minstakvadratmetoden

October 15, 2024

1 Minstakvadrat-metoden med tillämpningar

1.1 Linjär regression

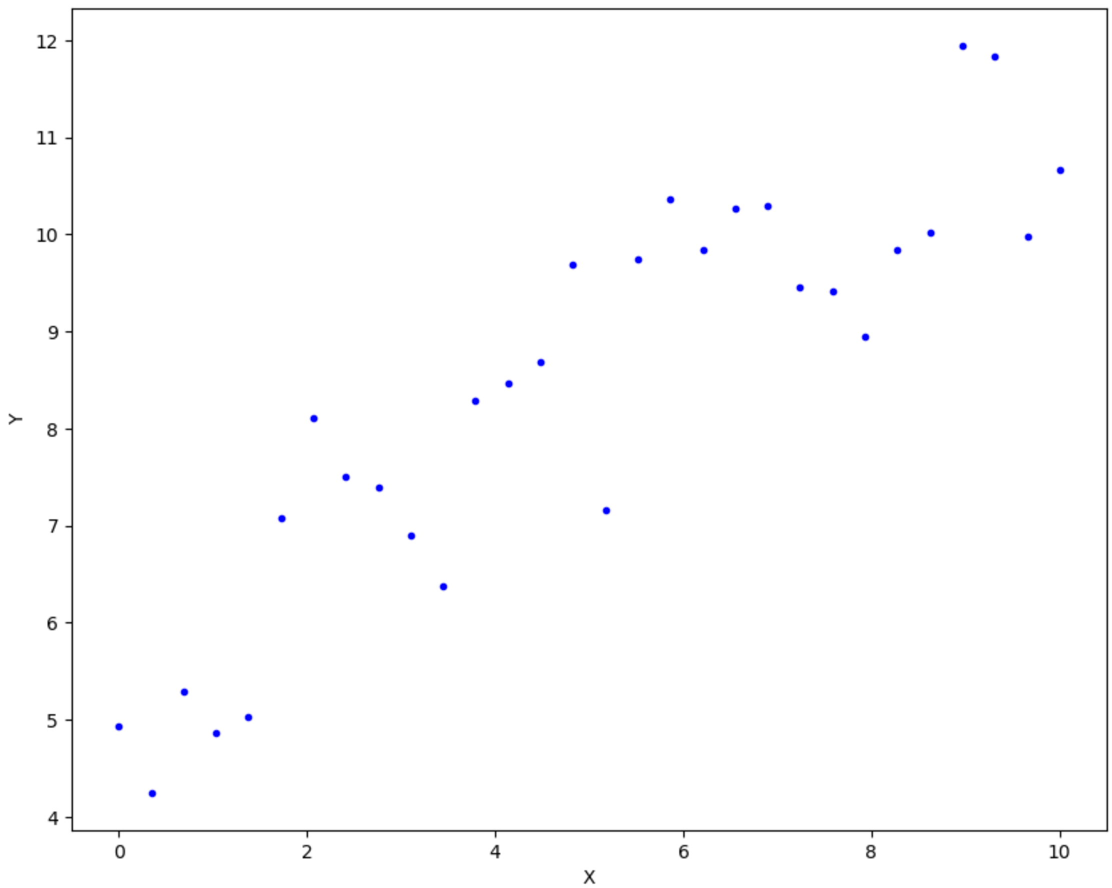
[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

- Vektor X, Y med x-koord och Y-koord
- Vi vill anpassa linje y = kx +m
- Bestämmer k, m genom att minstakvadrat-lösa AZ=B där Z^t=(k m)

[86]:
```
# generate X and Y
number_pts=30
true_m = 4
true_k = 0.6
spread = 3
X = np.linspace(0, 10, number_pts)
Y = true_m + true_k*X + spread*np.random.random(len(X))

plt.figure(figsize = (10,8))
plt.plot(X, Y, 'b.')

plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```



[87]:
```python
# assemble matrix A
A = np.vstack([X, np.ones(len(X))]).T
# turn Y into a column vector
B = Y[:, np.newaxis]
A[:10],B[:10]
```

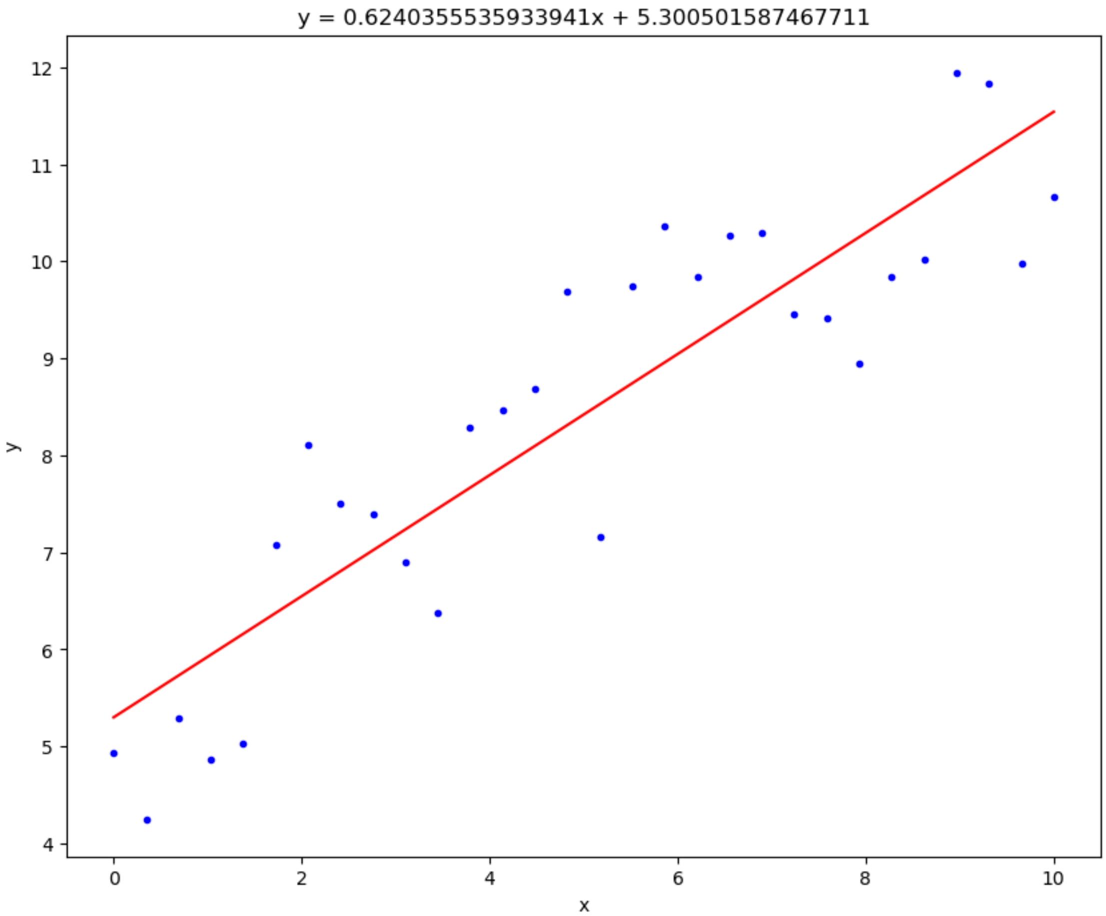
[87]:
```
(array([[0.        , 1.        ],
        [0.34482759, 1.        ],
        [0.68965517, 1.        ],
        [1.03448276, 1.        ],
        [1.37931034, 1.        ],
        [1.72413793, 1.        ],
        [2.06896552, 1.        ],
        [2.4137931 , 1.        ],
        [2.75862069, 1.        ],
        [3.10344828, 1.        ]]),
 array([[4.93044461],
        [4.25144944],
        [5.29417852],
```

```
           [4.86238453],
           [5.03111568],
           [7.0811378 ],
           [8.11478067],
           [7.50730365],
           [7.39201724],
           [6.90249417]]))
```

[88]:
```
# Löser normalekvationerna A^t*A*Z=A^t*B
# genom Z = (A^t*A)^{-1}*A^t*B
alpha = np.dot((np.dot(np.linalg.inv(np.dot(A.T,A)),A.T)),B)
print(alpha)
```

```
[[0.62403555]
 [5.30050159]]
```

[89]:
```
# plot the results
plt.figure(figsize = (10,8))
plt.plot(X, Y, 'b.')
plt.plot(X, alpha[0]*X + alpha[1], 'r')
plt.title(f'y = {alpha[0][0]}x + {alpha[1][0]}')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



1.2 Anpassning till polynom av okänd grad

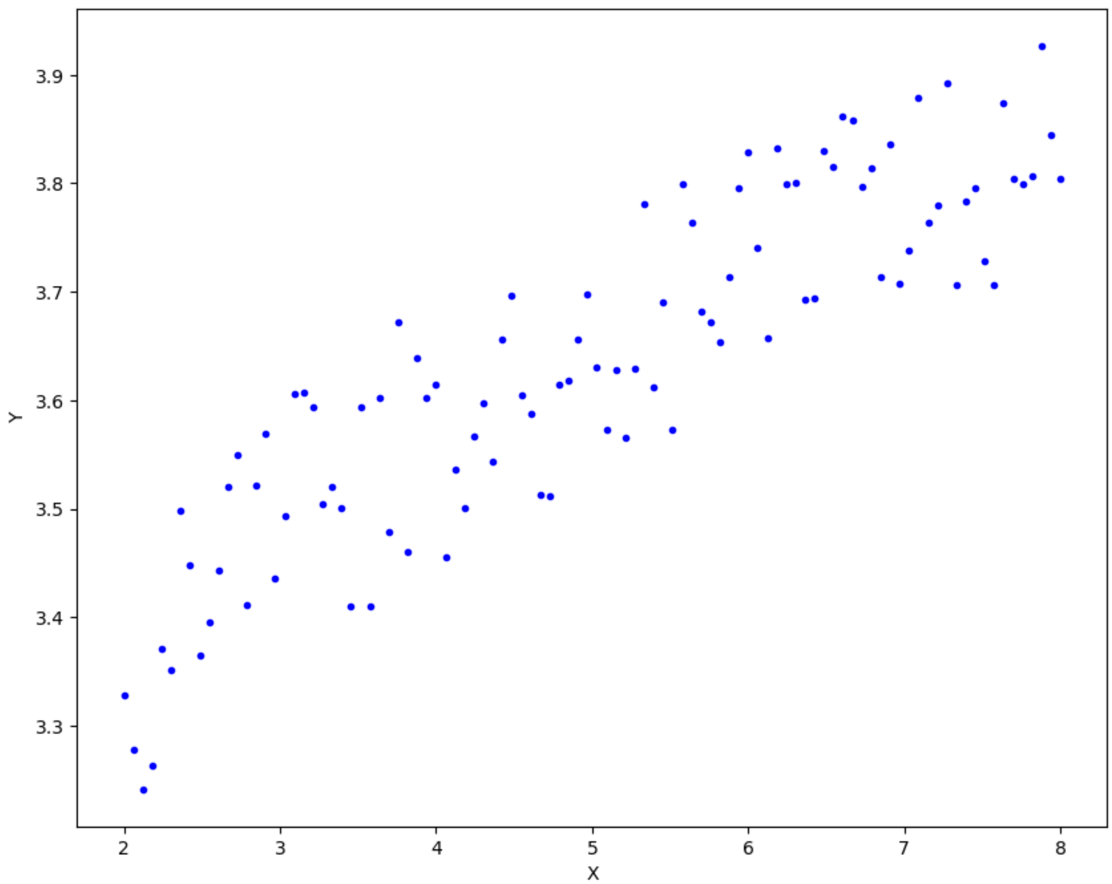
Om $y = cx^d$ approximativt, så är $\log(y) = \log(c) + d\log(x)$. Om vi hittar (m,k) som anpassar rät linje till $(\log(x_i), \log(y_i))$ så kommer alltså $c = \exp m$, $d = k$ vara bra approximation med polynom.

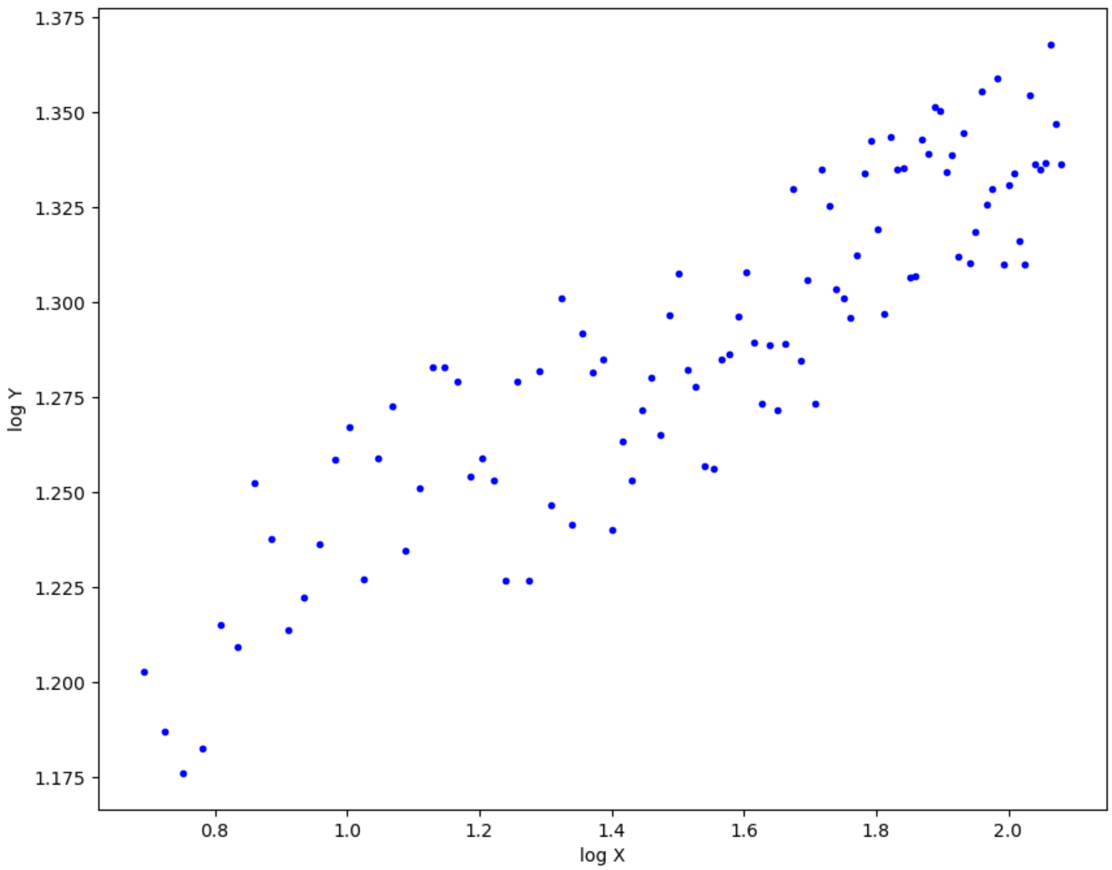
[99]:
```
# generate X and Y
true_c = 3
true_d= 0.1
number_pts=100
spread= 1/4
# no zeroes in X or Y, we are gonna log!
X = np.linspace(2, 8, number_pts)
Y = true_c*X**true_d +  spread*np.random.random(len(X))
lX = np.log(X)
lY = np.log(Y)

plt.figure(figsize = (10,8))
plt.plot(X, Y, 'b.')
```

```
plt.xlabel('X')
plt.ylabel('Y')
plt.show()

plt.figure(figsize = (10,8))
plt.plot(lX, lY, 'b.')
plt.xlabel('log X')
plt.ylabel('log Y')
plt.show()
```





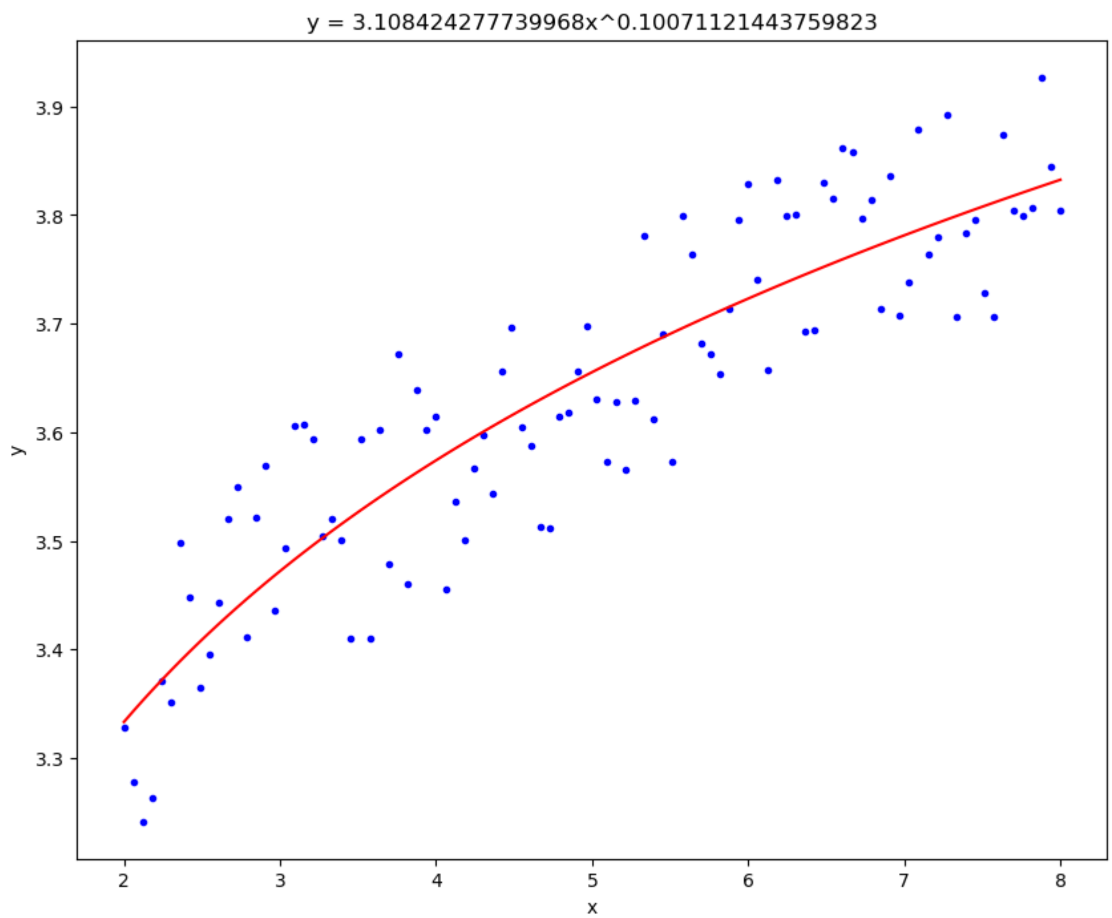
[100]:

```python
# assemble matrix A
A = np.vstack([lX, np.ones(len(lX))]).T
# turn Y into a column vector
B = lY[:, np.newaxis]
print(A[:10])
print(B[:10])
alpha = np.dot((np.dot(np.linalg.inv(np.dot(A.T,A)),A.T)),B)
print(alpha)
```

```
[[0.69314718 1.        ]
 [0.72300014 1.        ]
 [0.75198768 1.        ]
 [0.78015856 1.        ]
 [0.80755753 1.        ]
 [0.83422578 1.        ]
 [0.86020127 1.        ]
 [0.88551907 1.        ]
 [0.91021169 1.        ]
 [0.93430924 1.        ]]
[[1.2025139 ]
```

```
 [1.18701321]
 [1.17603062]
 [1.1825326 ]
 [1.21515103]
 [1.20923603]
 [1.25216835]
 [1.23763688]
 [1.21345189]
 [1.22224878]]
[[0.10071121]
 [1.13411593]]
```

[101]:
```python
# plot the results
import math
plt.figure(figsize = (10,8))
plt.plot(X, Y, 'b.')
plt.plot(X, math.exp(alpha[1][0])*X**alpha[0][0], 'r')
plt.title(f'y = {math.exp(alpha[1][0])}x^{alpha[0][0]}')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



[ ]:

```

```

[ ]:

```

```

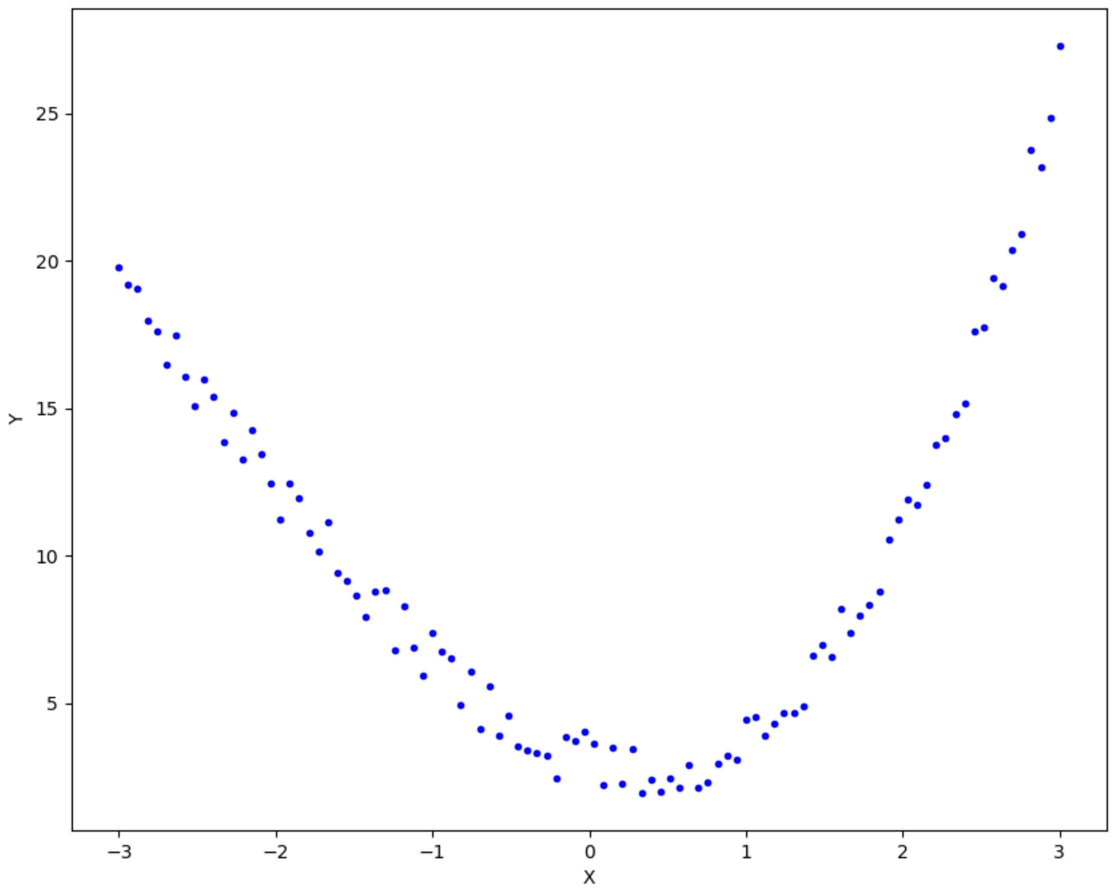
1.3 Anpassning till tredjegradspolynom

Vi anpassar $y = a_3x^3 + a_2x^2 + a_1x + a_0$

[102]:

```
# generate X and Y
a = [2.0, -1.5, 2.2, 0.3]
number_pts=100
spread=2
# no zeroes in X or Y, we are gonna log!
X = np.linspace(-3, 3, number_pts)
Y = a[0] + a[1]*X + a[2]*X**2 + a[3]*X**3 + spread*np.random.random(len(X))


plt.figure(figsize = (10,8))
plt.plot(X, Y, 'b.')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```




```
[103]: # assemble matrix A
       A = np.vstack([X**3, X**2, X, np.ones(len(X))]).T
       # turn Y into a column vector
       B = Y[:, np.newaxis]
       A[:10],B[:10]
```

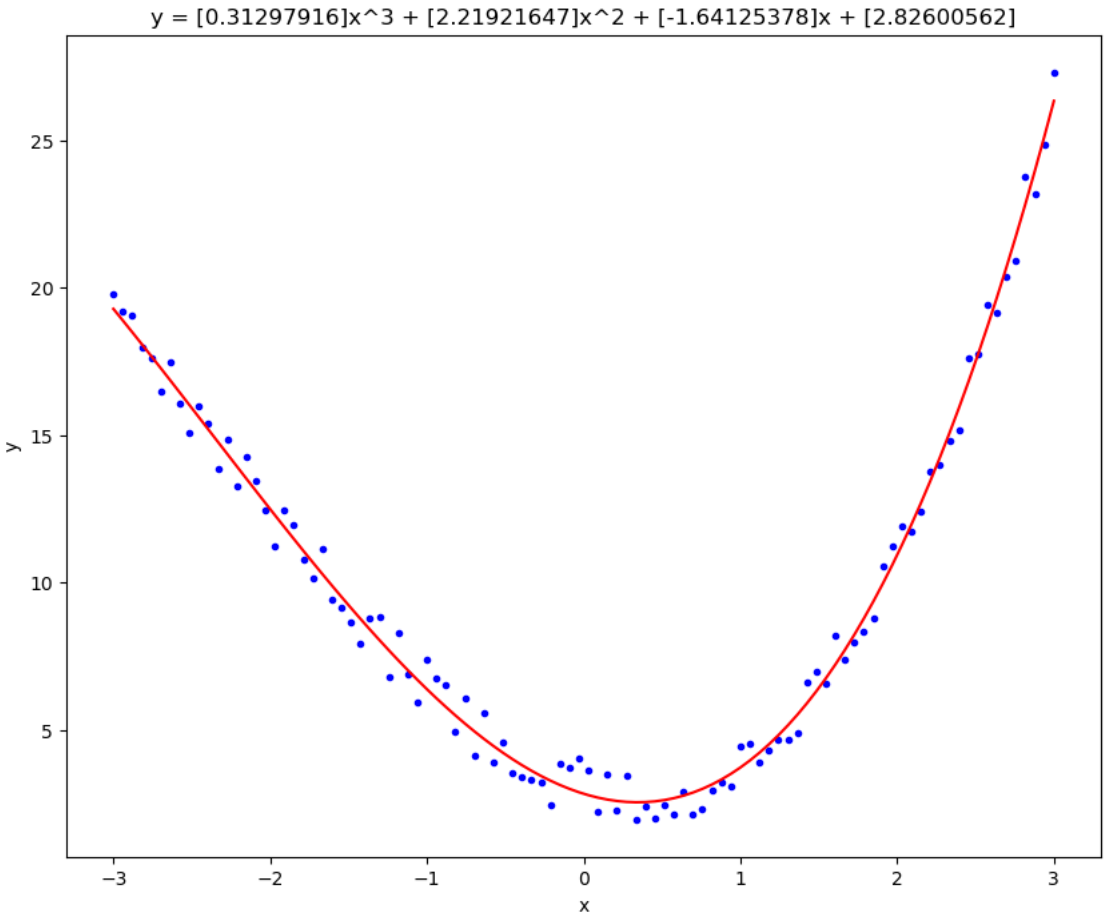
```
[103]: (array([[-27.        ,   9.        ,  -3.        ,   1.        ],
               [-25.3964716 ,   8.64003673,  -2.93939394,   1.        ],
               [-23.85772324,   8.28741965,  -2.87878788,   1.        ],
               [-22.38241923,   7.94214876,  -2.81818182,   1.        ],
               [-20.96922392,   7.60422406,  -2.75757576,   1.        ],
               [-19.61680163,   7.27364555,  -2.6969697 ,   1.        ],
               [-18.32381668,   6.95041322,  -2.63636364,   1.        ],
               [-17.08893341,   6.63452709,  -2.57575758,   1.        ],
               [-15.91081615,   6.32598714,  -2.51515152,   1.        ],
               [-14.78812923,   6.02479339,  -2.45454545,   1.        ]]),
        array([[19.75605733],
               [19.19063   ],
               [19.02841516],
```

```
        [17.94733341],
        [17.61303265],
        [16.4584552 ],
        [17.46855893],
        [16.06574002],
        [15.08340718],
        [15.95538607]]))
```

