Instance-Based Learning
(*aka* non-parametric methods)

CSE 446 Machine Learning Daniel Weld

March 7, 2012

with slides from Carlos Guestrin & Bishop

Machine Learning

Supervised Learning

Unsupervised Learning

Reinforcement Learning

Parametric

Non-parametric

Space of ML Problems

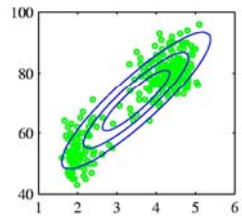
Type of Supervision
(eg, Experience, Feedback)

What is Being Learned?

	Labeled Examples	Reward	Nothing
Discrete Function	**Classification**		Clustering
Continuous Function	**Regression**		
Policy	Apprenticeship Learning	Reinforcement Learning	

Nonparametric Methods

- Parametric models restricted to specific forms
  - May not be a good fit



- Nonparametric methods make few assumptions
  - Often work extremely well in practice

Todo

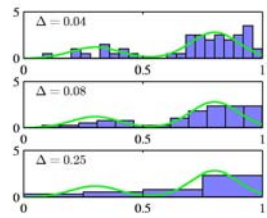
- Bishop approach is too theoretical – don't do it!

Histograms

Histogram methods partition the data space into distinct bins with widths Δ_i and count the number of observations, n_i, in each bin.

$$p_i = \frac{n_i}{N\Delta_i}$$

- Often, the same width is used for all bins, $\Delta_i = \Delta$.
- Δ acts as a smoothing parameter.



In an D-dimensional space, using M bins in each dimension will require M^D bins!

Why is this bad?

Nonparametric Methods

- Assume observations drawn from a density p(x) and consider a small region R containing x such that

$$P = \int_{\mathcal{R}} p(\mathbf{x})\,\mathrm{d}\mathbf{x}.$$

- The probability that K out of N observations lie inside R is Bin(K | N,P) and if N is large

$$K \simeq NP.$$

If V (the volume of R) is sufficiently small, p(x) is approximately constant over R and

$$P \simeq p(\mathbf{x})V$$

Thus

$$\boxed{p(\mathbf{x}) = \frac{K}{NV}.}$$

Nonparametric Methods (3.5)

$$\boxed{p(\mathbf{x}) = \frac{K}{NV}.}$$

- Hold K fixed, determine V from data
  - K-nearest-neighbour approach
- Hold V fixed, determine K from data
  - Kernel-density estimation

Kernel Density Estimation

- Fix V, estimate K from the data.
- Let R be a hypercube centred on x
  - Define the kernel function (Parzen window)

$$k((\mathbf{x}-\mathbf{x}_n)/h) = \begin{cases} 1, & |(x_i - x_{ni})/h| \leqslant 1/2, \\ 0, & \text{otherwise.} \end{cases} \qquad i = 1, \ldots, D,$$

It follows that

$$K = \sum_{n=1}^{N} k\left(\frac{\mathbf{x}-\mathbf{x}_n}{h}\right) \text{ and hence } p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{h^D}k\left(\frac{\mathbf{x}-\mathbf{x}_n}{h}\right).$$

Kernel Density Estimation

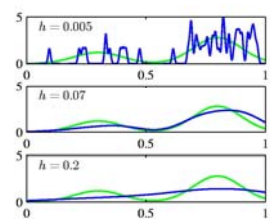
To avoid discontinuities in p(x), use a smooth kernel, e.g. a Gaussian

$$p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{(2\pi h^2)^{D/2}} \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}_n\|^2}{2h^2}\right\}$$

Any kernel such that

$$k(\mathbf{u}) \geqslant 0, \qquad \int k(\mathbf{u})\,\mathrm{d}\mathbf{u} = 1$$

will work.

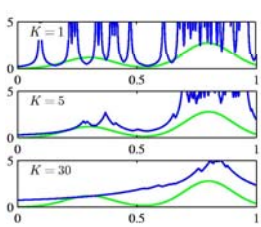


h acts as a smoother.

Nearest Neighbour Density Estimation

Fix K, estimate V from data.
Consider a hypersphere centred on x
Let it grow to a volume, V*
that includes K of N data points.
Then

$$p(\mathbf{x}) \simeq \frac{K}{NV^*}.$$



K acts as a smoother.

Pros / Cons

- Nonparametric models (besides histograms)
  - Requires storing and computing with the entire data set.
- Parametric models, once fitted,
  - Much more efficient in terms of storage and computation.

