

NONPARAMETRIC CLASSIFICATION TECHNIQUES

C&PE 940, 30 November 2005

Geoff Bohling
Assistant Scientist
Kansas Geological Survey
geoff@kgs.ku.edu
864-2093

Overheads and other resources available at:

<http://people.ku.edu/~gbohling/cpe940>

Modeling Categorical Variables: Classification

In classification applications we are trying to develop a model for predicting a categorical *response* variable, G , from one or more *predictor* variables, X . That is, if we know that an observation arises from one of K different mutually exclusive classes or groups, G_k , then we are trying to estimate the probability of occurrence of each group at each point in the predictor space

$$\hat{P}_k(x) = \text{Prob}(G_k | X = x)$$

We could then assign each estimation point to the class with the highest probability at that point, segmenting the predictor space into regions assigned to the different classes.

The fact that the probabilities are continuous variables immediately suggests the possibility of using the nonparametric regression techniques we discussed last time to model them. However, we will have to apply transforms to ensure that the estimated $P_k(x)$ values represent a legitimate set of probabilities for mutually exclusive events, meaning

$$0 \leq P_k(x) \leq 1, \quad k = 1, \dots, K$$
$$\sum_{k=1}^K P_k(x) = 1.$$

To use this approach, the group memberships in the training dataset are coded as a set of indicator variables, one for each group, with $y_{k,i} = 1$ for the group to which data point i belongs and 0 for all other groups – the set of probabilities corresponding to our certain knowledge of the group membership for each training data point.

Another approach to the classification problem is to model the probability density function for each group, $f_k(X)$, and then plug the density estimates (along with those pesky prior probabilities, q_k) into Bayes' theorem to get

$$\hat{P}_k(x) = \frac{\hat{f}_k(x)q_k}{\sum_{j=1}^K \hat{f}_j(x)q_j}$$

In this case we automatically get a legitimate set of probabilities, but our models for the probability density functions should obey the constraints

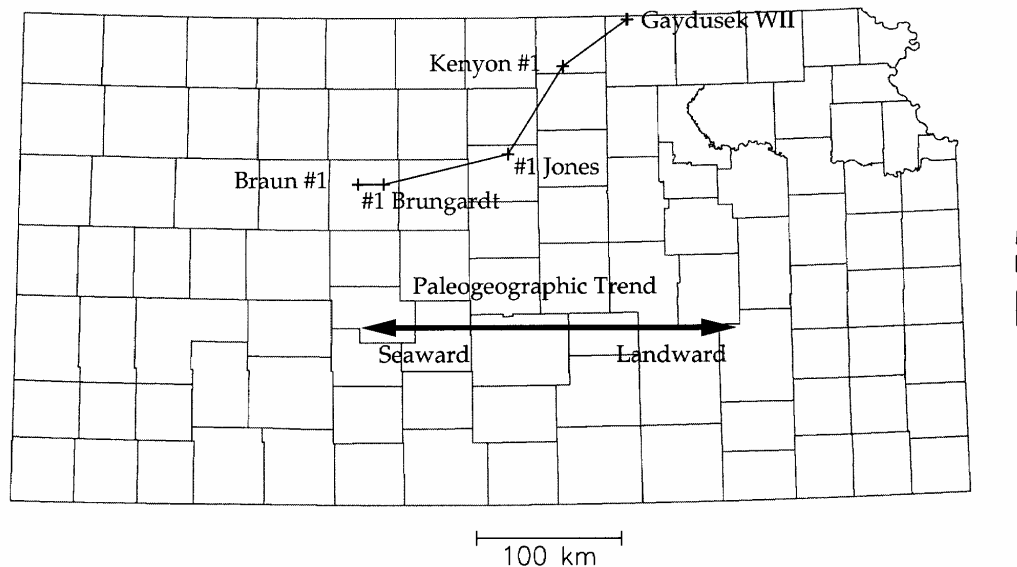
$$f_k(x) \geq 0$$
$$\int f_k(x)dx = 1$$

We can employ a variety of nonparametric density estimation techniques to accomplish this task. Not surprisingly, these techniques are closely related to nonparametric regression techniques, with counts of data points taking the place of the continuous-valued response variable.

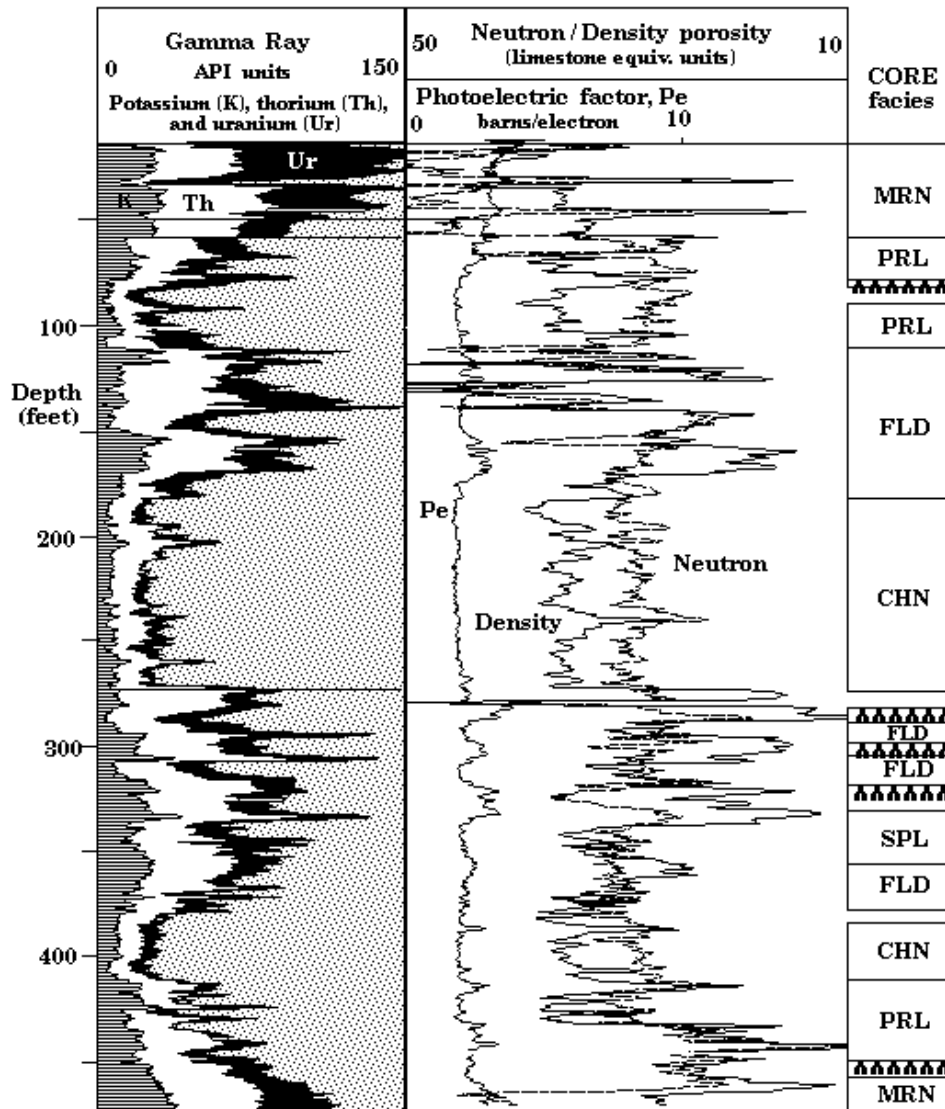
We will look at a few procedures following both of these approaches – modeling density functions or directly modeling probabilities. For the density function based approaches, we will assume that the prior probabilities are equal so that they cancel out in Bayes' formula. Again, we will only consider supervised learning using a training dataset with known group memberships and will assume that the X values are known without error.

