Linear Regression Algorithms Demo

LAST UPDATED	LANGUAGE
21 Dec 2017, 2:30 AM	Python 2.7

This notebook shows:

- Install SystemML Python package and jar file
  - pip
  - SystemML 'Hello World'
- Example 1: Matrix Multiplication
- Load diabetes dataset from scikit-learn
- Example 2: Implement three different algorithms to train linear regression model
  - Algorithm 1: Linear Regression - Direct Solve (no regularization)
  - Algorithm 2: Linear Regression - Batch Gradient Descent (no regularization)
  - Algorithm 3: Linear Regression - Conjugate Gradient (no regularization)
- Example 3: Invoke existing SystemML algorithm script LinearRegDS.dml using MLContext API
- Example 4: Invoke existing SystemML algorithm using scikit-learn/SparkML pipeline like API
- Example 5: Invoking a Keras model with SystemML

Install SystemML Python package and jar file

In [ ]:
```
#!pip install --upgrade systemml
!pip install --upgrade https://github.com/niketanpansare/future_of_dat
a/raw/master/systemml-1.1.0-SNAPSHOT-python.tar.gz
!ln -s -f ~/.local/lib/python2.7/site-packages/systemml/systemml-java/
*.jar ~/data/libs/
```

In [1]:
```
!pip show systemml
```

```
Name: systemml
Version: 1.1.0
Summary: Apache SystemML is a distributed and declarative machine learn
ing platform.
Home-page: http://systemml.apache.org/
Author: Apache SystemML
Author-email: dev@systemml.apache.org
License: Apache 2.0
Location: /gpfs/global_fs01/sym_shared/YPProdSpark/user/scf4-b69284e162
5908-5ca7710237a9/.local/lib/python2.7/site-packages
Requires: Pillow, scikit-learn, pandas, scipy, numpy
```

Import SystemML API

In [2]:
```
sc.version
```

Out[2]: u'2.1.0'

In [3]:
```
from systemml import MLContext, dml
# Create a MLContext object
ml = MLContext(sc)
# And print the information of SystemML version
print(ml.info())
```

```
Archiver-Version: Plexus Archiver
```

```
Artifact-Id: systemml
Build-Jdk: 1.8.0_111
Build-Time: 2017-12-19 13:17:52 CST
Built-By: biuser
Created-By: Apache Maven 3.0.5
Group-Id: org.apache.systemml
Main-Class: org.apache.sysml.api.DMLScript
Manifest-Version: 1.0
Minimum-Recommended-Spark-Version: 2.1.0
Version: 1.1.0-SNAPSHOT
```

In [4]:

```
# Create a DML script for a Hello World' example and execute it using M
LContext
script = dml("""
print('Hello World');
""")
ml.execute(script)
```

```
Hello World
SystemML Statistics:
Total execution time:           0.001 sec.
Number of executed Spark inst:  0.
```

Out[4]: MLResults

In [5]:

```
# Let's modify the above script to get the Hello World string
script = dml("""
s = 'Hello World'
""").output("s")

hello_world_str = ml.execute(script).get("s")

print(hello_world_str)
```

```
SystemML Statistics:
Total execution time:           0.000 sec.
Number of executed Spark inst:  0.


Hello World
```

Import numpy, sklearn, and define some helper functions

In [6]:

```
import sys, os
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets
plt.switch_backend('agg')
```

Example 1: Matrix Multiplication

SystemML script to generate a random matrix, perform matrix

SystemML script to generate a random matrix, perform matrix multiplication, and compute the sum of the output

In [7]:
```
script = """
    X = rand(rows=$nr, cols=1000, sparsity=0.5)
    A = t(X) %*% X
    s = sum(A)
"""
prog = dml(script).input('$nr', 1e6).output('s')
s = ml.execute(prog).get('s')
print s
```

```
[Stage 0:>                                                          (0 +
0) / 59]
[Stage 0:>                                                          (0 +
1) / 59]
[Stage 0:>                                                         (0 +
10) / 59]
[Stage 0:>                                                         (1 +
10) / 59]
[Stage 0:==========>                                              (11 +
10) / 59]
[Stage 0:=================>                                       (18 +
10) / 59]
[Stage 0:=======================>                                 (25 +
10) / 59]
[Stage 0:================================>                        (34 +
10) / 59]
[Stage 0:=========================================>               (44 +
10) / 59]
[Stage 0:=====================================================>    (54 +
5) / 59]
SystemML Statistics:
Total execution time:           14.134 sec.
Number of executed Spark inst:  2.



62608781691.5
```

