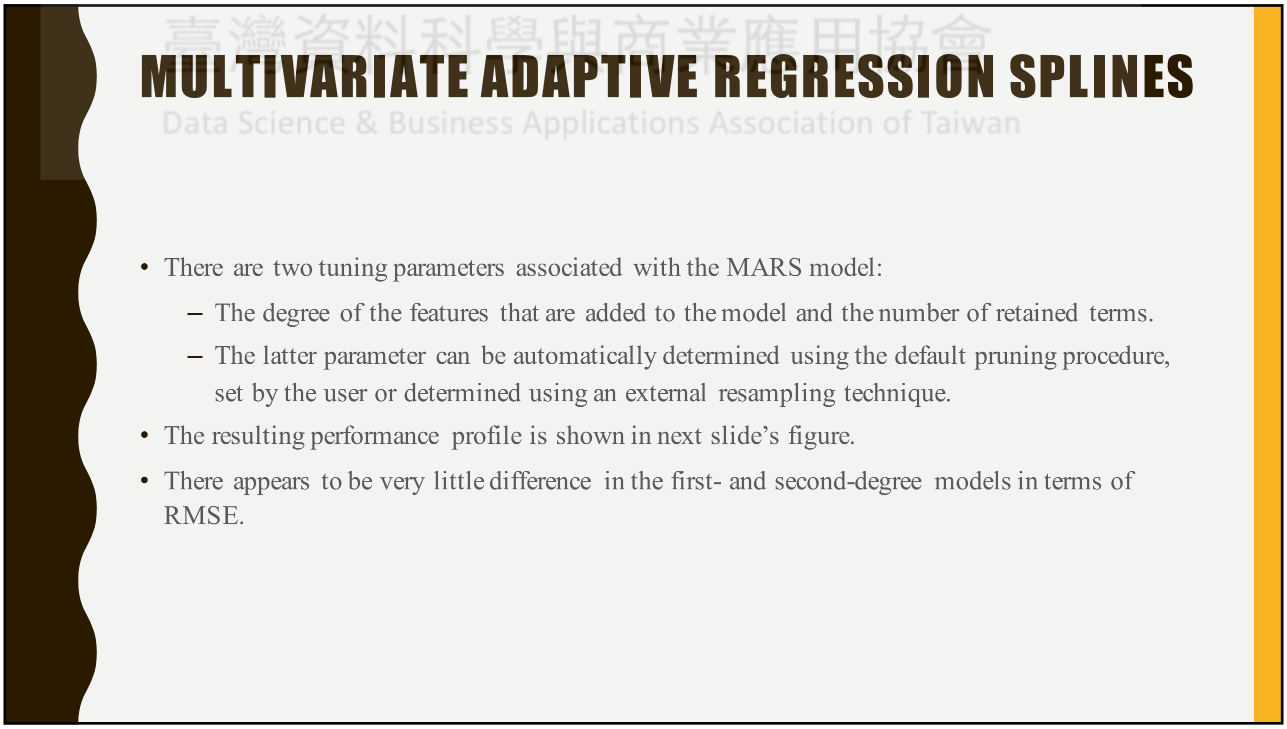
MULTIVARIATE ADAPTIVE REGRESSION SPLINES – PARAMETER TUNING

- To determine the contribution of each feature to the model, the GCV statistic is used.
- This value is a computational shortcut for linear regression models that produces an error value that approximates leave-one-out cross-validation
- GCV produces better estimates 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 a tuning parameter and determined using some other form of resampling.



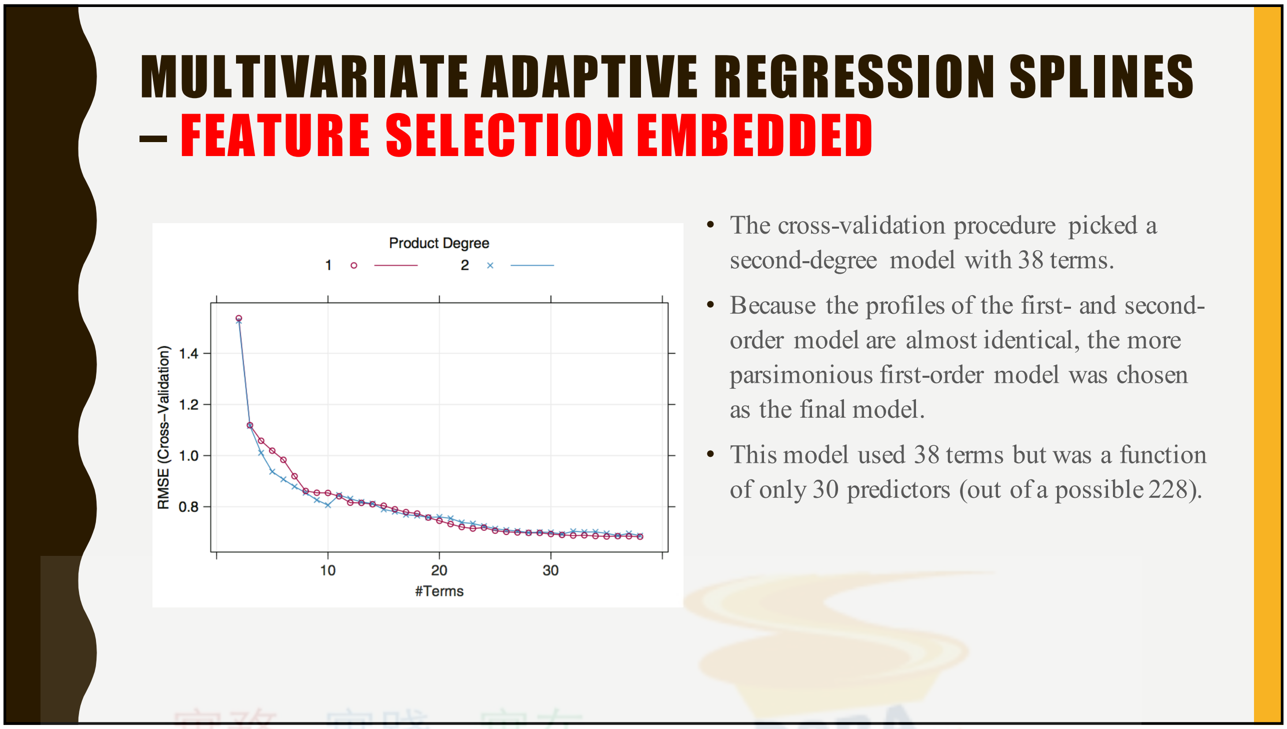
MULTIVARIATE ADAPTIVE REGRESSION SPLINES

- The process above is a description of an additive MARS model where each surrogate feature involves a single predictor.
- MARS can build models where the features involve multiple predictors at once.
- With a second- degree MARS model, the algorithm would conduct the same search of a single term that improves the model and, after creating the initial pair of features.
- Would instigate another search to create new cuts to couple with each of the original features.



