The Auton Project

AUTON

How general-purpose can we make Machine Learning?

This Talk:

1. Memory-based Learning

Justin Boyan
Scott Davies
Kan Deng
Artur Dubrawski
Mary Soon Lee
Andrew Moore
Andrew Ng
Jeff Schneider
David Stott

Relevant Papers: `http://www.cs.cmu.edu/~awm/papers.html`

Overview

- Some Function Approximation Scenarios
- Cross Validation
- What is Memory-based Learning?
- Nearest Neighbor , k-Nearest Neighbor
- Kernel Regression
- Locally weighted Linear Regression
- Multivariate memory-based learning
- Blackbox: Automatic Parameter tuning
- Parameter / Feature selection searching
- Task transfer
- Bayesian Memory Based Learning
- Practical use of MBL
- MBL vs Neural Nets

Warning/Apology: Slides are not handcrafted with loving care! Wild layouts, hideous font choices and font inconsistencies abound!

Question

What is memory-based function approximation useful for?

Use (1): "Hands-off" non-parametric relation finding.

You run an HMO (or a steel tempering process) (or a 7-dof dynamic robot arm)

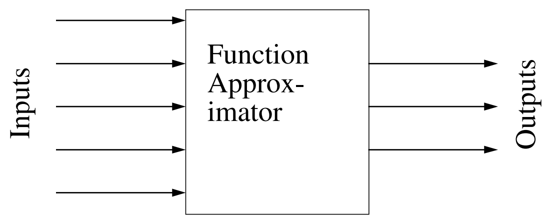
You want an intelligent assistant to spot patterns and regularities among pairs or triplets of variables in your database...

HMO variables:	**Steel Variables:**	**Robot Variables:**
Physician Age	**Line Speed**	**Roll**
Patient Age	**Line Spd -10mins**	**DRoll**
Charge/Day	**Line Spd -20mins**	**DDRoll**
Charge/Discharg	**Slab width**	**Pitch**
Discharges/1000	**Slab height**	**DPitch**
ICD-9 Diagnosis	**Slab Temp Stg1**	**DDPitch**
Market Share	**Slab Temp Stg2**	**SonarHeight**
Mortality/1000	**CoolTunn2 Setp**	**LaserHeight**
Patient ZIP	**CoolTunn5 Setp**	**FlightTime**
Zip Median Age	**CoolTunn2 Temp**	**ThrustRate**
.....	.....	.....

You especially want it to find more than just the linear correlations...

Use (2): Low Dimensional Supervised Learning

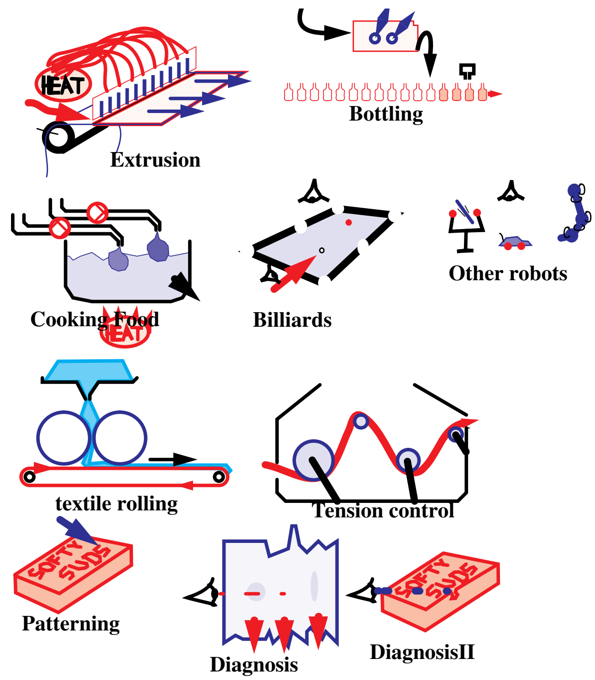
You have lots of data, not many input variables (less than 7, say) and you expect a very complex non-linear function of the data.



Examples:

- Skin Thickness vs τ , ϕ for face scanner .
- Topographical Map
- Tumor density vs (x,y ,z)
- Mean wasted Asprin vs (fill-target, mean-weight, weight-sdev , rate) for an asprin-bottle filler .
- Object-ball collision-point vs (x,y ,θ) in Pool

Use (3): Complex Function of a subset of inputs[



Use (4): Simple function of most inputs but complex function of a few .

