Statistical Machine Learning Notes 4

Basis Expansions

Instructor: Justin Domke

1 Bias Terms

We have defined our linear methods as

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}.$$

Many machine learning textbooks, however, introduce linear methods with an explicit intercept[1] term w_0, as something like

$$f(\mathbf{x}) = w_0 + \mathbf{w} \cdot \mathbf{x}. \tag{1.1}$$

In learning, both the parameters $\mathbf{w}$ and w_0 need to be adjusted. We have not bothered with this because our original model can be made equivalent by "tacking" a constant term onto $\mathbf{x}$. Define the function ϕ which just takes the vector $\mathbf{x}$, and prepend a constant of 1 by

$$\boldsymbol{\phi}(\mathbf{x}) = (1, \mathbf{x}). \tag{1.2}$$

Then, if we take all our training data, and replace each element $(\hat{y}, \hat{\mathbf{x}})$ by $(\hat{y}, \boldsymbol{\phi}(\hat{\mathbf{x}}))$, then we will have done the equivalent of adding an intercept term.

This is a special example of a straightforward but powerful idea known as "basis expansion". Take the input $\mathbf{x}$, and add "expand" it to make whatever method you are using more powerful.

Unfortunately, using a basis expansion as in Eq. 1.2 isn't always *exactly* equivalent to adding an intercept term. The difference comes about when we introduce regularization. It is common, with models of the form in Eq. 1.1, to regularize $\mathbf{w}$ only, leaving w_0 free to be as large or small as it wants, with no penalty. There are good reasons to think that this will give somewhat better performance. (Namely, the variance penalty for leaving the intercept unregularized is likely to be smaller than the rewards we reap it terms of decreased bias,

[1]The terminology "bias" is more common, but we will stick to "intercept", since this has nothing to do with the "bias" we discuss in the bias-variance tradeoff.

since most linear models do have some nonzero intercept.) Nevertheless, for simplicity, we won't worry about this.

Protip: A quick and dirty way to approximate an unregularized intercept term is to add a very large constant instead of 1. If we use, say, $\phi(\mathbf{x}) = (10^6, \mathbf{x})$, a very small weight can be used with almost no regularization penalty, since the equivalent of an intercept of a can be accomplished with a weight coefficient of only $a/10^6$. This doesn't work for an l_0 penalty, however.

2 Polynomials

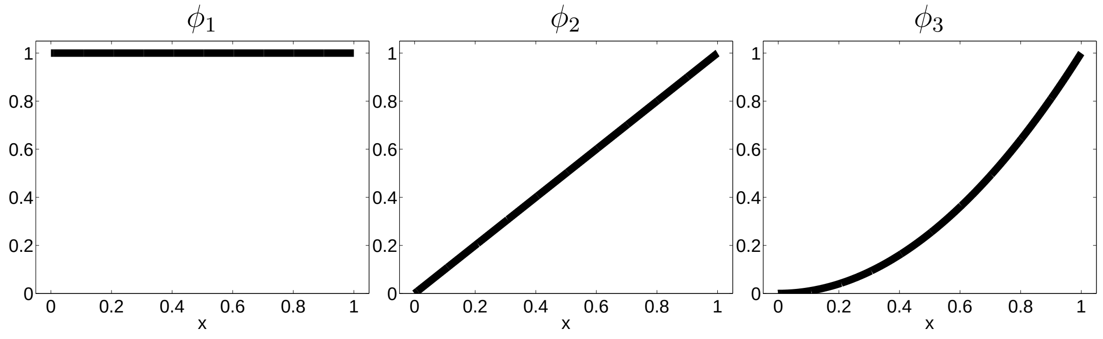
In the first notes, we looked at fitting polynomials of a scalar input like

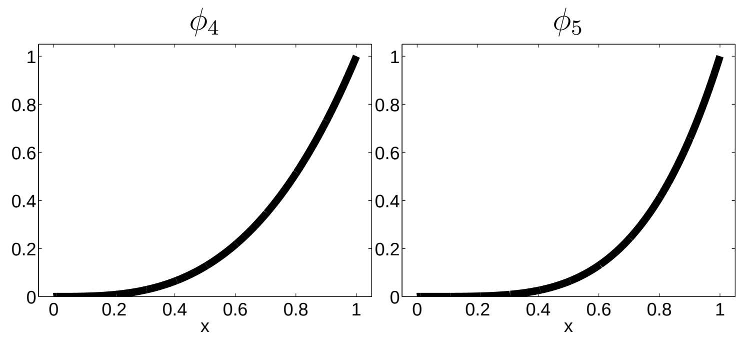
$$f(x) = w_0 + w_1 x + w_2 x^2 + ... + w_p x^p.$$

