Untitled

October 20, 2019

[1]:
```
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()


def genTestData(npoints, max):
    rng = np.random.RandomState(1)
    return max * rng.rand(npoints)


def genPointsFromGroundTruthEquationWithNoise(npoints, maxX, noise=0):
    xPointsList = genTestData(npoints, maxX)
    print(npoints, max)

    rng = np.random.RandomState(9)
    points = []
    for x in xPointsList:
        y = (rng.rand() * noise) + np.sin(x) + x
        points.append((x,y))
    return points
```

[ ]:
```
```

[4]:
```
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler


def genTesting(i):
    trainingPoints = genPointsFromGroundTruthEquationWithNoise(i, 30, noise=4.0)
    x = [x for x, y in trainingPoints]
    y = [y for x, y in trainingPoints]
    plt.scatter(x, y, s=15, color='black')
```

```
    y_all = np.array(y)
    x_all = np.array(x)

    size = len(x)

    train_size = int(size*0.8)
    test_size = int(size*0.2)

    x_train = x_all[:train_size]
    x_test = x_all[test_size:]

    y_train = y_all[:train_size]
    y_test = y_all[test_size:]
    return x_train, x_test, y_train, y_test
```

1000 points with 3 features seems to have the best MSE with only changing the number of points and the number of features as 1.658 against the test, and 4.408 against the ground data.

Now testing regularization

[16]:
```
mse_values = []
mse_data = []
for j in ["regular", "lasso", "ridge"]:
    for k in [True, False]:
        for i in range(5):
            for ii in [500,1000,5000,10000]:
                x_train, x_test, y_train, y_test = genTesting(ii)
            if not k:
                if j=="regular":
                    poly_model = make_pipeline(PolynomialFeatures(i),␣
 ↪LinearRegression())
                if j=="lasso":
                    a = 1 * 10**(i)
                    poly_model = make_pipeline(PolynomialFeatures(i),␣
 ↪Lasso(alpha=a))
                if j=="ridge":
                    a = 1 * 10**(i)
                    poly_model = make_pipeline(PolynomialFeatures(i),␣
 ↪Ridge(alpha=a))
            elif k:
                if j=="regular":
                    poly_model = make_pipeline(StandardScaler(),␣
 ↪PolynomialFeatures(i), LinearRegression())
                if j=="lasso":
                    a = 1 * 10**(i)
                    poly_model = make_pipeline(StandardScaler(),␣
 ↪PolynomialFeatures(i), Lasso(alpha=a))
                if j=="ridge":
                    a = 1 * 10**(i)
```

```
                    poly_model = make_pipeline(StandardScaler(),␣
↪PolynomialFeatures(i), Ridge(alpha=a))
            poly_model.fit(x_train[:, np.newaxis], y_train)
            xfit = np.linspace(0,30,100)
            yfit = poly_model.predict(xfit[:, np.newaxis])
            y_predict = poly_model.predict(x_test[:,np.newaxis])
            print("Features = ", i, ", regularization = ", j, ", standarization␣
↪= ", k, ", MSE against test w/ noise", mean_squared_error(y_predict, y_test))
            y_ground = [np.sin(x)+x for x in x_test]
            print("MSE against ground", mean_squared_error(y_predict, y_ground))
            mse_values.append(mean_squared_error(y_predict, y_ground))
            mse_data.append((i, j, k, ii))
```

