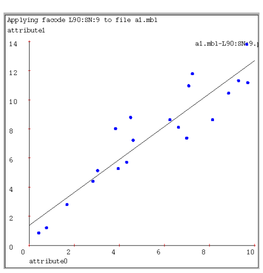
Instance-based Learning

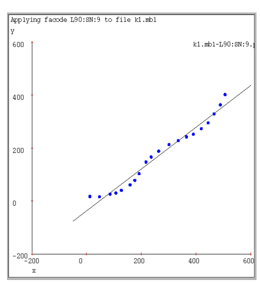
Nearest Neighbors/Non-Parametric Methods

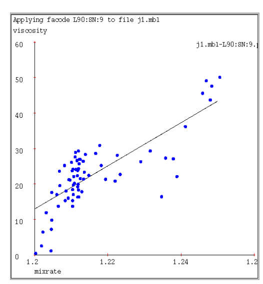
Machine Learning – CSE546
Carlos Guestrin
University of Washington

October 21, 2013

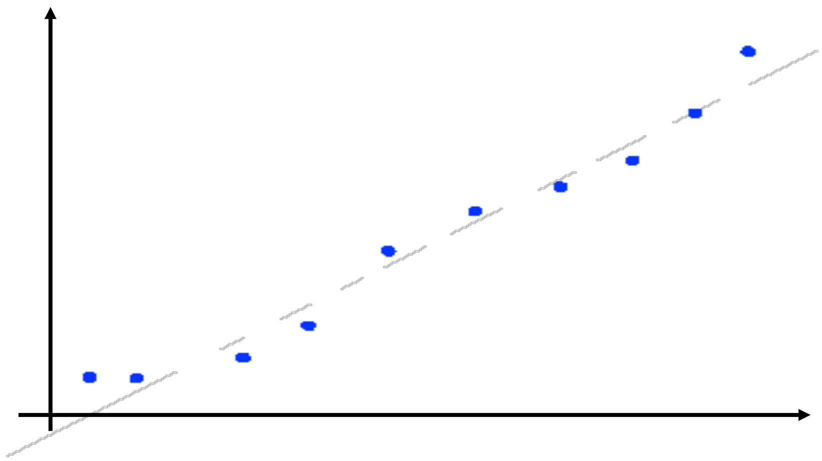
Why not just use Linear Regression?



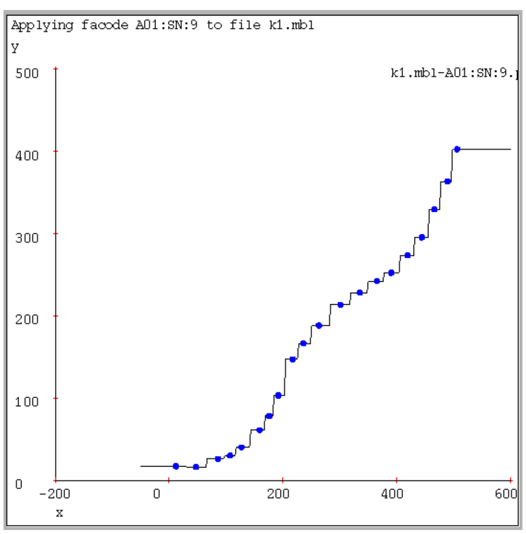




Using data to predict new data



Nearest neighbor



Univariate 1-Nearest Neighbor

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

Given query point x^q, your job is to predict $\hat{y} \approx f\left(x^q\right)$

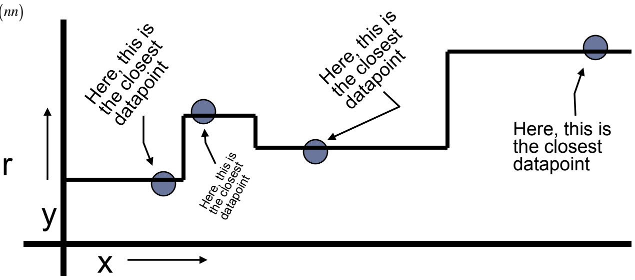
Nearest Neighbor:

1. Find the closest x_i in our set of datapoints

$$j(nn) = \operatorname*{argmin}_j \left|x^j - x^q\right|$$

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

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

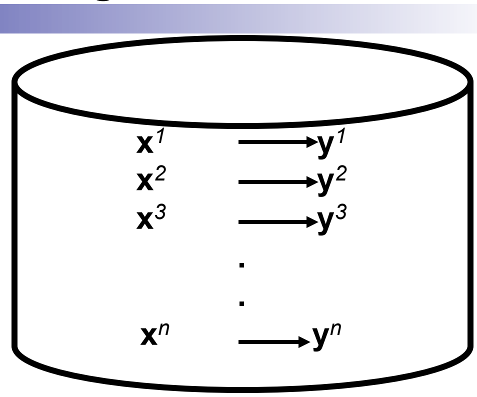


©Carlos Guestrin 2005-2013

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.



Four things make a memory based learner:

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

©Carlos Guestrin 2005-2013

1-Nearest Neighbor

Four things make a memory based learner:

1. *A distance metric*
   Euclidian (and many more)
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.

©Carlos Guestrin 2005-2013

Multivariate 1-NN examples

Classification

Regression

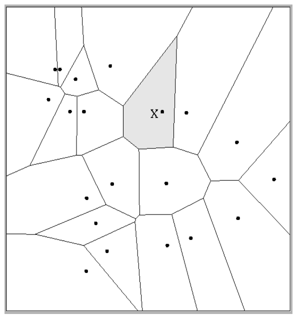
©Carlos Guestrin 2005-2013

Multivariate distance metrics

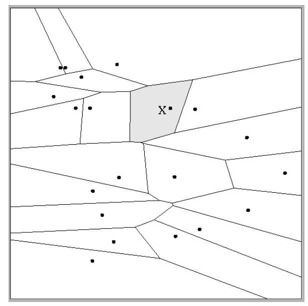
Suppose the input vectors x^1, x^2, …x^N are two dimensional:

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

One can draw the 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 scalings in the distance metric affect region shapes

©Carlos Guestrin 2005-2013

Euclidean distance metric

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

Or equivalently,

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

where

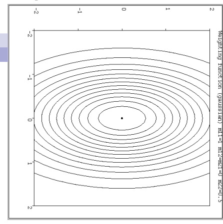
$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & \sigma_N^2 \end{bmatrix}$$

Other Metrics…

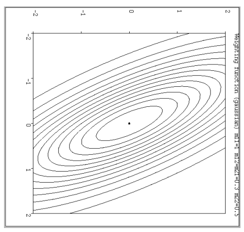
- Mahalanobis, Rank-based, Correlation-based,…

©Carlos Guestrin 2005-2013

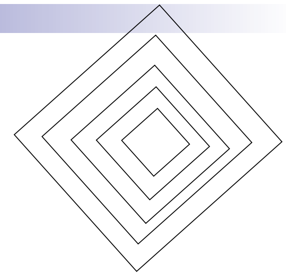
Notable distance metrics (and their level sets)



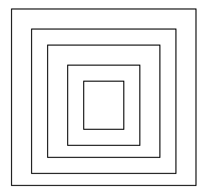
Scaled Euclidian (L_2)



Mahalanobis (here, Σ on the previous slide is not necessarily diagonal, but is symmetric



L_1 norm (absolute)



L1 *(max)* norm

©Carlos Guestrin 2005-2013

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:

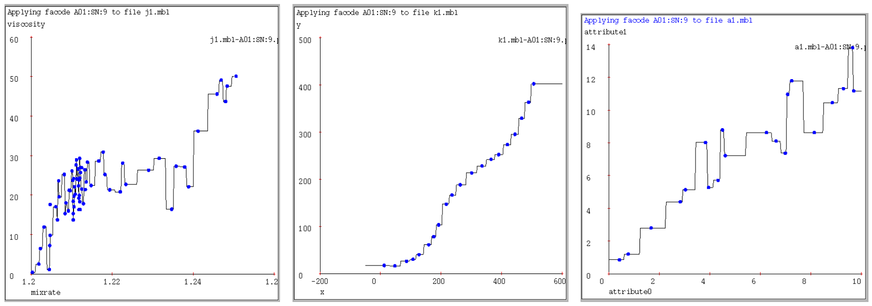
$$\lim_{n \rightarrow \infty} MSE(f_n) = 0$$

- Regression is not consistent!
  - Representation bias
- **1-NN is consistent** (under some mild fineprint)

What about variance???

©Carlos Guestrin 2005-2013

1-NN overfits?

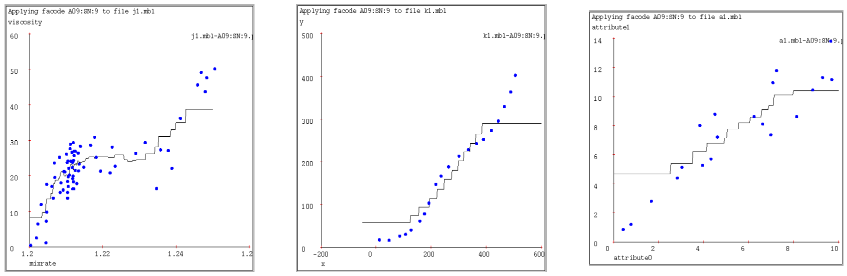


k-Nearest Neighbor

Four things make a memory based learner:

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?*
   Just predict the average output among the k nearest neighbors.

k-Nearest Neighbor (here k=9)



K-nearest neighbor for function fitting smoothes away noise, but there are clear deficiencies.

What can we do about all the discontinuities that k-NN gives us?

Weighted k-NNs

- Neighbors are not all the same

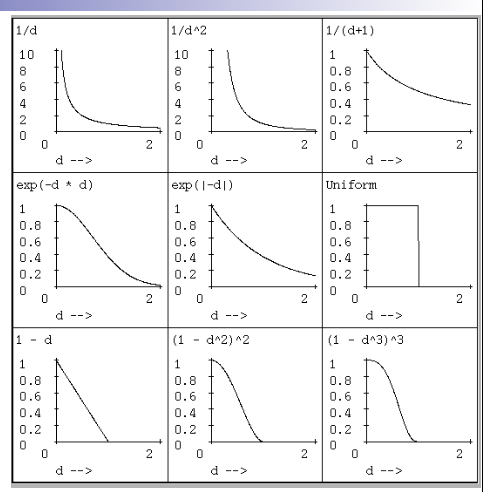
Kernel regression

Four things make a memory based learner:

1. *A distance metric*
   Euclidian (and many more)
2. *How many nearby neighbors to look at?*
   All of them
3. *A weighting function (optional)*
   $\pi^i = exp(-D(x^i, query)^2 / \rho^2)$
   Nearby points to the query are weighted strongly, far points weakly. The ρ parameter is the **Kernel Width**. Very important.
4. *How to fit with the local points?*
   Predict the weighted average of the outputs:
   predict = $\Sigma\pi^i y^i / \Sigma\pi^i$

Weighting functions

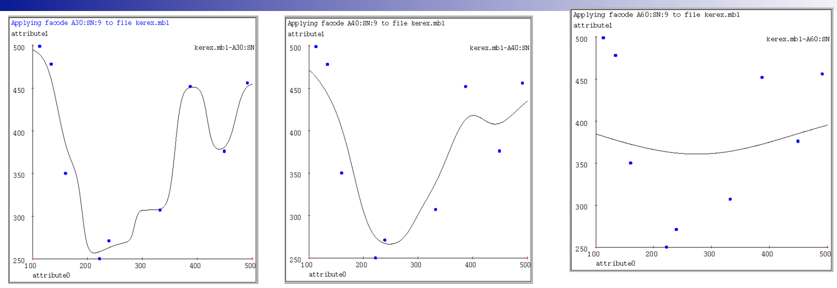
$\pi^i = exp(-D(x^i, query)^2 / \rho^2)$



