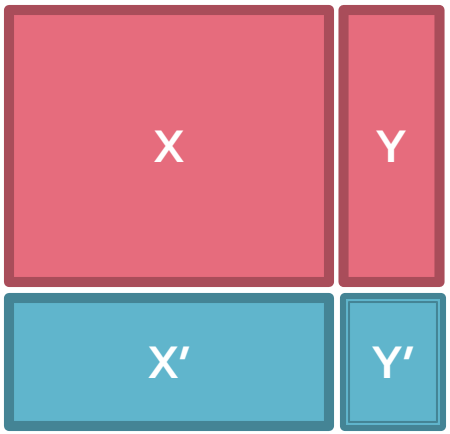
Large Scale Machine Learning: Nearest Neighbors

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University

Supervised Learning

- **Would like to do prediction:**
  estimate a function **f(x)** so that ***y = f(x)***
- **Where *y* can be:**
  - **Real number:** Regression
  - **Categorical:** Classification
  - Complex object:
    - Ranking of items, Parse tree, etc.
- **Data is labeled:**
  - Have many pairs **{(x, y)}**
    - **x** … vector of binary, categorical, real valued features
    - **y** … class ({+1, -1}, or a real number)



Training and test set

Estimate *y = f(x)* on *X, Y*.
Hope that the same *f(x)*
also works on unseen X', Y'

Large Scale Machine Learning

- **We will talk about the following methods:**
  - **k**-Nearest Neighbor (Instance based learning)
  - Support Vector Machines
  - Decision trees

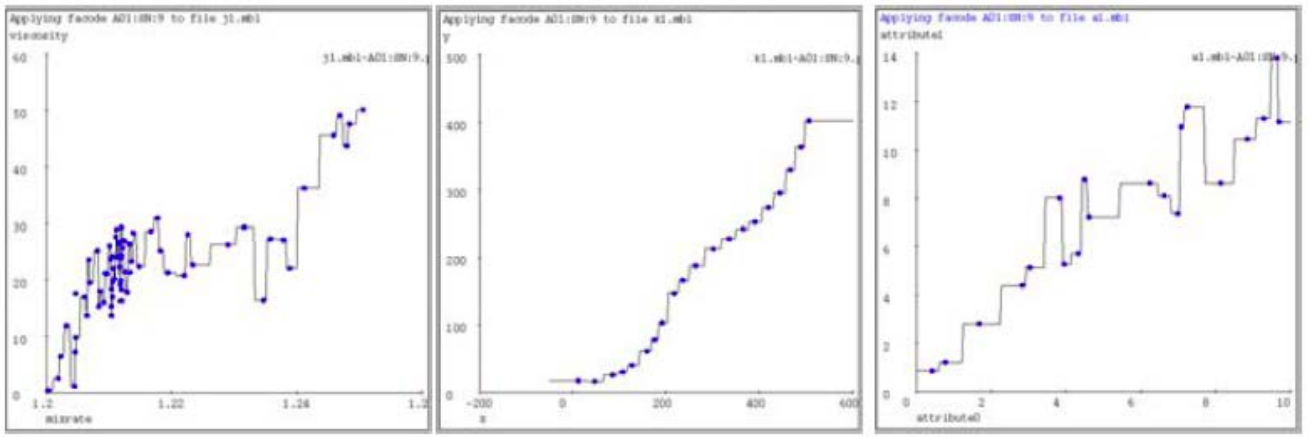
- **Main question:**
  How to efficiently train
  (build a model/find model parameters)**?**

Instance Based Learning

- **Instance based learning**
- **Example: Nearest neighbor**
  - Keep the whole training dataset: **{(x, y)}**
  - A query example (vector) ***q*** comes
  - Find closest example(s) $\mathbf{x}^*$
  - Predict $\mathbf{y}^*$
- **Works both for regression and classification**
  - **Collaborative filtering** is an example of k-NN classifier
    - Find ***k*** most similar people to user **x** that have rated movie **y**
    - Predict rating $\mathbf{y_x}$ of **x** as an average of $\mathbf{y_k}$

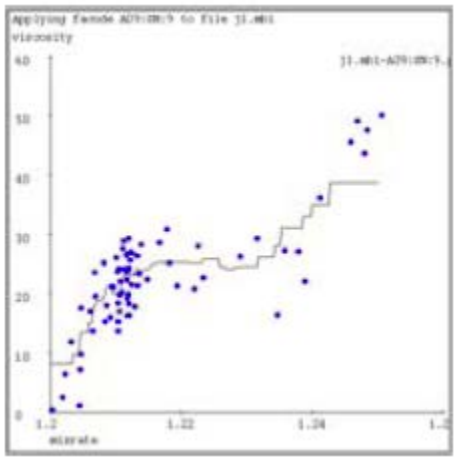
1-Nearest Neighbor

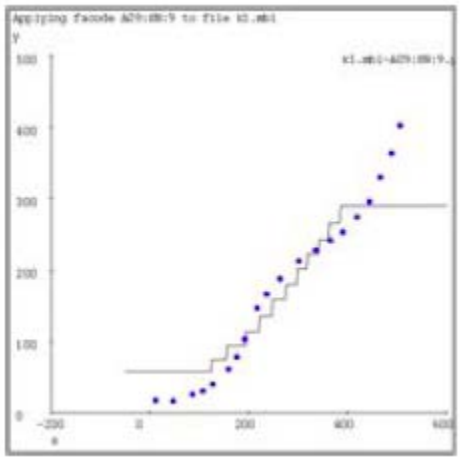
- **To make Nearest Neighbor work we need 4 things:**
  - **Distance metric:**
    - Euclidean
  - **How many neighbors to look at?**
    - One
  - **Weighting function (optional):**
    - Unused
  - **How to fit with the local points?**
    - Just predict the same output as the nearest neighbor

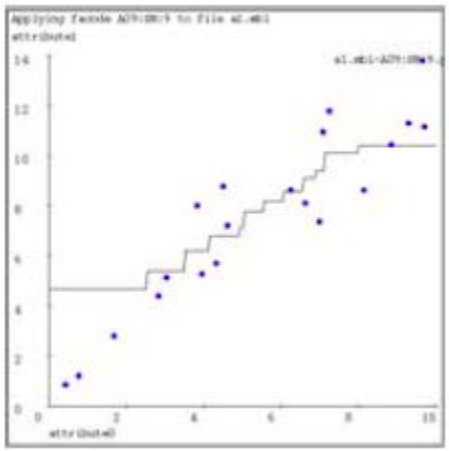


k-Nearest Neighbor

- **Distance metric:**
  - Euclidean
- **How many neighbors to look at?**
  - ***k***
- **Weighting function (optional):**
  - Unused
- **How to fit with the local points?**
  - Just predict the average output among *k* nearest neighbors



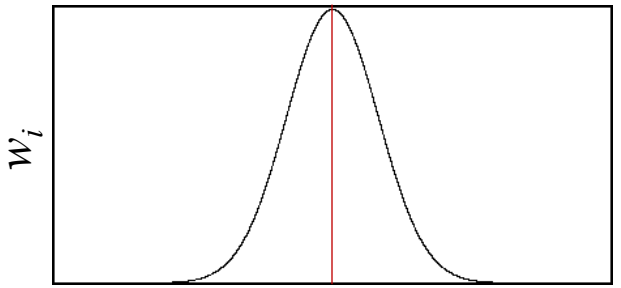


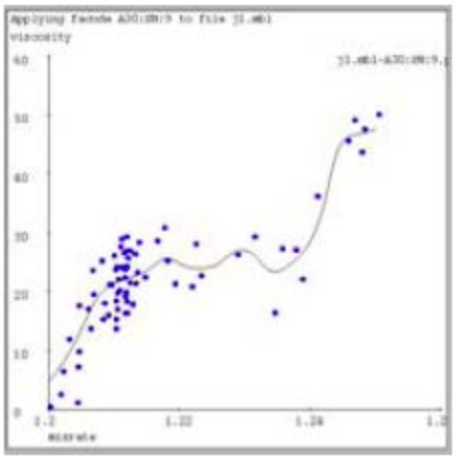


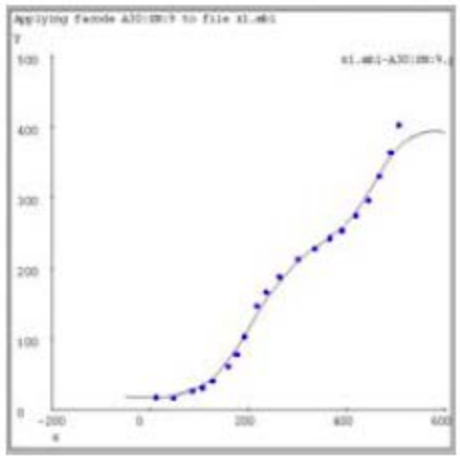
k=9

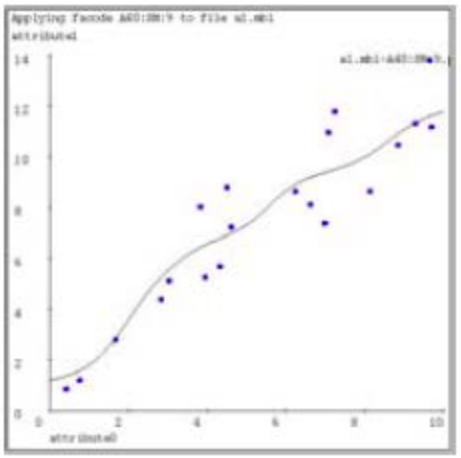
Kernel Regression

- **Distance metric:**
  - Euclidean
- **How many neighbors to look at?**
  - **All of them (!)**
- **Weighting function:**
  - $w_i = \exp(-\frac{d(x_i,q)^2}{K_w})$
    - Nearby points to query q are weighted more strongly. K_w…kernel width.
- **How to fit with the local points?**
  - **Predict weighted average:** $\frac{\sum_i w_i y_i}{\sum_i w_i}$



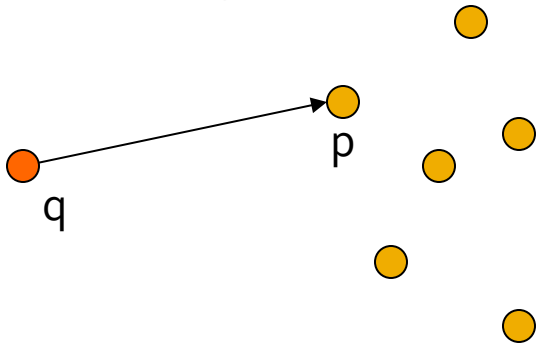






How to find nearest neighbors?

- **Given:** a set P of n points in R^d
- **Goal: Given a query point q**
  - **NN:** Find the *nearest neighbor* p of q in P
  - **Range search:** Find one/all points in P within distance r from q



Use locality sensitive hashing!