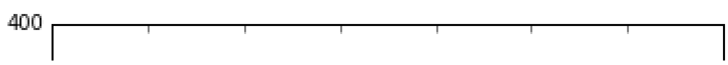
Load diabetes dataset from scikit-learn

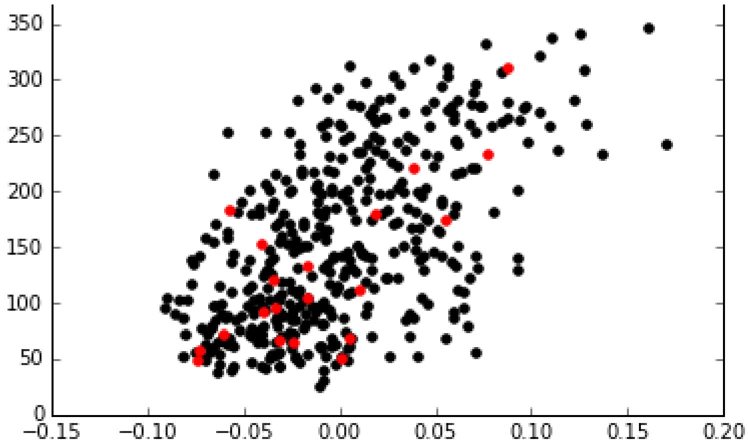
In [8]:
```
%matplotlib inline
```

In [9]:
```
diabetes = datasets.load_diabetes()
diabetes_X = diabetes.data[:, np.newaxis, 2]
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]
diabetes_y_train = np.matrix(diabetes.target[:-20]).T
diabetes_y_test = np.matrix(diabetes.target[-20:]).T

plt.scatter(diabetes_X_train, diabetes_y_train,  color='black')
plt.scatter(diabetes_X_test, diabetes_y_test,  color='red')
```