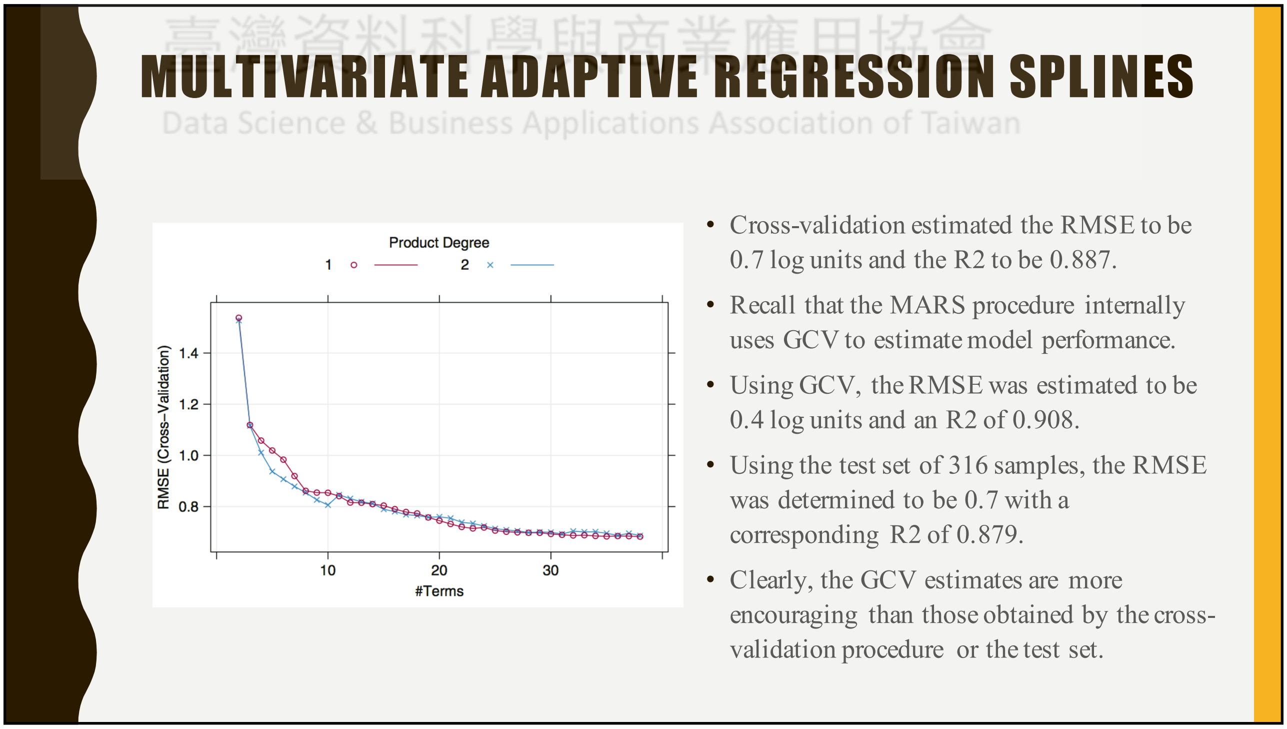
MULTIVARIATE ADAPTIVE REGRESSION SPLINES

- There are two tuning parameters associated with the MARS model:
  - The degree of the features that are added to the model and the number of retained terms.
  - The latter parameter can be automatically determined using the default pruning procedure, set by the user or determined using an external resampling technique.
- The resulting performance profile is shown in next slide’s figure.
- There appears to be very little difference in the first- and second-degree models in terms of RMSE.



MULTIVARIATE ADAPTIVE REGRESSION SPLINES – FEATURE SELECTION EMBEDDED

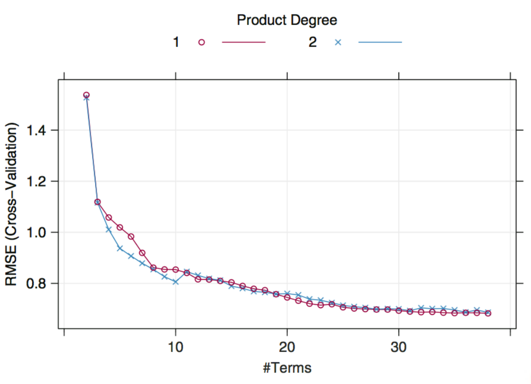
- The cross-validation procedure picked a second-degree model with 38 terms.
- Because the profiles of the first- and second-order model are almost identical, the more parsimonious first-order model was chosen as the final model.
- This model used 38 terms but was a function of only 30 predictors (out of a possible 228).



MULTIVARIATE ADAPTIVE REGRESSION SPLINES

- Cross-validation estimated the RMSE to be 0.7 log units and the R2 to be 0.887.
- Recall that the MARS procedure internally uses GCV to estimate model performance.
- Using GCV, the RMSE was estimated to be 0.4 log units and an R2 of 0.908.
- Using the test set of 316 samples, the RMSE was determined to be 0.7 with a corresponding R2 of 0.879.
- Clearly, the GCV estimates are more encouraging than those obtained by the cross-validation procedure or the test set.

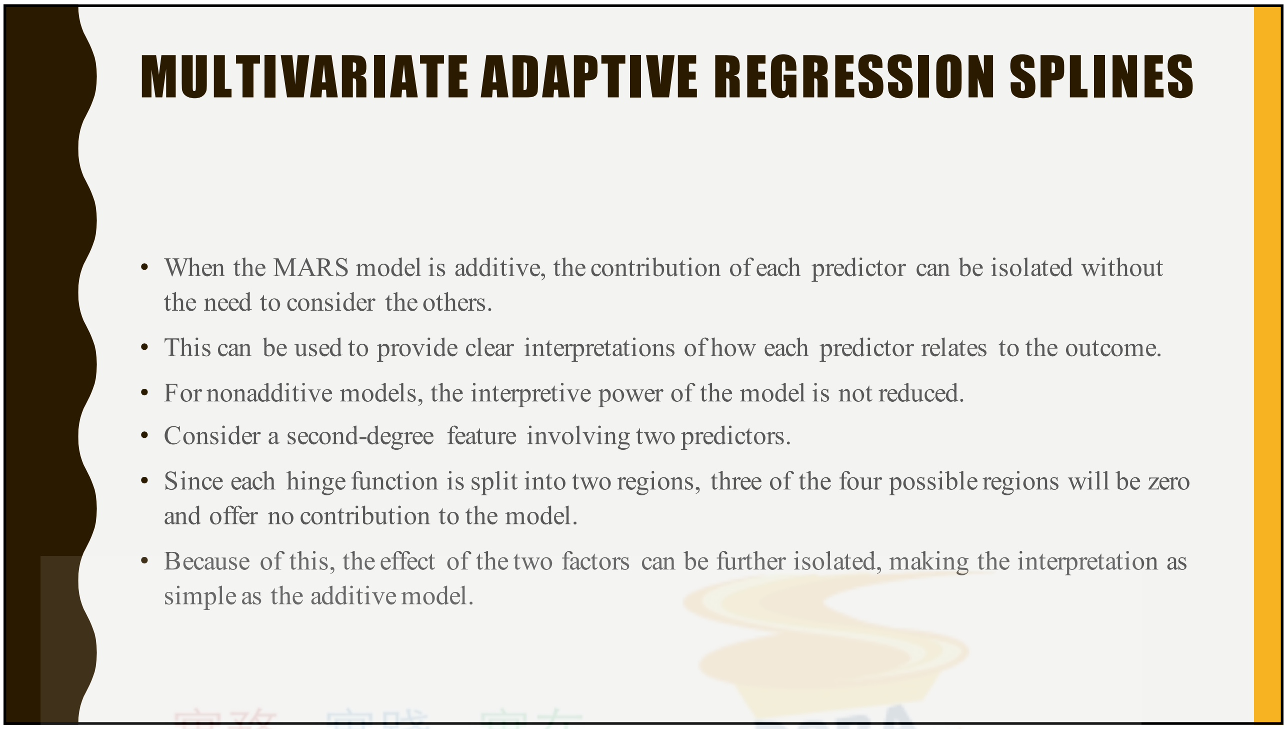
MULTIVARIATE ADAPTIVE REGRESSION SPLINES



- The internal GCV estimate that MARS employs evaluates an individual model while the external cross- validation procedure is exposed to the variation in the entire model building process.
- Including feature selection. Since the GCV estimate does not reflect the uncertainty from feature selection, it suffers from selection bias.

