KIM LARSEN

July 30, 2015 - San Francisco, CA

Tweet this post! Post on LinkedIn

# Introduction

Imagine that you step into a room of data scientists; the dress code is casual and the scent of strong coffee is hanging in the air. You ask the data scientists if they regularly use generalized additive models (GAM) to do their work. Very few will say yes, if any at all.

Now let's replay the scenario, only this time we replace GAM with, say, random forest or support vector machines (SVM). Everyone will say yes, and you might even spark a passionate debate.

Despite its lack of popularity in the data science community, GAM is a powerful and yet simple technique. Hence, the purpose of this post is to convince more data

scientists to use GAM. Of course, GAM is no silver bullet, but it is a technique you should add to your arsenal. Here are three key reasons:

- Easy to interpret.
- Flexible predictor functions can uncover hidden patterns in the data.
- Regularization of predictor functions helps avoid overfitting.

In general, GAM has the interpretability advantages of GLMs where the contribution of each independent variable to the prediction is clearly encoded. However, it has substantially more flexibility because the relationships between independent and dependent variable are not assumed to be linear. In fact, we don't have to know a priori what type of predictive functions we will eventually need. From an estimation standpoint, the use of regularized, nonparametric functions avoids the pitfalls of dealing with higher order polynomial terms in linear models. From an accuracy standpoint, GAMs are competitive with popular learning techniques.

In this post, we will lay out the principles of GAM and show how to quickly get up and running in R. We have also put together a PDF that gets into more detail around smoothing, model selection and estimation.

All code and data used for this post can be downloaded from this Github repo: https://github.com/klarsen1/gampost.

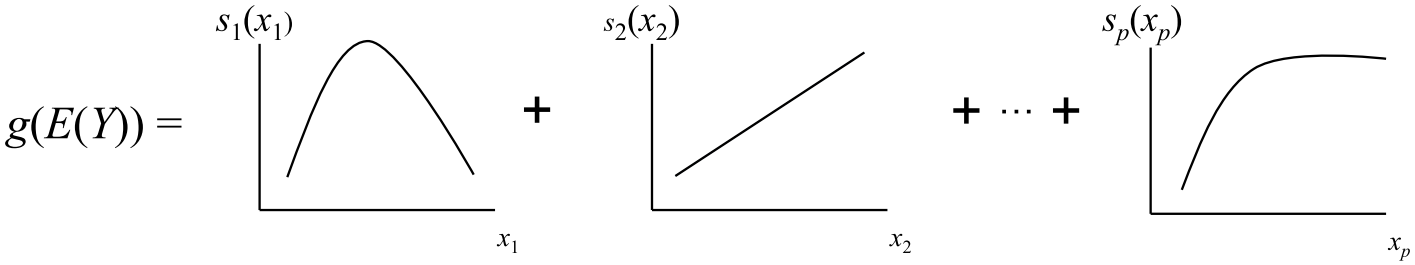
# What is GAM?

Generalized additive models were originally invented by Trevor Hastie and Robert Tibshirani in 1986 (see [1], [2]). The GAM framework is based on an appealing and

simple mental model:

- Relationships between the individual predictors and the dependent variable follow smooth patterns that can be linear or nonlinear.
- We can estimate these smooth relationships *simultaneously* and then predict $g(E(Y))$ by simply adding them up.

Mathematically speaking, GAM is an additive modeling technique where the impact of the predictive variables is captured through smooth functions which—depending on the underlying patterns in the data—can be nonlinear:




We can write the GAM structure as:

$$g(E(Y)) = \alpha + s_1(x_1) + \cdots + s_p(x_p),$$

where $Y$ is the dependent variable (i.e., what we are trying to predict), $E(Y)$ denotes the expected value, and $g(Y)$ denotes the *link function* that links the expected value to the predictor variables $x_1, \ldots, x_p$.

The terms $s_1(x_1), \ldots, s_p(x_p)$ denote smooth, *nonparametric* functions. Note that, in the context of regression models, the terminology *nonparametric* means that the shape of predictor functions are fully determined by the data as opposed to *parametric* functions that are defined by a typically small set of parameters. This can allow for more flexible estimation of the underlying predictive patterns without knowing upfront what these patterns look like. For more details on how to create these smooth functions, see the section called “Splines 101” in the PDF.

Note that GAMs can also contain parametric terms as well as two-dimensional smoothers. Moreover, like generalized linear models (GLM), GAM supports multiple link functions. For example, when $Y$ is binary, we would use the logit link given by

$$g(E(Y)) = \log \frac{P(Y=1)}{P(Y=0)}.$$

# Why Use GAM?

As mentioned in the intro, there are at least three good reasons why you want to use GAM: interpretability, flexibility/automation, and regularization. Hence, when your model contains nonlinear effects, GAM provides a regularized and interpretable solution – while other methods generally lack at least one of these three features. In other words, GAMs strike a nice balance between the interpretable, yet biased, linear model, and the extremely flexible, "black box" learning algorithms.

