A Comparison of Basis Expansions in Regression

Beginners in machine learning often write off regression methods after learning of more exotic algorithms like Boosting, Random Forests, and Support Vector machines; why bother with a *linear* method when powerful non-parametric methods are readily avalable?

It's easy to point out that the *linear* in linear regression is not meant to convey that the resulting model predictions are linear in the raw *feaure*, just the estimated parameters! The modeler can certainly capture non-linearities in thier regression, they only need to transform the raw predictors!

A common response is that it is error prone and annoying work to explore data by hand and somehow divine correct transformations of predictors: other methods do it automatically.

Ususally the only truly flexible method beginners learn to capture non-linearities in regression is polynomial regression, which is a real shame, as it is about the worst performing method available.

The purpose of this post is to spread awareness of better options for capturing non-lineararities in regression models, we would like to advocate more widespread adoption of linear and cubic splines.

Acknowledgements

The idea for this post is based on my answer to hxd1011 (https://stats.stackexchange.com/users/113777/hxd1011)s question regarding grouping vs. splines (https://stats.stackexchange.com/questions/230750/when-should-we-discretize-bin-continuous-independent-variables-features-and-when) at CrossValidated.

Software

I've taken the oppurtunity to write a small python module (https://github.com/madrury/basis-expansions) that is useful for using the basis expansions in this post here. It conforms to the sklearn transfrmation interface, so can be used in pipelines and other high level processes in sklearn.

Basis Expansions in Regression

To capture non-linearities in regression models, we need to transform some or all of the predictors. To avoid having to treat every predictor as a special case needing investigation, we would like some way of applying a very general *family* of transformations to our predictors, which is flexible enough to adapt (when the model is fit) to a wide variety of shapes.

This takes the general form of a *basis expansion*. Basis here is used in the linear algebraic sense: a linearly independent set of objects. In this case our objects are *functions*:

$$B = f_1, f_2, \ldots, f_k$$

and we create new sets of features by applying every function in our basis to the given feature:

$$f_1(x),\ f_2(x),\ \ldots,\ f_k(x)$$

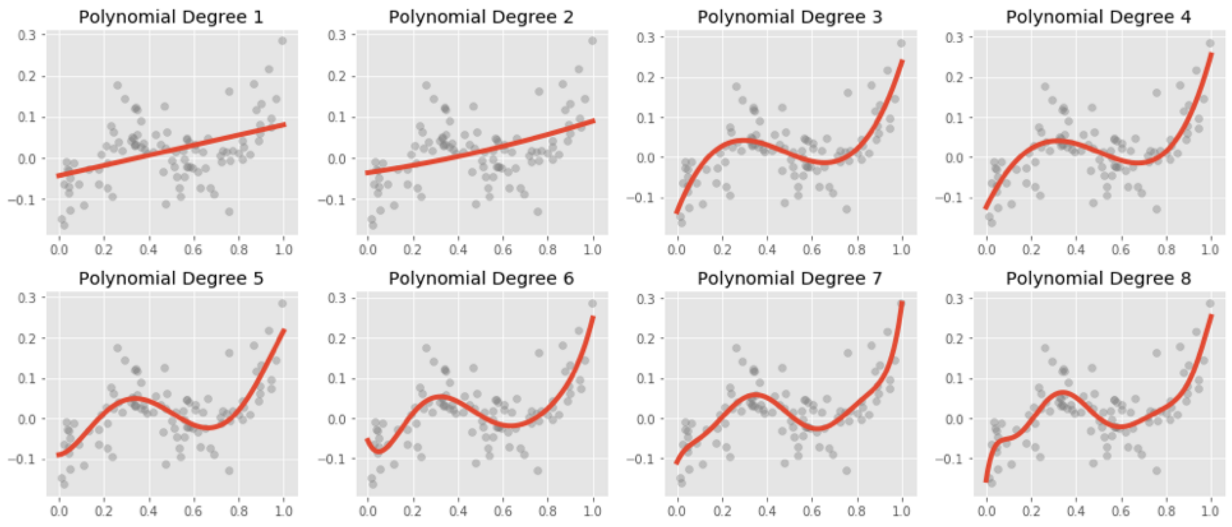
Polynomial Expansion

The most commonly, and often *only*, example taught in introductory modeling courses or textbooks is **polynomial regression**.

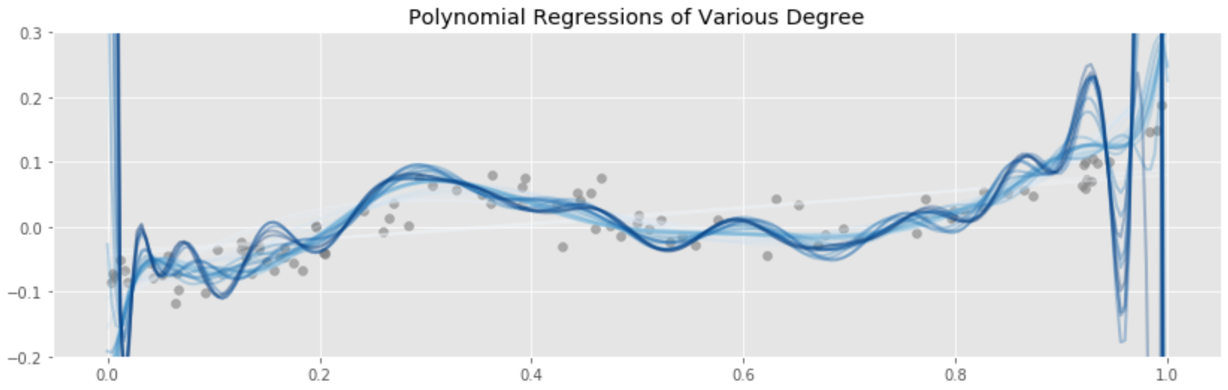
In polynomial regression we choose as our basis a set of polynomial terms of increasing degree:

$$f_1(x) = x,\ f_2(x) = x^2,\ \ldots,\ f_r(x) = x^r$$

This allows us to fit *polynomial* curves to features:

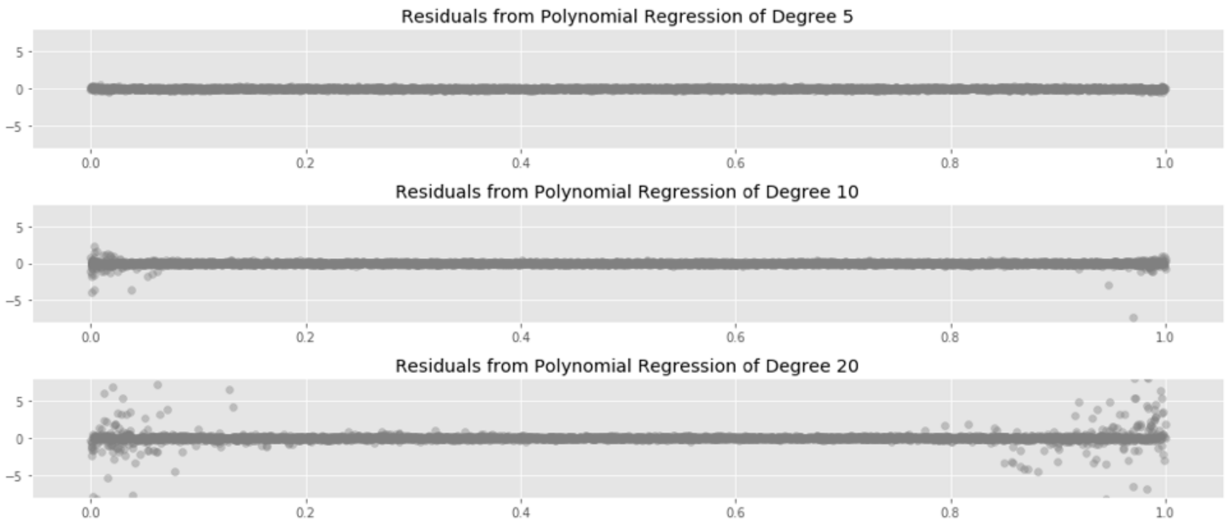


Unfortunately, polynomial regression has a fair number of issues. The most often observed is a very high varaince (sensitivity to data), especially near the boundaries of the data:



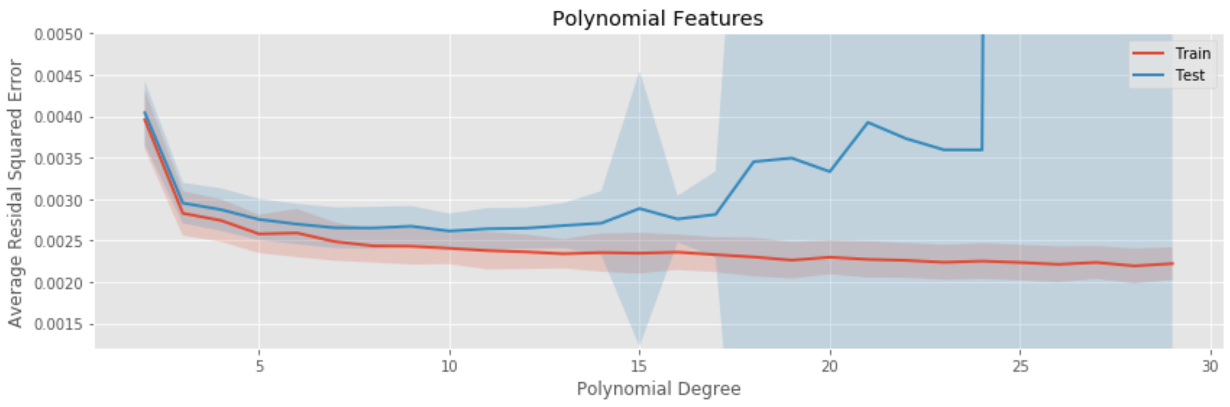
Above we have a fixed data set, and we have fit and plotted polynomial regressions of various degrees. The most striking feature is how badly the higher degree polynomials fit the data near the edges. The variance explodes! This is especially problamatic in high dimensional situations where almost *all* of the data is near the boundaries, due to the curse of dimensionality.

Another way to look at this is to plot residuals for each data point x over many samples from the same population, as we vary the degree of the polynomial we fit to the data:



Here we see the same pattern from earlier, the instability in our fits starts at the edges of the data, and moves inward as we increase the degree.

