Variable Selection

November 28, 2018

0.1 Variable Selection

We want to implement forward selection to get the list of best parameters out of six possibles. The possible parameters are the coefficients of the following linear regression model:

$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \theta_5 x_2^2$

The parameter list are represented as a python list, with integer numbers referring to each parameter number. That is: [1,3] represent the list of parameters θ_1 and θ_3.

As follows, we implement forward search in python and run the algorithm five times.

```
In [304]: import numpy as np

          f =  open("tutorial4_2.txt", "r")

          #Reading lines: help taken from PythonForBeginners.com
          lines = f.readlines()

          #Processing each line to separate the pair of numbers
          lines1 = [l[:-2].split(" ") for l in lines ]

          #converting to array to better processing
          data = np.array(lines1)

          #converting from string to float
          data = data.astype(float)

          #Separating data
          x1 = data[:,0]
          x2 = data[:,1]
          y = data[:,2]
          n_samples = data.shape[0]

          X = np.vstack(( np.ones((n_samples)),x1, x2, x1**2, np.multiply(x1,x2), x2**2)).T

          train_pct=0.5

          def split_train_test(X, y, train_pct):

              n_samples = X.shape[0]
```

```
    #shuffling indexes to separate train and test randoming
    idx = np.arange(0,n_samples)
    np.random.shuffle(idx)

    #creating test indexes
    train_idx = idx[:train_size]

    #creating test indexes
    test_idx = idx[train_size:]

    X_train = X[train_idx,]
    X_test = X[test_idx,]

    y_train = y[train_idx]
    y_test = y[test_idx]

    return X_train, X_test, y_train, y_test

def error (X, beta, y):

    '''Computes the loss of linear regression (MSE).
   The parameters are:
    - X is the matrix of features
    - beta is the vector of parameters for the linear regression
    - y is the target vector     '''
    X = np.matrix(X)
    y = np.matrix(y).T
    y_pred = X*beta
    out = np.mean(np.array(y_pred-y)**2)

    return out


def train_lr(X,y):

    '''Computes the parameter fitting for the linear regression
    using closed form'''

    X = np.matrix(X)
    y = np.matrix(y).T
    beta = np.linalg.inv(np.matrix(X).T*X)*X.T*y

    return beta

def forward_search(X,y):

    train_pct= 0.5
    X_train, X_test, y_train, y_test = split_train_test(X, y, train_pct)
```

```
    parameters_selected=[] ###list of parameters selected: empty at the beginning
    parameters = [0, 1,2,3,4,5] ### list of possible parameters (each one refers to
    e_allbest = 10000 #initial error
    v_best = 1 #initialition of the best
    parameters_tmp = [] #temporal list useful for inner loop

    while v_best !=-1:#stop when no parameter on the list improves performance

        v_best = -1
        e_best = e_allbest
        print("current selected",parameters_selected)

        for v in parameters:

            #add a new parameter to a temporary list of parameters
            parameters_tmp = parameters_selected + [v]

            #subsetting with the temporary list of parameters
            X_train_sub = X_train[:,parameters_tmp]
            X_test_sub = X_test[:, parameters_tmp]

            #fitting beta with linear regression
            beta = train_lr(X_train_sub, y_train)

            #test the fitted model with the new set of parameters
            e = error(X_test_sub, beta, y_test)

            if(e<e_best):
                v_best = v
                e_best = e

        ###if the new added parameter improves performance, then hold it
        if (e_best< e_allbest):
            parameters_selected.append(v_best)
            idx = parameters.index(v_best)
            parameters.pop(idx)
            e_allbest=e_best

    print("Final list of parameters:", parameters_selected)
```

In [305]:
```
print("Run 1:")
forward_search(X, y)
print("Run 2:")
forward_search(X, y)
print("Run 3:")
forward_search(X, y)
print("Run 4:")
```

```
        forward_search(X, y)
        print("Run 5:")
        forward_search(X, y)
```

```
Run 1:
current selected []
current selected [5]
current selected [5, 3]
Final list of parameters: [5, 3]
Run 2:
current selected []
current selected [5]
current selected [5, 3]
current selected [5, 3, 4]
current selected [5, 3, 4, 2]
Final list of parameters: [5, 3, 4, 2]
Run 3:
current selected []
current selected [3]
current selected [3, 5]
current selected [3, 5, 4]
current selected [3, 5, 4, 2]
current selected [3, 5, 4, 2, 1]
Final list of parameters: [3, 5, 4, 2, 1]
Run 4:
current selected []
current selected [5]
current selected [5, 3]
Final list of parameters: [5, 3]
Run 5:
current selected []
current selected [5]
current selected [5, 3]
current selected [5, 3, 4]
Final list of parameters: [5, 3, 4]
```

