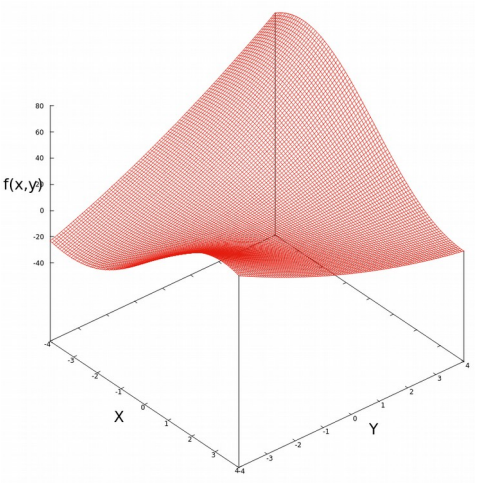
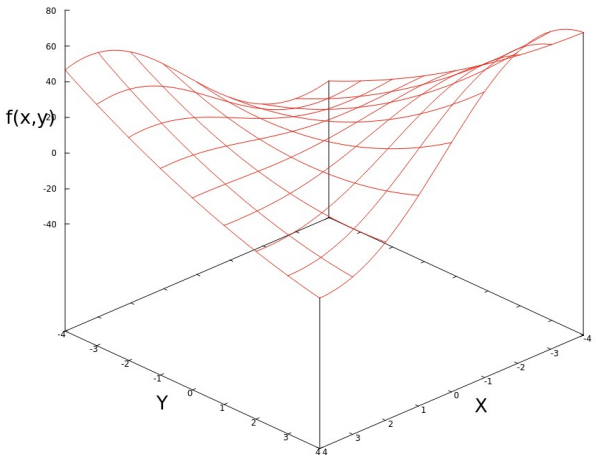
Homework assignment 6

Task:
Build a neuro-fuzzy system which uses Takagi-Suengo-Kang (TSK) inference method.
In literature, models which uses combination of fuzzy logic and neural networks are called ANFIS models (*Adaptive neuro fuzzy inference system*).

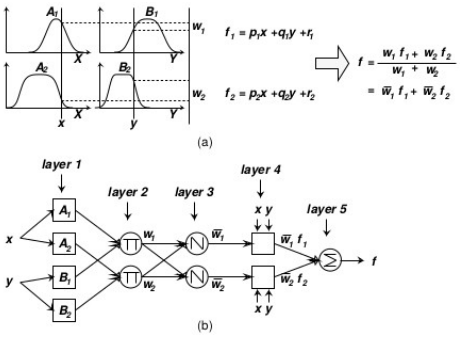
In our task we used ANFIS model to learn function:
$f(x,y) = ((x-1)^2 + (y+2)^2 - 5xy + 3)cos^2(\frac{x}{5})$



3d plot of this function can be seen on right hand side. Both coordinate domains are D=[-4,4] and for both coordinates we sample all whole numbers from that domain. This sampling method gives as 9x9 grid on which we sample 81 samples. Goal function sampled on 9x9 grid looks like this:



ANFIS

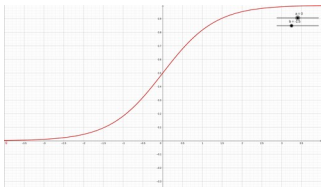
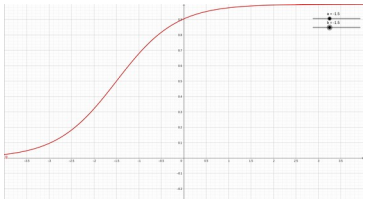
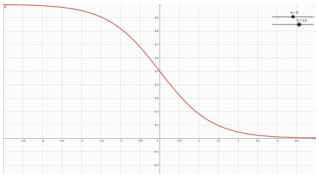
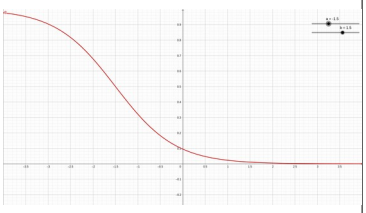
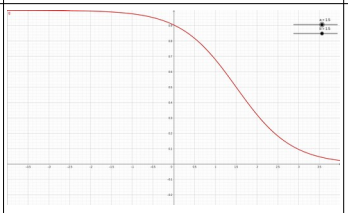
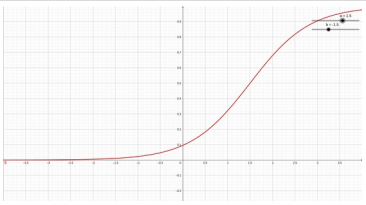


