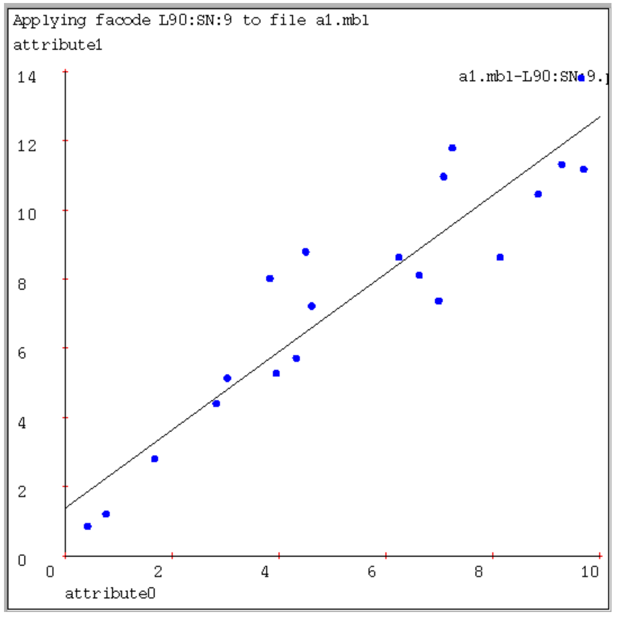
CSE446: Instance-based Learning (a.k.a. non-parametric methods)

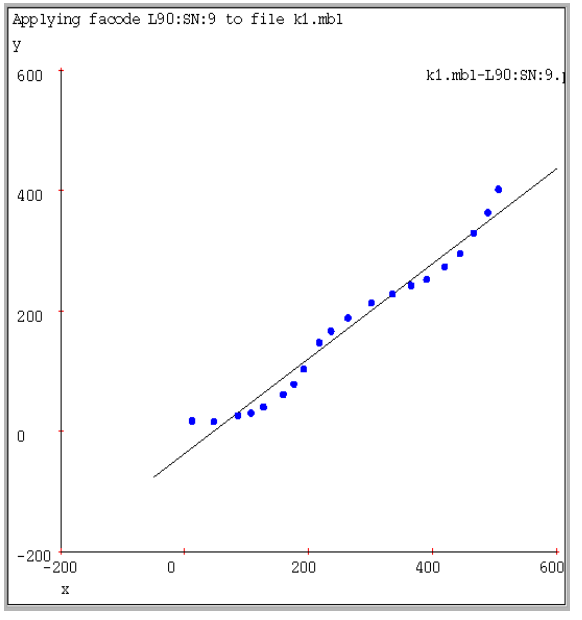
Winter 2015

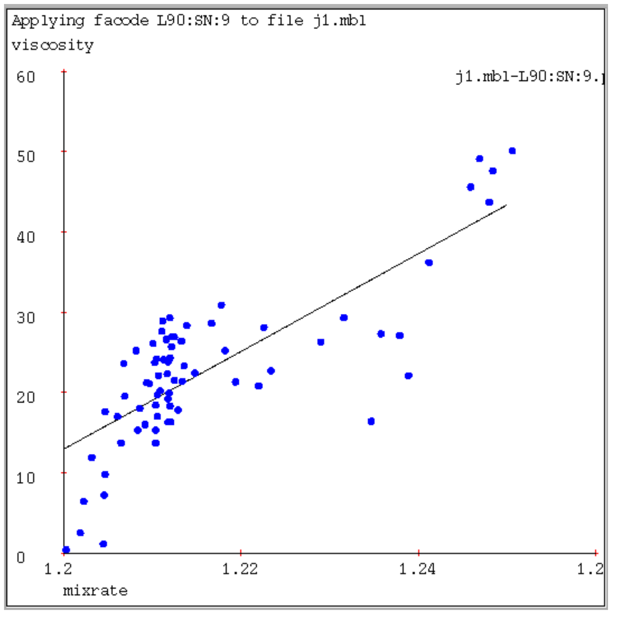
Luke Zettlemoyer

Slides adapted from Carlos Guestrin

Linear Regression: What can go wrong?