Out[9]:
```
<matplotlib.collections.PathCollection at 0x7f63be96d850>
```





In [10]:
```
diabetes_y_train
```

Out[10]:
```
matrix([[ 151.],
        [  75.],
        [ 141.],
        [ 206.],
        [ 135.],
        [  97.],
        [ 138.],
        [  63.],
        [ 110.],
        [ 310.],
        [ 101.],
        [  69.],
        [ 179.],
        [ 185.],
        [ 118.],
        [ 171.],
        [ 166.],
        [ 144.],
        [  97.],
        [ 168.],
        [  68.],
        [  49.],
        [  68.],
        [ 245.],
        [ 184.],
        [ 202.],
        [ 137.],
        [  85.],
        [ 131.],
        [ 283.],
        [ 129.],
        [  59.],
        [ 341.],
        [  87.],
        [  65.],
        [ 102.],
        [ 265.],
        [ 276.],
        [ 252.],
        [  90.],
        [ 100.],
        [  55.],
        [  61.],
        [  92.],
        [ 259.],
```

```
[  53.],
[ 190.],
[ 142.],
[  75.],
[ 142.],
[ 155.],
[ 225.],
[  59.],
[ 104.],
[ 182.],
[ 128.],
[  52.],
[  37.],
[ 170.],
[ 170.],
[  61.],
[ 144.],
[  52.],
[ 128.],
[  71.],
[ 163.],
[ 150.],
[  97.],
[ 160.],
[ 178.],
[  48.],
[ 270.],
[ 202.],
[ 111.],
[  85.],
[  42.],
[ 170.],
[ 200.],
[ 252.],
[ 113.],
[ 143.],
[  51.],
[  52.],
[ 210.],
[  65.],
[ 141.],
[  55.],
[ 134.],
[  42.],
[ 111.],
[  98.],
[ 164.],
[  48.],
[  96.],
[  90.],
[ 162.],
[ 150.],
[ 279.],
[  92.],
[  83.],
[ 128.],
[ 102.],
[ 302.],
[ 198.],
[  95.],
[  53.],
[ 134.]
```

```
[ 134.],
[ 144.],
[ 232.],
[  81.],
[ 104.],
[  59.],
[ 246.],
[ 297.],
[ 258.],
[ 229.],
[ 275.],
[ 281.],
[ 179.],
[ 200.],
[ 200.],
[ 173.],
[ 180.],
[  84.],
[ 121.],
[ 161.],
[  99.],
[ 109.],
[ 115.],
[ 268.],
[ 274.],
[ 158.],
[ 107.],
[  83.],
[ 103.],
[ 272.],
[  85.],
[ 280.],
[ 336.],
[ 281.],
[ 118.],
[ 317.],
[ 235.],
[  60.],
[ 174.],
[ 259.],
[ 178.],
[ 128.],
[  96.],
[ 126.],
[ 288.],
[  88.],
[ 292.],
[  71.],
[ 197.],
[ 186.],
[  25.],
[  84.],
[  96.],
[ 195.],
[  53.],
[ 217.],
[ 172.],
[ 131.],
[ 214.],
[  59.],
[  70.],
[ 220.],
```

```
[ 268.],
[ 152.],
[  47.],
[  74.],
[ 295.],
[ 101.],
[ 151.],
[ 127.],
[ 237.],
[ 225.],
[  81.],
[ 151.],
[ 107.],
[  64.],
[ 138.],
[ 185.],
[ 265.],
[ 101.],
[ 137.],
[ 143.],
[ 141.],
[  79.],
[ 292.],
[ 178.],
[  91.],
[ 116.],
[  86.],
[ 122.],
[  72.],
[ 129.],
[ 142.],
[  90.],
[ 158.],
[  39.],
[ 196.],
[ 222.],
[ 277.],
[  99.],
[ 196.],
[ 202.],
[ 155.],
[  77.],
[ 191.],
[  70.],
[  73.],
[  49.],
[  65.],
[ 263.],
[ 248.],
[ 296.],
[ 214.],
[ 185.],
[  78.],
[  93.],
[ 252.],
[ 150.],
[  77.],
[ 208.],
[  77.],
[ 108.],
[ 160.],
[  53.],
```

```
[ [illegible].],
[ 220.],
[ 154.],
[ 259.],
[  90.],
[ 246.],
[ 124.],
[  67.],
[  72.],
[ 257.],
[ 262.],
[ 275.],
[ 177.],
[  71.],
[  47.],
[ 187.],
[ 125.],
[  78.],
[  51.],
[ 258.],
[ 215.],
[ 303.],
[ 243.],
[  91.],
[ 150.],
[ 310.],
[ 153.],
[ 346.],
[  63.],
[  89.],
[  50.],
[  39.],
[ 103.],
[ 308.],
[ 116.],
[ 145.],
[  74.],
[  45.],
[ 115.],
[ 264.],
[  87.],
[ 202.],
[ 127.],
[ 182.],
[ 241.],
[  66.],
[  94.],
[ 283.],
[  64.],
[ 102.],
[ 200.],
[ 265.],
[  94.],
[ 230.],
[ 181.],
[ 156.],
[ 233.],
[  60.],
[ 219.],
[  80.],
[  68.],
[ 332.],
```

```
[ 248.],
[  84.],
[ 200.],
[  55.],
[  85.],
[  89.],
[  31.],
[ 129.],
[  83.],
[ 275.],
[  65.],
[ 198.],
[ 236.],
[ 253.],
[ 124.],
[  44.],
[ 172.],
[ 114.],
[ 142.],
[ 109.],
[ 180.],
[ 144.],
[ 163.],
[ 147.],
[  97.],
[ 220.],
[ 190.],
[ 109.],
[ 191.],
[ 122.],
[ 230.],
[ 242.],
[ 248.],
[ 249.],
[ 192.],
[ 131.],
[ 237.],
[  78.],
[ 135.],
[ 244.],
[ 199.],
[ 270.],
[ 164.],
[  72.],
[  96.],
[ 306.],
[  91.],
[ 214.],
[  95.],
[ 216.],
[ 263.],
[ 178.],
[ 113.],
[ 200.],
[ 139.],
[ 139.],
[  88.],
[ 148.],
[  88.],
[ 243.],
[  71.],
[  77.],
```

```
[ 109.],
[ 272.],
[  60.],
[  54.],
[ 221.],
[  90.],
[ 311.],
[ 281.],
[ 182.],
[ 321.],
[  58.],
[ 262.],
[ 206.],
[ 233.],
[ 242.],
[ 123.],
[ 167.],
[  63.],
[ 197.],
[  71.],
[ 168.],
[ 140.],
[ 217.],
[ 121.],
[ 235.],
[ 245.],
[  40.],
[  52.],
[ 104.],
[ 132.],
[  88.],
[  69.],
[ 219.],
[  72.],
[ 201.],
[ 110.],
[  51.],
[ 277.],
[  63.],
[ 118.],
[  69.],
[ 273.],
[ 258.],
[  43.],
[ 198.],
[ 242.],
[ 232.],
[ 175.],
[  93.],
[ 168.],
[ 275.],
[ 293.],
[ 281.],
[  72.],
[ 140.],
[ 189.],
[ 181.],
[ 209.],
[ 136.],
[ 261.],
[ 113.],
```

```
[ 131.],
[ 174.],
[ 257.],
[  55.],
[  84.],
[  42.],
[ 146.],
[ 212.]])
```

Example 2: Implement three different algorithms to train linear regression model

Algorithm 1: Linear Regression - Direct Solve (no regularization)

Preliminaries

1. The builtin function `solve(A, b)` computes the least squares solution for system of linear equations
$$Ax = b$$
for the vector x such that
$$||\ Ax - b\ ||$$
is minimized. It is important to note that this function can operate only on small-to-medium sized input matrix that can fit in the driver memory. See the [DML language reference (http://apache.github.io/systemml/dml-language-reference.html)](http://apache.github.io/systemml/dml-language-reference.html) for more details.
2. Linear regression model assumes that relationship between input explanatory (feature) variables X and numerical response variable y is linear. The goal is to estimate regression coefficient w (and residual variable) such that
$$y = \text{Normal}(Xw, \sigma^2)$$
$$\text{Cost function, } J(w) = \frac{1}{2}(Xw - y)^2$$

