Instance Based Learning

Some material on these is slides borrowed from Andrew Moore's excellent machine learning tutorials located at:

http://www.cs.cmu.edu/~awm/tutorials/

Problems with Neural Networks

- Networks learn by tweaking parameters to fit the data.
- Then the data is thrown away.
- Problems:
  - Training to fit new data may erase what we learned before.
  - We need to have the right set of parameters.

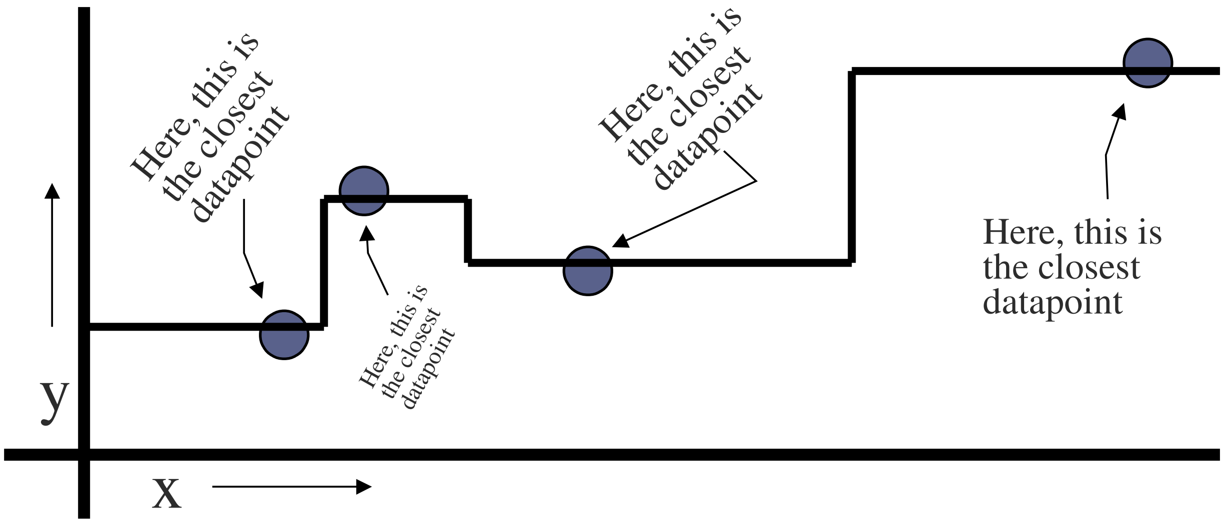
Instance Based Learning

- Keep all of the training data around.
  - Refer back to it when we need to make a prediction
- Simplest example: 1-Nearest Neighbor.
- Given input-output pairs: $(\boldsymbol{x}_{\mathbf{1},} y_1), (\boldsymbol{x}_{\mathbf{2},} y_2), \ldots, (\boldsymbol{x}_{\boldsymbol{N}}, y_N)$ that come from some unknown function $y = f(\boldsymbol{x})$.
- Given a query, find the nearest input point:

$$c = \underset{\boldsymbol{i}}{argmin}(\|\boldsymbol{x}_{\boldsymbol{i}} - \boldsymbol{x}_{\boldsymbol{q}}\|)$$