## Interpretability

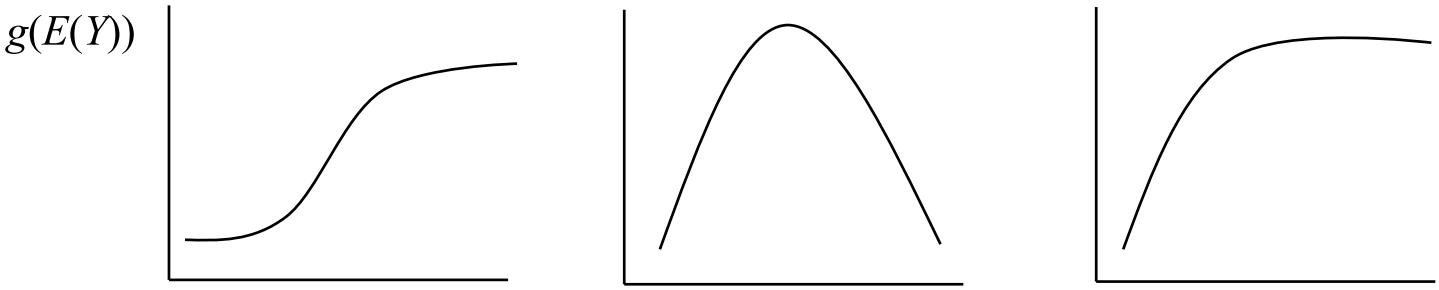
When a regression model is additive, the interpretation of the marginal impact of a single variable (the partial derivative) does not depend on the values of the other variables in the model. Hence, by simply looking at the output of the model, we can make simple statements about the effects of the predictive variables that make sense to a nontechnical person. For example, for the graphic illustration above, we can say that the (transformed) expected value of $Y$ increases linearly as $x_2$ increases, holding everything else constant. Or, the (transformed) expected value

of $Y$ increases with $x_p$ until $x_p$ hits a certain point, etc.

In addition, an important feature of GAM is the ability to control the smoothness of the predictor functions. With GAMs, you can avoid wiggly, nonsensical predictor functions by simply adjusting the level of smoothness. In other words, we can impose the prior belief that predictive relationships are inherently smooth in nature, even though the dataset at hand may suggest a more noisy relationship. This plays an important role in model interpretation as well as in the believability of the results.

## Flexibility and Automation

GAM can capture common nonlinear patterns that a classic linear model would miss. These patterns range from "hockey sticks" – which occur when you observe a sharp change in the response variable – to various types of "mountain shaped" curves:




When fitting *parametric* regression models, these types of nonlinear effects are typically captured through binning or polynomials. This leads to clumsy model formulations with many correlated terms and counterintuitive results. Moreover, selecting the best model involves constructing a multitude of transformations, followed by a search algorithm to select the best option for each predictor – a potentially greedy step that can easily go awry.

We don't have this problem with GAM. Predictor functions are automatically derived *during* model estimation. We don't have to know up front what type of functions we will need. This will not only save us time, but will also help us find patterns we may have missed with a parametric model.

Obviously, it is entirely possible that we can find parametric functions that look like the relationships extracted by GAM. But the work to get there is tedious, and we do not have 20/20 hindsight prior to model estimation.

## Regularization

As mentioned above, the GAM framework allows us to control smoothness of the predictor functions to prevent overfitting. By controlling the wiggliness of the predictor functions, we can directly tackle the bias/variance tradeoff. Moreover, the type of penalties applied in GAMs have connections to Bayesian regression and $l_2$ regularization (see the PDF for details).

In order to see how this works, let's look at a simple, simulated example in R. We are simulating a dataset with 100 data points and two variables, $x$ and $Y$. The *true* relationship between $x$ and $Y$ follows the sine function, but our data has normally distributed random errors.

```
set.seed(3)
x <- seq(0,2*pi,0.1)
z <- sin(x)
y <- z + rnorm(mean=0, sd=0.5*sd(z), n=length(x))
d <- cbind.data.frame(x,y,z)
```

We want to predict $Y$ given $x$ by fitting the simple model:

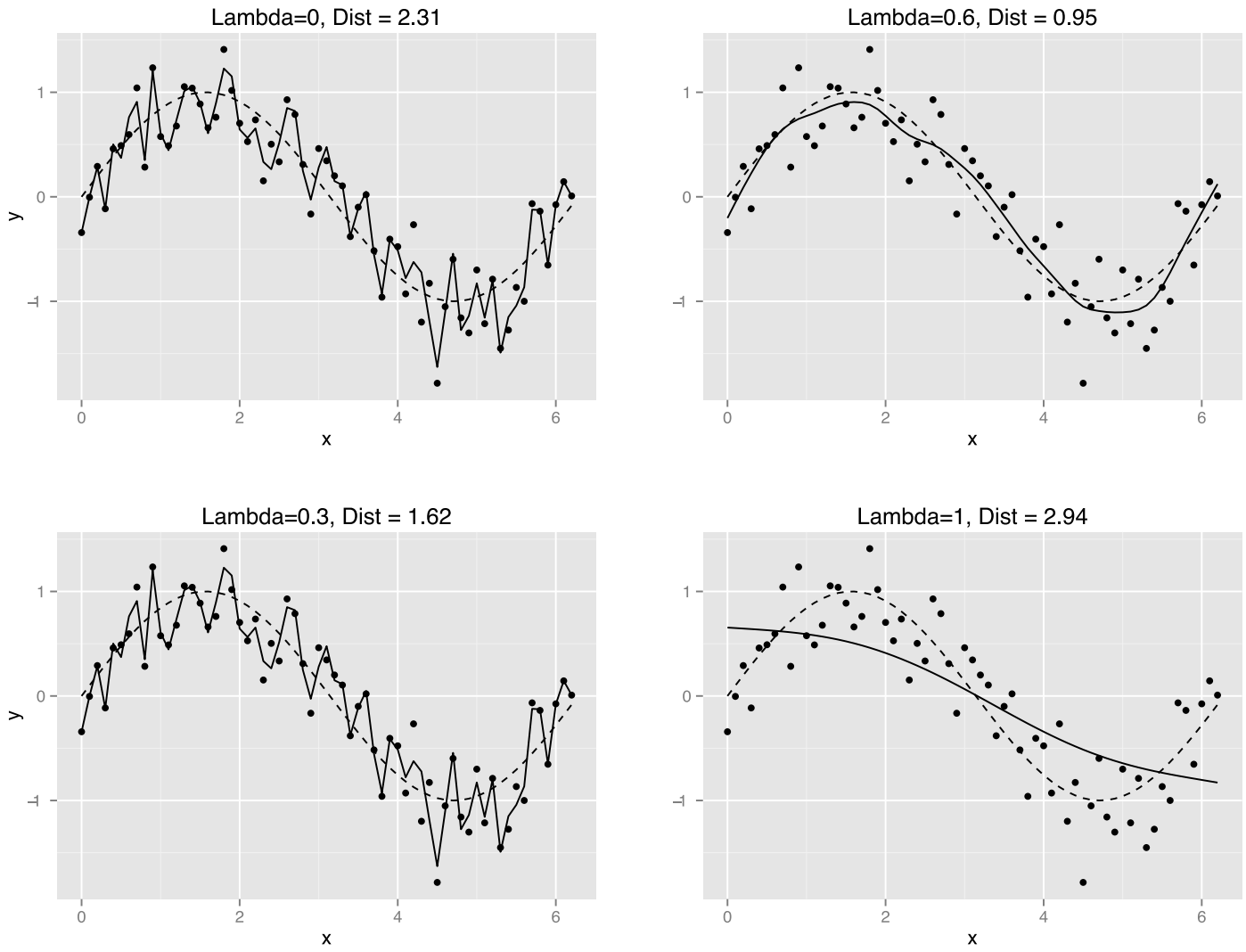
$$y = s_\lambda(x) + e,$$

where $s_\lambda(x)$ is some smooth function. The level of smoothness is determined by the *smoothing parameter*, which we denote by $\lambda$. The higher the value of $\lambda$, the smoother the curve. In the PDF, you can find a more details on how $\lambda$ works to create smoothness as well as how to estimate $s(x)$. But for now, let's just think of $s(x)$ as a smooth function. For more details on smoothers, see the section called "Splines 101" in the PDF.

We fit the model above to the simulated data with four different values for $\lambda$. For each value of $\lambda$, we calculated the distance against the true function (the underlying sine curve). The results are shown in the charts below. The dots represent the actual data points, the punctuated line is the true curve, and the solid line is this smoother.

Clearly, the model with $\lambda = 0$ provides the best fit of the data, but the resulting curve looks very wiggly and would be hard to explain. Moreover, it has the highest distance to the sine curve, which means that it does not do a good job of capturing the true relationship. Indeed, the best choice in this case seems to be some intermediate value, like $\lambda = 0.6$.

Notice how the smoothing parameter allows us to *explicitly* balance the bias/variance tradeoff; smoother curves have more bias (in-sample error), but also less variance. Curves with less variance tend to make more sense and validate better in out-of-sample tests. However, if the curve is too smooth, we may miss an important pattern.