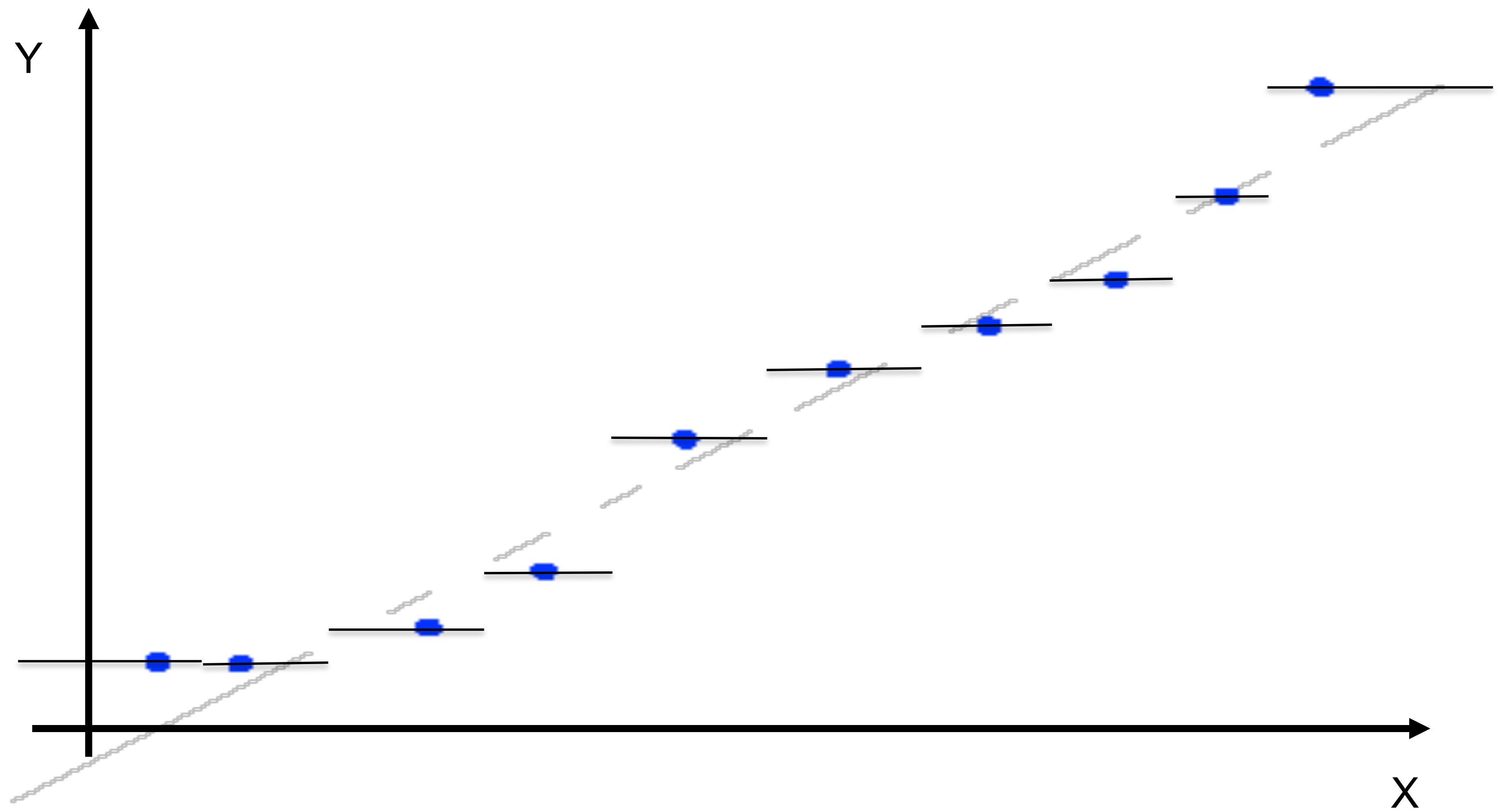




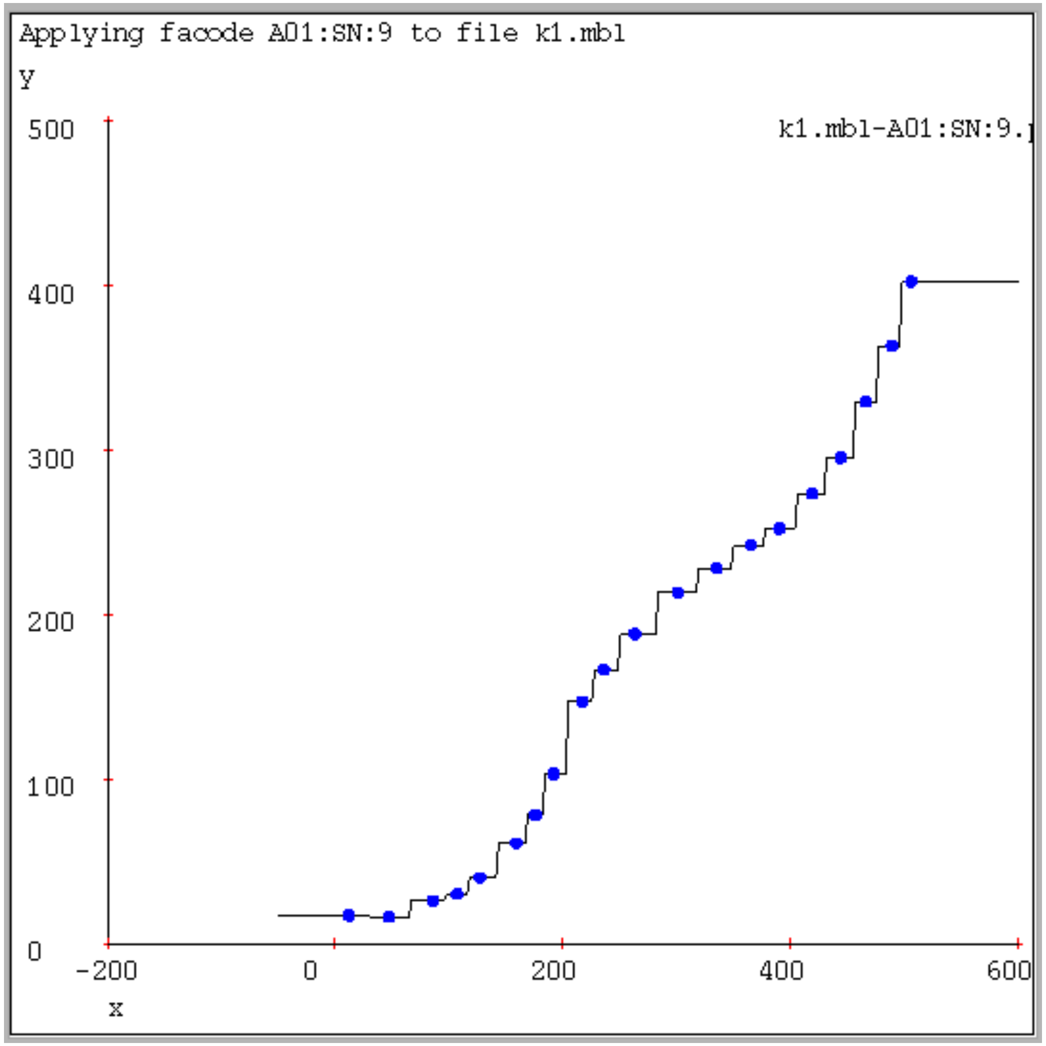
What do we do if the bias is too strong?

- Might want the data to drive the complexity of the model!
- Try instance-based Learning (a.k.a. non-parametric methods)?

Using data to predict new data



Nearest neighbor with lots of data!



Univariate 1-Nearest Neighbor

Given data (x^1,y^1) $(x^2,y^2)..(x^N,y^N)$,where we assume $y=f(x)$ for some unknown function f.

Given query point x, your job is to predict y=f(x)

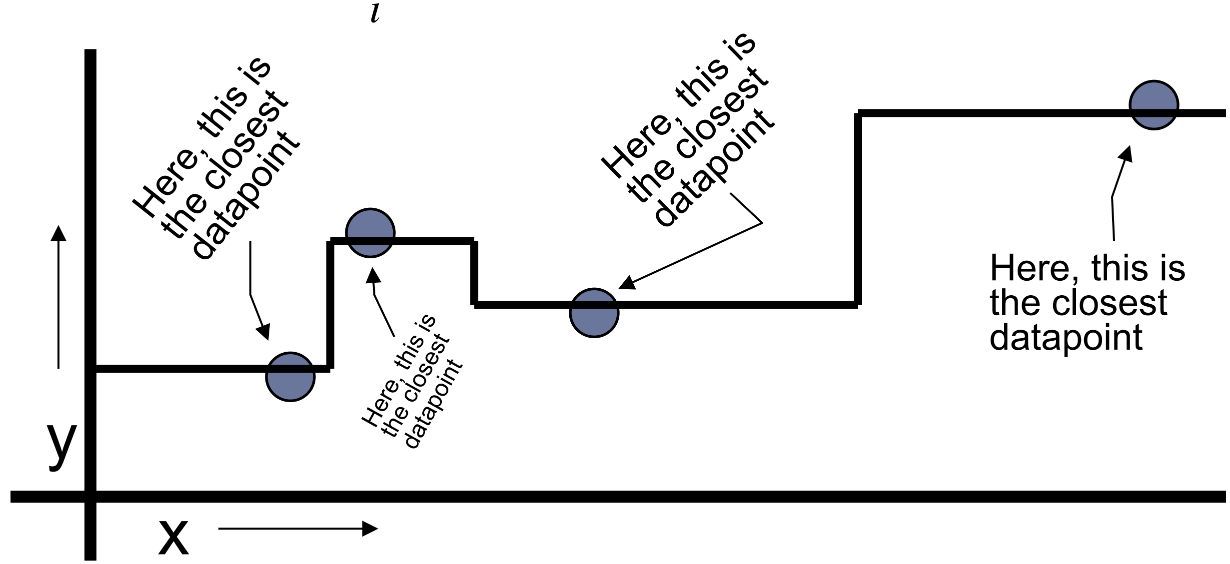
Nearest Neighbor:

1. Find the closest x^i in our set of datapoints

$$i(nn) = \underset{i}{\operatorname{argmin}} \left| x^i - x \right|$$

2. Predict $y^{i(nn)}$

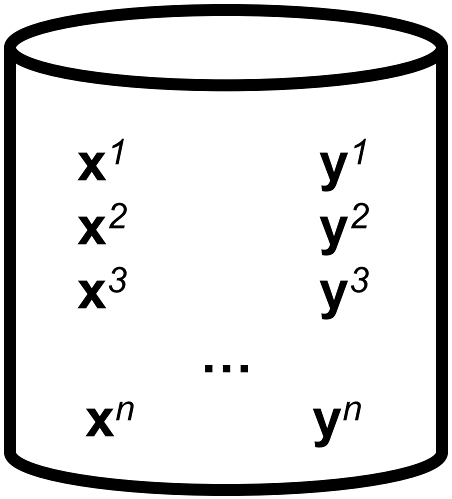
Here's a dataset with one input, one output and four datapoints.



1-Nearest Neighbor is an example of….
Instance-based learning

A function approximator that has been around since about 1910.

To make a prediction, search database for similar datapoints, and fit with the local points.



Instance-based learning, four things to specify:

- A distance metric
- How many nearby neighbors to look at?
- A weighting function (optional)
- How to fit with the local points?

1-Nearest Neighbor

Instance-based learning, four things to specify:

1. *A distance metric*

   Often Euclidian (many more are possible)

