

Regression - Beyond The Linear Model

Although our aim is to go beyond the simple linear model, let us start recalling its fundamental properties, some tests for it and, briefly, some of its special types (like ANOVA, ANCOVA, mixed models and multilevel models) ...

As was stated many times already, we consider:

$(x_{i0}, x_{i1}, x_{i2}, \dots, x_{ip})$: **values of explanatory variables for i th subject, $x_{i0} \equiv 1$.**

y_i : **its response, $i = 1, 2, \dots, n$. $X := (x_{ij})$ and $Y = (y_1, y_2, \dots, y_n)^T$.**

For $y_i \in \{1, 2, \dots, g\}$ - classification problem.

LS estimator $\hat{\beta}$ for a linear model: $(\beta = (\beta_0, \beta_1, \dots, \beta_p)^T)$

$$\hat{\beta} = \operatorname{argmin}_{\beta} \left\{ \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j \right)^2 \right\}$$

We are not always satisfied with LS estimates even if linear model $Y = X\beta + \epsilon$ holds.

Two reasons:

- **overfitting** occurs, resulting in poor prediction accuracy (estimators have low bias but large variance);
- **interpretation problems**: given a large number of predictors we would like to determine a smaller subset which exhibits the strongest effect on the response.

Remedies:

- subset selection of predictors;
- regularization (\approx procedures which would not allow overfitting).

12. Regularization of LS estimators: Ridge regression and LASSO

Ridge regression

$$\hat{\beta}^{\text{RIDGE}} = \operatorname{argmin}_{\beta} \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^p \beta_j^2$$

$\lambda > 0$ - parameter,

$\lambda \sum_{j=1}^p \beta_j^2$ penalty for large magnitude of coefficients.

Equivalent to minimization of

$$\left\{ \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p x_{ij}\beta_j)^2 \right\}$$

subject to

$$\sum_{i=1}^p \beta_i^2 \leq s \quad \text{for some } s = s(\lambda). \quad (1)$$

Solution:

$$\hat{\beta}^{\text{RIDGE}} = (X^T X + \lambda I)^{-1} X^T Y$$

- we add λ to the diagonal of $X^T X$ before inverting it (helps when $X^T X$ is almost singular i.e. explanatory variables are approx. linearly dependent (collinear));
- the ridge approach makes $\hat{y} = (\hat{\beta}^{\text{RIDGE}})^T \mathbf{x}$ more sensitive to directions in which x_i s are varying most;
- one needs to standardize explanatory variables before applying ridge regression;
- parameter λ is usually chosen by cross-validation.

LASSO

Constraint (1) is replaced by

$$\sum_{i=1}^p |\beta_i| \leq s.$$

LASSO, in contrast to ridge regression, eliminates for small s some variables from the model. Can be used as a **feature selection method**. Let us recommend, however, LARS, which is an extension of, and improvement over, LASSO.

13. Parametric nonlinear regression

$$y_i = f(\mathbf{x}_i, \beta) + \varepsilon_i, \quad i = 1, 2, \dots, n,$$

where $f(\cdot)$ is a known function and (ε_i) are iid random variables.

β is the sole unknown parameter of the model. Usually, ML or Nonlinear LS estimator is considered:

$$\hat{\beta}^{NLS} = \operatorname{argmin}_{\beta} \left\{ \sum_{i=1}^n (y_i - f(\mathbf{x}_i, \beta))^2 \right\}.$$

No explicit solution usually, iterative Newton-Raphson procedure usually used to find a stationary point of a criterion above.

14. Regression trees

Regression trees are a kind of nonparametric regression methods for multivariate models of the form $y = f(\mathbf{x}) + \varepsilon$ which produce \hat{f} being constant on hyperrectangles in R^p with sides parallel to the axes.

The idea is in fact the same as in the case of classification trees:

- $\hat{f}(\mathbf{x}) = \bar{y}_{\text{leaf}_j}$ for $\mathbf{x} \in N_j$,

where $N_j \subset R^p$ is a hyperrectangle specified by the conditions in the j^{th} leaf and \bar{y}_{leaf_j} is an average of the response for elements of training set falling into this leaf.

