Program 10.

Implement the non-parametric **Locally Weighted regression** algorithm In order to fit the data points . Select appropriate data set for your experiment and draw graph.

**Nonparametric regression :**  is a category of regression analysis in which the predictor does not take a predetermined form but is constructed according to information derived from the data. Nonparametric regression requires larger sample sizes than regression based on parametric models because the data must supply the model structure as well as the model estimates.

Nonparametric regression is used for prediction and is reliable even if hypotheses of linear regression are not verified.

**Locally weighted Learning** also known as memory-based learning, instance-based learning, lazy-learning, and closely related to kernel density estimation, similarity searching and case-based reasoning.

LOWESS (Locally Weighted Scatterplot Smoothing), sometimes called LOESS (locally weighted smoothing), is a popular tool used in [regression analysis](http://www.statisticshowto.com/probability-and-statistics/regression-analysis/) that creates a smooth line through a [timeplot](http://www.statisticshowto.com/timeplot/" \t "_blank) or [scatter plot](http://www.statisticshowto.com/probability-and-statistics/regression-analysis/scatter-plot-chart/#definition) to help you to see relationship between [variables](http://www.statisticshowto.com/variable/) and foresee trends.

[Locally weighted regression](https://en.wikipedia.org/wiki/Local_regression) is a very powerful non-parametric model used in statistical learning.

**Introduction :**

Scatter-diagram smoothing (e.g. using the **lowess()** or **loess()** functions) involves drawing a smooth curve on a scatter diagram to summarize a relationship, in a fashion that makes few assumptions initially about the form or strength of the relationship. It is related to (and is a special case of) nonparametric regression, in which the objective is to represent the relationship between a response variable and one or more predictor variables, again in way that makes few assumptions about the form of the relationship. In other words, in contrast to “standard” linear regression analysis, no assumption is made that the relationship is represented by a straight line (although one could certainly think of a straight line as a special case of nonparametric regression).

If the basic decomposition-of-the-data model is:

        data = predictable component + noise,

then for the standard bivariate or multiple (linear) regression, the model is

        data = straight-line, polynomial or linearizable function + noise,

while for nonparametric regression, the model is

        data = smooth function determined by data + noise.

Another way of looking at scatter diagram smoothing is as a way of depicting the “local” relationship between a response variable and a predictor variable over parts of their ranges, which may differ from a “global” relationship determined using the whole data set. Nonparametric regression can be thought of as generalizing the scatter plot smoothing idea to the multiple-regression context.

Locally Weighted Learning is a class of function approximation techniques, where a prediction is done by using an approximated local model around the current point of interest.

The **goal** of function approximation and regression is to find the underlying relationship between input and output. In a supervised learning problem training data, where each input is associated to one output, is used to create a model that predicts values which come close to the true function. All of these models use complete training data to derive global function.

**Locally weighted regression**

**Local** means using nearby points (i.e. a nearest neighbors approach)

**Weighted** means we value points based upon how far away they are.

**Regression** means approximating a function

This is **an instance-based learning method**

**The idea:** **whenever you want to classify a sample:**

* Build a local model of the function (using a linear function, quadratic, neural network, etc.)
* Use the model to predict the output value
* Throw the model away.

## *Locally Weighted Regression*

Our final method combines advantages of parametric methods with non-parametric. The idea is to fit a regression model locally, weighting examples by the kernel K.

Locally Weighted Regression Algorithm

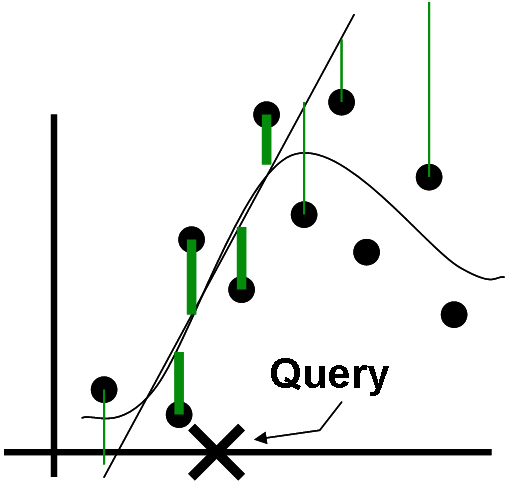
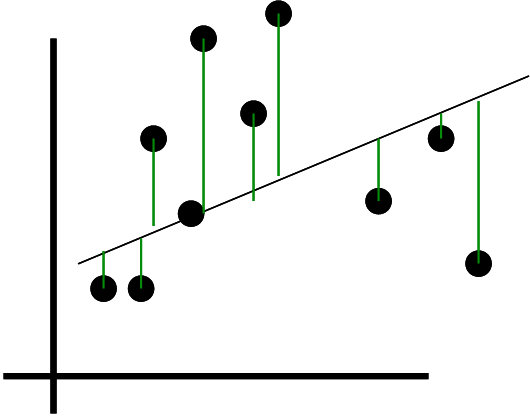
1. Given training data *D*=http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmsy10/alpha/100/char66.png**x***i*http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmmi10/alpha/100/char3B.png*yi*http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmsy10/alpha/100/char67.png , Kernel function *K*(http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmsy10/alpha/100/char01.pnghttp://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmmi10/alpha/100/char3B.pnghttp://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmsy10/alpha/100/char01.png) and input **x**
2. Fit weighted regression **w**http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmr10/alpha/100/char5E.png(**x**)=argmin*w*http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmex10/alpha/100/char50.png*ni*=1*K*(**x**http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmmi10/alpha/100/char3B.png**x***i*)(**w**http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmsy10/alpha/70/char3E.png**x***i*−*yi*)2
3. Return regression prediction **w**http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmr10/alpha/100/char5E.png(**x**)http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmsy10/alpha/70/char3E.png**x**.

