Fitting Generalized Additive Models for very large datasets with Apache Spark

An exposé for a bachelor thesis

Kai Thomas Brusch

31.10.2015

Summary This exposé describes content, goals and motivation and gives a general introduction to my bachelor thesis titled "Fitting General Additive Models for very large datasets with Apache Spark".

Contact kai.brusch@gmail.com

Location Hochschule für Angewandte Wissenschaften Hamburg

Department Dept. Informatik

Examiner Prof. Dr. Michael Köhler-Bußmeier

Second examiner TBA

Contents

1 Introduction

1.1 Context

Regression analysis stands at the heart of modern statistical analysis. Discovering the relationship between an dependant and independant variables may offer great insight into the underlying system and can even provide far reaching predictive capabilities. The traditional linear models explain the dependant variable as the sum of the independant variables and the linear interaction with a coefficent.[Wood, 2006] describes the generallized additve model (GAM) as powerful extension of the generalized linear model, explaining the relationship between dependant and independant variables as the sum of smoothing functions. This modeling approach offers more flexiblity and power than linear model and has been succesfuly established in academia and industry. However, the additional power of GAMs comes at the cost of additional memory requirements and computational effort, placing strict constraints on the model and data size.[Wood et al., 2015] suggest two optimization to eliviate this constraint: parallelization and numerical optimization. These optimizations have been implemented and used in high-level statiscial languanges such as R, SPSS and Stata. While these implementations already allow to fit larger models with more data they have never been implemented in a truly distributed enviroment.[Zaharia et al., 2010] introduced Apache Spark which has recently established itself as one of the most promising general purpose cluster computing engines. It's general and distributed nature offers the ideal enviroment to implement the by [Wood et al., 2015] suggested optimizations to bring GAMs beyond the limitations of high-level statistical languages.

2 Content

2.1 General Linear Models

Linear models are statistical models in which a univariate response is modelled as the sum of a 'linear predictor' and a zero mean random error term. The linear predictor depends on some predictor variables, measured with the response variable, and some unknown parameters, which must be estimated. A key feature of linear models is that the linear predictor depends linearly on these parameters. Statistical inference with such models is usually based on the assumption that the response variable has a normal distribution. Linear models are used widely in most branches of science, both in the analysis of designed experiments.

$$y_\mathrm{i} = \beta x_\mathrm{i} + \varepsilon_\mathrm{i} \tag{1}$$

Equation 1 introduces formal notation on how to describe the dependant variable y as linear combination of x and the unknown parameter β plus an error term ε. We want β to fit all data as closely as possible and hence introduce S as the squared difference between an estimated $x_i\beta$ and y_i

$$\mathbf{S} = \sum_{i=1}^{n} (y_\mathrm{i} - x_\mathrm{i}\beta)^2 \tag{2}$$

A good choice of β should bring S close to 0. The Markov-Gauss Theorem states that the minimization of S in yields $\widehat{\beta}$ which is the best possible estimation of b. 1 represents the univivariate case where y is explained with only one variable. This model can be extendend to multiple explanatory variables yielding in a scalar form of 1

$$y = \mathbf{X}\beta \tag{3}$$

The dependante variable vector y is the linear combination of model matrix X and the vector of unknown parameter β.

2.2 Generalized Linear Models

Generalized linear models (GLMs) somewhat relax the strict linearity assumption of linear models, by allowing the expected value of the response to depend on a smooth monotonic function of the linear predictor. Similarly the assumption that the response is normally distributed is relaxed by allowing it to follow any distribution from the exponential family (for example, normal, Poisson, binomial, gamma etc.).

$$g(\mu_{\mathrm{i}}) = \mathbf{X}\beta \tag{4}$$

Equation 4 introduces formal notation for GLMs and illustrates the similarities to the genral linear model. It is important to recognize the smooth monotonic function *g()*. The smooth monotonic function, also known as link function, is the key extension which enables GLMs to model members of the expotential family with a linear model. Finding the best possible estimator $\widehat{\beta}$ for 4 is formalized stated as the minimal length of the eucledean distance:

$$\|y - X\beta\|^2 \tag{5}$$

Minimizing 5 will yield in the best possible estimator $\widehat{\beta}$. This has been well established and many of the following problems leverage knowledge on how to obtain $\widehat{\beta}$ through 5.

