Fitting Generalized Additive Models for very large datasets with Apache Spark

An exposé for a bachelor thesis

Kai Thomas Brusch

05.11.2015

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 **3**
1.1 Context . . . 3

2 Content **3**
2.1 General Linear Models . . . 3
2.2 Generalized Linear Models . . . 4
2.3 Generalized Additive Models . . . 4
2.4 Smoothing parameter Estimation with generalized cross validation . . . 6
2.5 Generalized Additive Models for large data sets . . . 6
2.6 Apache Spark . . . 7
2.7 General additive models in Apache Spark . . . 7

3 Guiding Questions **8**
3.1 What is novel in this approach? . . . 8
3.2 When is the thesis regarded as completed? . . . 8
3.3 Which tasks are realistic to be accomplished within the scope of this thesis? . . . 8

4 Time table **8**

5 References **9**

1 Introduction

1.1 Context

Regression analysis stands at the heart of modern statistical analysis. Discovering the relationship between dependent and independent variables offers great insight into the underlying system and can even provide far reaching predictive capabilities. The traditional linear models explain the dependent variable as the sum of the independent variables and the linear interaction with an estimated coefficent.[Wood, 2006] describes the generalized additive model (GAM) as a powerful extension of the generalized linear model, explaining the relationship between dependent and independent variables as the sum of smoothing functions. This modeling approach offers more flexibility than linear models and has been successfully established in academia and industry. However, the additional flexibility of GAMs comes at the cost of additional computational and memory requirements, placing strict constraints on the model and data size.[Wood et al., 2015] suggest two optimizations to eliviate these constraints: parallelization and numerical optimization. These optimizations have been implemented and used in high-level statistical languages such as R, SPSS and Stata. While these implementations already allow for fitting of larger models with more data they have not 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 environment to implement the suggested optimizations by [Wood et al., 2015]. Ideally proving better than previous implementations.

2 Content

2.1 General Linear Models

Linear models are statistical models in which an univariate response is modeled 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.

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

Equation 1 introduces formal notation on how to describe the dependent 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 yields $\widehat{\beta}$ which is the best possible estimation for b. 1 represents the univariate case where y is explained with only one variable. This model can be extended to multiple independent variables yielding in a scalar matrix form.

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

The dependent variable vector y is the linear combination of model matrix X and the vector of unknown parameter β. Finding the best possible estimator $\widehat{\beta}$ for 3 is formalized stated as the minimal length of the Euclidean distance:

$$\|y - X\beta\|^2 \tag{4}$$

Minimizing 4 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 4.

2.2 Generalized Linear Models

Generalized linear models (GLMs) 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 (normal, Poisson, binomial, gamma etc.).

$$g(\mu_{\mathrm{i}}) = \mathbf{X}\beta \tag{5}$$

Equation 5 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.

2.3 Generalized Additive Models

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

- 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.

GAMs are fomally described by the following equation:

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

6 explains y_i as the model matrix for this row and the smooth functions $f_j(x_1 j)$ 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 influence per explanatory variable. The natural question that arises now are: How do I find proper smoothing functions? 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 easily be fund by specifying 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 7 can be rewriten 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 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 disadvantages. A common choice however is a fourth order polynomial 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 modeling of y_i 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 does a fine job of illustrating the used concepts but enforce rather strict limitations. 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 knots
- Each cubic polynomial must match it's neighbors 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 a good polynmial. The number and location of knots will influence the fit of our splines and are carefully chosen. The right number and location of knots can be estimated but are usually a problem speicifc design decision. After establishing how to represent the smooth functions and the use of knots we will now turn to the smoothing parameter λ. λ is a penalty factor that controls 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 derivative 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 a more concise form.

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

12 shows that describing an unknown variable as the sum of smooth functions can be restated as the minimization of penalized regression splines and hence becomes problem of minimizing equation 12. While the the problem of finding a smooth function for a univariate function has almost been solved by 12 we are still left with an unknown parameter of λ. Finding the optimal $\lambda$is subject of the following section.

2.4 Smoothing parameter Estimation with generalized cross validation

Finding a good smoothing parameter through minimizing 12 the essential question for GAM research. Finding an 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

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

13 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 12 as the full process of finding an 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 12:

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

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 trial λ. Iterating 14 to obtain the new estimates can place a heavy burden on the memory requirements. Each step of the PIRLS requires to form the whole model matrix X. 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 discover and implement the in [Wood et al., 2015] suggested optimizations.

2.5 Generalized Additive Models for large data sets

[Wood et al., 2015] introduces two extensions of the development methods for fitting GAMs. Woods methods aim at numerical optimization and parallelization to eliviate the rescrictions on data size.

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 stategy on how model matrix factorization can be used to avoid formation of the whole model matrix in each iteration.

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 $f = Q^T y$ and then expression 12 becomes

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

.

15 can be rewritten in a form that does not require the full model matrix X.

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

.

16 brings a tramendous advantage over 13 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 13.

The parallelization optimimization of 13 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. [Zaharia et al., 2010] advertises Spark as a fast and general cluster computing engine for large-scale data processing. Here 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 environment 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 available in a cluster.

2.7 General additive models in Apache Spark

I strongly believe that Apache Spark offers an ideal environment 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 library 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 effect performance?

3.1 What is novel in this approach?

The suggested optimizations have been implemented in high-level statistical languages, proving its effectiveness. The suggested methods have not been implemented in an environment 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 environment and hence should be discovered

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 environment.
- Familiarize with Breeze in Apache Spark.
- Coordinate with FSB and my employer to request my employer as second examiner

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 methods
46	Develop a deep theoretical unterstanding of GAMs and used methods
47	Develop a deep theoretical unterstanding of GAMs and used methods
48	Set up an Apache Spark Environment
49	Start implementing matrix decompositions with Breeze in Apache Spark
50	Continue implementing matrix decompositions
51	Winter break
52	Winter break
1	Implementation and Exams
2	Implementation and Exams
3	Implementation and Exams
4	Implementation and Exams
5	Post Exams Break
6	Start implemeting GAM fitting
7	Implementation of GAM fitting
8	Implementation of GAM fitting and run bechmark tests
9	Run benchmarks tests
10	Buffer
11	writing bachelor thesis
12	writing bachelor thesis
13	writing bachelor thesis
14	writing bachelor thesis
15	writing bachelor thesis
16	Done

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.