Differentiating with respect to w,

$$\begin{aligned} dw &= \frac{\partial}{\partial w}\frac{1}{2}(Xw - y)^2 \\ &= \frac{1}{2}2X^T(Xw - y) \\ &= (X^TX)w - X^Ty \end{aligned}$$

Setting the gradient

To find minima, we set the derivative with respect to w to zero,

$$\begin{aligned} (X^TX)w - (X^Ty) &= 0 \\ w &= (X^TX)^{-1}(X^Ty) \\ \text{Let } \ A &= X^TX \\ \text{and } \ b &= X^Ty \\ \text{Therefore, } w &= solve(A, b) \end{aligned}$$

In [12]:
```
script = """
```

```
In [12]: script =
    # add constant feature to X to model intercept
    ones = matrix(1, rows=nrow(X), cols=1)
    X = cbind(X, ones)
    A = t(X) %*% X
    b = t(X) %*% y
    w = solve(A, b)
    bias = as.scalar(w[nrow(w),1])
    w = w[1:nrow(w)-1,]
"""
```

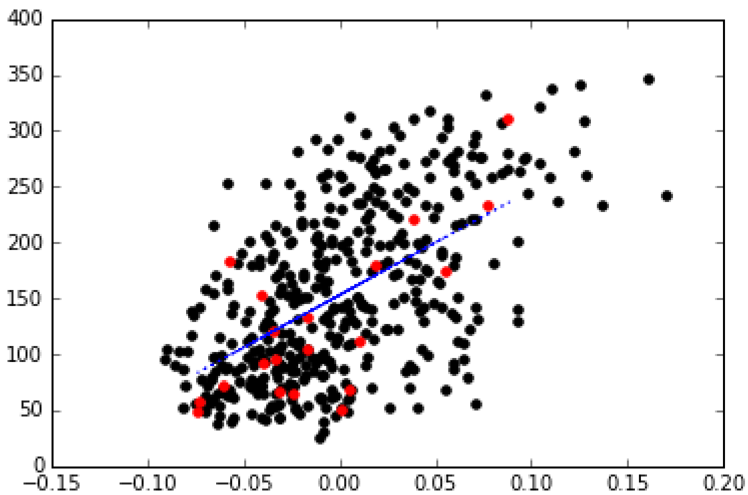
```
In [13]: prog = dml(script).input(X=diabetes_X_train, y=diabetes_y_train).output
('w', 'bias')
w, bias = ml.execute(prog).get('w','bias')
w = w.toNumPy()
```

```
SystemML Statistics:
Total execution time:            0.028 sec.
Number of executed Spark inst:   2.
```

```
In [14]: plt.scatter(diabetes_X_train, diabetes_y_train,  color='black')
plt.scatter(diabetes_X_test, diabetes_y_test,  color='red')

plt.plot(diabetes_X_test, (w*diabetes_X_test)+bias, color='blue', lines
tyle ='dotted')
```

```
Out[14]: [<matplotlib.lines.Line2D at 0x7f6348328d10>]
```