- Partition criterion: choose a predictor and a threshold which yields the maximal decrease of Sum of Squared Errors (SSE):

$$\mathcal{R}_L(j, s) = \{\mathbf{x} : x_j \leq s\} \quad \mathcal{R}_R(j, s) = \{\mathbf{x} : x_j > s\},$$

$$\min_{j,s} \min_{c_1, c_2} \left\{ \sum_{\mathcal{R}_L} (y_i - c_1)^2 + \sum_{\mathcal{R}_R} (y_i - c_2)^2 \right\}.$$

The inner minimization is solved by \hat{c}_i equal to averaged response over respective child.

- trees are grown using cost complexity criterion

$$R_\alpha(T) = SSE + \alpha|T|$$

and then pruned using crossvalidation estimator of SSE.

Drawbacks: difficulty with adopting to a linear structure and introducing high level interaction effects(as for classification), regression tree fit is discontinuous because step function are discontinuous.

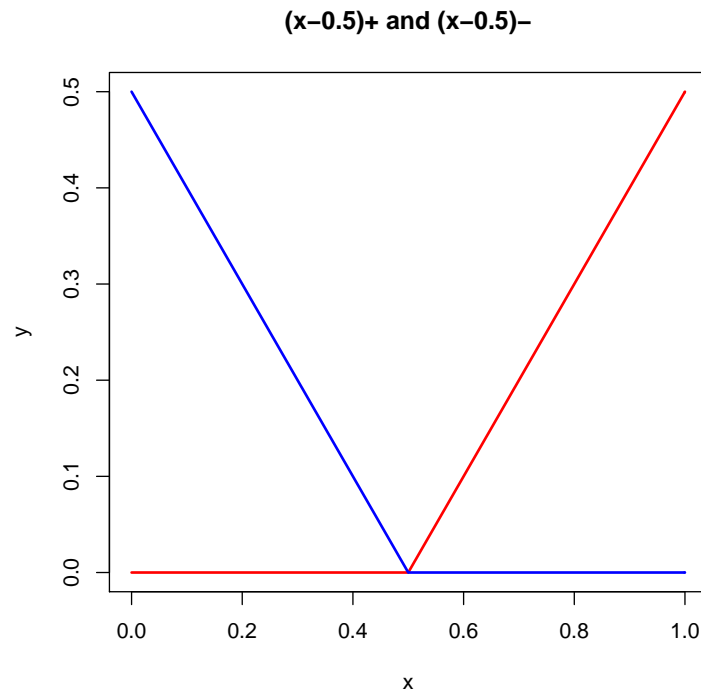
15. Multivariate Adaptive Regression Splines or MARS

Idea: Replace step functions with something smoother e.g. with

$$(x - t)_+ = \begin{cases} x - t, & \text{if } x > t; \\ 0 & \text{otherwise} \end{cases}$$

and

$$(x - t)_- = \begin{cases} t - x, & \text{if } t > x; \\ 0 & \text{otherwise} \end{cases}.$$



The collection of basis functions is

$$\mathcal{C} = \{(X_j - t)_+, (X_j - t)_-\}_{t \in \{x_{1j}, \dots, x_{nj}\}} \quad j = 1, \dots, p$$

Model building strategy is similar to forward stepwise linear regression, but with one essential difference: at each step we are allowed to use the base functions from the set \mathcal{C} **and their products**. Thus the model has the form

$$f(\mathbf{x}) = \beta_0 + \sum_{m=1}^M \beta_m h_m(\mathbf{x}),$$

where each $h_m(\mathbf{x})$ is a function from \mathcal{C} or the product of such functions.

- At stage 0 we fit a constant.
- At stage 1 we fit the function $\beta_1(x_j - t)_+ + \beta_2(x_j - t)_-$, where $j = 1, 2, \dots, p$ and $t \in \{x_{ij}\}$. Suppose that the best function is $\hat{\beta}_1(x_2 - t_{72})_+ + \hat{\beta}_2(x_2 - t_{72})_-$. This pair of basis functions is added to the set of basis functions

- **At the next stage we consider adding**

$$\beta_1 h_m(\mathbf{x})(x_j - t)_+ + \beta_2 h_m(\mathbf{x})(x_j - t)_-$$

where for h_m we have choices

- $h_0(\mathbf{x}) = 1$, ◦ $h_1(\mathbf{x}) = (x_2 - t_{72})_-$ **or** ◦ $h_2(\mathbf{x}) = (x_2 - t_{72})_+$

A large model is constructed which is then pruned using similar ideas as in regression trees.