Example Data

We will look at predicting facies from logs for a Cretaceous section in north central Kansas:



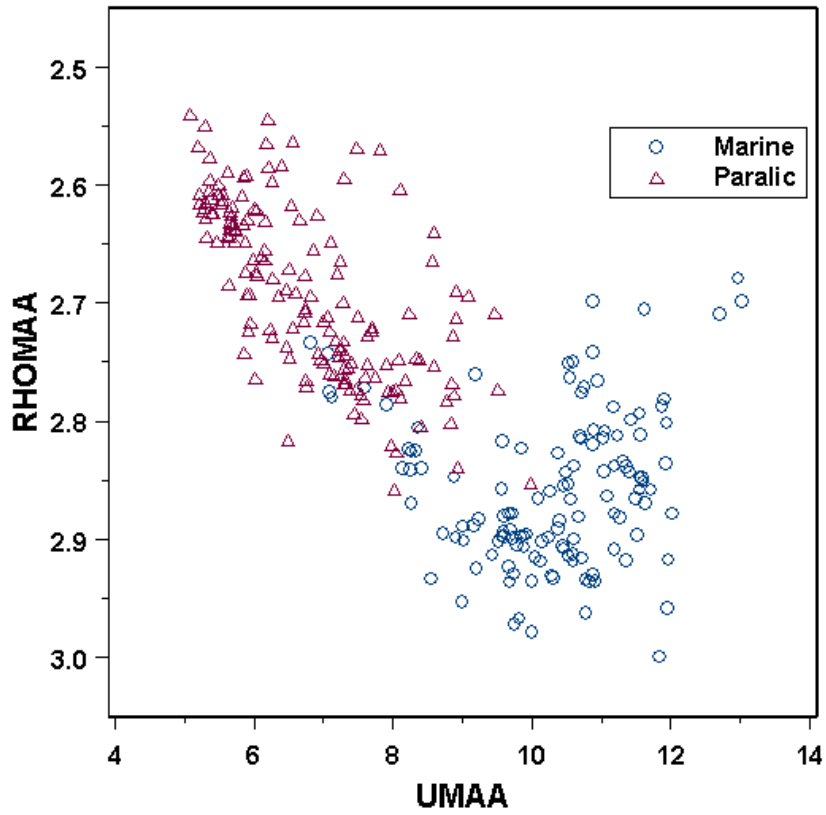
Facies assignments from core are available from the Jones well, along with a suite of logs including neutron and density porosity, photoelectric factor, and thorium, uranium and potassium components of the spectral gamma ray log. We will recast the density porosity and apparent matrix density (R_{homa}) and the photoelectric factor as apparent matrix volumetric photoelectric absorption (U_{maa}), so that the six logs employed for discrimination are: Th, U, K, R_{homa} , U_{maa} , and ϕ_N . The six facies picked from core are marine, paralic, floodplain, channel, splay, and paleosol.



Core facies interpretation key:
 MRN = marine; PRL = paralic; FLD = floodplain;
 CHN = channel; SPL = splay; ▲▲▲ = paleosol

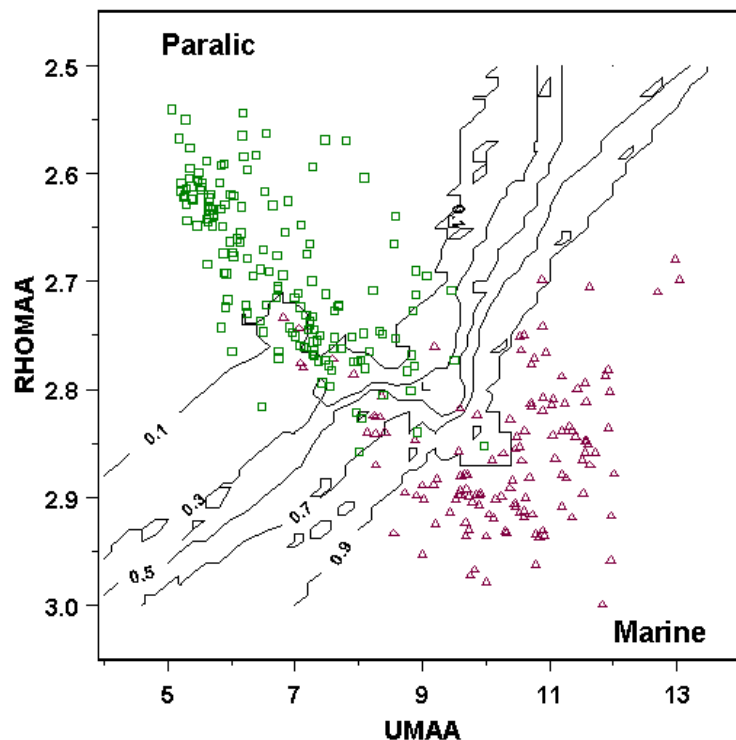
So, we will train on this data from the Jones well and look at predictions both in the Jones well and the Kenyon well.

For the sake of illustration, we will also look at a two-dimensional, two-group sub-example, trying to discriminate marine and paralic facies:



Nearest-Neighbor Classification

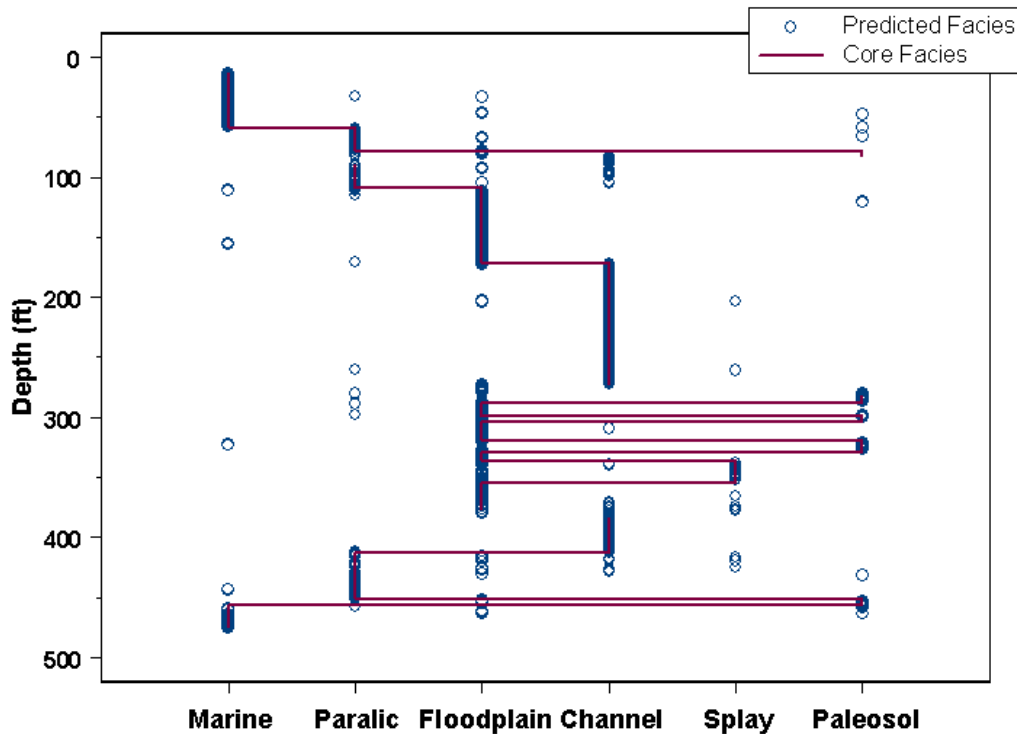
This is basically the same as nearest neighbor averaging, except that instead of taking the average of y values at neighboring points, we are assigning the class at each point based on a majority vote of the class memberships at neighboring points. Alternatively, one can compute a vector of group membership probabilities by dividing the count for each group by the total number of neighboring points. As usual, we would want to scale the predictor variables to comparable ranges before computing distances to neighboring points. Applying this approach to the two-facies Rhomaa-Umaa example and predicting over a grid of Rhomaa-Umaa values using 15 nearest neighbors, we get the following map of probability of membership in the marine facies:



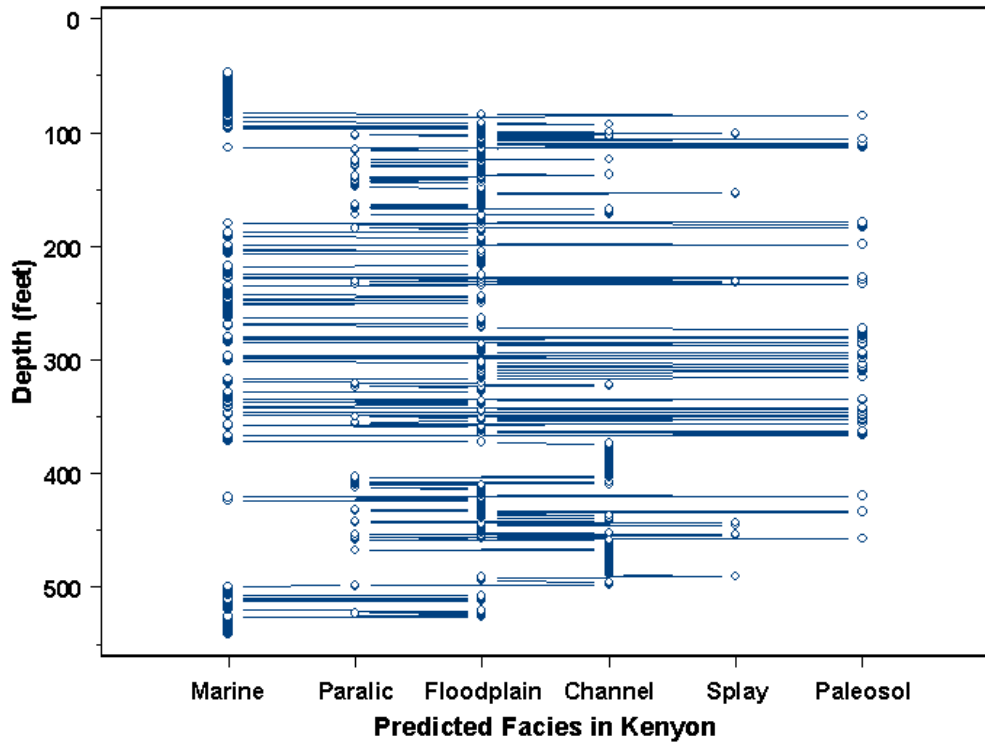
Applying nearest-neighbor classification with 20 neighbors to the entire Jones well dataset, using all six logs and predicting all six facies, we get the following table of predicted (columns) versus actual (rows) facies:

Core Facies	Predicted Facies					
	Marine	Paralic	Floodplain	Channel	Splay	Paleosol
Marine	112	2	8	0	0	6
Paralic	2	110	19	19	3	3
Floodplain	5	7	206	7	4	3
Channel	0	1	5	254	2	0
Splay	0	0	21	2	15	0
Paleosol	2	5	25	0	0	34

Overall, 82% of the intervals are classified correctly. The predicted facies versus depth look a little noisy:



The predicted facies versus depth in the Kenyon well are very ratty:



Kernel Density Estimation

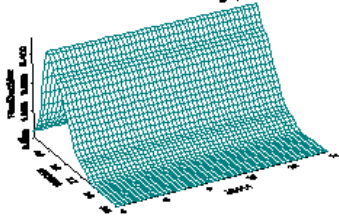
This basically uses the same procedure as kernel regression, using kernel basis functions to produce smooth estimates of the group-specific density functions, $\hat{f}_k(X)$. Here the kernel functions are not serving to smooth out the y values, but are in a sense spreading out the location of each training data point in predictor space, turning a delta (spike) function at its exact location into a Gaussian (or similar) curve centered at that location. The kernel density estimate at each location is basically a sum of the surrounding kernels rescaled to represent a density estimate. In general, we would have to estimate fully multivariate density functions, using high-dimensional kernel functions. However, if we make the simplifying assumption of independence among the variables, then the multivariate density function for each group is given by the product of the density functions of the individual variables for that group:

$$f_k(\mathbf{X}) = f_k(X_1) \cdot f_k(X_2) \cdots f_k(X_d)$$

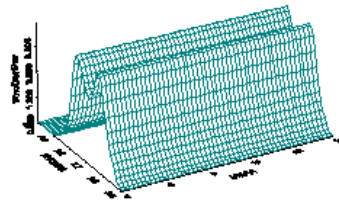
This simplification means that we only need to develop one-dimensional density function estimates. Plugging these density estimates into Bayes' theorem leads to a method called "naïve Bayes".

Applying the naïve Bayes approach to the Rhomaa-Umaa example, we get the following density estimates for each group – multiplying the one-dimensional kernel density estimates along each axis to get the combined density estimates.