# Estimating GAMs

As mentioned in the intro, GAMs consist of *multiple* smoothing functions. Thus, when estimating GAMs, the goal is to *simultaneously* estimate all smoothers, along with the parametric terms (if any) in the model, while factoring in the covariance between the smoothers. There are two ways of doing this:

- Local scoring algorithm.

- Solving GAM as a large GLM with penalized iterative reweighted least squares (PIRLS).

For details on GAM estimation, see the "Estimation" section in the PDF.

In general, the local scoring algorithm is more flexible in the sense that you can use any type of smoother in the model whereas the GLM approach only works for regression splines (see the "Smoothing 101" section in the PDF). However, the local scoring algorithm is computationally more expensive and it does not lend itself as nicely to automated selection of smoothing parameters as the GLM approach.

## Penalized Likelihood

For both local scoring and the GLM approach, the ultimate goal is to maximize the penalized likelihood function, although they take very different routes. The penalized likelihood function is given by

$$2l(\alpha, s_1(x_1), \ldots, s_p(x_p)) - \text{penalty},$$

where $l(\alpha, s_1, \ldots, s_p)$ is the standard log likelihood function. For a binary GAM with a logistic link function, the penalized likelihood is defined as

$$l(\alpha, s_1(x_1), \ldots, s_p(x_p)) = \sum_{i=1}^{n} (y_i \log \hat{p_i} + (1 - y_i) \log(1 - \hat{p_i})).$$

$$\hat{p_i} = \left(1 + \exp(-\hat{\alpha} - \sum_{j=1}^{p} s_j(x_{ij}))\right)^{-1}.$$

where $\hat{p_i}$ is given by

$$\hat{p_i} = P(Y = 1|x_1, \ldots, x_p) = \left(1 + \exp(-\hat{\alpha} - \sum_{j=1}^{p} s_j(x_{ij}))\right)^{-1}.$$

The penalty can, for example, be based on the second derivatives

$$\text{penalty} = \sum_{j=1}^{p} \lambda_j \int (s_j''(x_j))^2 dx.$$

The parameters, $\lambda_1, \ldots, \lambda_p$, are the aforementioned *smoothing parameters* which control how much penalty (smoothness) we want to impose on the model. The higher the value of $\lambda_j$, the smoother the curve. These parameters can be preselected or trained from the data. See the "Estimation" section of the PDF for more details.

Intuitively, this type of penalty function makes sense: the second derivative measures the slopes of the slopes. This means that wiggly curve will have large second derivatives, while a straight line will have second derivatives of 0. Thus we can quantify the total wiggliness by "adding up" the squared second derivatives.

## Choosing the Smoothing Parameters

When fitting a GAM, the choice of *smoothing parameters* – i.e., the parameters that control the smoothness of the predictive functions – is key for the aesthetics and fit of the model. We can choose to pre-select the smoothing parameters or we may choose to estimate the smoothing parameters from the data. There are two ways of estimating the smoothing parameter for a logistic GAM:

- Generalized cross validation criteria (GCV).
- Mixed model approach via restricted maximum likelihood (REML).

REML only applies if we are casting GAM as a large GLM. Generally the REML approach converges faster than GCV, and GCV tends to under-smooth (see [3], [9]). For more details, see the "Estimation" section of the PDF.

# Fitting GAMs in R

The two main packages in R that can be used to fit generalized additive models are gam and mgcv. The gam package was written by Trevor Hastie and closely follows the theory outlined in [2]. The mgcv package was written by Simon Wood, and, while it follows [2] in many ways, it is much more general because it considers GAM to be any penalized GLM (for more details, see [3]).

The differences are described in detail in the documentation for mgcv. Here is a cheat sheet:

| Component | gam | mgcv |
|---|---|---|
| Confidence intervals | Frequentist | Bayesian |
| Splines | Smoothing splines and loess | Does not support loess or smoothing splines, but supports a wide array of regression splines (P-splines, B-splines, thin plate splines, tensors) + tensors |
| Parametric terms | Supported | Supported, and you can penalize or treat as random effects |
| Variable selection | Stepwise selection | Shrinkage |
| Optimization | Local scoring | PIRLS |
| Selecting smoothing parameters | No default approach | Finds smoothing parameters by default. Supports both REML and GCV |
| Large datasets | Can parallelize stepwise variable selection with the doMC package | Special bam function for large datasets. Can also parallelize certain operations in the gam function through openMP |
| Missing values | Clever approach to dealing with missing values through na.action=gam.replace | No special treatment. Omits observations with missing values |
| Multi dimensional smoothers | Supported with loess | Supported with tensors and thin plate splines |
| Model diagnostics | Standard GAM diagnostics | Standard GAM diagnostics + the concurvity measure which is a generalization of collinearity |

gam and mgcv do not work well when loaded at the same time. Restart the R session if you want to switch between the two packages – detaching one of the packages is not sufficient.