Another final way to observe this effect is to estimate the average testing error of polynomial regressions fit repretedly to the same population as the degree is changed:



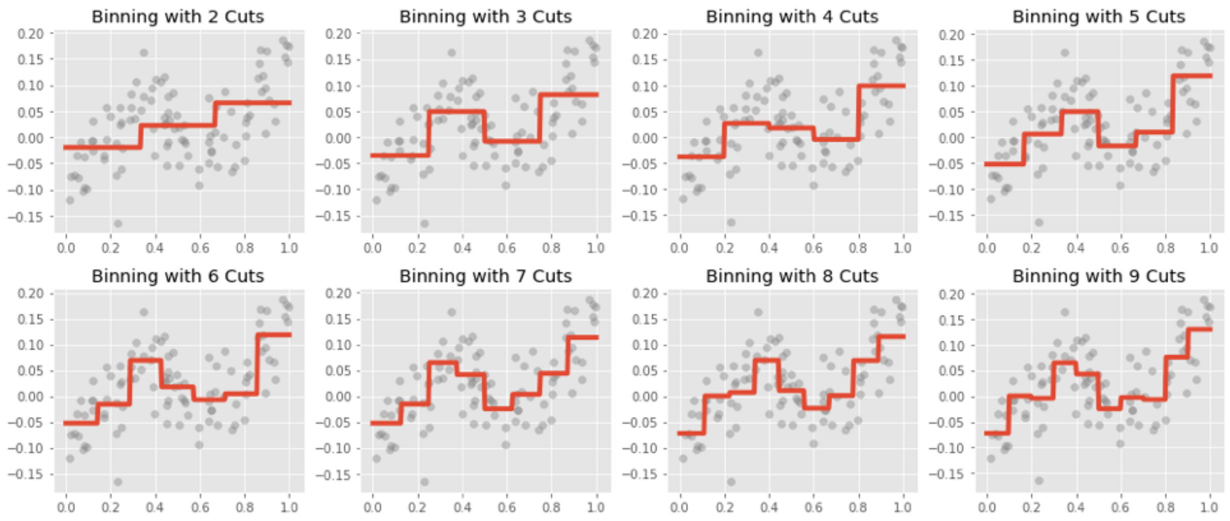
The polynomial regression eventually drastically overfits, even to this simple one dimensional data set.

There are other issues with polynomial regression; for example, it is inherently non-local, changing the value of y at one point in the training set can affect the fit of the polynomial at data points very far away, resulting in tight coupling across the space of our data (often referred to as *rigidity*). You can get a feel for this by playing around with the interactive scatterplot somoothers (http://madrury.github.io/smoothers/) app hoseted on this site.

The methods we will lay out in the rest of this post will go some way to alleviate these issues with polynomial regression, and serve as superior solutions to the same underlying problems.

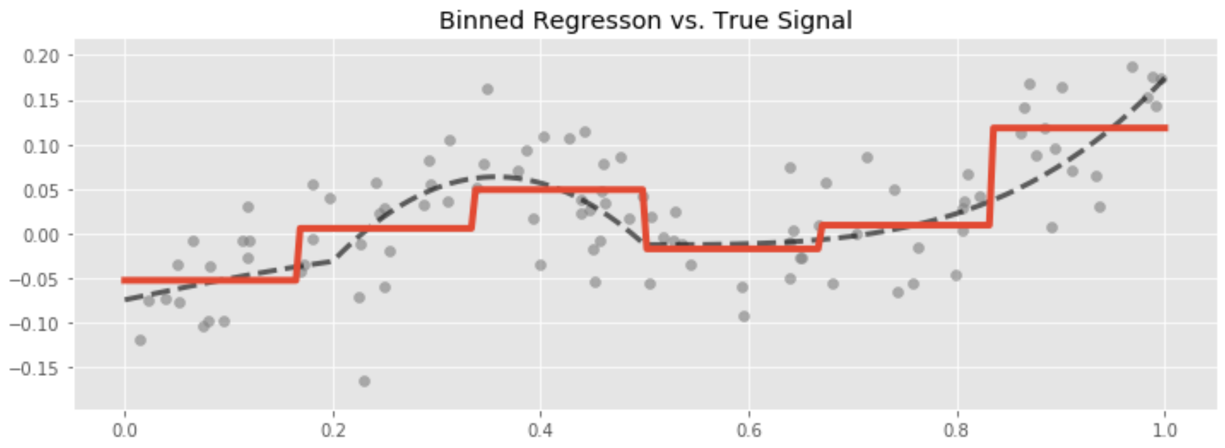
Binning Expansion

Probably the first thing that occurs to most modelers when reflecting on other ways to capture non-linear effects in regression is to **bin** the predictor varaible:



In binned regression we simply cut the range of the predictor varaible into equally sized intervals (though we could use a more sophisticated rule, like cutting into intervals at percentiles of the marginal distribution of the predictor). Membership in any interval is used to create a set of indicator variables, which are then regressed upon. In the one predictor case, this results in our regression predicting the mean value of y in each bin.

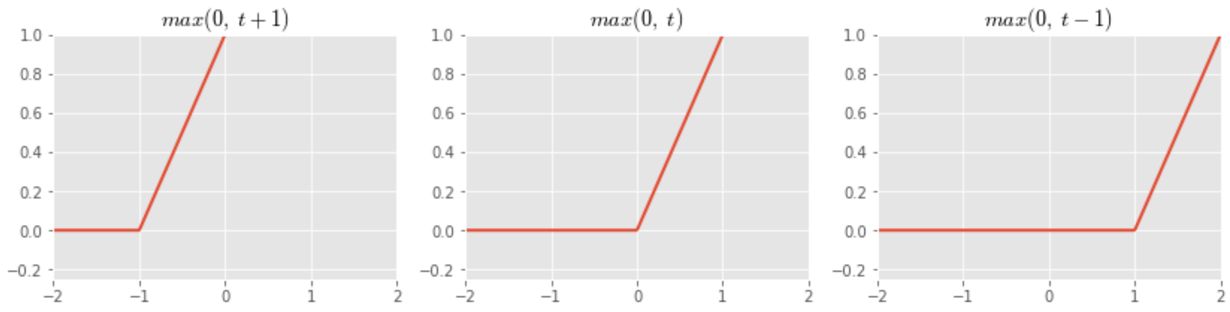
Binning has its obvious conceptual issues (https://stats.stackexchange.com/questions/68834/what-is-the-benefit-of-breaking-up-a-continuous-predictor-variable). Most prominently, we expect most phenomina we study to vary continuously with inputs. Binned regression does not create continuous functions of the predictor, so in most cases we would expect there to be some unavoidable bias within each bin. In the simple case where the true relationship is monotonic in an interval, we expect to be underpredicting the truth on the left hand side of each bin, and on the right hand side we are expect to overpredict.



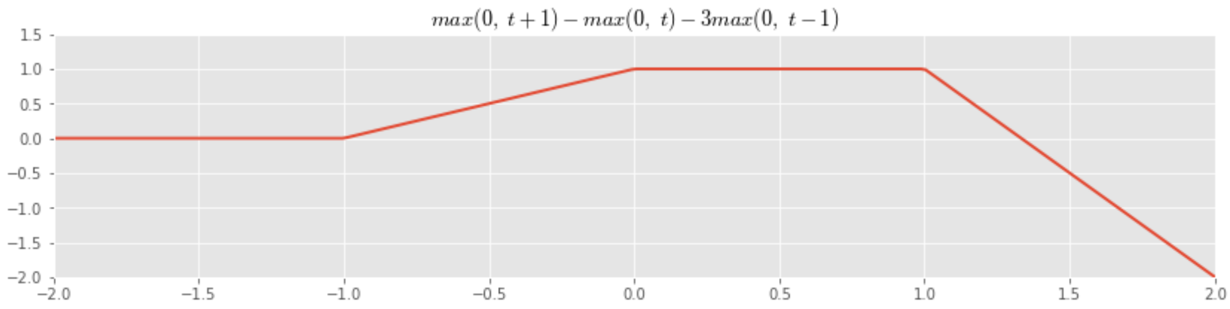
Even so, binning is popular. It is easy to discover, implement, and explain, and often does a good enough job of capturing the non-linear behaviour of the predictor response relationship. There are better options though, we will see later that other methods are both more cenceptually appealing, and do a better job of capturing the predictive power in the data with less estiamted parameters.

Piecewise Linear Splines

As a first step towards a general non-parametric *continuous* basis expansion, we would like to fit a **piecewise linear funtion** to our data. This turns out the be rather easy to do using translations of the template function $f(x) = max(0, x)$



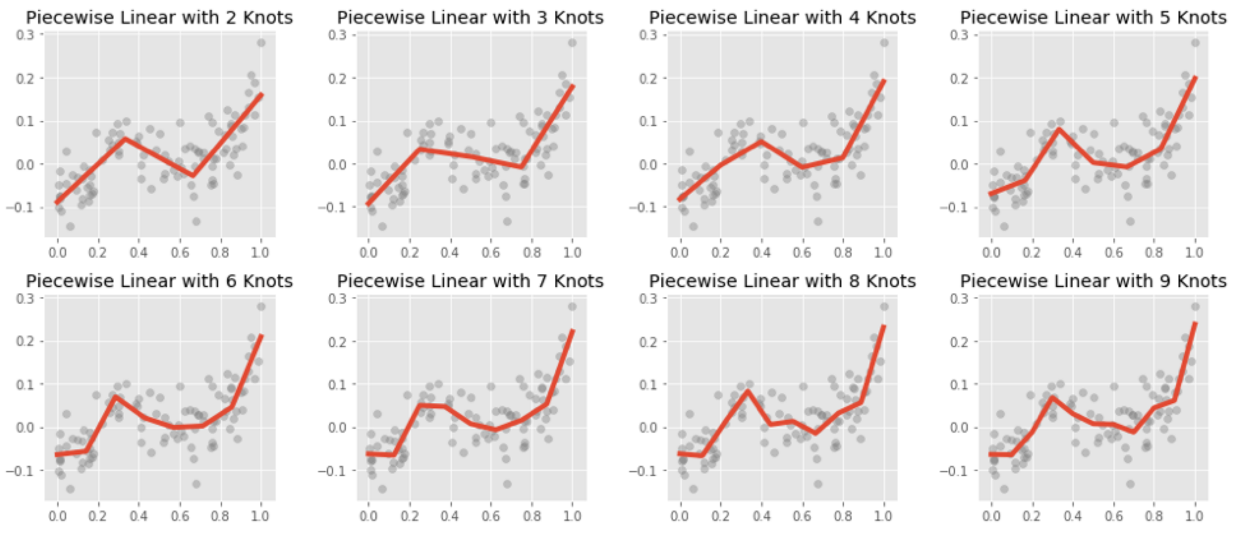
Taking linear combinations of these functions, we can create many piecewise linear shapes



If we choose a fixed set of points where we allow the slope of the linear segment to change, then we can use these as basis functions in our regression:

$$\begin{aligned} f_0(x) &= x \\ f_1(x) &= max(0, x - k_0) \\ f_2(x) &= max(0, x - k_1) \\ &\vdots \\ f_r(x) &= max(0, x - k_r) \end{aligned}$$

The result will be to fit a continuous piecewise linear function to our data:

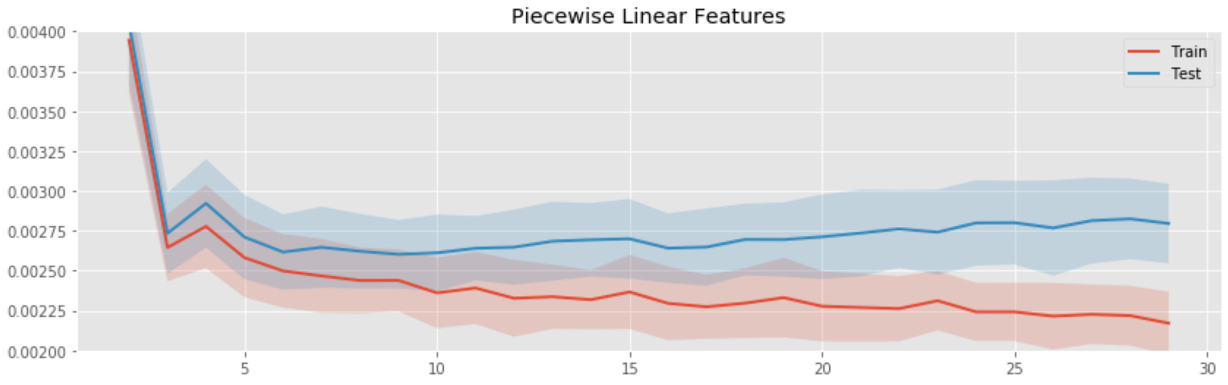


The values $\{k_1, k_2, \dots, k_r\}$, which are the x coordinates at which the slope of the segments may change, are called **knots**.