In upper picture we can see ANFIS model which has 2 rules:
1) IF x is A1 ∩ y is B1 THEN z1 = p1x + q1y + r1
2) IF x is A2 ∩ y is B2 THEN z2 = p2x + q2y + r2

Fuzzy sets denoted by A_i are sets which apply to x coordinate and fuzzy sets B_i apply to y coordinate.
Each fuzzy set, has it's own membership function which is modeled as

$$\mu_{A_i}(x) = \frac{1}{1 + e^{b_i(x - a_i)}}$$

So, in our task we modeled each neuron as one fuzzy set with mentioned membership function and with learnable parameters a_i, b_i. Below are given an examples of types of functions which each neuron can learn:

B<0 **A=0**		**B<0** **A<0**	
B>0 **A=0**		**B>0** **A<0**	
B>0 **A>0**		**B<0** **A>0**	

Each type of sigmoid has it's own semantics and we can deduce a meaning from analysing neuron parameters. First of all, we must understand what does it mean that neuron is active. Neuron is active for certain input x_i if membership function of that input x_i to set A is high enough so that we can say that x_i belongs to A. From upper plots we can see that parameter b controls sigmoid steepness and parameter a translates sigmoid in horizontal dericitions.
If parameter b is small than sigmoid will be smoother and this will make set A fuzzier. In contrast with that, if b is high sigmoid is reduced to step function and that reduces A to classical set, where resoning is quite simple.
As already said, parameter a controls translation of sigmoid. If we make b $\rightarrow$ -inf, and make a = y, than we can say that elements bigger than y are in set A, and those samller aren't in set A. So with a we control the thrashold which separates elements in A and those whcih aren't in A.
Summary of this reasoning is given below:

Parameter	Description
a	Threshold which separates elements which are in set A and those that aren't in it
b	Fuzzines of set A. b -> ∞ or -∞ ... A is reduced to classical set b -> 0 ... A is very fuzzy

Membership functions of A_i and B_i are combined with product T-norm:
$\mu_{A_i \cap B_i}(x) = \mu_{A_i}(x)\mu_{B_i}(x) = w_i$

With this line of reasoning we can see that each antecedents of some rule R_i learns boundories of some space Ψ of our domain D = [-4,4] x [-4,4] and if input $\vec{x}$ is very close to that space, antecedents of that rule R_i is activated or $w_i \to 1$.

Consequence of some rule R_i is a linear function with 3 parameters p_i, q_i, r_i and it is given by:
$f_i = p_i x + q_i y + r_i$

Each consequnece activation f_i is weightened by normalized antecedent activation $\overline{w_i}$ and in the end consequences are summed together to give final output f.
We can see that out model is trying to approximate goal function with linear functions.

Learning – Backpropagation

Loss function with whome we are learning our model is a mean squared error:
$Ł = \frac{1}{2}(y - o)^2$,
where y is target value and o our prediction. For each parameter we must calculate gradients.

Parameters p, q, r:

$$\frac{\partial L}{\partial p_i} = \frac{\partial L}{\partial o}\frac{\partial o}{\partial h_i}\frac{\partial h_i}{\partial f_i}\frac{\partial f_i}{\partial p_i} = -(y - o)\frac{w_i}{\Sigma w_i}x$$

$$\frac{\partial L}{\partial q_i} = \frac{\partial L}{\partial o}\frac{\partial o}{\partial h_i}\frac{\partial h_i}{\partial f_i}\frac{\partial f_i}{\partial q_i} = -(y - o)\frac{w_i}{\Sigma w_i}y$$

$$\frac{\partial L}{\partial r_i} = \frac{\partial L}{\partial o}\frac{\partial o}{\partial h_i}\frac{\partial h_i}{\partial f_i}\frac{\partial f_i}{\partial r_i} = -(y - o)\frac{w_i}{\Sigma w_i}$$

