Modeling with Python and SMAP

Sarah Koehler

April 2015

1 Modeling

Many say modeling is an art. Let's look at the simplest forms of this art.

Linear Model:

$$x(k+1) = Ax(k) + Bu(k) \tag{1}$$
$$y(k) = Cx(k) + Du(k) \tag{2}$$

Let's assume $y(k) = x(k)$.

In the most general case, define x to be all physical *outputs* and u to be all *controllable inputs*. You can also add an additive $+Ed(k)$ where d are all *uncontrollable inputs*.

A form that works well for Temperature dynamics is a bilinear form. Specifically:

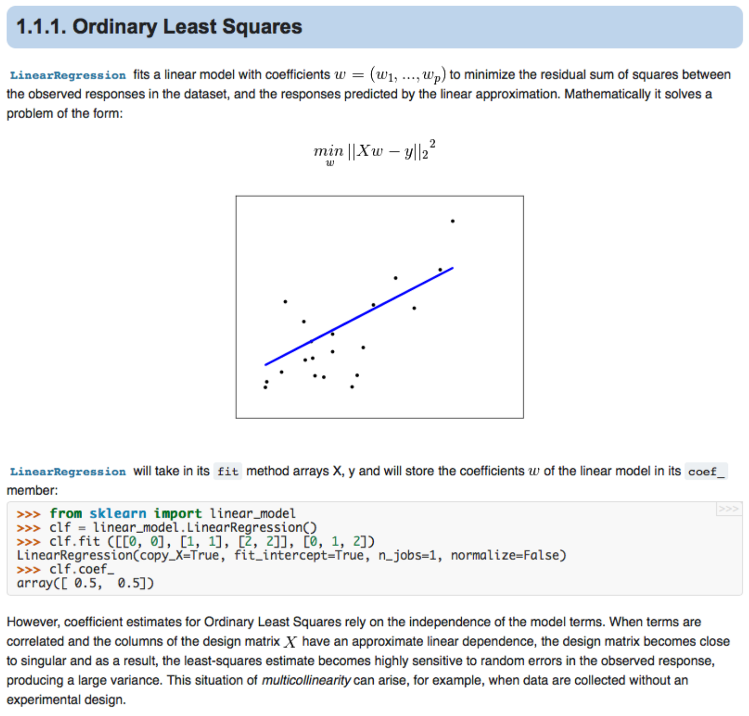
$$T(k+1) = AT(k) + ... \tag{3}$$

(TO DO: To be copied from ME 231 write ups)

2 Identification Method

Two methods are Least Squares and Lasso. An oversimplified explanation is that Least Squares considers all data streams equally important, while Lasso selects the important streams (i.e. feature selection). Both select the corresponding weight for that stream.

Least Squares from Python scikit-learn:



1.1.1. Ordinary Least Squares

`LinearRegression` fits a linear model with coefficients $w = (w_1, ..., w_p)$ to minimize the residual sum of squares between the observed responses in the dataset, and the responses predicted by the linear approximation. Mathematically it solves a problem of the form:

$$\underset{w}{min} \, ||Xw - y||_2{}^2$$

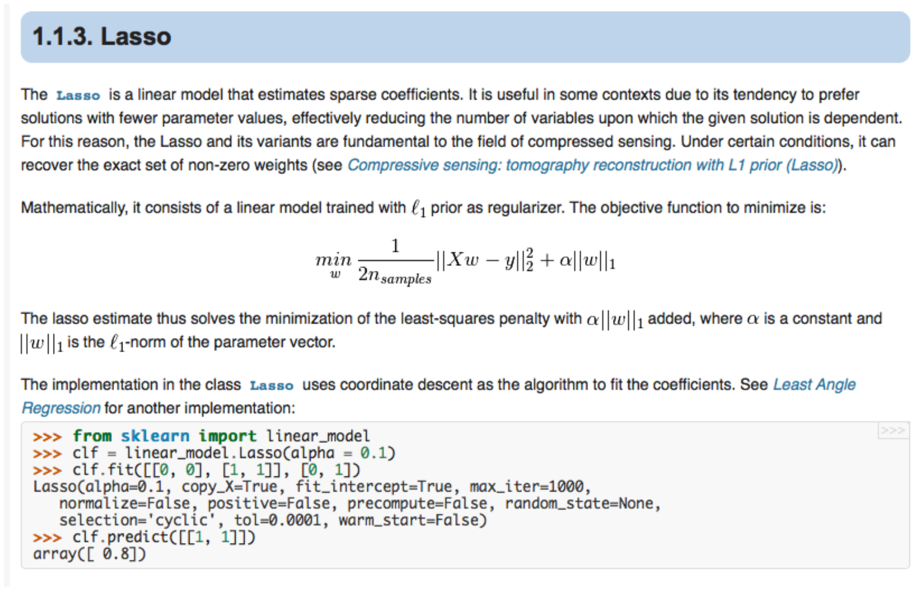
`LinearRegression` will take in its `fit` method arrays X, y and will store the coefficients w of the linear model in its `coef_` member:

```
>>> from sklearn import linear_model
>>> clf = linear_model.LinearRegression()
>>> clf.fit ([[0, 0], [1, 1], [2, 2]], [0, 1, 2])
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
>>> clf.coef_
array([ 0.5,  0.5])
```