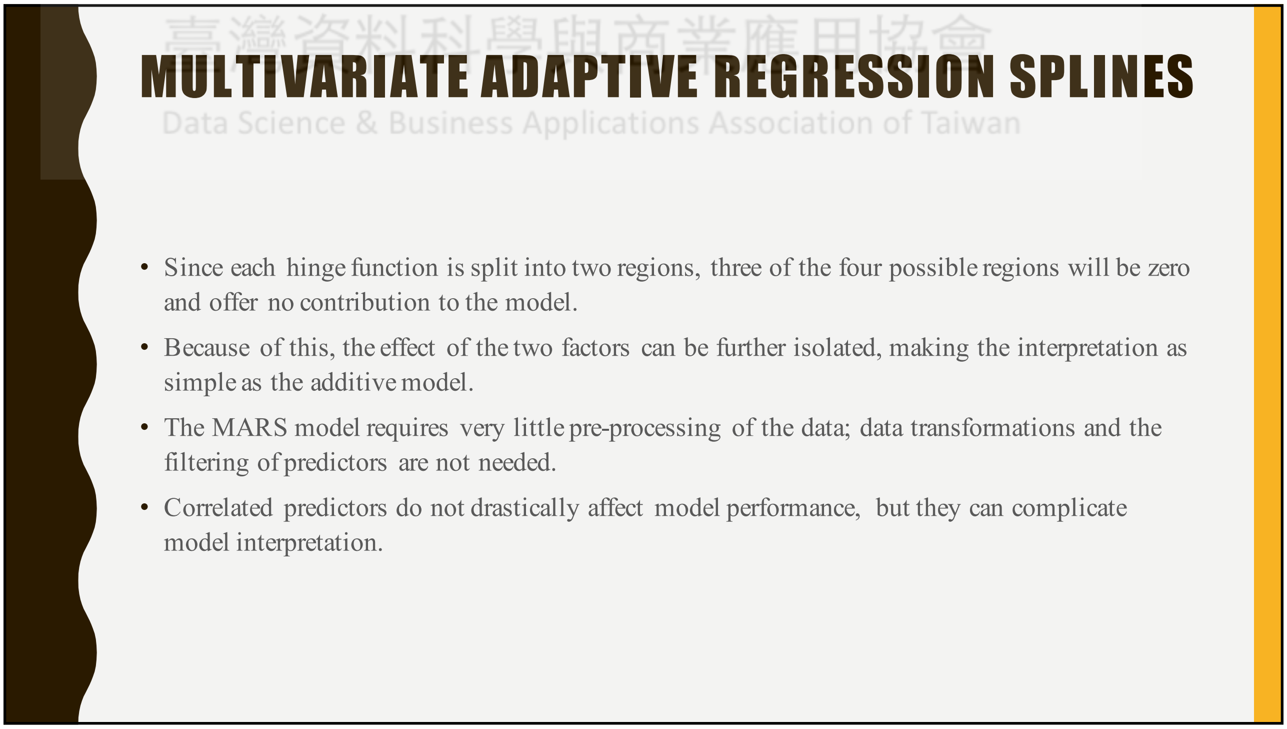
MULTIVARIATE ADAPTIVE REGRESSION SPLINES - ADVANTAGES

- There are several advantages to using MARS.
  - The model *automatically conducts feature selection*.
  - The model equation is independent of predictor variables that are not involved with any of the final model features.
  - This point cannot be underrated.



MULTIVARIATE ADAPTIVE REGRESSION SPLINES

- When the MARS model is additive, the contribution of each predictor can be isolated without the need to consider the others.
- This can be used to provide clear interpretations of how each predictor relates to the outcome.
- For nonadditive models, the interpretive power of the model is not reduced.
- Consider a second-degree feature involving two predictors.
- Since each hinge function is split into two regions, three of the four possible regions will be zero and offer no contribution to the model.
- Because of this, the effect of the two factors can be further isolated, making the interpretation as simple as the additive model.



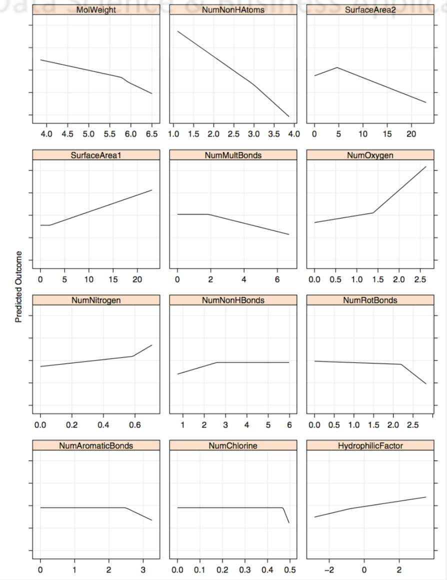
MULTIVARIATE ADAPTIVE REGRESSION SPLINES

- Since each hinge function is split into two regions, three of the four possible regions will be zero and offer no contribution to the model.
- Because of this, the effect of the two factors can be further isolated, making the interpretation as simple as the additive model.
- The MARS model requires very little pre-processing of the data; data transformations and the filtering of predictors are not needed.
- Correlated predictors do not drastically affect model performance, but they can complicate model interpretation.

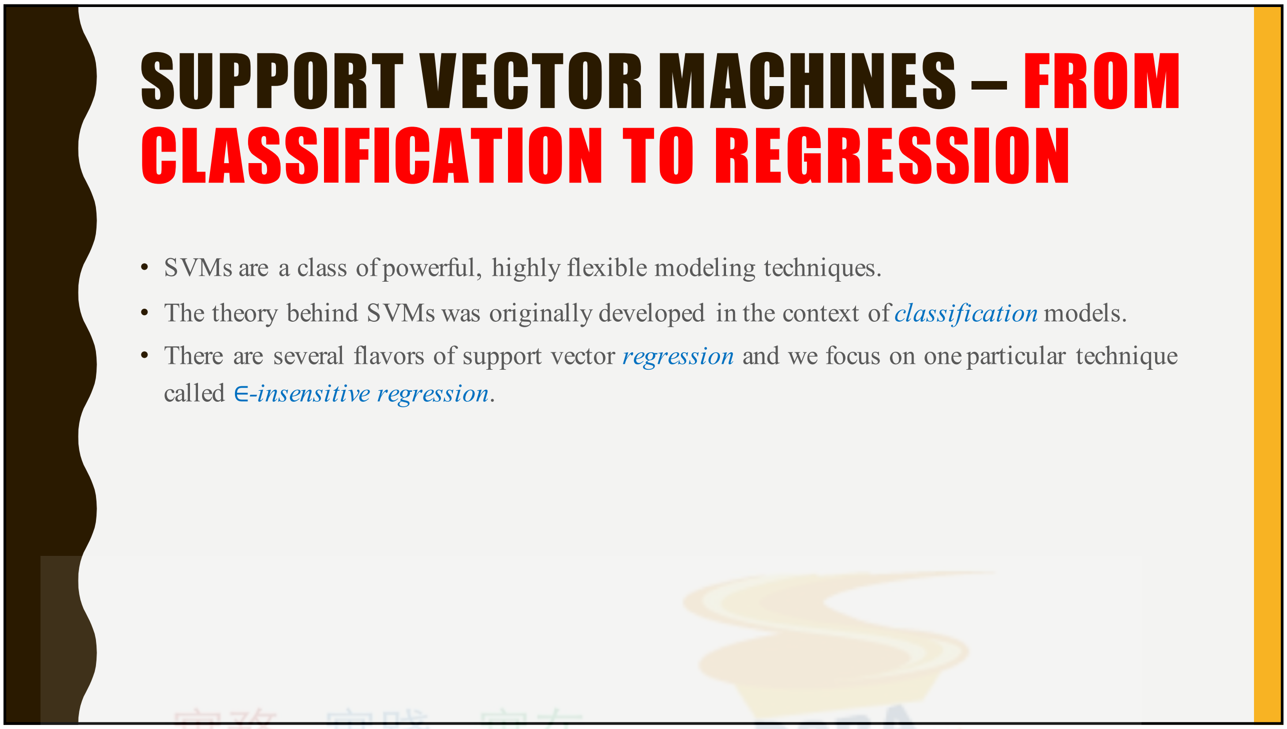
MULTIVARIATE ADAPTIVE REGRESSION SPLINES

- The training set contained two predictors that were nearly perfectly correlated.
- Since MARS can select a predictor more than once during the iterations, the choice of which predictor is used in the feature is essentially random.
- In this case, the model interpretation is hampered by two redundant pieces of information that show up in different parts of the model under different names.
- Another method to help understand the nature of how the predictors affect the model is to quantify their *importance* to the model.
- These improvements in the model can be aggregated for each predictor as a relative measure of the impact on the model.

