8. Nonlinear Regression

Bruce E. Shapiro

Getting Started in Machine Learning

Last revised: April 7, 2019

Nonlinear Regression: Goal

- Fit $(x_0, y_0), \ldots, (x_{n-1}, y_{n-1})$ to $y = f(x)$ in such a way that the objective function

$$\mathcal{E} = \sum_{i=0}^{n-1} |f(x_i; a_0, a_1, a_2, \ldots) - y_i|^2$$

  is minimized. Here $a_0, a_1, \ldots$ are parameters that determine the function $f(x)$.

- Some of the the $\partial\mathcal{E}/\partial a_i$ are nonlinear in the $a_0, a_1, \ldots, a_{n-1}$.

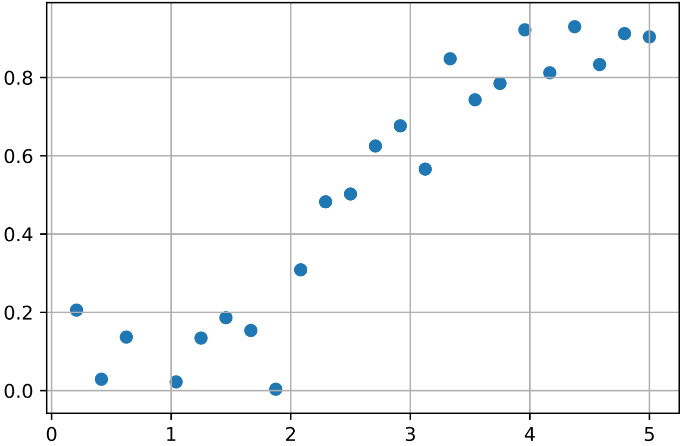
Nonlinear Regression: Goal

- Fit $(x_0, y_0), \ldots, (x_{n-1}, y_{n-1})$ to $y = f(x)$ in such a way that the objective function

$$\mathcal{E} = \sum_{i=0}^{n-1} |f(x_i; a_0, a_1, a_2, \ldots) - y_i|^2$$

  is minimized. Here $a_0, a_1, \ldots$ are parameters that determine the function $f(x)$.
- Some of the the $\partial\mathcal{E}/\partial a_i$ are nonlinear in the $a_0, a_1, \ldots, a_{n-1}$.
- For example, **Hill function** $f(x) = \dfrac{x^k}{a^k + x^k}$
  The derivatives $\partial\mathcal{E}/\partial k$ and $\partial\mathcal{E}/\partial a$ are nonlinear in k and a.

- **Example**. Toy data set that looks like a hill function.



Gradient Descent Algorithm

- Nonlinear Objective Function $\mathcal{E} = \sum \left[\frac{x_i^m}{a^m + x_i^m} - y_i \right]^2$
- Iterate using

$$x_{n+1} = x_n - \eta f'(x)$$

where η is a parameter

Gradient Descent Algorithm

- Nonlinear Objective Function $\mathcal{E} = \sum \left[\frac{x_i^m}{a^m + x_i^m} - y_i \right]^2$
- Iterate using

$$x_{n+1} = x_n - \eta f'(x)$$

  where η is a parameter
- Multidimensional version is

Gradient Descent Algorithm

- Nonlinear Objective Function $\mathcal{E} = \sum \left[\frac{x_i^m}{a^m + x_i^m} - y_i \right]^2$
- Iterate using

$$x_{n+1} = x_n - \eta f'(x)$$

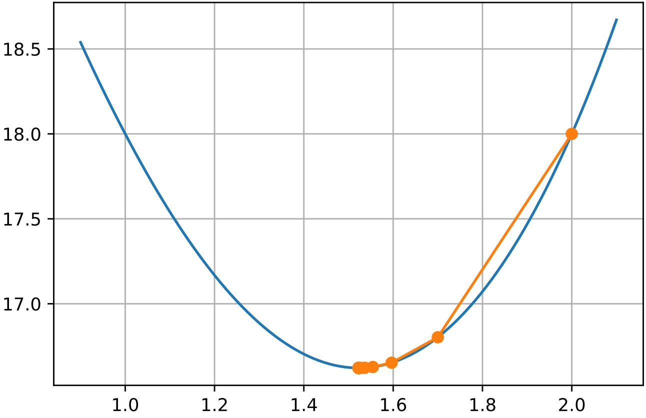
  where η is a parameter
- Multidimensional version is