Note that we include the basis function $f_0(x) = x$ which allows the spline to assume some non-zero slope *before* the first knot. If we had not included this basis function, we would have been forced to use a zero slope in the interval (∞, k_1).

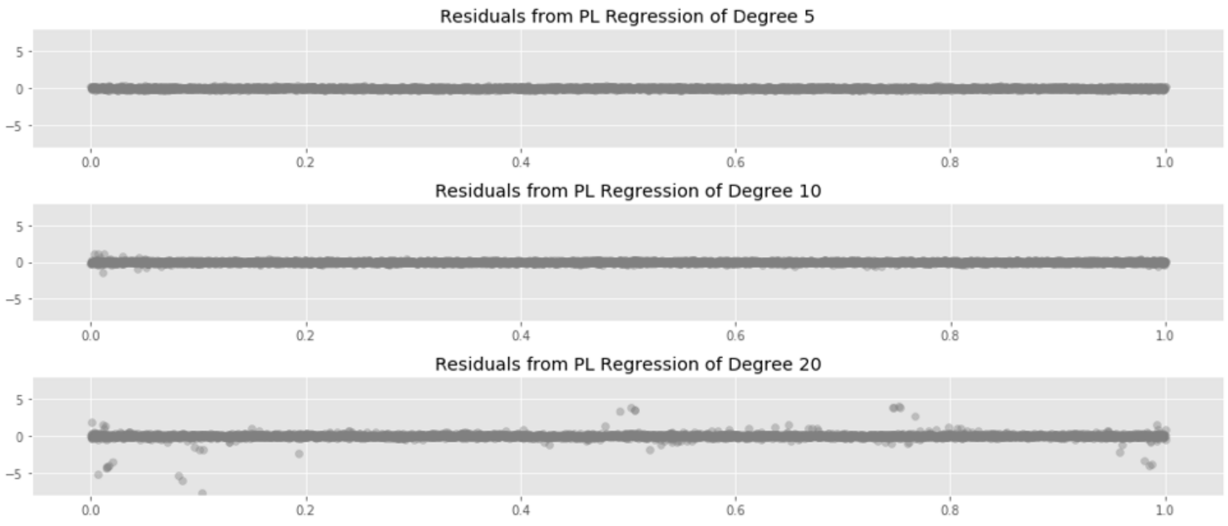
The estimated parameters for the basis function elements have a simple interpretation, they represent the *change in slope* when crossing from the left hand to the right hand side fo a knot.

Fitting piecewise linear curves instead of polynomials prevents the explosion of varaince when estimating a large number of parameters:



We see that as the number of knots increases, the linear spline *does* begin to overfit, but much more slowly than the polynomial regression with the sampe number of parameters.

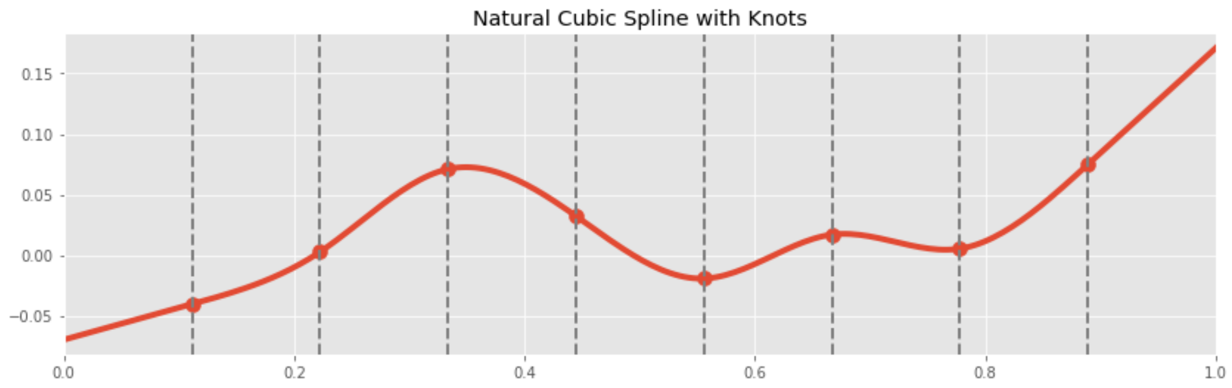
The varaince does not tend to accumulate in any one area (as it did in the polynomial case, always accumulating near the boundaries of the data). Instead the pockets of high varaiance are dependent on the specific data we happen to be fitting to



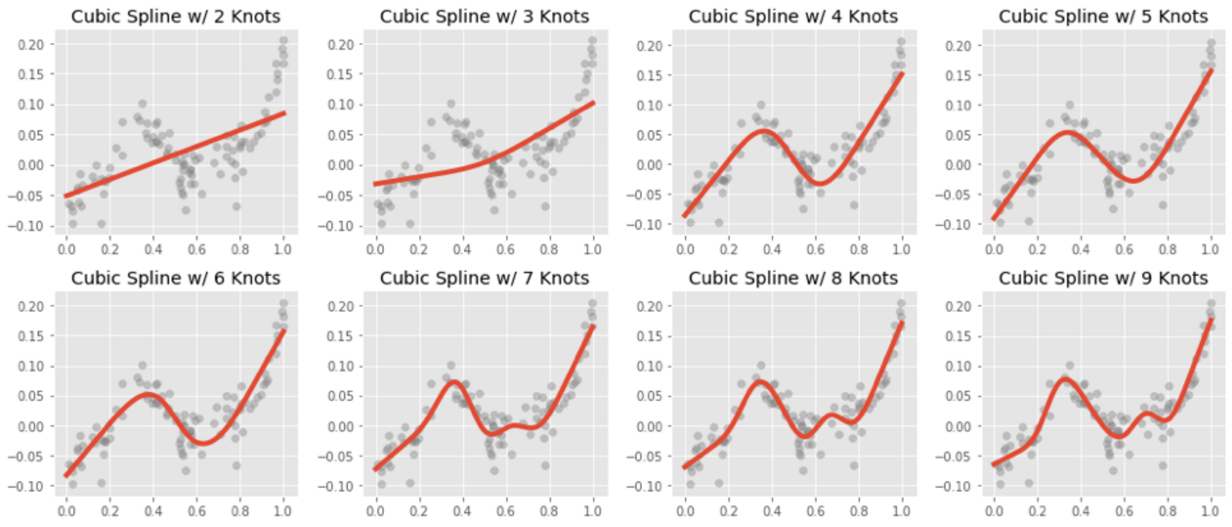
Natural Cubic Splines

Our final example of a basis expansion will be **natural cubic splines**. Cubic splines are motivated by the philosophy that most phenomina we would like to study vary as a *smooth* function of thier inputs. While the linear splines changed abruptly at the knots, we would like some way to fit curves to our data where the dependence is less violent.

A **natural cubic spline** is a curve that is continuous, has countinuous first derivatives (the tangent line makes no abrupt changes), continuous second derivatives (the tangent lines rate of rotation makes no abrupt changes); and it equal to a cubic polynomial except a points where it is allowed to high higher order discontinuities, these points are the **knots**. Additionally, we require that the curve be *linear* beyond the knots, i.e. to the left of the first knot, and right of the final knot.

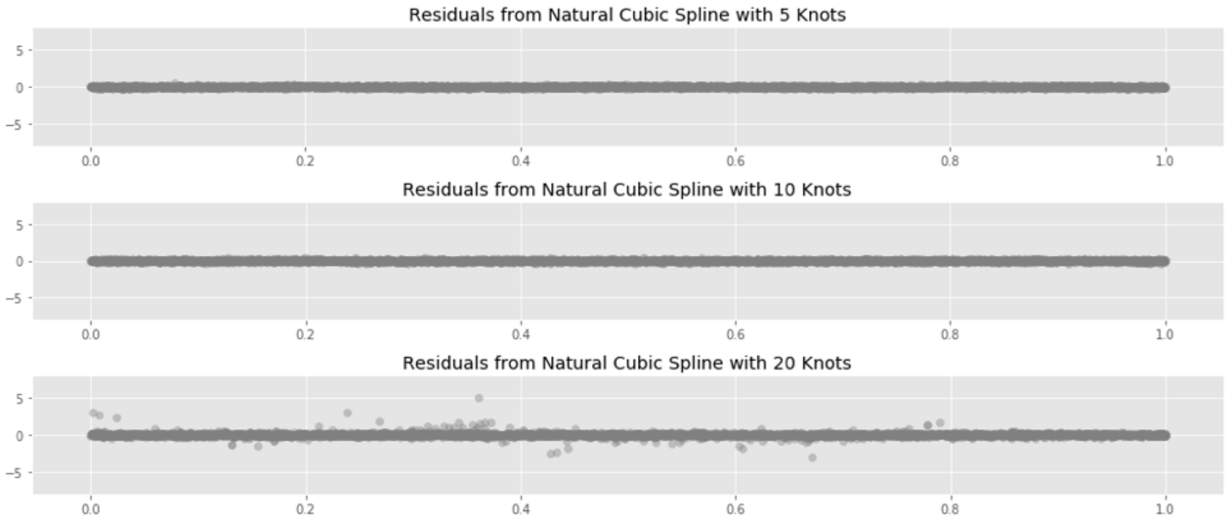


We will skip writing down the exact form of the basis functions for natural cubic spline, the interested reader can consult [The Elements of Statistical Learning (http://web.stanford.edu/~hastie/ElemStatLearn/)](http://web.stanford.edu/~hastie/ElemStatLearn/) or the [source code (https://github.com/madrury/basis-expansions/blob/master/basis_expansions.py#L205)](https://github.com/madrury/basis-expansions/blob/master/basis_expansions.py#L205) used in this post.



All thought they seem more complex, it is important to realize that the continuity constraints on the shape of the spline are strong. Whlie a estimating a piecewise linear spline with r knots uses $r + 1$ parameters (ignoring the intercept), estiamting a cubic spline takes only r parameters. Indeed, in the above picture, the spline with only one knot is a *line*; this is because the slopes of the left hand and right hand line must match.

The linear constraints near the edge of the data are intended to prevent the spline for overfitting near the boundaries of the data, as polynomial regression tends to do:



Experiments

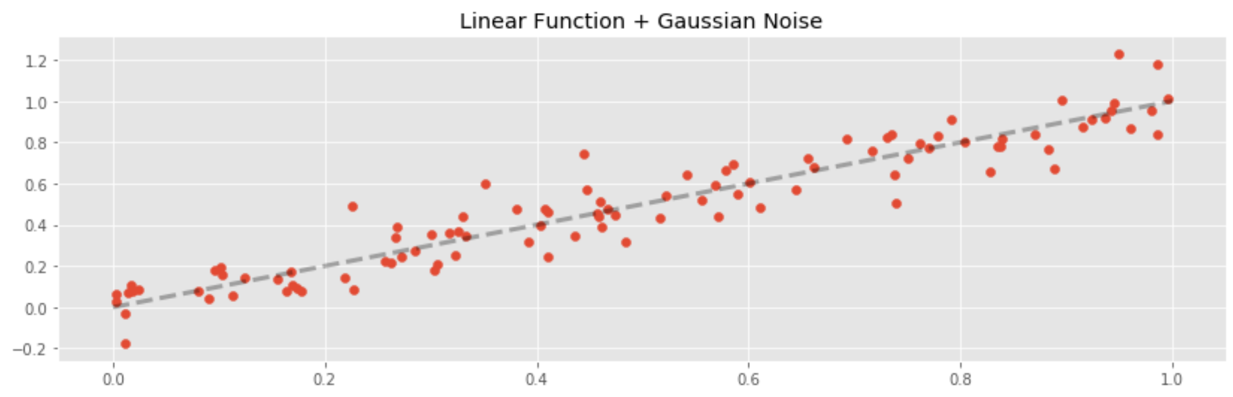
We describe below four experiments comparing the bahaviour of the basis expansions above. In each experiment we fit a regression to a dataset using one of the above expansions. The degrees of freedom (number of estimated parameters) was varied, and the preformance of the fit was measured on a held out dataset[^1]. This procedure what then repeated many times[^2] for each expansion and degree of freedom, and the average and varaince of the hold out error was computed.

Our four experiments each fit to datasets sampled from a different true signal:

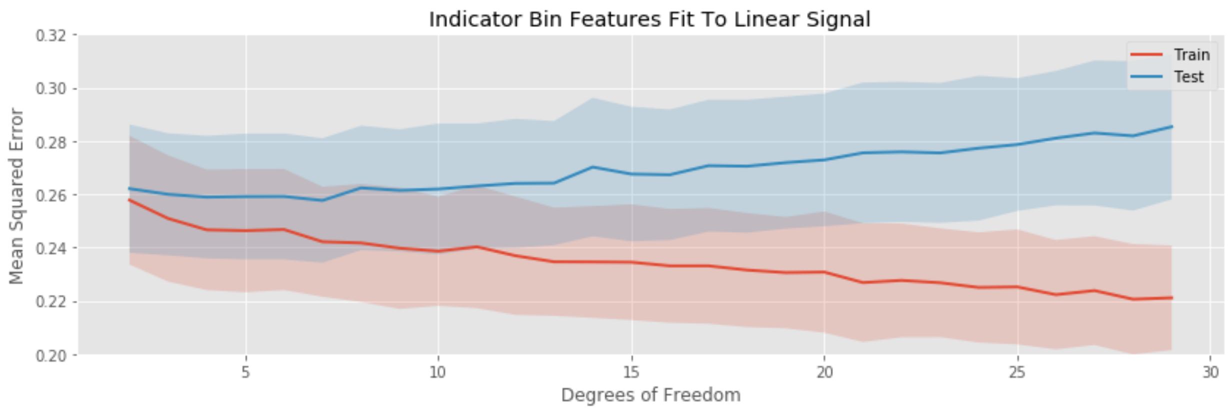
- A line plus random gaussian noise.
- A sin curve plus random gaussian noise.
- A weird continuous curve of mostly random design, plus random gaussian noise.
- A sin curve with random discontinuities added, plus random gaussian noise.

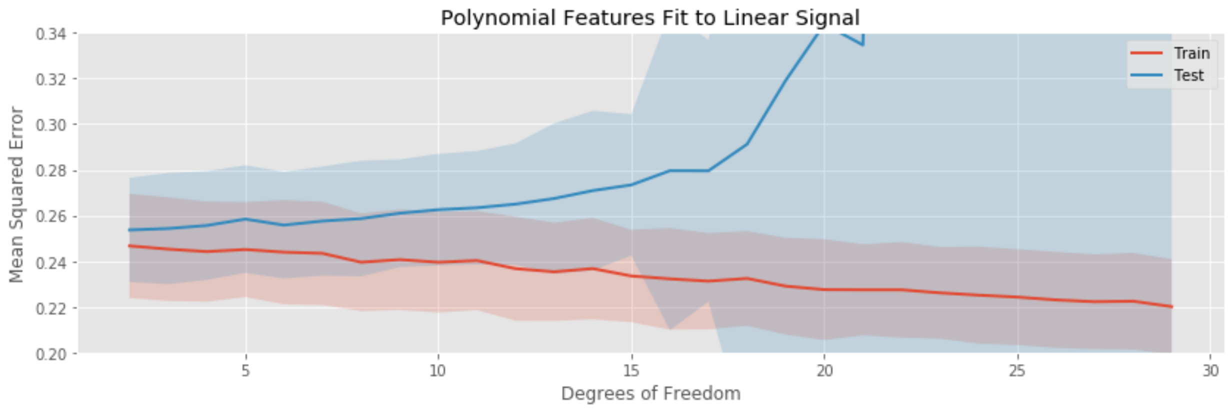
Fitting to a Noisy Line

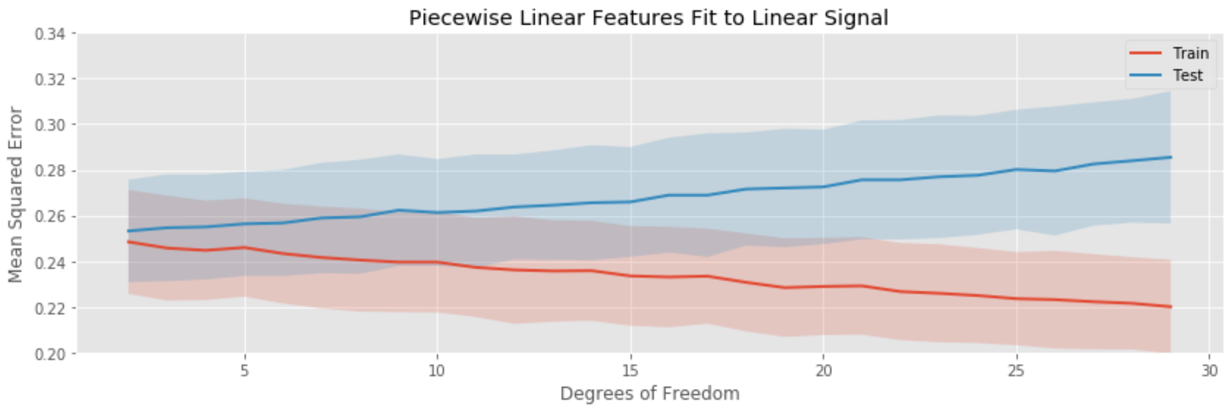
As a first esperiment, let's fit to linear truth. This is the simplest possible case for our regression methods.

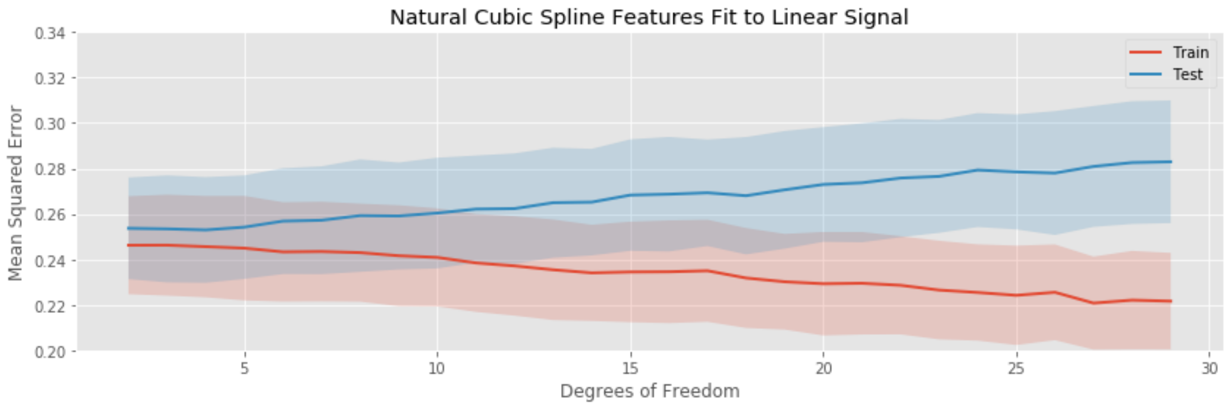


Let's look at how the training and testing error varies as we increase the number of fit parameters for each of our basis expansions





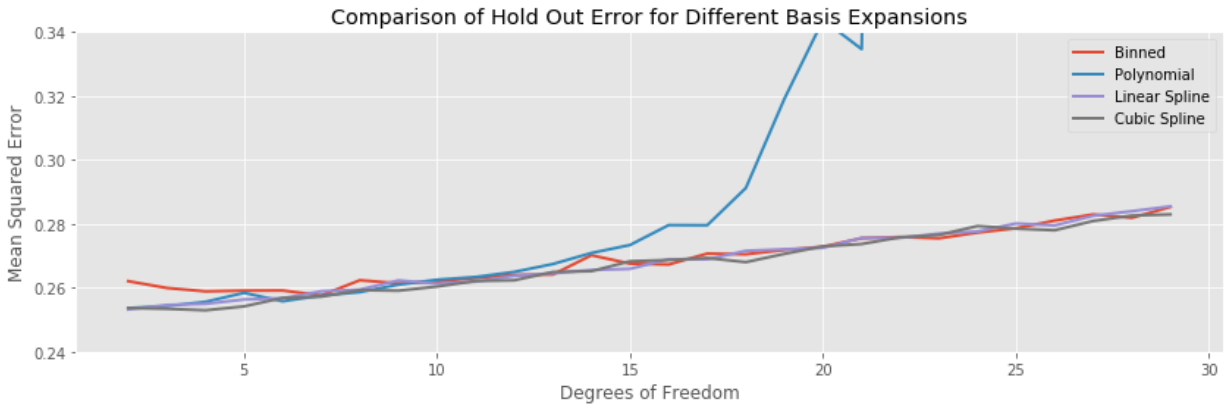




The polynomial expansion eventually overfits **drastically**, while the other tree transformations only overfit gradually. None of the models have any bias, so the minimum expected testing error occurs for the minimum number of fit parameters.