2.3 Generalized Additive Models

Generalized Additive Models (GAMs) extends the GLM by specifing the linear prediction in terms of the summation of smooth functions. This allows for a more flexible modelling of the influence for each explanatory variable. The gained flexibility comes at the cost of additional questions:

- The smooth functions must be represented somehow.
- The degree of smoothness of the functions must be made controllable.
- Some means for estimating the most appropriate degree of smoothness from data is required.

With the outlined additions to GLMs we can now introduce formal notation for GAMs:

$$g(\mu_{\mathrm{i}}) = \mathbf{X}_{\mathrm{i}}\Theta + f_1(x_{1\mathrm{i}}) + f_2(x_{2\mathrm{i}}) + f_3(x_{3\mathrm{i}}, x_{4\mathrm{i}})... \tag{6}$$

6 explains y_i as the model matrix for this row and the smooth functions $f_j(x1j)$ of the x values for this row. X_i is a row of the model matrix with parametric component. θ. Unlike the linear model we can now specify a smooth function for each explanatory variable. This proves to be way more flexible than only allowing for a constant influece per explanatory variable. The natural question that arises now is: How do I find a proper smoothing function? Finding the right smooth function stands at the heart of GAM fitting and can be best illustrated in the univariate case 7.

$$y_i = f(x_i) + \epsilon_i \tag{7}$$

Literature suggest that the unknown smooth function $f()$ can be best fund by specifiying the a linear base $b()$ for the smooth function. Only allowing linear basis allow us to heavily leverage the theory already developed for linear models and S as the optimal model fit. For the sake

of illustration we assume that 7can be rewrite as the following equation if $b_i(x)$ is the ith basis function:

$$f(x) = \sum_{i=1}^{q} b_{\mathrm{i}}(x)\beta_{\mathrm{i}} \tag{8}$$

In 8 we already know bi and that linearity 8is linear. We now have to specify a basis function to represent bi. We can choose from many basis functions for bi, each with advantages and disavantages. A common choice however is a fourth order polynomial allowing basis function. 8 represented by a fourth order polynomial yields the following model:

$$f(x) = \beta_1 + x\beta_2 + x^2\beta_3 + x^3\beta_4 + x^4\beta_5 \tag{9}$$

Applying 9 to 7 we get the description of yi as the sum of smoothing functions.

$$y_i = \beta_1 + x_i\beta_2 + x_i^2\beta_3 + x_i^3\beta_4 + x_i^4\beta_5 + \epsilon_i \tag{10}$$

Equation 10 gives a formal full description of $\mathrm{y_i}$ as the sum of a fourth degree polynomial. Fourth order polynomial basis function enforce rather strict limitations and often struggle to provide the desireable fit. Instead of trying to fit a single fourth order polynomial to all our data points we will now try to represent f with cubic splines. This approach is called regression splines and follows this general pattern:

- Divide the curve in to a fixed number of sections, the point between two sections is called a knot.
- Find a cubic polynomial for each section between two knot
- Each cubic polynomial must match its neighboors in value, first and second derivative at the location of the knot.
- Each spline has zero second derivatives at the knot location to ensure a smooth transition.

Using cubic splines as a basis for f means that 7 becomes a linear model identical to 3. We can hence rely well established linear model theory to fit the very flexible class of GAMs.

The number and location of knots play an essential role in GAM fitting and are commonly defined and hence not relevant. Since they are specified we will not futher eleborate on them.

After established the basis function and knots we will now turn to the smoothing parameter λ. λ is a penalty factor that controlls the smoothness by scaling the second derivative of the smooth function, this can be formalized by:

$$\|y - X\beta\|^2 + \lambda \int_0^1 [f''(x)]^2 dx \tag{11}$$

The integrated square root of second deriviative penalizes models that have a too 'rough' function. As λapproaches 0 f will become a straight regression line while $\lambda$of 0 becomes an unpenelized regression spline. The trade off between model fit and and model smoothness is controlled by the smoothing parameter λ.

The linear nature of f allows us to rewrite 11 in it's final form:

$$\|y - X\beta\|^2 + \lambda\beta^T S\beta \tag{12}$$

2.5 shows that describing an unknown variable as the sum of smooth functions can be restated as the minimization of penelized regression splines and hence becomes problem of minimizing equation 2.5. While the the problem of finding a smooth function for a univariate function has almost been solved by 2.5 we are still left with an unknown parameter of λ. Finding the optimal l is subject of the following section.