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \eta \nabla f(\mathbf{x})$$

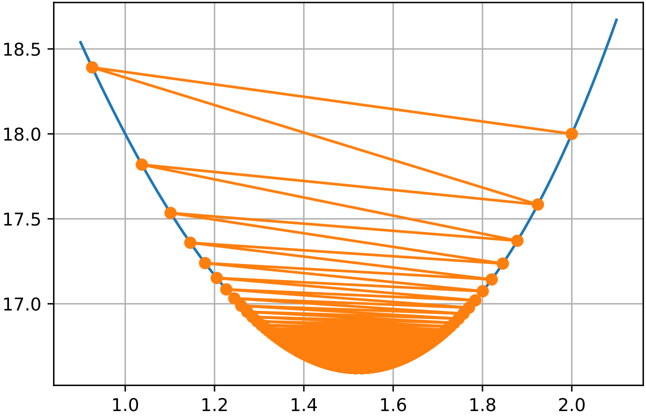
- Example: $f(x) = (x-5)^2 + x^3 + 1$,
  minimum is at $x \approx 1.522588$.
  Start at $x_0 = 2$, use $\eta = 0.05$.

i	x_i	$f'(x_i)$	$x_{i+1} = x_i - 0.05f'(x_i)$
0	2.0000000	6.0000000	1.7000000
1	1.7000000	2.0700000	1.5965000
2	1.5965000	0.8394367	1.5545282
3	1.5545282	0.3587297	1.5365917
4	1.5365917	0.1565253	1.5287654
5	1.5287654	0.0689019	1.5253203
6	1.5253203	0.0304469	1.5237980
7	1.5237980	0.0134768	1.5231241
8	1.5231241	0.0059697	1.5228257
9	1.5228257	0.0026452	1.5226934
10	1.5226934	0.0011723	1.5226348
11	1.5226348	0.0005196	1.5226088
12	1.5226088	0.0002303	1.5225973
13	1.5225973	0.0001021	1.5225922
14	1.5225922	0.0000452	1.5225899

$\eta = 0.05$, 27 iterations



$\eta = 0.179$, Gradient Descent



Newton's Method Fix to Gradient Descent

- Let x_n be a guess. Taylor series about x_n:

$$\begin{aligned} f(x) &\approx f(x_n) + f'(x_n)\Delta x + \frac{1}{2} f''(x_n)(\Delta x)^2 \\ &\approx f(x_n) + f'(x_n)(x - x_n) + \frac{1}{2} f''(x_n)(x - x_n)^2 \end{aligned}$$

Newton's Method Fix to Gradient Descent

- Let x_n be a guess. Taylor series about x_n:

$$\begin{aligned} f(x) &\approx f(x_n) + f'(x_n)\Delta x + \frac{1}{2} f''(x_n)(\Delta x)^2 \\ &\approx f(x_n) + f'(x_n)(x - x_n) + \frac{1}{2} f''(x_n)(x - x_n)^2 \end{aligned}$$

- $f(x_n)$, $f'(x_n)$, and $f''(x_n)$ are constants:

$$f'(x) \approx f'(x_n) + f''(x_n)(x - x_n)$$

Newton's Method Fix to Gradient Descent (Continued)

- Ideally, jump to minimum at x_{n+1}
- If x_{n+1} is a minimum then $f'(x_{n+1}) = 0$

Newton's Method Fix to Gradient Descent (Continued)

- Ideally, jump to minimum at x_{n+1}
- If x_{n+1} is a minimum then $f'(x_{n+1}) = 0$
- Evaluate Taylor series at x_{n+1}:

$$0 = f'(x_{n+1}) = f'(x_n) + f''(x_n)(x_{n+1} - x_n)$$

- Solve for x_{n+1}:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

- Multidimensional Equation:

$$x_{n+1} = x_n - \mathbf{H}^{-1} \nabla f(x_n)$$

Example: Hill Function

- Nonlinear Objective Function $\mathcal{E} = \sum \left[\frac{x_i^m}{a^m + x_i^m} - y_i \right]^2$
- Gradient:

$$\nabla \mathcal{E} = \left(\frac{\partial \mathcal{E}}{\partial a}, \frac{\partial \mathcal{E}}{\partial m} \right)$$

- Gradient of $\mathcal{E}$ requires partial derivatives of Hill function:

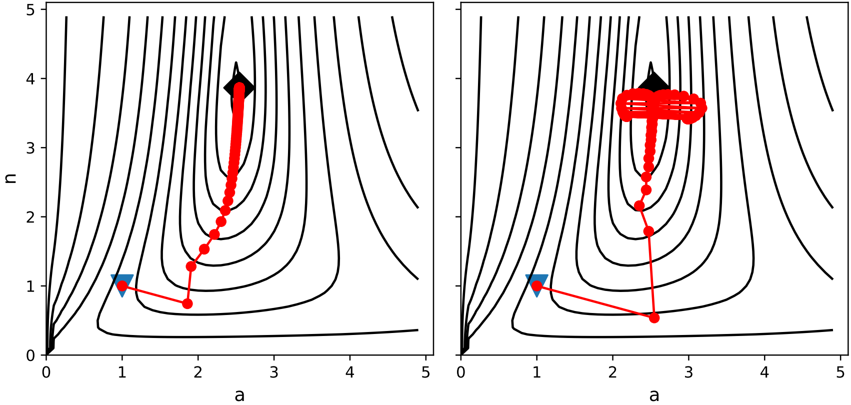
$$\frac{\partial \mathcal{E}}{\partial a} = 2 \sum_i \left[\frac{x_i^m}{a^m + x_i^m} - y_i \right] \frac{\partial}{\partial a} \left[\frac{x_i^m}{x_i^m + a^m} \right]$$

$$\frac{\partial \mathcal{E}}{\partial m} = 2 \sum_i \left[\frac{x_i^m}{a^m + x_i^m} - y_i \right] \frac{\partial}{\partial m} \left[\frac{x_i^m}{x_i^m + a^m} \right]$$

- Partial derivatives of Hill Function:

$$\frac{\partial}{\partial a}\left[\frac{x_i^m}{x_i^m + a^m}\right] = \frac{-m x_i^m a^{m-1}}{(a^m + x_i^m)^2}$$
$$\frac{\partial}{\partial m}\left[\frac{x_i^m}{x_i^m + a^m}\right] = \frac{(a^m + x_i^m) x_i^m \ln x_i - x_i^m (a^m \ln a + x_i^m \ln x_i)}{(a^m + x_i^m)^2}$$

- See textbook or notebook for implementation
- **`scipy.optimize import curve_fit`** uses a variation on gradient descent



n
a
a
0
1
2
3
4
5

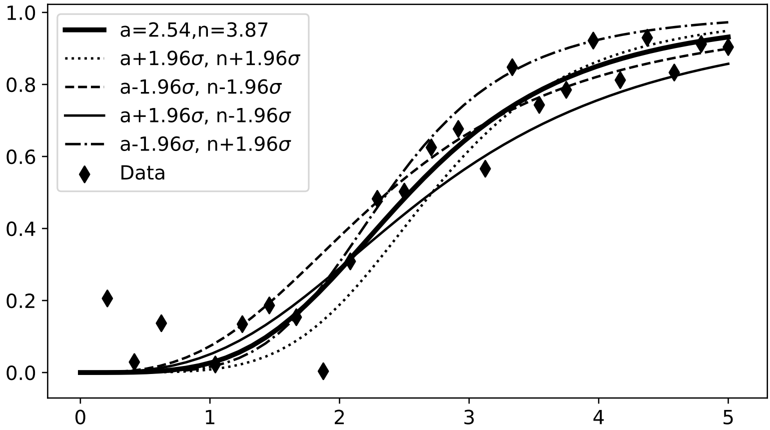
```
from scipy.optimize import curve_fit
def fhill(x,a=1,n=1):
    return x**n/(a**n+x**n)
```

```
parameters,covmatrix=curve_fit(fhill,x,y,p0=(1,1))
afit,nfit=parameters
asig,nsig = np.sqrt(np.diag(covmatrix))
print("a = ",round(afit,2),"+/-",round(1.96*asig,2))
print("n = ",round(nfit,2),"+/-",round(1.96*nsig,2))
```

```
a =  2.54 +/- 0.17
n =  3.87 +/- 0.93
```

```
covmatrix
```

```
array([[0.00762362, 0.00992891],
       [0.00992891, 0.22447413]])
```


a=2.54,n=3.87
a+1.96σ, n+1.96σ
a-1.96σ, n-1.96σ
a+1.96σ, n-1.96σ
a-1.96σ, n+1.96σ
Data
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5