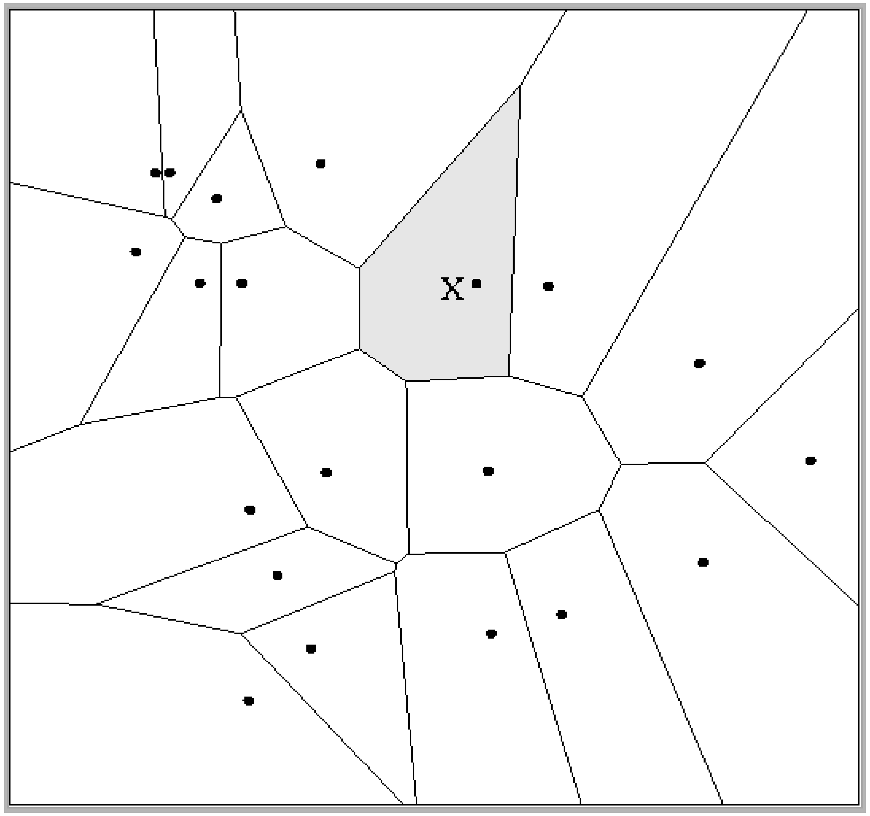
- Then predict $\hat{y} = y_c$

1-Dimensional Example...



http://www.cs.cmu.edu/~awm/tutorials/

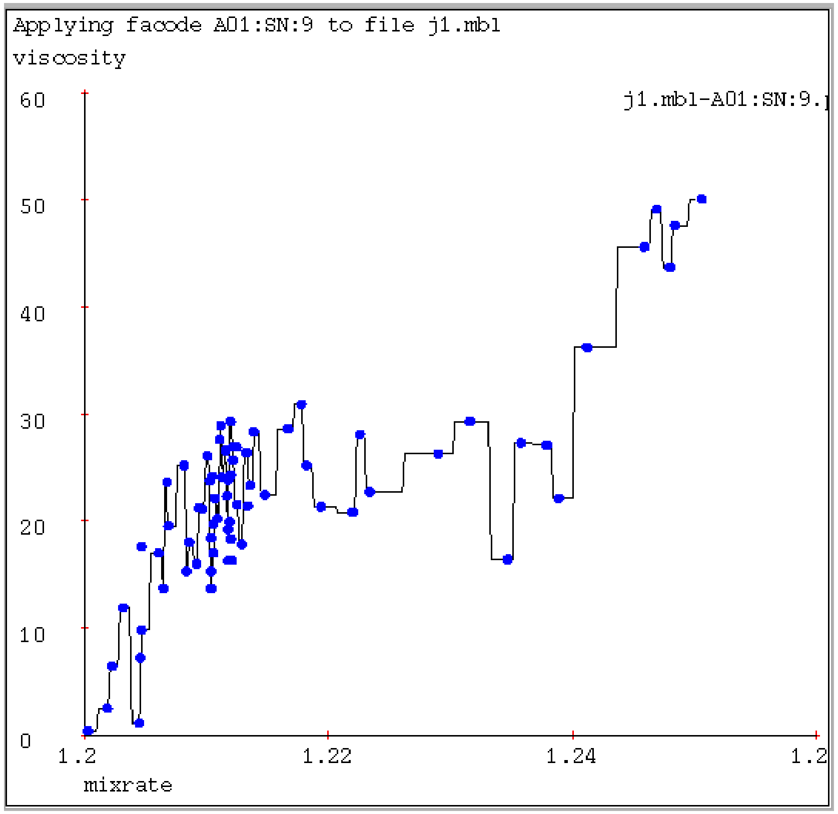
2-Dimensional Example



http://www.cs.cmu.edu/~awm/tutorials/

Problems With 1-NN

- No interpolation.
- Susceptible to noise.