2. *How many nearby neighbors to look at?*

   One

3. *A weighting function (optional)*

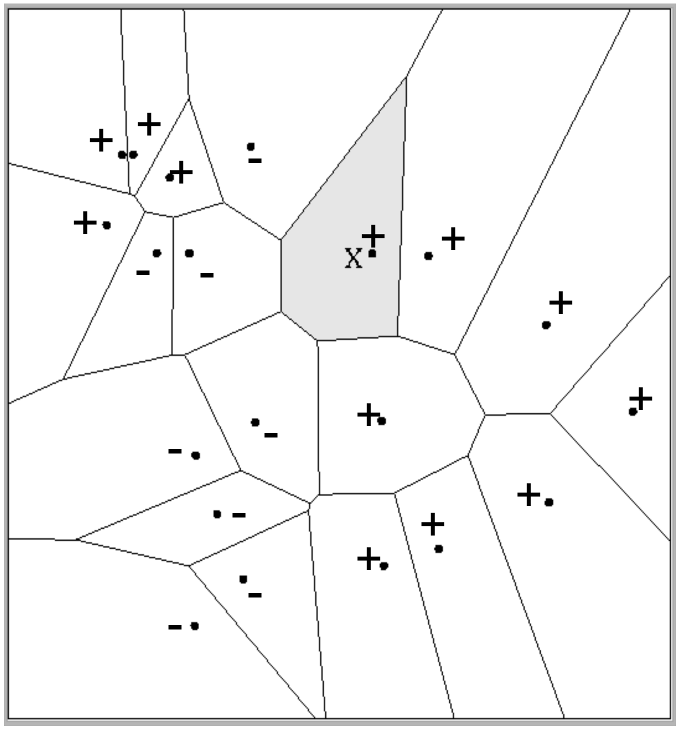
   Unused

4. *How to fit with the local points?*

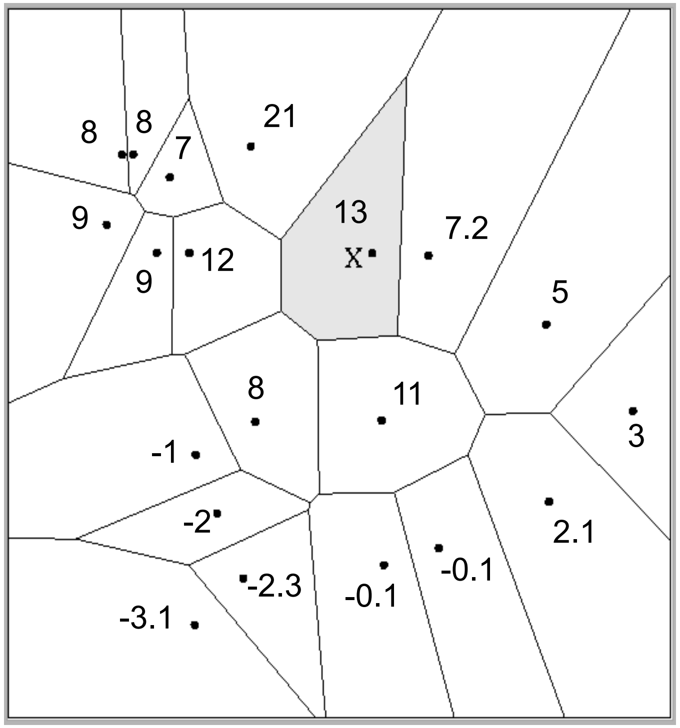
   Just predict the same output as the nearest neighbor.

Multivariate 1-NN examples

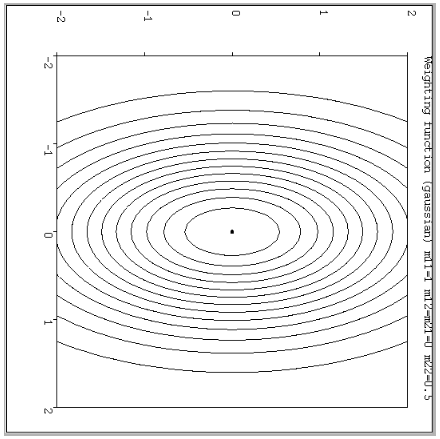
Classification



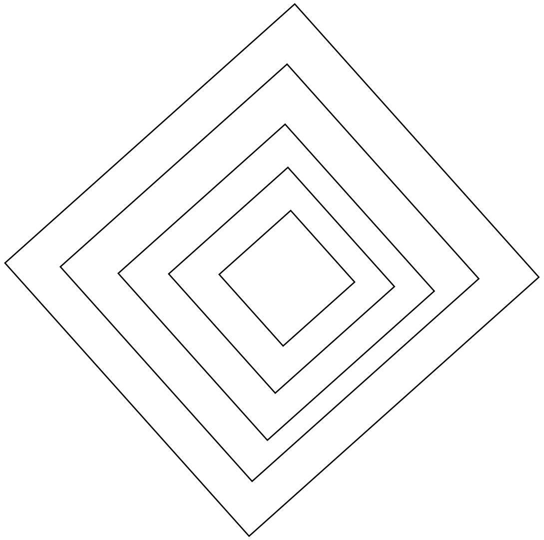
Regression



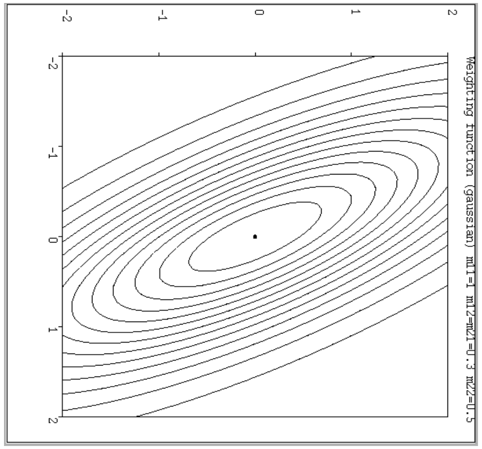
Notable distance metrics (and their level sets)



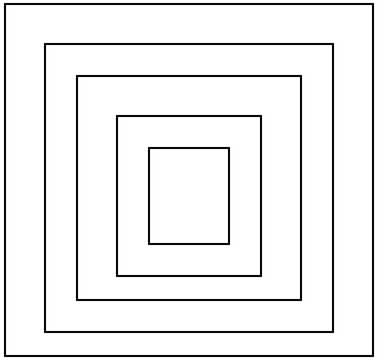
Weighted Euclidian (L_2)



L_1 norm (absolute)



Mahalanobis



L_∞ (max) norm

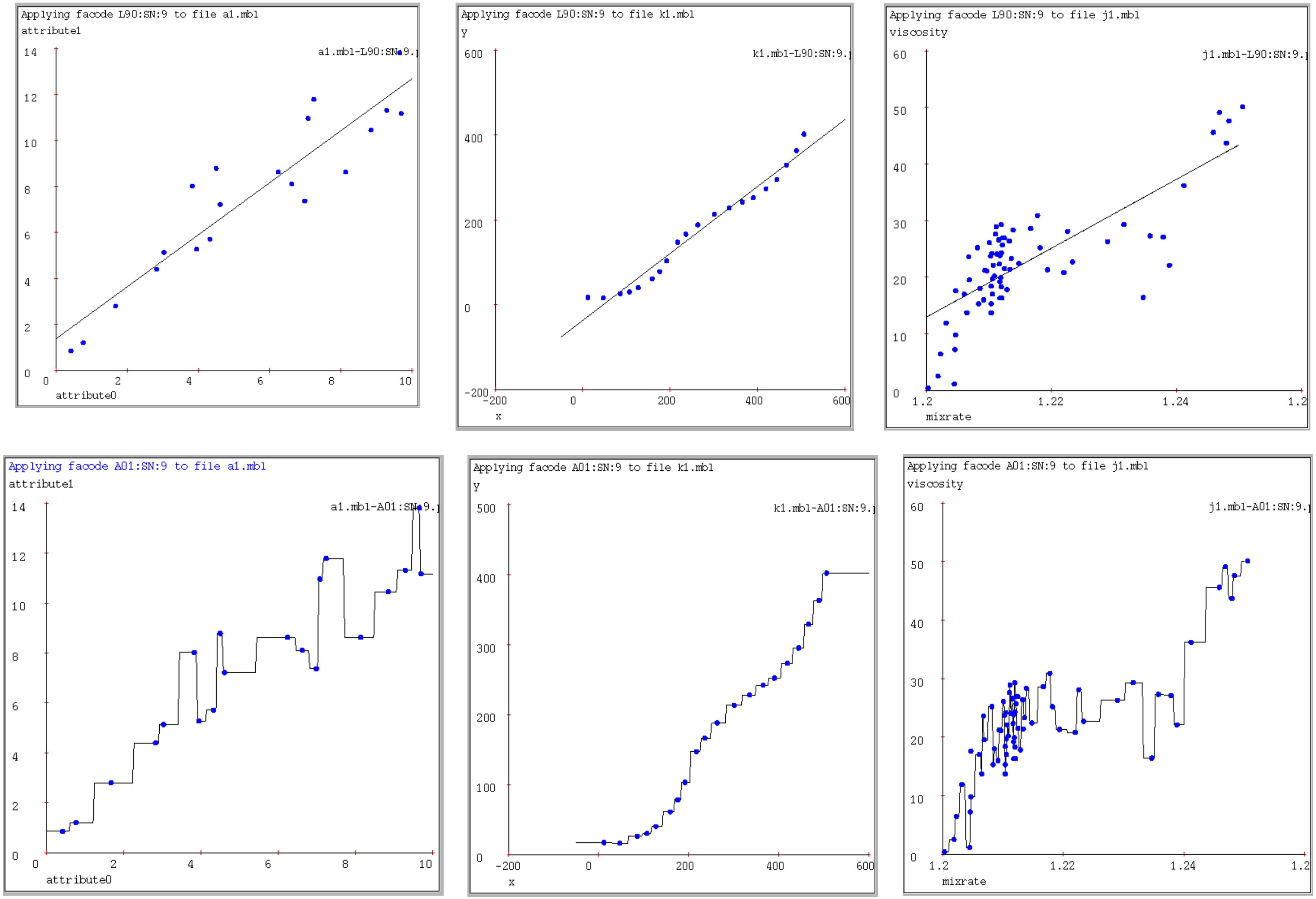
Consistency of 1-NN

- Consider an estimator f_n trained on n examples
  - e.g., 1-NN, neural nets, regression,...
- Estimator is *consistent* if true error goes to zero as amount of data increases
  - e.g., for no noise data, consistent if for any data distribution p(x):

$$\lim_{n \to \infty} MSE(f_n) = 0 \qquad MSE(f_n) = \int_x p(x)\left(f_n(x) - y_x\right)^2 dx$$

- Linear regression is not consistent!
  - Representation bias
- **1-NN is consistent**
  - What about noisy data?
  - What about variance?

