Lecture 02
Linear classification methods I

22 January 2016

Taylor B. Arnold
Yale Statistics
STAT 365/665

Yale

Course website:

A copy of the whole course syllabus, including a more detailed description of topics I plan to cover are on the course website.

`http://www.stat.yale.edu/~tba3/stat665/`

This is where all lecture notes, homeworks, and other references will appear.

Course website:

A copy of the whole course syllabus, including a more detailed description of topics I plan to cover are on the course website.

`http://www.stat.yale.edu/~tba3/stat665/`

This is where all lecture notes, homeworks, and other references will appear.

The first problem set is now posted.

Some thoughts from course survey

1. several people mentioned Matlab as alternative to R or Python
2. equal interest in CV and NLP
3. requests for finance and biological applications
4. several requests for explaining methods for large, distributed computations (Hadoop, Spark, MPI)
5. a diverse set of background in statistics, computer science, and data analysis

Today, we will consider 1-dimensional non-parametric regression. Specifically, we will consider observing n pairs (x_i, y_i) such that:

$$y_i = g(x_i) + \epsilon_i$$

For an unknown function g and random variable ϵ_i, where the mean of the random variable is zero.

Today, we will consider 1-dimensional non-parametric regression. Specifically, we will consider observing n pairs (x_i, y_i) such that:

$$y_i = g(x_i) + \epsilon_i$$

For an unknown function g and random variable ϵ_i, where the mean of the random variable is zero.

You are free to think of the errors being independent, identically distributed and with finite variances. You may even assume that they are normally distributed.

Our goal is to estimate the function g. More specifically, we want to estimate the value of g at the input point x_{new}, denoted as $\widehat{g}(x_{new})$.

Our goal is to estimate the function g. More specifically, we want to estimate the value of g at the input point x_{new}, denoted as $\widehat{g}(x_{new})$.

The value of x_{new} may be in the original set and it may not be; however, we assume that ϵ_{new} is an entirely different random sample.

Our goal is to estimate the function g. More specifically, we want to estimate the value of g at the input point x_{new}, denoted as $\widehat{g}(x_{new})$.

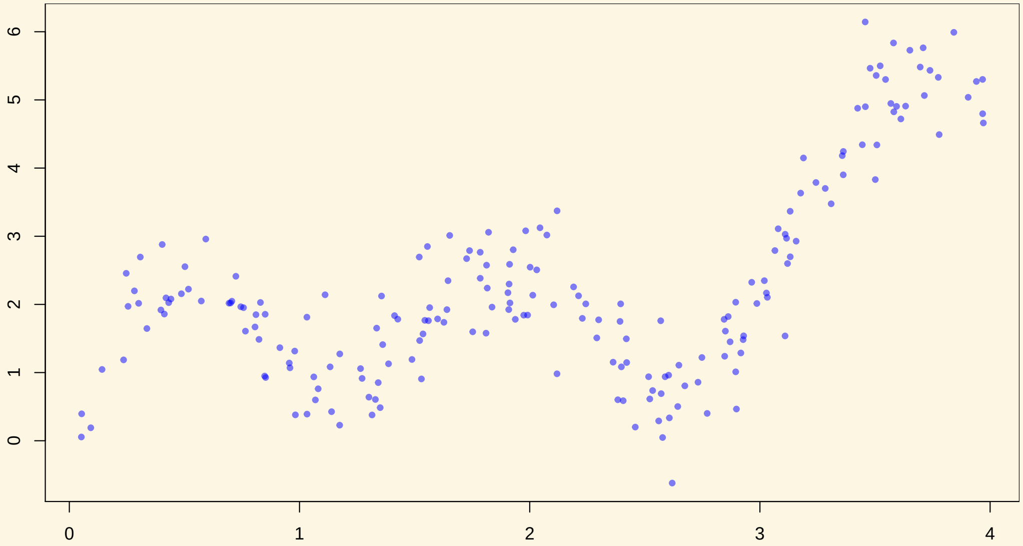
The value of x_{new} may be in the original set and it may not be; however, we assume that ϵ_{new} is an entirely different random sample.

Next class I will cover more of the details about training, testing, validation, and other techniques for evaluating how well a predictive model is performing. Today we will just spend some time introducing several standard techniques and gaining some intuition about them.

