EECS833 Facies classification problem, designed by Dubois, Bohling and Chakrabarti
Spring 2006

Objective:
Develop two feed forward back propagating artificial neural networks capable of satisfactorily estimating rock facies (classes) for the Council Grove gas reservoir in Southwest Kansas.

Provided:
Council Grove training data

- nine wells (4149 examples)
- seven predictor variables
- rock facies (class) for each example vector

Validation (test) data (830 examples from two wells)

available online at
www.people.ku.edu/~gbohling/EECS833/)

Seven predictor variables:

- Five wire line log curves include gamma ray (GR), resistivity (ILD_log10), photoelectric effect (PE), neutron-density porosity difference (DeltaPHI), and average neutron-density porosity (PHIND). Some wells do not have PE.
- Two geologic constraining variables: nonmarine-marine indicator (NM_M) and relative position (RELPOS)

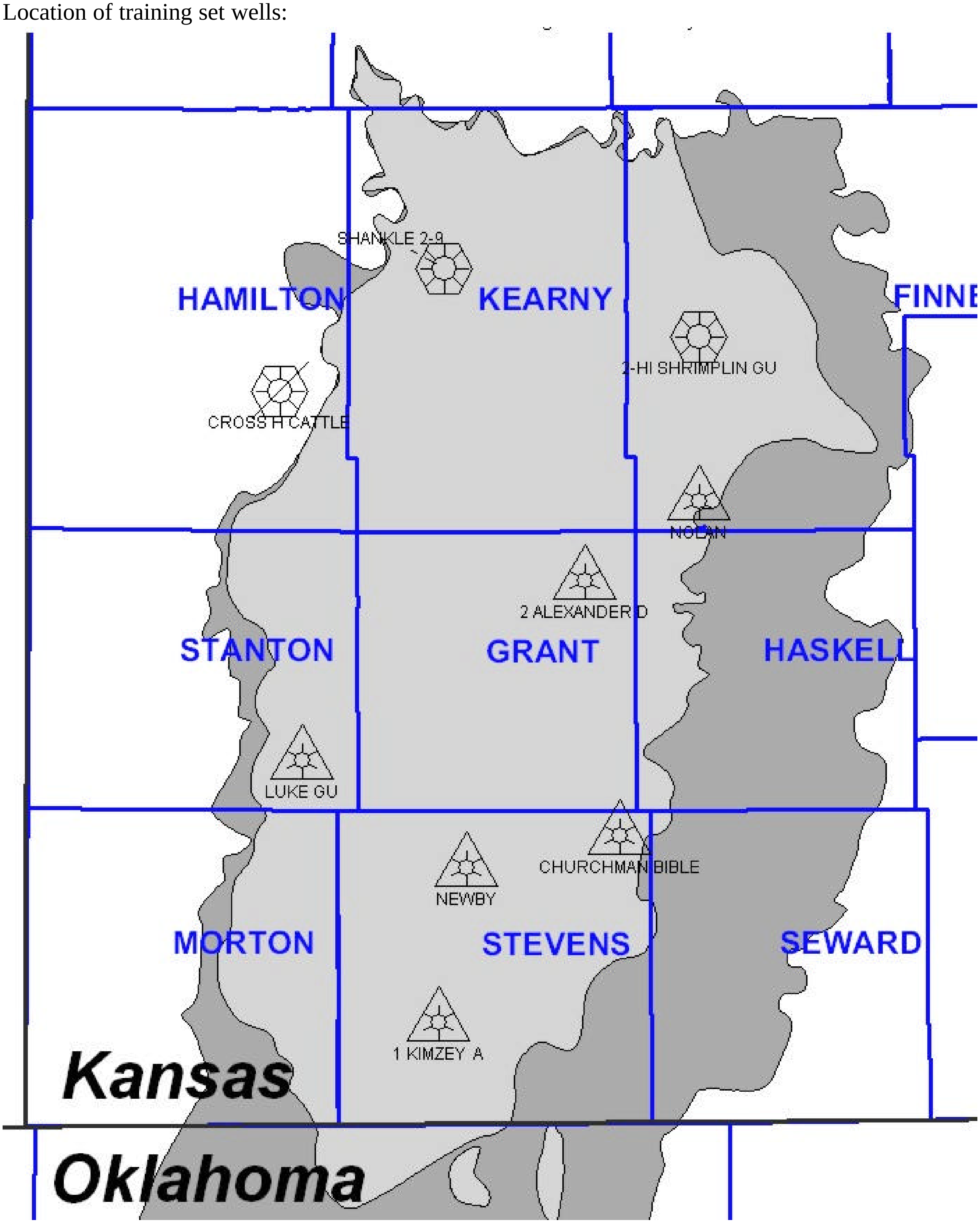
Nine discrete facies (classes of rocks) numbered 1-9