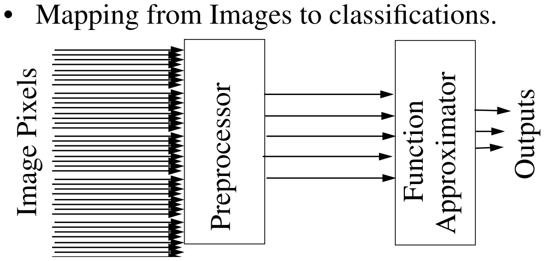
Examples:

- $f(x) = x_1 + 3x_2 - x_4 + sin(log(x_5)*x_6) - x_7^2 + x8 - x9 + 8x10$
- Car Engine Emissions
- Food Cooling T unnel

Use (5): Complex function of a few features of many input variables

Examples:

- Mapping from acoustic signals to “Probability of Machine Breakdown”.
- Time series data analysis
- Mapping from Images to classifications.



(e.g. Product inspection, Medical imagery , Thin Film imaging..)

Reminder

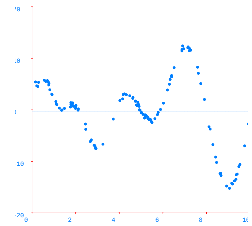
How do we judge the quality of some learning algorithm on some data?

In the next few slides we will review the reasons that “mean absolute error on training set” is inadequate.

And then we will discuss why Leave-one-out cross-validation can do a good job.

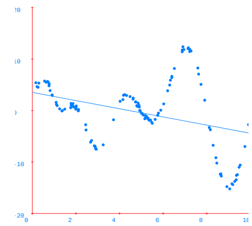
FunApprox Quality?

Suppose we have 3 function approximators:
FA1: Just predict the average of all the outputs
FA2: Do a linear regression on the outputs
FA3: “Join the dots” between output points



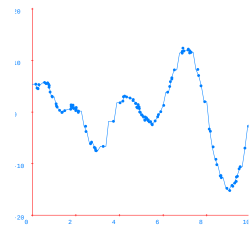
Global A verage

(A90:9)



Global Linear Regress

(L90:9)



Join-the-dots

(A01:9)

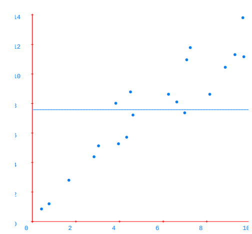
Quality Measure Idea I

Just compute the average residual.

In our example, that correctly prefers “join the dots”.

What’ s the problem with Residual as a measure of function approximator quality?

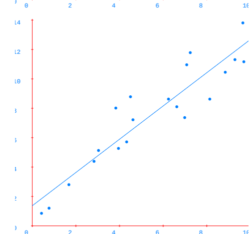
FunApprox Quality?



Global A verage

(A90:9)

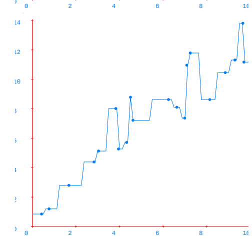
Mean Absolute Residual = 2.0



Global Linear Regress

(L90:9)

Mean Absolute Residual = 0.6



Join-the-dots

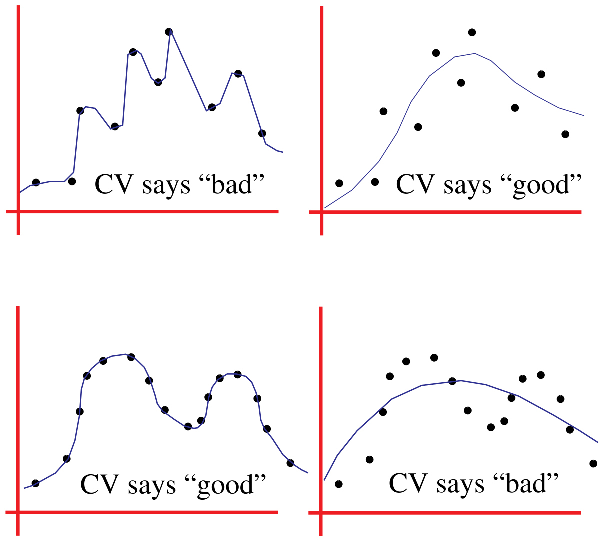
(A01:9)

Mean Absolute Residual = 0.0

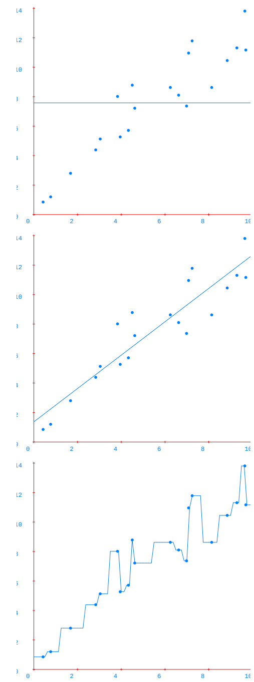
If we judge by mean-absolute residual the bottom function approximator wins. But that is clearly fitting noise. V ery undesirable!