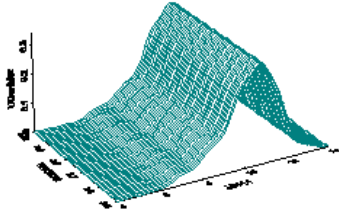
Rhomaa Density, Marine



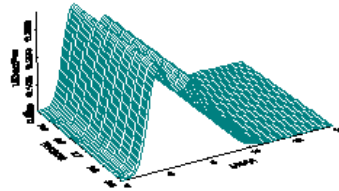
Rhomaa Density, Paralic



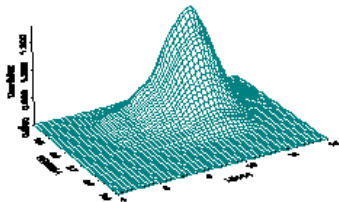
Umaa Density, Marine



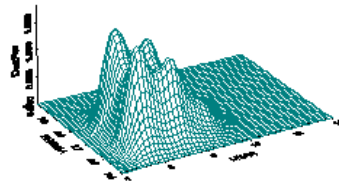
Umaa Density, Paralic



Combined Density, Marine

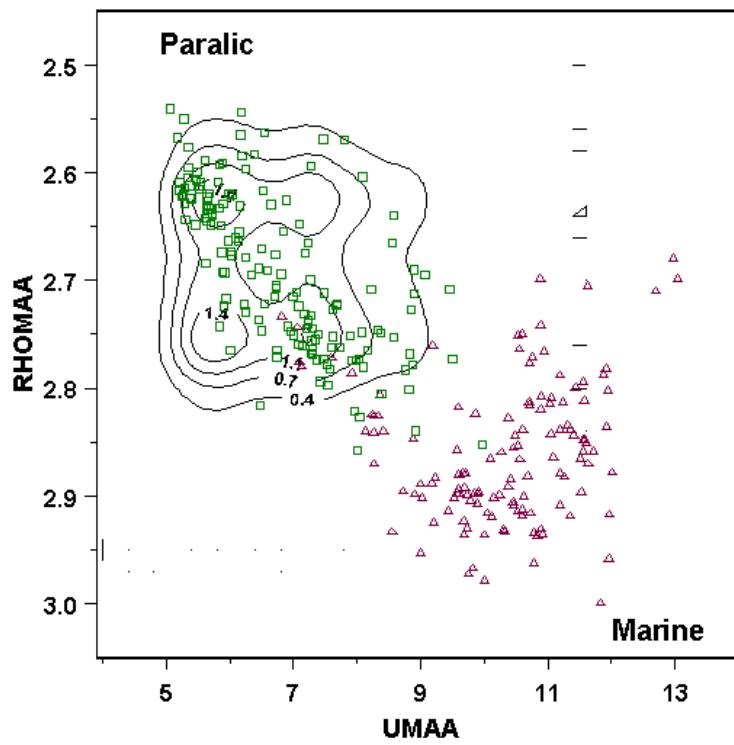
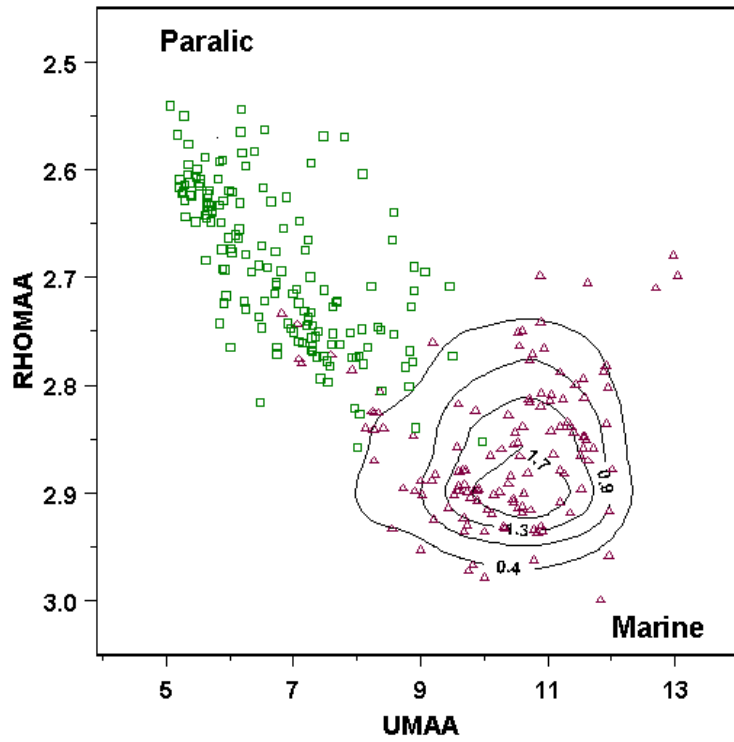


Combined Density, Paralic

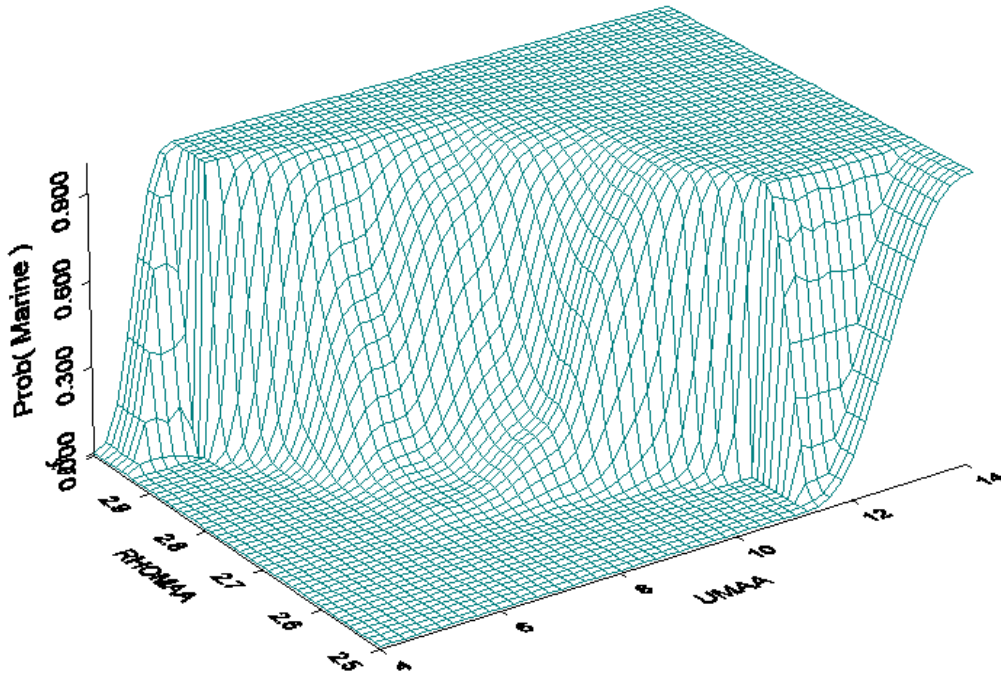


Note that the Rhomaa axis in these 3D plots is pointed in the opposite direction than the Rhomaa axis in the 2D plots, so the marine cluster is plotting towards the back rather than towards the front like it should.

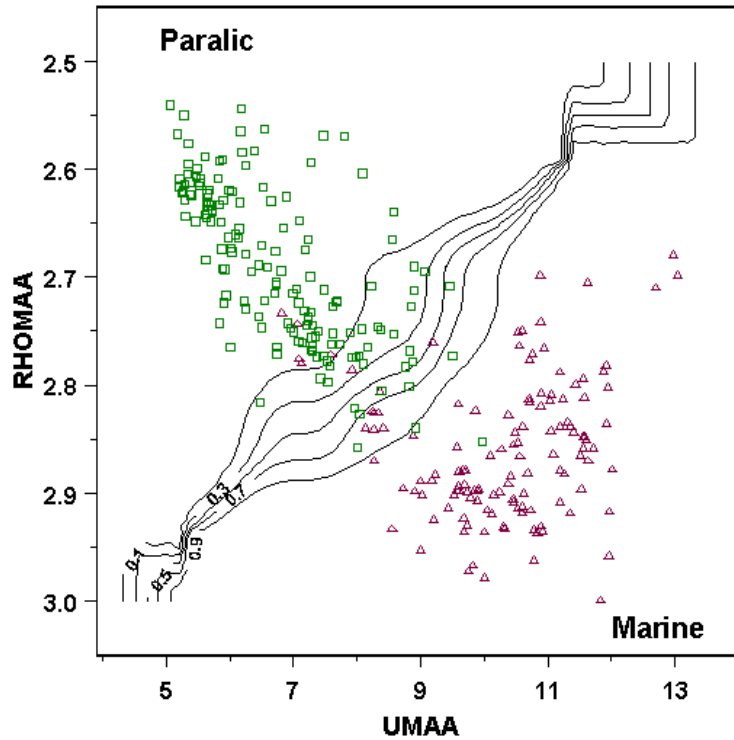
As contour plots, the two densities look like:



Combining the density estimates for the two groups in Bayes theorem give the following probability for the Marine facies:



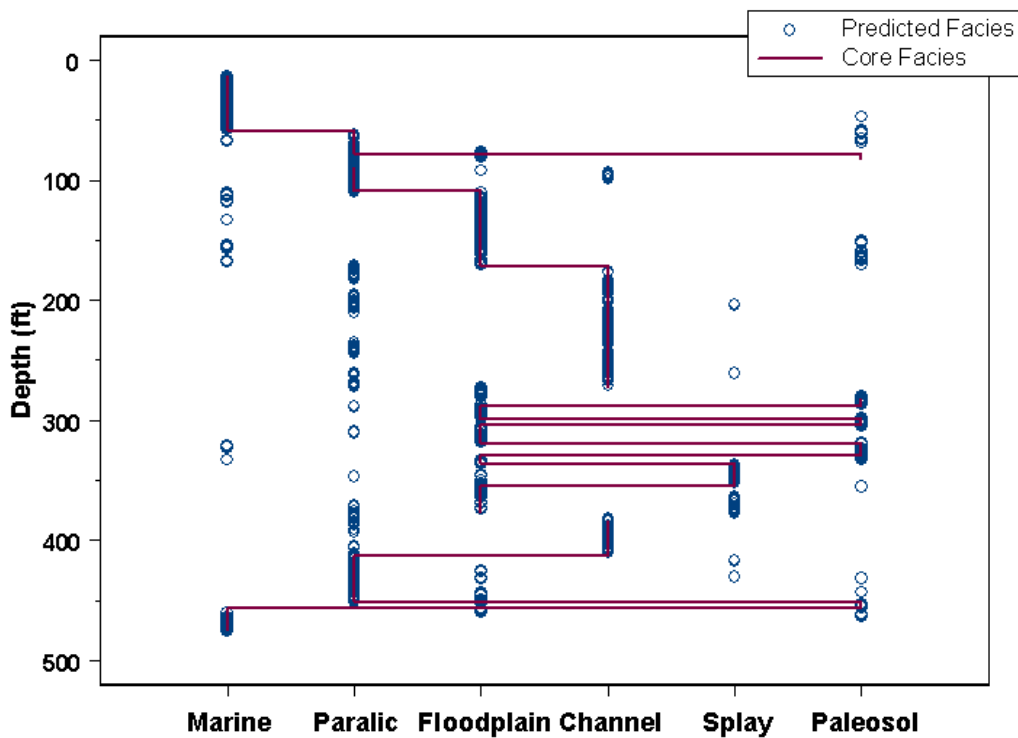
Or, as a contour plot (oriented the right way):



Applying the naïve Bayes procedure to the full Jones dataset we get the following table of results, with a 72% correct classification rate overall:

Core Facies	Predicted Facies					
	Marine	Paralic	Floodplain	Channel	Splay	Paleosol
Marine	116	0	6	0	0	6
Paralic	4	113	18	8	3	10
Floodplain	20	13	156	0	16	27
Channel	0	84	1	174	3	0
Splay	0	1	9	0	28	0
Paleosol	4	5	10	0	0	47

The predicted facies versus depth look like:



Neural Networks

For classification problems we can use essentially the same structure of neural network as we used for continuous-variable modeling. The difference is that we now have K output values, T_k , which we transform to probabilities using the *softmax* transfer function:

$$\hat{P}_k(x) = \frac{\exp(T_k(x))}{\sum_{j=1}^K \exp(T_j(x))}$$

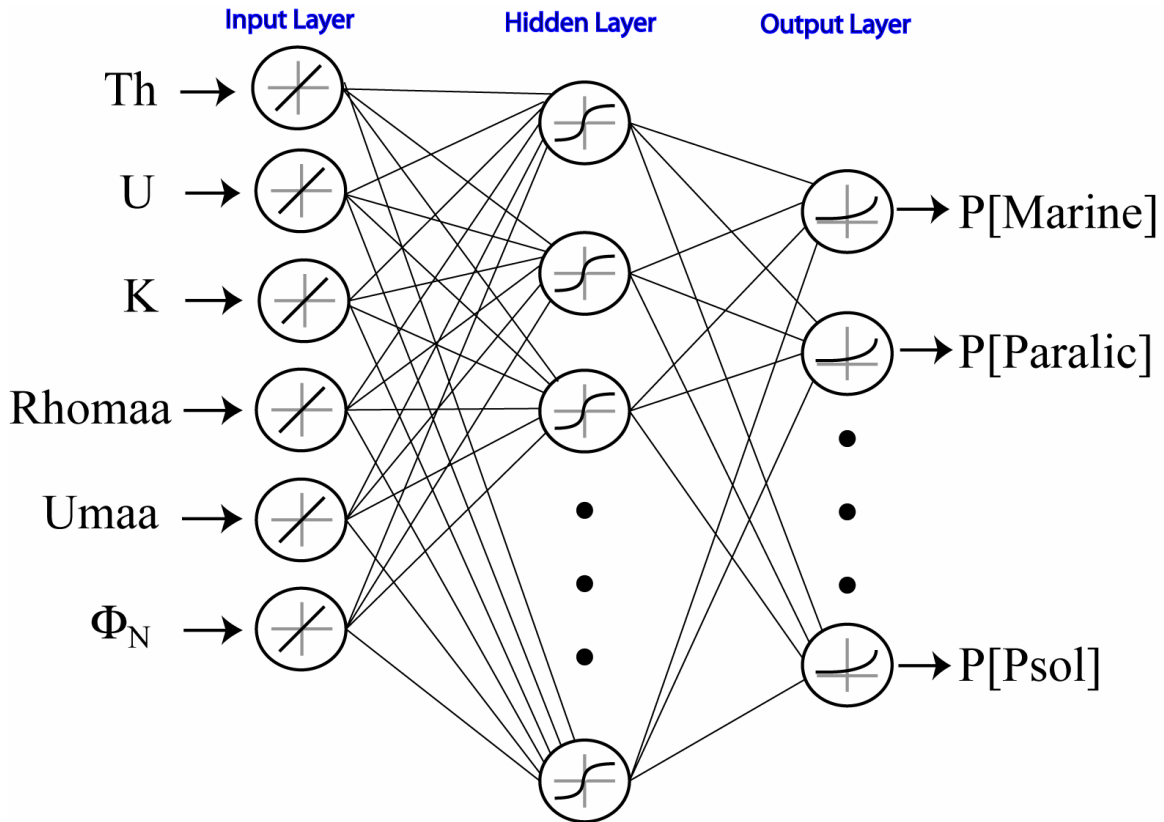
This transform ensures that we get non-negative probabilities that sum to 1.

To train the network, we adjust the weights so that the estimated probabilities match the group indicator values, $y_{i,k}$, as closely as possible for the N training data points. Although it is possible to use a least-squares objective function in this case, it is more common to use the following *cross-entropy* objective function:

$$R(\mathbf{a}, \mathbf{b}) = -\sum_{i=1}^N \sum_{k=1}^K y_{i,k} \log(P_k(x_i; \mathbf{a}, \mathbf{b}))$$

The first sum is over the training data points and the second is over the groups for each data point. Because the group indicators, $y_{i,k}$ are either 0 or 1, this objective function is really just the sum of the negative logarithms of the predicted probabilities associated with the actual class for each training data point. Because $-\log(P_k)$ goes to zero as P_k approaches 1 and goes to infinity as P_k approaches 0, minimizing the cross-entropy objective function tends to drive the predicted probabilities for the observed classes towards 1.