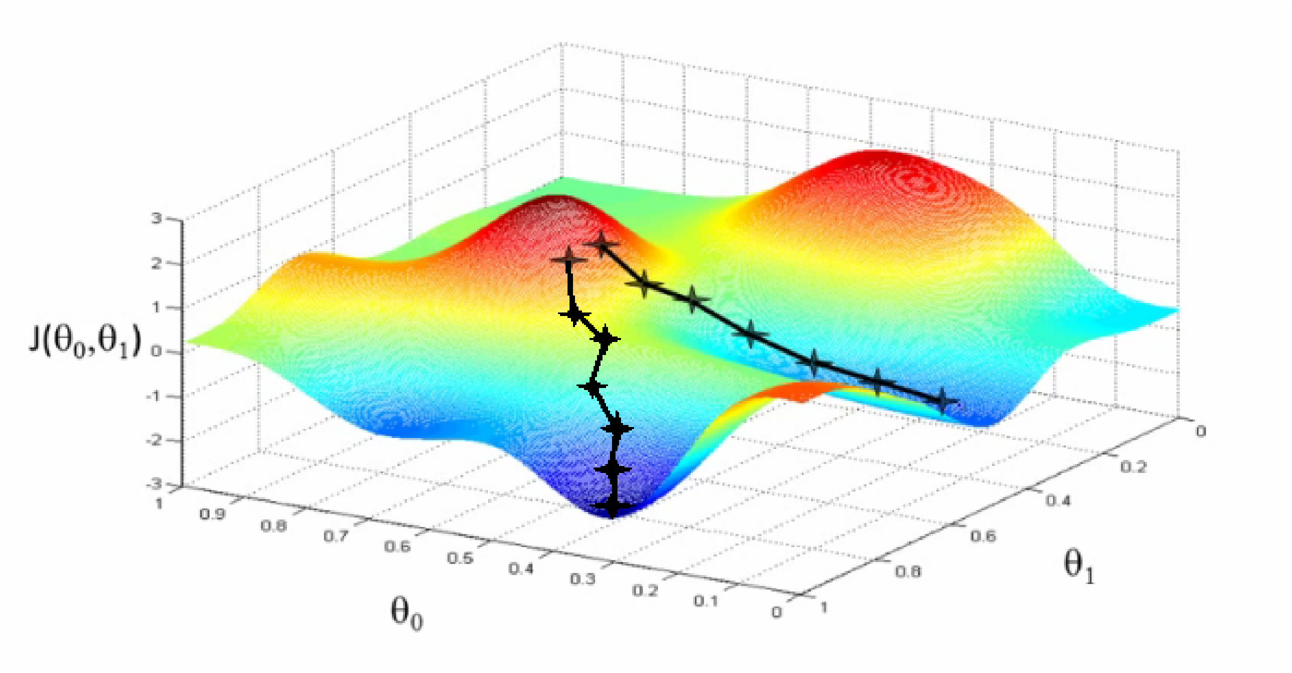


Algorithm 2: Linear Regression - Batch Gradient Descent (no regularization)

Algorithm

```
Step 1: Start with an initial point
while(not converged) {
  Step 2: Compute gradient dw.
  Step 3: Compute stepsize alpha.
  Step 4: Update: w_new = w_old - alpha*dw
}
```

J



Gradient formula

$$dw = r = (X^T X)w - (X^T y)$$

Step size formula

We perform a line search to choose the step size `alpha` to minimize the cost function J(w). From basic calculus, `alpha` minimizes the function J(w) when the directional derivative with respect to `alpha` is zero.

$$alpha = \frac{r^T r}{r^T X^T X r}$$

In [15]:

```
script = """
    # add constant feature to X to model intercepts
    ones = matrix(1, rows=nrow(X), cols=1)
    X = cbind(X, ones)
    max_iter = 100
    w = matrix(0, rows=ncol(X), cols=1)
    for(i in 1:max_iter){
        XtX = t(X) %*% X
        dw = XtX %*%w - t(X) %*% y
        alpha = (t(dw) %*% dw) / (t(dw) %*% XtX %*% dw)
        w = w - dw*alpha
    }
    bias = as.scalar(w[nrow(w),1])
    w = w[1:nrow(w)-1,]
"""
```

In [16]:

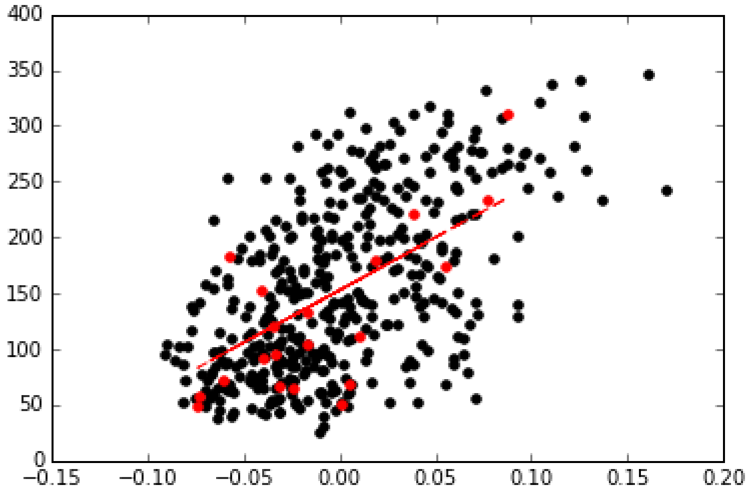
```
prog = dml(script).input(X=diabetes_X_train, y=diabetes_y_train).output
('w').output('bias')
w, bias = ml.execute(prog).get('w', 'bias')
w = w.toNumPy()
```

```
SystemML Statistics:
Total execution time:        0.081 sec.
Number of executed Spark inst:   2.
```

In [17]:
```
plt.scatter(diabetes_X_train, diabetes_y_train,  color='black')
plt.scatter(diabetes_X_test, diabetes_y_test,  color='red')

plt.plot(diabetes_X_test, (w*diabetes_X_test)+bias, color='red', linest
yle ='dashed')
```

Out[17]:
```
[<matplotlib.lines.Line2D at 0x7f6348296710>]
```



Algorithm 3: Linear Regression - Conjugate Gradient (no regularization)

Problem with gradient descent: Takes very similar directions many times

Solution: Enforce conjugacy

```
Step 1: Start with an initial point
while(not converged) {
   Step 2: Compute gradient dw.
   Step 3: Compute stepsize alpha.
   Step 4: Compute next direction p by enforcing conjugacy with previous direction.
   Step 4: Update: w_new = w_old + alpha*p
}
```