Cross Validation

A way of checking how well a function fit will generalize to new data.



CrossVal Gets it right



Global Average

(A90:9)

LOO-xve = 2.97

Global Linear Regress

(L90:9)

LOO-xve = 1.23

Join-the-dots

(A01:9)

Loo-xve = 1.76

Leave-one-one cross validation prefers the middle approximator .

Leave-One-Out Cross Validation

The leave-out-out-error of the i’ th data point is the different between the output of the i’ th datapoint, and the prediction you’d get if you had trained all all datapoints EXCEPT the i’th.

It is statistical estimator for the expected error if you are asked to predict a new datapoint drawn independently from the same distribution.

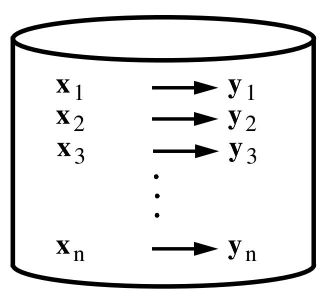
Other cross-validation methods:

- k-fold cross validation
- Bootstrapping

Memory-based learning

A kind of 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:

1. A distance metric
2. How many nearby neighbors to look at?
3. A weighting function (optional)
4. How to fit with the local points?

We’ll use LOO-CV to make these decisions....

Euclidian Distance Metric

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

Or equivalently ,

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

where

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \sigma_N \end{bmatrix}$$

Other Metrics...

- Mahalanobis
- Rank-based
- Correlation-based (Stanfill+W altz, Maes' Ringo system....)

Nearest Neighbor

Four things make a memory-based learner:

1. *A distance metric*

   Euclidian

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

   One

3. *A weighting function (optional)*

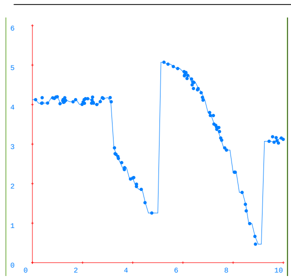
   Unused

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

   Just predict the same output as the nearest neighbor .

On the next slide we examine nearest neighbor working on three different datasets.

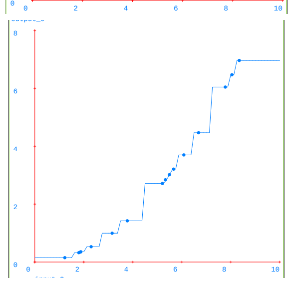
Nearest Neighbor (A01:9)



Nearest Neighbor

(A01:9)

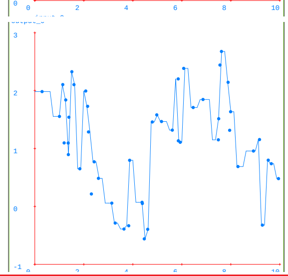
Looks Sensible!



Nearest Neighbor

(A01:9)

A little dodgy!

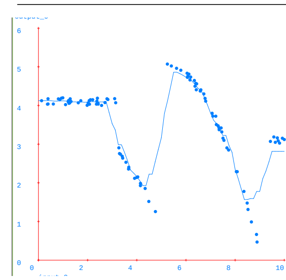


Nearest Neighbor

(A01:9)

Bad Function Approximator!

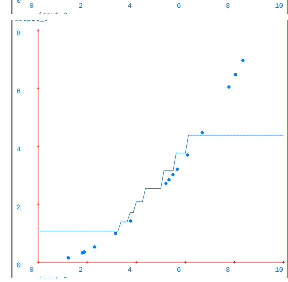
K-nearest neighbor (A0n:9)



9-Nearest Neighbor

(A09:9)

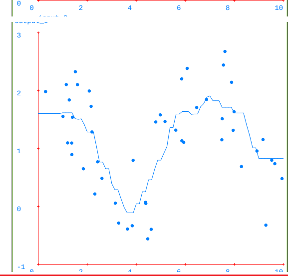
A little sad



9-Nearest Neighbor

(A09:9)

Check out those awful ends!