However, coefficient estimates for Ordinary Least Squares rely on the independence of the model terms. When terms are correlated and the columns of the design matrix X have an approximate linear dependence, the design matrix becomes close to singular and as a result, the least-squares estimate becomes highly sensitive to random errors in the observed response, producing a large variance. This situation of *multicollinearity* can arise, for example, when data are collected without an experimental design.

Figure 1: Documentation on Least Squares

Lasso from Python scikit-learn:



1.1.3. Lasso

The `Lasso` is a linear model that estimates sparse coefficients. It is useful in some contexts due to its tendency to prefer solutions with fewer parameter values, effectively reducing the number of variables upon which the given solution is dependent. For this reason, the Lasso and its variants are fundamental to the field of compressed sensing. Under certain conditions, it can recover the exact set of non-zero weights (see *Compressive sensing: tomography reconstruction with L1 prior (Lasso)*).

Mathematically, it consists of a linear model trained with ℓ_1 prior as regularizer. The objective function to minimize is:

$$\underset{w}{min} \frac{1}{2n_{samples}}||Xw - y||_2^2 + \alpha||w||_1$$

The lasso estimate thus solves the minimization of the least-squares penalty with $\alpha||w||_1$ added, where α is a constant and $||w||_1$ is the ℓ_1-norm of the parameter vector.

The implementation in the class `Lasso` uses coordinate descent as the algorithm to fit the coefficients. See *Least Angle Regression* for another implementation:

```
>>> from sklearn import linear_model
>>> clf = linear_model.Lasso(alpha = 0.1)
>>> clf.fit([[0, 0], [1, 1]], [0, 1])
Lasso(alpha=0.1, copy_X=True, fit_intercept=True, max_iter=1000,
   normalize=False, positive=False, precompute=False, random_state=None,
   selection='cyclic', tol=0.0001, warm_start=False)
>>> clf.predict([[1, 1]])
array([ 0.8])
```

Figure 2: Documentation on Lasso

3 Data Manipulation

This is the most annoying part and it would be great to never ever have to do this again.

TO DO: Copy python code when it's done and/or write the math here.

4 Validation

Cross-validation is important! The model usually fits the day it was identified on.

TO DO: Show some plots of same day verification and cross day verification.

5 Matlab code

```
% % Interpolate data
% interp_start = PDTtoUTC(datenum(2014,9,4,4,0,0));
% interp_end = PDTtoUTC(datenum(2014,9,4,16,0,0));
interp_start = datenum(2014,8,30,10,1,0);
interp_end = datenum(2014,8,30,16,0,0);
interp_times = (interp_start:1/24/60:interp_end)';


fan_interp = interp1(fan_data(:,1),fan_data(:,2),interp_times);
temp_interp = interp1(temp_data(:,1),temp_data(:,2),interp_times);
supplytemp_interp = interp1(supplytemp_data(:,1),supplytemp_data(:,2),interp_times);

% Solve least squares using quadprog
N = length(interp_times)-2;
H = 2*(blkdiag(zeros(2,2),eye(N)));
f = zeros(N+2,1);
Aeq = [zeros(N,2),-eye(N)];
beq = zeros(N,1);
for k = 1:N
   f(k+2) = -2*temp_interp(k+1);
   Aeq(k,1:2) = [-fan_interp(k)*temp_interp(k),fan_interp(k)*supplytemp_interp(k)];
   beq(k) = -temp_interp(k);
end

% options = optimset('Algorithm','interior-point-convex');
% [x,cost] = quadprog(H,f,[],[],Aeq,beq,[],[],[],options);
[x,cost] = quadprog(H,f,[],[],Aeq,beq);

a = x(1);
b = x(2);

%% Least Squares via A\b
A = Aeq(:,1:2);
b = zeros(N,1);
for k = 1:N
   b(k) = -temp_interp(k)+temp_interp(k+1);
end

x = A\b;
a = x(1);
b = x(2);
```

6 Python code

At this point can open up Python code like Arka did and expose the issues we have with its structure.

TO DO: Can also make a list of issues here:

- Cannot choose system ID procedure without digging into code
- Need a library of models to choose from? Not everything is linear or bilinear...