An important property of products of functions from \mathcal{C} is their ability to operate locally. They are 0 over a part of the feature space, unlike, e.g., polynomials. This property allows to fit a parsimonious model.

16. Nonparametric methods

General remark: General nonparametric methods are based on **local smoothing** and work well for low dimensional data ($p \leq 3$) for moderate sample sizes. They require enormous sample sizes for large p (curse of dimensionality).

Case $p = 1$ - important, as many special nonparametric methods of regression fitting are based on reducing the problem to one-dimensional projections by imposing some structure on regression (**structured regression**).

We observe $(X_i, Y_i), i = 1, 2, \dots, n$, satisfying

$$Y_i = f(X_i) + \varepsilon_i,$$

where ε_i are mean-zero errors and X_i may be random or deterministic.

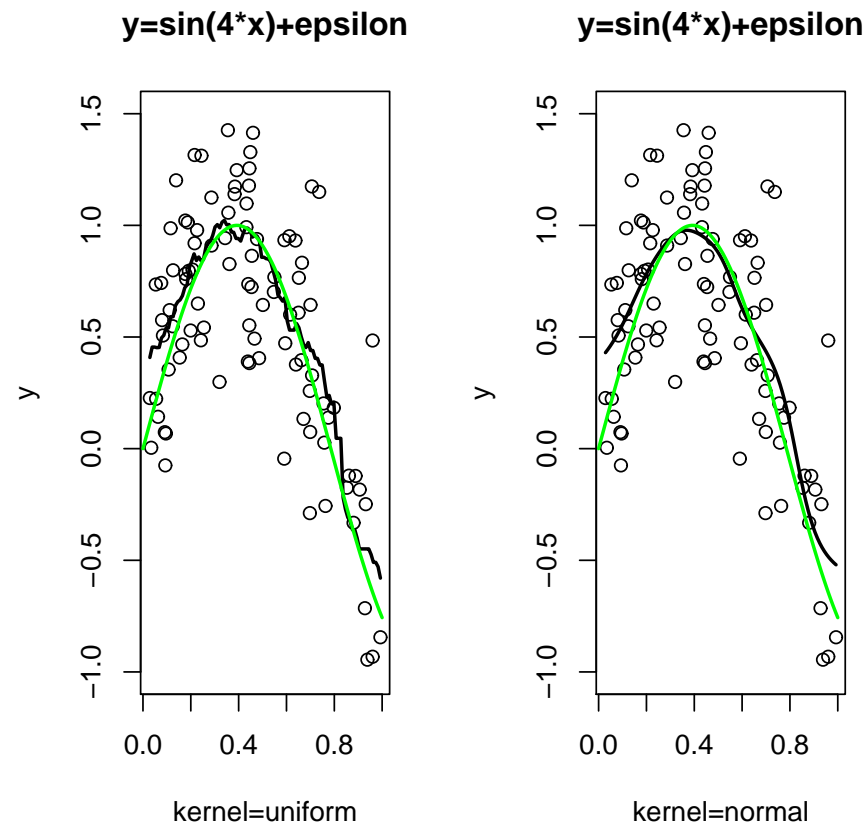
Running mean

$$\hat{f}_\lambda(x) = \frac{\sum_{i=1}^n Y_i I\{|X_i - x| \leq \lambda\}}{\#N_\lambda(x)},$$

where $N_\lambda(x) = \{i : |X_i - x| \leq \lambda\}$.

We simply take the average of Y_i s corresponding to X_i s in a small neighborhood of x .

λ specifies the size of the neighbourhood and is responsible for the amount of smoothing. It is called the bandwidth, window, or smoothing parameter.



Running median: mean is replaced by median

In the running mean algorithm, Y_i are assigned equal positive weights when corresponding X_i are close to x and 0 otherwise.