MULTIVARIATE ADAPTIVE REGRESSION SPLINES

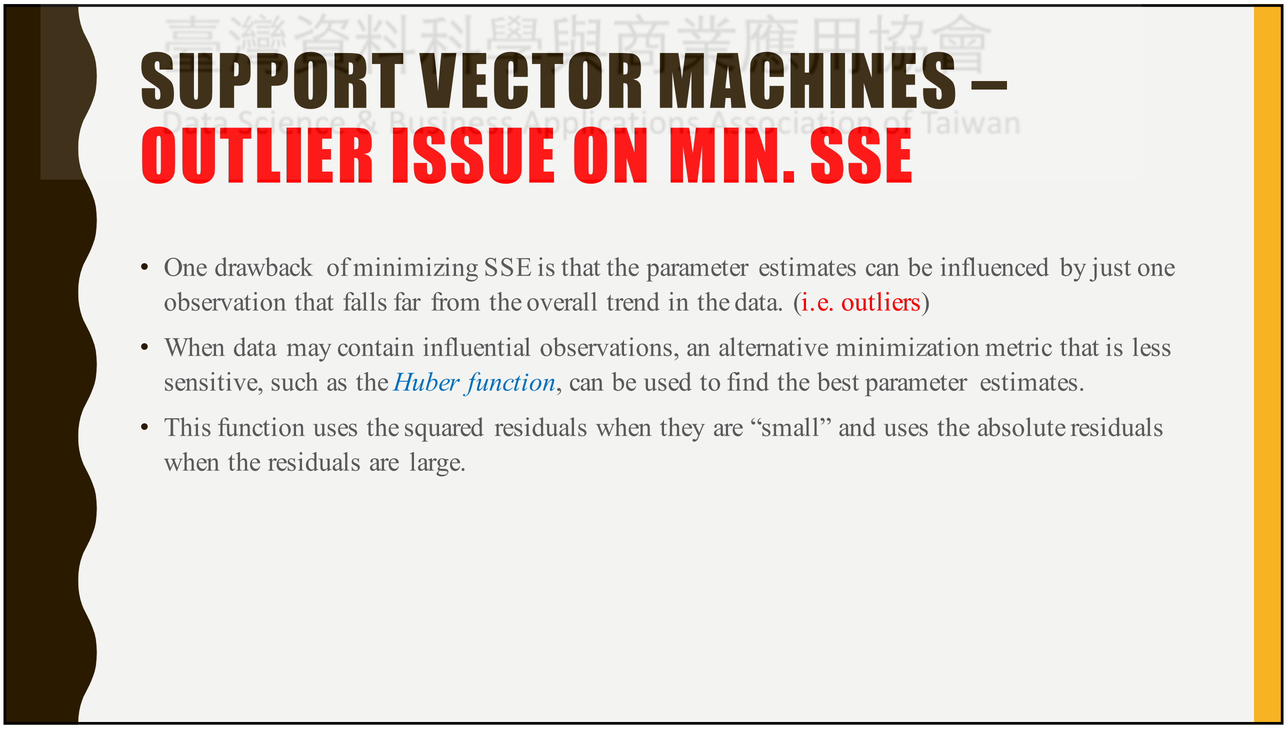


- For each panel, the line represents the prediction profile for that variable when all the others are held constant at their mean level.
- The *additive* nature of the model allows each predictor to be viewed in isolation.
- Changing the values of the other predictor variables will not alter the shape of the profile, only the location on the *y*-axis where the profile starts.



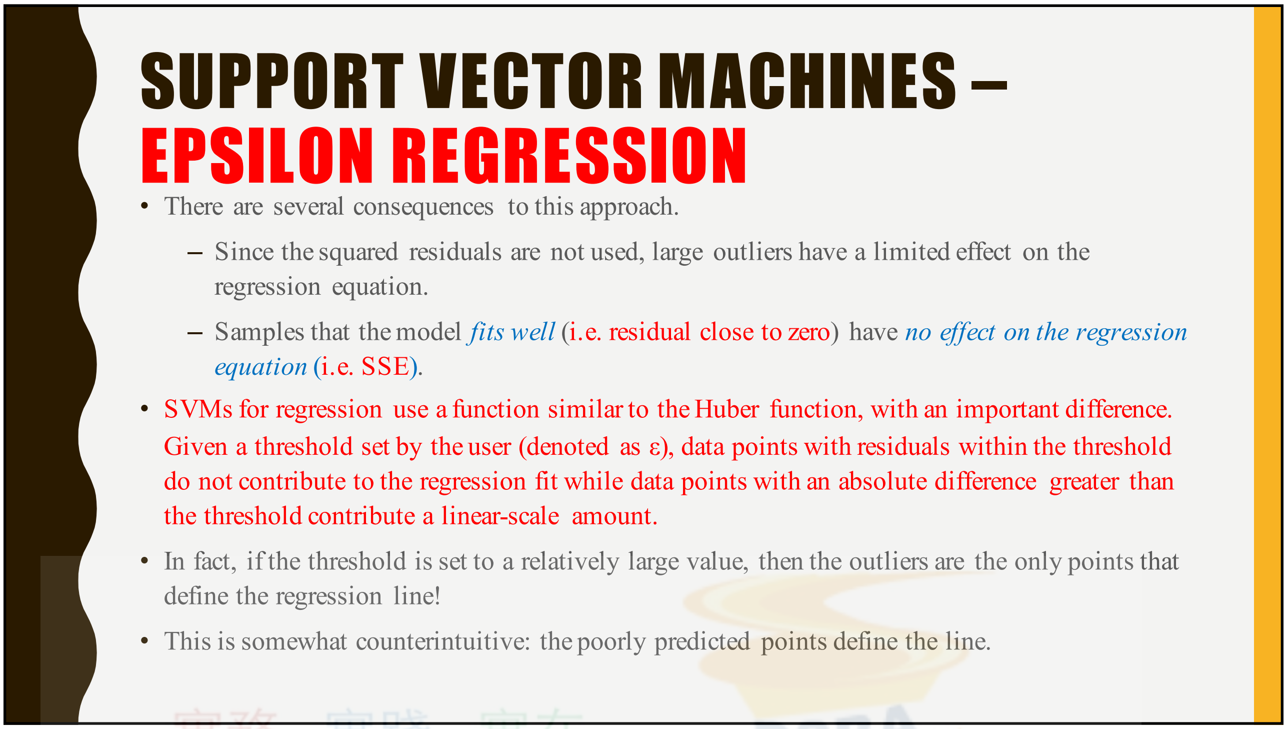
SUPPORT VECTOR MACHINES – FROM CLASSIFICATION TO REGRESSION

- SVMs are a class of powerful, highly flexible modeling techniques.
- The theory behind SVMs was originally developed in the context of *classification* models.
- There are several flavors of support vector *regression* and we focus on one particular technique called ∈*-insensitive regression*.



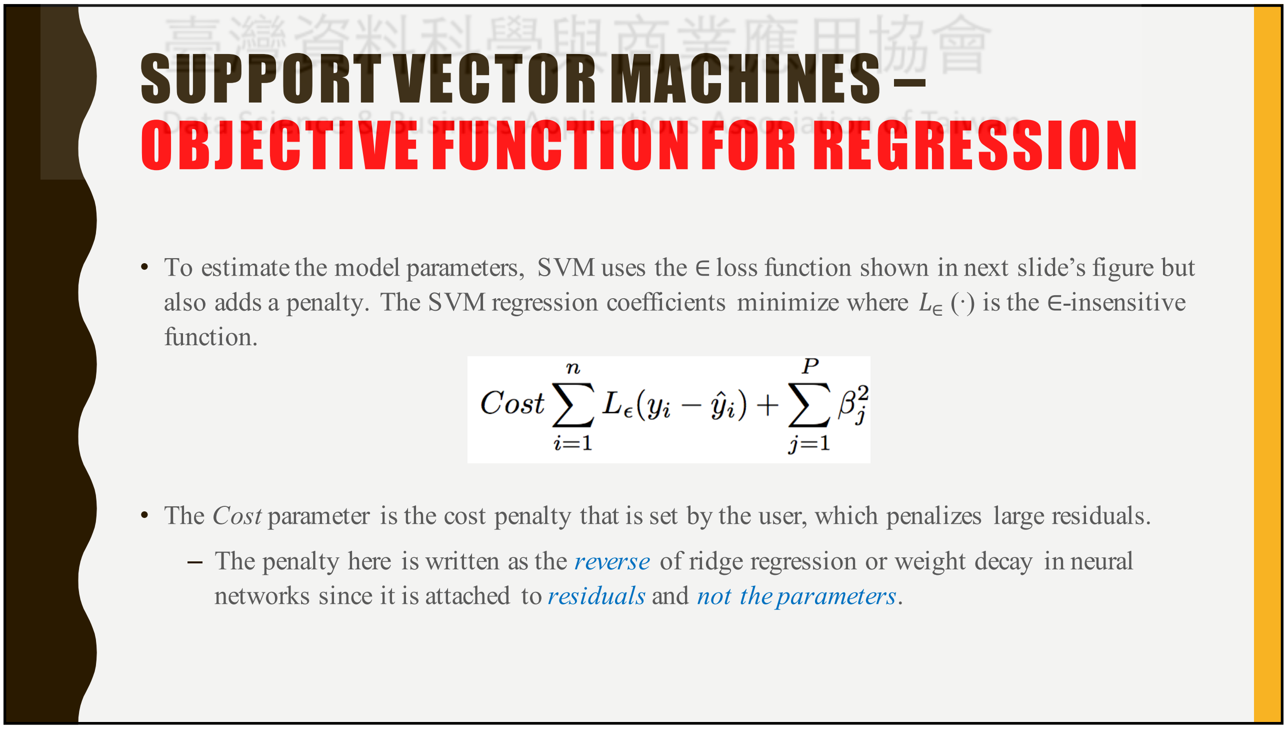
SUPPORT VECTOR MACHINES – OUTLIER ISSUE ON MIN. SSE

- One drawback of minimizing SSE is that the parameter estimates can be influenced by just one observation that falls far from the overall trend in the data. (i.e. outliers)
- When data may contain influential observations, an alternative minimization metric that is less sensitive, 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.