Here is an example of how to fit a GAM in R:

```
### GAM example using mgcv

library(mgcv)
library(ggplot2)
```

```
# fake data
n <- 50
sig <- 2
dat <- gamSim(1,n=n,scale=sig)

# P-spline smoothers (with lambda=0.6) used for x1 and x2; x3 is parametric.
b1 <- mgcv::gam(y ~ s(x1, bs='ps', sp=0.6) + s(x2, bs='ps', sp=0.6) + x3, data = d
summary(b1)
plot(b1)


# plot the smooth predictor function for x1 with ggplot to get a nicer looking gr
p <- predict(b1, type="lpmatrix")
beta <- coef(b1)[grepl("x1", names(coef(b1)))]
s <- p[,grepl("x1", colnames(p))] %*% beta
ggplot(data=cbind.data.frame(s, dat$x1), aes(x=dat$x1, y=s)) + geom_line()


# predict
newdf <- gamSim(1,n=n,scale=sig)
f <- predict(b1, newdata=newdf)


# select smoothing parameters with REML, using P-splines
b2 <- mgcv::gam(y ~ s(x1, bs='ps') + s(x2, bs='ps') + x3, data = dat, method="REML

# select variables and smoothing parameters
b3 <- mgcv::gam(y ~ s(x0) + s(x1) + s(x2) + s(x3) , data = dat, method="REML", se

# loess smoothers with the gam package (restart R before loading gam)
library(gam)
b4 <- gam::gam(y ~ lo(x1, span=0.6) + lo(x2, span=0.6) + x3, data = dat)
summary(b4)
```

# Comparing GAM Performance With Other Techniques

## Business Problem

We will be using a marketing example from the insurance industry (source undisclosed). The data contains information on customer responses to a historical direct mail marketing campaign. Our goal is to improve the performance of future waves of this campaign by targeting people who are likely to take the offer. We will do this by building a "look-alike" model to predict the probability that a given client will accept the offer, and then use that model to select the target audience going forward [1f].

Obviously, we want a model that is accurate so that we can find the best possible target audience. In addition, we want to be able to provide insights from the model, such as *partial impact* charts, that show how the average propensity changes across various client features. We want to make sure that the relationships we find stand to reason from a business perspective.

## Data

The dataset has 68 predictive variables and 20k records. For modeling and validation purposes, we split the data into 2 parts:

- 10k records for training. This dataset will be used to estimate models.
- 10k records for testing. This dataset will be kept in a vault to the very end and used to compare models.

The success of the model will be based on its ability to predict the probability that the customer takes the offer (captured by the PURCHASE indicator), for the validation dataset.

Most variables contain credit information, such as number of accounts, active account types, credit limits, and utilization. The dataset also captures the age and

location of the individuals.

Let's return to our marketing case study. Recall that we are trying to predict whether a person takes a direct marketing offer. Hence, we are trying to build a GAM that looks like this:

$$\log \frac{P(\text{convert})}{1 - P(\text{convert})} = s_1(x_1) + \cdots + s_p(x_p) + x'\beta$$

where $x'\beta$ are parametric terms (dummy variables in our case).

## Model Comparison Strategy

We built six models with six different techniques using the training dataset. The models were then validated against the validation dataset. The area under the ROC curve was used to evaluate model performance.

In order to make the comparison as fair as possible, we used the same set of variables for each model. The variables were selected using the following procedure:

1. Remove all variables with an information value (IV) less than 0.05. (See the PDF for more details on IV.). You can use the Information Package to calculation information values.
2. Eliminate highly correlated variables using variable clustering (ClustOfVar package). We generated 20 clusters and picked the variable with the highest IV within each cluster.

Obviously, we could have used variable selection techniques inside GAM as described above, but we wanted to use the same 20 variables for each model.

List of the seven models tested:

1. Random forest with 100 trees using the openMP enabled randomForestSRC package.
2. GAM (mgcv) using P-splines with smoothing parameters of 0.6 for all variables (except dummy variables).
3. Same as #2, but optimal smoothing parameters are selected with REML (instead of using 0.6 for all variables).
4. Same as #2, but optimal smoothing parameters are selected with REML (see the PDF for details) and weak variables are shrunk towards 0 using selection=TRUE in mgcv.
5. SVM built with the e1071 package, using a Gaussian radial kernel.
6. KNN classifier with k=100. Distance metrics were weighted using an Epanechnikov kernel. See the kknn package for more details.
7. Linear logistic regression model.

Note, all the code and data used for this post can be downloaded from this Github repo: https://github.com/klarsen1/gampost.