Parameters a,b:

$$\frac{\partial L}{\partial a_i} = \frac{\partial L}{\partial o} \sum_{j=1}^{rules} \frac{\partial o}{\partial \overline{w_j}} \frac{\partial \overline{w_j}}{\partial w_j} \frac{\partial w_j}{\partial \alpha_j} \frac{\partial \alpha_j}{\partial a_i}$$

$$\frac{\partial L}{\partial b_i} = \frac{\partial L}{\partial o} \sum_{j=1}^{rules} \frac{\partial o}{\partial \overline{w_j}} \frac{\partial \overline{w_j}}{\partial w_j} \frac{\partial w_j}{\partial \alpha_j} \frac{\partial \alpha_j}{\partial b_i}$$

$$\frac{\partial o}{\partial \overline{w_i}} = f_i \qquad \frac{\partial \overline{w_i}}{\partial w_i} = \frac{1}{\Sigma w_i} - \frac{w_i}{(\Sigma w_i)^2}$$

$$\frac{\partial \overline{w_j}}{\partial w_i} = -\frac{w_i}{(\Sigma w_i)^2} \qquad \frac{\partial w_i}{\partial \alpha_i} = \beta_i$$

$$\frac{\partial \alpha_i}{\partial a_i} = b_i \alpha_i (1 - \alpha_i) \qquad \frac{\partial \alpha_i}{\partial b_i} = -(x - a_i)\alpha_i(1 - \alpha_i)$$

$$\frac{\partial L}{\partial \theta_i} = -(y - o)(f_i \frac{1}{\Sigma w_.} \beta_i \frac{\partial \alpha_i}{\partial \theta_i} - \sum_{j=1, j \neq i}^{rules} f_j \frac{w_j}{(\Sigma w_.)^2} \beta_j \frac{\partial \alpha_j}{\partial \theta_j})$$

Batch Gradient Descent

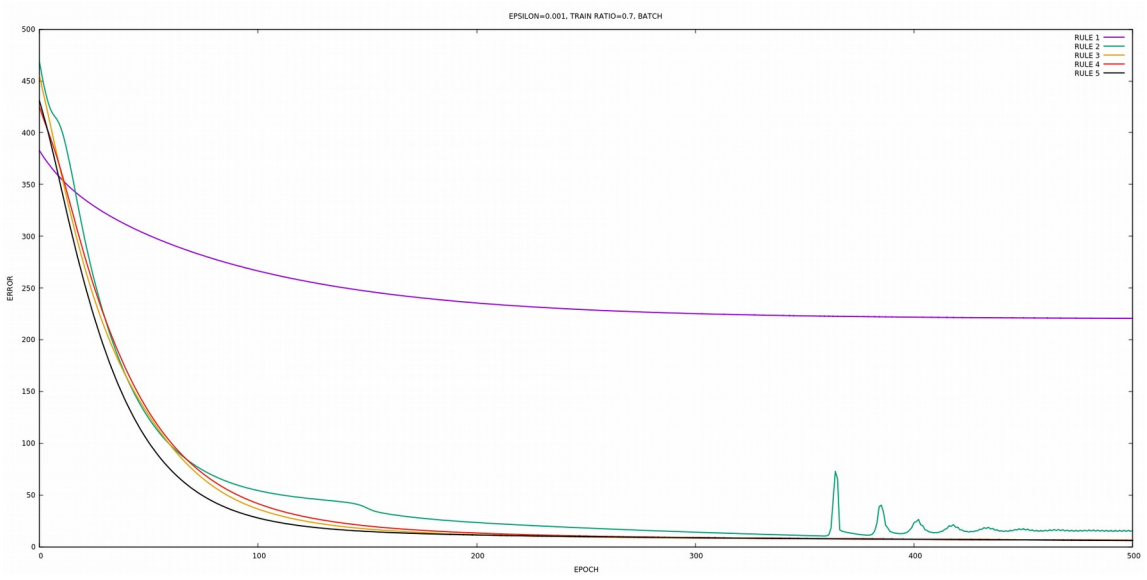
$$\theta_{t+1} = \theta_t - \eta \frac{1}{N} \sum_{\vec{x} \epsilon dataset} \frac{\partial L(\vec{x}; \theta_t)}{\partial \theta}$$