Smooth weights: Nadaraya-Watson estimator

$$\hat{f}_\lambda(x) = \frac{\sum_{i=1}^n K\left(\frac{x-X_i}{\lambda}\right)Y_i}{\sum_{j=1}^n K\left(\frac{x-X_j}{\lambda}\right)} = \sum_{i=1}^n w_i(x)Y_i,$$

K is a probability density chosen by the user. Anything smooth and compact is OK, but under some standard assumptions the optimal choice is the Epanechnikov kernel:

$$K(x) = \begin{cases} \frac{3}{4}(1 - x^2) & \text{for } |x| \leq 1; \\ 0 & \text{otherwise.} \end{cases}$$

Choice of the smoothing parameter λ

This is critical to the performance of the estimator.

- For λ too small and the estimator will be too rough (large variance- average over small number of observations, but small bias)
- For λ too large and important features will be smoothed out (small variance but large bias)

- One may choose λ interactively using the eyeball method. Plot $\hat{f}_\lambda(x)$ for a range of different λ 's and pick the one that looks best.
- Cross-validation may be used. The criterion is

$$CV(\lambda) = \sum_{i=1}^n (Y_i - \hat{f}_{\lambda,-i}(X_i))^2,$$

where $-i$ indicates that the point i is left out of the fit.

Kernel smoother has **bias problems** near boundaries of the domain. At the left endpoint when regression is increasing, most observations in a neighborhood have a **higher** mean than the target point \implies a smoother is biased upwards. Choice of λ by crossvalidation and other criteria is affected by this.

Local linear smoother

Idea: approximate locally $f(x)$ by a linear function $\beta_0(x) + \beta_1(x)x$.

$$(\hat{\beta}_0(x), \hat{\beta}_1(x)) = \operatorname{argmin}_{\beta_0(x), \beta_1(x)} \sum_{i=1}^n K\left(\frac{x - X_i}{\lambda}\right) (Y_i - \beta_0(x) - \beta_1(x)x)^2$$

We define $\hat{f}_\lambda(x) = \hat{\beta}_0(x)$.

Local polynomial smoother can be defined analogously.

Splines

Suppose we choose to minimise the MSE

$$\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{f}(x_i))^2.$$

The solution is $\hat{f}(X_i) = Y_i$ i.e. ' join the dots' regression, which is probably too rough. Instead minimize

$$\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{f}(x))^2 + \lambda \int (\hat{f}^{(2)}(x))^2 dx,$$

where $\lambda > 0$ and $\int (\hat{f}^{(2)}(x))^2 dx$ reflects global **roughness** of the estimator. The second term is a penalty for roughness.

A solution can be sought in the class of **natural cubic splines**:

Piecewise cubic polynomial on each interval $(X_{(i)}, X_{(i+1)})$ such that it has continuous second derivative and linear outside $(X_{(1)}, X_{(n)})$.

Cross-validation is often used to select λ . Usually works very well in practice.

17. Additive models and projection pursuit regression

This is a multivariate nonparametric modeling attempt to avoid curse of dimensionality. It postulates the following structure

$$Y_i = \alpha + \sum_{j=1}^p f_j(X_{ij}) + \varepsilon_i, \quad i = 1, 2, \dots, n$$

where f_j are smooth arbitrary functions. We set $E f_j(X_{ij}) = 0$ for identifiability of α .

- More flexible than the linear model but still interpretable since the functions f_j may be plotted;
- Will do poorly when strong interactions exist. In this case one might consider adding $f_{ij}(X_i X_j)$.
- Categorical variables may be incorporated using the usual regression approach.

The backfitting algorithm is used to estimate f_i :

1. **Initilize:** set $\alpha = \bar{Y}$, $f_j \equiv \hat{\beta}_j$, where $\hat{\beta}$ is an initial estimate like LS for $j = 1, \dots, p$.

2. **Cycle** $j = 1, \dots, p, 1, \dots, p, 1, \dots$

$$f_j = S(X_j, y - \alpha - \sum_{i \neq j} f_i(X_i)),$$

where $S(x, y)$ means a smoother on the data (x, y) .