K-Nearest-Neighbours for Classification

- Given a data set with N_k data points from class C_k and , we have

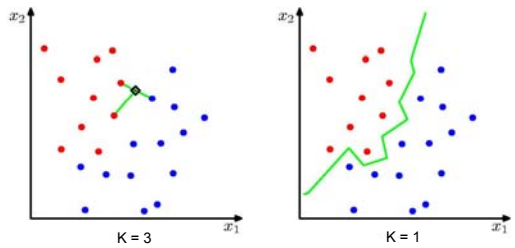
$$\sum_k N_k = N$$

- and correspondingly $$p(\mathbf{x}) = \frac{K}{NV}$$
- Since $$p(\mathbf{x}|C_k) = \frac{K_k}{N_k V},$$, Bayes' theorem gives

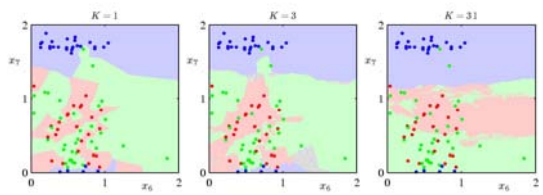
$$p(C_k) = N_k/N$$

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})} = \frac{K_k}{K}.$$

K-Nearest-Neighbours for Classification

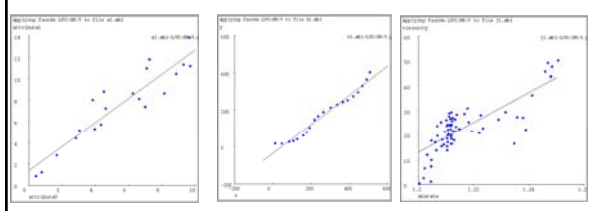


K-Nearest-Neighbours for Classification



- K acts as a smoother
- As N→∞, the error rate of the 1-nearest-neighbour classifier is never more than twice the optimal error (obtained from the true conditional class distributions).

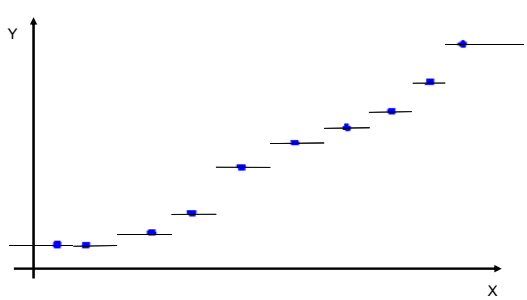
Linear Regression: What can go wrong?



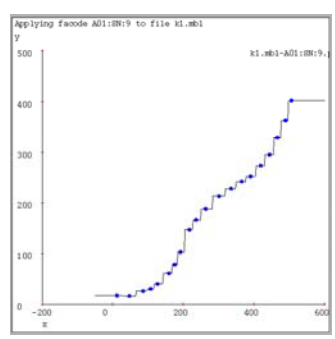
What do we do if the bias is too strong?

- 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

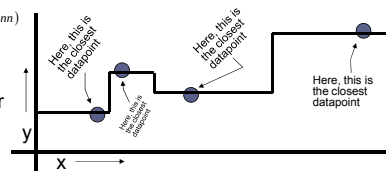
Given datapoints (x_1,y_1) (x_2,y_2)..(x_N,y_N), where we assume $y_i=f(x_i)$ for some unknown function f.
Given query point x_q, your job is to predict $\hat{y} \approx f(x_q)$
Nearest Neighbor:

1. Find the closest x_i in our set of datapoints

$$i(nn) = \underset{i}{\operatorname{argmin}} |x_i - x_q|$$

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

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.

x_1	y_1
x_2	y_2
x_3	y_3
…	
x_n	y_n

To define an instance-based learner, specify four things:

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

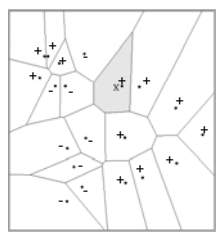
1-Nearest Neighbor

To define an instance-based learner, specify four things:

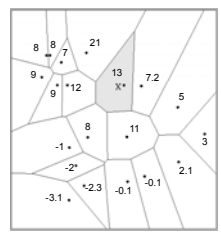
1. *A distance metric*
   Euclidian (and many more)
2. *How many nearby neighbors to look at?*
   One
3. *A weighting function (optional)*
   Unused
4. *How to fit with the local points?*
   Just predict the same output as the nearest neighbor.

Multivariate 1-NN examples

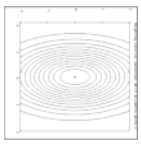
Classification



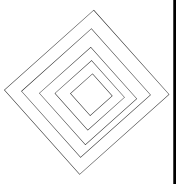
Regression



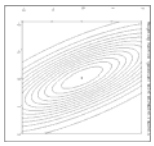
Notable distance metrics (and their level sets)



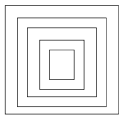
Scaled Euclidian (L_2)



L_1 norm (absolute)



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



L1 *(max) norm*

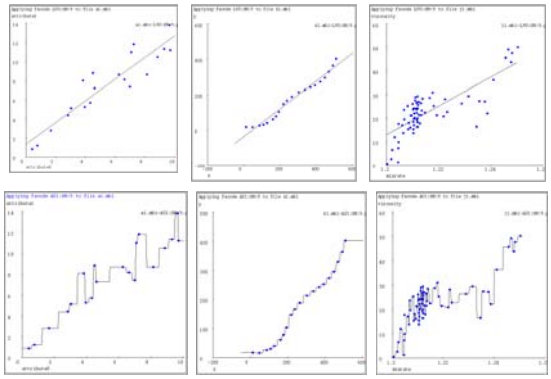
Consistency of 1-NN

- 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 noise-free 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?



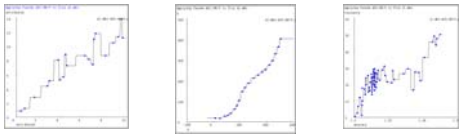
k-Nearest Neighbor

Instance-based learning, four things to specify:

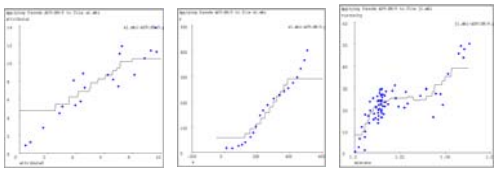
1. *A distance metric*
   Euclidian (and many more)
2. *How many nearby neighbors to look at?*
   k
1. *A weighting function (optional)*
   Unused
2. *How to fit with the local points?*
   Return the average output
   predict: (1/k) Σy_i (summing over k nearest neighbors)

k-Nearest Neighbor

k=1



k=9

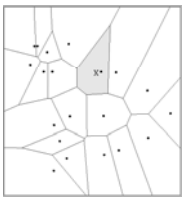


Which is better? What can we do about the discontinuities?

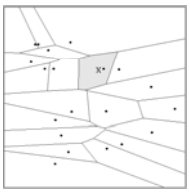
Weighted distance metrics

Suppose the input vectors $x_1, x_2, \ldots x_n$ are two dimensional:
$\mathbf{x}_1 = (x_{11}, x_{12})$, $\mathbf{x}_2 = (x_{21}, x_{22})$, $\ldots \mathbf{x}_N = (x_{N1}, x_{N2})$.

Nearest-neighbor regions in input space:



$Dist(\mathbf{x}_i,\mathbf{x}_j) = (x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2$



$Dist(\mathbf{x}_i,\mathbf{x}_j) = (x_{i1} - x_{j1})^2 + (3x_{i2} - 3x_{j2})^2$

The relative scaling of the distance metric affect region shapes

Weighted Euclidean distance metric

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

Or equivalently,

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

where

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

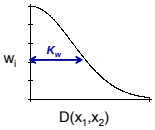
Other Metrics…

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

Kernel Regression

Instance-based learning:

1. *A distance metric*
   Euclidian (and many more)
2. *How many nearby neighbors to look at?*
   All of them

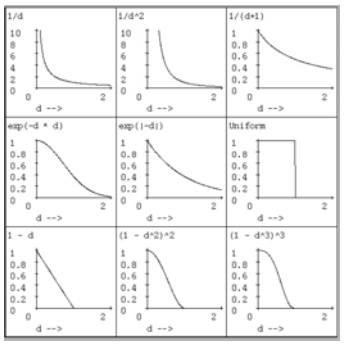


3. *A weighting function*
   $w_i = exp(-D(x_i, query)^2 / K_w^2)$
   Nearby points to the query are weighted strongly,
   Far points weighted weakly.
   The K_W parameter is the **Kernel Width**. Very important.
4. *How to fit with the local points?*
   Predict the weighted average of the outputs:
   predict = $\Sigma w_i y_i / \Sigma w_i$

Many possible weighting functions

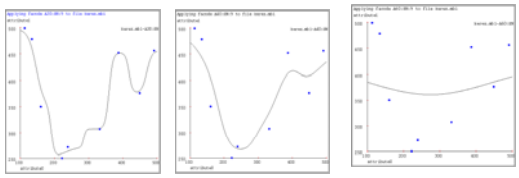
$w_i = exp(-D(x_i, query)^2 / K_w^2)$

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

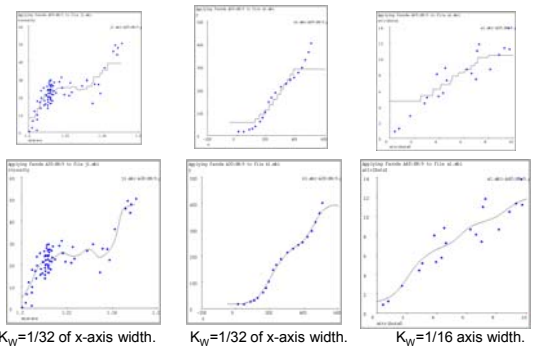
Increasing the kernel width K_w means further away points get an opportunity to influence you.

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

Kernel regression on our test cases

NN k=9

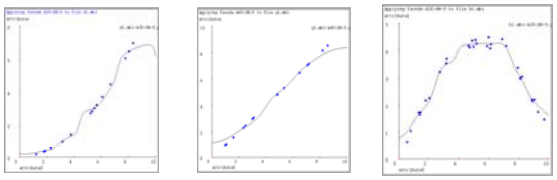
Kernel regression



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?*

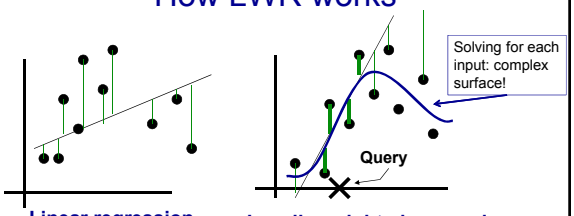
  All of them
- *A weighting function (optional)*

  Kernels: ***wi = exp(-D(xi, query)² / Kw²)***
- *How to fit with the local points?*

  General weighted regression:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{k=1}^{N} w_k^2 \left(y_k - \beta^T x_k\right)^2$$

How LWR works



Linear regression

- Same parameters for all queries

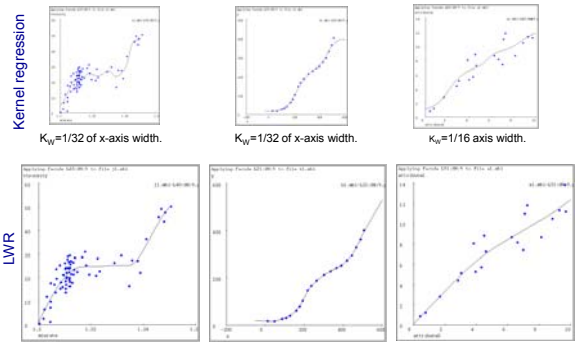
$$\hat{\beta} = \left(X^T X\right)^{-1} X^T Y$$

Locally weighted regression

- Solve weighted linear regression for each query

$$\beta = \left((WX)^T WX\right)^{-1} (WX)^T WY \qquad W = \begin{pmatrix} w_1 & 0 & 0 & 0 \\ 0 & w_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & w_n \end{pmatrix}$$

LWR on our test cases

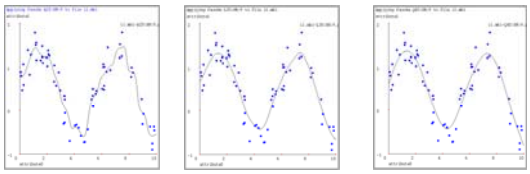


K_W=1/32 of x-axis width. K_W=1/32 of x-axis width. K_W=1/16 axis width.

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.



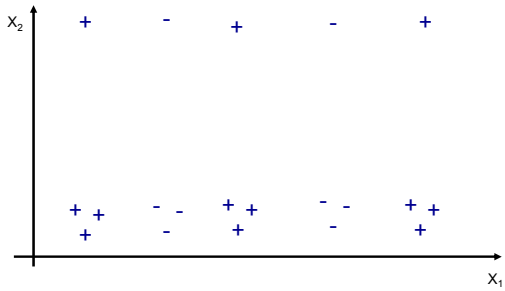
K_W = 1/100 x-axis K_W = 1/40 x-axis K_W = 1/15 x-axis

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 based learning

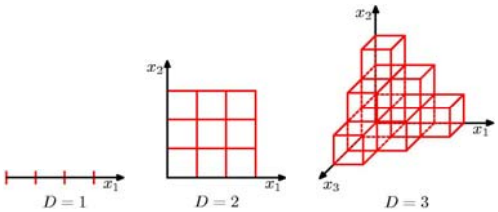
- Must store and retrieve all data!
  - Most real work done during testing
  - For every test sample, must search through all dataset – very slow!
  - But, there are fast methods for dealing with large datasets
- Instance-based learning often poor with noisy or irrelevant features
  - In high dimensional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

Curse of the irrelevant feature



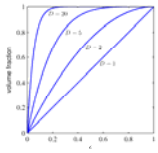
This is a contrived example, but similar problems are common in practice
Need some form of feature selection!!

Curse of Dimensionality

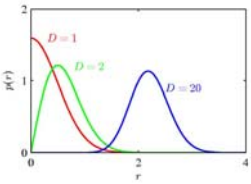


Curse of Dimensionality

Fraction of volume of sphere lying in the range r = [1, 1-ε]



Gaussian Densities in higher dimensions



Side-Stepping the Curse

- Dimensionality reduction
  - Eg, PCA
- Then use NN

44

In Summary: 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

45

Acknowledgment

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

46