1. Nonmarine sandstone
2. Nonmarine coarse siltstone
3. Nonmarine fine siltstone
4. Marine siltstone and shale
5. Mudstone (limestone)
6. Wackestone (limestone)
7. Dolomite
8. Packstone-grainstone (limestone)
9. Phylloid-algal bafflestone (limestone)

Example of data:

Facies	Formation	Well Name	Depth	GR	ILD_log10	DeltaPHI	PHIND	PE	NM_M	RELPOS
3	A1 SH	NOLAN	2854.5	94.375	0.553	7.097	14.583	3.195	1	0.955
3	A1 SH	NOLAN	2855	89.813	0.554	7.081	14.11	2.963	1	0.932
3	A1 SH	NOLAN	2855.5	91.563	0.56	6.733	13.189	2.979	1	0.909
3	A1 SH	NOLAN	2856	87.188	0.567	6.578	12.806	3.16	1	0.886
SKIP										
5	A1 SH	NOLAN	2874	76.25	0.295	7.123	19.977	2.981	1	0.068
2	A1 SH	NOLAN	2874.5	73.938	0.301	10.693	21.143	3.065	1	0.045
3	A1 SH	NOLAN	2875	75.5	0.323	8.346	20.17	3.133	1	0.023
5	A1 LM	NOLAN	2875.5	77.938	0.35	5.703	16.27	3.32	2	1
5	A1 LM	NOLAN	2876	85.75	0.384	1.07	13.579	3.662	2	0.983
5	A1 LM	NOLAN	2876.5	81.875	0.438	0.969	9.822	4.394	2	0.966
8	A1 LM	NOLAN	2877	75.125	0.518	1.062	7.629	4.562	2	0.948
8	A1 LM	NOLAN	2877.5	73.375	0.614	0.67	6.898	4.894	2	0.931
SKIP										
6	A1 LM	NOLAN	2881	55.063	0.685	2.932	10.984	3.461	2	0.81
4	A1 LM	NOLAN	2881.5	58.844	0.658	4.265	11.75	3.434	2	0.793
4	A1 LM	NOLAN	2882	55.906	0.622	4.477	11.734	3.473	2	0.776
4	A1 LM	NOLAN	2882.5	54.781	0.576	3.558	11.383	3.412	2	0.759
4	A1 LM	NOLAN	2883	56.375	0.53	2.147	10.89	3.527	2	0.741
6	A1 LM	NOLAN	2883.5	65.125	0.498	2.587	11.329	3.424	2	0.724
6	A1 LM	NOLAN	2884	67.813	0.487	5.048	12.872	3.441	2	0.707
4	A1 LM	NOLAN	2884.5	64.938	0.495	7.595	14.28	3.201	2	0.69
7	A1 LM	NOLAN	2885	58.5	0.511	9.249	16.044	3.19	2	0.672
7	A1 LM	NOLAN	2885.5	61.313	0.523	8.082	15.111	3.244	2	0.655
7	A1 LM	NOLAN	2886	80.563	0.529	6.129	13.842	3.67	2	0.638

Location of training set wells:



Procedure:

Train two neural networks

- seven predictor variables (with PE)
- six (without PE, call it NoPE).

The object is to train generalized neural networks that are able to predict the facies for wells not in the training set provided but having the same set of predictor variables and known facies. We will test the neural networks that you build on data from two wells not in the training set provided.