## Testing Results

| Model | Validation AUROC | Estimation Time | Scoring Time |
|---|---|---|---|
| Random forest | 0.809 | 6.39 | 39.38 |
| GAM, lambda=0.6 | 0.807 | 3.47 | 0.52 |
| GAM, estimate lambdas | 0.815 | 42.72 | 0.29 |
| GAM, estimate lambdas, extra shrinkage | 0.814 | 169.73 | 0.33 |
| SVM | 0.755 | 13.41 | 1.12 |
| Linear logit | 0.800 | 0.1 | 0.006 |
| KNN with K=100 | 0.800 | NA | 3.34 |

Note that in order to get the AUROC for the SVM, we used the enhanced version of Platt's method to convert class assignments into probabilities (to get a continuous measure, see [11]). Settings for KNN and SVM were based on trying different combinations.

As we can see, GAM performs well compared to the other methods. Obviously, this test is based on a single dataset, so no universal conclusions can be drawn, but the dataset has enough correlation and "chunky" variables to make the results relevant.

The GAM models where smoothing parameters were automatically selected with REML perform better than the model where we used a flat smoothing parameter of 0.6 across all variables (which tends to work well for most models). However, in this example, the models with automatic selection also tend to produce more wiggly functions than the model with $\lambda = 0.6$ across all variables. For a targeting model, the additional wiggliness is not worth the loss of model intuition.

The biggest surprises in this test are the performances of SVM and the linear logit model. The linear logit model is performing surprisingly well given that the strongest variable in the model (N_OPEN_REV_ACTS) is not linearly correlated with the log odds of success (PURCHASE). The reason could be that, while this relationship is not linear, it is monotonic. Also, the AUROC is based on the

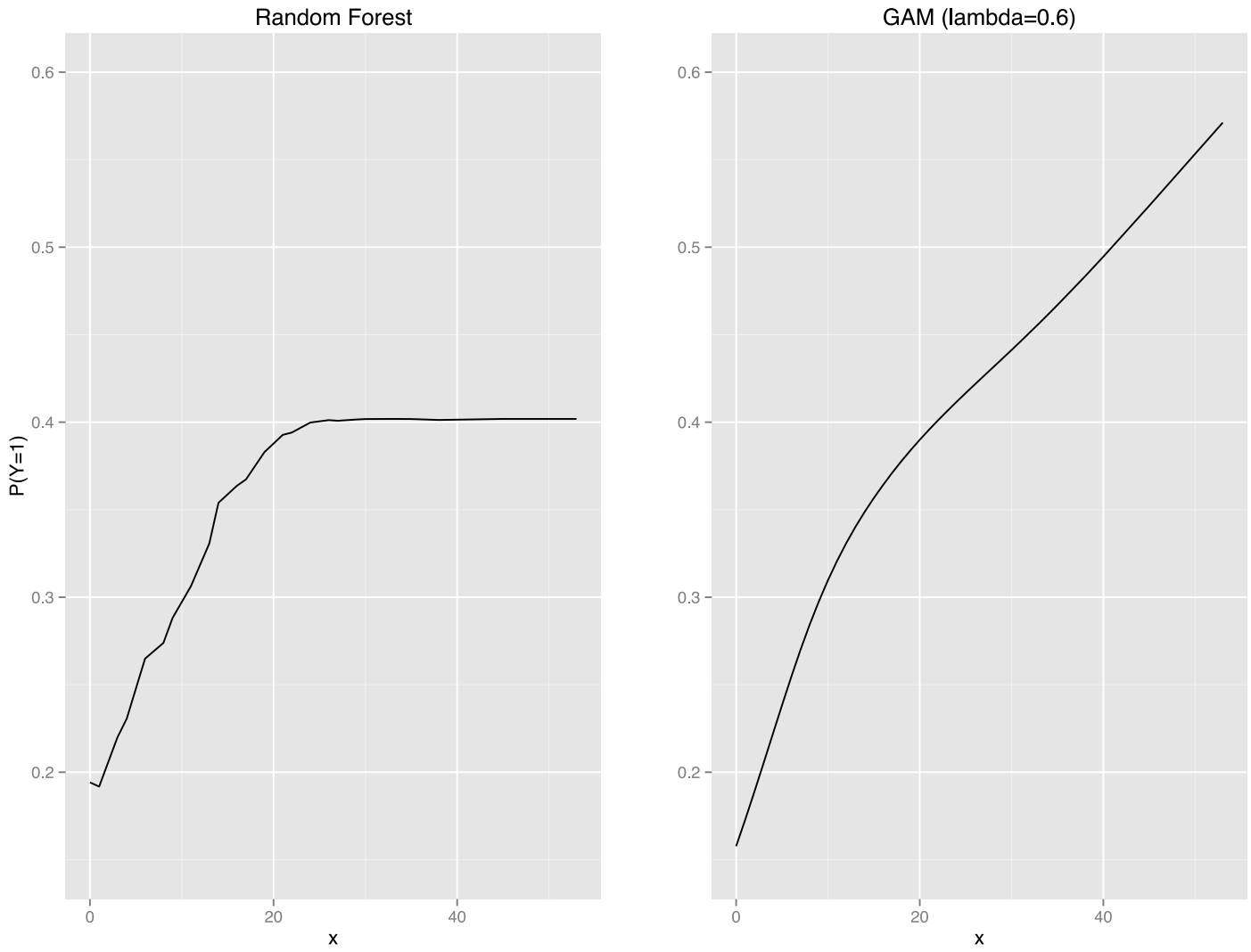
estimated probability, which is indeed not linear in the predictive variables due to the Sigmoidal transformation $(1 + exp(-\nu))^{-1}$. SVM on the other hand, is performing surprisingly poorly. However, it should be mentioned that the author of this post has very little experience with SVM, which could be a disadvantage for SVM. Also, the Pratt conversion of SVM classification to probabilities could play a role here.