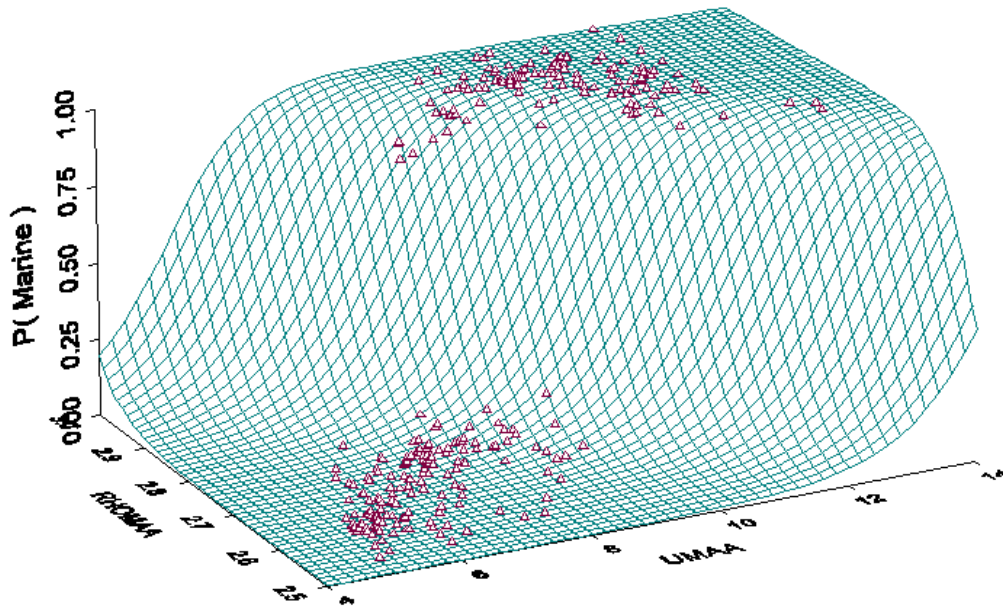
Here is a schematic representation of the classification network for our example (leaving the bias nodes out of the picture):



The exponential curves in the output nodes represent the softmax transform.

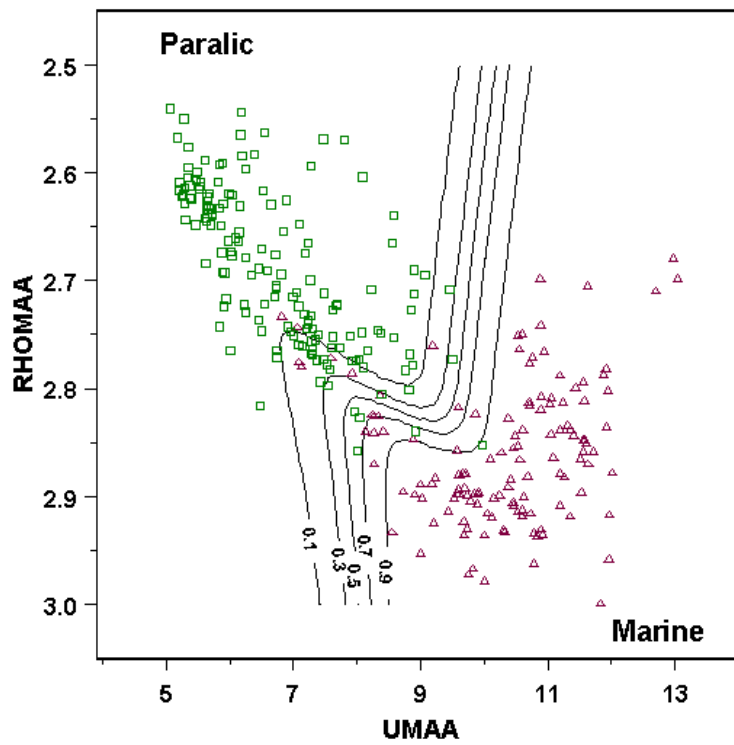
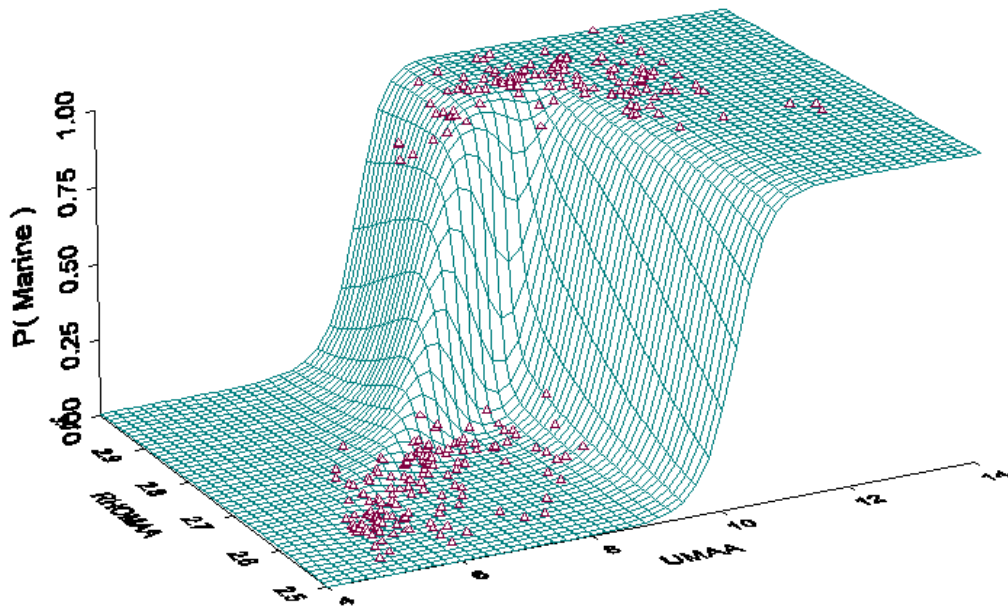
For d input variables, M hidden-layer nodes ($M+1$ including the bias node), and K output classes, the number of weights in this network is $M \cdot (d + 1) + K \cdot (M + 1)$. Again, we can use any of a number of optimization algorithms to adjust the weights. And, just as for the regression problem, we can include a weight decay term in the objective function, forcing a smoother representation of the boundaries between classes than we would obtain otherwise.

Using a single hidden-layer node for the Rhomaa-Umaa example, the sigmoid basis simply forms a step or boundary between the paralic and marine data points. I've included those points on the plot in their indicator form – 0's for paralic and 1's for marine:



Since the classification problem is about drawing boundaries, the sigmoid basis functions are in a sense more natural to this problem than to the regression problem, since the step in the basis function forms a boundary.