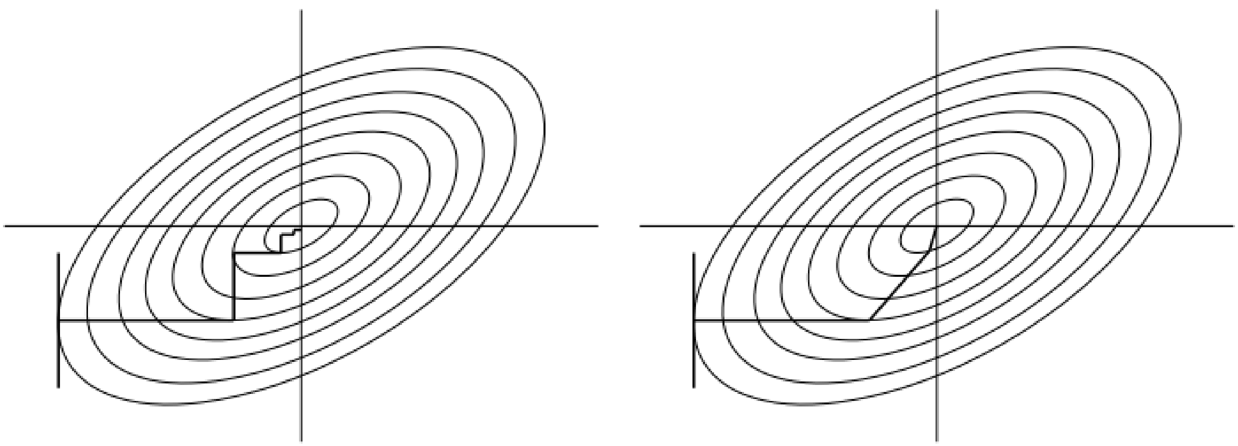


Figure 6.14: Steepest descent vs. conjugate gradient.

In [18]:

```
script = """
    # add constant feature to X to model intercepts
    X = cbind(X, matrix(1, rows=nrow(X), cols=1))
    m = ncol(X); i = 1;
    max_iter = 20;
    w = matrix (0, rows = m, cols = 1); # initialize weights to 0
    dw = - t(X) %*% y; p = - dw;        # dw = (X'X)w - (X'y)
    norm_r2 = sum (dw ^ 2);
    for(i in 1:max_iter) {
        q = t(X) %*% (X %*% p)
        alpha = norm_r2 / sum (p * q);  # Minimizes f(w - alpha*r)
        w = w + alpha * p;              # update weights
        dw = dw + alpha * q;
        old_norm_r2 = norm_r2; norm_r2 = sum (dw ^ 2);
        p = -dw + (norm_r2 / old_norm_r2) * p; # next direction - conju
gacy to previous direction
        i = i + 1;
    }
    bias = as.scalar(w[nrow(w),1])
    w = w[1:nrow(w)-1,]
"""
```

In [19]:

```
prog = dml(script).input(X=diabetes_X_train, y=diabetes_y_train).output
('w').output('bias')
w, bias = ml.execute(prog).get('w','bias')
w = w.toNumPy()
```

```
SystemML Statistics:
Total execution time:           0.007 sec.
Number of executed Spark inst:  2.
```

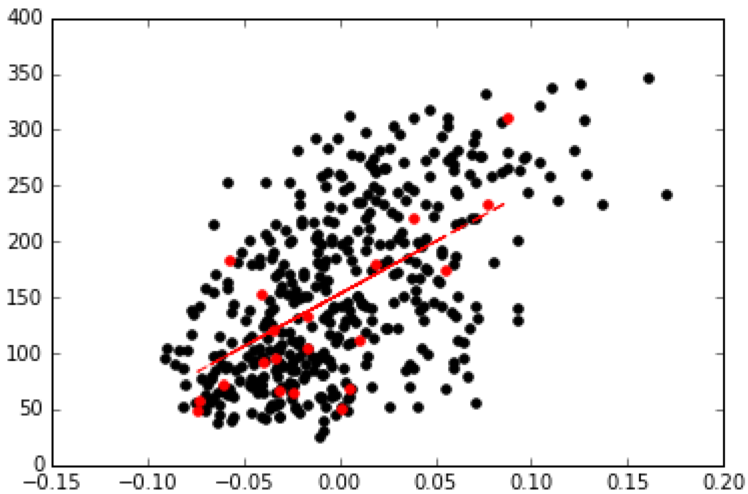
In [20]:

```
plt.scatter(diabetes_X_train, diabetes_y_train,  color='black')
plt.scatter(diabetes_X_test, diabetes_y_test,  color='red')

plt.plot(diabetes_X_test, (w*diabetes_X_test)+bias, color='red', linest
yle ='dashed')
```