## Partial Relationships

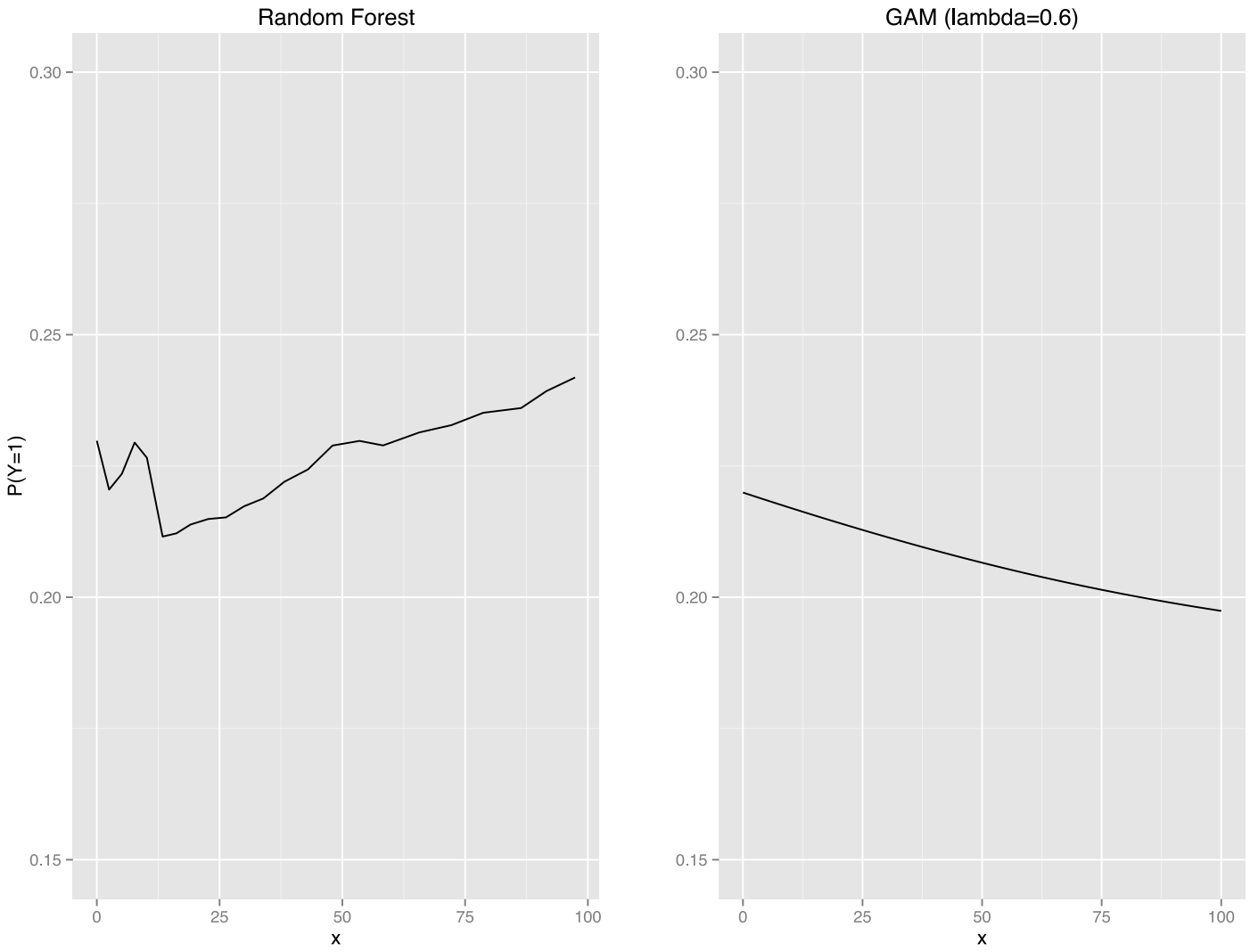
As stated earlier, an important part of this modeling exercise is to look at the partial relationships between the binary dependent variable (PURCHASE) and the predictors.