1-NN overfits?



k-Nearest Neighbor

Instance-based learning, four things to specify:

1. *A distance metric*

   Euclidian (and many more)

2. *How many nearby neighbors to look at?*

   k

1. *A weighting function (optional)*

   Unused

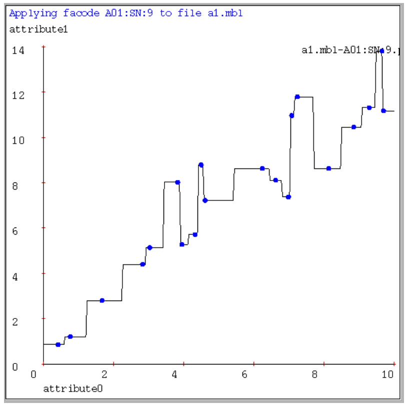
2. *How to fit with the local points?*

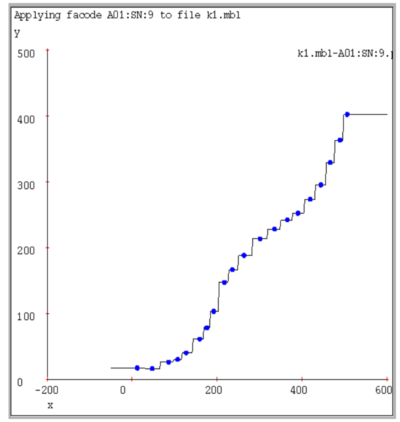
   Return the average output

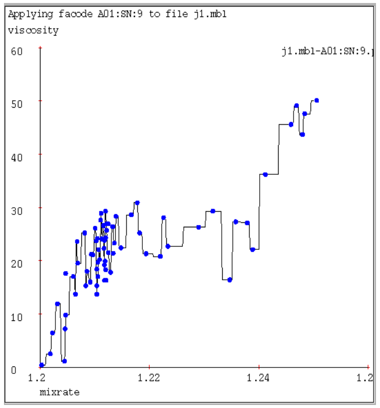
   predict: $(1/k)\ \Sigma_i y^i$ (summing over k nearest neighbors)

k-Nearest Neighbor

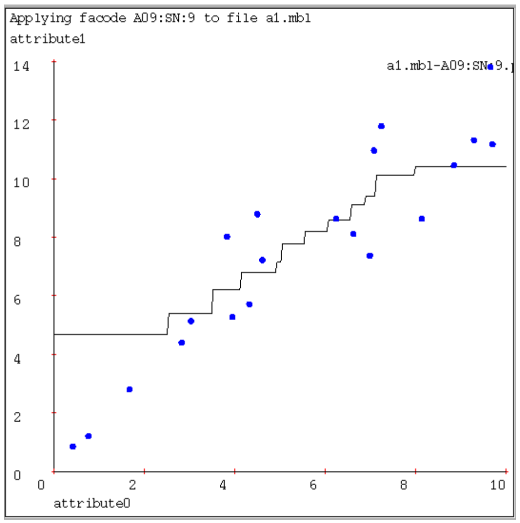
k=1

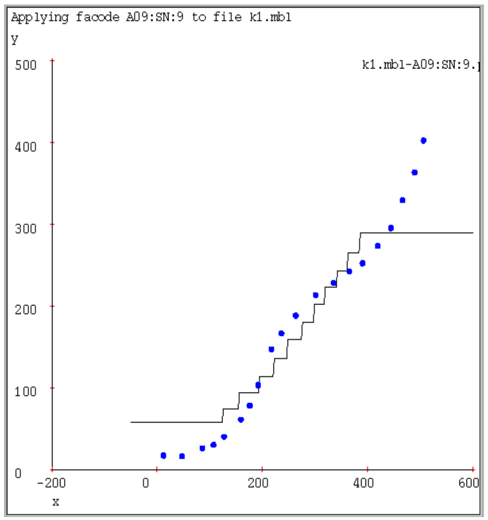


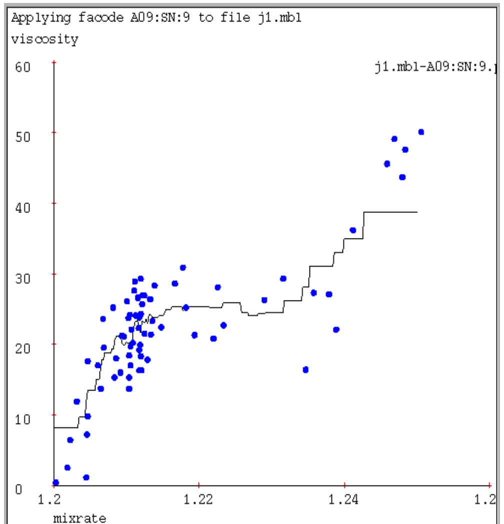




k=9







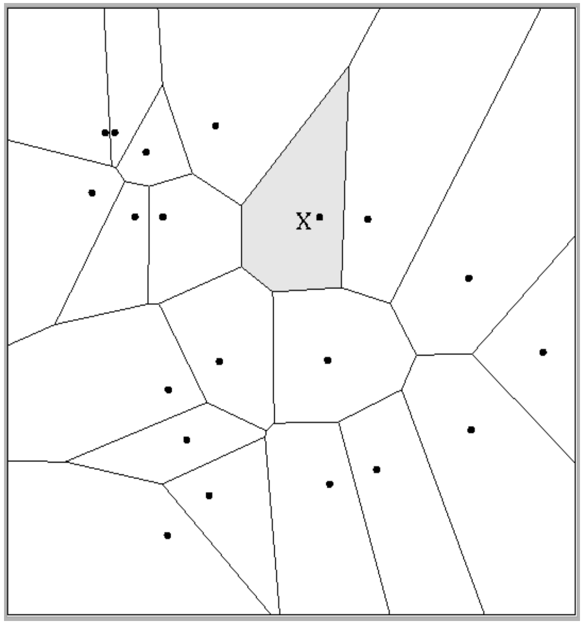
Which is better? What can we do about the discontinuities?

Weighted distance metrics

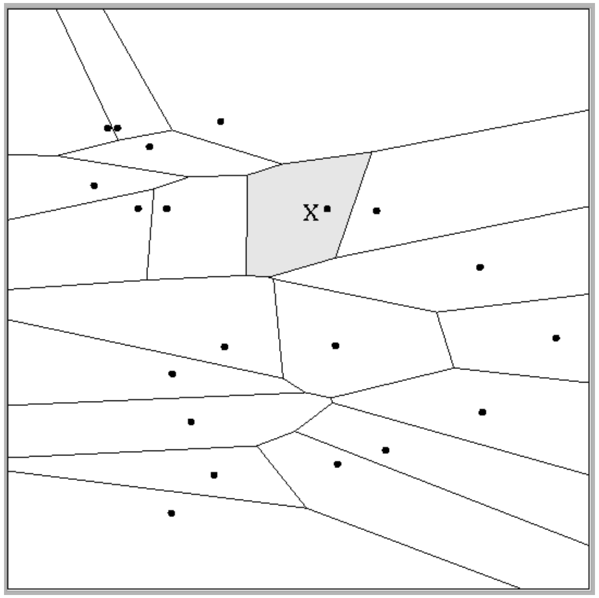
Suppose the input vectors $x^1, x^2, \ldots x^N$ are two dimensional:

$\mathbf{x}^1 = (x^1_1 , x^1_2)$, $\mathbf{x}^2 = (x^2_1 , x^2_2)$, …$\mathbf{x}^N = (x^N_1 , x^N_2)$.