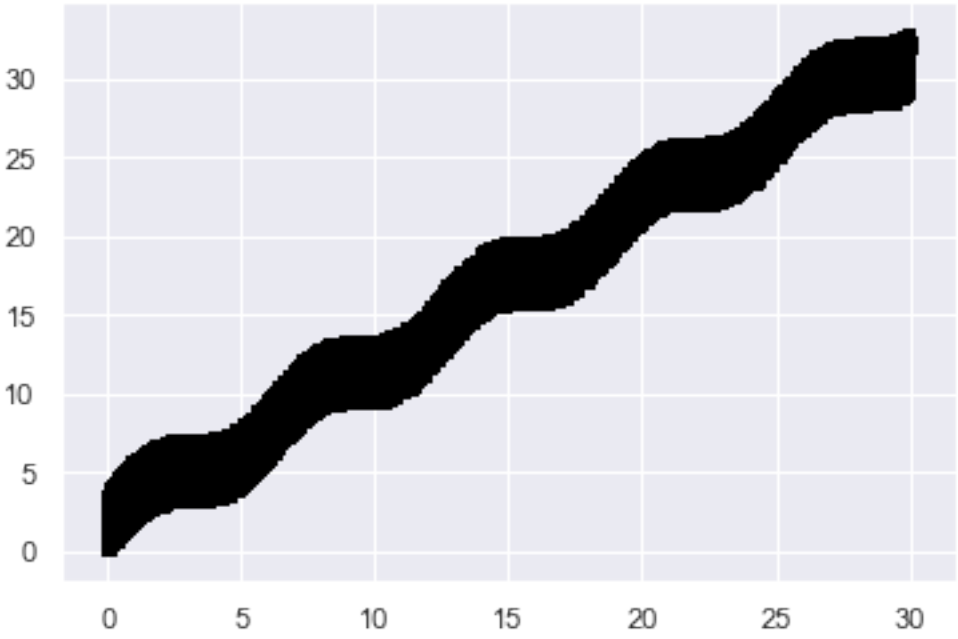
```
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  0 , regularization =  regular , standarization =  True , MSE against
test w/ noise 74.45296580258
MSE against ground 77.6562409311669
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  1 , regularization =  regular , standarization =  True , MSE against
test w/ noise 1.796221518110032
MSE against ground 4.4720542297511345
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  2 , regularization =  regular , standarization =  True , MSE against
test w/ noise 1.7922260492126456
MSE against ground 4.468580597329669
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  3 , regularization =  regular , standarization =  True , MSE against
test w/ noise 1.7801102145058576
MSE against ground 4.450201319508428
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  4 , regularization =  regular , standarization =  True , MSE against
test w/ noise 1.7800712234742455
```

```
MSE against ground 4.450261521850174
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  0 , regularization =  regular , standarization =  False , MSE
against test w/ noise 74.45296580258
MSE against ground 77.6562409311669
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  1 , regularization =  regular , standarization =  False , MSE
against test w/ noise 1.7962215181100327
MSE against ground 4.4720542297509835
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  2 , regularization =  regular , standarization =  False , MSE
against test w/ noise 1.7922260492126465
MSE against ground 4.468580597329505
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  3 , regularization =  regular , standarization =  False , MSE
against test w/ noise 1.7801102145058636
MSE against ground 4.450201319508348
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  4 , regularization =  regular , standarization =  False , MSE
against test w/ noise 1.7800712234740481
MSE against ground 4.450261521849149
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  0 , regularization =  lasso , standarization =  True , MSE against
test w/ noise 74.45296580258
MSE against ground 77.6562409311669
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  1 , regularization =  lasso , standarization =  True , MSE against
```

```
test w/ noise 74.45296580258
MSE against ground 77.6562409311669
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  2 , regularization =  lasso , standarization =  True , MSE against
test w/ noise 74.45296580258
MSE against ground 77.6562409311669
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  3 , regularization =  lasso , standarization =  True , MSE against
test w/ noise 74.45296580258
MSE against ground 77.6562409311669
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  4 , regularization =  lasso , standarization =  True , MSE against
test w/ noise 74.45296580258
MSE against ground 77.6562409311669
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  0 , regularization =  lasso , standarization =  False , MSE against
test w/ noise 74.45296580258
MSE against ground 77.6562409311669
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  1 , regularization =  lasso , standarization =  False , MSE against
test w/ noise 3.1402943467462343
MSE against ground 5.887758437682194
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  2 , regularization =  lasso , standarization =  False , MSE against
test w/ noise 6.255867680935083
MSE against ground 9.004321781164203
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
```

```
Features =  3 , regularization =  lasso , standarization =  False , MSE against
test w/ noise 13.323801702917212
MSE against ground 16.144546778709316
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  4 , regularization =  lasso , standarization =  False , MSE against
test w/ noise 20.054801212843728
MSE against ground 22.947317706209606
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  0 , regularization =  ridge , standarization =  True , MSE against
test w/ noise 74.45296580258
MSE against ground 77.6562409311669
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  1 , regularization =  ridge , standarization =  True , MSE against
test w/ noise 1.7963771239858404
MSE against ground 4.472868315548224
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  2 , regularization =  ridge , standarization =  True , MSE against
test w/ noise 1.8037946344708866
MSE against ground 4.486645439340898
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  3 , regularization =  ridge , standarization =  True , MSE against
test w/ noise 3.7944257311923937
MSE against ground 6.567627905922499
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  4 , regularization =  ridge , standarization =  True , MSE against
test w/ noise 11.518945592038499
MSE against ground 14.435373324294854
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
```