The specific methods I will introduce are:

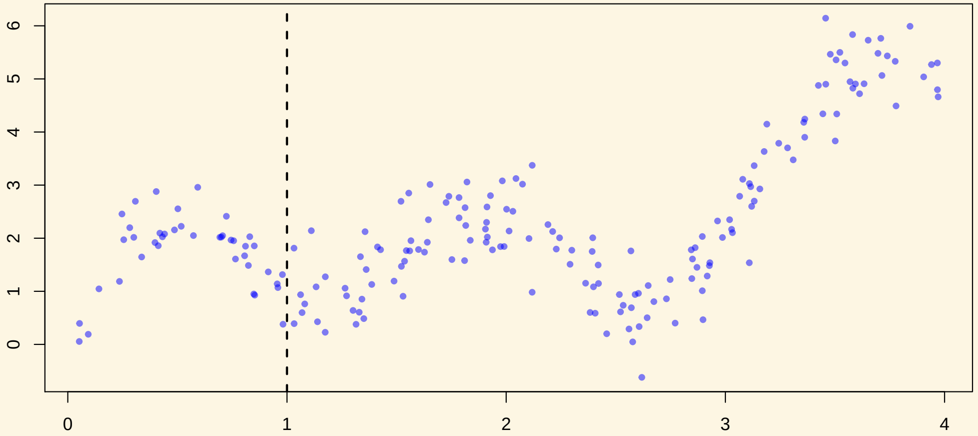
1. k-nearest neighbors (knn)
2. kernel smoothing (Nadaraya-Watson estimator)
3. linear regression (OLS)
4. local regression (LOESS)

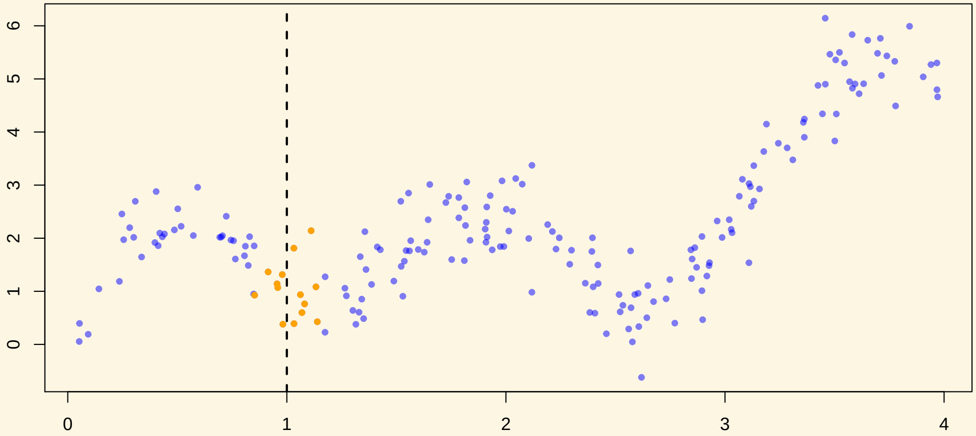
Here is an example dataset that we will use to illustrate these four techniques:

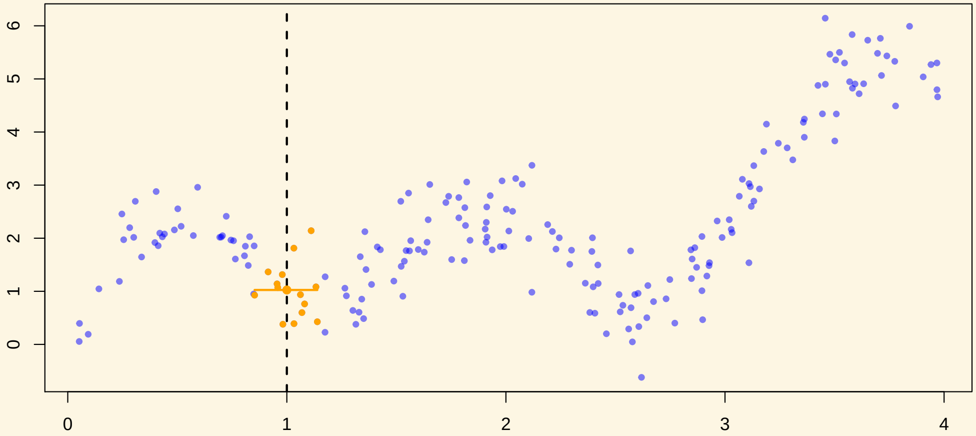


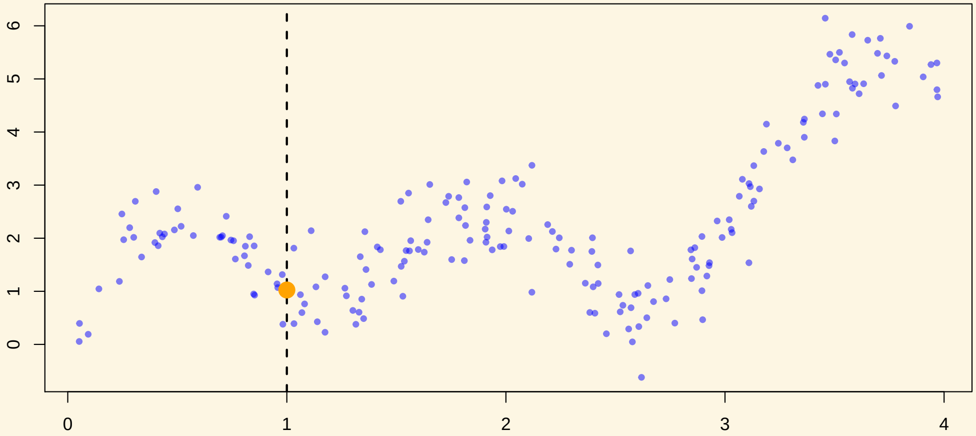
k-Nearest neighbors (knn)

One of the most straightforward estimators, knn simply sets $\widehat{g}(x_{new})$ to be the average value of the observed y of the k closest points x_j to x_{new}.


0
1
2
3
4
5
6
0
1
2
3
4


0
1
2
3
4
5
6
0
1
2
3
4


0
1
2
3
4
5
6
0
1
2
3
4


0
1
2
3
4
5
6
0
1
2
3
4

k-Nearest neighbors (knn)

How might you expect the optimal parameter k to change as n increases?

Kernel smoother

Rather than averaging the k-closest points, kernel smoothers average observations in the dataset using weights that are inversely proportional to the distance from the prediction point.

Kernel smoothers

As an example, consider this estimator

$$\widehat{g}(x_{new}) = \frac{\sum_i y_i \cdot \phi(||x_i - x_{new}||_2^2)}{\sum_i \phi(||x_i - x_{new}||_2^2)}$$

Where ϕ is defined as the density function of a standard normal distribution:

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2}$$

Kernel smoothers

The function ϕ can be replaced with any other function that you would like to use. It is often replaced by a truncated variant, as observations more than a few standard deviations do not give a noticeable impact on the result.

Kernel smoothers - bandwidth

What is the main tuning parameter for kernel smoothers? We need to modify our estimator slightly to include the bandwidth parameter h:

$$\widehat{g}(x_{new}) = \frac{\sum_i y_i \cdot \phi(||x_i - x_{new}||_2^2/h)}{\sum_i \phi(||x_i - x_{new}||_2^2/h)}$$

What does the bandwidth control?