[104]:
```
# Löser normalekvationerna A^t*A*Z=A^t*B
# genom Z = (A^t*A)^{-1}*A^t*B
alpha = np.dot((np.dot(np.linalg.inv(np.dot(A.T,A)),A.T)),B)
print(alpha)
```

```
[[ 0.31297916]
 [ 2.21921647]
 [-1.64125378]
 [ 2.82600562]]
```

[105]:
```
# plot the results
plt.figure(figsize = (10,8))
plt.plot(X, Y, 'b.')
plt.plot(X, alpha[0]*X**3 + alpha[1]*X**2 + alpha[2]*X + alpha[3], 'r')
plt.xlabel('x')
plt.ylabel('y')
plt.title(f'y = {alpha[0]}x^3 + {alpha[1]}x^2 + {alpha[2]}x + {alpha[3]}')
plt.show()
```



1.4 Anpassning med plan

Antag att $z = Ax + By + C$ Hitta A,B,C

[101]:
```
import numpy as np
# import plotly.graph_objects as go
import matplotlib.pyplot as plt

# generate X and Y
number_pts=20
true_A = 0.3
true_B = 0.9
true_C = -0.3
spread = 3/2
x = np.linspace(0, 1, number_pts)
y = np.linspace(0, 1, number_pts)
xx, yy = np.meshgrid(x, y)
```

```
noise = spread*np.random.random(len(xx)**2)
noise = noise.reshape(number_pts,number_pts)

zz = true_A*xx + true_B*yy + true_C
zz = zz + noise
```

[102]:
```
xx[:2]
```

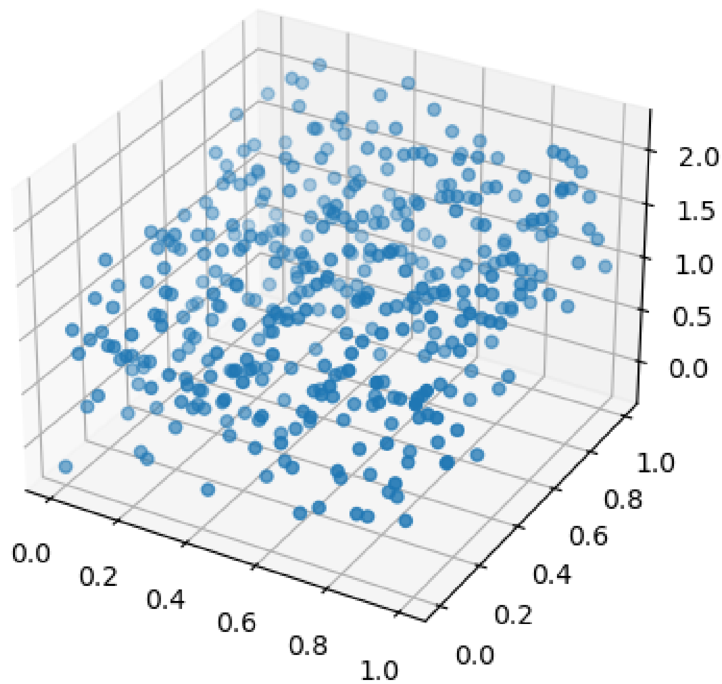
[102]:
```
array([[0.        , 0.05263158, 0.10526316, 0.15789474, 0.21052632,
        0.26315789, 0.31578947, 0.36842105, 0.42105263, 0.47368421,
        0.52631579, 0.57894737, 0.63157895, 0.68421053, 0.73684211,
        0.78947368, 0.84210526, 0.89473684, 0.94736842, 1.        ],
       [0.        , 0.05263158, 0.10526316, 0.15789474, 0.21052632,
        0.26315789, 0.31578947, 0.36842105, 0.42105263, 0.47368421,
        0.52631579, 0.57894737, 0.63157895, 0.68421053, 0.73684211,
        0.78947368, 0.84210526, 0.89473684, 0.94736842, 1.        ]])
```