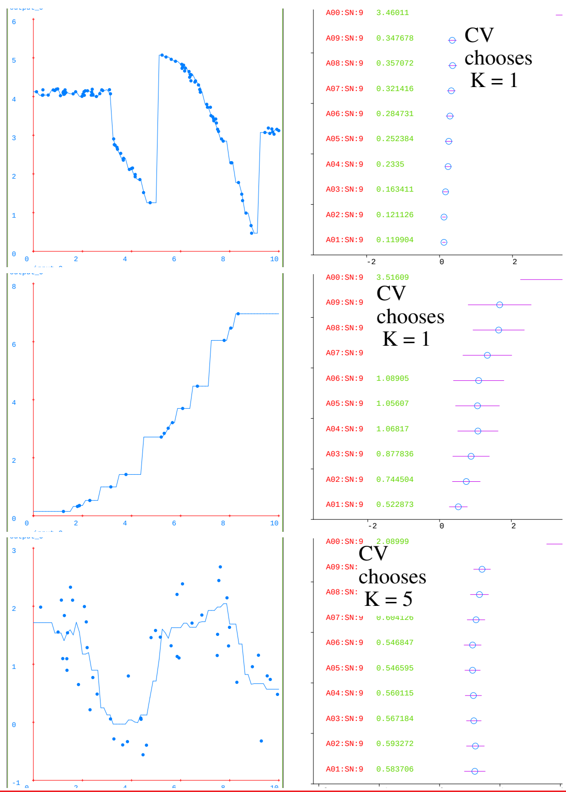


9-Nearest Neighbor

(A09:9)

Not such a bad Function Approximator!

Choose K with CrossValid'n



Kernel Regression

Four things make a memory-based learner:

1. *A distance metric*

   Euclidian

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

   All of them

3. *A weighting function (optional)*

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

   Nearby points to the query are weighted strongly, far points weakly.

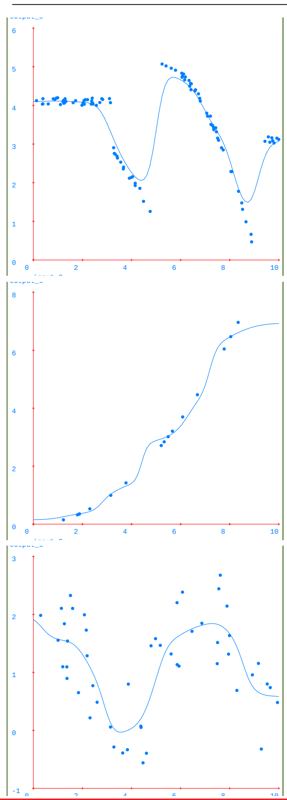
   The W parameter is the Kernel Width. Very important.

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

   Predict the weighted average of the outputs:

   predict = $\Sigma w_i y_i / \Sigma w_i$

Kernel Regression (As0:9)



Kernel Regression

(A30:9)

Smooth but Smears

Kernel Regression

(A30:9)

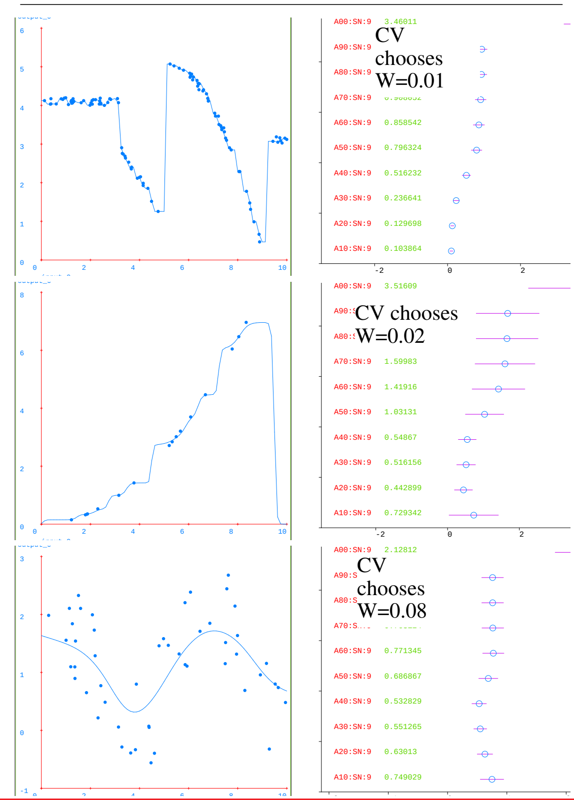
Smooth but, well, pathetic.