2.4 Smoothing parameter Estimation with generalized cross validation

Finding a good smoothing parameter through minimizing 2.5 poses a central question for GAM research. Finding a optimal choice for λmeans to trade goodness of fit to smoothness. A high λwill overfit the data while a low λwill create undersmoothed splines. The common approach is to define a measure for the predictive capabilities of our estimated spline. For our purposes we will use the generalized cross validation (GCV) which formalizes the notion that an ideal splines minimizes the average distance between our splines and the function f in relation to the mean weight: $tr(I-A)/n$ yielding

$$V_g(\lambda) = \frac{n \left\| y - X\widehat{\beta}_\lambda \right\|^2}{\{n - tr(F_\lambda)\}^2} \tag{13}$$

2.5 describes the full process of finding the optimal λ. Literature suggests a Newton method to optimize V_g with respect to $log(\lambda)$ with $\widehat{\beta}_\lambda$ being obtained from the minimization of as the full process of finding a optimal smooth function. In practice however has penalized likelihood maximization, in place of penalized least squares proven to be more useful. The algorithm that is used to maximize the penalized likelihood is penalized iteratively reweighted least squares (PIRLS) which proceeds as follows, where V is the function such that $var(y_i) = \phi V(\mu_i)/$ and $\mu_i = E(y_i)$. First initialize $\widehat{\mu}_i = y_i + \epsilon_i$ and $\eta_i = g(\mu_i)$ where ϵ_i is a small quantity (often 0) added to ensure that $g(\mu_i)$ exists.

Then iterate the following steps to convergence.

1. form $z_i = g'(\widehat{\mu}_i)(y_i\widehat{\mu}_i) + \eta_i$ and $w_i = V(\widehat{\mu}_i)^(1/2)g'(\widehat{\mu}_i)$.
2. putting the wi in a diagonal matrix W, minimize the weighted version of expression 2.5:

$$\|Wz - WX\beta\|^2 + \lambda\beta^T S\beta \tag{14}$$

with respect to β to obtain $\widehat{\beta}_i$ and the updates $\widehat{\eta}_i = X\widehat{\beta}$, and $\widehat{\mu}_i = g^-1(\widehat{\eta}_i)$.

For moderate size data PIRLS will converge for each tiral λ. The main issue with PIRLS is the requirement to form the whole model X at each iteration. The difficulty is simply that the model matrix for the model can become too big: if n and p are respectively the number of rows and columns of the model matrix, and M is the number of smoothing parameters, then the memory requirements of GAM fitting methods are typically $O(Mnp^2)$. The aim of my bachelor thesis is to lay out a faster way of finding the smoothing term.

2.5 Generalized Additive Models for large data sets

[Wood et al., 2015] introduces two extensions of the development methods for fitting GAMs. Those suggested methods aim at numerical optimization and parallelization.

The numerical optimization propossed by [Wood et al., 2015] rewrites the general cross validation step to not require the full model matrix X in each step. Woods describes a simple stategy how model matrix factorization can be used to avoid formation of the whole model matrix in each iteration.

Now suppose that the model matrix is first QR decomposed into a column orthogonal $n * p$ factor Q and an upper triangular $p * p$ factor R so that $X = QR$: If we also form $f = QTy$ and then expression becomes

$$\|f - R\beta\|^2 + \|r\|^2 + \sum_j \lambda_j \beta^T S_j \beta \tag{15}$$

. 15 allows us to rewrite 15 in a form that does not require the full model matrix X.

$$V_g(\lambda) = \frac{n \left\| f - R\widehat{\beta}_\lambda \right\|^2 + \|r\|^2}{\{n - tr(F_\lambda)\}^2} \tag{16}$$

.

brings a tramendous advantage over in terms of memory and computational usage. Receiving all relevant information to find λfor a given X only forming subsections $R\ f$ and $||r||^2$ gives a significant advantage over the proposed method in 2.5.

The parallelization optimimization of envolves the computations of $R\ f$ and $||r||^2$ in a way that only requires small, disjunct subsections of X. This allows us to distribute the computation by grouping X into equally sized non-overlapping sets and allocate them to different processors.