Note that we can do the same for classification, fitting a locally weighted logistic regression:

Locally Weighted Logistic Regression Algorithm

1. Given training data *D*=http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmsy10/alpha/100/char66.png**x***i*http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmmi10/alpha/100/char3B.png*yi*http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmsy10/alpha/100/char67.png , Kernel function *K*(http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmsy10/alpha/100/char01.pnghttp://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmmi10/alpha/100/char3B.pnghttp://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmsy10/alpha/100/char01.png) and input **x**
2. Fit weighted logistic regression **w**http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmr10/alpha/100/char5E.png(**x**)=argmin*w*http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmex10/alpha/100/char50.png*ni*=1*K*(**x**http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmmi10/alpha/100/char3B.png**x***i*)log(1+exphttp://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmsy10/alpha/100/char66.png−*yi***w**http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmsy10/alpha/70/char3E.png**x***i*http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmsy10/alpha/100/char67.png)
3. Return logistic regression prediction *sign*(**w**http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmr10/alpha/100/char5E.png(**x**)http://learning.cis.upenn.edu/cis520_fall2009/pub/jsMath/fonts/cmsy10/alpha/70/char3E.png**x**).

The difference between regular linear regression and locally weighted linear regression can be visualized as follows:



**x** ---- is an instance,

**D** ---- is the set of possible instances

**ai(x)**---- is the value of the i th attribute

**wi** ----The weights form our hypothesis

**f** ----is the target function

**f ˆ** ----is our approximation to the target function

In this case, we use a linear model to do the local approximation f ˆ :

**fˆ(x) = w0 + w1a1(x) + · · · + wnan(x)**

**LOWESS is typically used for:**

* Fitting a line to a scatter plot or time plot where noisy data values, sparse data points or weak interrelationships interfere with your ability to see a line of best fit.
* [Linear regression](http://www.statisticshowto.com/probability-and-statistics/regression-analysis/find-a-linear-regression-equation/) where [least squares fitting](http://www.statisticshowto.com/least-squares-regression-line/#LSFitting) doesn’t create a line of good fit or is too labor-intensive to use.
* Data exploration and analysis in the social sciences, particularly in elections and voting behavior.

**Parametric and Non-Parametric Fitting**

LOWESS, and least squares fitting in general, are [**non-parametric**](http://www.statisticshowto.com/parametric-and-non-parametric-data/) strategies for fitting a smooth curve to data points. “[Parametric](http://www.statisticshowto.com/parametric-statistics/)” means that the researcher or analyst assumes in advance that the data fits some type of distribution (i.e. the [normal distribution](http://www.statisticshowto.com/probability-and-statistics/normal-distributions/)). Because some type of distribution is assumed in advance, parametric fitting can lead to fitting a smooth curve that misrepresents the data. In those cases, non-parametric smoothers may be a better choice. Non-parametric smoothers like LOESS try to find a curve of best fit without assuming the data must fit some distribution shape. In general, both types of smoothers are used for the same set of data to offset the advantages and disadvantages of each type of smoother.  
  
**Benefits of Non-Parametric Smoothing**

* Provides a flexible approach to representing data.
* Ease of use.
* Computations are relatively easy.

**Disadvantages of Non-Parametric Smoothing**

* Can’t be used to obtain a simple equation for a set of data.
* Less well understood than parametric smoothers.
* Requires the analyst to use a little guesswork to obtain a result.

**PROGRAM**

|  |
| --- |
| """ |
|  | This module implements the Lowess function for nonparametric regression. |
|  |  |
|  | Functions: |
|  | lowess Fit a smooth nonparametric regression curve to a scatterplot. |
|  |  |
|  | For more information, see |
|  |  |
|  | William S. Cleveland: "Robust locally weighted regression and smoothing |
|  | scatterplots", Journal of the American Statistical Association, December 1979, |
|  | volume 74, number 368, pp. 829-836. |
|  |  |
|  | William S. Cleveland and Susan J. Devlin: "Locally weighted regression: An |
|  | approach to regression analysis by local fitting", Journal of the American |
|  | Statistical Association, September 1988, volume 83, number 403, pp. 596-610. |
|  | """ |
|  |  |
|  | from math import ceil |
|  | import numpy as np |
|  | from scipy import linalg |
|  |  |
|  |  |
|  | def lowess(x, y, f=2./3., iter=3): |
|  | """lowess(x, y, f=2./3., iter=3) -> yest |
|  |  |
|  | Lowess smoother: Robust locally weighted regression. |
|  | The lowess function fits a nonparametric regression curve to a scatterplot. |
|  | The arrays x and y contain an equal number of elements; each pair |
|  | (x[i], y[i]) defines a data point in the scatterplot. The function returns |
|  | the estimated (smooth) values of y. |
|  |  |
|  | The smoothing span is given by f. A larger value for f will result in a |
|  | smoother curve. The number of robustifying iterations is given by iter. The |
|  | function will run faster with a smaller number of iterations.""" |
|  | 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(weights\*y), np.sum(weights\*y\*x)]) |
|  | A = np.array([[np.sum(weights), np.sum(weights\*x)], |
|  | [np.sum(weights\*x), np.sum(weights\*x\*x)]]) |
|  | beta = linalg.solve(A, b) |
|  | yest[i] = beta[0] + beta[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 |
|  |  |
|  | return yest |
|  |  |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  | import math |
|  | n = 100 |
|  | x = np.linspace(0, 2 \* math.pi, n) ###Generate the datset## |
|  | y = np.sin(x) + 0.3\*np.random.randn(n) |
|  |  |
|  | f = 0.25 |
|  | yest = lowess(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() |