Kernel Regression

(A30:9)

Looks Okay-ish

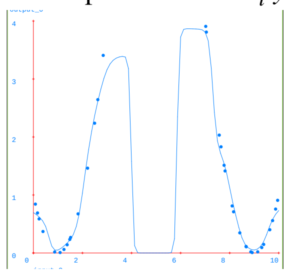
CV Chooses Kernel Width...



Stable Kernel Regress:As1:9

If no points have non-negligible weight, then the prediction

predict = $\Sigma\, w_i\, y_i \,/\, \Sigma\, w_i = 0/0.$

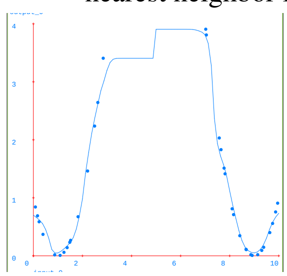


Kernel Regression

(A20:9)

No datapoints in the center . 0/0's ensue: graphed as value 0.

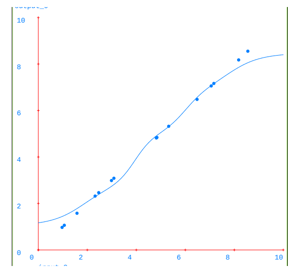
Sensible solution: Always weight the nearest neighbor fully .



Kernel Regression with 1-nearest neigh full weight.

(A21:9)

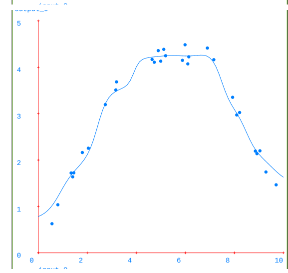
Kernel Regression can do badly if used inappropriately .



Kernel Regression

(A40:9)

Optimal smooth level. Nevertheless, unsatisfying



Kernel Regression

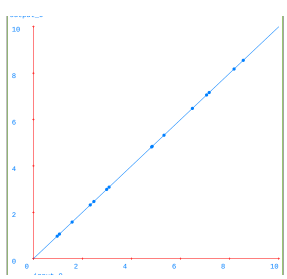
(A40:9)

Optimal smooth level. Kind of bumpy . Notice the misleading derivatives.

What can be done about this?

Locally W eighted Linear Regression (Lsn:9)

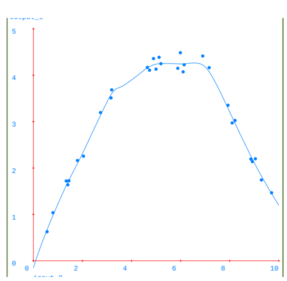
These look much nicer:



Global Linear Regression

(L90:9)

Optimal smooth level. Perfect.



Locally weighted Regression (L WR)

(L30:9)

Optimal smooth level. Much better linear fits.

But how does locally weighted regression work?

Locally weighted Regression

Four things make a memory-based learner:

1. *A distance metric:* **Euclidian**
2. *How many nearby neighbors to look at?*

   All of them

3. *A weighting function (optional)*

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

   Just like kernel regression.

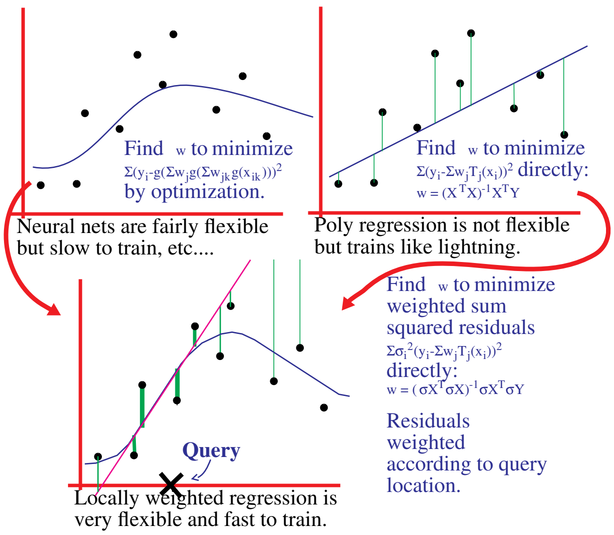
4. *How to fit with the local points?*

   Find b to minimize the locally weighted sum-of-squares of residuals:

   $E = \Sigma\, w_i^2\, (y_i - b^T x_i)^2$

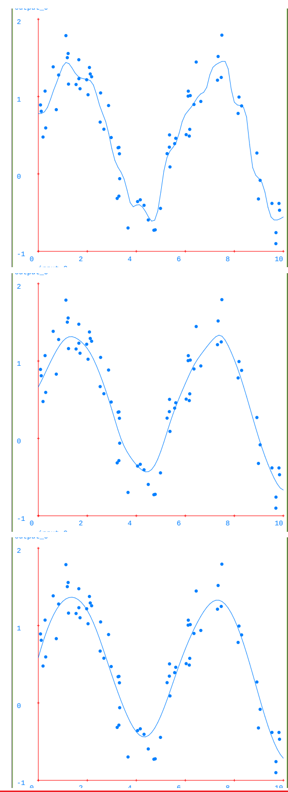
   Done directly by matrix inversion (or SVD).

Locally weighted regression



- Training, adding new data, is almost free.
- Adding new data is almost free.
- "One-shot" learning---not incremental like a neural net.
- Variable resolution.
- Doesn' t forget old train data unless statistics warrant.
- Prediction is expensive. W e have ways around that.
- **Cross-validation** is cheap. THIS IS IMPORT ANT.

LW Poly Regression



Kernel Regression

(A20:9)

Smooth level optimized.

LW Linear Regression

(L30:9)

Smooth level optimized.

LW Quadratic Regression

(Q40:9)

Smooth level optimized.

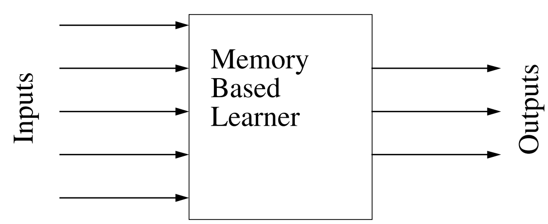
When is quadratic better than linear?

It can mean you can use a wider kernel without introducing bias.

Sometimes you want more than a prediction, you want an estimate of the local Hessian. Then quadratic is you friend!

But in higher dimensions is appallingly expensive, and needs a lot of data.

Multivariate Memory-based learning

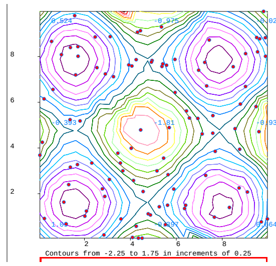


All the methods described so far can generalize to multivariate input and output.

But new questions arise:

- What are good scalings for a Euclidean distance metric?
- What is a better Euclidean distance metric?
- Are all features relevant?
- Do some features have a global rather than local influence?

Two Inputs

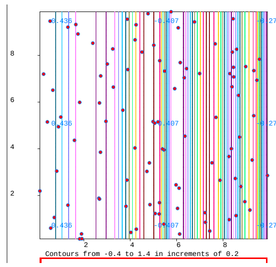


LWQ Regression

(E34:99)

Smooth level, neighs, feature scales all optimized.

f(x,y) = sin(x) + sin(y) + noise



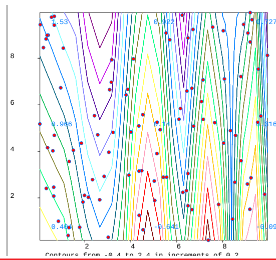
Kernel Regression.

(A10:9-)

Smooth level, neighs, feature scales all optimized.

f(x,y) = sin(x * x)

CV ignores y



LWL Regression

(L20:90)

Smooth level, neighs, feature scales all optimized.

f(x,y) = sin(x*x)+y+ noise

CV treats y globally, x locally

Fabricated Example

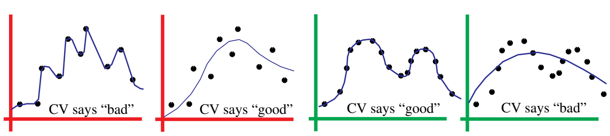
f(x1,x2,x3,x4,x5,x6,x7,x8,x9) =

noise + x2 + x4 + 4sin(0.3x6 + 0.3x8).

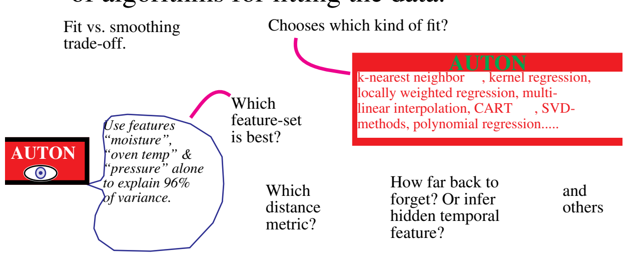
```
2. Recommendation.

Based on the search results so far, the
recommended facode is L20:SN:-0-0-9-9. Let me
explain the meaning:

Locally weighted regression. The following
features define the distance metric:

                    x6 (full strength).
                    x8 (full strength).

A gaussian weighting function is used with
kernel width 0.0441942 in scaled input space.
We do a weighted least squares with the
following terms:

                    Term 0 = 1
                    Term 1 = x2/10
                    Term 2 = x4/10
                    Term 3 = x6/10
                    Term 4 = x8/10
```


Blackbox learning

Cross-validation permits massive self testing.



- There is a large family of memory based learning algorithms. When you combine in all the combinations of algorithm options, choice of basic features, choice of compound features, there is a huge space of algorithms for fitting the data.



- Space of algorithms represented by ***gmstrings,*** searched with bounded-rationality methods .

Q85:MR:099052099035

Making memory-based learning practical

- Algorithms for computationally efficient predictions.

  Kdtrees [Bentley 75, Omohundro 87, Moore 91]

  Multiresolution methods [Deng+Moore 95]

- Very intense use of cross-validation for self tuning over algorithms, features, metrics, smoothers.

- Making intense cross-validation tractable

  Gradient Methods [Atkeson 90, Lowe 93]

  "Bounded rationality" methods for search cutoffs [Maron+Moore 94]

  Feature-set bounded rationality search [Moore+Lee 94]

All incorporated in an autonomous modeler called GMBL *[Moore+Schneider+Lee 1995,1996]* .

In use in several companies and institutions.

Users gratefully sought!

Search Algorithms in GMBL Blackbox

- Forward Selection
- Promising-First Selection
- Backward Elimination
- Hill-Climbing
- Racing
- Schemata Search
- Conjugate Gradient (for numerical parameters)
- AutoRSM (“meta learning”)

Bayesian Memory-based Learning

Combining Bayesian regression analysis [DeGroot 1970] with Locally Weighted Regression [Cleveland and Delvin, 1988] allows us to efficiently ask....

<...Lots of statistics, linear algebra, monte-carlo integration tricks ommitted...>

- What is the predicted value?
- What is the confidence in the predicted value?
- What is the predicted gradient?
- What is the confidence in that?
- What is the local Hessian, and the confidence in that?
- What is the local noise and the confidence in that?
- With what probability is there a local optimum nearby?
- Which direction will head me fastest to local optimum? And what’s the confidence in that?
- Where is the local optimum, and what’s the confidence in that?
- Where, locally, would a single experiment most increase my confidence in a given one of the above questions?

Using MBL

Robot Control

Process Control

Stochastic Optimization

Reinforcement Learning

MBL vs Neural Nets

MBL has some advantages:

- Can fit low dimensional, very complex, functions very accurately. Neural nets require considerable tweaking to do this.
- Doesn’t “forget” old data
- You can get meaningful confidence intervals, local gradients back, not merely a prediction.
- Cross-validation is cheap. Makes MBL an ideal function approximator to place a “black box” auto-tuner around.

Neural Nets have some advantages:

- With large datasets, MBL predictions are slow (although Auton’s kdtree aproximations, and newer cache approximations help a lot). Neural Net predictions are independent of training set size (though training time is linear).
- Neural nets can be trained directly on problems with hundreds or thousands of inputs (e.g. from images). MBL would need someone to defined a smaller set of image features instead.

MBL Current research

Conjugate gradient parameter tuning (Atkeson, Lowe)

Kd-trees (Auton Group, also Omohundro, Grosse)

Racing methods (Auton Group)

Locally weighted logistic regression (Kan Deng, Auton Group): Statistical memory-based learning for Classification.

Receptive field L WR (Stefan Schaal)

Probability density estimation / manifold estimation / learning “families” of related tasks (Steve Omohundro)

Simplex-Interpolation-based cacheing (Auton Group).