This was done not by using some specialized polynomial fitting algorithm, but just by standard least-squares regression, with the basis expansion

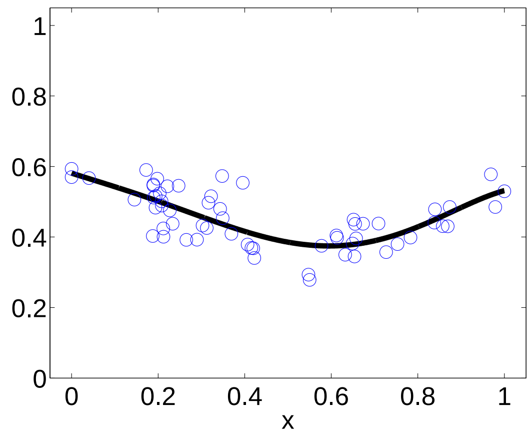
$$\boldsymbol{\phi}(x) = (1, x, x^2, ..., x^p).$$

The way to understand this is to plot $\phi_i(x)$ for a degree five expansion:





Then by simply mixing these curves with weights w_i, we can represent any degree five polynomial. For example, if we do a least-squares fit to the data from the cross-validation notes, we recover the fit



with $\mathbf{w} \approx (.58, -.39, .31, -2.53, .11, -2.55)^T$.

We notice the first glimpse of something interesting here. Let $\mathbf{x}' = \boldsymbol{\phi}(x)$. Then, what looks like a *linear* model to $\mathbf{x}'$ is a highly *nonlinear* model in the original space x.

This can also be done in higher-dimensions. If $\mathbf{x}$ is a vector input, a **quadratic expansion** of $\mathbf{x}$ is a model of the form

$$\phi_m(x) = x_i x_j.$$

One can, of course, define higher-order, e.g. **cubic expansions**, like

$$\phi_m(x) = x_i x_j x_k.$$

Now, the main problem with higher order expansions like this is that if $\mathbf{x}$ has N dimensions, $\boldsymbol{\phi}(\mathbf{x})$ will have N^p dimensions, where p is the degree of the polynomial. Thus, for high-dimensional $\mathbf{x}$, this quickly gets out of hand, both in terms of computational expense, and in terms of the amount of data needed to accurately fit the weights. It is actually common to first reduce dimensionality through a method like PCA before increasing it again through a polynomial expansion like this.

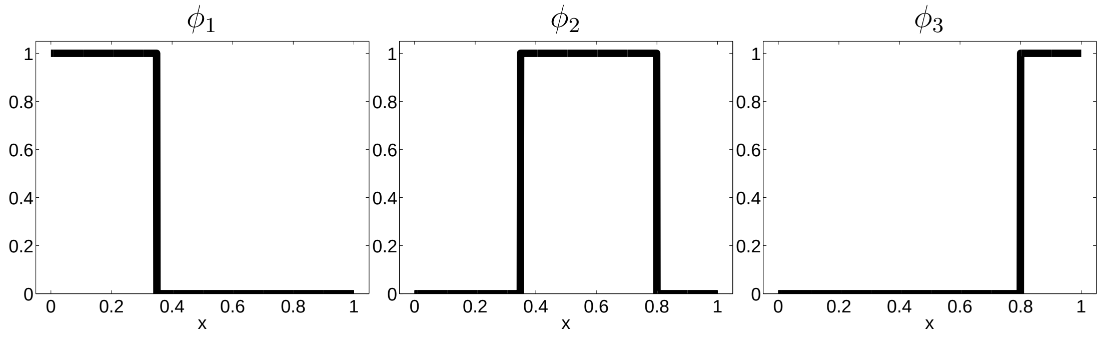
In general, if using a higher-order expansion, it is a good idea also to include lower-order expansions as well. Usually, one gets "more predictive accuracy per parameter" with a linear model than with a quadratic one.

3 Splines

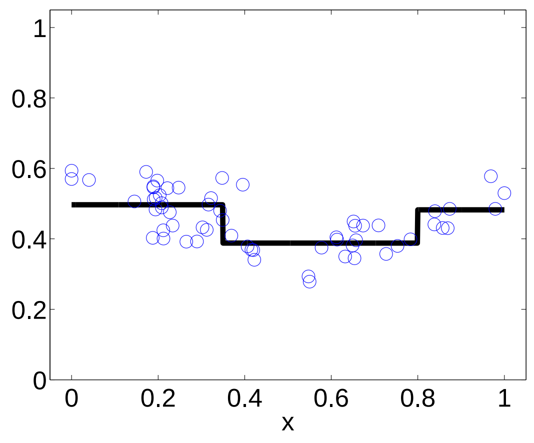
Another very natural way to proceed for one dimensional models would be a basis expansion of the form

$$\begin{aligned} \phi_1(x) &= I[x < \xi_1] \\ \phi_2(x) &= I[\xi_1 \leq x < \xi_2] \\ &\vdots \\ \phi_p(x) &= I[\xi_{p-1} \leq x] \end{aligned}$$

where $0 < \xi_1 < \xi_2 < ... < \xi_{p-1} < 1$. Here we assume that $0 \leq x \leq 1$. Fitting a linear model to this is equivalent to fitting a piecewise constant model with discontinuities at ξ_i.