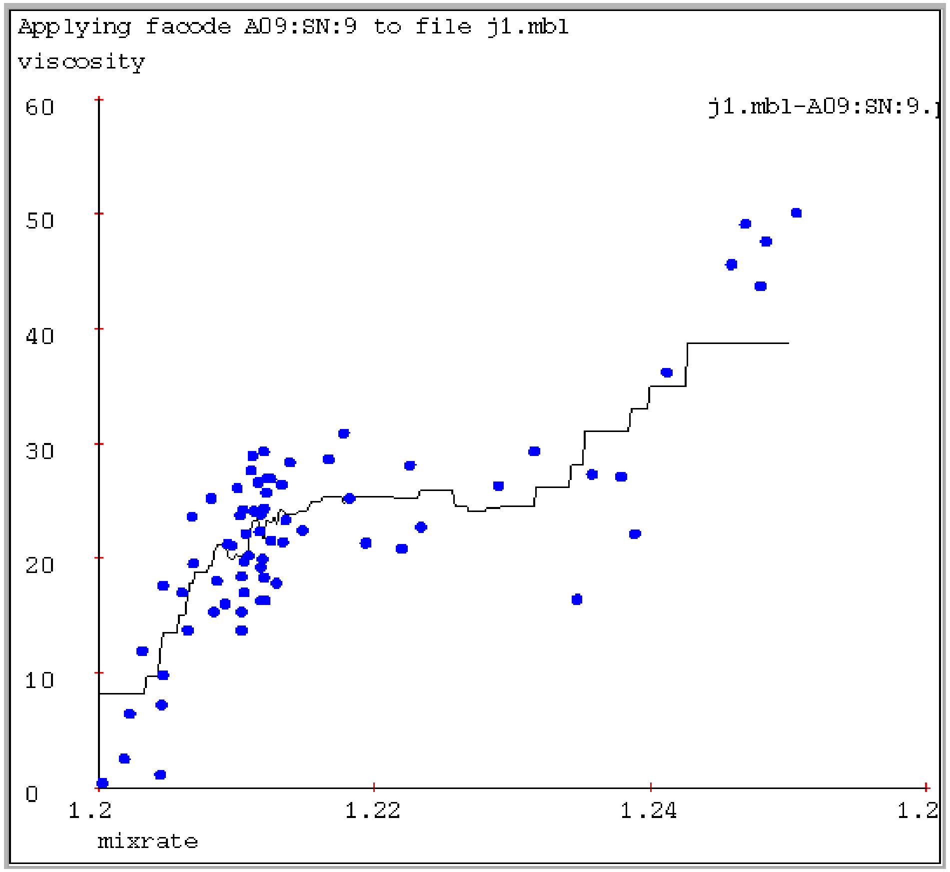


http://www.cs.cmu.edu/~awm/tutorials/

Solution (?) K-NN

- Average the output values of the k nearest neighbors – Better.
- Odd behavior at the edges.
- The fit is jerky.
- (We can find neighbors efficiently using kd-trees)

9-NN



http://www.cs.cmu.edu/~awm/tutorials/

Solution (?) Kernel Regression

- Use <u>all</u> of the training points for every query.
- Take a weighted average, where the weight is based on a kernel function *K*:

$$\hat{f}(\boldsymbol{x}_q) = \frac{\sum_i K(\boldsymbol{x}_i, \boldsymbol{x}_q)\, y_i}{\sum_i K(\boldsymbol{x}_i, \boldsymbol{x}_q)}$$