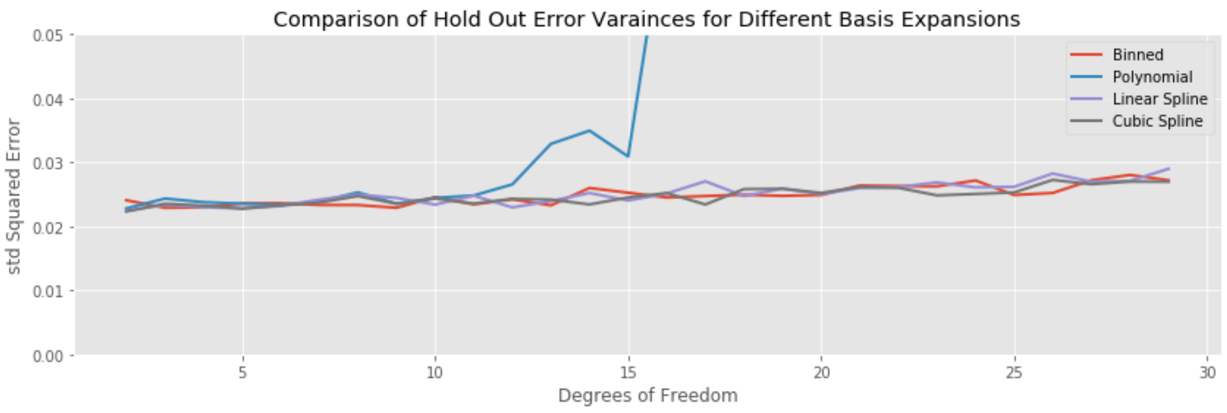
The binning, piecewise linear, and piecewise cubic transformations do about equally well in terms of average testing error throughout, and seem to have about the same varaition in testing error (i.e. the bands around the testing error curves area ll approximately the same width).

We can get a clearer comparison by plotting only the testing errors all on the same axis:



Between two and fifteen fit parameters, all the methods but polynomial regression perform about eqaully well, and overfit the training data at the same rate. So in this simple case, if we are only going for test performance, it doesnt so much matter what we choose.

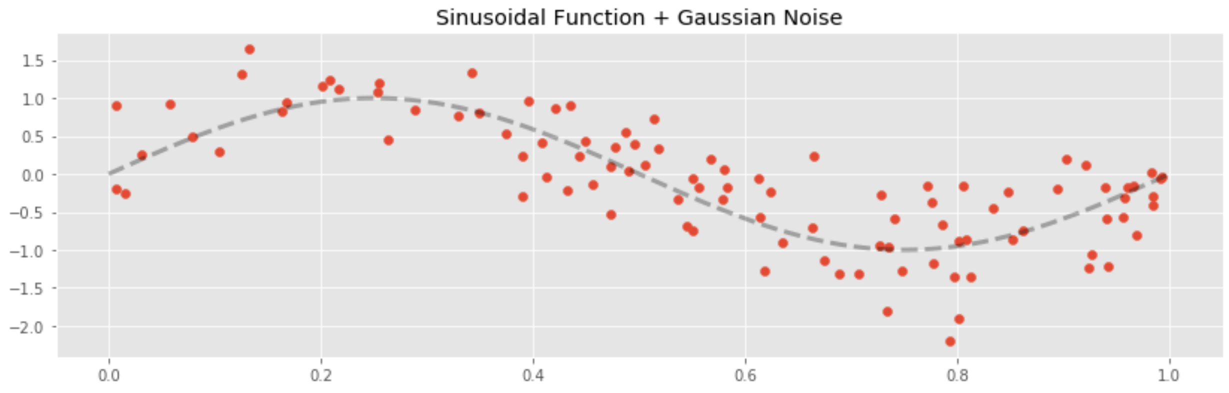
To examine the varaince of the methods, we can plot the width of the varaince bands all on the same axis:



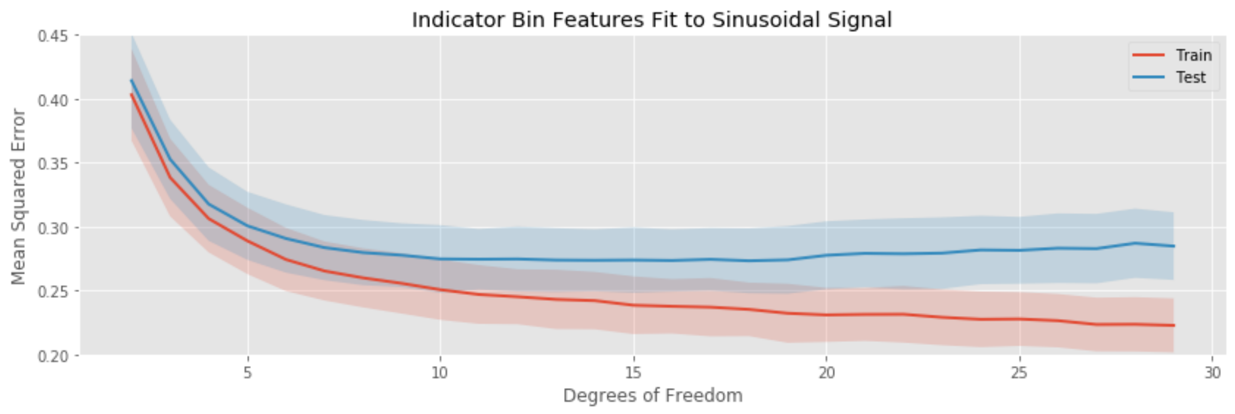
In this example the variance of the testing error stays the same for all basis exapansions throughout the entire range of complexity, with the obvious exception of polynomial regression, where the varaince explodes along with the expectation.

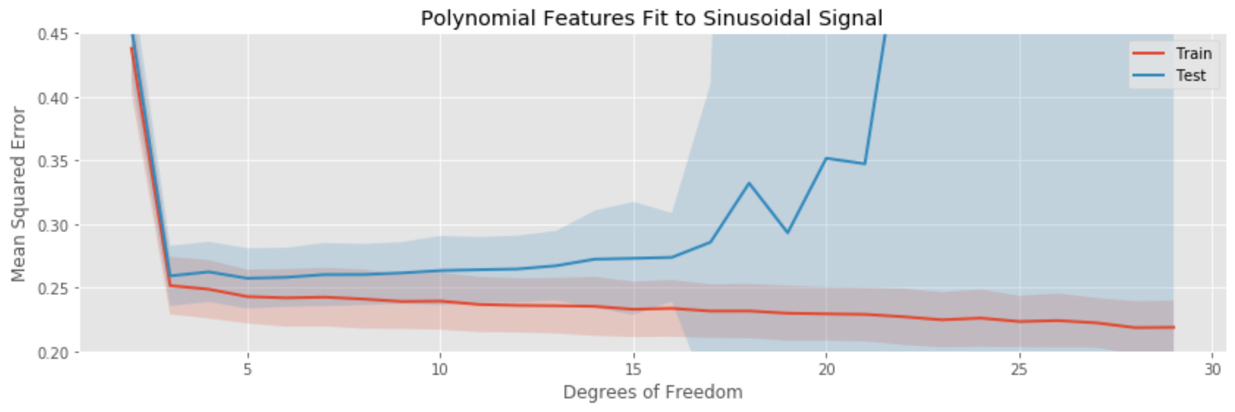
Fitting to a Noisy Sinusoid

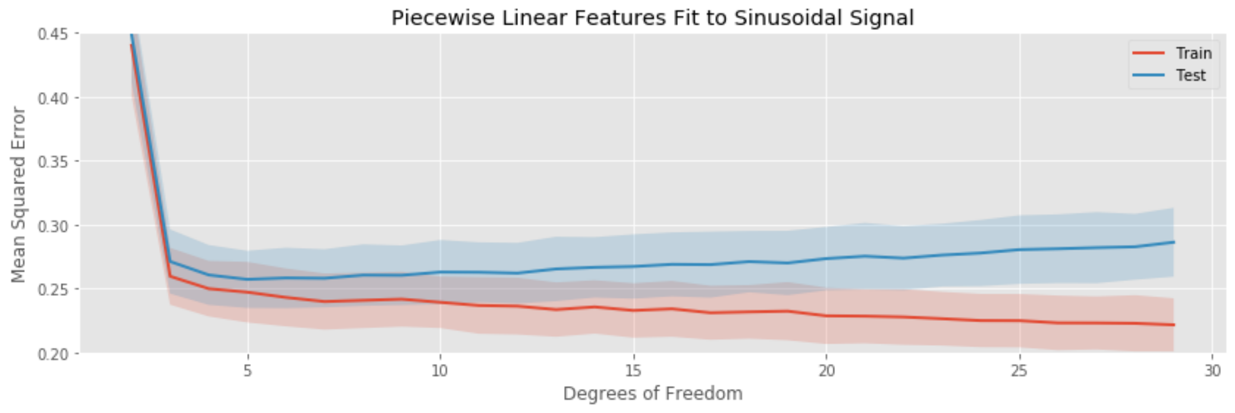
Our next experiment will fit the various basis expansions to a sinusoidal signal:

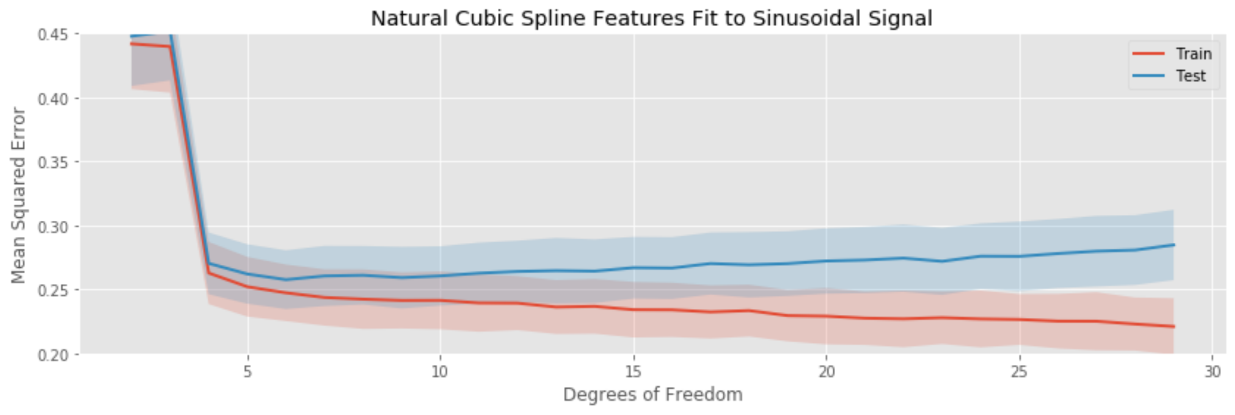


The training / testing error plots are created in the same way as for the linear signal:


Indicator Bin Features Fit to Sinusoidal Signal
Train
Test
Mean Squared Error
Degrees of Freedom
0.45
0.40
0.35
0.30
0.25
0.20
5
10
15
20
25
30


Polynomial Features Fit to Sinusoidal Signal
Train
Test
Mean Squared Error
Degrees of Freedom
0.45
0.40
0.35
0.30
0.25
0.20
5
10
15
20
25
30

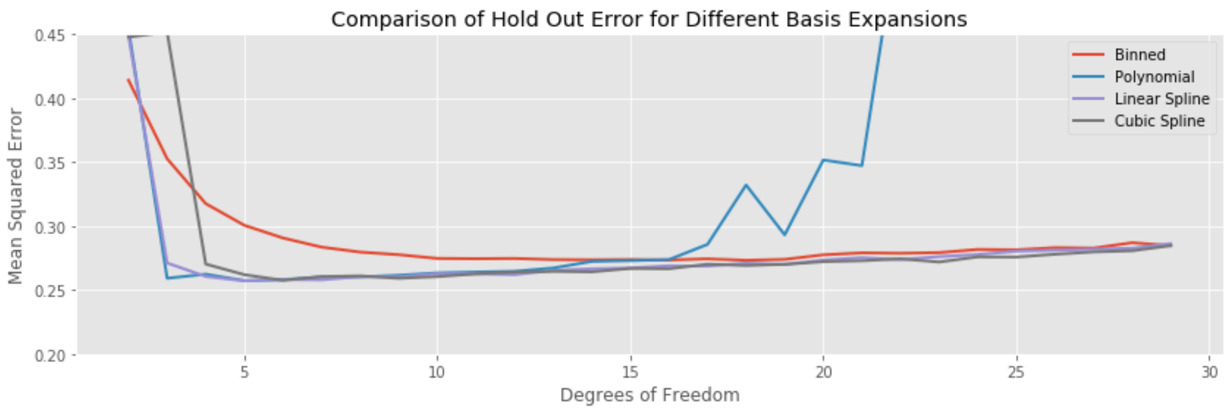

Piecewise Linear Features Fit to Sinusoidal Signal
Train
Test
Mean Squared Error
Degrees of Freedom
0.45
0.40
0.35
0.30
0.25
0.20
5
10
15
20
25
30


Natural Cubic Spline Features Fit to Sinusoidal Signal
Train
Test
Mean Squared Error
Degrees of Freedom
0.45
0.40
0.35
0.30
0.25
0.20
5
10
15
20
25
30

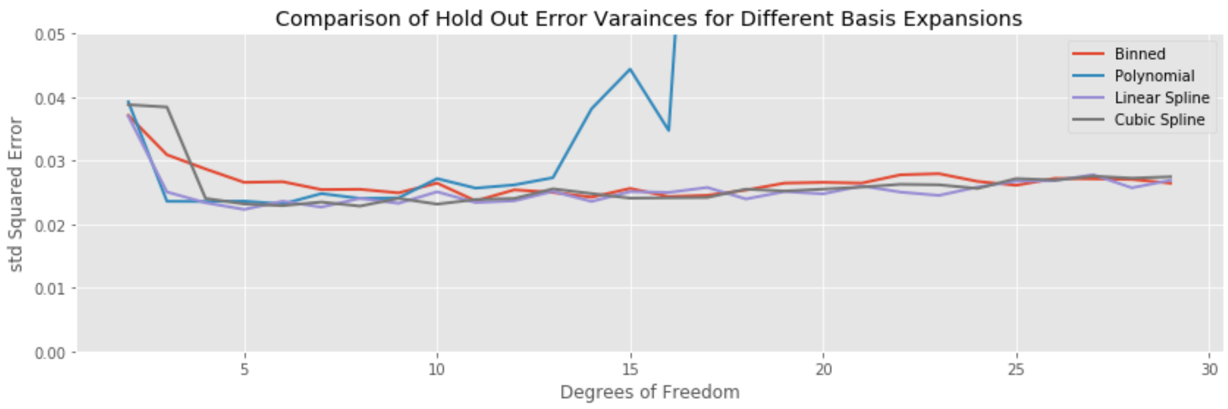
The most noticable difference in this sitation is that at a low number of estimated parameters, all the models are badly biased: while the fits to the linear function started at their lowest testing and training errors, for the sinuousid each basis expansion regression starts with high error, and decreases quickly as more parameters enter the model and the bias decreases.

The rates of overfitting tell the same story as before. The polynomial regression eventaully overfits drastically, while the others overfit slowly.

Plotting the hold out error curves all on the same plot highlights another interesting feature to the story:



The binned regression fits to the data much more slowly than polynomial and the two splines. While the splines achive thier lowest average testing error at around five estimated parameters, it takes around eighteen bins to minimize the average testing error. Furthermore, the minimum testing error for the polynomial and spline regressions is *lower* than for the bins (and again, this minimum is achieved with *less* estimated parameters).



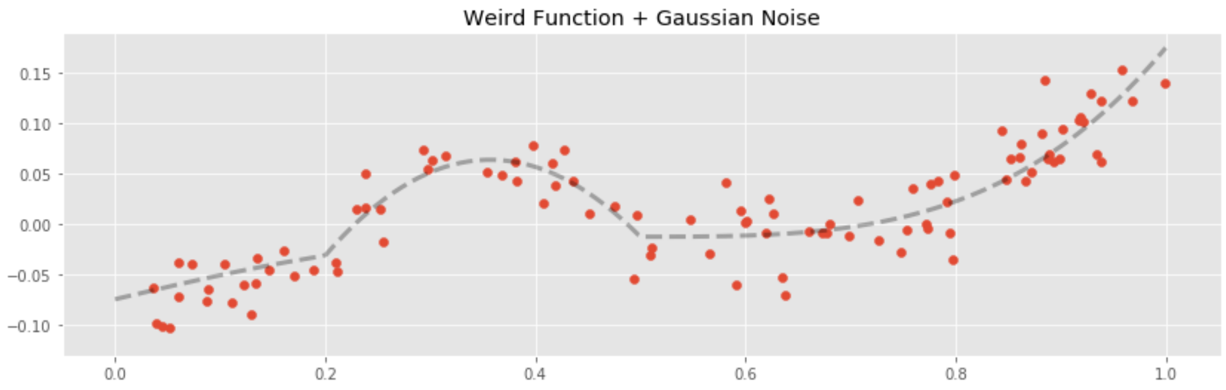
A similar story holds for the testing error varainces: the binning transformation has slightly higher varaince than the two spline methods.

Fitting to a Noisy Weird Curve

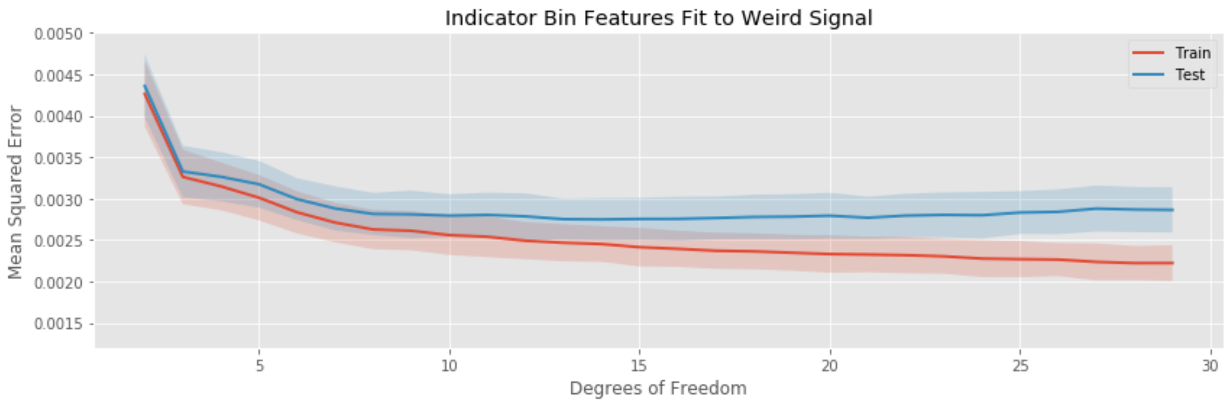
Our next example is a weird function constructed by hand to have an unusual shape

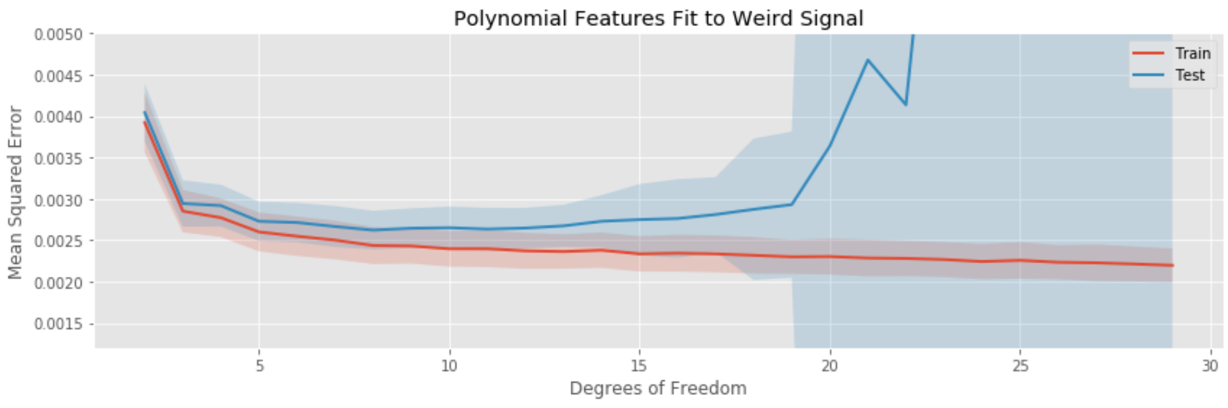
```
def weird_signal(x):
    return (x*x*x*(x-1)
            + 2*(1/(1 + np.exp(-0.5*(x - 0.5))))
            - 3.5*(x > 0.2)*(x < 0.5)*(x - 0.2)*(x - 0.5)
            - 0.95)
```

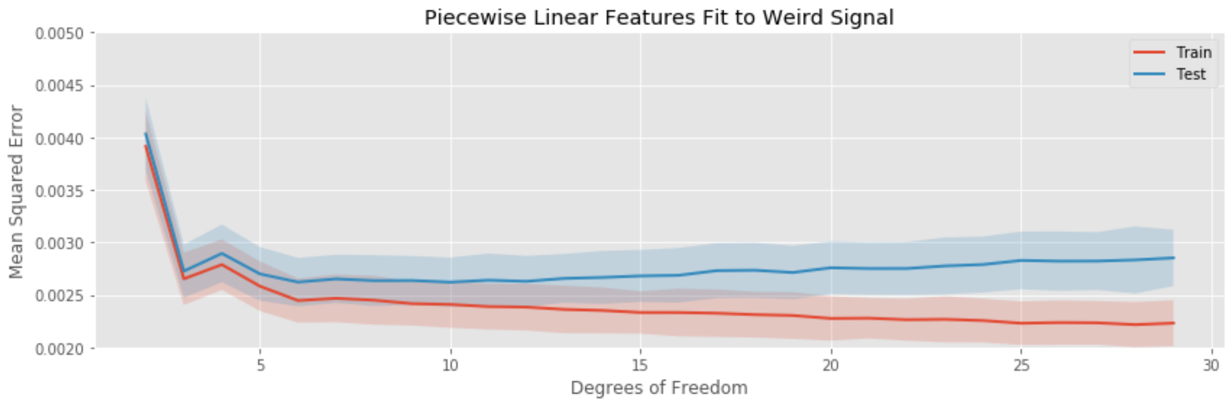
The graph of this function has exponential, quadratic, and linear qualities in different segments of its domain.