This is shown below for the variable N_OPEN_REV_ACTS (number of open revolving accounts) for random forest and GAM. Note that for the random forest model, these plots are generated by sending different values of $x_j$ (in our case 20) through the forest and getting the estimated probabilities at each value of $x_j$. For GAM, we simply plot the final regression spline.

Note that, unlike GAM, random forest does not try to promote smoothness. This clearly shows in the chart below, as the GAM-based predictive function is smoother than the one from random forest. However, the GAM model does some potentially dangerous interpolation beyond $x = 20$ where the data is thin. (only 1.64% of the sample have N_OPEN_REV_ACTS>20 although the conversion rate for this group is 2.3 times higher than the average).




And here are the partial impact plots for one the weakest variables. The random forest curve does not look very intuitive:




# Final Words

As stated in the introduction, the purpose of this post is to get more data scientists to use GAM. Hopefully, after reading this post, you'll agree that GAM is a simple, transparent, and flexible modeling technique that can compete with other popular methods. The code in the github repository should be sufficient to get started with GAM.

Of course, GAM is no silver bullet; one still needs to think about what goes into the model to avoid strange results. In fact, random forest is probably the closest thing to a silver bullet. However, random forest is much more of a black box, and you cannot control smoothness of the predictor functions. This means that you cannot combat the bias variance tradeoff as directly as with GAMs or ensure interpretable predictor functions. For those reasons, every data scientist should make room in their toolbox for GAM.

# References

[1] Hastie, Trevor and Tibshirani, Robert. (1990), Generalized Additive Models, New York: Chapman and Hall.

[2] Hastie, Trevor and Tibshirani, Robert. (1986), Generalized Additive Models, Statistical Science, Vol. 1, No 3, 297-318.

[3] Wood, S. N. (2006), Generalized Additive Models: an introduction with R, Boca Raton: Chapman & Hall/CRC

[4] Wood, S. N. (2004). Stable and efficient multiple smoothing parameter estimation for generalized additive models. Journal of the American Statistical Association 99, 673–686

[5] Marx, Brian D and Eilers, Paul H.C. (1998). Direct generalized additive modeling with penalized likelihood, Computational Statistics & Data Analysis 28 (1998) 193-20

[6] Sinha, Samiran, A very short note on B-splines, http://www.stat.tamu.edu/~sinha/research/note1.[PDF](/assets/files/gam.pdf)

[7] German Rodrıguez (2001), Smoothing and Non-Parametric Regression, http://data.princeton.edu/eco572/smoothing.pd

[8] Notes on GAM By Simon Wood. http://people.bath.ac.uk/sw283/mgcv/tampere/gam.[PDF](/assets/files/gam.pdf)

[9] Notes on Smoothing Parameter Selection By Simon Wood, http://people.bath.ac.uk/sw283/mgcv/tampere/smoothness.[PDF] (/assets/files/gam.pdf)

[10] Notes on REML & GAM By Simon Wood, http://people.bath.ac.uk/sw283/talks/REML.[PDF](/assets/files/gam.pdf)

[11] Karatzoglou, Alexandros, Meyer, David and Hornik, Kurt (2006), Support Vector Machines in R, Journal of Statistical Software Volume 15, Issue 9, http://www.jstatsoft.org/v15/i09/paper

[12] “e1071” package, https://cran.r-project.org/web/packages/e1071/e1071.[PDF] (/assets/files/gam.pdf)

[13] “mgcv” package, https://cran.r-project.org/web/packages/mgcv/mgcv.[PDF] (/assets/files/gam.pdf)

[14] “gam” package, https://cran.r-project.org/web/packages/gam/gam.[PDF] (/assets/files/gam.pdf)

[15] “randomForestSRC” package, https://cran.r-project.org/web/packages/randomForestSRC/randomForestSRC.[PDF] (/assets/files/gam.pdf)

[1f] When we target clients with the highest propensity, we may end up preaching to the choir as opposed to driving uplift. But that is beyond the scope of this post.

in Tweet this post! in Post on LinkedIn


# COME WORK WITH US!

We're a diverse team dedicated to building great products, and we'd love your help. Do you want to build amazing products with amazing peers? Join us!

All Careers at Stitch Fix

STITCH FIX
Your partner in personal style

Stitch Fix and Fix are trademarks of Stitch Fix, Inc.

Stitch Fix Home
FAQ
Press

Maternity
Big and tall
Jeans
Business Casual
Petite
Plus

Tech Blog
Tech Careers
Terms of Use
Privacy Policy

Follow Us!

Follow Us!

TECH BLOG
TECH CAREERS
STITCH FIX HOME
FAQ
PRESS
TERMS OF USE
PRIVACY POLICY

Stitch Fix and Fix are trademarks of Stitch Fix, Inc.