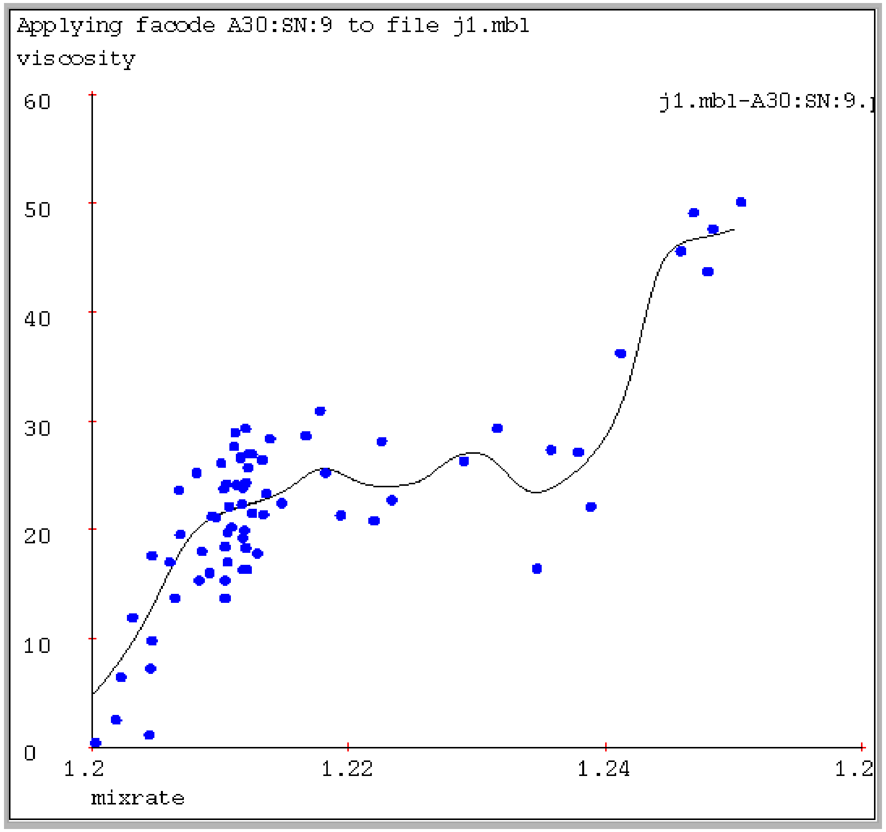
- K is often the Gaussian function:

$$K(\boldsymbol{x}_q, \boldsymbol{x}_i) = \frac{1}{\left(w^2\sqrt{2\pi}\right)^d}\; e^{-\frac{D(\boldsymbol{x}_q, \boldsymbol{x}_i)^2}{2\mathrm{w}^2}}$$