Success Metrics:

- Absolute accuracy: correct / all
- Accuracy within 1 facies:

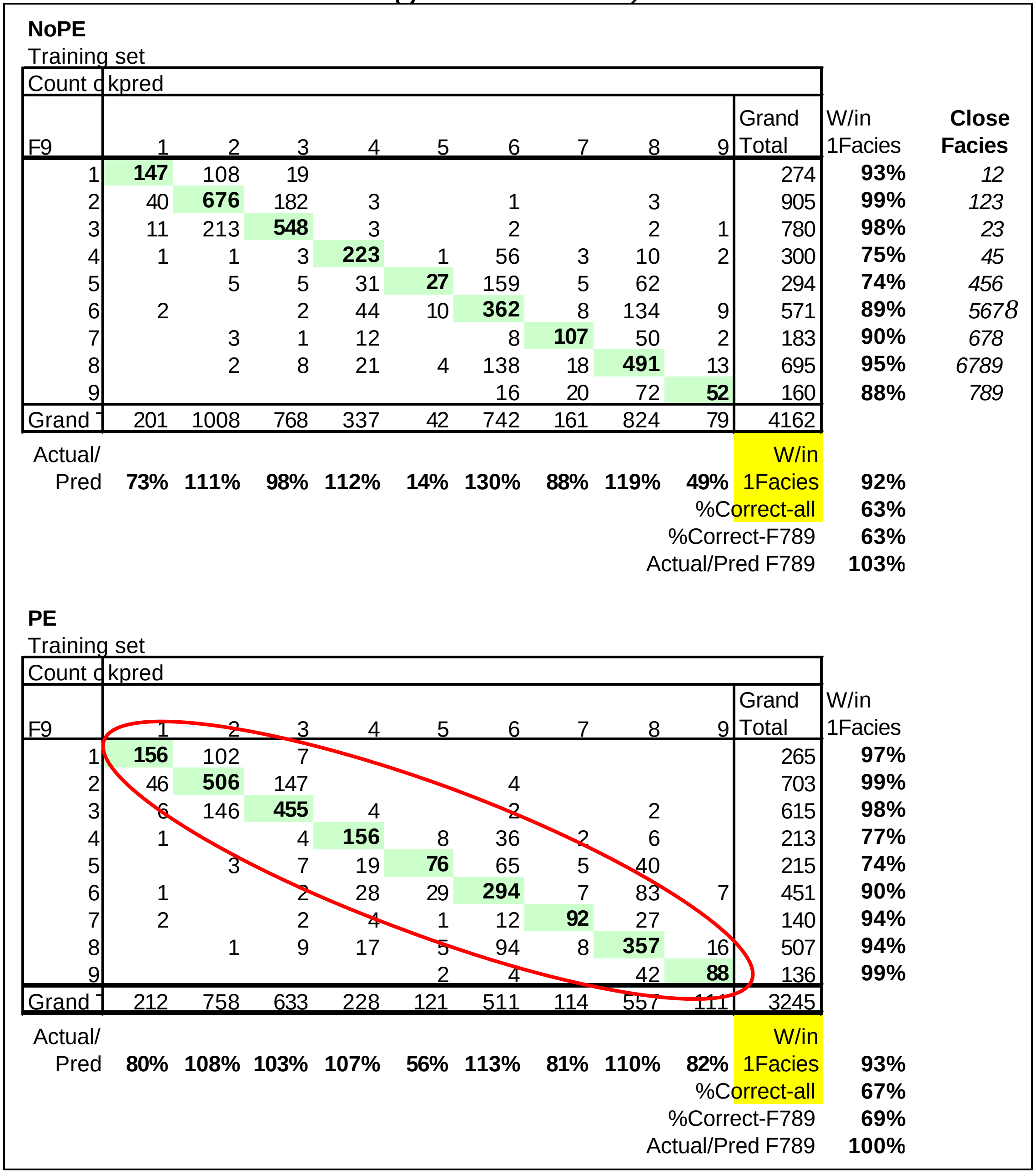
  (correct + w/in 1 facies) / all

Adjacent facies:

Facies	Adjacent Facies
1	2
2	1,3
3	2
4	5
5	4,6
6	5,7,**8**
7	6,8
8	6,7,9
9	7,8

Note change from that in handout

Training results for rock classification problem (NoPE and with PE): Train on entire train set and predict facies for same (not the way to train a neural network, but serves to illustrate something to shoot for).



