Machine Learning II

Prof. Dr. Tim Downie

Week 7 Lecture – 27th November 2019

Projection Pursuit Regression

Version: 26. November 2019

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Contents

- Adaptation from a GAM to the projection pursuit regression model
- With one ridge function
- Multiple ridge functions
- Similarity to artificial neural networks

Text books:

- Hastie, Tibshirani and Friedman (2009), *The Elements of Statistical Learning (2nd edition).* Springer-Verlag.
  `https://web.stanford.edu/~hastie/ElemStatLearn/`
- W.N. Venables and B.D. Ripley (2002). *Modern Applied Statistics with* S (fourth edition, Springer). *Modern Data Science with R.* Chapman and Hall.
  `https://www.stats.ox.ac.uk/pub/MASS4/`

Generalised additive Models

In supervised learning, given an input vector $X = (X_1, X_2, \ldots, X_p)^\top$ and outcome variable Y (Target), the aim is to find a predictor function $f(X)$ which predicts Y well.

A multiple linear regression model has the form:

$$Y \approx f(X) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p.$$

A non-linear equivalent is the generalised additive Model (GAM) ML2 Week 3.

$$Y \approx f(X) = \beta_0 + g_1(x_1) + g_2(x_2) + \ldots + g_p(x_p).$$

In a GAM, each g_m is allowed to be a different type of regression function on one variable.
So for example, g_1 could be a quadratic function, g_2 a smoothing spline function and g_p a loess function.

Projection pursuit regression model

The GAM Model restricts each function g_m to “respond to” just one variable, corresponding to one dimension of the X vector. To get more flexibility we can apply g_m to a linear combination of $X_1, X_2, \ldots, X_p$.

Let V_1 be any linear combination of the elements of X. We can define V_1 using the scalar product with the *direction vector* ω_1:

$$V_1 = \omega_1^\top X \tag{1}$$

A direction vector is normalised to be a unit vector, i.e. $\|\omega_1\| = 1$, which ensures the expression in Equation 1 is unique up to the sign.

The data $X_1, X_2, \ldots, X_p$ are projected on to a line in p-dimensional space along the direction vector ω_1. The value of V_1 is then the signed distance of the projected point from the origin.

Graphical example of projection and ω direction vectors in two dimensions

$p = 2$. Diagram on blackboard

We can fit the model

$$Y \approx f(X) = \beta_0 + \beta_1 g_1(V_1), \qquad \text{where } V_1 = \omega_1^\top X.$$

The function g_1 is called a ridge function, it “responds to” the vector X in the direction of ω_1, which graphically looks like a ridge.

For any observed element or prediction, V_1 is a scalar value, meaning so g_1 can be any line/curve fitting method, popular choices are loess and spline smoothing methods, as with GAMs.

The Projection pursuit regression (PPR) model, allows for multiple projections and multiple smoothing functions, specified by M

$$Y \approx f(X) = \beta_0 + \beta_1 g_1(V_1) + \cdots \beta_M g_M(V_M),$$

where $V_m = \omega_m^\top X$, $m = 1, \ldots, M$.

Appropriate combinations of direction vectors and ridge functions allow for a large range of predictor functions to be modelled including interactions and higher order polynomials without needing to specify the required form in advance, see Hastie et al. for the mathematical details.

The algorithm used to fit a PPR models closely resembles the back fitting algorithm for GAMs.

1) Calculate the constant parameter $\widehat{\beta}_0 = \overline{y}$.
2) Initialise $M = 1$ and ω_1. A common initial direction vector is $\omega_1^\top = \frac{1}{\sqrt{p}}(1, 1, \ldots, 1)$.
3) Iteratively repeat steps a) and b) using $m = 1, \ldots M$.
   a) Select a ridge function component, $m \in \{1, \ldots, M\}$ and obtain the m-th partial residuals $\eta_{i,m}$.
   b) Fit the ridge function $g_m(v)$ to the partial residuals transformed in the direction $\omega_m^\top$. Calculate $V_{i,m} = \omega_m^\top \eta_{i,m}$.
   c) Update ω_m to minimise the sum of squared errors

$$SSE = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{m=1}^{M} \beta_m g_m(\omega_m^\top \boldsymbol{x}_i) \right)^2$$

4) Increase M by one, to add a new ridge function, and initialise ω_M. Repeat steps 3) a)-c) above

Comments on the algorithm

- Reminder the *m*th-partial residuals are defined as:

$$\eta_{im} = y_i - (\beta_0 + \sum_{k \neq m} \beta_k g_k(\omega_k^\top \boldsymbol{x}_i)).$$

  They are the residuals plus the current $g_m(V_m)$ component.

- Step 3 b) depends on the chosen regression/smoothing method covered in various ML1/ML2 lectures.
- The precise ordering of: i) improve g_m, ii) improve ω_m and iii) increase M is not critical to the algorithm results and so we will not consider this in detail.
- The minimisation in Step 3 c) is done using Gauß-Newton optimisation. An important algorithm for function minimisation, which we will investigate in more detail in the coming weeks.

Example

In ML1 Lecture 9, we fitted a regression tree model to a data set called `cpus` from the MASS library. A relative performance measure and characteristics of 209 CPUs from 1987.

The data include the variables:

`name`: manufacturer and model.

`syct`: cycle time in nanoseconds.

`mmin`: minimum main memory in kilobytes.

`mmax`: maximum main memory in kilobytes.

`cach`: cache size in kilobytes.

`chmin`: minimum number of channels.

`chmax`: maximum number of channels.

`perf`: published performance on a benchmark mix relative to an IBM-370/158-3.

As `perf` is a heavily skewed variable we will use $y = \log_{10}(\texttt{perf})$ as our outcome variable.

A couple of PPR models will be fitted on training data and the MSE will be compared to a linear model and a RPART tree model.

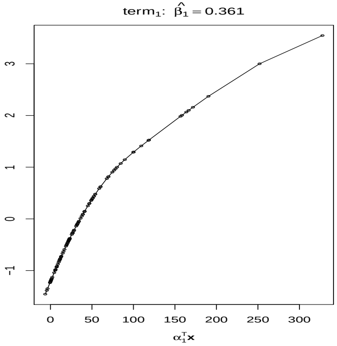
```
cpus.ppr <- ppr(log10(perf) ~ syct+mmin+mmax+cach+chmin+chmax,
         data = cpus[cpus.train,], nterms = 2,sm="spline")
```

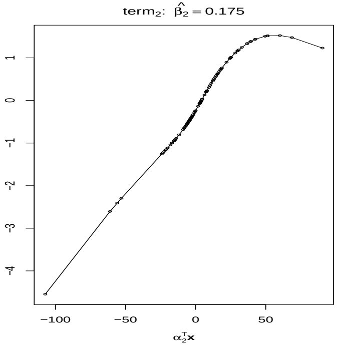
- The function called `ppr` is part of the MASS library.
- The model formula specifies the variables to include, which uses similar notation to `lm` and `rpart`.
- The model will fitted with two components $M = 2$: `nterms = 2`.
- The type of non-linear regression to be used is spline smoothing, specified by `sm="spline"`

The direction vectors are

```
             term 1       term 2
syct  -0.013314130  0.128598978
mmin   0.006151632 -0.004512696
mmax   0.002680398  0.002356409
cach   0.939999246 -0.894044023
chmin  0.322811306  0.210153382
chmax  0.109416569  0.374108577
```

The two fitted functions are





Comparison of models

Using the test data, the following MSE values were obtained

Multiple linear regression	0.0460
Rpart tree	0.0565
PPR	0.0290

The PPR model gives in this one example substantially better prediction on the test data set.

The disadvantage is that it is not easy to interpret the results, unlike with a linear or tree model

Similarity between PPR and artificial neural networks

A simple artificial neutral network (NN) with one hidden layer is often portrayed graphically in the following form.

Diagram on blackboard

The input nodes correspond to the p elements of one data vector $\boldsymbol{X}$.

The m-th node in the hidden layer takes a linear combination of $\boldsymbol{X}$:
$Z_m = \omega_m^\top \boldsymbol{X}_i$.

The output of each node is linear combination Z_m after transformation of an activation function: $g_m(Z_m)$

The output layer takes the activations applies another linear combination and outputs the results.

$$\widehat{y} = \phi_0 + \phi_1 g_m(Z_1) + \cdots + \phi_M g_m(Z_M).$$

The PPR model is a specific type of NN model. Usually with a NN the activation functions are specified in advance. With PPR non-linear regression function are fitted to the transformed data.