SUPPORT VECTOR MACHINES – EPSILON REGRESSION

- There are several consequences to this approach.
  - Since the squared residuals are not used, large outliers have a limited effect on the regression equation.
  - Samples that the model *fits well* (i.e. residual close to zero) have *no effect on the regression equation* (i.e. SSE).
- SVMs for regression use a function similar 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.
- In fact, if the threshold is set to a relatively large value, then the outliers are the only points that define the regression line!
- This is somewhat counterintuitive: the poorly predicted points define the line.



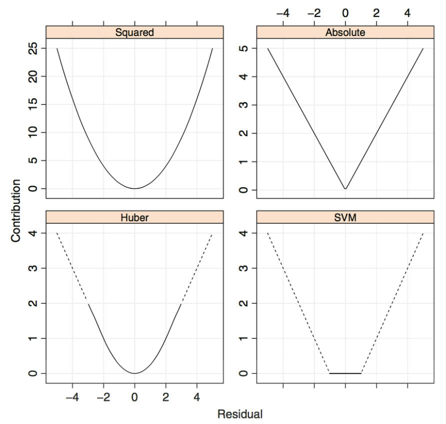
SUPPORT VECTOR MACHINES – OBJECTIVE FUNCTION FOR REGRESSION

- To estimate the model parameters, SVM uses the ∈ loss function shown in next slide's figure but also adds a penalty. The SVM regression coefficients minimize where $L_{\in}$ (·) is the ∈-insensitive function.

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

- The *Cost* parameter is the cost penalty that is set by the user, which penalizes large residuals.
  - The penalty here is written as the *reverse* of ridge regression or weight decay in neural networks since it is attached to *residuals* and *not the parameters*.

SUPPORT VECTOR MACHINES – HUBER FUNCTION



- 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 ∈= 1 was used.
- The *y*-axis scales are different to make the figures easier to read

SUPPORT VECTOR MACHINES

- Recall that the simple linear regression model predicted new samples using linear combinations of the data and parameters.
- For a new sample, *u*, the prediction equation is

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

SUPPORT VECTOR MACHINES

- The *linear support vector machine* prediction function is very similar.
- The parameter estimates can be written as functions of *a set of unknown parameters (α_i)* and *the training set data points* so that

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

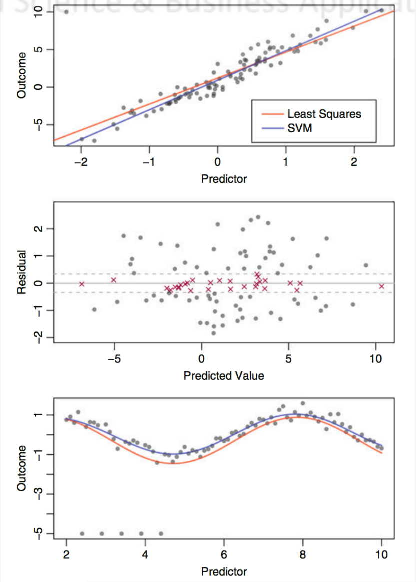
SUPPORT VECTOR MACHINES – OVER-PARAMETERIZED & REGULARIZATION

- There are several aspects of this equation worth pointing out.
  1. First, there are *as many α parameters as there are data points*.
     - From the stand-point of classical regression modeling, this model would be considered *over-parameterized*; typically.
     - It is better to estimate fewer parameters than data points.
     - The use of the *cost* value *effectively regularizes the model* to help alleviate this problem.
  2. Second, the *individual training set data points* are *required for new predictions*.
     - When the training set is large, this makes the prediction equations less compact than other techniques.
     - For *some* percentage of the training set samples, the *α_i parameters will be exactly zero*, indicating that they have no impact on the prediction equation.

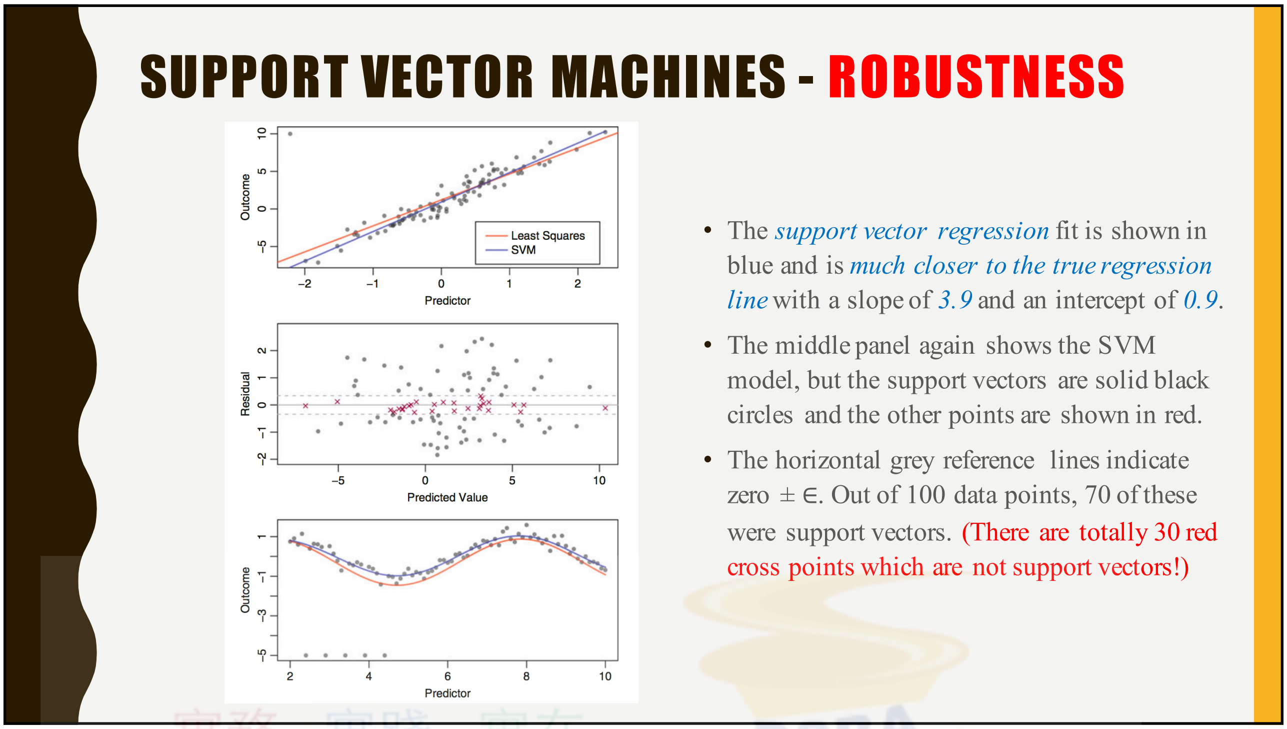
SUPPORT VECTOR MACHINES – SUPPORT VECTORS

- The data points associated with an α_i parameter of zero are the training set samples that are within ±∈ of the regression line.
- *Only a subset of training set data points, where $\alpha \neq 0$*, are needed for prediction.
- Since the regression line is determined using these samples, they are called the *support vectors* as they support the regression line.