NoPE

Training set

Count o kpred

F9	1	2	3	4	5	6	7	8	9	Grand Total	W/in 1Facies	**Close Facies**
1	**147**	108	19							274	**93%**	*12*
2	40	**676**	182	3		1		3		905	**99%**	*123*
3	11	213	**548**	3		2		2	1	780	**98%**	*23*
4	1	1	3	**223**	1	56	3	10	2	300	**75%**	*45*
5		5	5	31	**27**	159	5	62		294	**74%**	*456*
6	2		2	44	10	**362**	8	134	9	571	**89%**	*5678*
7		3	1	12		8	**107**	50	2	183	**90%**	*678*
8		2	8	21	4	138	18	**491**	13	695	**95%**	*6789*
9						16	20	72	**52**	160	**88%**	*789*
Grand T	201	1008	768	337	42	742	161	824	79	4162		
Actual/ Pred	**73%**	**111%**	**98%**	**112%**	**14%**	**130%**	**88%**	**119%**	**49%**	W/in 1Facies	**92%**	
										%Correct-all	**63%**	
										%Correct-F789	**63%**	
										Actual/Pred F789	**103%**	

PE

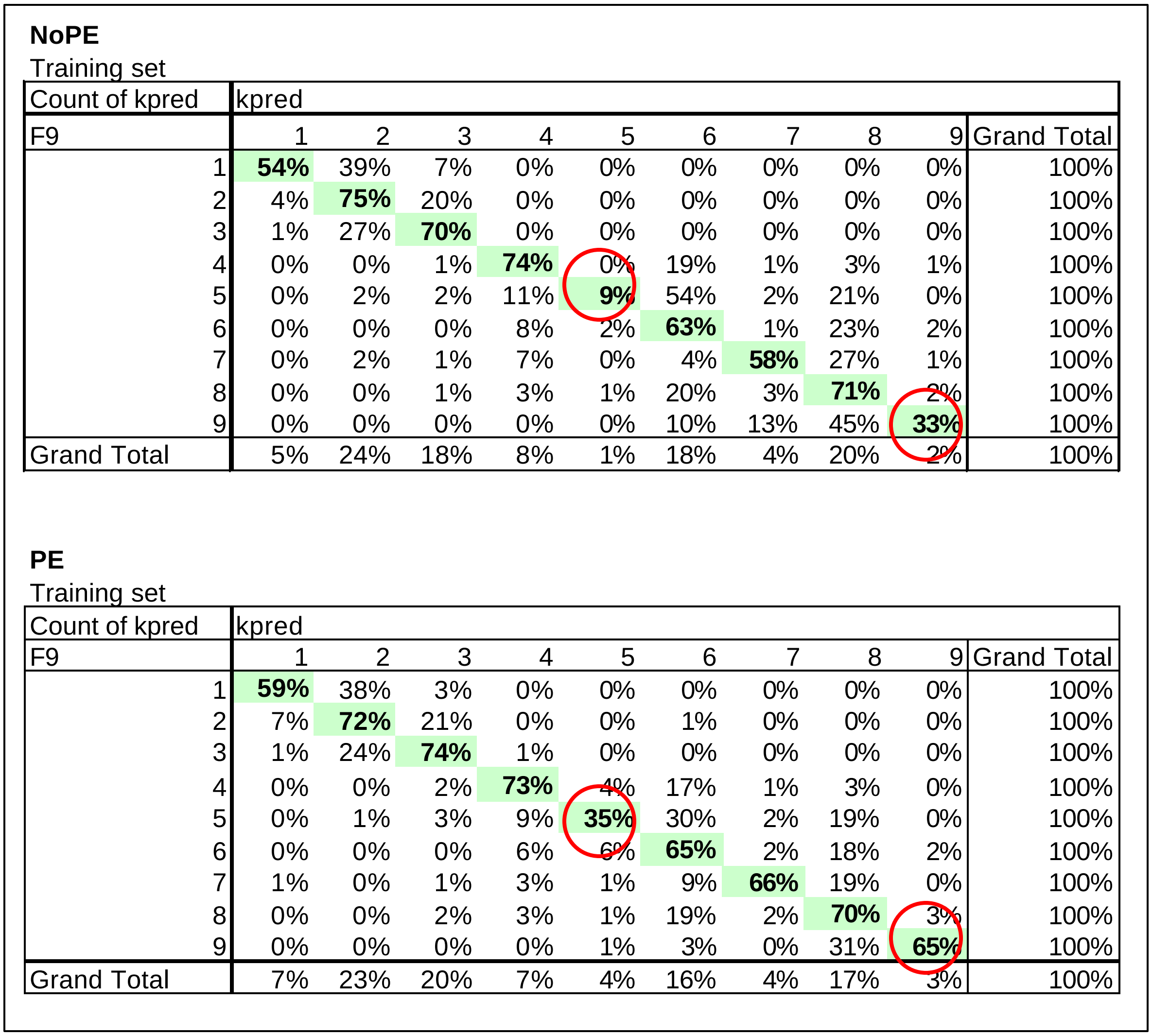
Training set

Count o kpred

F9	1	2	3	4	5	6	7	8	9	Grand Total	W/in 1Facies
1	**156**	102	7							265	**97%**
2	46	**506**	147			4				703	**99%**
3	6	146	**455**	4		2		2		615	**98%**
4	1		4	**156**	8	36	2	6		213	**77%**
5		3	7	19	**76**	65	5	40		215	**74%**
6	1		2	28	29	**294**	7	83	7	451	**90%**
7	2		2	4	1	12	**92**	27		140	**94%**
8		1	9	17	5	94	8	**357**	16	507	**94%**
9					2	4		42	**88**	136	**99%**
Grand T	212	758	633	228	121	511	114	557	111	3245	
Actual/ Pred	**80%**	**108%**	**103%**	**107%**	**56%**	**113%**	**81%**	**110%**	**82%**	W/in 1Facies	**93%**
										%Correct-all	**67%**
										%Correct-F789	**69%**
										Actual/Pred F789	**100%**

Facies 7, 8, and 9 are the most critical facies to identify correctly, but being close is almost as good as being correct.

% facies classified correctly for same training/testing sessions:



NoPE

Training set

Count of kpred	kpred									
F9	1	2	3	4	5	6	7	8	9	Grand Total
1	**54%**	39%	7%	0%	0%	0%	0%	0%	0%	100%
2	4%	**75%**	20%	0%	0%	0%	0%	0%	0%	100%
3	1%	27%	**70%**	0%	0%	0%	0%	0%	0%	100%
4	0%	0%	1%	**74%**	0%	19%	1%	3%	1%	100%
5	0%	2%	2%	11%	**9%**	54%	2%	21%	0%	100%
6	0%	0%	0%	8%	2%	**63%**	1%	23%	2%	100%
7	0%	2%	1%	7%	0%	4%	**58%**	27%	1%	100%
8	0%	0%	1%	3%	1%	20%	3%	**71%**	2%	100%
9	0%	0%	0%	0%	0%	10%	13%	45%	**33%**	100%
Grand Total	5%	24%	18%	8%	1%	18%	4%	20%	2%	100%

PE

Training set

Count of kpred	kpred									
F9	1	2	3	4	5	6	7	8	9	Grand Total
1	**59%**	38%	3%	0%	0%	0%	0%	0%	0%	100%
2	7%	**72%**	21%	0%	0%	1%	0%	0%	0%	100%
3	1%	24%	**74%**	1%	0%	0%	0%	0%	0%	100%
4	0%	0%	2%	**73%**	4%	17%	1%	3%	0%	100%
5	0%	1%	3%	9%	**35%**	30%	2%	19%	0%	100%
6	0%	0%	0%	6%	6%	**65%**	2%	18%	2%	100%
7	1%	0%	1%	3%	1%	9%	**66%**	19%	0%	100%
8	0%	0%	2%	3%	1%	19%	2%	**70%**	3%	100%
9	0%	0%	0%	0%	1%	3%	0%	31%	**65%**	100%
Grand Total	7%	23%	20%	7%	4%	16%	4%	17%	3%	100%

Some facies are more difficult to predict than others. Having the PE curve is helpful, but not all of the wells have this curve, so we must rely on a neural network trained without the PE where the PE is not available.

Training suggestions:

Train neural networks that are capable of predicting facies for wells that are not part of the training set.

Your goal is to determine the <u>optimal training parameters</u> for neural networks that are sufficiently general to estimate facies on the withheld data.

Devise and execute a plan that uses the given data for training and testing in a manner that most closely mimics the real test (on withheld data).