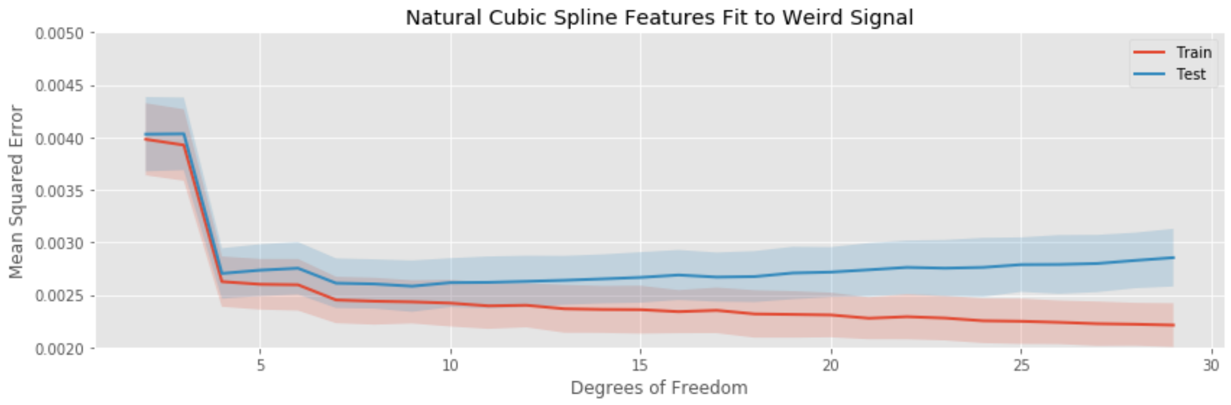


We follow the usual experiment at this point, here are the training and testing curves for the various basis expansion methods we are investigating.





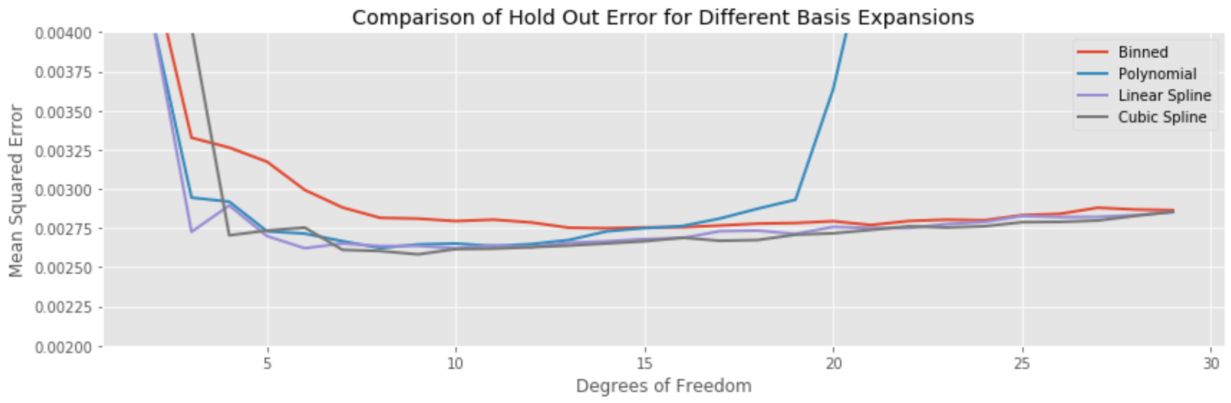




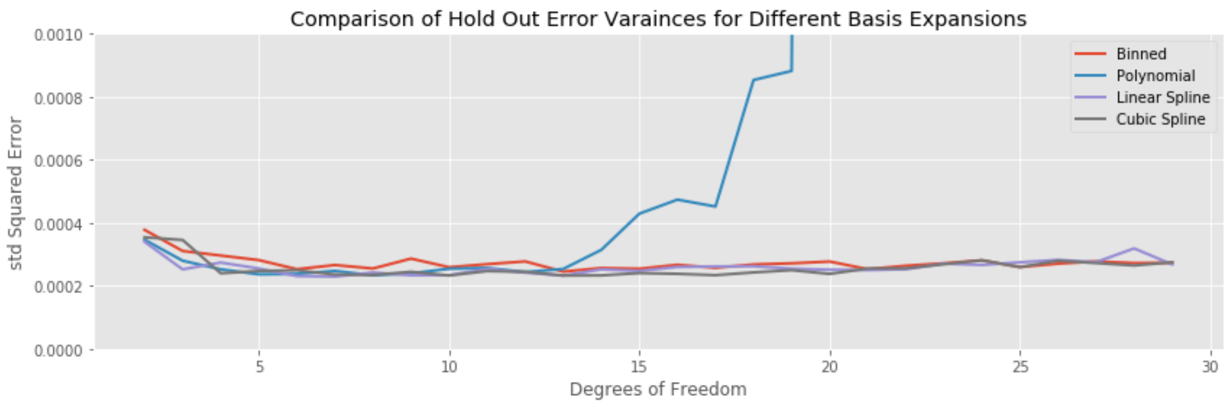
The story here captures the same general themes as in the previous example:

- The quadratic basis expansion eventually drastically overfits.
- The other basis expansions overfit slowly.
- For very a very small number of estiamted parameters, each of the basis expansions is biased.
- The varaince in test error is generally stable as more parameters are added to the model, *except* for the polynomial regression, which explodes.

Comparing all the test errors on the same plot again highlights similar points to the sin curve



The bias of the binned regression decreases more slowly than the spline methods, and its minimum hold out error is both larger than the splines, and achieved at more estiamted parameters. The linear and cubic spline generally achieve a similar testing error for the same number of estimated parameters.



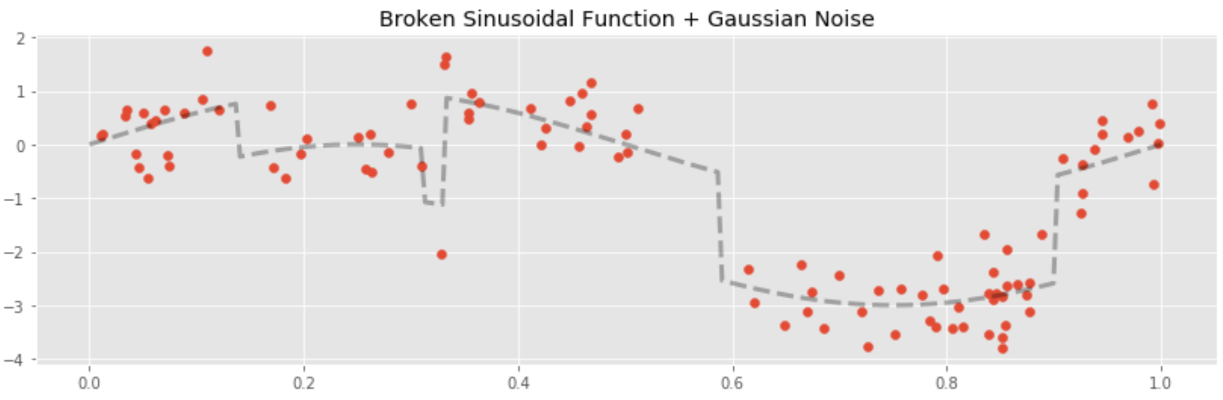
The varainces for the non-polynomial models are stable, with a hint that the binned regression has a larger overall varaince across a range of model complexities.

Fitting to a Noisy Broken Sinusoid

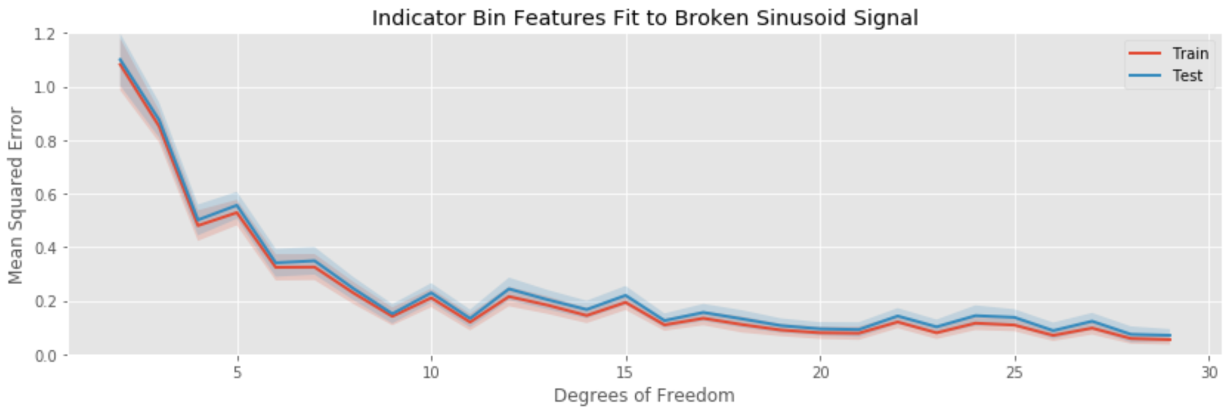
For our final example, we try a discontinuous function.