2.6 Apache Spark

Distributed computing is a special interest of mine. I've had the pleasure of setting up a SparkR cluster and have been following its development closely. Apache Spark advertises itself as a fast and general cluster computing engine for large-scale data processing. He a overview of it's key features:

- fast, Apache Spark currently holds the world record in sorting 1 TB on data, beating the previus record help by Hadoop. Spark reaches this high speed by removing I/O to disc and maintaining all relevant in memory.
- general, Aparche Spark offers a high level API called Resilient Data Set. RDDs are designed to facilitate a resiliend, distributed data set as a optimal level of interaction with data. Each of the key features of RDDs are outlined below:
  - Resilient: Every RDD is evaluated lazy and constructed from it's lineage, point to different previous RDD.
  - Distributed: RDDs are partiioned accross many machines
  - Data set: RDDs are ment to think about in terms of matrixes, this should feel very natural for people familar with data.
- cluster computing, Apache Spark is build to be executed in a cluster enviroment like Mesos or Yarn. The management and design of these systems are not subject of my bachelor thesis.
- large scale data processing, the data managable by Apache Spark is limited by the amount of memory availble in a cluster.

2.7 General additive models in Apache Spark

I strongly believe that Apache Spark offers an ideal enviroment for gitting GAMs for large data sets. The optimizations suggested by [Wood et al., 2015] are optimal for an general purpose in-cluster computing engine for the following reasons:

- The iterative nature of PIRLS is optimal for Spark's in memory computing.
- Spark's high level RDD allows for a flexible data structure to implement the suggested methods.
- Spark is designed to facilitate the parallel nature of finding $R\ f$ and $||r||^2$ in subblocks of X.
- Spark supports [Hall, 2009], a linear algebra libraries for Scala with far reaching capabilities.

3 Guiding Questions

The leading question for my bachelor thesis can be stated as:

Can the distributed, numerically optimized, in-memory version of GAM fitting outperform previous benchmark implementation in model size, data size and fitting speed?

Subquestions arising are:

- Can the suggested methods be implemented in Apache Spark?
- How much faster is a parallel variant against the sequential one?
- How much faster is the Spark in-memory variant against the R one?
- How do the limit for data and model size change?
- How much does horizontal scaling affect performance?

3.1 What is novel in this approach?

The suggested approach has been implemented in high-level statistical languages, proving its effectiveness. The suggested methods have not been implemented in an enviroment designed for in-memory cluster computing. As I lined out earlier the iterative nature of the proposed methods is well suited for a distributed in-memory enviroment and hence should be discoved

3.2 When is the thesis regarded as completed?

The thesis is considered finished, once the optimized version of GAM fitting has been implemented in Spark made available in SparkR and compared to other implementation of GAMs.

Additional tasks

- Develop a deep theoretical unterstanding of GAMs with [Wood, 2006]
- Develop a better understanding of used matrix decompositions with [Strang, 2009]
- Set up an Apache Spark enviroment
- Cordinate with my employer to facilitate support

3.3 Which tasks are realistic to be accomplished within the scope of this thesis?

I strongly belive that the suggested methods can be implemented in Apache Spark within this thesis. In an ideal situation I would like to make my implementation available to the Apache project.

4 Time table

My time plan is visible at table 1.

Table 1: Time table

CW	thesis work
45	Develop a deep theoretical unterstanding of GAMs and used numerical methods
46	Develop a deep theoretical unterstanding of GAMs and used numerical methods
47	Develop a deep theoretical unterstanding of GAMs and used numerical methods
48	Set up an Apache Spark Enviroment
49	Start implementing
50	Implementation
51	Winter break
52	Winter break
1	Imlementation and Exams
2	Imlementation and Exams
3	Imlementation and Exams
4	Imlementation and Exams
5	Break
6	Finish implementation
7	Start writing
8	writing
9	writing
10	writing
11	writing
12	writing

5 References

[Hall, 2009] Hall, D. (2009). Breeze.

[Strang, 2009] Strang, G. (2009). *Linear Algebra and Its Applications*. Wellesley-Cambridge Press, Wellesley, MA.

[Wood, 2006] Wood, S. (2006). *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC.

[Wood et al., 2015] Wood, S. N., Goude, Y., and Shaw, S. (2015). Generalized additive models for large data sets. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 64(1):139–155.

[Zaharia et al., 2010] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA. USENIX Association.