(Our examples use Gaussian)

Typically optimize ρ using gradient descent

Kernel regression predictions



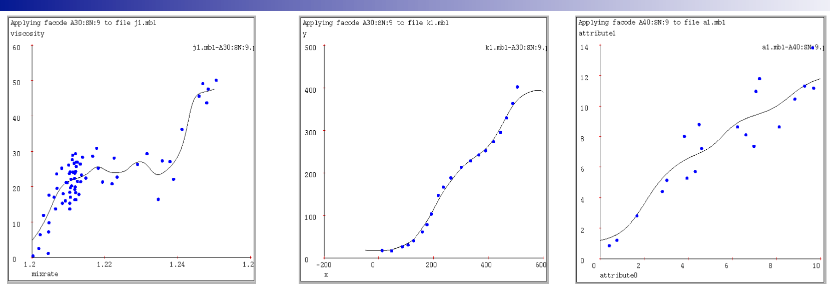
ρ=10 ρ=20 ρ=80

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

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

©Carlos Guestrin 2005-2013

Kernel regression on our test cases

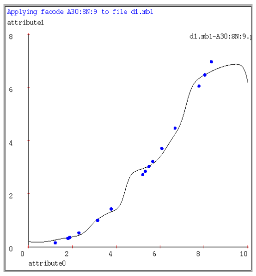


ρ=1/32 of x-axis width. ρ=1/32 of x-axis width. ρ=1/16 axis width.

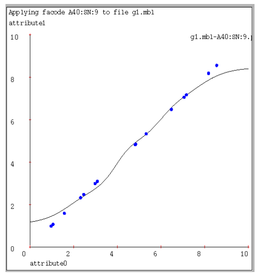
Choosing a good ρ is important. Not just for Kernel Regression, but for all the locally weighted learners we're about to see.

©Carlos Guestrin 2005-2013

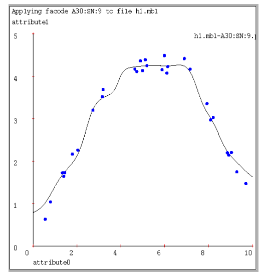
Kernel regression can look bad



ρ = Best.



ρ = Best.



ρ = Best.

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:

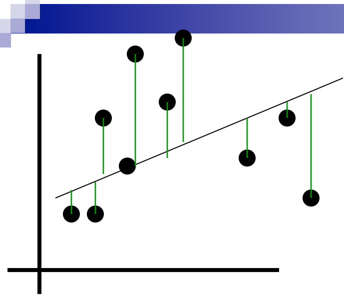
Take a conservative function approximator called LINEAR REGRESSION. Locally weight it.

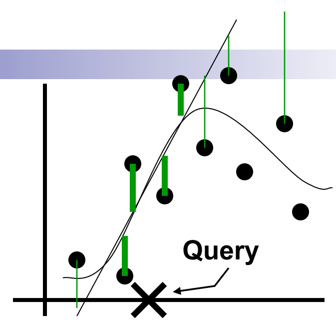
Locally weighted regression

- **Four things make a memory based learner:**
- *A distance metric*
  - **Any**
- *How many nearby neighbors to look at?*
  - **All of them**
- *A weighting function (optional)*
  - **Kernels**
    - ***$\pi^i = \exp(-D(x^i, query)^2 / \rho^2)$***
- *How to fit with the local points?*
  - **General weighted regression:**