- Here *w* controls width, *d* is the number of dimensions *D* is distance

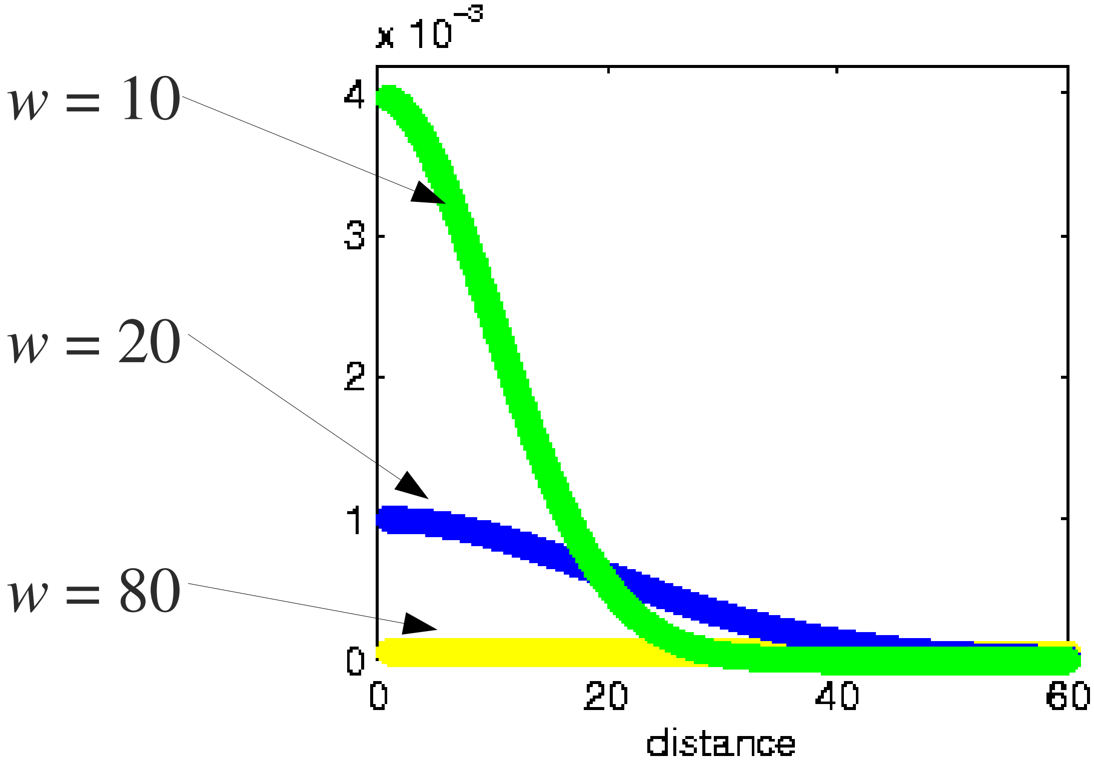
Kernel Regression Example

- Looks better.
- Still a little bumpy.