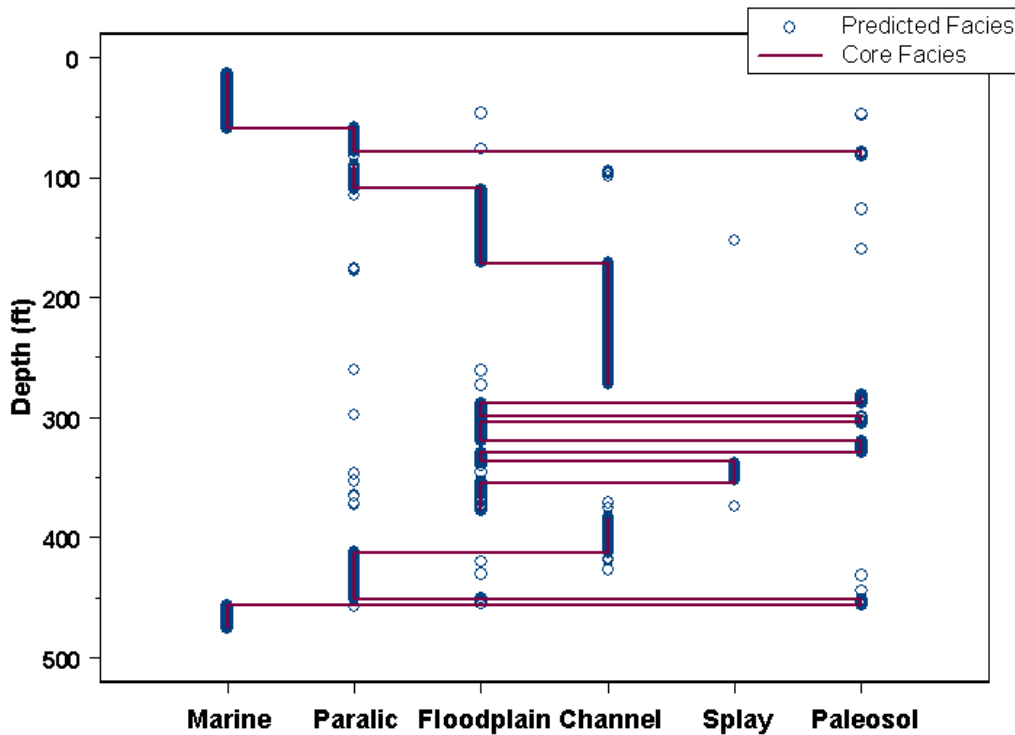
Fitting five basis functions with a decay constant of 0.01 yields:



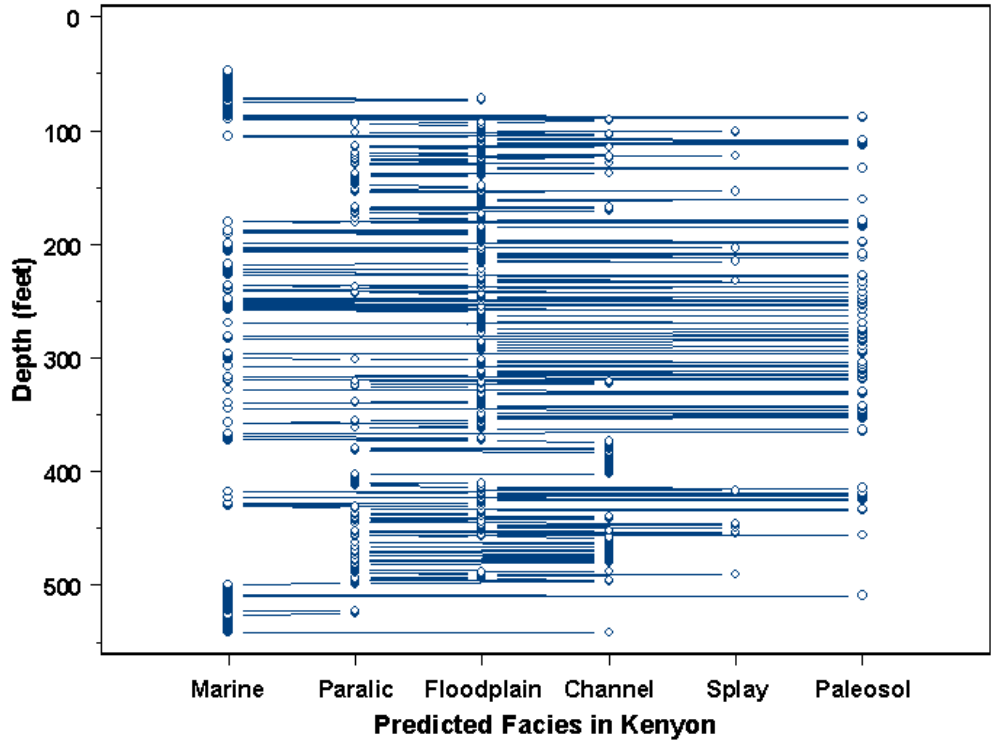
Fitting a network with 20 hidden-layer nodes to the entire Jones dataset using a decay constant of 0.1 (just guessing at reasonable values for the tuning parameters), yields the following classification table, with 93% of the facies predicted correctly:

Core Facies	Predicted Facies					
	Marine	Paralic	Floodplain	Channel	Splay	Paleosol
Marine	124	1	1	0	0	2
Paralic	0	140	5	9	0	2
Floodplain	0	9	216	2	2	3
Channel	0	4	2	256	0	0
Splay	0	2	8	0	28	0
Paleosol	0	2	7	0	0	57