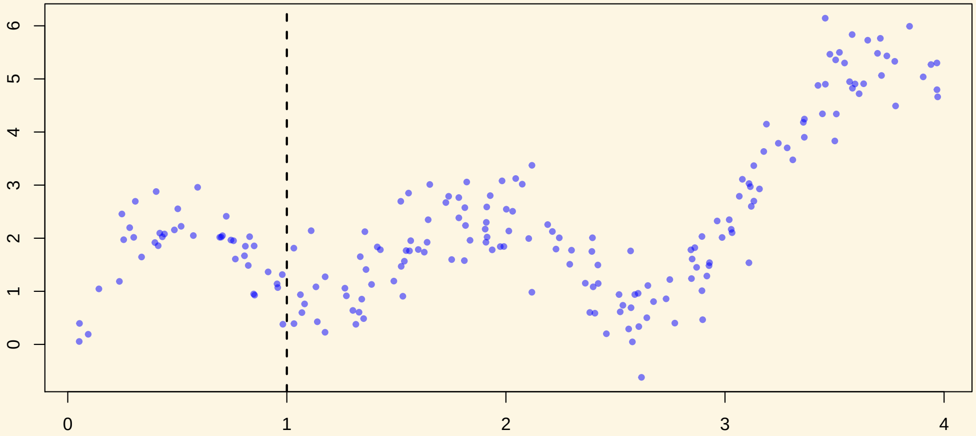
Kernel smoothers - bandwidth

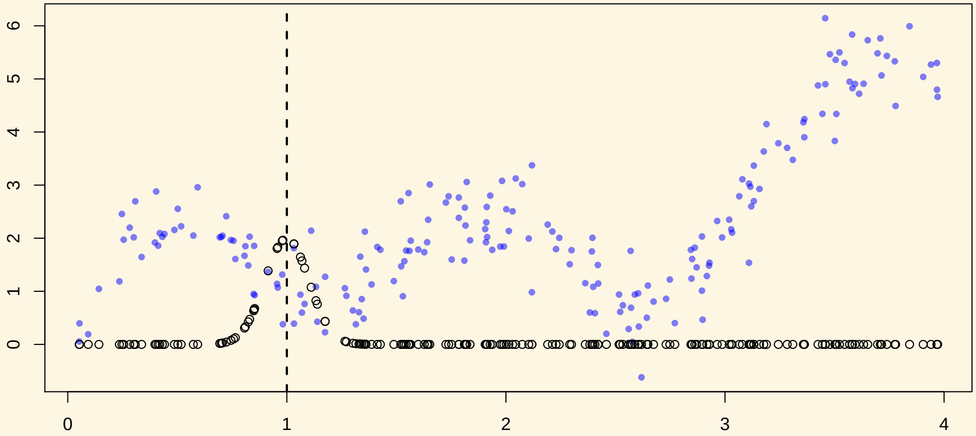
What is the main tuning parameter for kernel smoothers? We need to modify our estimator slightly to include the bandwidth parameter h:

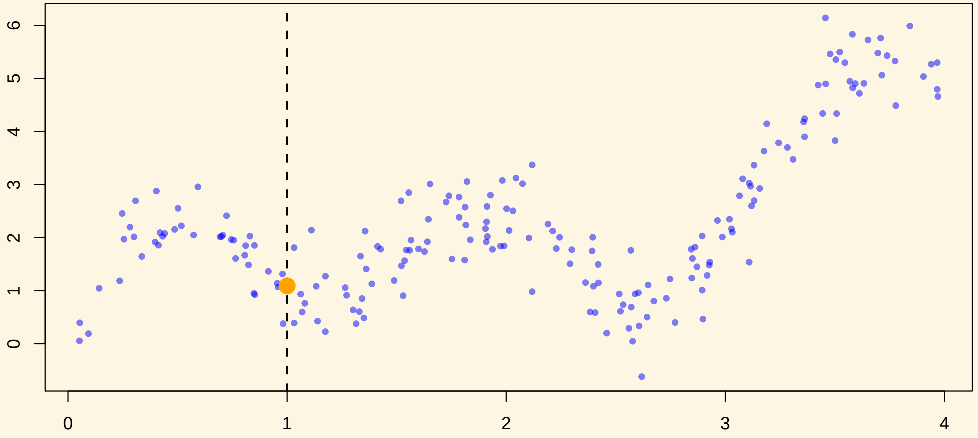
$$\widehat{g}(x_{new}) = \frac{\sum_i y_i \cdot \phi(||x_i - x_{new}||_2^2/h)}{\sum_i \phi(||x_i - x_{new}||_2^2/h)}$$

What does the bandwidth control?

Why might we think of the bandwidth as a standard deviation?


0
1
2
3
4
5
6
0
1
2
3
4


0
1
2
3
4
5
6
0
1
2
3
4


0
1
2
3
4
5
6
0
1
2
3
4

Linear regression

Hopefully you have already seen linear regression in some context. Recall that here we assume the data are generated by a process that looks something like this:

$$y_i = x_{1,i}\beta_1 + x_{2,i}\beta_2 + \cdots + x_{p,1}\beta_p + \epsilon_i$$

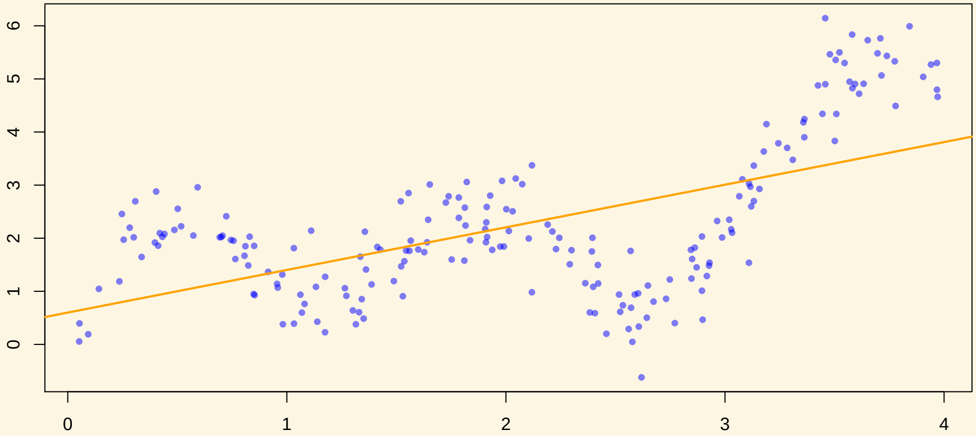
For some random variable ϵ_i and fixed (but unknown) parameters β_j.

The parameters β_j are most commonly estimated by finding the values that minimize the squared residuals (known as ols, or ordinary least squares).

Linear regression

In our one dimensional case, even including an intercept, this does not seem very interesting as we have only two parameters to use to fit the data:

$$y_i = \beta_0 + x_i\beta_1 + \epsilon_i$$



0
1
2
3
4
5
6
0
1
2
3
4

Linear regression

Why might ordinary least squares be useful in non-parametric regression?

Linear regression

Why might ordinary least squares be useful in non-parametric regression? basis functions!

Linear regression

Why might ordinary least squares be useful in non-parametric regression? basis functions!

We can expand the model to include non-linear terms. For example, we can expand to write y as a power basis:

$$y_i = \beta_0 + \sum_{j=1}^{p} x_i^j \cdot \beta_j + \epsilon_i$$

Or as a Fourier basis:

$$y_i = \beta_0 + \sum_{j=1}^{p} \sin(x_i * j) \cdot \beta_{2j} + \cos(x_i * j) \cdot \beta_{2j+1} + \epsilon_i$$

Linear regression

With enough terms this basis expansion can approximate nearly all functional forms of g.

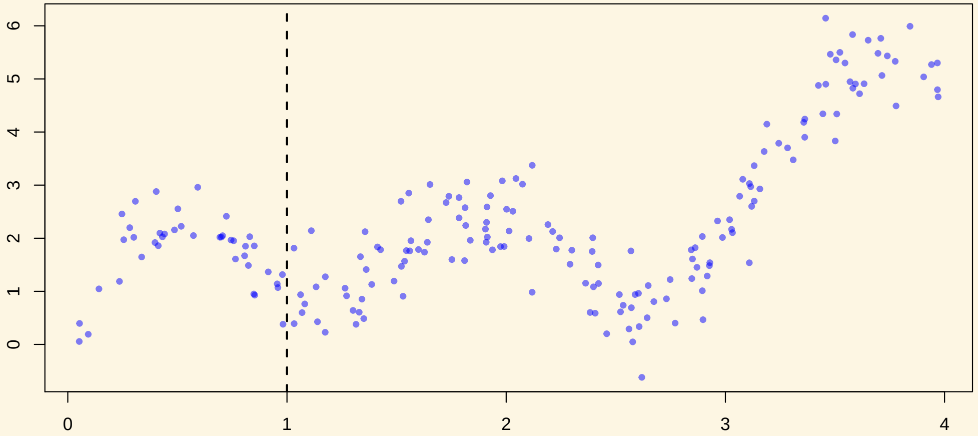
local regression (LOESS)