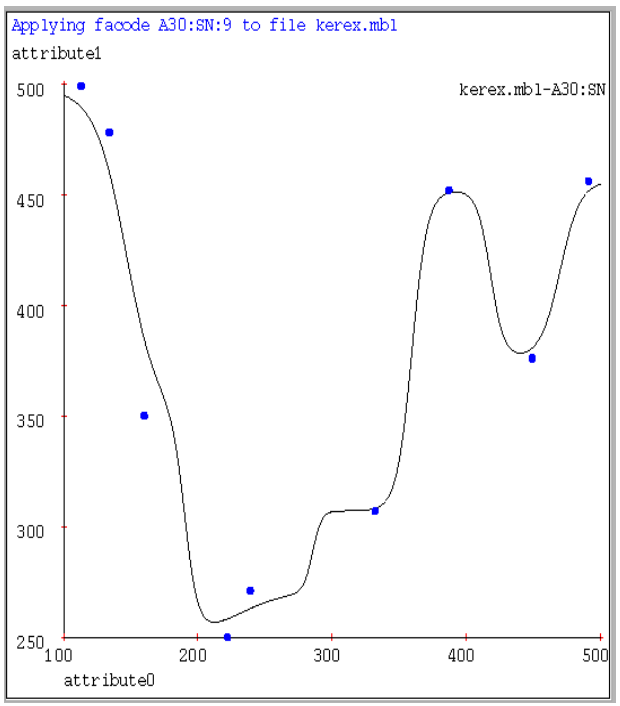
http://www.cs.cmu.edu/~awm/tutorials/

Effect of *w* on the Kernel Function

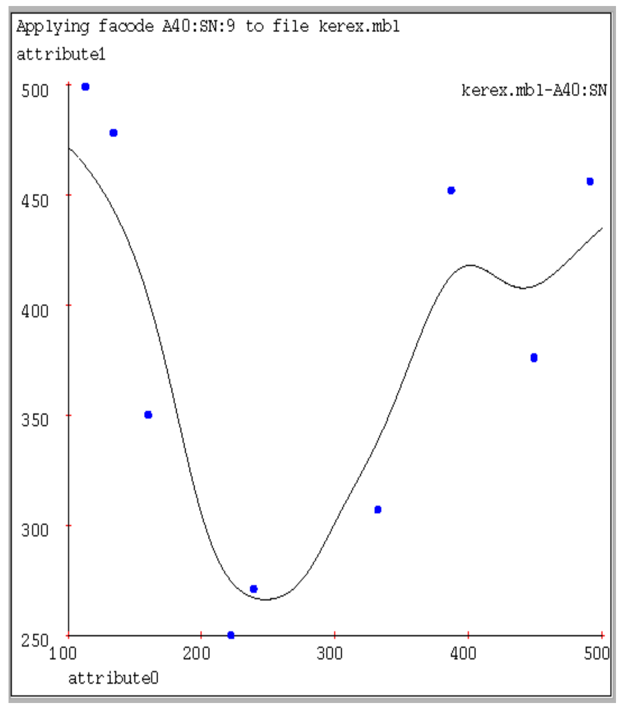


Effect of Changing w on Regression

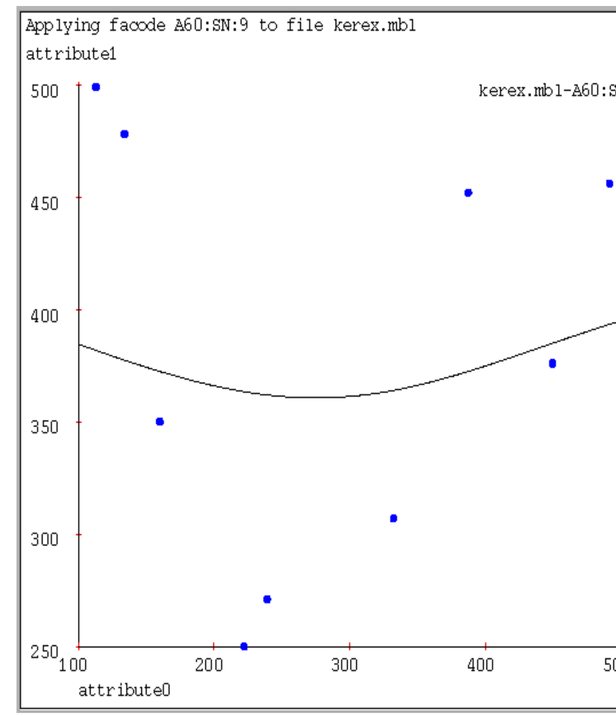
$w = 10$



$w = 20$



$w = 80$