Nearest-neighbor regions in input space:



$Dist(\mathbf{x}^i,\mathbf{x}^j) = (x^i_1 - x^j_1)^2 + (x^i_2 - x^j_2)^2$



$Dist(\mathbf{x}^i,\mathbf{x}^j) =(x^i_1 - x^j_1)^2+(3x^i_2 - 3x^j_2)^2$

The relative scaling of the distance metric affect region shapes

Weighted Euclidean distance metric

Or equivalently,

$$D(\mathrm{x},\mathrm{x}') = \sqrt{\sum_i \sigma_i^2 \left(x_i - x'_i\right)^2}$$

$$D(\mathrm{x},\mathrm{x}') = \sqrt{(\mathrm{x}\text{-}\mathrm{x}')^T \sum (\mathrm{x}\text{-}\mathrm{x}')}$$

where

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & \sigma_N^2 \end{bmatrix}$$

Other Metrics…

- Mahalanobis, Rank-based, Correlation-based,…

Kernel regression

Instance-based learning:

1. *A distance metric*

   Euclidian (and many more)

2. *How many nearby neighbors to look at?*

   All of them

3. *A weighting function*

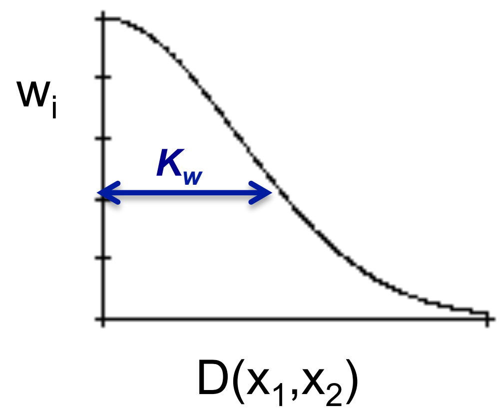
   $w^i = \exp(-D(x^i, query)^2 / K_w^2)$

   Nearby points to the query are weighted strongly, far points weakly. The K_W parameter is the **Kernel Width**. Very important.

4. *How to fit with the local points?*

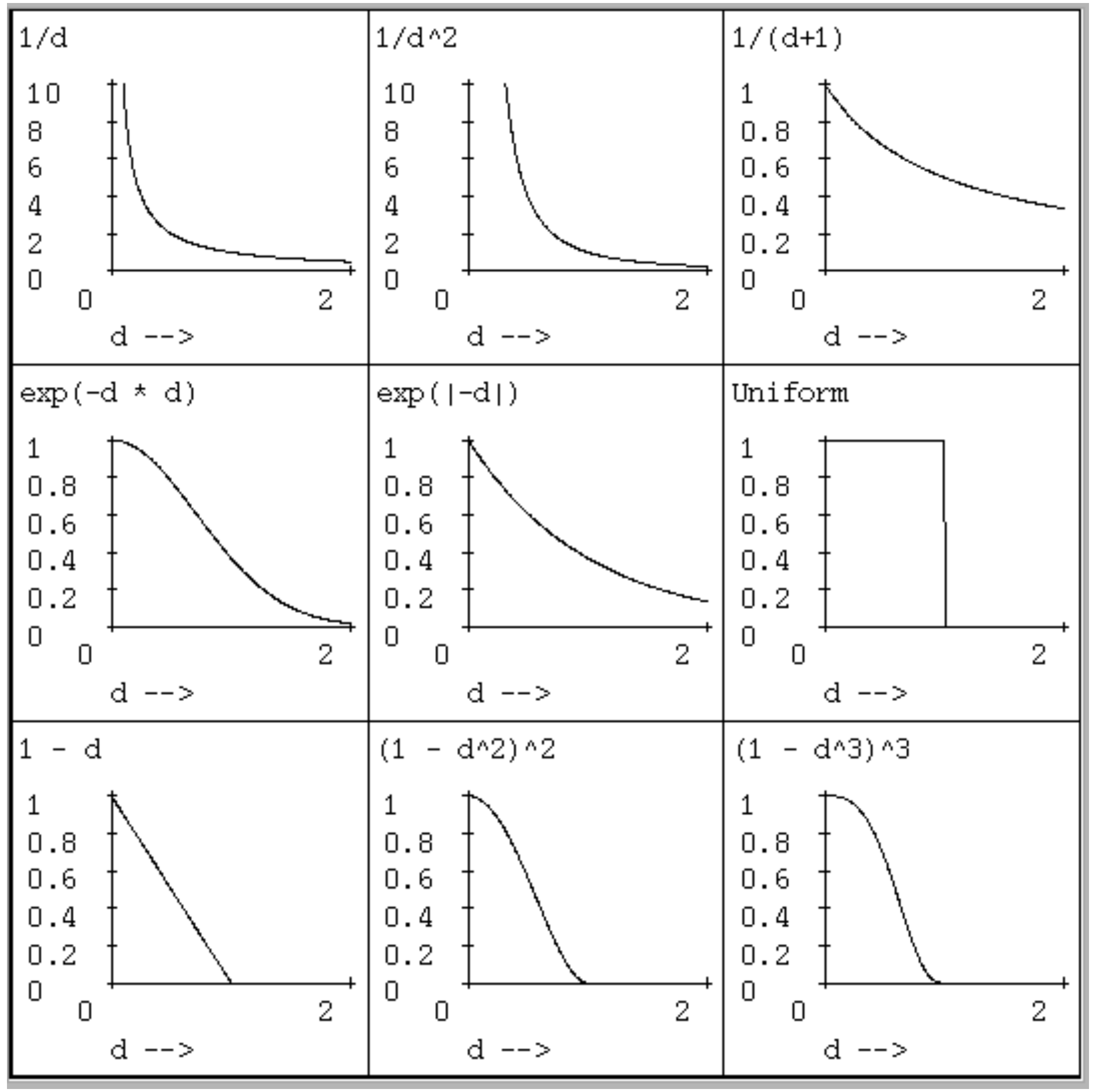
   Predict the weighted average of the outputs:

   $$\textbf{predict} = \Sigma w^i y^i / \Sigma w^i$$



Many possible weighting functions

$w^i = exp(-D(x^i, query)^2 / K_w^2)$

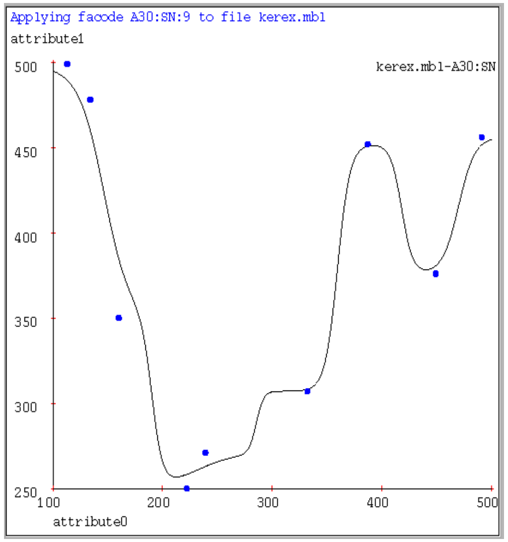


Typically:

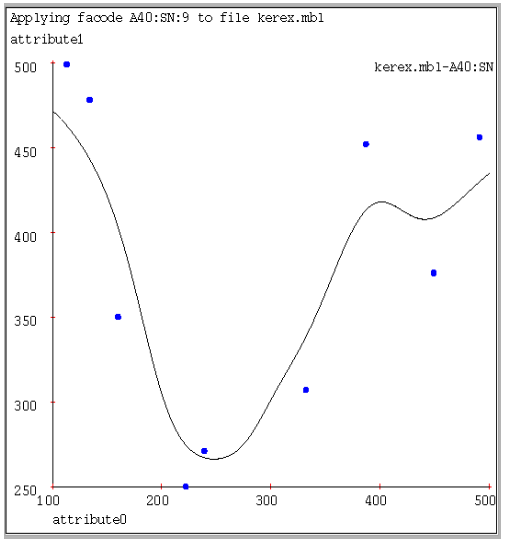
- Choose D manually
- Optimize K_w using gradient descent

(Our examples use Gaussian)

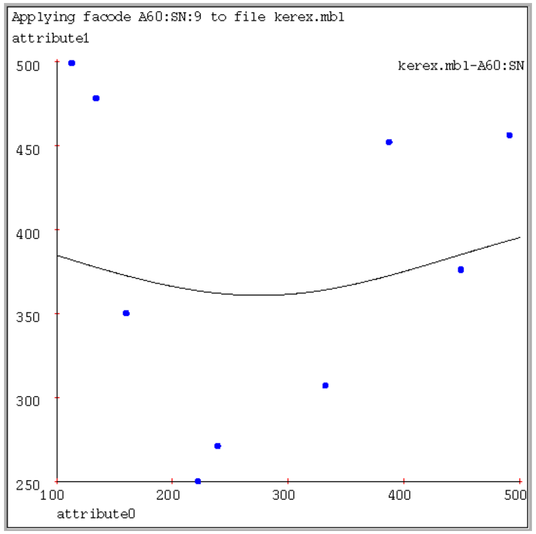
Kernel regression predictions



K_W=10



K_W=20



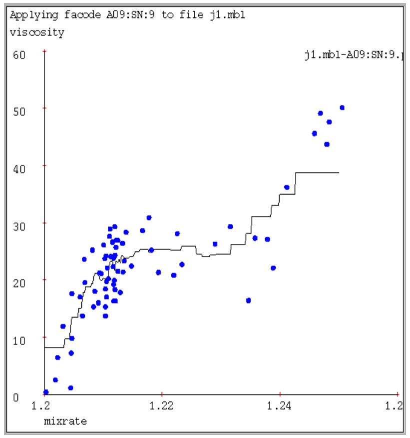
K_W=80

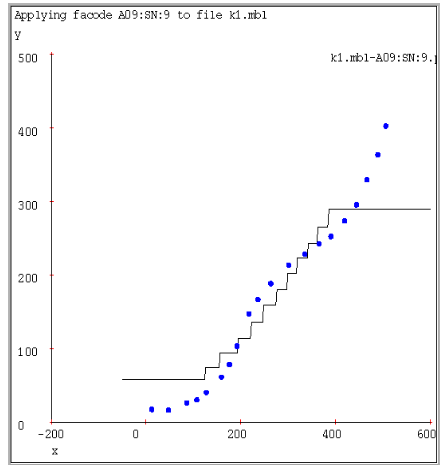
Increasing the kernel width K_w means further away points get an opportunity to influence you.