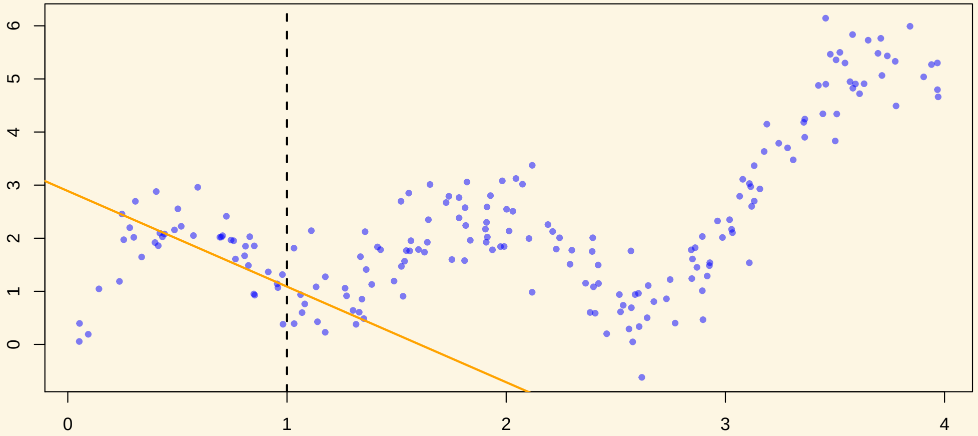
Local regression combines the ideas of kernel smoothers and linear regression. A separate linear regression is fit at each input point x_{new} for which $\widehat{g}$ is to be evaluated at. The regression uses sample weights based on how for each sample is from x_{new}.

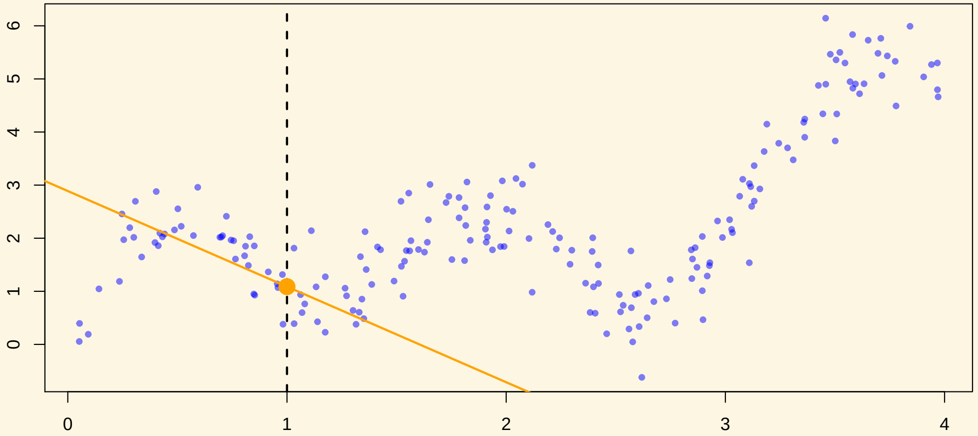
local regression (LOESS)

Local regression combines the ideas of kernel smoothers and linear regression. A separate linear regression is fit at each input point x_{new} for which $\widehat{g}$ is to be evaluated at. The regression uses sample weights based on how for each sample is from x_{new}.

One can use linear regression or use higher order polynomials. Typically at most cubic functions are used.


0
1
2
3
4
5
6
0
1
2
3
4


0
1
2
3
4
5
6
0
1
2
3
4


0
1
2
3
4
5
6
0
1
2
3
4

Linear smoothers

How are these four techniques all related?

Linear smoothers

How are these four techniques all related?

All of them can be written as:

$$\widehat{y}_{new} = \sum_i w_i y_i$$

Where the weights w_i depend only on the values x_i (and x_{new}) and not on the values of y_i.

Linear smoothers

How are these four techniques all related?

All of them can be written as:

$$\widehat{y}_{new} = \sum_i w_i y_i$$

Where the weights w_i depend only on the values x_i (and x_{new}) and not on the values of y_i.

Anything that can be written in this form is called a linear smoother.

Linear smoothers, cont.

For example, in the case of knn, the weights are $\frac{1}{k}$ for the k closest values and 0 for the remained of them.

Linear smoothers, cont.

For example, in the case of knn, the weights are $\frac{1}{k}$ for the k closest values and 0 for the remained of them.

We have already seen the weights for the kernel smoother.

Linear smoothers, cont.

For example, in the case of knn, the weights are $\frac{1}{k}$ for the k closest values and 0 for the remained of them.

We have already seen the weights for the kernel smoother.

The weights for ordinary least squares are given by our derived ordinary least squares estimator:

$$w = x_{new}(X^t X)^{-1} X^t$$

And LOWESS is simply a sample weighted variant of this.

Still to come...

- computational issues of computing them
- how to pick tuning parameters and choose amongst them
- what happens when we have higher dimensional spaces