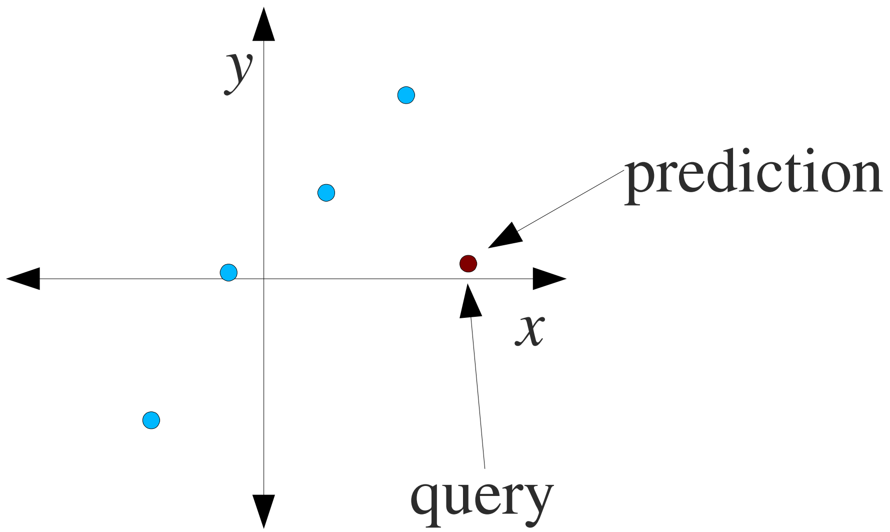


http://www.cs.cmu.edu/~awm/tutorials/

Locally Weighted Regression

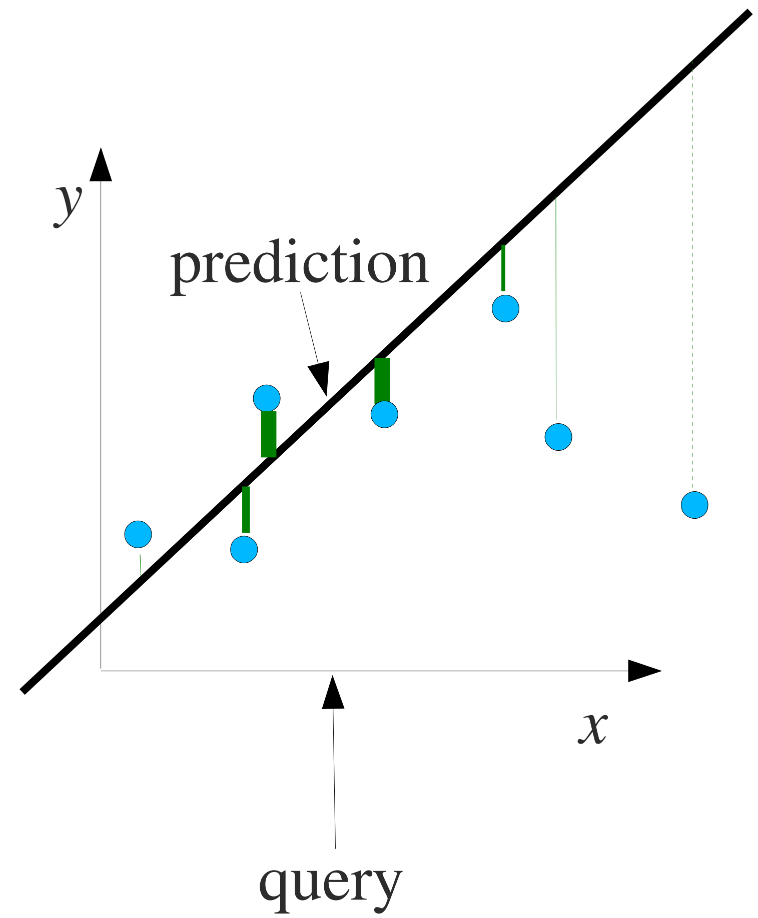
- So far we have been averaging points to come up with a prediction.
- Looks like we are throwing away some useful information.

<u>4-NN example</u>



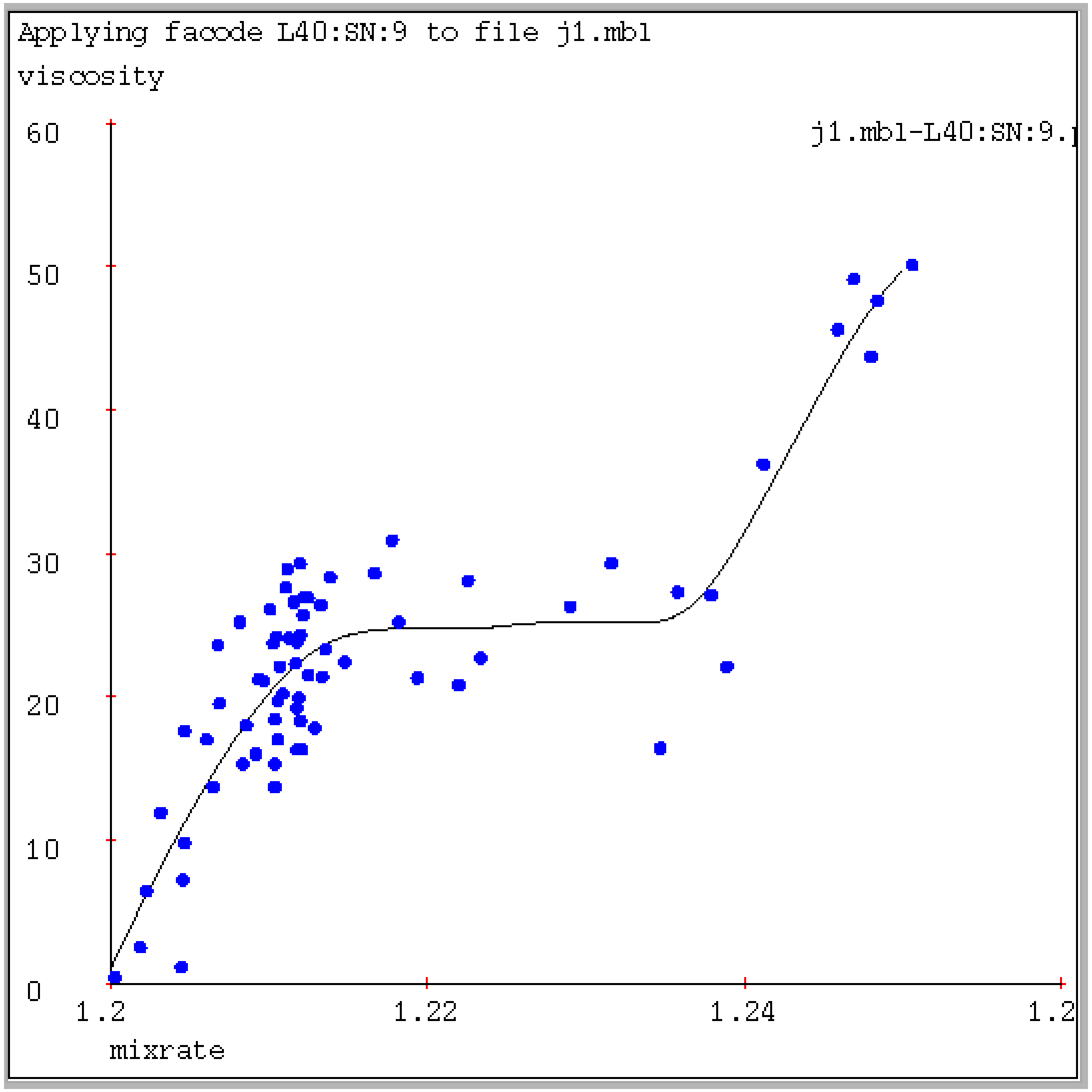
Locally Weighted Linear Regression

- It is easy to find the least squares fit to a line - Not much harder to find a weighted least squares fit.
- Get weights from the kernel function.
- Compute output using linear regression instead of averaging.



LWLR Example

- Looks pretty good!



http://www.cs.cmu.edu/~awm/tutorials/

Instance Based Learning Recipe

- A distance metric
  - So far euclidean
- How many neighbors to look at
  - 1, k, or all.
- A weighting function
  - Gaussian, none, or other.
- How to fit points.
  - Averaging, least squared linear fit, polynomial fit...

Instance Based Classification

- Very easy to extend these techniques to classification.
- K-NN classification:
  - Find the K-NN to the query point.
  - Return the class that has the most votes.
  - Break ties randomly.
- Kernel classification
  - Exactly the same thing, except weight votes by the kernel function.
- Multi-class classification is just as easy as two class.

Difficulties

- The curse of dimensionality – all of this breaks down in high dimensional spaces.
- Distance metric – need to be careful if different dimensions are scaled differently.