3. **Repeat until convergence.**

Note:

1. $y - \alpha - \sum_{i \neq j} f_i(X_i)$ is a partial residual - the result of fitting everything except X_j ;

2. Choice of S is left open to the user: could be splines or Loess, say.

The algorithm converges under some rather loose conditions.

Projection pursuit regression

Note that

$$x_1x_2 = \frac{1}{4}((x_1 + x_2)^2 - (x_1 - x_2)^2)$$

and $x_1 + x_2 = (x_1, x_2)(1, 1)^T$, $x_1 - x_2 = (x_1, x_2)(1, -1)^T$.

This is a partial case of a general property saying that functions of the form

$$\alpha_0 + \sum_{j=1}^J f_j(\alpha_j^T x) \tag{2}$$

approximate arbitrarily well continuous functions on hypercubes.

Projection pursuit regression uses this property approximating unknown regression function by functions of the form (2).

Curse of dimensionality is avoided as only one dimensional projections are considered.

The f_j 's are estimated nonparametrically and J is chosen in an adaptive way. Projection pursuit regression is an iterative algorithm which looks for 'interesting' directions in the data, which can explain the largest part of the remaining variability of Y .

Projection pursuit regression algorithm

Let $\hat{f}_{\alpha_j}(\alpha_j^T x) = \hat{f}_j(\alpha_j^T x)$.

1. $J := 0$, $\alpha_0 = \bar{Y}$, $Y_i = Y_i - \bar{Y}$, $R_j := Y_j$.

2. For any linear combination $z = \alpha^T x$ and sample (Z_i, R_i) , $i = 1, 2, \dots, n$ with $Z_i = \alpha^T X_i$, estimate $\hat{f}_\alpha(z)$ and

$$I(\alpha) = 1 - \frac{\sum_{i=1}^n (R_i - \hat{f}_\alpha(Z_i))^2}{\sum_{i=1}^n R_i^2} \quad (= 1 - SSE/SST).$$

Find $\alpha_{J+1} = \operatorname{argmax} I(\alpha)$ and store $\hat{f}_{\alpha_{J+1}}(\cdot)$.

3. Stop if $I(\alpha) \geq$ given threshold,

otherwise $J := J + 1$ and $R_i := R_i - \hat{f}_{\alpha_j}(\alpha_j^T X_i)$.

Optimisation in 2 is not trivial !

18. Neural Networks

The idea is similar to Projection Pursuit Regression.

Feed-forward neural network with one hidden layer

- $\mathbf{X} = (X_1, X_2, \dots, X_p)^T$ - p -dimensional input vector,
 $\mathbf{Y} = (Y_1, Y_2, \dots, Y_C)^T$ - C -dimensional target vector.
- Derived features are created from linear combinations of inputs

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T \mathbf{X}), \quad m = 1, 2, \dots, M,$$

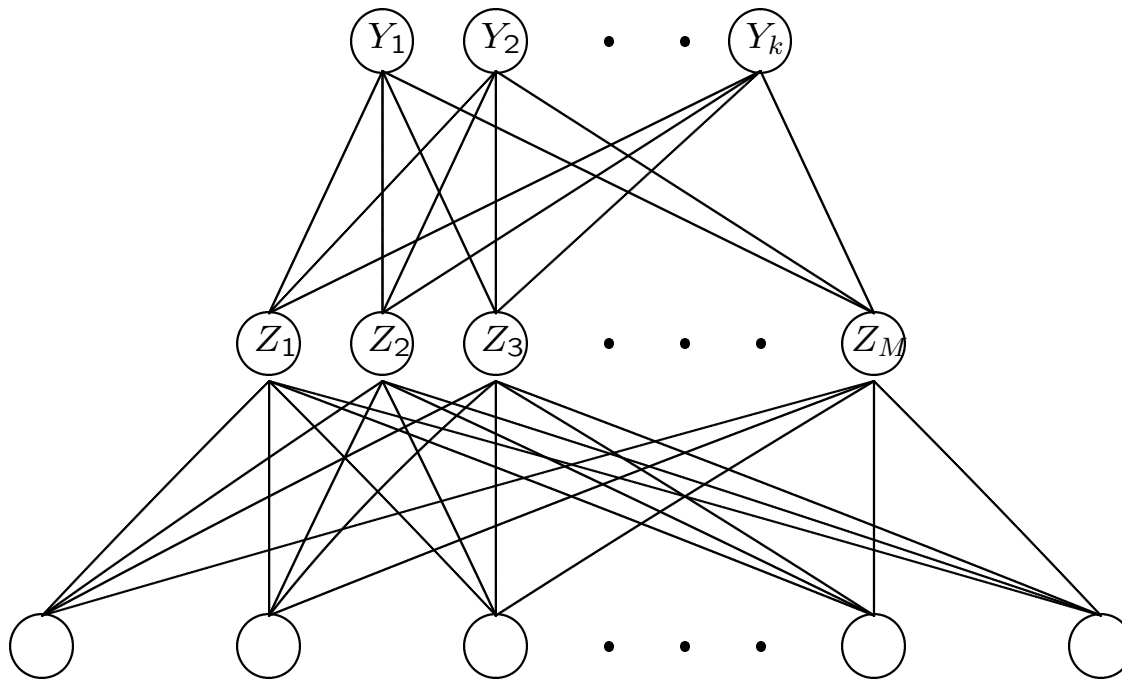
where $\sigma(\cdot)$ is a **known** function (whereas ridge functions in PPR are unknown).

- Target Y_k ($k=1,2,\dots,C$) is modeled as a function of a linear combinations of the Z_m

$$T_k = \beta_{0,k} + \beta_k^T \mathbf{Z}, \quad k = 1, \dots, C, \quad f_k(\mathbf{X}) = g_k(\mathbf{T}),$$

where $\mathbf{Z} = (Z_1, \dots, Z_M)^T$ and $\mathbf{T} = (T_1, \dots, T_C)^T$.

Z_1, Z_2, \dots, Z_m form a hidden layer of neural network



Neural networks used for regression estimation with $C = 1$ and g_k -identity, or for classification with g_k being softmax functions

$$g_k(\mathbf{T}) = \frac{e^{T_k}}{\sum_{l=1}^C e^{T_l}}.$$

Without the hidden layer neural network would then reduce to multiclass logistic regression.

Activation function σ : usually a sigmoid function

$$\sigma(v) = \frac{1}{1 + e^{-v}}.$$

Remarks

1. Sum-of-squared errors $R(\theta)$ (θ -unknown weights) used as a measure of fit. Optimization is nontrivial due to occurrence of multiple local minima. Optimization algorithms: back-propagation (very slow), conjugate gradients and simulated annealing.

2. Overfitting is a serious problem here: frequently neural networks have too many weights and will overfit the data at the global minimum of $R(\theta)$. Remedy: weight decay penalty analogous to ridge regression. Penalized criterion

$$R(\theta) + \lambda \left(\sum_{k,m} \beta_{k,m}^2 + \sum_{m,l} \alpha_{km}^2 \right)$$

with λ chosen by crossvalidation. Recommended strategy: choose many hidden units and allow for regularization.

3. Scaling of inputs is recommended.

19. Generalized Additive Models or GAMs

Generalized Additive Models combine the ideas of additive modelling and generalized linear (in particular logistic) modelling. Namely, we assume that for some function g (link function)

$$g(E(Y|X_1, X_2, \dots, X_p)) = \alpha + f_1(X_1) + f_2(X_2) + \dots + f_p(X_p),$$

where f_i are unspecified smooth ('nonparametric') functions and, for identifiability, $E f_i(X_i) = 0$. In particular logistic additive model has the form

$$\log \frac{P(Y = 1|\mathbf{X})}{P(Y = 0|\mathbf{X})} = \alpha + f_1(X_1) + f_2(X_2) + \dots + f_p(X_p),$$

where $\mathbf{X} = (X_1, X_2, \dots, X_p)$. For (X_i, Y_i) $R^p \times R$ -valued independent random variables we assume that

$$(Y_i|\mathbf{X}_i = (x_1, x_2, \dots, x_p)) \sim \text{Bin}(1, \text{ilogit}(\alpha + \sum_{j=1}^p f_j(x_j))),$$

where $\text{ilogit} = \text{logit}^{-1}$.