As we can see, the results are not always the same since there are some randomnes when splitting the dataset. The first two parameters added to the model are [3,5], that is, θ_3 and θ_5,since they represent the ground truth ($y = x_1^2 - x_2^2$).

To make improvements to the model and select only the true variables, we can set a threshold to the difference of the previous error and the current error, so that we avoid to add a new variable to the parameter set if the improvement in the error is too low. If the difference is too low (less than the threshold), we stop adding variables. As we show in the following run of the modified function, we can see that a threshold of 1 to the difference of errors (last_error - current_error), would give the desired subset of variables.

```
In [307]: def forward_search(X,y):
```

```
    train_pct= 0.5
    X_train, X_test, y_train, y_test = split_train_test(X, y, train_pct)

    parameters_selected=[] ###list of parameters selected: empty at the beginning
    parameters = [0, 1,2,3,4,5] ### list of possible parameters (each one refers to
    e_allbest = 10000 #initial error
    v_best = 1 #initialition of the best
    parameters_tmp = [] #temporal list useful for inner loop

    while v_best !=-1:#stop when no parameter on the list improves performance

        v_best = -1
        e_best = e_allbest
        print("current selected",parameters_selected)

        for v in parameters:

            #add a new parameter to a temporary list of parameters
            parameters_tmp = parameters_selected + [v]

            #subsetting with the temporary list of parameters
            X_train_sub = X_train[:,parameters_tmp]
            X_test_sub = X_test[:, parameters_tmp]

            #fitting beta with linear regression
            beta = train_lr(X_train_sub, y_train)

            #test the fitted model with the new set of parameters
            e = error(X_test_sub, beta, y_test)

            if(e<e_best):
                v_best = v
                e_best = e

        ###if the new added parameter improves performance, then hold it
        if (e_best< e_allbest):
            parameters_selected.append(v_best)
            idx = parameters.index(v_best)
            parameters.pop(idx)

            #calculating difference between previous and current error
            print("Difference of errors with added variable:", e_allbest-e_best)
            e_allbest=e_best


    print("Final list of parameters:", parameters_selected)

print("Run 1:")
```

```
forward_search(X, y)
print("Run 2:")
forward_search(X, y)
print("Run 3:")
forward_search(X, y)
print("Run 4:")
forward_search(X, y)
print("Run 5:")
forward_search(X, y)
```

```
Run 1:
current selected []
Difference of errors with added variable: 9987.19655222
current selected [5]
Difference of errors with added variable: 11.8894959469
current selected [5, 3]
Difference of errors with added variable: 0.00490031564319
current selected [5, 3, 4]
Final list of parameters: [5, 3, 4]
Run 2:
current selected []
Difference of errors with added variable: 9987.50941561
current selected [5]
Difference of errors with added variable: 11.6022202511
current selected [5, 3]
Difference of errors with added variable: 0.000961204328476
current selected [5, 3, 2]
Final list of parameters: [5, 3, 2]
Run 3:
current selected []
Difference of errors with added variable: 9987.67488341
current selected [5]
Difference of errors with added variable: 11.3577375064
current selected [5, 3]
Difference of errors with added variable: 0.000970433684546
current selected [5, 3, 2]
Final list of parameters: [5, 3, 2]
Run 4:
current selected []
Difference of errors with added variable: 9988.15582196
current selected [3]
Difference of errors with added variable: 10.9688193512
current selected [3, 5]
Final list of parameters: [3, 5]
Run 5:
current selected []
Difference of errors with added variable: 9987.20434462
current selected [5]
```

```
Difference of errors with added variable: 11.7893094799
current selected [5, 3]
Difference of errors with added variable: 0.00335311900379
current selected [5, 3, 4]
Difference of errors with added variable: 0.000171848791325
current selected [5, 3, 4, 2]
Final list of parameters: [5, 3, 4, 2]
```