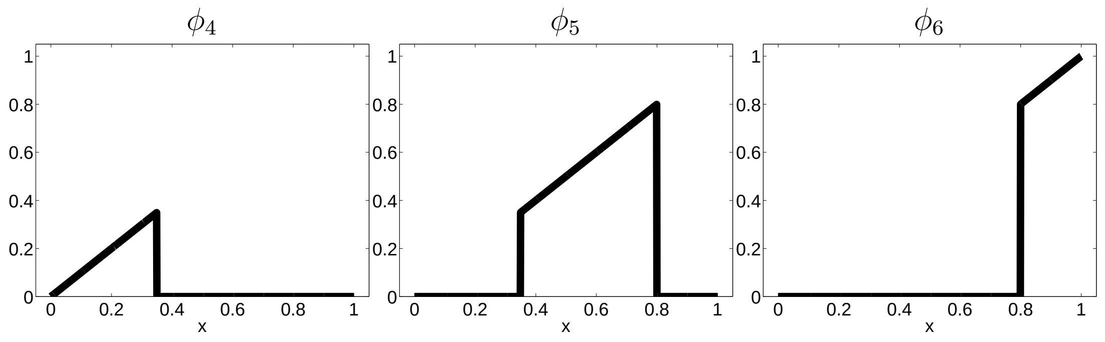
If we fit this to our scatter data, we get the fit



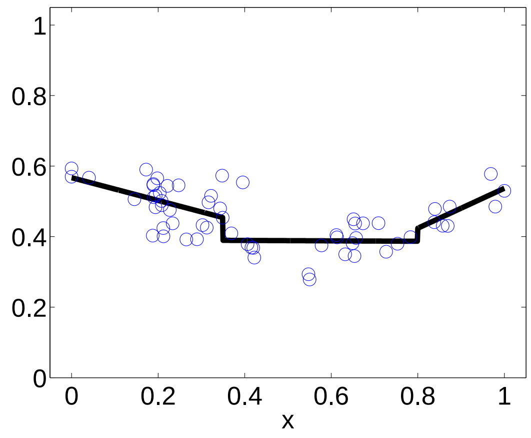
with $\mathbf{w} \approx (.49, .38, .48)^T$.

A natural way to improve this would be to fit a piecewise *linear* model. This can be done by tacking on extra basis functions like

$$\begin{aligned}\phi_{p+1}(x) &= xI[x < \xi_1]\\ \phi_{p+2}(x) &= xI[\xi_1 \le x < \xi_2]\\ &\vdots\\ \phi_{2p}(x) &= xI[\xi_{p-1} \le x].\end{aligned}$$



If we fit this to our scatter data, we get the fit

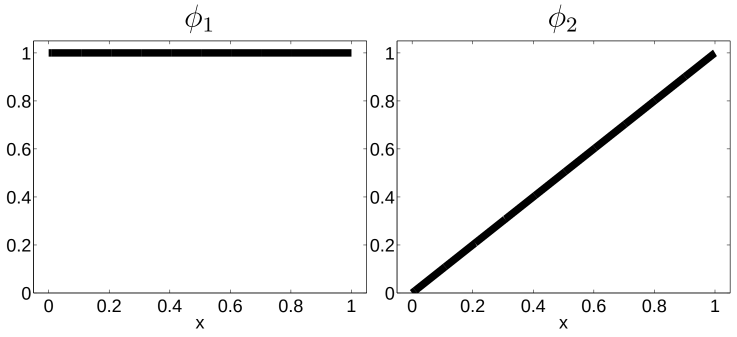


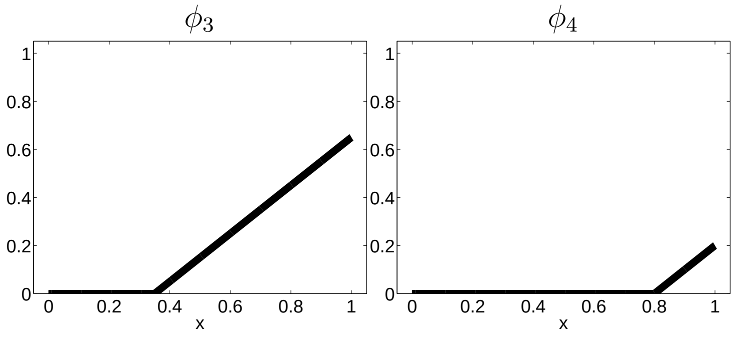
with $\mathbf{w} \approx (0.567, .39, -.03, -.32, -.01, .56)^T$.

This works, but it has one seeming downside: as a function of x, the model that is fit is still discontinuous. One way to enforce continuity would be to incorporate some constraints on the weights while fitting $\mathbf{w}$. This can work, but it a bit messy. It turns out that one can instead create a basis expansion that incorporates these constraints into it. This can be done with the basis expansion

$$\begin{aligned}
\phi_1(x) &= 1\\
\phi_2(x) &= x\\
\phi_3(x) &= (x-\xi_1)_+\\
\phi_4(x) &= (x-\xi_2)_+\\
&\vdots\\
\phi_{p+1}(x) &= (x-\xi_{p-1})_+. \qquad (3.1)
\end{aligned}$$

The notation $(a)_+$means $\max(a,0)$.

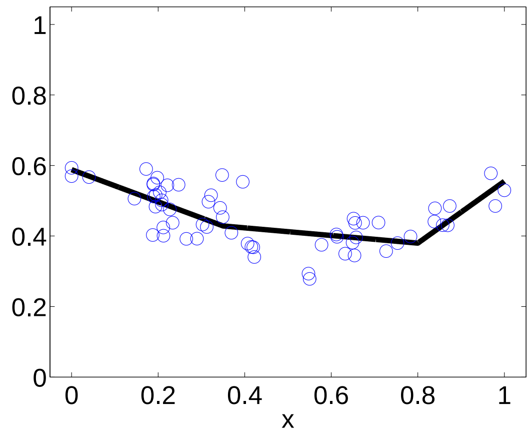




We claim two things here:

1. Any mixture of these basis functions will be continuous. This is pretty clear, since each of the basis functions themselves is continuous.

2. We can represent any piecewise constant function, with "ties" at $\xi_1, \xi_2, ..., \xi_{p-1}$. Let's convince ourselves of this. To start out with, in the region where $x \leq \xi_1$, we just have a generic linear function $w_1 + w_2 x$, which can match any linear function. Then, at ξ_1, the new function $(x - \xi_1)_+$ "kicks in". So, in the region $\xi_2 \leq x \leq \xi_3$, we have the function $w_1 + w_2 x + w_3 (x - \xi_1)$. Clearly, by setting w_3 appropriately, we can achieve any slope in that region, while remaining continuous at $x = \xi_1$. Similarly, when the function $(x - \xi_2)_+$ "kicks in" at $x = \xi_2$, we can change slope again.



with $\mathbf{w} \approx (.58, -.45, .34, .98)^T$.

Notice here that as well as enforcing continuity in the function, we have also reduced the number of free parameters, and so presumably reducing variance.

A model $f(x) = \mathbf{w} \cdot \boldsymbol{\phi}(x)$ with $\boldsymbol{\phi}$ as in Eq. 3.1 is known as a **linear spline**. There are extensions that fit higher-order models (quadratic or cubic) while enforcing not just continuity at the knot value ξ_i, but also continuous first or second-order derivatives. It is rare to go above a third-degree model, known as a "cubic spline".

4 Conclusions

Basis expansions are most commonly used with linear methods, though they can also be applied with neural networks, decision trees, etc. Some methods, such as neural networks, can be seen as *adaptively* finding a basis expansion, as we will discuss later.

We will see a great deal more about basis expansions when we discuss support vector machines. Under one interpretation, SVMs can be seen as fitting linear models for certain basis expansions with an *infinite* number of components.

For real-world problems, use of basis expansions can be something of a black art. What particular expansion performs well depends hugely on the problem at hand. Many people believe that hand-engineered "features" are the single-most important factor in getting good real-world performance. Because there is essentially no restriction on what features can be used, there is great room for creativity. Many research papers are written, essentially proposing new basis expansions. (Though they usually aren't phrased that way.)

An excellent example is David Lowe's SIFT features, proposed for computer vision. Roughly speaking, these consist of *histograms* of the directions of image gradients, at multiple scales. These perform amazingly well in practice, and are possibly the most influential idea in computer vision in the last 15-20 years.

Because basis expansions are so application specific, there isn't too much we can say in general. However, this does not mean that these methods always work better. Rather than using a fancy method, you carefully consider using smart basis expansions instead.