SUPPORT VECTOR MACHINES

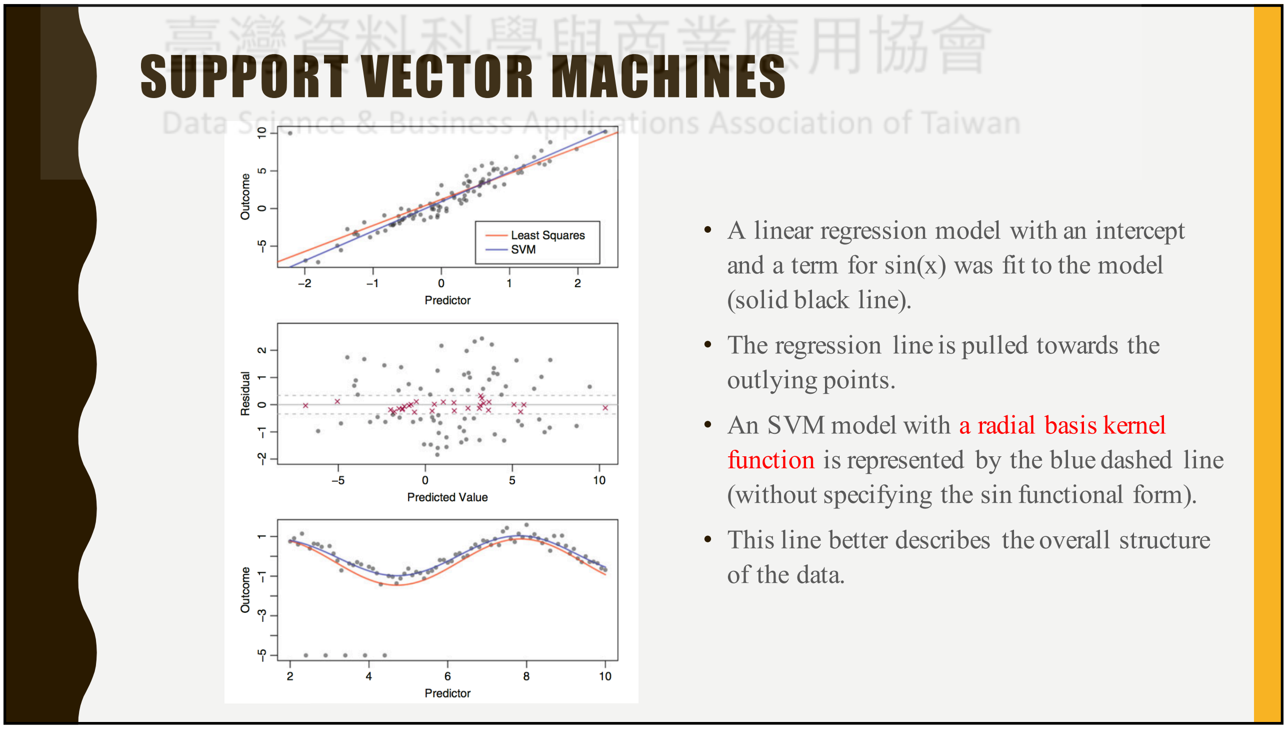


- The top panel shows the model fit for a linear regression model (black solid line).
- A support vector machine regression model (blue dashed line) with ∈ = 0.01.
- The linear regression line is pulled towards this point (attention to the upper left point), resulting in estimates of the slope and intercept of *3.5* and *1.2*, respectively.



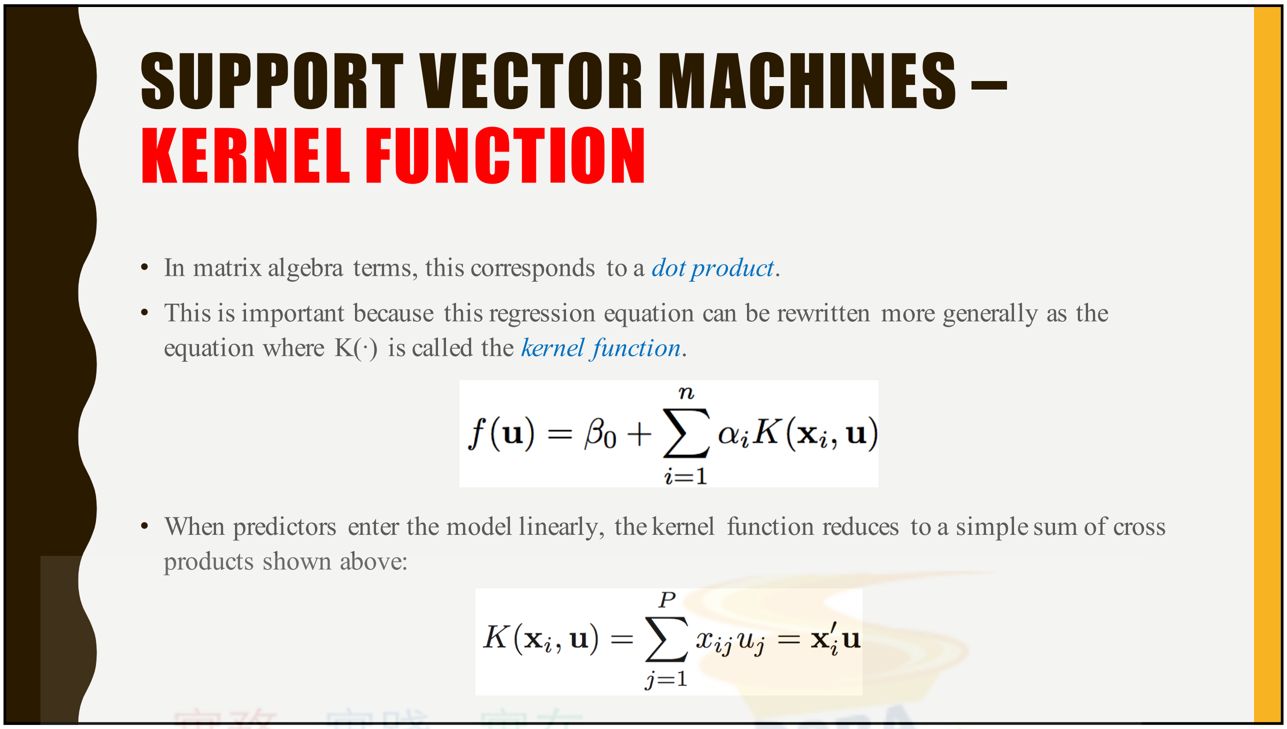
SUPPORT VECTOR MACHINES - ROBUSTNESS

- The *support vector regression* fit is shown in blue and is *much closer to the true regression line* with a slope of *3.9* and an intercept of *0.9*.
- The middle panel again shows the SVM model, but the support vectors are solid black circles and the other points are shown in red.
- The horizontal grey reference lines indicate zero ± ∈. Out of 100 data points, 70 of these were support vectors. (There are totally 30 red cross points which are not support vectors!)



SUPPORT VECTOR MACHINES

- A linear regression model with an intercept and a term for sin(x) was fit to the model (solid black line).
- The regression line is pulled towards the outlying points.
- An SVM model with a radial basis kernel function is represented by the blue dashed line (without specifying the sin functional form).
- This line better describes the overall structure of the data.



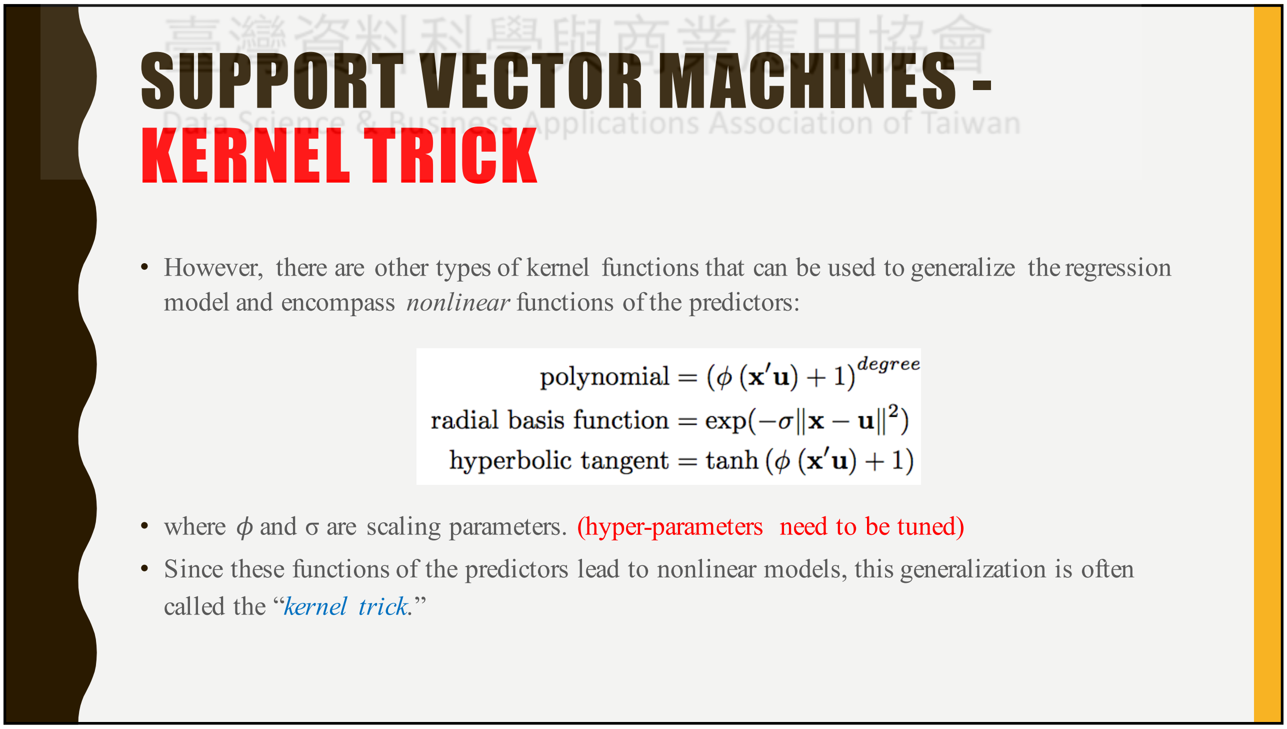
SUPPORT VECTOR MACHINES – KERNEL FUNCTION

- In matrix algebra terms, this corresponds to a *dot product*.
- This is important because this regression equation can be rewritten more generally as the equation where K(·) is called the *kernel function*.