Out[20]:

```
[<matplotlib.lines.Line2D at 0x7f634816a550>]
```



Example 3: Invoke existing SystemML algorithm

script LinearRegDS.dml using MLContext API

In [21]:

```
from systemml import dmlFromResource
prog = dmlFromResource('scripts/algorithms/LinearRegDS.dml').input(X=di
abetes_X_train, y=diabetes_y_train).input('$icpt',1.0).output('beta_ou
t')
w = ml.execute(prog).get('beta_out')
w = w.toNumPy()
bias=w[1]
```

```
BEGIN LINEAR REGRESSION SCRIPT
Reading X and Y...
Calling the Direct Solver...
Computing the statistics...
AVG_TOT_Y,153.36255924170615
STDEV_TOT_Y,77.21853383600028
AVG_RES_Y,3.633533705616816E-14
STDEV_RES_Y,63.038506337610244
DISPERSION,3973.853281276927
R2,0.3351312506863875
ADJUSTED_R2,0.33354822985468835
R2_NOBIAS,0.3351312506863875
ADJUSTED_R2_NOBIAS,0.33354822985468835
Writing the output matrix...
END LINEAR REGRESSION SCRIPT
SystemML Statistics:
Total execution time:           0.008 sec.
Number of executed Spark inst:  2.
```

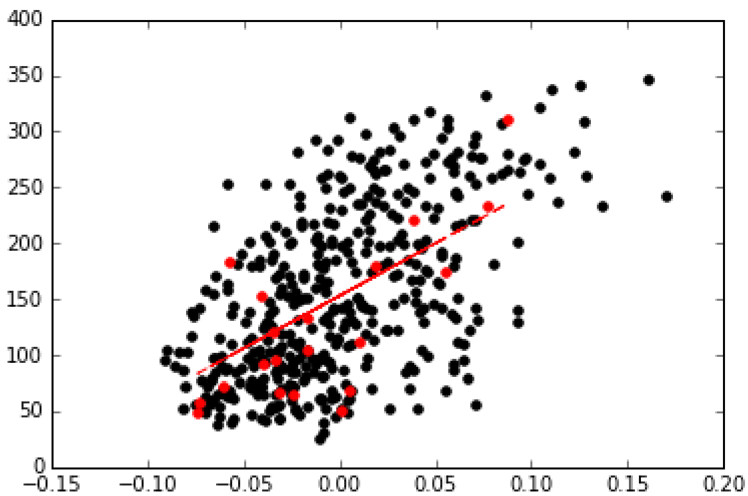
In [22]:

```
plt.scatter(diabetes_X_train, diabetes_y_train,  color='black')
plt.scatter(diabetes_X_test, diabetes_y_test,  color='red')

plt.plot(diabetes_X_test, (w[0]*diabetes_X_test)+bias, color='red', lin
estyle ='dashed')
```

Out[22]:

```
[<matplotlib.lines.Line2D at 0x7f63480b9810>]
```



Example 4: Invoke existing SystemML algorithm using scikit-learn/SparkML pipeline like API

mllearn API allows a Python programmer to invoke SystemML's algorithms using scikit-learn like API as well as Spark's MLPipeline API.

In [23]:
```
from pyspark.sql import SQLContext
from systemml.mllearn import LinearRegression
sqlCtx = SQLContext(sc)
```

In [24]:
```
regr = LinearRegression(sqlCtx)
# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)
```

```
BEGIN LINEAR REGRESSION SCRIPT
Reading X and Y...
Running the CG algorithm...
||r|| initial value = 64725.64237405237,  target value = 0.064725642374
05237
Iteration 1:  ||r|| / ||r init|| = 0.013822097283108787
Iteration 2:  ||r|| / ||r init|| = 5.369915930350396E-14
The CG algorithm is done.
Computing the statistics...
AVG_TOT_Y,153.36255924170615
STDEV_TOT_Y,77.21853383600028
AVG_RES_Y,-8.227243004822623E-12
STDEV_RES_Y,63.03850633759284
DISPERSION,3973.853281274733
R2,0.33513125068675453
ADJUSTED_R2,0.3335482298550564
R2_NOBIAS,0.33513125068675453
ADJUSTED_R2_NOBIAS,0.3335482298550564
Writing the output matrix...
END LINEAR REGRESSION SCRIPT
SystemML Statistics:
Total execution time:           0.003 sec.
Number of executed Spark inst:  2.
```

Out[24]: `lr`

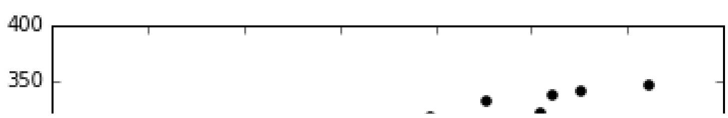
In [25]:
```
predictions = regr.predict(diabetes_X_test)
```