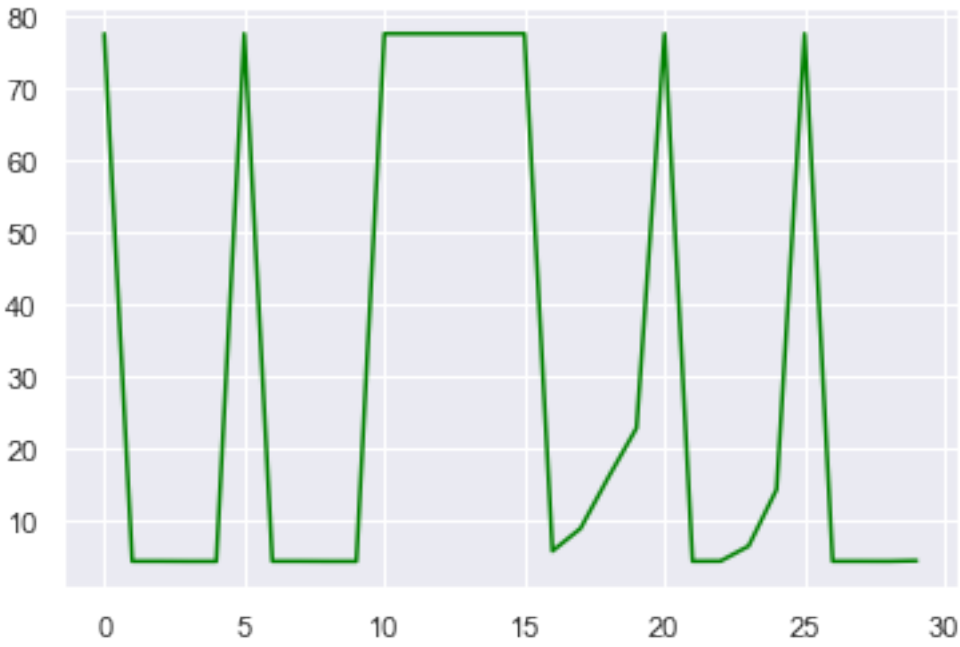
```
10000 <built-in function max>
Features =  0 , regularization =  ridge , standarization =  False , MSE against
test w/ noise 74.45296580258
MSE against ground 77.6562409311669
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  1 , regularization =  ridge , standarization =  False , MSE against
test w/ noise 1.7962221098319082
MSE against ground 4.472063688905236
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  2 , regularization =  ridge , standarization =  False , MSE against
test w/ noise 1.7922033303889524
MSE against ground 4.468692326023027
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  3 , regularization =  ridge , standarization =  False , MSE against
test w/ noise 1.7930620130944503
MSE against ground 4.459540291731312
500 <built-in function max>
1000 <built-in function max>
5000 <built-in function max>
10000 <built-in function max>
Features =  4 , regularization =  ridge , standarization =  False , MSE against
test w/ noise 1.8812640790210913
MSE against ground 4.54091867405922
```



[17]:
```
list_vals = list(range(len(mse_values)))
plt.plot(list_vals, mse_values, color='green')
count = 0
for i, j, k, ii in mse_data:
    print("Features = ", i, ", Standardization = ", k, ", Regularization = ",␣
 ↪j, ", values = ", ii, ", mse against ground = ", mse_values[count])
    count += 1
print("Minimum mse = ", min(mse_values))
print("Minimum value is found at ", mse_data[np.argmin(mse_values)])
```

```
Features =  0 , Standardization =  True , Regularization =  regular , values =
10000 , mse against ground =  77.6562409311669
Features =  1 , Standardization =  True , Regularization =  regular , values =
10000 , mse against ground =  4.4720542297511345
Features =  2 , Standardization =  True , Regularization =  regular , values =
10000 , mse against ground =  4.468580597329669
Features =  3 , Standardization =  True , Regularization =  regular , values =
10000 , mse against ground =  4.450201319508428
Features =  4 , Standardization =  True , Regularization =  regular , values =
10000 , mse against ground =  4.450261521850174
Features =  0 , Standardization =  False , Regularization =  regular , values =
10000 , mse against ground =  77.6562409311669
Features =  1 , Standardization =  False , Regularization =  regular , values =
10000 , mse against ground =  4.4720542297509835
Features =  2 , Standardization =  False , Regularization =  regular , values =
10000 , mse against ground =  4.468580597329505
```

```
Features =  3 , Standardization =  False , Regularization =  regular , values =
10000 , mse against ground =  4.450201319508348
Features =  4 , Standardization =  False , Regularization =  regular , values =
10000 , mse against ground =  4.450261521849149
Features =  0 , Standardization =  True , Regularization =  lasso , values =
10000 , mse against ground =  77.6562409311669
Features =  1 , Standardization =  True , Regularization =  lasso , values =
10000 , mse against ground =  77.6562409311669
Features =  2 , Standardization =  True , Regularization =  lasso , values =
10000 , mse against ground =  77.6562409311669
Features =  3 , Standardization =  True , Regularization =  lasso , values =
10000 , mse against ground =  77.6562409311669
Features =  4 , Standardization =  True , Regularization =  lasso , values =
10000 , mse against ground =  77.6562409311669
Features =  0 , Standardization =  False , Regularization =  lasso , values =
10000 , mse against ground =  77.6562409311669
Features =  1 , Standardization =  False , Regularization =  lasso , values =
10000 , mse against ground =  5.887758437682194
Features =  2 , Standardization =  False , Regularization =  lasso , values =
10000 , mse against ground =  9.004321781164203
Features =  3 , Standardization =  False , Regularization =  lasso , values =
10000 , mse against ground =  16.144546778709316
Features =  4 , Standardization =  False , Regularization =  lasso , values =
10000 , mse against ground =  22.947317706209606
Features =  0 , Standardization =  True , Regularization =  ridge , values =
10000 , mse against ground =  77.6562409311669
Features =  1 , Standardization =  True , Regularization =  ridge , values =
10000 , mse against ground =  4.472868315548224
Features =  2 , Standardization =  True , Regularization =  ridge , values =
10000 , mse against ground =  4.486645439340898
Features =  3 , Standardization =  True , Regularization =  ridge , values =
10000 , mse against ground =  6.567627905922499
Features =  4 , Standardization =  True , Regularization =  ridge , values =
10000 , mse against ground =  14.435373324294854
Features =  0 , Standardization =  False , Regularization =  ridge , values =
10000 , mse against ground =  77.6562409311669
Features =  1 , Standardization =  False , Regularization =  ridge , values =
10000 , mse against ground =  4.472063688905236
Features =  2 , Standardization =  False , Regularization =  ridge , values =
10000 , mse against ground =  4.468692326023027
Features =  3 , Standardization =  False , Regularization =  ridge , values =
10000 , mse against ground =  4.459540291731312
Features =  4 , Standardization =  False , Regularization =  ridge , values =
10000 , mse against ground =  4.54091867405922
Minimum mse =  4.450201319508348
Minimum value is found at  (3, 'regular', False, 10000)
```



0.0.1 Conclusions

As we can see above, the lowest mse against the ground truth is best with 3 polynomial features, no lasso or ridge regression, no standardization, and 10000 values.

[ ]: