COS 424: Interacting with Data

Lecturer: Rob Schapire Lecture #14 Scribe: Zia Khan April 3, 2007

Recall from previous lecture that in regression we are trying to predict a real value given our data. Specifically, in the New Jersey school district example discussed in class, we looked at a linear model ˆ*f*(*xi*) = *wxi* where we related *xi* enrollment in district *i* and tried to predict the budget *yi* for district *i* based on *w* the money spent per student. In general, we would like to handle multiple dimensions or t a more complex model to the data.

1 Linear Regression

To generalize our original model we start with a vector x *∈* R*n* where *n* is the dimensionality of the data. We use the notation *xi*(*j*) to denote the *j*th component of the *i*th vector xi. We can write all of the components of

xi = *hxi*(1)*, . . . , xi*(*n*)*i*

which are referred to as variables, predictors, features or attributes - depending on the data we are interested in modeling. In general, the linear model will look as follows

ˆ*f*(x) = *w*1*x*(1) + *wwx*(2) + *. . .* + *wn*(*n*) = w *·* x

where *n* is the dimensionality of the data. To t our model we want to minimize

X*m i*=1

ˆ*f*(*xi*) *− yi* 2=X*m i*=1

(w *·* xi *− yi*)2

where *m* is the number of data points. The process of fitting this model is called linear regression.

If we look back at the New Jersey school district data, we might want to model *w*0 the xed cost for students as well as *w*1 the cost per student. Consequently, we should t the modelˆ*f*(*xi*) = *w*0 + *w*1*xi*

to our data set where *xi*is the enrollment in the district. To t this model we use a trick where we replace each data point with a 2-dimensional vector *xi → h*1*, xii*. It is easy to see that when we add 1 as the rst dimension of each vector xi, we obtain the model above.

When we fit the model to the data, we obtain *w*0 = *−*3*,* 540*,* 476 for the fixed cost for students and *w*1 = 12*,* 054 for the cost per student. The fixed cost number makes little sense. We can fix this by allowing for larger variance in the cost for school districts and

we obtain *w*0 = 99*,* 138 for the fixed cost for students and *w*1 = 12*,* 054 for the cost per student. This illustrates one of the many ways we can adjust our data so that we obtain a model that is a better t for our data.

We can take this further and consider a model where we account for the higher cost of special education students

ˆ*f*(*xi*) = *w*0 + *w*1*xi*(1) + *w*2*xi*(2)

where *xi*(1) is the number of regular students and *xi*(2) is the number of special education students in district *i*. When we t our model we prepend one to the vector *hxi*(1)*, xi*(2)*i → h*1*, xi*(1)*, xi*(2)*i* to account for *w*0 the xed cost for students. With this higher dimensional model we get *w*0 = 154*,* 192 for the xed cost for students and *w*1 = 8*,* 495 for the cost per regular student and a larger *w*2 = 35*,* 288 cost per special education student.

2 Linear Model

Now we consider the general problem of tting a vector





to data. To do so we de ne a matrix

w =



*w*1... *wn*

X =



x1

x2...

xm

 

that has *m* rows and *n* columns where *m* is the number of data points and *n* is the dimen sionality of the data. Thus, the *i*th row of X is equal to the data vector xi. We also de ne

a vector

y =



*y*1... *ym*

 

where *yi*is the predicted value associated with the *i*th data vector xi. With these de nitions we can rephrase the minimization in terms of matrices and vectors

Xw *−* y =



w *·* x1 *− y*1

...

w *·* xm *− ym* 2

 

The Euclidean length squared, or *L*2-norm, of the vector on the right hand side is just our original objective for linear regression

X*m i*=1

(w *·* xi *− yi*)2 = *k*Xw *−* y*k*22(1)

We can minimize this objective by multiplying out the norm w(Xw *−* y)*>*(Xw *−* y) = min

min

and computing the gradient

ww*>*X*>*Xw *−* 2w*>*X*>*y + y*>*y

*5*w = 2X*>*Xw *−* 2X*>*y = 0

and setting it equal to zero. Now we can solve for w by computing the inverse

X*>*Xw = X*>*y

w =

X*>*X *−*1X*>*y

The *n* by *m* matrix X*>*X *−*1 X*>* is referred to as the pseudo-inverse. Of course, there is no guarantee that the pseudo-inverse will exist. It only exists when X*>*X *−*1is non-singular, and in this case, the solution w is unique. However, even when X*>*X is singular, there are techniques for computing the minimum of equation (1).

It is not all that limiting to use just a linear model. Many problems can be reduced to the linear case. In this lecture we considered two: the polynomial model and linear splines.

2.1 Fitting Polynomials

We can use linear models to t polynomial models. In Figure 1, we t a cubic model

ˆ*f*(*x*) = *w*0 + *w*1*x* + *w*2*x* + *w*3*x*3

by mapping each data point to a vector *xi →*1*, xi, x*2*i, x*3*i*and tting a linear model as described above. If the data is in *n >* 1 dimensions and we want to t a polynomial we use a similar mapping. For instance, for *n* = 2 we can t a quadratic

ˆ*f*(*x*1*, x*2) = *a* + *bx*1 + *cx*2 + *dx*1*x*2 + *ex*21 + *fx*22

using the mapping *hx*1*, x*2*i →* 1*, x*1*, x*2*, x*1*x*2*, x*21*, x*22 . In general, if we start with *n* di mensions and want to t a degree *k* polynomial we will need to map our data up to *O*(*nk*) dimensions. As with SVMs we can solve the problem of explicitly mapping data vectors in higher dimensions by using the Kernel trick where we replace an inner product

3

Figure 1: We can use linear regression to do polynomial regression. To t a cubic polynomial to each data point *xi* we map the data point to a vector *xi →*1*, xi, x*2*i, x*3*i*.

xi*·* xj = *K*(xi*,* xj) with a kernel function. For the polynomial model we would use the kernel *K*(xi*,* xj) = (1 + xi*·* xj)*k*. To obtain a procedure called kernel regression, our orig

inal objective must be rewritten in terms of inner products minP*mi*=1 (w *·* xi *− yi*)2. Of course, we face the same problem in kernel regression as we do in SVMs. As we increase the dimensionality of the data points we require more data.

2.2 Linear Splines

Linear splines o er another example of how to reduce a problem to linear regression. Here, we are trying to build a piece-wise linear function to t the data (see Figure 2). For our discussion, we assume the knots, *k*1 and *k*2 in the example in Figure 2, are known in advance. We can imagine splitting the data points up into three groups using the given knots *k*1and *k*2 and solve three separate regression problems, but this does not assure continuity.

If we want the curve to be continuous, we can reduce the problem to straight linear regression by noticing that any linear spline can be a simple linear combination of basis functions. In the example in Figure 3, given knots *k*1 and *k*2 we can build up a piece-wise linear function step by step. We rst start with a linear function

ˆ*f*(*x*) = *a* + *bx*

which we use to account for points before the rst knot *k*1 (green line in Figure 3). Next, we add second line, the blue line in Figure 3, that is zero up until the rst knot *k*1

ˆ*f*(*x*) = *a* + *bx* + *c*(*x − k*1)+

4

k1 k2

Figure 2: The aim of linear splines is to t a piece-wise linear function such as the one shown in red. We can imagine splitting the data points up into three groups using the given knots *k*1and *k*2 and solve three separate regression problems. This, however, does not assure continuity.

k1 k2

Figure 3: Any linear spline can be a simple combination of linear basis functions. To t linear splines given the knots *k*1and *k*2 we map each point *xi* we map the data point to a vector *xi → h*1*, x,*(*x − k*1)+*,*(*x − k*2)+*i* and solve the standard linear regression problem.

5

where

(*z*)+ =

*z* if *z ≥* 0 0 if *z <* 0

Last, we add a third line, the purple line in Figure 3, that is zero up until the second knot *k*2

ˆ*f*(*x*) = *a* + *bx* + *c*(*x − k*1)+ + *d*(*x − k*2)+

The coe cients in the function can be found by mapping each data point to a vector *x → h*1*, x,*(*x − k*1)+*,*(*x − k*2)+*i* and solving the standard linear regression problem. This approach can generalize to higher order splines. It can incorporate additional constraints and generalize to higher dimensions.

3 Over tting

3.1 Feature Selection

As we increase the dimensionality of our data, we run into the problem of over tting. It is at least as bad in regression as it is in classi cation. The aim of feature selection is to nd a subset of features/dimensions/variables that are best in some sense for our data. One can try to enumerate all 2*n*features but for a large number of features this can be computationally expensive. Often one tries a greedy search method where features are added one-by-one to improve the objective or features are removed one-by-one to improve the objective.

3.2 Regularization

Another approach is to blame over tting on weights that are too large. With large weight vectors it is easier to over t our data. The idea is to constrain weight vectors in some way. Regularization refers to any time we add a penalty term that involves the weight vector. Shrinkage refers to any technique where we are shrinking the weights in some way.

One example of a regularization technique is ridge regression. Here we optimize the

objective

min

w

X*m i*=1

(w *·* xi *− yi*)2 + *λ k*w*k*22

where the smoothing constant *λ* is chosen in advance. Larger values of *λ* tend to smooth curves by penalizing larger weight values through the *L*2 norm of the weight vector *k*w*k*. Finding a weight vector w for ridge regression is not hard. We can nd a closed for solution in which we modify the pseudo-inverse

w =

where I is an *n* by *n* identity matrix.

X*>*X + *λ*I *−*1X*>*y 6