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

How LWR works





Linear regression
- Same parameters for all queries

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

Locally weighted regression
- Solve weighted linear regression for each query

$$w^q = \left((\Pi X)^T \Pi X\right)^{-1} (\Pi X)^T \Pi Y$$

$$\Pi = \begin{pmatrix} \pi_1 & 0 & 0 & 0 \\ 0 & \pi_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \pi_n \end{pmatrix}$$

Another view of LWR

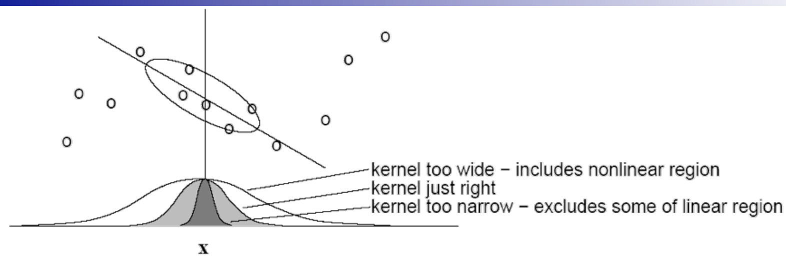
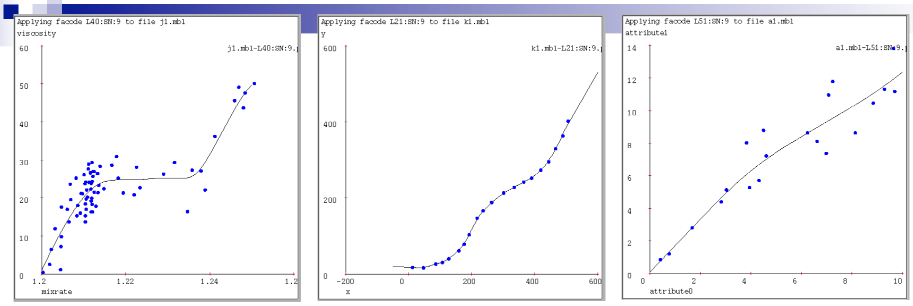


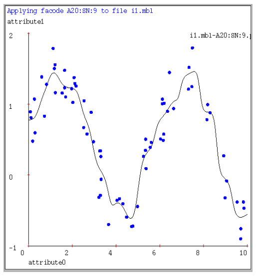
Image from Cohn, D.A., Ghahramani, Z., and Jordan, M.I. (1996) "Active Learning with Statistical Models", JAIR Volume 4, pages 129-145.

LWR on our test cases



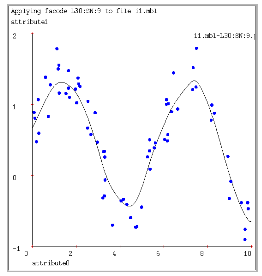
ρ = 1/16 of x-axis width. ρ = 1/32 of x-axis width. ρ = 1/8 of x-axis width.

Locally weighted polynomial regression



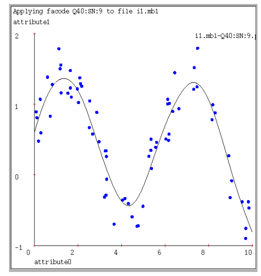
Kernel Regression
Kernel width ρ at optimal level.

ρ = 1/100 x-axis



LW Linear Regression
Kernel width ρ at optimal level.

ρ = 1/40 x-axis



LW Quadratic Regression
Kernel width ρ at optimal level.

ρ = 1/15 x-axis

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

©Carlos Guestrin 2005-2013

Curse of dimensionality for instance-based learning

- Must store and retreve all data!
  - Most real work done during testing
  - For every test sample, must search through all dataset – very slow!
  - There are (sometimes) fast methods for dealing with large datasets
- Instance-based learning often poor with noisy or irrelevant features

©Carlos Guestrin 2005-2013

Curse of the irrelevant feature

©Carlos Guestrin 2005-2013

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

©Carlos Guestrin 2005-2013

Acknowledgment

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

©Carlos Guestrin 2005-2013