Parameters to optimize:

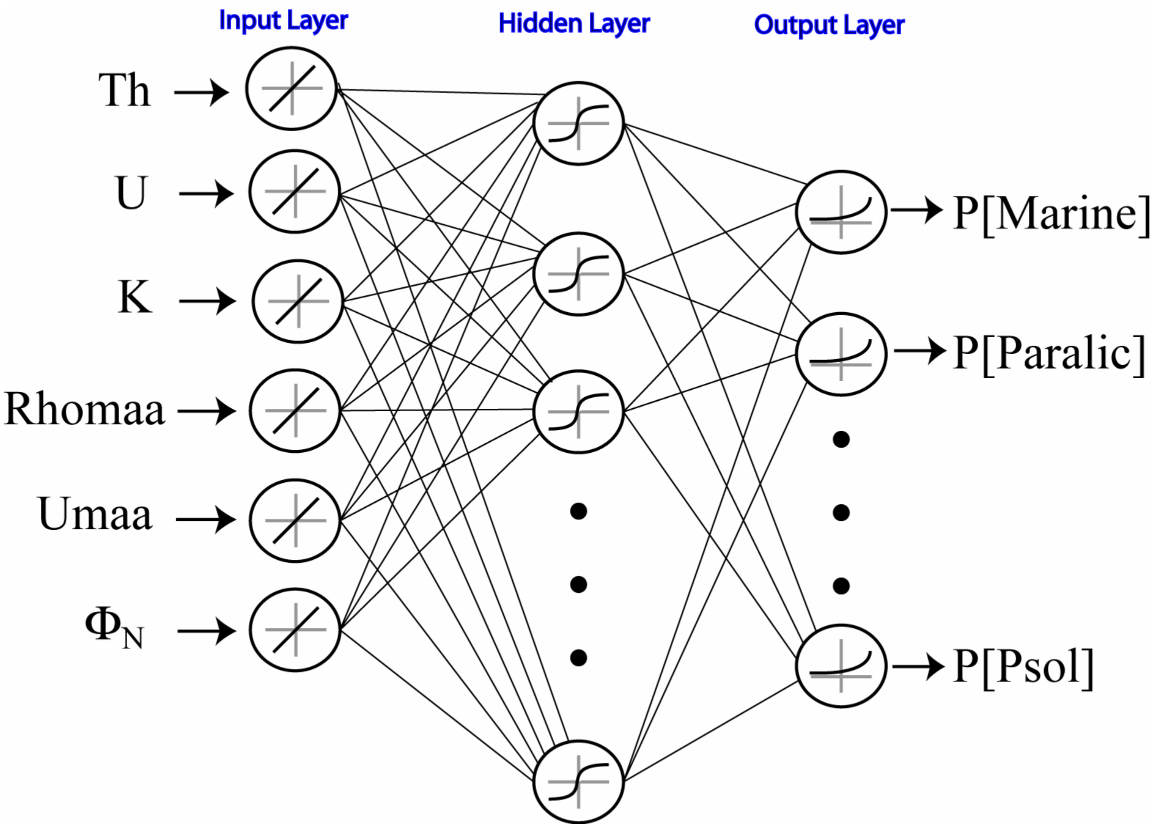
- Number of hidden layers
- Hidden layer nodes
- Training function
- Learning function
- Damping or momentum constants
- Iterations (epochs)
- Transfer functions (Hard limit, log sigmoid, tan sigmoid)

Suggestion: In our work, we have been using a neural network (not in MatLab) and have had success with a neural network with a single hidden layer with 20-50 nodes, damping parameter of 0.1-1, and 100 iterations.

The error will be higher than you may be used to so you will not be training towards a specified target error rate and will want to limit the iterations (epochs).

As far as I (Geoff) can tell, it would be difficult to use Matlab's neural net toolbox to set up exactly the same kind of network we have used and we do not expect you to do so. However, just to give you a better idea of what we have done, here is a little more detail/review:

Here is the network for the Jones data example:



The Hugoton/Panoma problem has different logs & facies, but the structure is the same. Bias nodes are not shown, but they are there. The input logs are scaled to zero mean and unit standard deviation (*zscore*) prior to analysis. This means the scaled inputs fall mainly between -3 and 3 in general, but they are not constrained to hard limits.