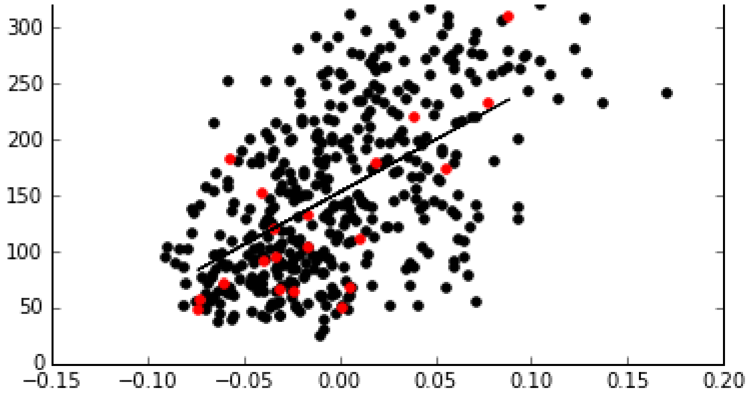
```
SystemML Statistics:
Total execution time:           0.000 sec.
Number of executed Spark inst:  1.
```

In [26]:
```
# Use the trained model to perform prediction
%matplotlib inline
plt.scatter(diabetes_X_train, diabetes_y_train,  color='black')
plt.scatter(diabetes_X_test, diabetes_y_test,  color='red')

plt.plot(diabetes_X_test, predictions, color='black')
```

Out[26]: `[<matplotlib.lines.Line2D at 0x7f632e6e5b90>]`





(Optional) Install OpenBLAS

!wget https://github.com/xianyi/OpenBLAS/archive/v0.2.20.tar.gz !tar -xzf v0.2.20.tar.gz !cd OpenBLAS-0.2.20/ && make clean !cd OpenBLAS-0.2.20/ && make USE_OPENMP=1

Example 5: Invoking a Keras model with SystemML

See SystemML's deep learning documentation (http://apache.github.io/systemml/deep-learning) for more detail.

In [ ]:
```
from mlxtend.data import mnist_data
import numpy as np
from sklearn.utils import shuffle
# Download the MNIST dataset
X, y = mnist_data()
X, y = shuffle(X, y)
# Split the data into training and test
n_samples = len(X)
X_train = X[:int(.9 * n_samples)]
y_train = y[:int(.9 * n_samples)]
X_test = X[int(.9 * n_samples):]
y_test = y[int(.9 * n_samples):]
from keras.models import Sequential
from keras.layers import Input, Dense, Conv2D, MaxPooling2D, Dropout,Fl
atten
from keras import backend as K
from keras.models import Model
input_shape = (1,28,28) if K.image_data_format() == 'channels_first' el
se (28,28, 1)
keras_model = Sequential()
keras_model.add(Conv2D(32, kernel_size=(5, 5), activation='relu', input
_shape=input_shape, padding='same'))
keras_model.add(MaxPooling2D(pool_size=(2, 2)))
keras_model.add(Conv2D(64, (5, 5), activation='relu', padding='same'))
keras_model.add(MaxPooling2D(pool_size=(2, 2)))
keras_model.add(Flatten())
keras_model.add(Dense(512, activation='relu'))
keras_model.add(Dropout(0.5))
keras_model.add(Dense(10, activation='softmax'))

# Scale the input features
scale = 0.00390625
X_train = X_train*scale
```

```
X_train = X_train*scale
X_test = X_test*scale

from systemml.mllearn import Keras2DML
sysml_model = Keras2DML(spark, keras_model, input_shape=(1,28,28), weig
hts='weights_dir')
sysml_model.setConfigProperty('sysml.native.blas', 'openblas')
sysml_model.setConfigProperty('sysml.native.blas.directory', os.path.jo
in(os.getcwd(),'OpenBLAS-0.2.20/'))
# sysml_model.setGPU(True).setForceGPU(True)
sysml_model.summary()
sysml_model.fit(X_train, y_train)
```

```
Using TensorFlow backend.

Loading the model from weights_dir...
SystemML Statistics:
Total execution time:           0.000 sec.
Number of executed Spark inst:  0.


[Stage 9:=======================================>                  (2
+ 1) / 3]
+------------------+-------------+-------------+-----------+------
---+--------------------+------------------+--------------------+
|              Name|         Type|       Output|     Weight|     B
ias|                 Top|            Bottom|Memory* (train/test)|
+------------------+-------------+-------------+-----------+------
---+--------------------+------------------+--------------------+
|     conv2d_1_input|         Data| (, 1, 28, 28)|           |
   |conv2d_1_input,label|                  |                 1/0|
|           conv2d_1|    Convolution|(, 32, 28, 28)|   [32 X 25]| [32 X
1]|            conv2d_1|     conv2d_1_input|                25/13|
```