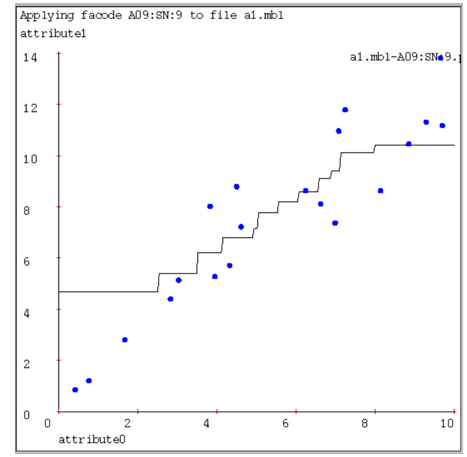
As $K_w \rightarrow \infty$, the prediction tends to the global average.

Kernel regression on our test cases

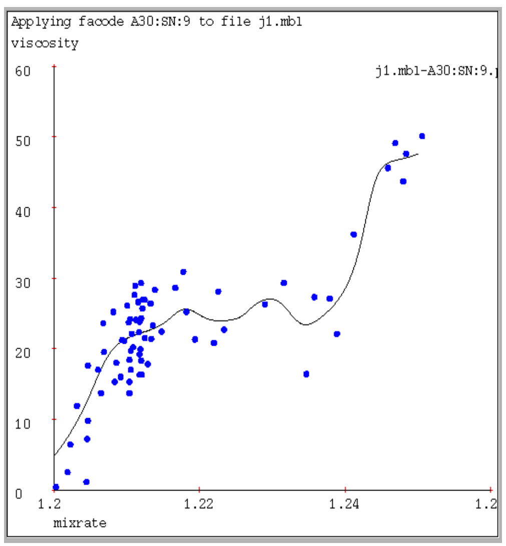
NN k=9



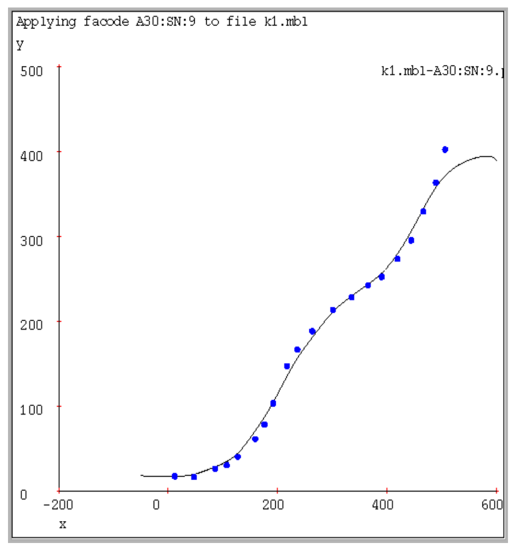




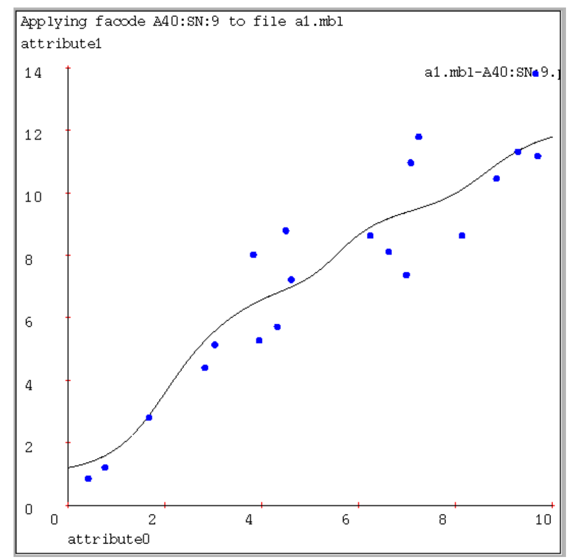
Kernel regression



K_W=1/32 of x-axis width.



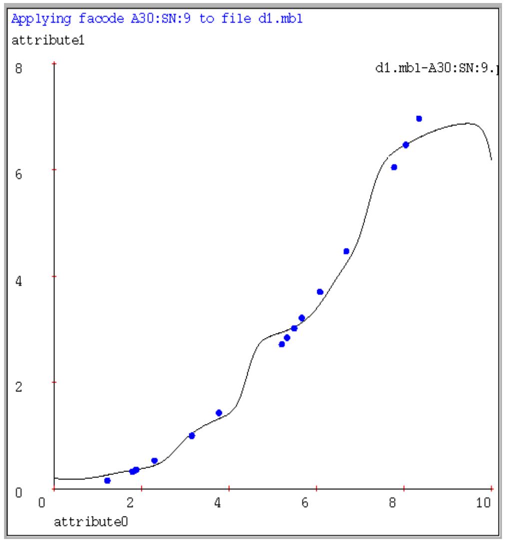
K_W=1/32 of x-axis width.



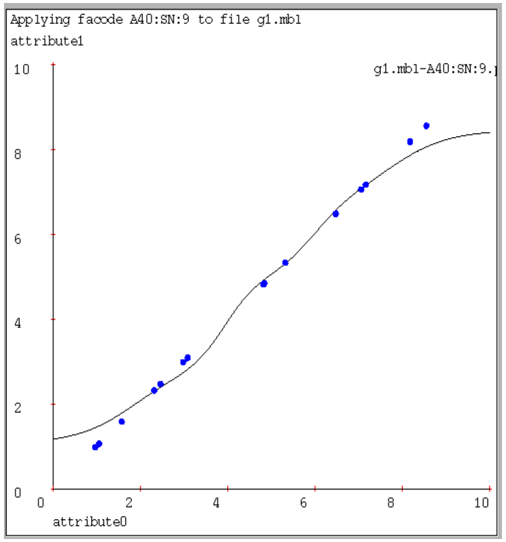
K_W=1/16 axis width.

Choosing a good K_w is important! Remind you of anything we have seen?

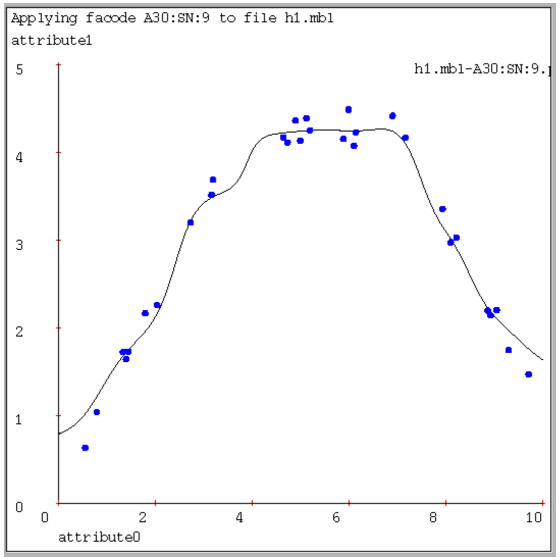
Kernel regression: problem solved?



K_W = Best.



K_W = Best.



K_W = Best.

Where are we having problems?

- Sometimes in the middle…
- Generally, on the ends (extrapolation is hard!)

Time to try something more powerful…!!!

Locally weighted regression

Kernel regression:

- Take a very very conservative function approximator called AVERAGING.
- Locally weight it.

Locally weighted regression:

- Take a conservative function approximator called LINEAR REGRESSION.
- Locally weight it.