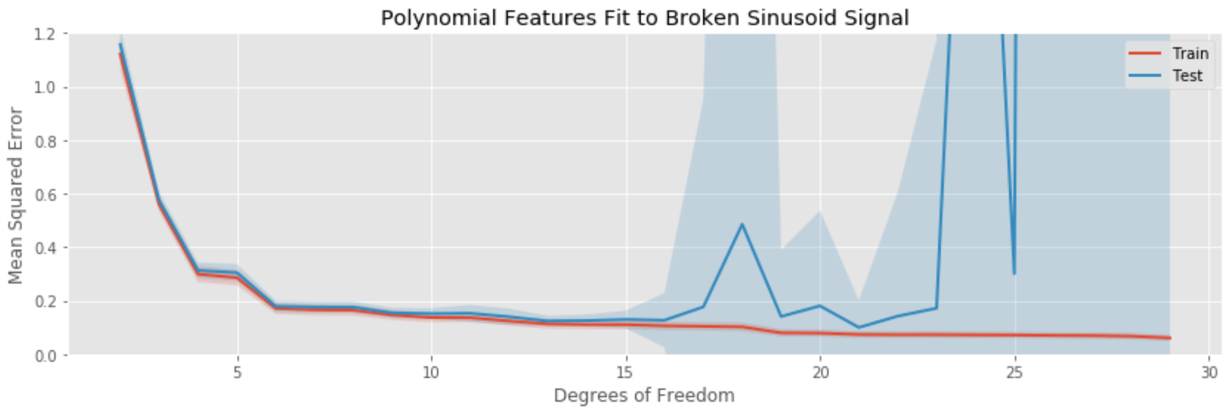
```
cutpoints = sorted(np.random.uniform(size=6))
def broken_sin_signal(x):
    return  (np.sin(2*np.pi*x)
               - (cutpoints[0] <= x)*(x <= cutpoints[2])
               - (cutpoints[1] <= x)*(x <= cutpoints[2])
               - 2*(cutpoints[3] <= x)*(x <= cutpoints[4]))
```

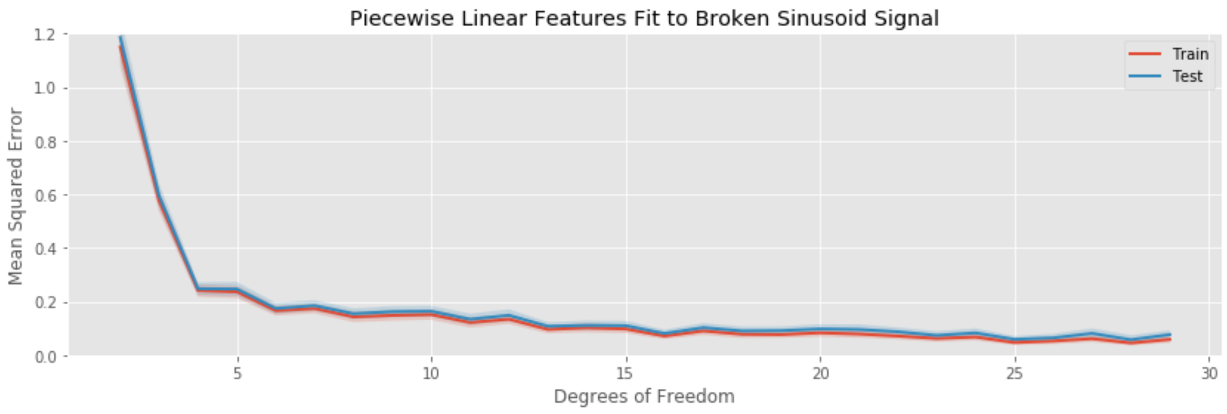
To create this function, we have taken the sin curve from before, and created discontinuities by shifting intervals of the graph up or down. The result is a broken sin curve

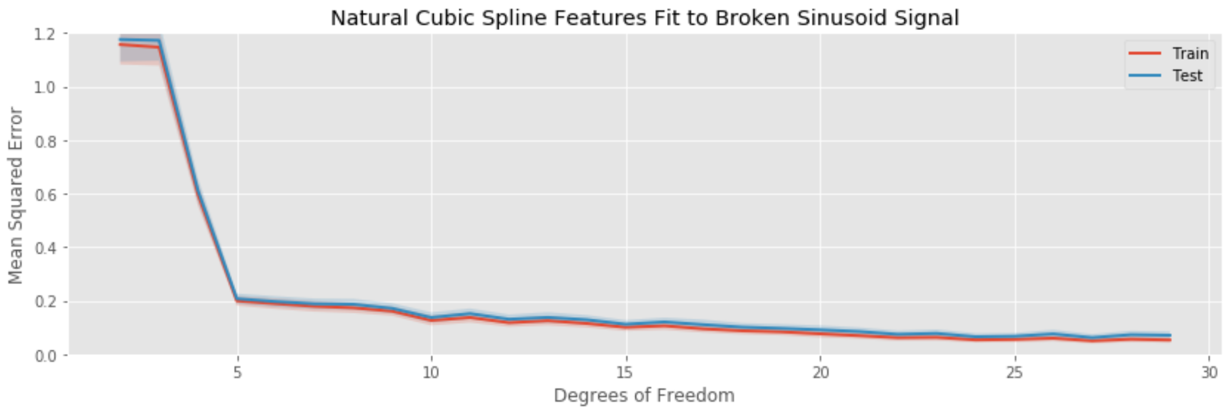


Let's repreat our experiments and see if the discontinuities change the patterns we have been observing among the different basis expansions



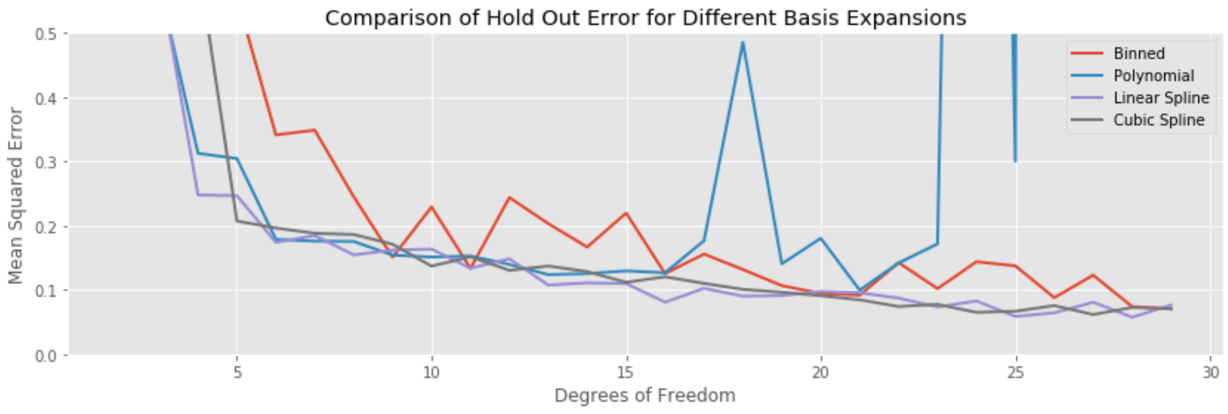




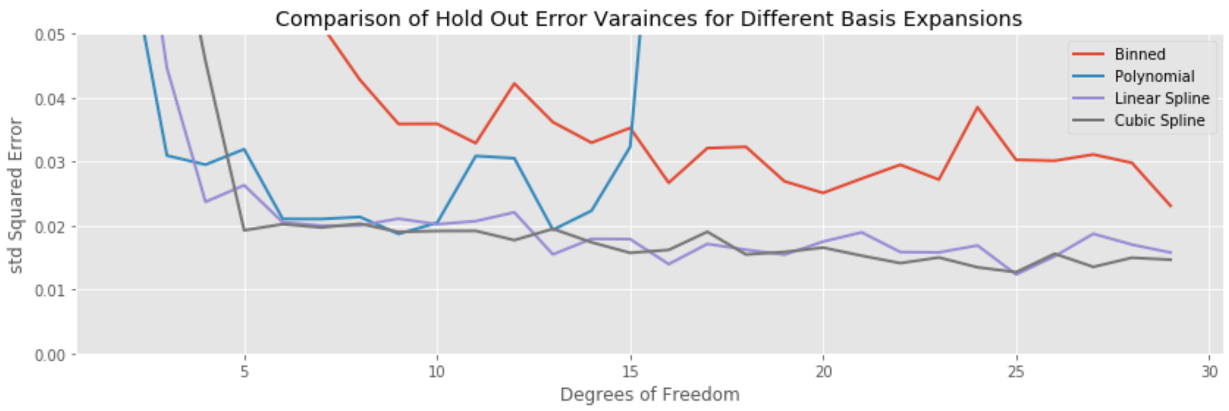


The major difference in this example is the difficulty in decreasing the bias of the fit. Since the models we are fitting are all continuous, it is *impossible* for any of them to be unbiased. The best we can do is trap the discontinuities between two ever closer knots (in the case of splines), or inside a small bin. Indeed, even at 29 estiamted parameters, the test error is still decreasing (slowly), indicating the our models can still do a better job decreasing the bias of thier fits before becoming overhelmed with varaince.

Otherwise, there are no striking differneces to the patterns we observe when comparing the various basis expansions.



Comapring the testing errors of the various methods, this time the binned expansions does noticably worse than the other methods throughout the entire range of complexities. Otherwise, the two spline models are mostly equivelent in performance.



Finally, the variance of the binned model is also significantly worse than the other methods, while the two spline methodologies are equivelent.

Conclusions

In this post we have compared four different approches to capturing general non-linear behaviour in regression.

Binning

Binning, which conceptualy simple, was seen to suffer from a few issues in comparison to the other methods

- With a small number of estiamted parameters, the binned regression suffered from a higher bias than its competitors. It was seen across multiple experiments to achive its minimal hold out error at a larger number of estiamted parameters than the other methods, and furthermore, often the minimal error achieved by the binning method was *larger* than the minimum achieved by the other methods.
- The binned regression's hold out error often had a higher varaince than the other method's. This means that, even if it performed just as well as another method *on average*, any *individual* binned regression is less trustrowthy than if using another method.

Additionally, the binned regression method has the disadvantage of producing *discontinuous* functions, while we expect most processes we encounter in nature or business to vary continuously in thier inputs. This is philosophically unnapealling, and also accounts for some of the bias seen when comparing the binning regressions to the other basis expansions.

Polynomial Regression

The major problem with polynomial regression is its instability once a moderate number of estimated parameters is reached. Polynomial regressions as *highly* sensitive to the noise in our data, and in every example were seen to eventuall overfit violently; this feature was not seen in any of the other methods we surveyed.

The polynomial regression also does not accumulate varaince uniformly throughout the support of the data. We saw that polynomial fits become highly unstable near the boundaries of the available data. This can be a sweious issue in higher dimensional situations (though we did not explore this), where the curse of dimensionality tells us that the *majority* of our data will lie close to the boundary.

Linear and Cubic Splines

These methods performed well across the range of our experiments.

- They overfit slowly, especially compared to polynomial regression.
- They generally achive a lower hold out error than the binned regressions, and achive thier minimum hold out error at a lower number of estimated parameters.
- Their hold out error is generally the lowest out of the methods we surveyed, making each *individual* regression more trustworthy.

The two spline models are a clear winner over the binned and polynomial regressions. This is a real shame, because they are often not taught in introductory courses in predictive modeling, even though they (at least the piecewise linear spline) are just as easy to implement as polynomial or binned regression.

We hope we have convinced you to use these methods in your own modeling work. The author sees no real reason to fit polynomial or binned expansions in any predictive or inferential model.