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

- When predictors enter the model linearly, the kernel function reduces to a simple sum of cross products shown above:

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



SUPPORT VECTOR MACHINES - KERNEL TRICK

- However, there are other types of kernel functions that can be used to generalize the regression model and encompass *nonlinear* functions of the predictors:

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

- where ϕ and σ are scaling parameters. (hyper-parameters need to be tuned)
- Since these functions of the predictors lead to nonlinear models, this generalization is often called the "*kernel trick*."

SUPPORT VECTOR MACHINES – PARAMETERS TUNING

- Similarly, the radial basis function has a parameter (σ) that controls the scale.
- These parameters, along with the cost value, constitute the tuning parameters for the model.
- In the case of the radial basis function, there is a possible computational shortcut to estimating the kernel parameter.
- The parameter can be estimated using combinations of the training set points to calculate the distribution of $\|x - x'\|^2$, then use the 10th and 90th percentiles as a range for σ.
- Instead of tuning this parameter over a grid of candidate values, we can use the midpoint of these two percentiles.

SUPPORT VECTOR MACHINES – COST PARAMETER

- The cost parameter is the main tool for adjusting the complexity of the model.
  - When the cost is *large*, the model becomes very *flexible* since the effect of errors is amplified.
  - When the cost is *small*, the model will "*stiffen*" and become *less likely to over-fit* because the contribution of the squared parameters is proportionally large in the modified error function.
- We have found that the cost parameter provides more flexibility for tuning the model. So we suggest fixing a value for ∈ and tuning over the other kernel parameters.

SUPPORT VECTOR MACHINES – DATA PREPROCESSING

- Since the *predictors enter into the model as the sum of cross products*, differences in the *predictor scales can affect the model*.
- Therefore, we recommend *centering and scaling* the predictors prior to building an SVM model.
- The literature on SVM models and other kernel methods has been *vibrant* and *many alternate methodologies* have been proposed.

SUPPORT VECTOR MACHINES - EXTENSION

- One method, the *relevance vector machine*, is a *Bayesian analog* to the *SVM* model.
- In this case, the α parameters described above have associated prior distributions and the selection of relevance vectors is determined using their posterior distribution.
- If the posterior distribution is highly concentrated around zero, the sample is not used in the prediction equation.
- There are usually less relevance vectors in this model than support vectors in an SVM model.

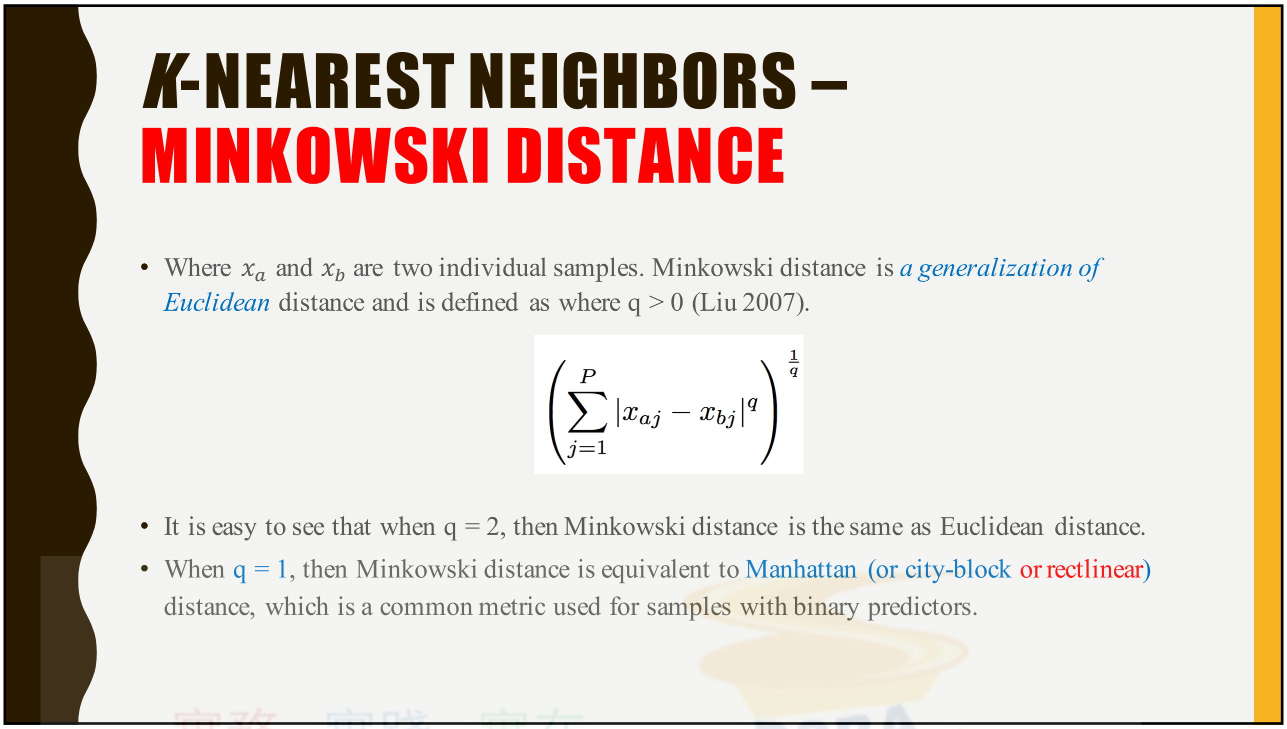
K-NEAREST NEIGHBORS

- The *K*NN approach simply predicts a new sample using the *K-closest samples* from the training set. (local information to make prediction)
- 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 then the mean of the *K* neighbors' responses. (k-NN regression)
- Other summary statistics, such as the median, can also be used in place of the mean to predict the new sample.

K-NEAREST NEIGHBORS – DISTANCE MEASURE

- The basic *K*NN method as described above depends on how the user defines distance between samples.
- Euclidean distance is the most commonly used metric and is defined as follows:

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

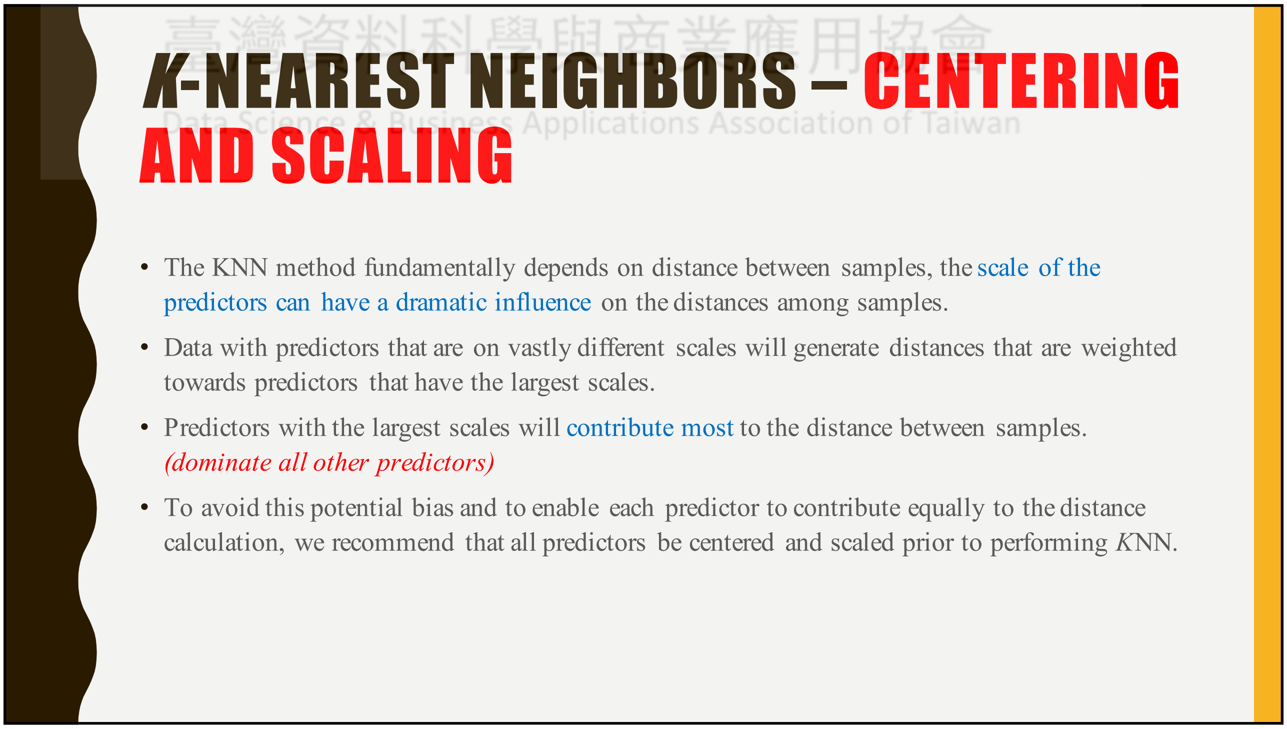


K-NEAREST NEIGHBORS – MINKOWSKI DISTANCE

- Where x_a and x_b are two individual samples. Minkowski distance is *a generalization of Euclidean* distance and is defined as where q > 0 (Liu 2007).

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

- It is easy to see that when q = 2, then Minkowski distance is the same as Euclidean distance.
- When q = 1, then Minkowski distance is equivalent to Manhattan (or city-block or rectlinear) distance, which is a common metric used for samples with binary predictors.



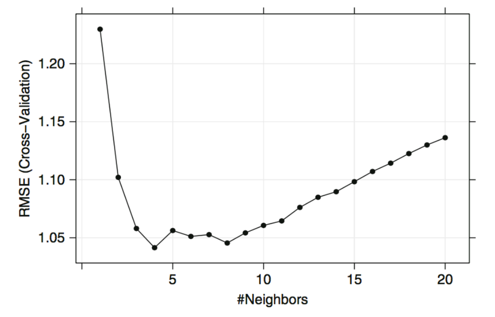
K-NEAREST NEIGHBORS – CENTERING AND SCALING

- The KNN method fundamentally depends on distance between samples, the scale of the predictors can have a dramatic influence on the distances among samples.
- Data with predictors that are on vastly different scales will generate distances that are weighted towards predictors that have the largest scales.
- Predictors with the largest scales will contribute most to the distance between samples. *(dominate all other predictors)*
- To avoid this potential bias and to enable each predictor to contribute equally to the distance calculation, we recommend that all predictors be centered and scaled prior to performing *K*NN.

K-NEAREST NEIGHBORS – MISSING VALUE & PARAMETER TUNING

- In addition to the issue of scaling, using distances between samples can *be problematic if one or more of the predictor values for a sample is missing*.
- Since it is then *not possible to compute the distance between samples*.
- Upon pre-processing the data and selecting the distance metric, the next step is to find the *optimal number of neighbors*.
- Like tuning parameters from other models, *optimal K can be determined by resampling*.
- For the solubility data, 20 values of K ranging between 1 and 20 were evaluated.

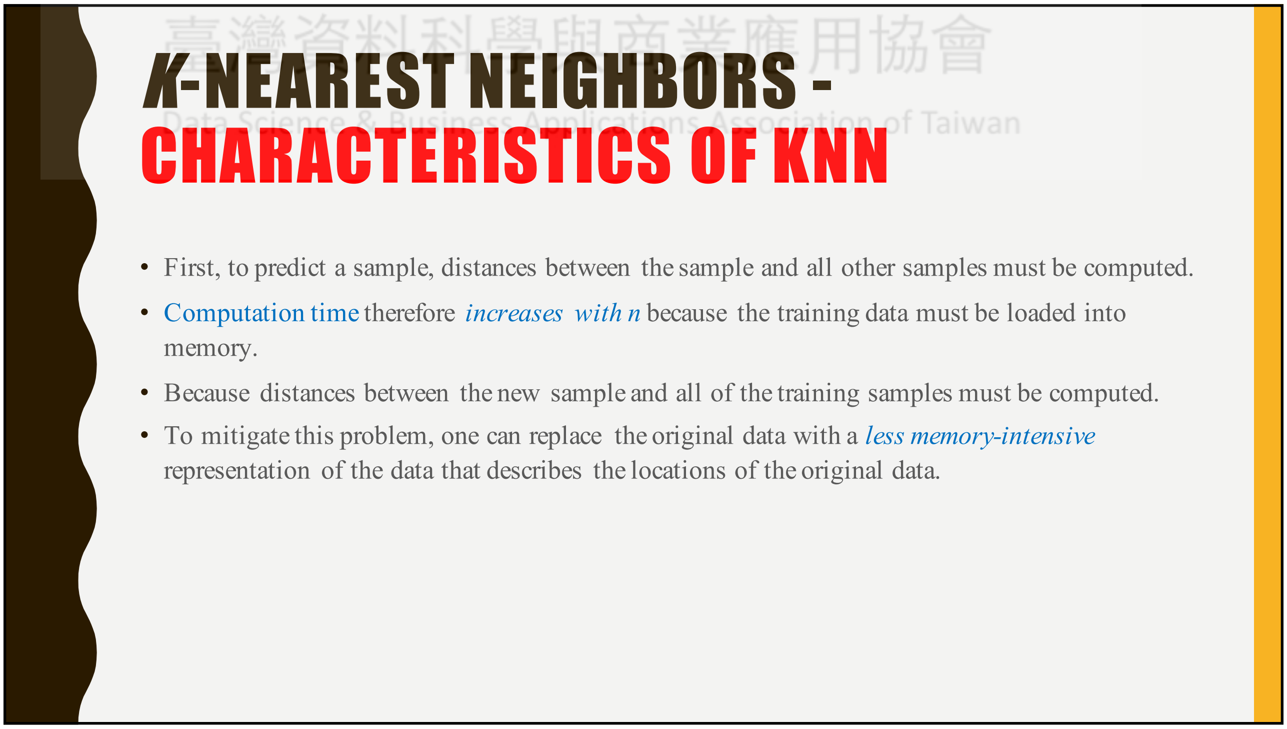
K-NEAREST NEIGHBORS - PARAMETER TUNING



- The RMSE profile *rapidly decreases across the first four values of K*.
- Then *levels off through K = 8*, followed by a *steady increase* in RMSE as K increases.
- *This performance profile is typical for KNN*, since *small values of K usually over-fit* and *large values of K underfit* the data.
- RMSE ranged from 1.041 to 1.23 across the candidate values, with the minimum occurring at K = 4.
- *Cross-validated R2 at the optimum K is 0.747*.

K-NEAREST NEIGHBORS – CHARACTERISTICS OF KNN

- The elementary version of *K*NN is *intuitive and straightforward* and can produce *decent predictions*.
- Especially when the *response is dependent* on the *local predictor structure*.
- This version does have some notable problems, of which researchers have sought solutions.
- Two commonly noted problems are *computational time* and the disconnect between local structure and the predictive ability of KNN.



K-NEAREST NEIGHBORS - CHARACTERISTICS OF KNN

- First, to predict a sample, distances between the sample and all other samples must be computed.
- Computation time therefore *increases with n* because the training data must be loaded into memory.
- Because distances between the new sample and all of the training samples must be computed.
- To mitigate this problem, one can replace the original data with a *less memory-intensive* representation of the data that describes the locations of the original data.

K-NEAREST NEIGHBORS - EXTENSIONS

- The KNN method can have *poor* predictive performance when *local predictor structure* is *NOT relevant* to the *response*.
- Irrelevant or noisy predictors are one culprit, since these can cause similar samples to be driven away from each other in the predictor space.
- Hence, *removing irrelevant*, *noise-laden predictors* is a *key pre-processing* step for *K*NN.
- Another approach to enhancing KNN predictivity is to *weight the neighbors' contribution* to the prediction of a new sample *based on their distance to the new sample. (local in local)*
- In this variation, 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.

THANK YOU

cstsou@ntub.edu.tw