The input-to-hidden layer weights are represented by α , the hidden-to-output layer weights by β . These are all initialized to random values distributed around 0 to start with.

Each hidden-layer node, *i*, computes a linear combination of the scaled log values X_j, and feeds it through a logsig transfer function to produce its output, Z_i:

$$Z_i = \mathrm{logsig}\left(\alpha_{0,i} + \sum_{j=1}^{p} \alpha_{j,i} X_j \right)$$

where $\mathrm{logsig}(v) = 1/(1 + \exp(-v))$, just as in Matlab, $\alpha_{0,i}$ is the bias weight for node *i*, and *p* is the number of input variables (logs).

Each output layer node, *k*, first forms a general linear output, T_k, from a linear combination of hidden-layer outputs:

$$T_k = \beta_{0,k} + \sum_{i=1}^{M} \beta_{i,k} Z_i$$

where *M* is the number of hidden-layer nodes. There are *K* output nodes, one for each class (facies).

The set of (unconstrained) linear outputs, T_k, are fed through the softmax transfer function to create outputs that act like probabilities (each between 0 and 1 and all summing to 1):

$$P_k = \frac{\exp(T_k)}{\sum_{\ell=1}^{K}\exp(T_\ell)}$$

So, for a certain input vector of scaled logs, *X*, we treat the resulting output vector, *P*, as representing the mutually exclusive set of probabilities of membership in each of the *K* different facies.

For training we use a cross-entropy objective (performance) function with a *regularization* term which penalizes large weights:

$$R(\boldsymbol{a}, \boldsymbol{b}) = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{i,k}\log\left(P_k\left(x_i; \boldsymbol{a}, \boldsymbol{b}\right)\right) + \boldsymbol{l}\left(\sum \boldsymbol{a}^2 + \sum \boldsymbol{b}^2\right)$$

N is the number of training data points and $y_{i,k}$ represents the set of target indicator values for data point *i*: $y_{i,k} = 1$ for the class to which data point *i* belongs and 0 for all other classes (a set of probabilities representing certain knowledge of class membership). The first term above (the cross-entropy part) picks out the predicted probability associated with the actual class for each data point, takes the negative log of that, and sums up the results over all training data points. Minimizing that sum drives the predicted probabilities for the target classes towards 1 and the others towards 0. *I can't find a cross-entropy performance function in Matlab*.

The second term in the objective/performance function is the *regularization* term – I have represented the sums over the weights conceptually, without indices. The parameter l is the *decay*, *damping*, or *regularization* parameter. Note that it is introduced explicitly as a piece of the objective function, not as a parameter of the training algorithm (like momentum). We are trying to minimize an objective function that includes l times the sum of the squared weights, so increasing l forces the weights to be smaller in magnitude than they would be otherwise. This drives the network to produce smoother boundaries between classes. Without damping, the network can develop fairly large weights resulting in very sharp sigmoid basis functions. With smaller weights the basis functions are more spread out.

We minimize the objective function using BFGS optimization (*trainbfg*-ish in Matlab).

Matlab includes a mean-squared error performance function including a regularization term, *msereg*. The regularization parameter there describes the proportional weighting of the mean-squared-error and mean-squared-weight (regularization) terms in the performance function.

So, the primary piece that is lacking in terms of reproducing our neural net in Matlab is a cross-entropy performance function. However, this may not be very crucial – minimizing a mean-squared error directly on the target outputs may do as well or better.

Required:

1. Two trained neural networks, one using seven predictor variables (w/ PE) and one using six (NoPE) including program for simulating the neural networks on a validation (test) set of data and calculating the measures of success.

2. Predicted discrete facies for each of the 830 examples in the validation set, identified by well name and depth.

3. Written report that should include:
   1) Objective
   2) Data analysis and preparation. Please provide some analysis of the data that demonstrates that you have compared the variable space for the different classes of rocks (e.g.: simple statistics, 3D cross plots, density functions, etc.). If you desire, consider evaluating dimension reduction and/or cluster analysis techniques.
   3) Training and testing procedures for developing optimal neural networks (two- PE and NoPE), along with test results.
   4) Testing performance of the trained neural networks on the entire training set
   5) Documented program listing
   6) Neural network and your program on diskette or CD