Locally weighted regression

Instance-based learning, four things to specify:

- *A distance metric*

  Any

- *How many nearby neighbors to look at?*

  All of them

- *A weighting function (optional)*

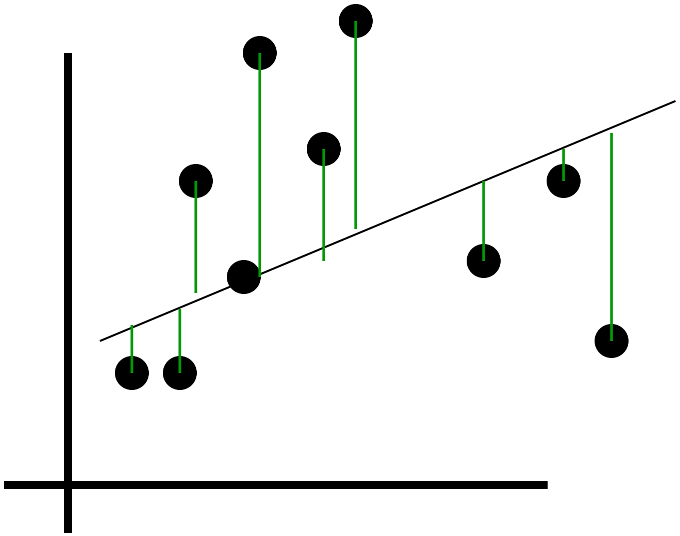
  Kernels: $w^i = exp(-D(xi, query)^2 / Kw^2)$

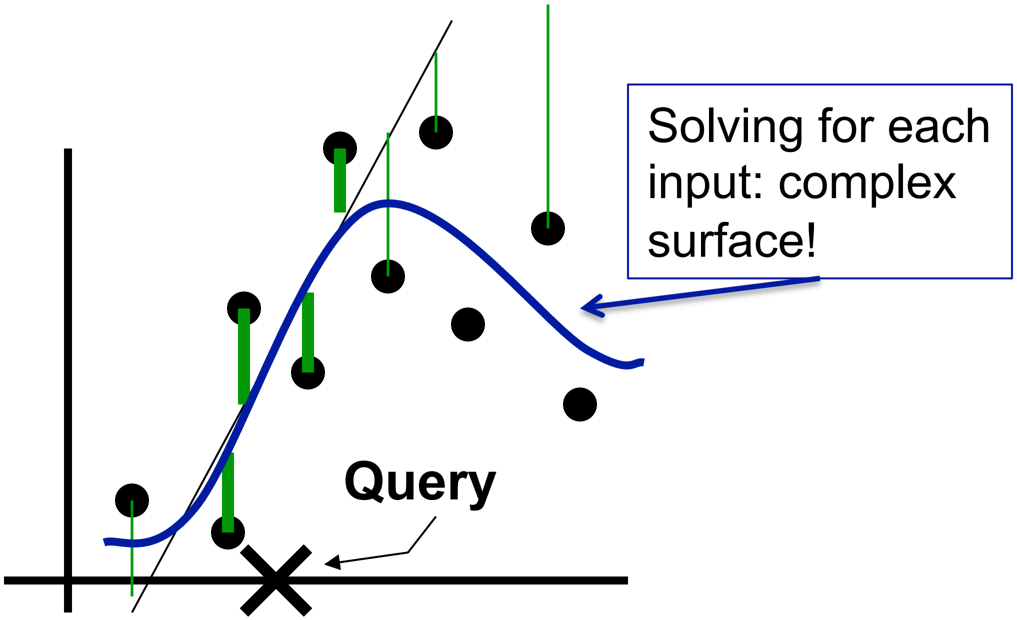
- *How to fit with the local points?*

  General weighted regression:

$$\hat{w} = \underset{w}{\operatorname{argmin}} \sum_{k=1}^{N} \left(w^k \left(y^k - w^T x^k \right) \right)^2$$

How LWR works





Linear regression

- Same parameters for all queries

$$\hat{\beta} = \left(X^T X\right)^{-1} X^T Y$$

Locally weighted regression

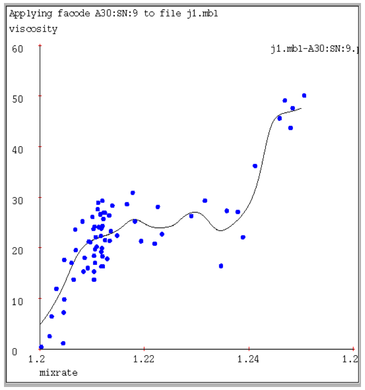
- Solve weighted linear regression for each query

$$\beta = \left((WX)^T WX\right)^{-1} (WX)^T WY$$

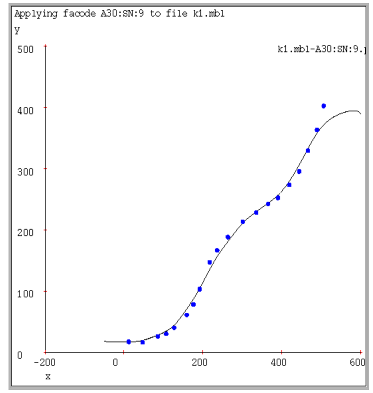
$$W = \begin{pmatrix} w_1 & 0 & 0 & 0 \\ 0 & w_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & w_n \end{pmatrix}$$

LWR on our test cases

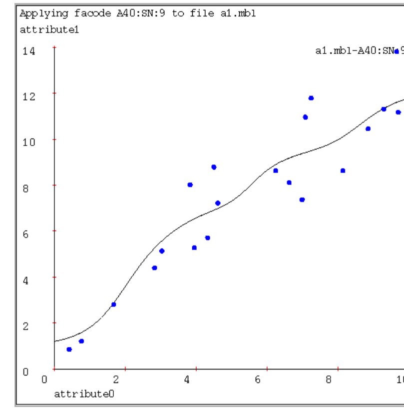
Kernel regression



K_W=1/32 of x-axis width.

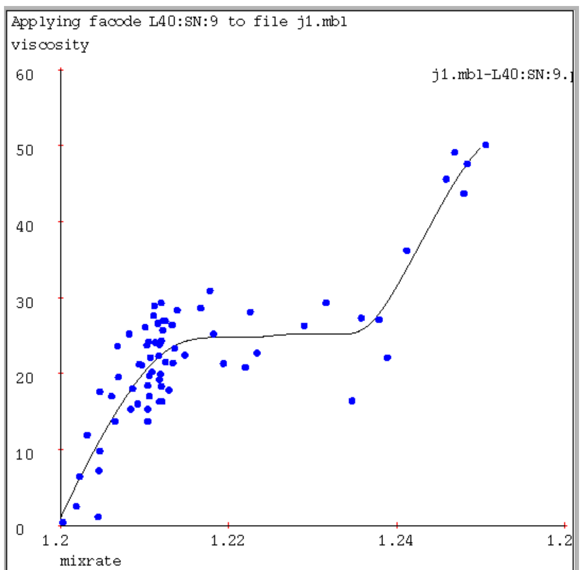


K_W=1/32 of x-axis width.

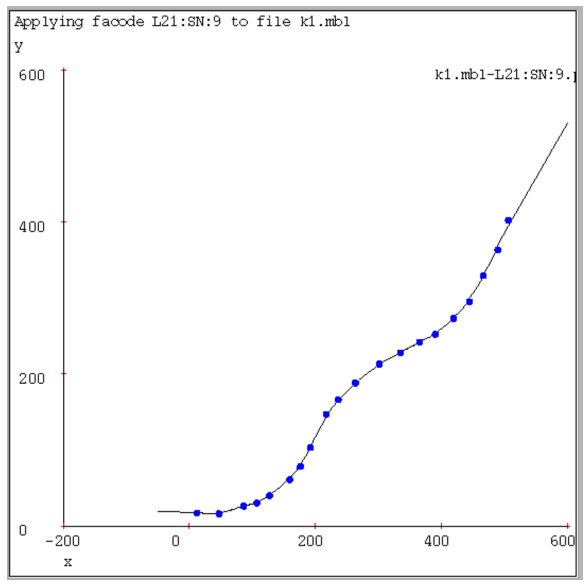


K_W=1/16 axis width.

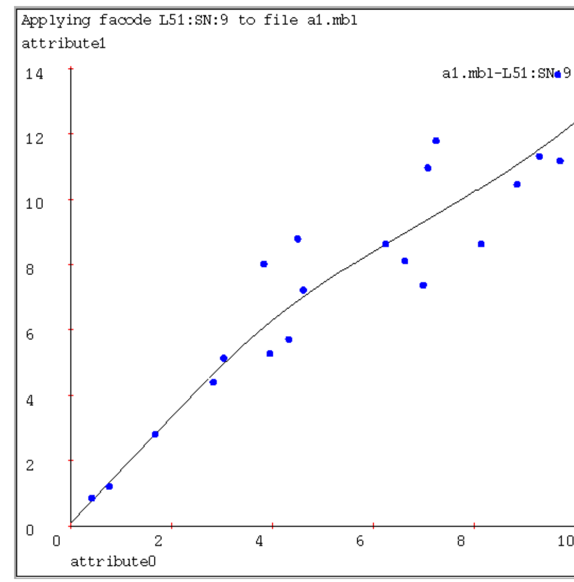
LWR



K_W = 1/16 of x-axis width.



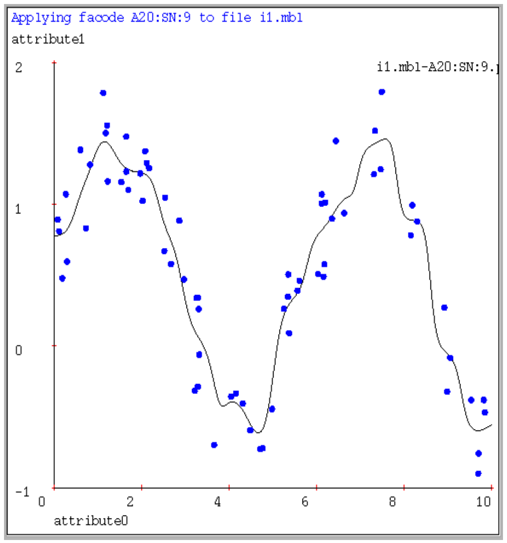
K_W = 1/32 of x-axis width.



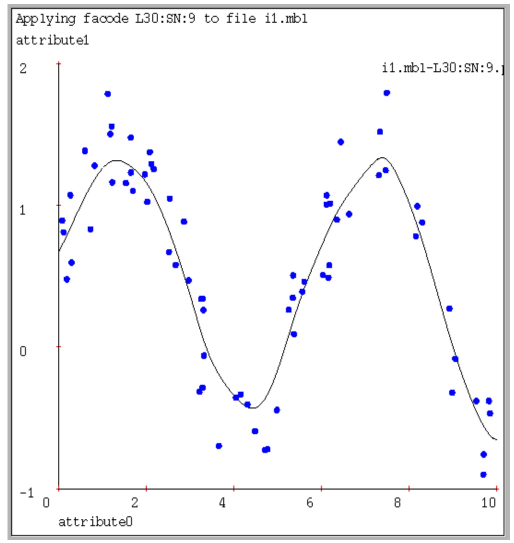
K_W = 1/8 of x-axis width.

Locally weighted polynomial regression

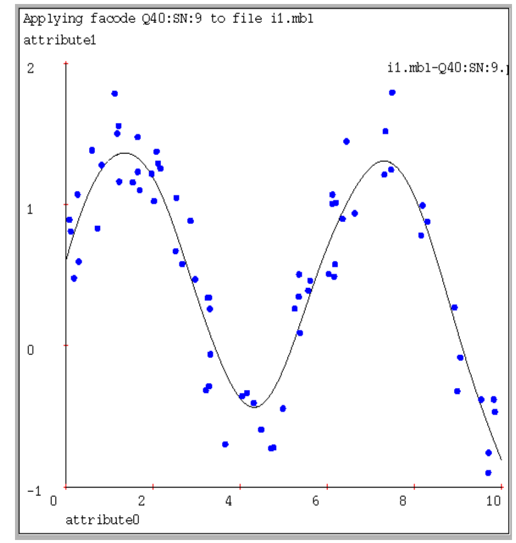
Kernel Regression: Kernel width K_W at optimal level.



K_W = 1/100 x-axis



K_W = 1/40 x-axis



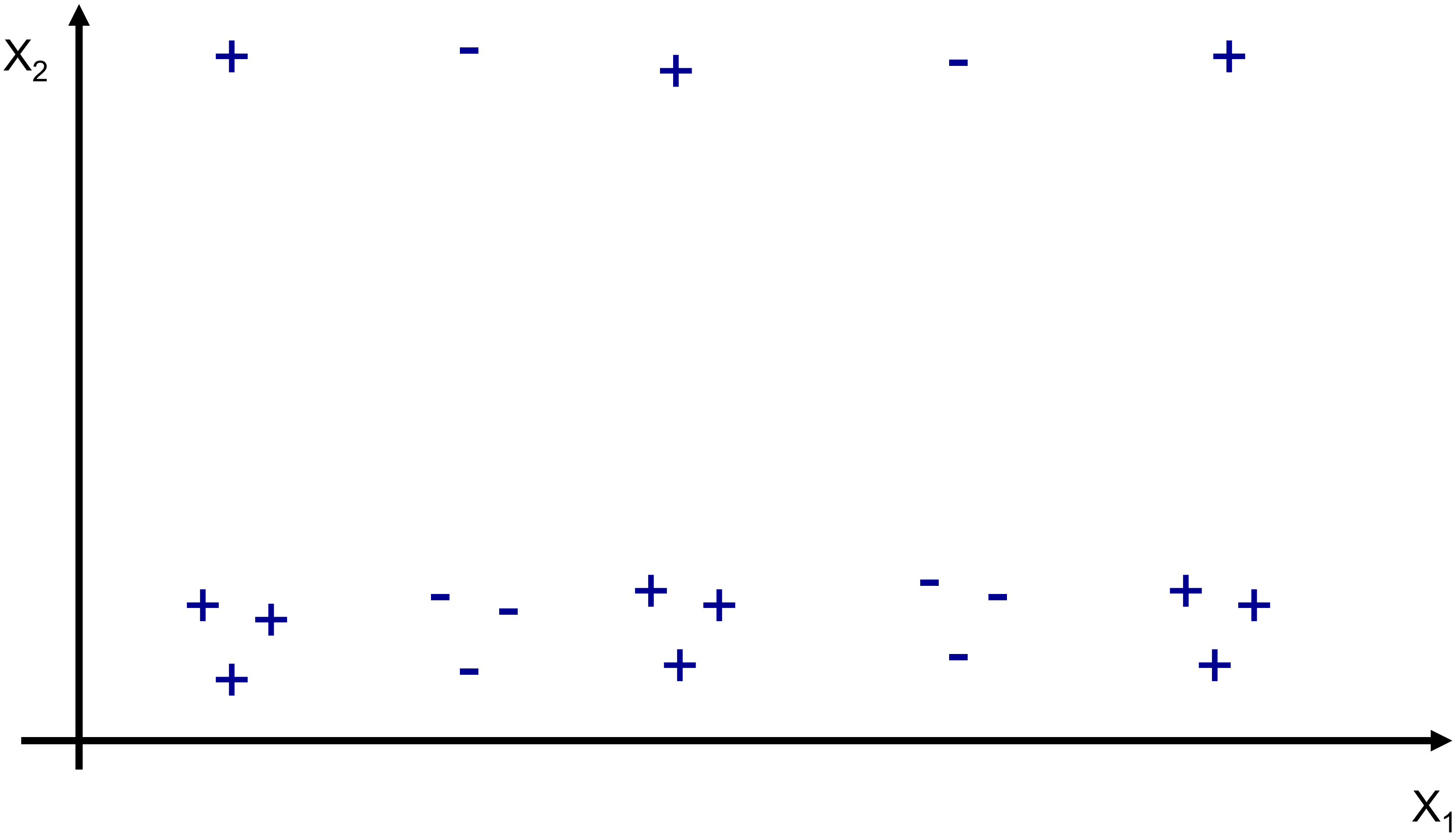
K_W = 1/15 x-axis

Local quadratic regression is easy: just add quadratic terms to the WXTWX matrix. As the regression degree increases, the kernel width can increase without introducing bias.

Challenges for instance based learning

- Must store and retrieve all data!
  - Most real work done during testing
  - For every test sample, must search through all dataset – very slow!
  - But, there are fast methods for dealing with large datasets
- Instance-based learning often poor with noisy or irrelevant features
  - In high dimensional spaces, all points will be very far from each other
  - Typically need a number of examples that is exponential in the dimension of X
  - But, sometimes you are ok if you are cleaver about features

Curse of the irrelevant feature



This is a contrived example, but similar problems are common in practice

Need some form of feature selection!!

What you need to know about instance-based learning

- k-NN
  - Simplest learning algorithm
  - With sufficient data, very hard to beat "strawman" approach
  - Picking k?
- Kernel regression
  - Set k to n (number of data points) and optimize weights by gradient descent
  - Smoother than k-NN
- Locally weighted regression
  - Generalizes kernel regression, not just local average
- Curse of dimensionality
  - Must remember (very large) dataset for prediction
  - Irrelevant features often killers for instance-based approaches

Acknowledgment

- This lecture contains some material from Andrew Moore's excellent collection of ML tutorials:
  - http://www.cs.cmu.edu/~awm/tutorials