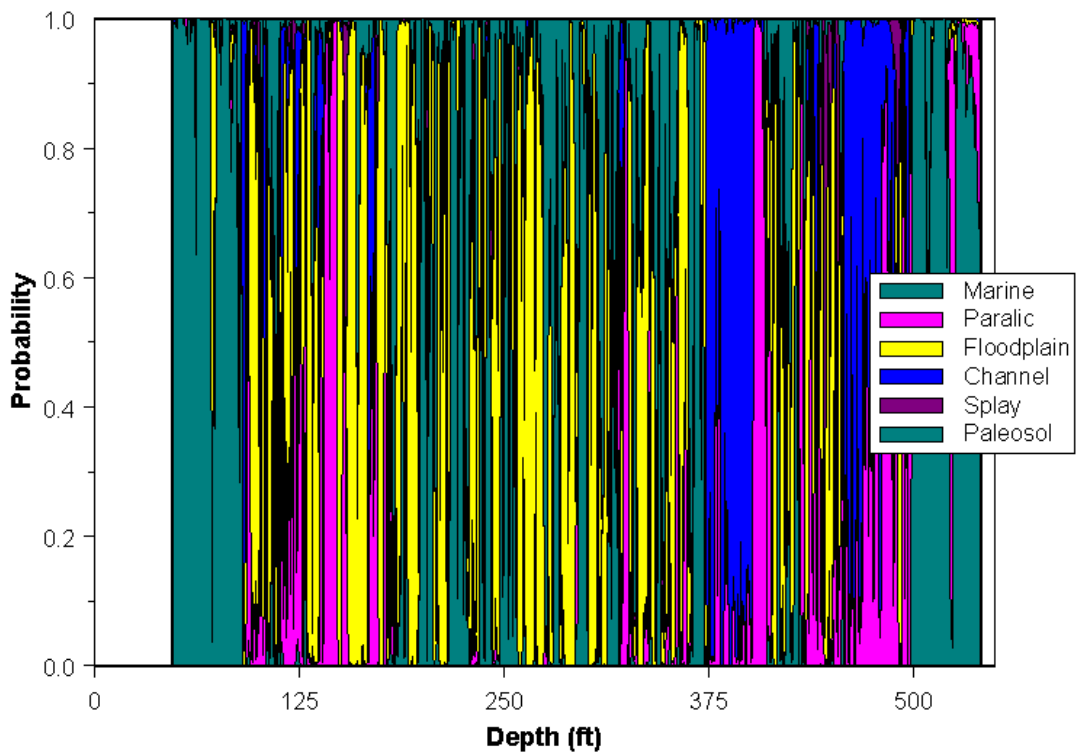
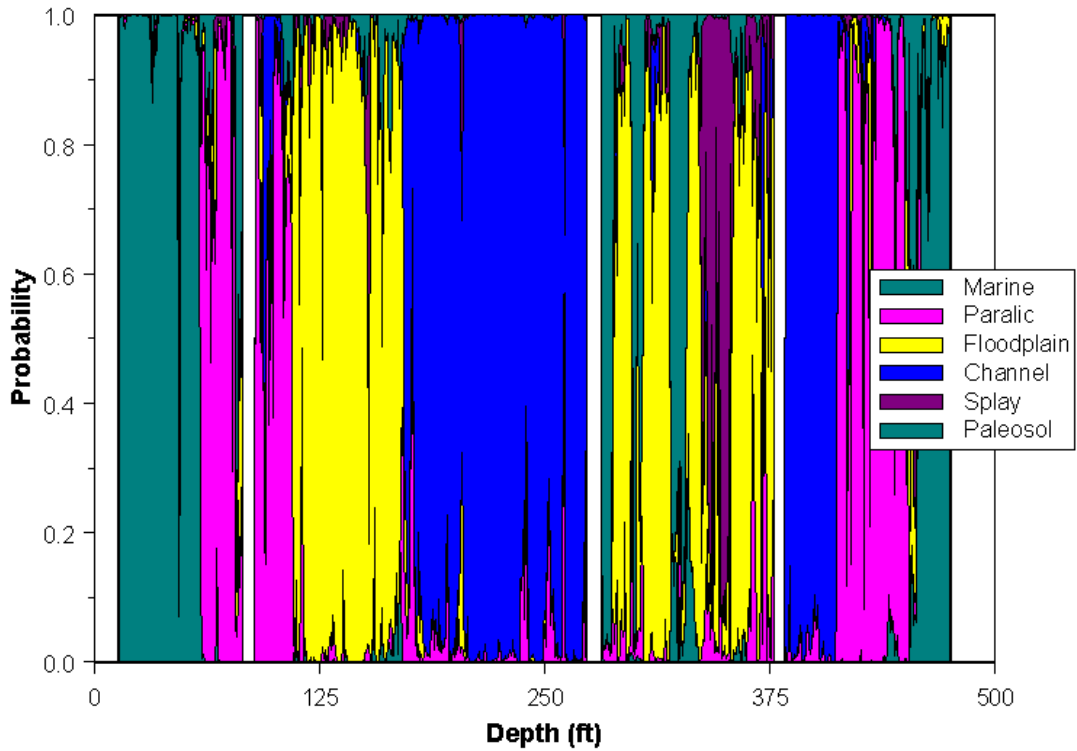
And the predictions versus depth look like:



The predictions in the Kenyon are still quite messy:



We can also look at the facies probabilities versus depth in each well, first the Jones, then the Kenyon:



For the facies classification problem we have certain expectations regarding typical facies thicknesses and regarding which facies occur more frequently above other facies. We can encode these expectations into a transition probability matrix that describes the probability of observing each facies at certain depth given that a particular facies occurs at the next depth below that.

The transition probability matrix developed from the facies observed at half-foot intervals (with a little modification) is:

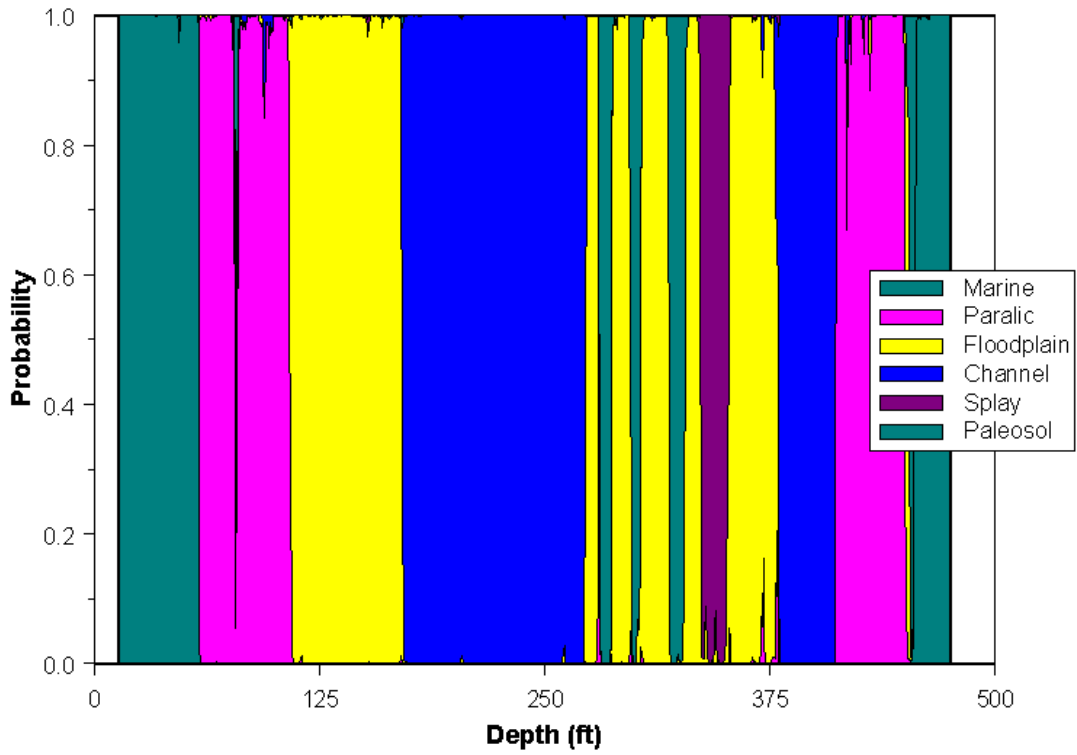
Facies Below	Facies Above					
	Marine	Paralic	Floodplain	Channel	Splay	Paleosol
Marine	0.99	0.00	0.00	0.00	0.00	0.01
Paralic	0.01	0.98	0.01	0.01	0.00	0.00
Floodplain	0.00	0.01	0.96	0.01	0.00	0.01
Channel	0.00	0.00	0.01	0.97	0.01	0.01
Splay	0.00	0.00	0.03	0.00	0.97	0.00
Paleosol	0.00	0.03	0.03	0.00	0.00	0.94

We can use Bayes' theorem (again) to combine the probabilities predicted from the logs at depth m , p_k^m , with facies probabilities derived from the probabilities at the underlying interval, $m-1$, weighted by the upward transition probabilities, $t_{j,k}$, to derive modified probabilities at each depth, combining information from the logs with our expectations regarding spatial adjacency:

$$w_k^m = \frac{p_k^m \sum_j t_{j,k} w_j^{m-1}}{\sum_j p_j^m \sum_r t_{r,j} w_r^{m-1}}$$

Applying this procedure in both wells cleans up the predicted probabilities considerably:

Jones



Kenyon

