ATE: PAGE NO.

Lab Program - 10

→ Implement the non-parametric Locally Weighted Regression algorithm to fit data points. Select appropriate data set for your experiment and draw graphs.

Theory -

Let us consider the case of locally weighted regression in which the target function f is approximated near x, using a linear function of the form.

1 Minimize the squared error over just the k nearest neighbours:

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in K \text{ nearest nbrs of } x_q} (f(x) - \hat{f}(x))^2$$

2 Minimized the squared error over the entire set D of training examples, while weighting the error of each training example by some decreasing function K of its distance from x_q:

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

3 Combine 1 and 2:

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in K \text{ nearest nbrs of } x_q} (f(x) - \hat{f}(x))^2 K(d(x_q, x$$

Teacher's Signature

Shubhransh Gupta : 1BG17CS094

DATE: PAGE NO.

Program-

```
from math import ceil
import numpy as np
from scipy import linalg

def lowest(x, y, f=2/3, iter=3):
    n = len(x)
    r = int(ceil(f*n))
    h = [np.sort(np.abs(x-x[i]))[r] for i in range(n)]
    w = np.clip(np.abs((x[:,None]-x[None,:])/h),0.0,1.0)
    w = (1-w**3)**3
    yest = np.zeros(n)
    delta = np.ones(n)
    for iteration in range(iter):
        for i in range(n):
            weights = delta * w[:,i]
            b = np.array([np.sum(weight*y), np.sum(weights*
                                                   y*n)])
            A = np.array([[np.sum(weights), np.sum(weights*n)],
                          [np.sum(weights*n), np.sum(weights*x)]])
            beta = linalg.solve(A,b)
            yest[i] = beta[0] + bet[1] * x[i]


    residuals = y- yest
    s = np.median(np.abs(residuals))
    delta = np.clip(residuals / (6.0 * s), -1, 1)
    delta = (1- delta**2)**2
```

Teacher's Remarks

81

Teacher's Signature

DATE: PAGE NO.

```
return yest

if __name__ == '__main__':
    import math
    n = 100
    x = np.linspace(0, 2 * math.pi, n)
    print(x)
    y = np.sin(x) + 0.3 * np.random.randn(n)
    print(y)
    f = 0.25
    yest = lowest(x, y, f=f, iter=3)
    import pylab as pl
    pl.clf()
    pl.plot(x, y, label='y noisy')
    pl.plot(x, yest, label='y pred')
    pl.legend()
    pl.show()
```

```
*Python 3.7.0b1 Shell*
File  Edit  Shell  Debug  Options  Window  Help
>>>
========== RESTART: D:\ML\ML Lab Programs\10-regression\regression.py ==========
==========values of x==========
[0.         0.06346652 0.12693304 0.19039955 0.25386607 0.31733259
 0.38079911 0.44426563 0.50773215 0.57119866 0.63466518 0.6981317
 0.76159822 0.82506474 0.88853126 0.95199777 1.01546429 1.07893081
 1.14239733 1.20586385 1.26933037 1.33279688 1.3962634  1.45972992
 1.52319644 1.58666296 1.65012947 1.71359599 1.77706251 1.84052903
 1.90399555 1.96746207 2.03092858 2.0943951  2.15786162 2.22132814
 2.28479466 2.34826118 2.41172769 2.47519421 2.53866073 2.60212725
 2.66559377 2.72906028 2.7925268  2.85599332 2.91945984 2.98292636
 3.04639288 3.10985939 3.17332591 3.23679243 3.30025895 3.36372547
 3.42719199 3.4906585  3.55412502 3.61759154 3.68105806 3.74452458
 3.8079911  3.87145761 3.93492413 3.99839065 4.06185717 4.12532369
 4.1887902  4.25225672 4.31572324 4.37918976 4.44265628 4.5061228
 4.56958931 4.63305583 4.69652235 4.75998887 4.82345539 4.88692191
 4.95038842 5.01385494 5.07732146 5.14078798 5.2042545  5.26772102
 5.33118753 5.39465405 5.45812057 5.52158709 5.58505361 5.64852012
 5.71198664 5.77545316 5.83891968 5.9023862  5.96585272 6.02931923
 6.09278575 6.15625227 6.21971879 6.28318531]
==========Values of y==========
[ 6.88106030e-04 -1.53527916e-01 -4.98737481e-01  1.66052067e-02
  4.86586342e-01  5.59773603e-01  1.24770523e+00 -2.69282742e-01
  3.83486911e-01  4.78805302e-01  7.18032090e-01  1.06634463e-01
  8.42933418e-01  1.04366178e+00  9.30413454e-01  1.01345445e+00
  9.82183103e-01  9.45759176e-01  4.75508863e-01  1.46202254e+00
  9.63727398e-01  7.29364864e-01  1.08280866e+00  1.34037389e+00
  7.84259178e-01  7.66175421e-01  1.31624987e+00  1.47139621e+00
  8.04798599e-01  8.12591280e-01  1.22360577e+00  1.15719055e+00
  6.84623550e-01  7.69221924e-01  2.78289358e-01  5.81659677e-01
  1.37037559e+00  9.40211539e-01  5.31299720e-01  6.26264032e-01
  7.49614838e-01  8.13147491e-01  5.46453233e-01  4.81099301e-01
  6.22229293e-01  3.21262080e-02  2.53146620e-01 -3.44801420e-02
  4.89829335e-01  8.75060677e-02 -6.03342638e-01 -2.17350719e-01
  2.32082134e-01 -4.38989269e-01  3.99743480e-02 -1.89615065e-01
 -1.91664335e-01 -6.42399360e-01 -3.72766160e-01 -2.82936950e-01
 -6.77408578e-01 -9.65816954e-01 -7.16975131e-01 -4.65038775e-01
 -2.35525338e-01 -7.51120676e-01 -5.92095239e-01 -4.98764638e-01
```