[103]:
```
# X = xx.flatten()
# Y = yy.flatten()
# Z = zz.flatten()
```

[104]:
```
# needs plotly
#planfig = go.Figure(data=[go.Scatter3d(
#    x=X,
#    y=Y,
#    z=Z,
#    mode='markers',
#    marker=dict(
#        size=3,
#        color=Z,                # set color to an array/list of desired values
#        colorscale='Viridis',   # choose a colorscale
#        opacity=0.8
#    )
#)])

# tight layout
#planfig.update_layout(margin=dict(l=0, r=0, b=0, t=0), width=800, height=500)
#planfig.show()
```

[105]:
```
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
ax.scatter(xx, yy, zz, marker='o')
plt.show()
```



[106]:
```
# assemble matrix A
A = np.vstack([X, Y, np.ones(len(X))]).T
# turn Z into a column vector
B = Z[:, np.newaxis]
A[:10],B[:10]
```

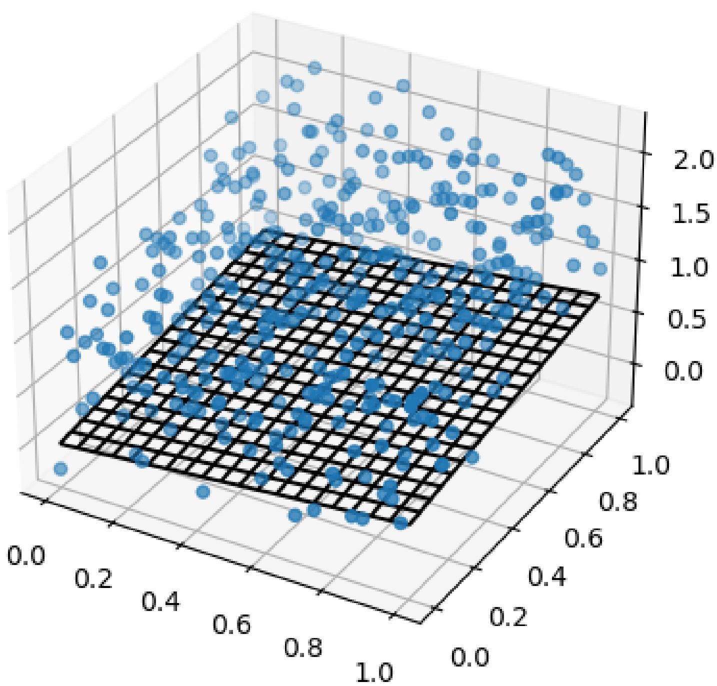
[106]:
```
(array([[0.        , 0.        , 1.        ],
        [0.11111111, 0.        , 1.        ],
        [0.22222222, 0.        , 1.        ],
        [0.33333333, 0.        , 1.        ],
        [0.44444444, 0.        , 1.        ],
        [0.55555556, 0.        , 1.        ],
        [0.66666667, 0.        , 1.        ],
        [0.77777778, 0.        , 1.        ],
        [0.88888889, 0.        , 1.        ],
        [1.        , 0.        , 1.        ]]),
 array([[-0.0275095 ],
        [ 0.06238034],
        [-0.15536034],
        [ 0.16322372],
        [-0.14262796],
        [ 0.15143792],
        [ 0.02380345],
```

```
       [ 0.04888575],
       [ 0.43710728],
       [ 0.00215741]]))
```

[107]:
```
# Löser normalekvationerna A^t*A*Z=A^t*B
# genom Z = (A^t*A)^{-1}*A^t*B
alpha = np.dot((np.dot(np.linalg.inv(np.dot(A.T,A)),A.T)),B)
print(alpha)
```

```
[[0.27627034]
 [0.42914214]
 [0.00475092]]
```

[108]:
```
fig2 = plt.figure()
ax2 = fig2.add_subplot(projection='3d')
ax2.scatter(xx, yy, zz, marker='o')
zzfit= alpha[0]*xx + alpha[1]*yy + alpha[2]
ax2.plot_wireframe(xx, yy, zzfit,color='black')
plt.show()
```



[ ]: