Learning and Recognition by Model Merging

Stephen M. Omohundro

International Computer Science Institute

Berkeley, California

- Learning and Recognition.
- The Bayesian framework.
- Model merging.
- RBF's, surfaces, stochastic grammars.
- Bumptrees.

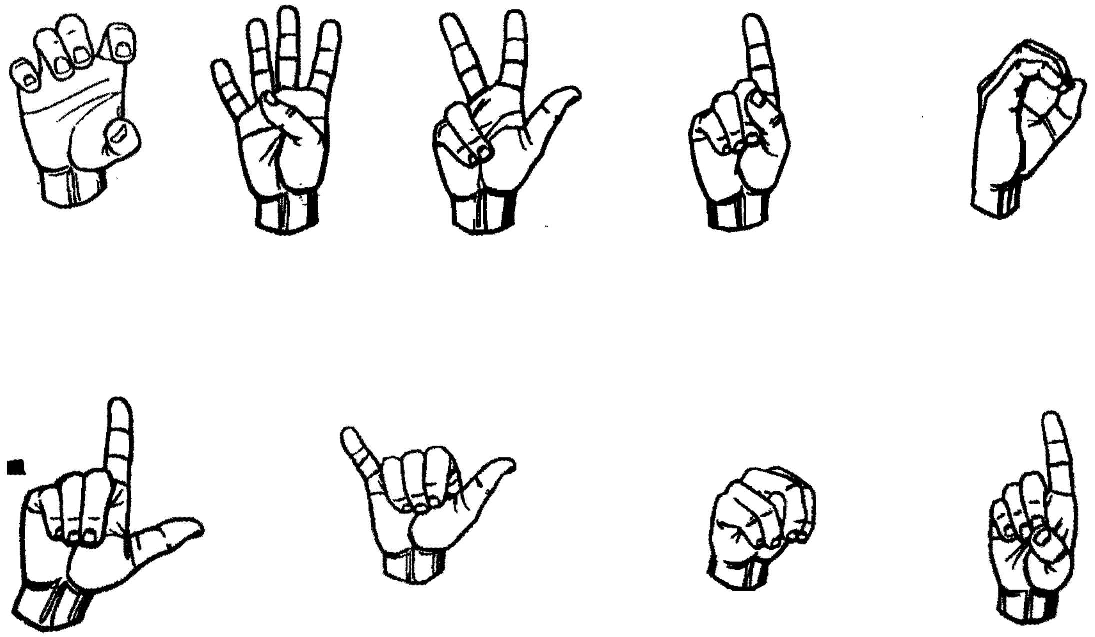
Recognition and Learning

- Both are inductive model building processes.

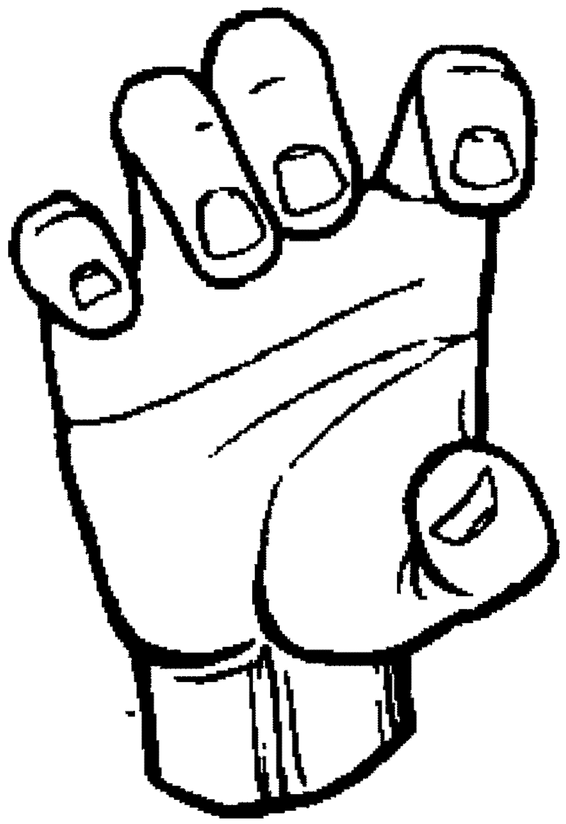
- ***Recognition:*** sensation -> perception
- ***Learning:*** past experience -> present predelections

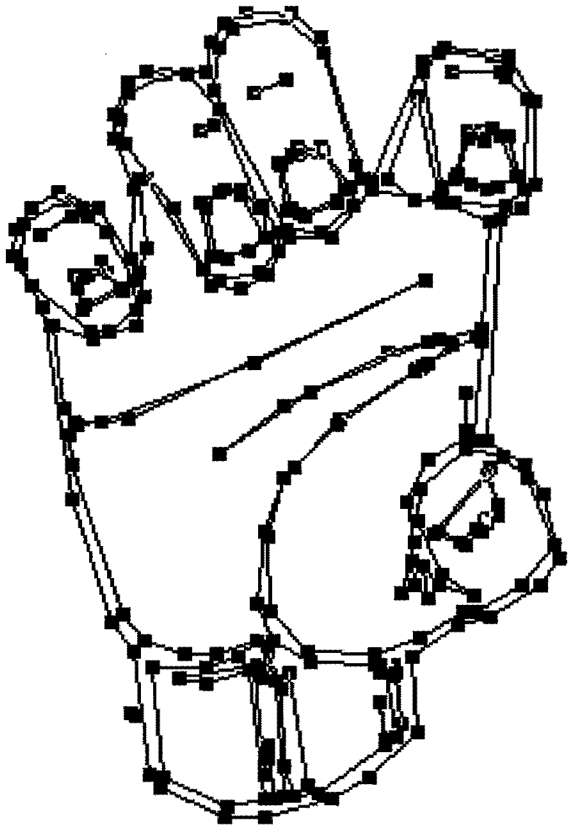
- Learning provides the model construction materials for recognition.

Scanned Hands



Approximate boundaries with segments

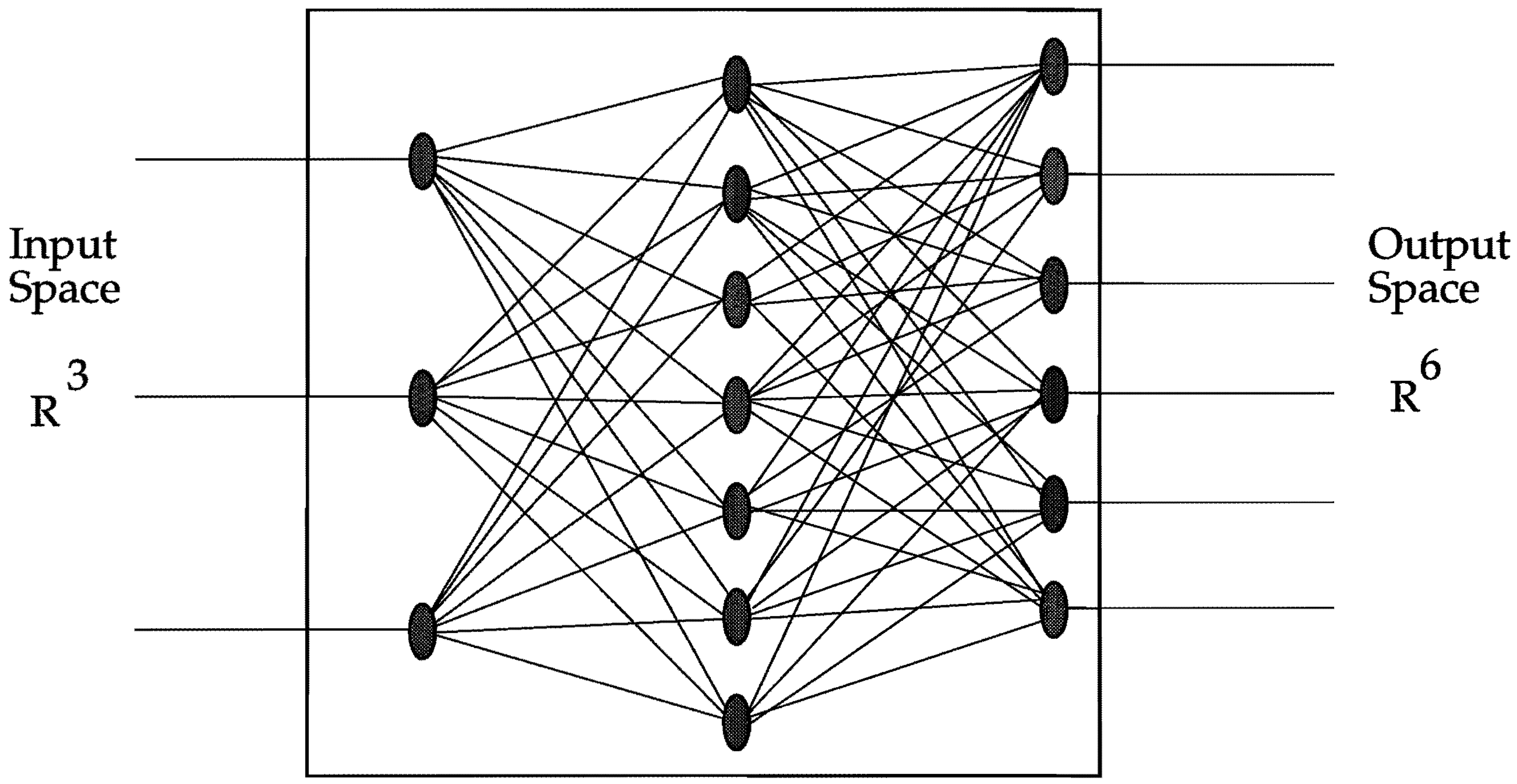




Popular Learning Algorithms

- Fixed space of possible models.
  (eg. a neural network)
- Simple parameterization of model space.
  (eg. network weights)
- Start at random parameter value.
  (eg. random weights)
- Modify parameter to improve performance on data.
  (eg. backpropagation for stochastic error gradient descent)
- Possibly include terms to help prevent overfitting
  (eg. weight decay or cross-validation)

Backpropagation Neural Network



Problems with Popular Learning Approaches

- Cognitively implausible: doesn't know what it knows. *(always thinks it has the full model)*
- Incapable of one-shot learning. *(single examples rarely have a big impact)*
- Subject to inappropriate choice of model space. *(to small to represent or approximate the true model)*
- Subject to overfitting when the model space is too large. *(too many parameters for the data)*
- Can't allocate resources. *(may be large enough, but can't put model components where needed)*
- Slow to learn. *(gradually varies parameters)*
- Liable to get stuck in local minima. *(model parts interfere with one another)*
- Computationally expensive. *(full model is evaluated for each example)*
- No coherent underlying semantics. *(compositionality?)*

Human and Animal Induction

- Very different from the popular models.
- One shot learning: individual experiences can have a big impact.
- Episodic memory for specific events, especially when first learning about a domain. (each experience is precious)
- Initially generalize by similarity. (eg. Shepard's universal exponential law of generalization after one training example).
- As experience accumulates, build more complex models as warranted.
- Adaptive structure: complex models in one portion, simple models in others.
- Mostly avoid overfitting and getting stuck in local minima.
- Mechanisms to only access relevant information.
- Organism knows what it knows. (confidence in model).
- Humans are adaptive to a wide variety of domains.
- Some (eg. Anderson) claim Bayesian underlying semantics.

Bayesian Approach

- Assume agents that must take actions based on uncertain data.
- Assume they can compare their preferences for states of the world.
- Under very mild assumptions: There exists a "prior" probability distribution and a "utility" function such that any rational agent behaves according to Bayesian decision making.
- If not, agent is subject to "dutch books" *(loses however things come out).*
- Procedure: Using prior and data compute posterior. Act so as to maximize posterior expected utility.
- Act to maximize posterior expected utility. The expectation requires an expensive integral over the model space.
- Need approximations to do the average.
- Poor approximations (eg. MAP) lead to overfitting.

Choice of Prior

- Choice of prior is the choice of language for a model.
- Prior for recognition is the posterior for learning.
- Prior for learning should be based on universal properties of physics (and maybe biology, sociology, psychology, etc).
- Bertrand Russell's "On Human Knowledge".
- **Time:** Expect the future to be like the past.
- **Continutity:** Unless known to be otherwise, assume that nearby perceptions correspond to nearby states of the world.
- **Sparseness:** The world is composed from component models which typically interact only sparsely with one another.
- **Locality:** Sensory data is composed of components which typically interact only sparsely with the components of the world.
- **Natural kinds:** World model components tend to clump into a small number of repeated kinds.

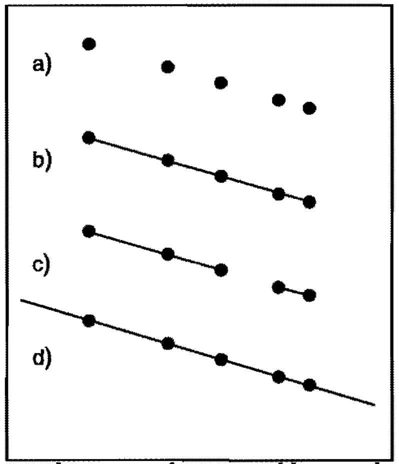
Best-first Model Merging

-
- Use best first model merging to construct arbitrarily complex models efficiently without overfitting.
- Use adaptive patches and bumptrees to efficiently represent geometric information.

Overfitting

Posterior won't be peaked if there is too little data for the complexity of the model space.

Bayesian segmentation

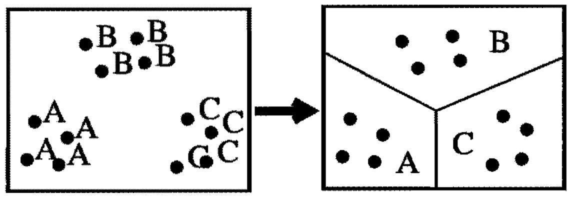


Prefer the simplest explanation because there are so many complicated explanations that their prior is small.

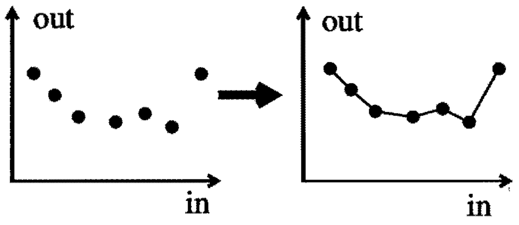
Occam's razor **Rissanen's minimum description length principle**

Goodness of fit vs. complexity represented by likelihood vs. prior.

Geometric Learning Algorithms



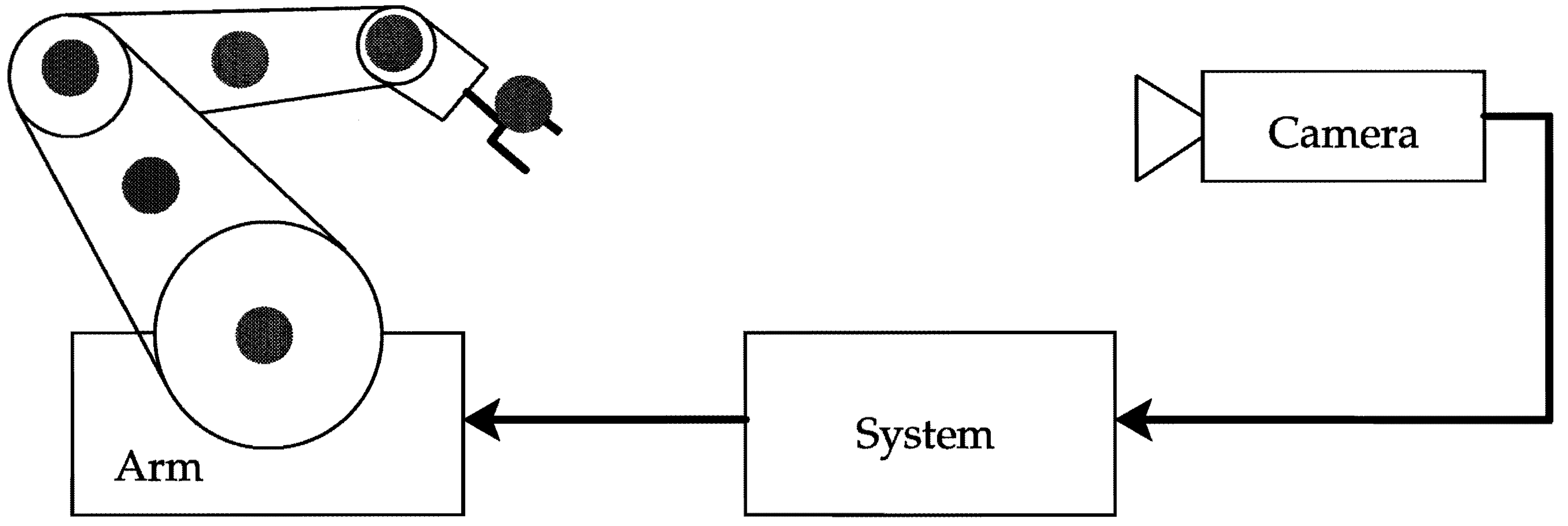
Classification Learning



Mapping Learning

- Learning smooth mappings.
- Learning discrete mapping.
- Probability density estimation.
- Learning constraint submanifolds.
- Least squares inverse of a map.
- Nearest point in a parameterized family.
- Partial match queries.
- Discovering product structure.

A Visually Guided Robot Arm.

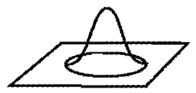


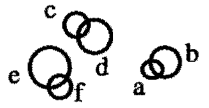
$$\text{Kinematic space} \longrightarrow \text{Visual space}$$

$$R^3 \longrightarrow R^{12}$$

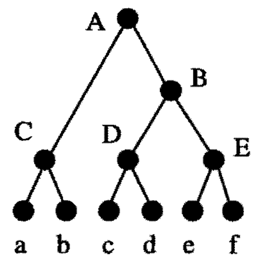
Bumptrees.

A *bumptree* is a complete binary tree with a function associated to each node such that an interior node's function bounds all leaf functions beneath it.

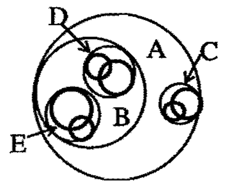

Ball supported bump



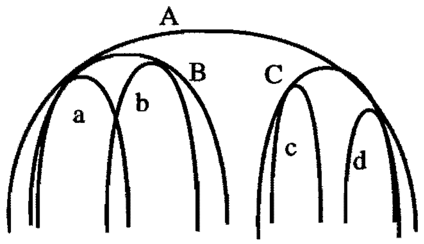
2-d leaf functions

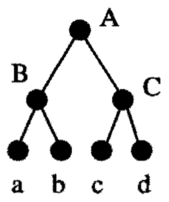


tree structure

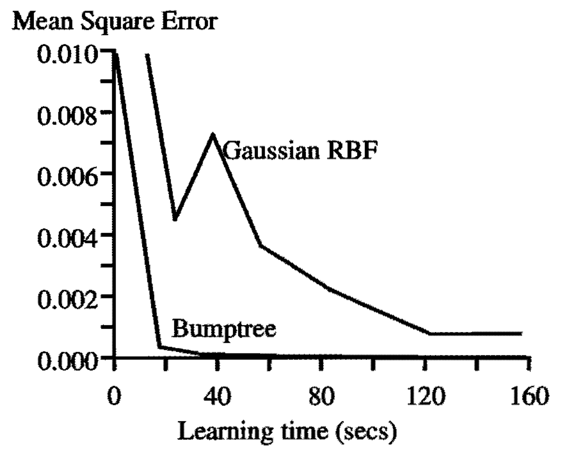


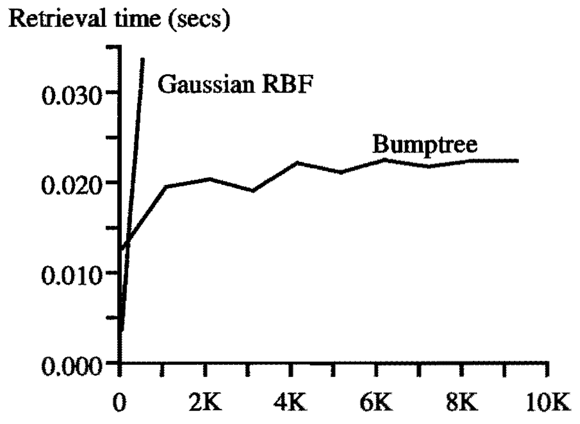
tree functions



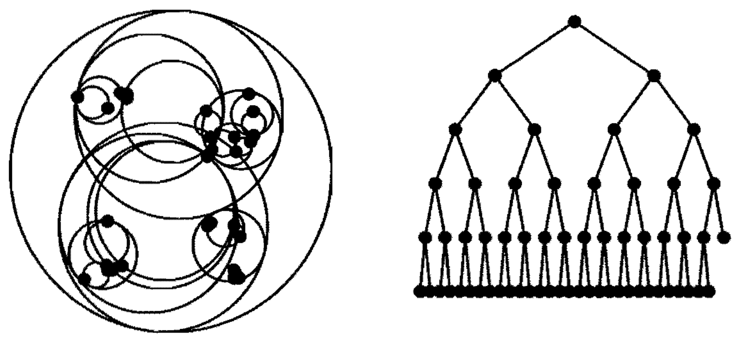


Bumptree learning and retrieval times

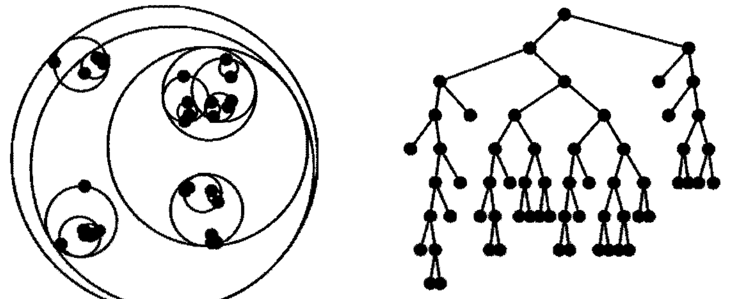




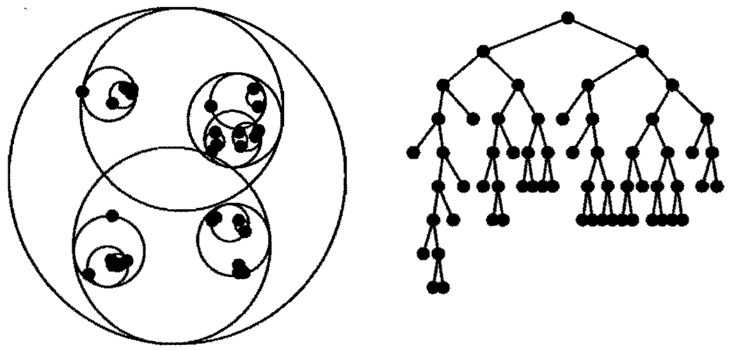
Bumptrees produced from a Cantor set by the five construction algorithms.



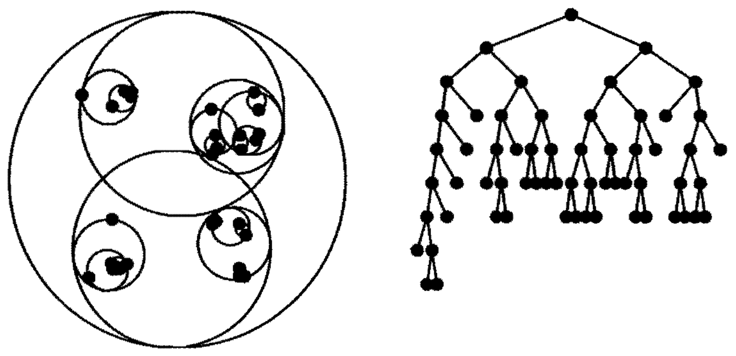
Kd



Insertion and
Cheap insertion



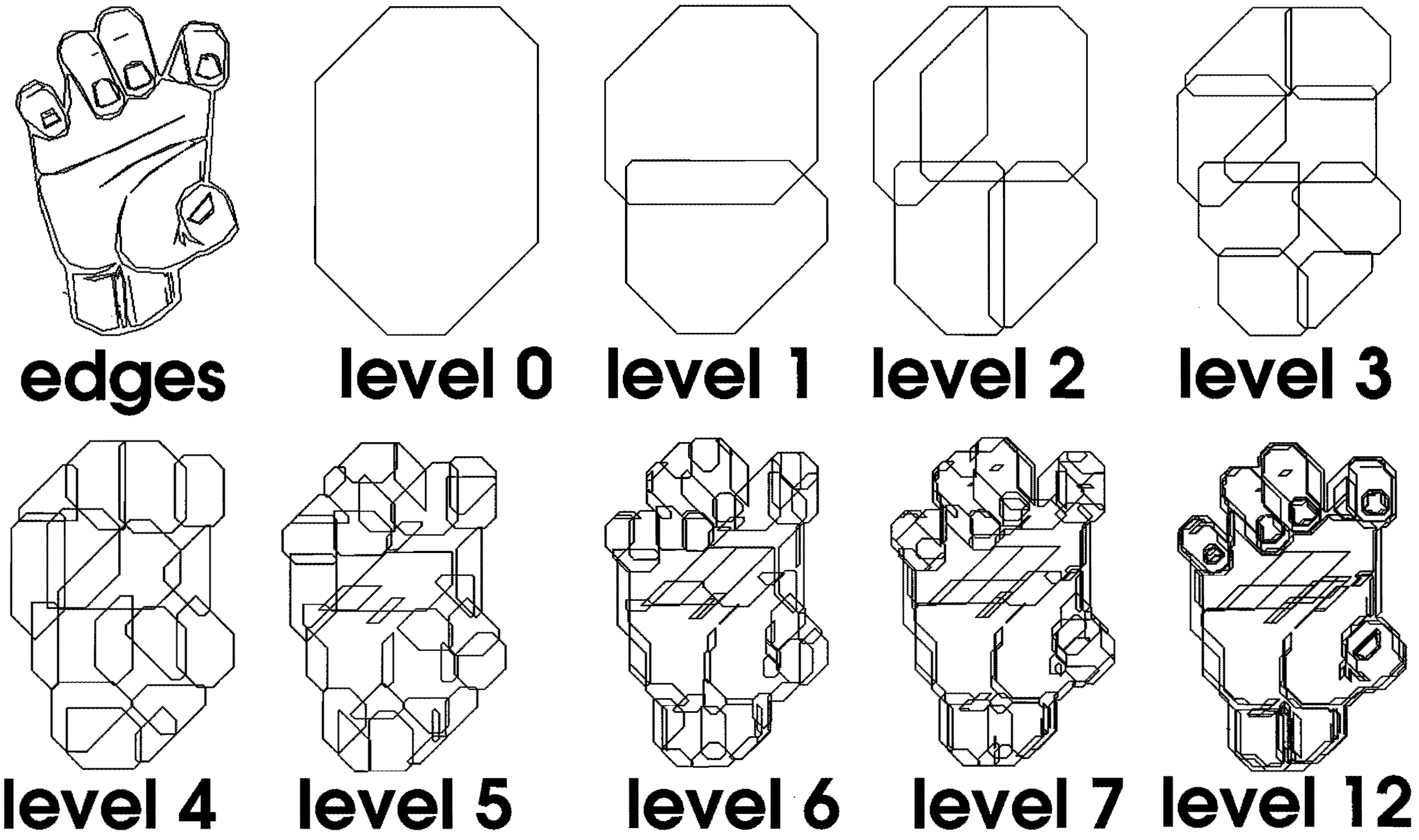
Top down



Bottom up

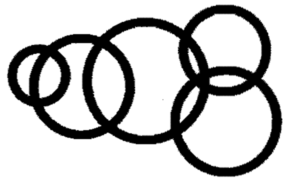
Octbox image trees

289 leaves, depth 12, average depth 8.7

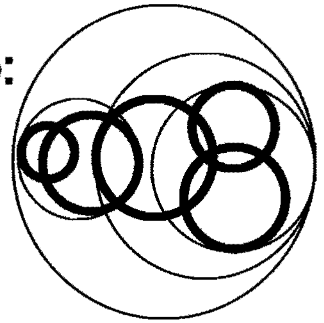


Mapping Learning with Bumptrees

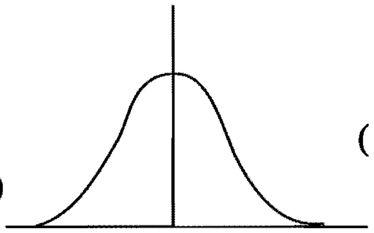
Bumps:



Bumptree:



Influence bumps: $b_i(x)$



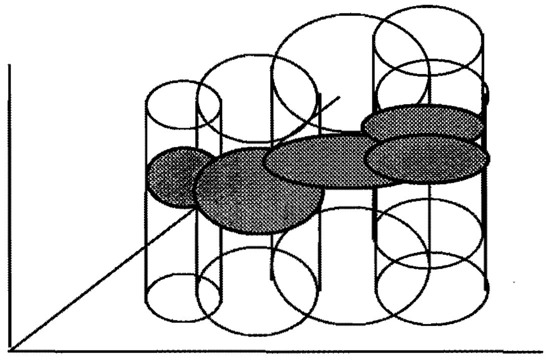
$(1-r^2)^2$

Partition of Unity: $n_i(x) = \dfrac{b_i(x)}{\sum_j b_j(x)}$

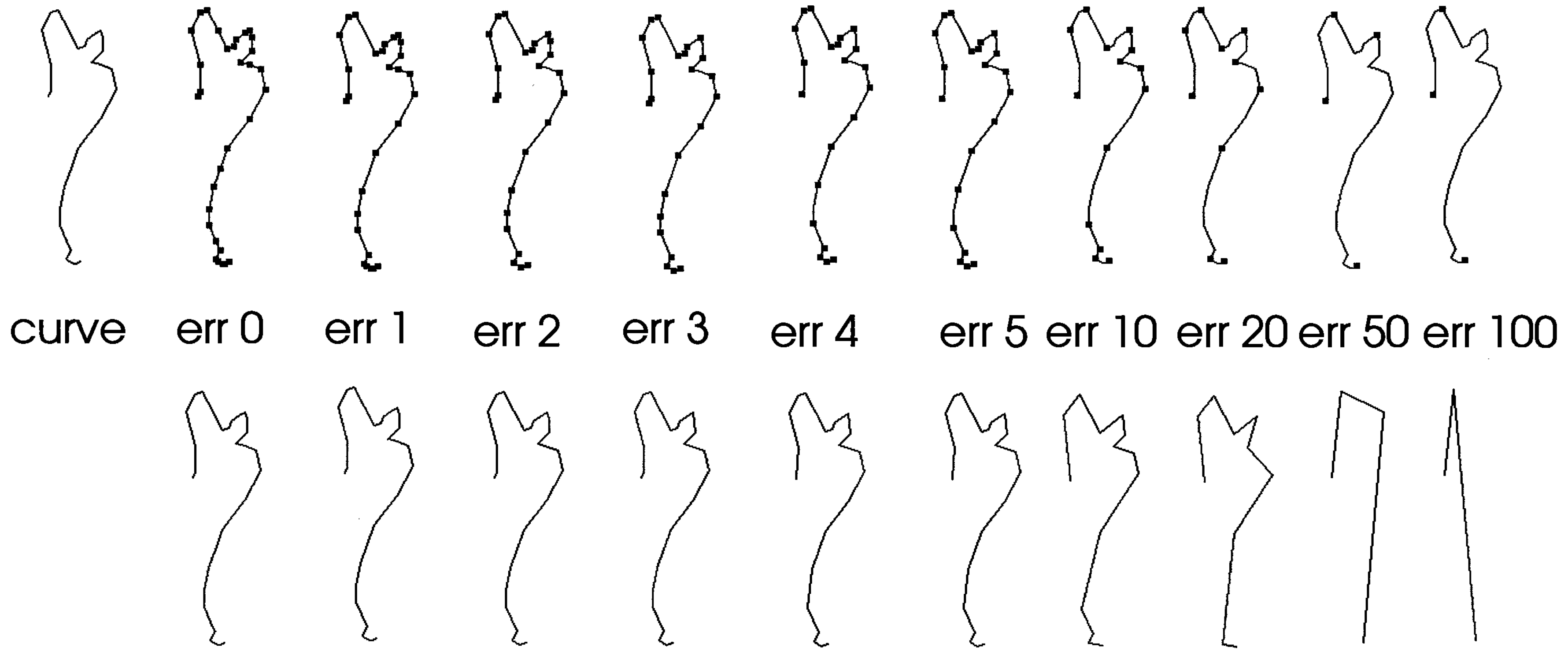
Local Models: $m_i(x)$ eg. affine models

Smoothly interpolate the models:

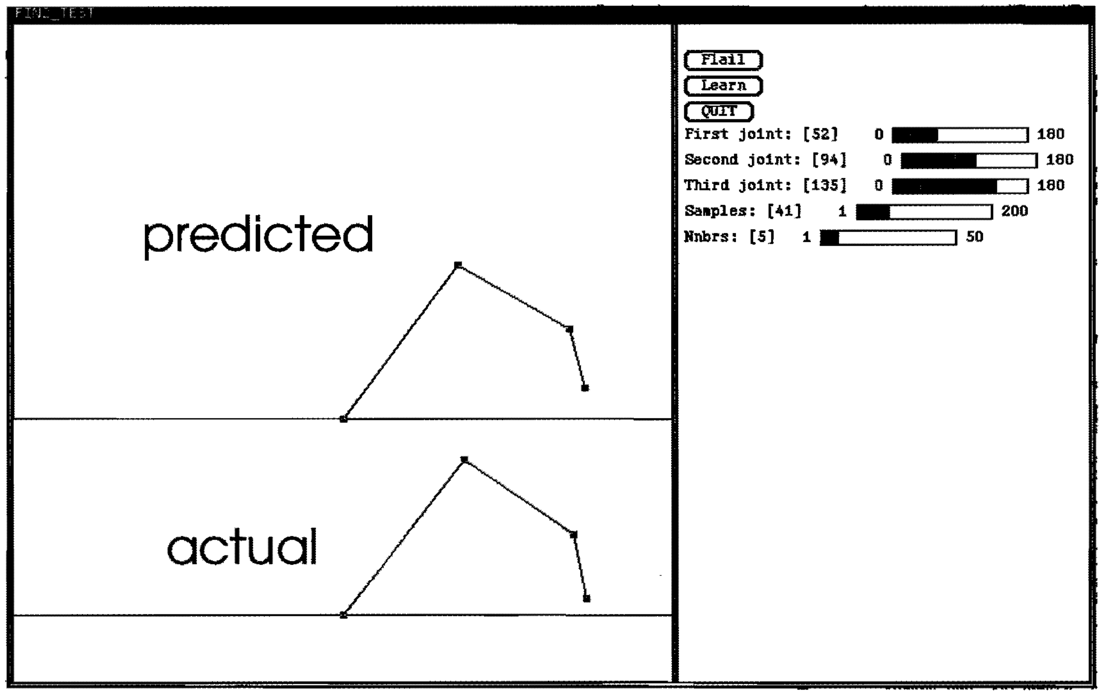
$$f(x) = \sum_i n_i(x)\, m_i(x)$$



Best first curve approximation



Learning Finger Kinematics



First joint: [52] 0 180
Second joint: [94] 0 180
Third joint: [135] 0 180
Samples: [41] 1 200
Nnbrs: [5] 1 50

Submanifold approximation (eg. learning constraint surfaces).

In addition to learning mappings, we would like to learn constraint surfaces. Mapping learning is a special case if we think of the sample points defining the graph of the function.

Determine the local dimension by doing a singular value decomposition of near neighbors of each sample. Approximate the surface with a balltree and affine subspace in each ball. Use partition of unity idea to smoothly fit the pieces together cheaply (must iterate taking combinations until convergence).

Querries:

1) Given a point return closest surface point.

2) Return smallest box containing same portion of surface as given box. (Generalizes mapping evaluation and inversion as well as partial match queries.)

Networks of Constraint Models

Can do **constraint propagation** in a network of balltree models.

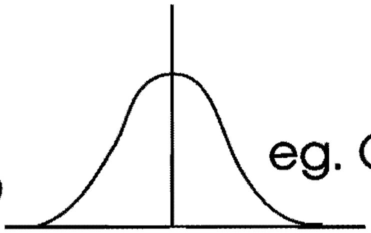
Keep volume decrease at each node in a priority queue, update in best first order.

Similarly, identify **high probability** regions in statistical case.

In the full system, both recognition and learning consist of building a network of models, inserting them in order of salience as measured in the current context, and propagating statistical influence through them.

Eg. Representing functions by smoothly gluing together local models

Influence bumps: $b_i(x)$



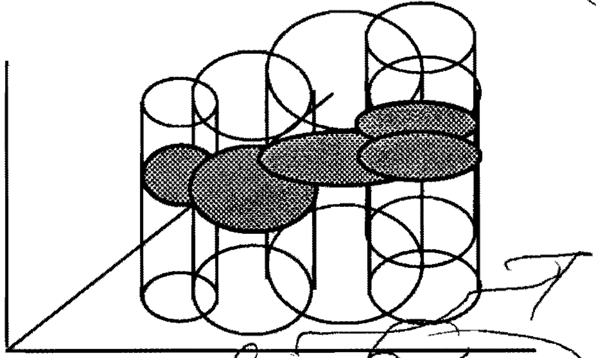
eg. Gaussians

Partition of Unity: $n_i(x) = \dfrac{b_i(x)}{\sum_j b_j(x)}$

Local Models: $m_i(x)$ eg. affine models

$f(x_i) + f'(x_i)(x-x_i)$

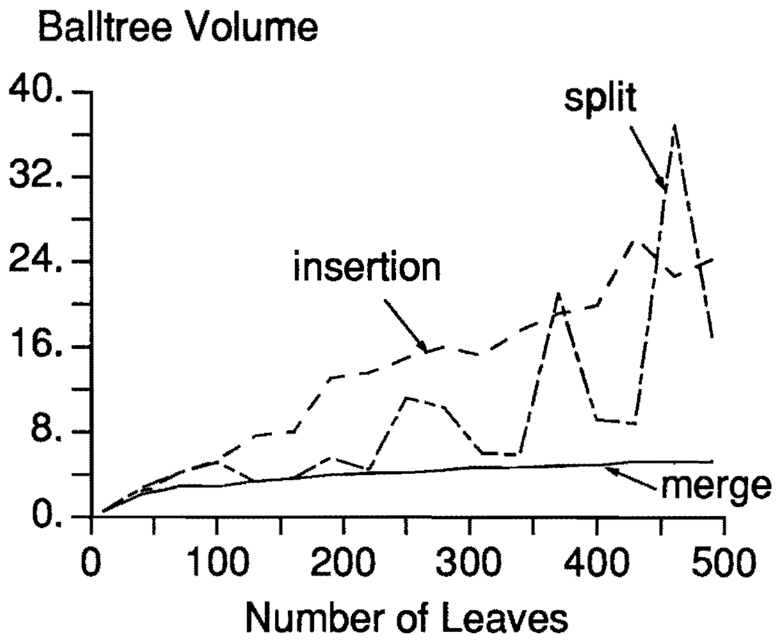
Smoothly interpolate the models:

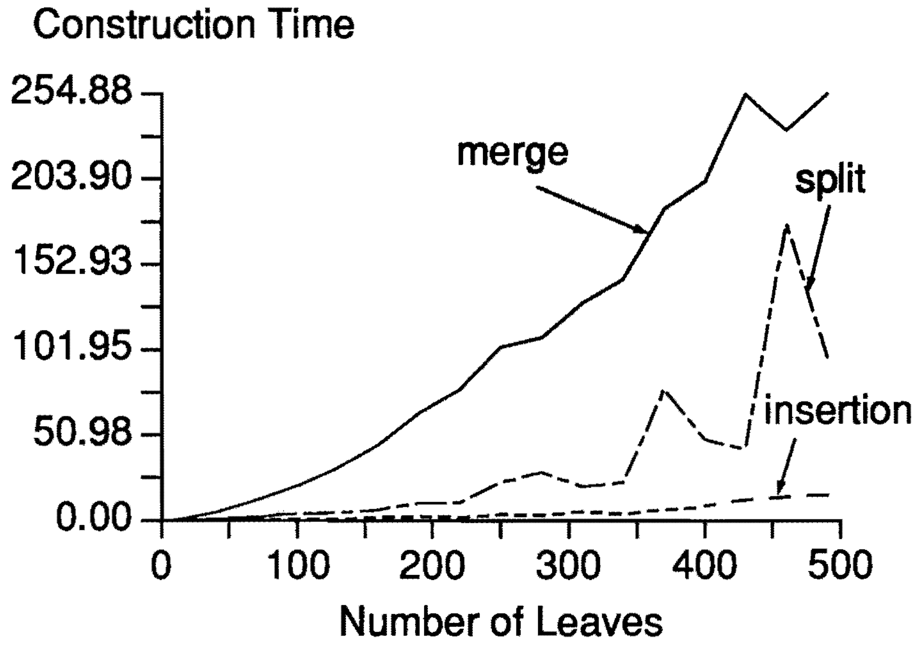


$$f(x) = \sum_i n_i(x)\, m_i(x)$$

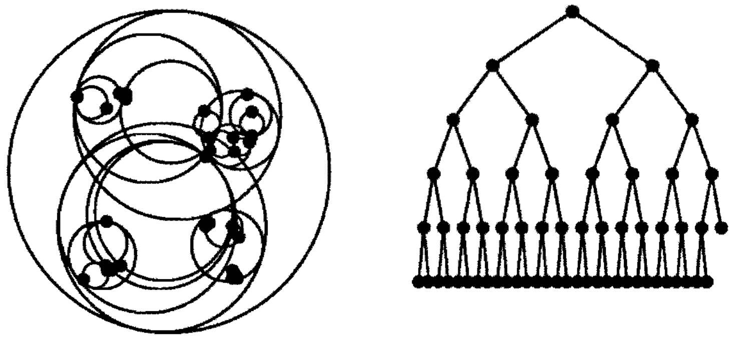
$$f(x) = \frac{f(x) \sum g_i(x)}{\sum g_i(x)} = \frac{\sum_i f(x)\, g_i(x)}{\sum g_i(x)}$$

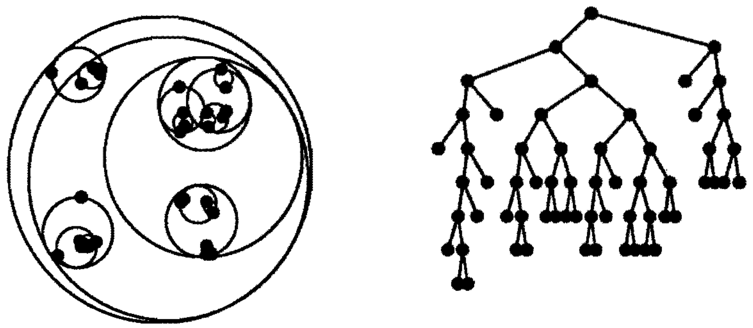
Balltree volume and construction time vs. number of 2-d uniformly distributed point leaves.

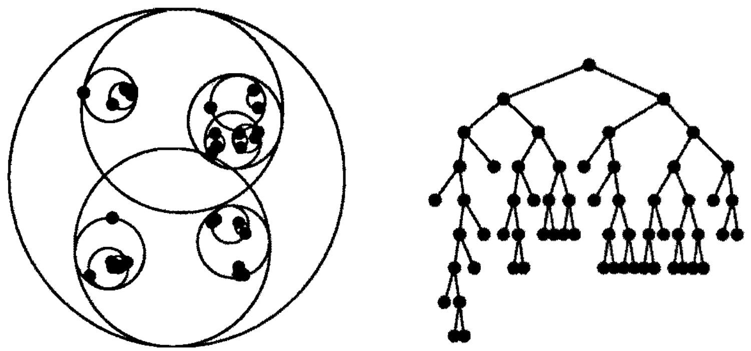


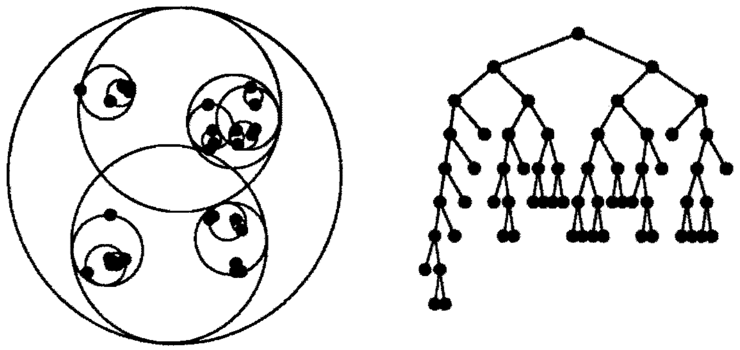


Eg. Balltrees


Kd


Insertion and
Cheap insertion

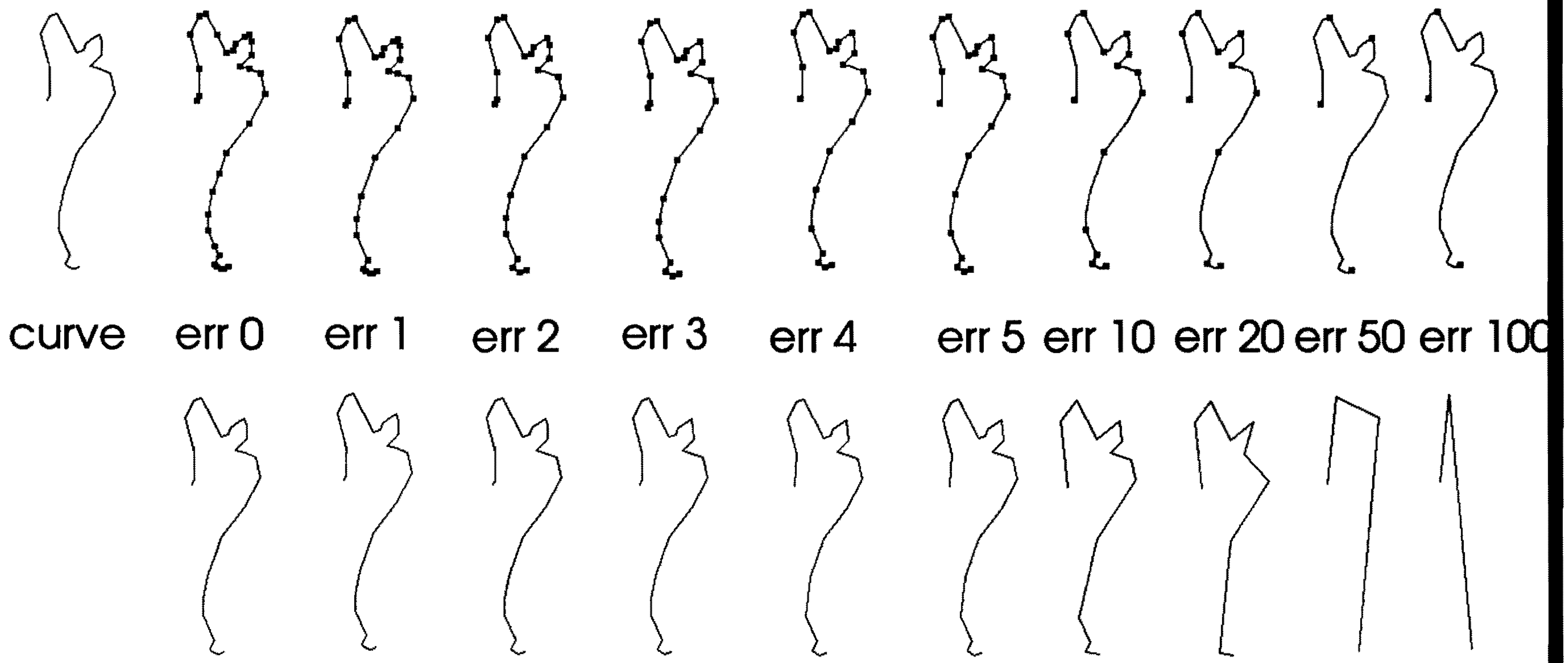

Top down


Bottom up

Eg. Construction of Adaptive Hierarchical Data Structures

- Boxtrees, Balltrees, and Bumptrees provide fast access to relevant model portions in geometric spaces.
- Must construct these structures to adapt to the represented data.
- Five different approaches have been explored: two top down, two incremental, and one bottom up.
- The bottom up merged structures tend to fit the best but are more expensive to construct.
- These structures can be used to speed up general merging algorithms by pruning away less important potential merges.

Best first merging of segments to fit a curve



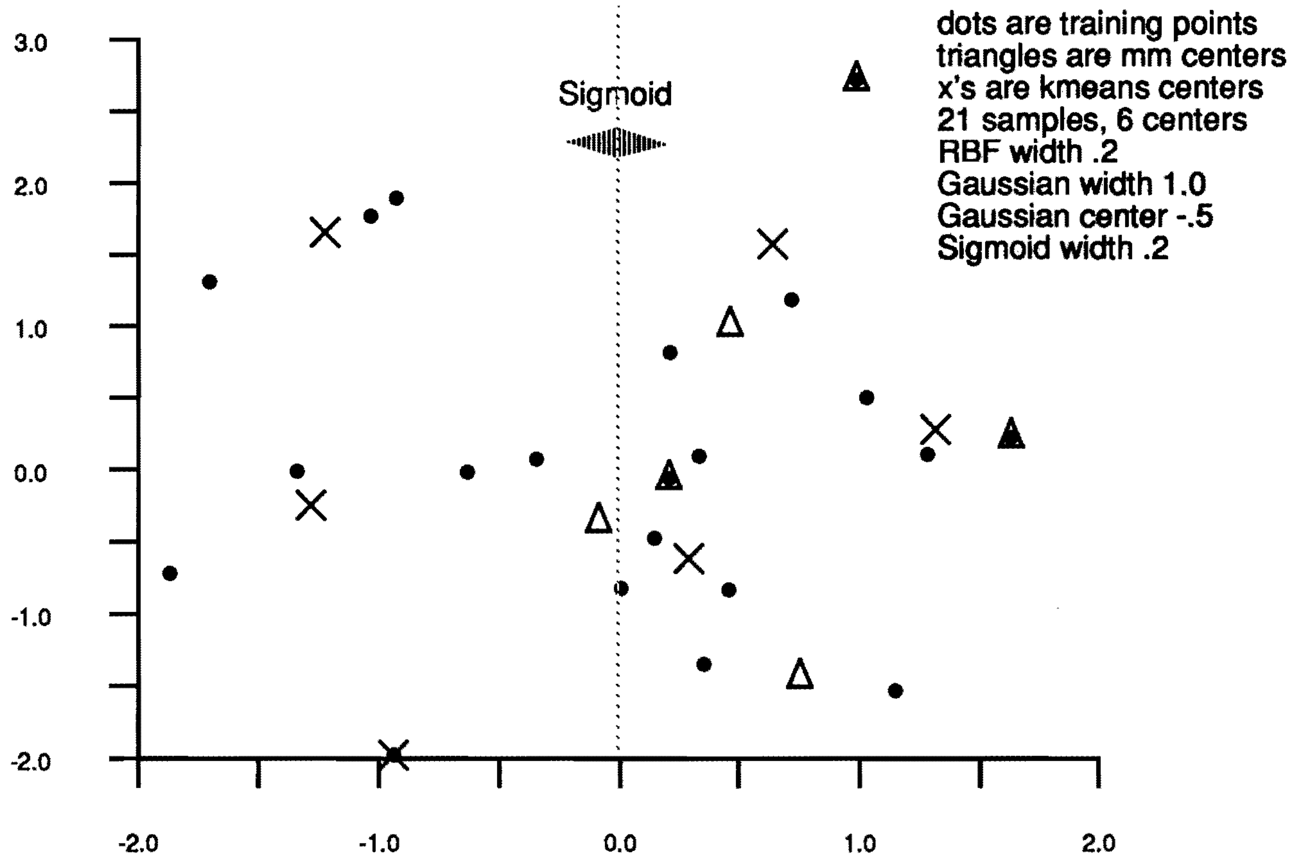
Approximating Curves and Surfaces

- Use local models to approximate curves and surfaces.
- Eg. find best set of segments approximating a given curve.

  Merge the pair of segments which increases the error the least repeatedly until an error criterion is reached.
- This best first approach does a good job of "homing" in on the "corners" and makes ambiguous decisions only after all the clear decisions have been made.
- The class of the merged model may be from a more complex family since there is more data to provide evidence.
- eg. A pair of affine segments may be fit by a quadratic.

Model Merging vs. K-means RBF centers

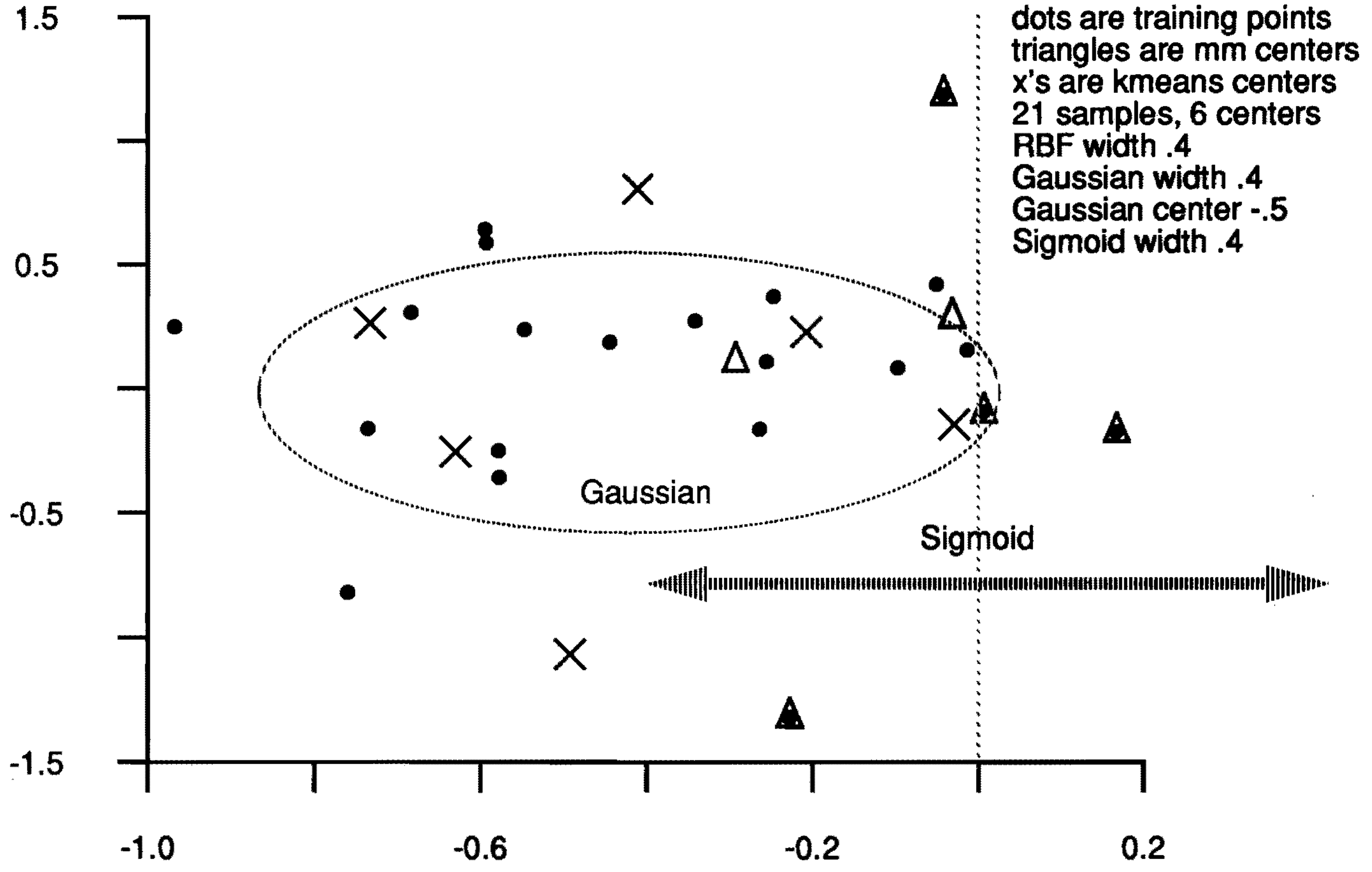
training error:	k-means=.074586,	model merging=.004326
test error:	k-means=.077387	model merging=.028974



Model Merging vs. K-means RBF centers

training error: k-means=.008098, model merging=.000604

test error: k-means=.012463 model merging=.004638



Example: Choosing Centers for Radial Basis Functions

- Generally want fewer centers than training points.
- One approach is to have more centers in higher density regions.
- Leads to using k-means or other clustering method to choose center locations.
- For best generalization would rather choose centers to best adapt to the function being approximated.
- Eg. Draw samples from a Gaussian.

Try to approximate the sigmoid: $\frac{1}{1+e^{-x/\lambda}}$

k-means puts more emphasis near the center of the Gaussian, model merging puts more emphasis along the changing sigmoid but includes the Gaussian center as well.

Best First Model Merging

- Human learning is not like backprop at all.
  small amount of data -> remember and use similarity
  large amount of data -> fit models and find regularities
- Build complex model from simple model components
- Start with one component per example
- Successively merge models always doing the best first
- Stop when error is not made up for in reduction of complexity
- The merged models may be from a more complex class than the merged components because there is more data available.
- eg. Density estimation: Gaussian mixture kernel estimate, successively more complex components
- eg. Mapping learning: Start with local affine fits, move to quadratic, etc.
- eg. Grammar learning: Start with products, move to regular, context free, etc.

Model Sparseness and Locality

- Practical situations are often governed by sparseness and locality.
- Both models and data decompose into components which interact with one another sparsely if at all.
- Data components are often only strongly determined by a small number of "local" model components.
- Allows us to adaptively choose the complexity of the model components based on the data which is relevant to it.

- *Computational efficiency:* Build access structures which allow us to compute only those model components actually relevant to the data.
- *Statistical efficiency:* Dynamically choose model component complexity locally via model merging.

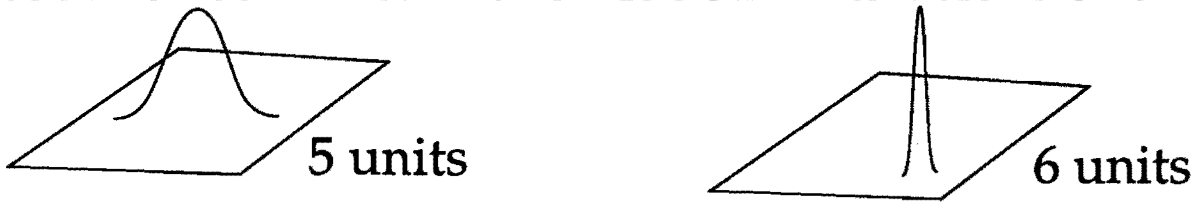
Vapnik and Nested Families

- Vapnik computes number of samples needed to get bounds on error in selecting distributions from families.
- The larger the family, the more samples are needed in general.
- If we choose too small a family, we may not be able to fit the data, if too large, we may need to much data to validate the choice.

- Vapnik suggests that we work with a nested family of models, trying to fit in the small families first, working our way up.
- The number of samples needed to validate a model depends only on the size of the space it is from, not on the size we potentially could have gone to.

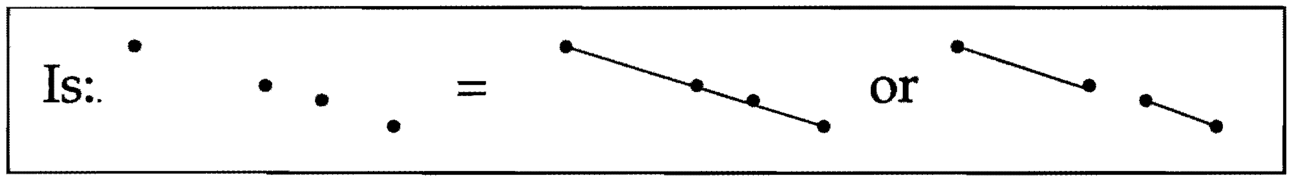
Bayesian Occam's Razor

Eg. choose between a net with 5 hidden units versus one with 6.



6 might give a better fit and so have higher posterior, but because space is larger, the integral of the posterior might be larger over 5. So MAP picks the wrong dimension. "Overfitting"

Same issues apply to perception as learning:

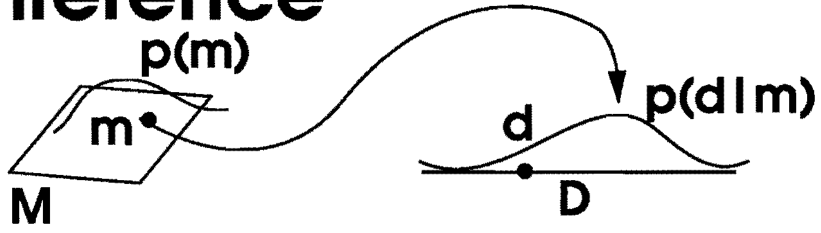


Goodness of fit loses validity because of the large number of possibilities we could have picked from.

Statistical and Computational Efficiency

- Too computationally expensive to do full averages over posterior, so heuristic and computational techniques are essential
- Approximations like MAP can lead to inefficient use of data: ***Overfitting***
- Occam's Razor
- Vapnik-Chervonenkis Dimension
- Akaike Information Criterion
- Rissanen's Minimum Description Length Criterion
- Weight decay and complexity terms in error functions
- Cross validation, bootstrap, jacknife methods
- ....

Bayesian Inference



Space of **models** M with **prior p(m)**

Space of **data** D,

Data d leads to **likelihood** of model m of **p(d|m)**

Baye's theorem gives the **posterior**

$$p(m|d) = \frac{p(d|m)\,p(m)}{p(d)}$$

$$\boxed{\textbf{posterior} \quad \propto \quad \textbf{likelihood} \quad \times \quad \textbf{prior}}$$

To make decisions, need **utility function U(a,m)** for taking **action** a when the true model is m.

The **Baye's decision rule** says choose the action a which maximizes the **expected utility** with respect to the posterior:

$$max_a \int_M U(a, m)\, p\,(m|d)\, dm$$

Recognition and Learning

- Inductively choosing models from data
  models are probability distributions

- Rational decision making is Bayesian
  i.e. exists prior and utility that reproduce behavior of rational agent

- ***Recognition:*** sensation -> perception

- ***Learning:*** previous history -> model priors

- The prior for recognition is the posterior for learning.

Model Merging for Improved Generalization

Stephen M. Omohundro

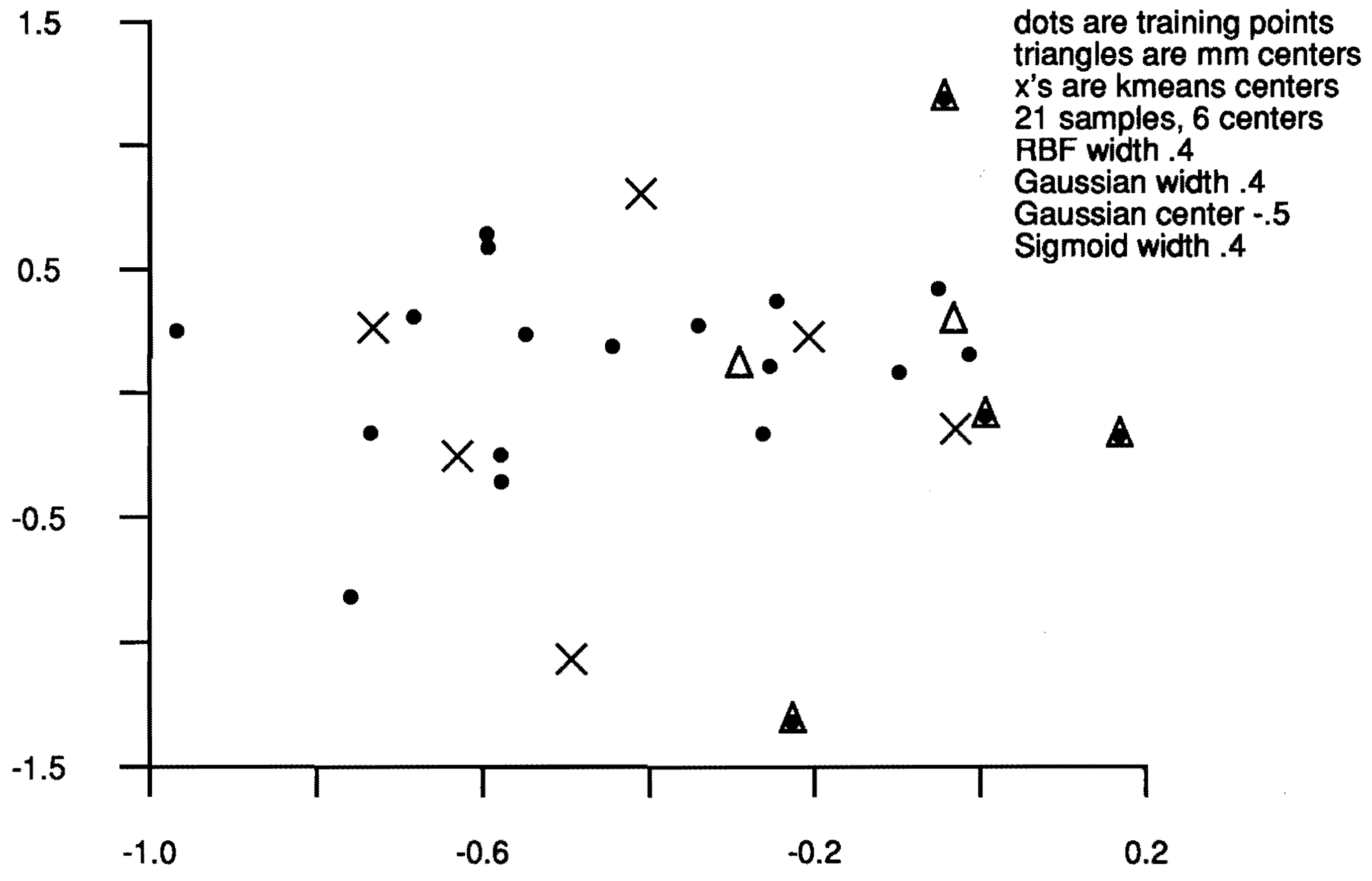
International Computer Science Institute

Berkeley, California

In this paper we describe a general technique which we call "best-first model merging" for dynamically choosing the structure of a neural or related architecture. The approach addresses a number of issues including: determination of the model complexity for a given set of examples, avoidance of overfitting, improved generalization, and efficient computational implementation. Empirical examples include: choosing a better basis for function learning with radial basis functions, classification using localized receptive field units, constraint learning based on smoothly interpolated piecewise-linear affine planes, curve approximation by segments, and construction of efficient balltrees and bumptrees.

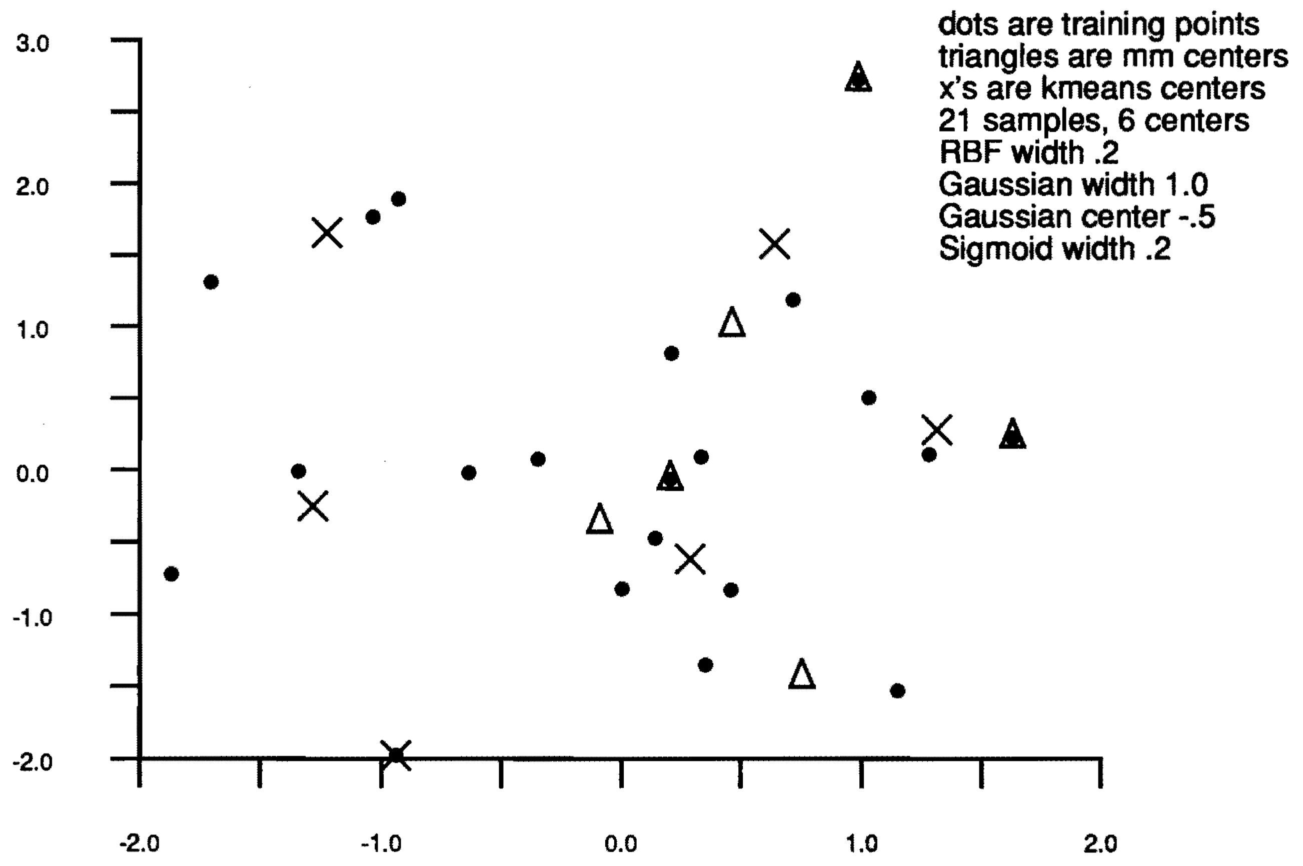
Model Merging vs. K means RBF centers

training error:	kmeans=.008098,	model merging=.000604
test error:	kmeans=.012463	model merging=.004638



Model Merging vs. K means RBF centers

training error: kmeans=.074586, model merging=.004326
test error: kmeans=.077387 model merging=.028974


dots are training points
triangles are mm centers
x's are kmeans centers
21 samples, 6 centers
RBF width .2
Gaussian width 1.0
Gaussian center -.5
Sigmoid width .2