Multivariate polynomial regression with numpy

Asked 7 years, 11 months ago Active today Viewed 38k times

21

I have many samples `(y_i, (a_i, b_i, c_i))` where `y` is presumed to vary as a polynomial in `a,b,c` up to a certain degree. For example for a given set of data and degree 2 I might produce the model

```
y = a^2 + 2ab - 3cb + c^2 +.5ac
```

8

This can be done using least squares and is a slight extension of numpy's polyfit routine. Is there a standard implementation somewhere in the Python ecosystem?

python numpy statistics regression

asked Jun 11 '12 at 21:55

MRocklin
41k 14 106 171

2 I've posted code here to solve this problem https://github.com/mrocklin/multipolyfit – MRocklin Jul 4 '12 at 23:55

3 Answers

Active | Oldest | Votes

16

sklearn provides a simple way to do this.

Building off an example posted here:

```
#X is the independent variable (bivariate in this case)
X = array([[0.44, 0.68], [0.99, 0.23]])

#vector is the dependent data
vector = [109.85, 155.72]

#predict is an independent variable for which we'd like to predict the value
predict= [0.49, 0.18]

#generate a model of polynomial features
poly = PolynomialFeatures(degree=2)

#transform the x data for proper fitting (for single variable type it returns,
[1,x,x**2])
X_ = poly.fit_transform(X)

#transform the prediction to fit the model type
predict_ = poly.fit_transform(predict)

#here we can remove polynomial orders we don't want
#for instance I'm removing the `x` component
X_ = np.delete(X_,(1),axis=1)
predict_ = np.delete(predict_,(1),axis=1)

#generate the regression object
```

```
print("X_ = ",X_)
print("predict_ = ",predict_)
print("Prediction = ",clf.predict(predict_))
```

And heres the output:

```
>>> X_ =  [[ 0.44    0.68    0.1936  0.2992  0.4624]
>>>  [ 0.99    0.23    0.9801  0.2277  0.0529]]
>>> predict_ =  [[ 0.49    0.18    0.2401  0.0882  0.0324]]
>>> Prediction =  [ 126.84247142]
```

edited Jun 12 '17 at 17:39

answered Jul 17 '15 at 2:23

David Hoffman
1,074 1 8 23

Would it be possible for you to include the implementation of the `delete` function? Cheers! – Shivam Gaur Jun 12 '17 at 5:55

1 Sorry, it's numpy, docs.scipy.org/doc/numpy/reference/generated/numpy.delete.html – David Hoffman Jun 12 '17 at 17:40

what does `PolynomialFeatures` do explicitly? can I see the code? – Charlie Parker Aug 26 '17 at 16:01

1 this doesn't make sense to me, why does `fit_transform` return both the polynomial feature matrix (vandermonde matrix) AND also the predictions? :/ – Charlie Parker Oct 25 '17 at 18:51

how does this compare to just doing it manually like `c_pinv = np.dot(np.linalg.pinv( Kern_train ),Y_train)` ? – Charlie Parker Oct 25 '17 at 18:54

2

polyfit does work, but there are better least square minimizers out there. I would recommend kmpfit, available at

http://www.astro.rug.nl/software/kapteyn-beta/kmpfittutorial.html

It is more robust that polyfit, and there is an example on their page which shows how to do a simple linear fit that should provide the basics of doing a 2nd order polynomial fit.

```
def model(p, v, x, w):
   a,b,c,d,e,f,g,h,i,j,k = p      #coefficients to the polynomials
   return  a*v**2 + b*x**2 + c*w**2 + d*v*x + e*v*w + f*x*w + g*v + h*x + i*y + k

def residuals(p, data):        # Function needed by fit routine
   v, x, w, z = data            # The values for v, x, w and the measured
hypersurface z
   a,b,c,d,e,f,g,h,i,j,k = p   #coefficients to the polynomials
   return (z-model(p,v,x,w))   # Returns an array of residuals.
                               #This should (z-model(p,v,x,w))/err if
                               # there are error bars on the measured z values


#initial guess at parameters. Avoid using 0.0 as initial guess
par0 = [1.0, 1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0]

#create a fitting object. data should be in the form
```

```
# call the fitter
fitobj.fit(params0=par0)
```

The success of these things is closely dependent on the starting values for the fit, so chose carefully if possible. With so many free parameters it could be a challenge to get a solution.

edited Jun 12 '12 at 15:23

answered Jun 11 '12 at 22:17

reptilicus
9,397 5 44 71

1 Can you post an example of multivariate regression using polyfit? I'm not convinced that this is supported. After looking through the documentation for kmpfit I fear this might be true of this library as well. – MRocklin Jun 11 '12 at 22:33

What are you trying to fit, y(x) = a*x**2 + bx + c? Anyway, you can certainly do multivariable fitting with mpfit/kmpfit. – reptilicus Jun 12 '12 at 13:47

No, y(v, x, w) = a*v2* + ***bx***2 + c*w**2* + d*vx* + e*vw* + f*xw* + g*v* + h*x* + i*y* + k – MRocklin Jun 12 '12 at 14:39

2 So this library would work but it solves the problem through an iterative method. Least squares polynomial fitting can be done in one step by solving a linear system. I've posted code in another answer that does this using numpy. – MRocklin Jul 6 '12 at 14:22

1

sklearn has a nice example using their Pipeline here. Here's the core of their example:

```
polynomial_features = PolynomialFeatures(degree=degrees[i],
                                         include_bias=False)
linear_regression = LinearRegression()
pipeline = Pipeline([("polynomial_features", polynomial_features),
                     ("linear_regression", linear_regression)])
pipeline.fit(X[:, np.newaxis], y)
```

You don't need to transform your data yourself -- just pass it into the Pipeline.

answered Sep 30 '16 at 15:09

amos
3,876 3 25 31

3 That example does not use multivariate regression. – user4322543 Nov 20 '16 at 20:28