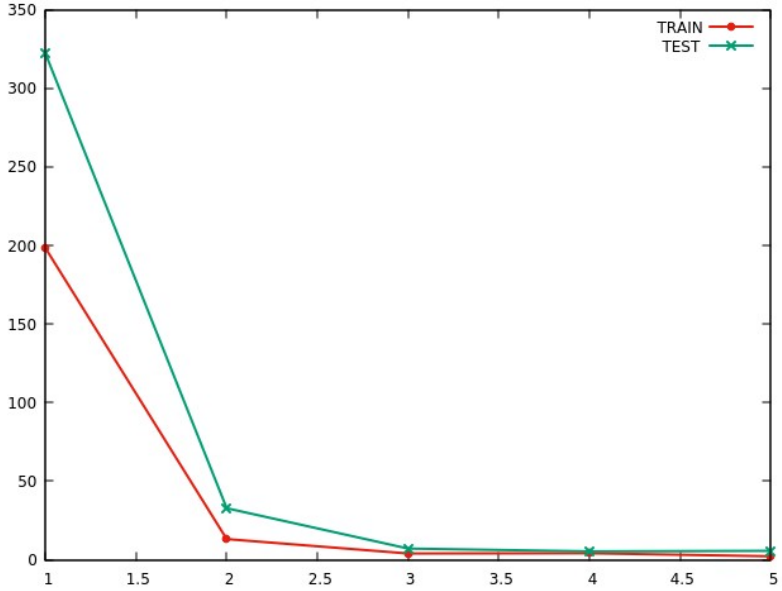
Stochastic Gradient Descent

$$\theta_{t+1} = \theta_t - \eta \frac{\partial L(\vec{x}; \theta_t)}{\partial \theta}$$

Number of rules - BATCH

In this experiment we are learning ANFIS with variable number of rules and we examine test and train error. For training we took 70% of overall dataset for training and remaining part for evaluation, learning rate was 0.0001 and we learned for 1000 epochs

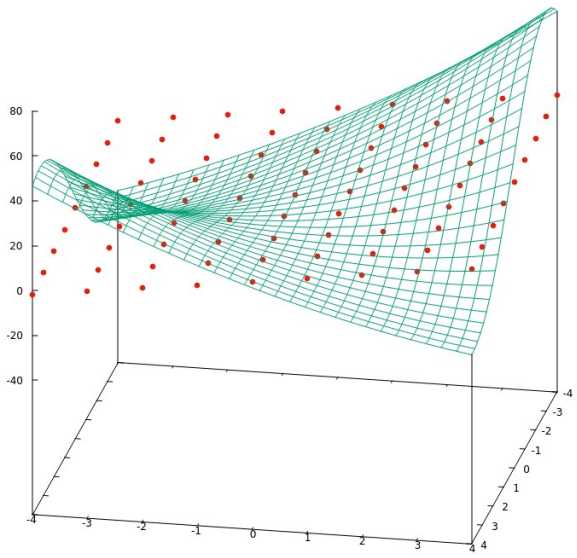


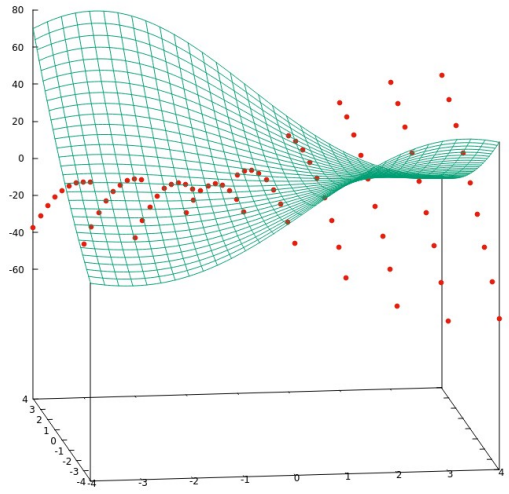


From the following plot we can see how dramatically performance changes when we have more than 1 rule in our ANFIS. Best performance is accomplished for 3 rules because following rules don't contibute much to the performance

On the following plots we can see what function model heas learned for 1, 2 and 3 rules.
On the left picures we can see what function has model learned and on the right we ploted differences between target and predicted values

1 RULE

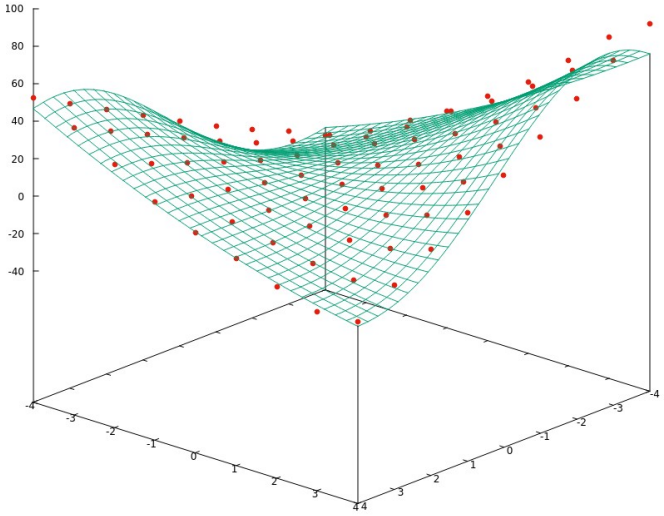


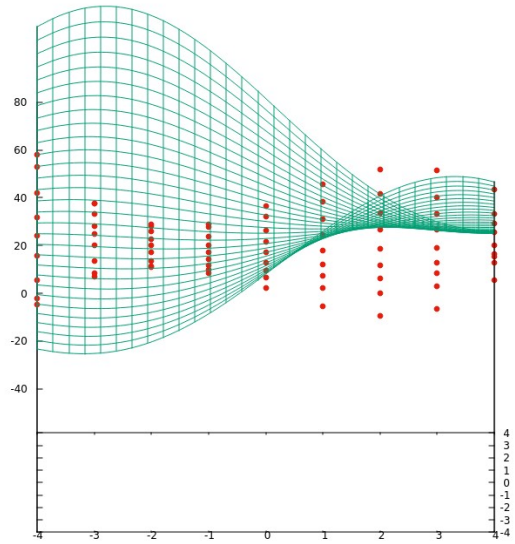


For 1 rule we can see that model don't have capacity for modeling nonlinearities. Hypothesis is a linear function and residuals behave nonlineary which shows us that model has learned all linerity in dataset but was incapable for learning nonlinearity.

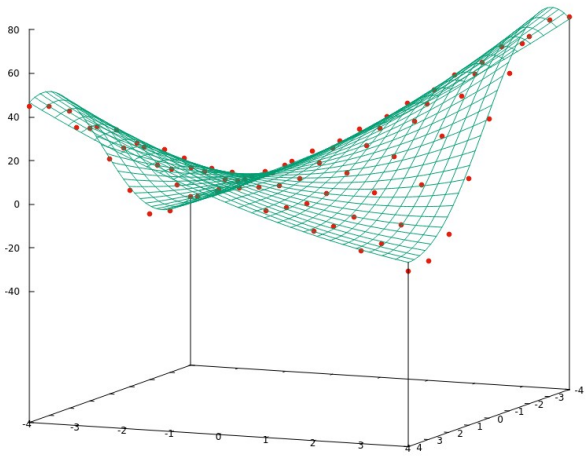
---

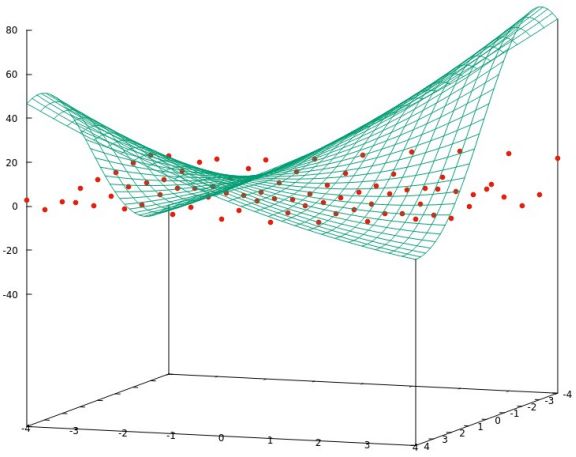
2 RULES





3 RULES

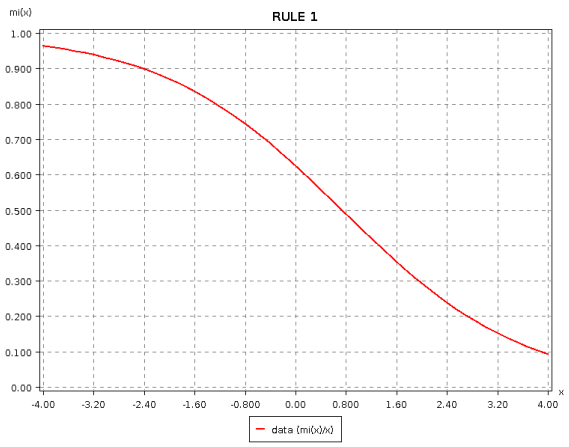


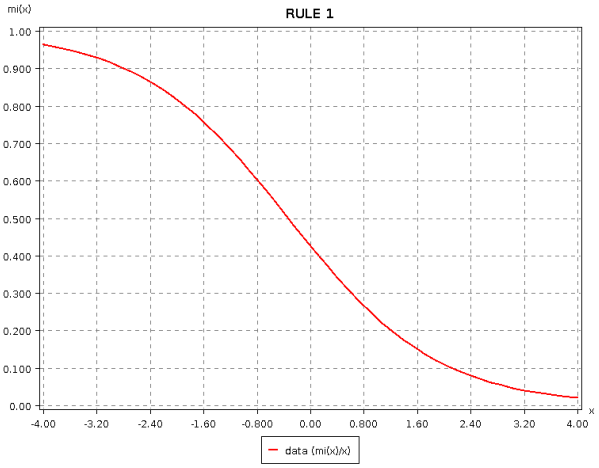


For 2 and 3 rules we can see that hypothesis is greatly improved and residuals are almost 0.

Unlike normal neural networks here we have interpretability feature which can tell us why and what neurons have learned. In this setting we have learned ANFIS model with 1,2 and 3 rules. Every model has been learned for 1E5 epochs and with learning rate of 1E-5. Plots of membership functions for each rule are given below.
First column of this plots represents membership function of neurons which learn of x component and second column neurons which learn on y component of goal function.

1 RULE

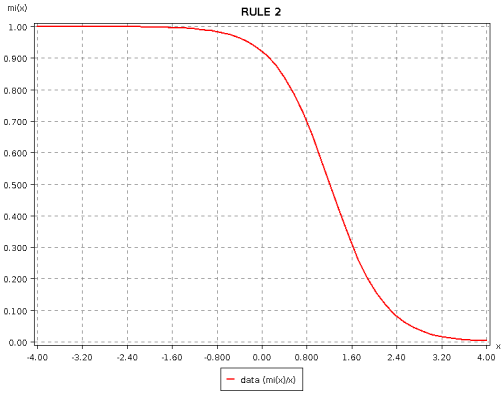


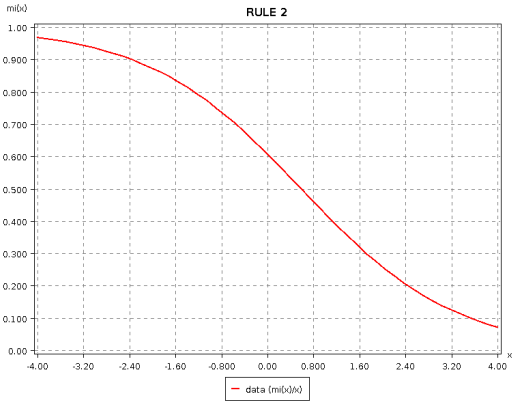


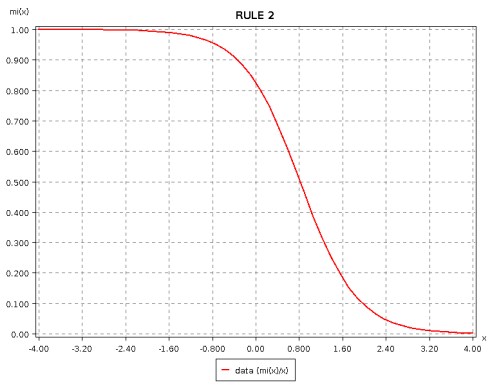
2 RULES

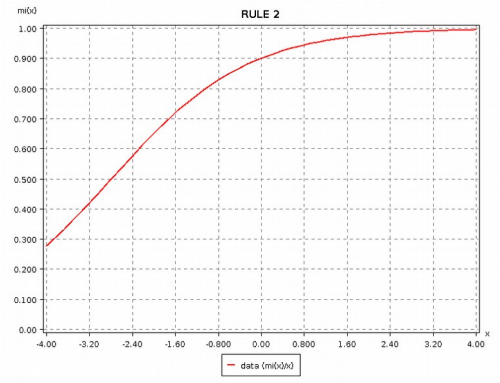
A1

B1

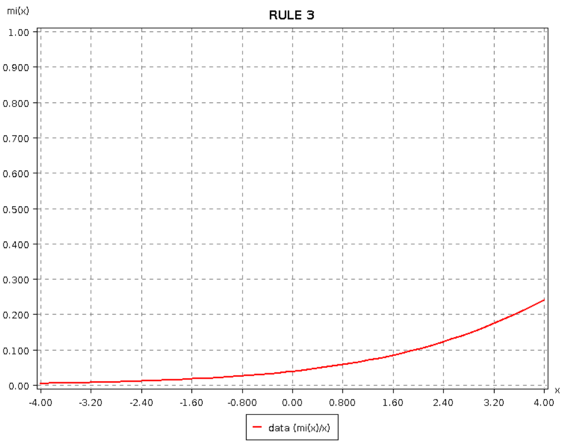


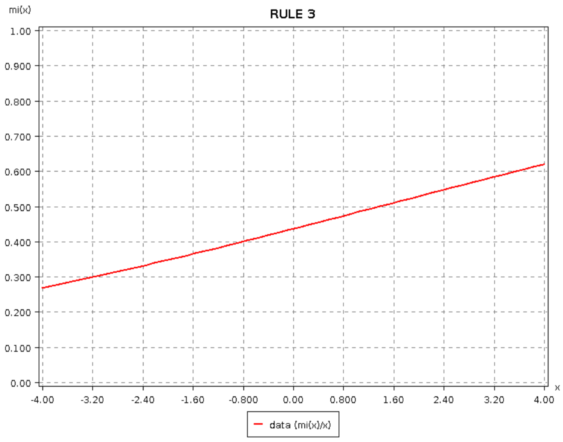


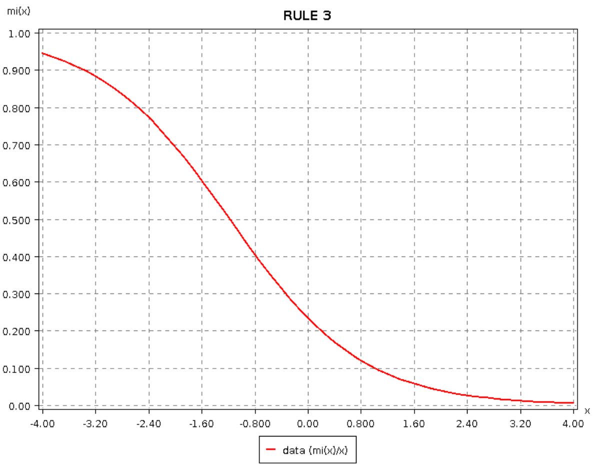


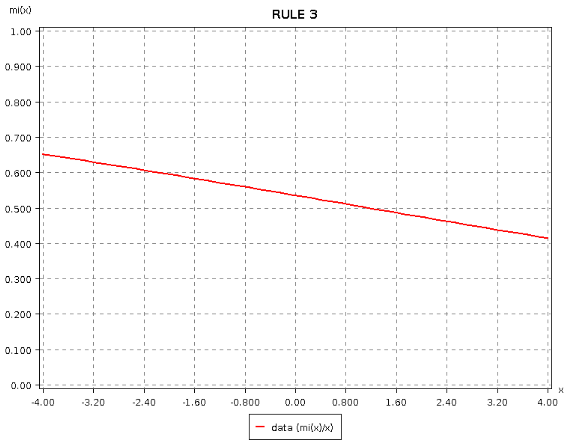


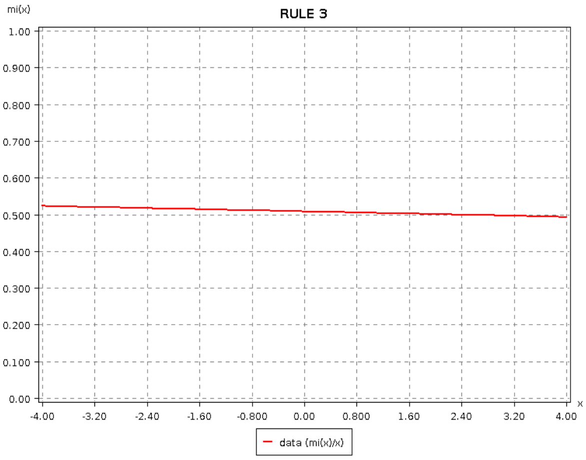
3 RULES

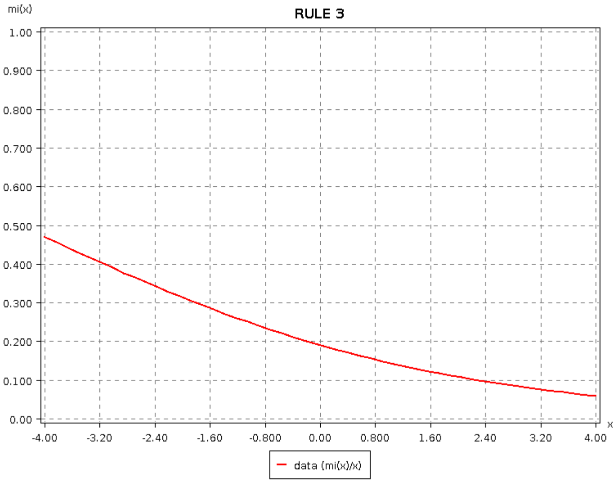

RULE 3
mi(x)
x
data (mi(x)/x)


RULE 3
mi(x)
x
data (mi(x)/x)


RULE 3
mi(x)
x
data (mi(x)/x)

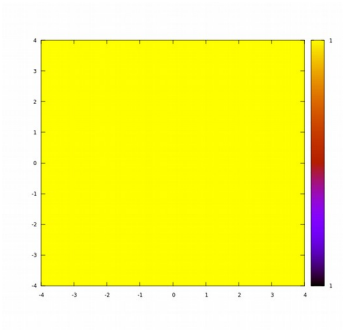

RULE 3
mi(x)
x
data (mi(x)/x)

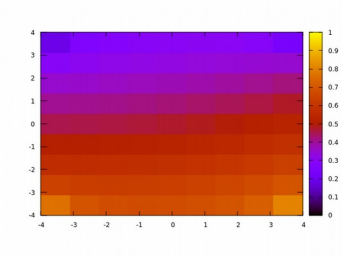

RULE 3
mi(x)
x
data (mi(x)/x)

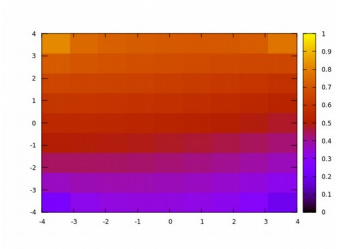

RULE 3
mi(x)
x
data (mi(x)/x)

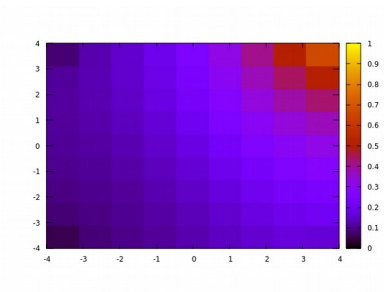
Comments:

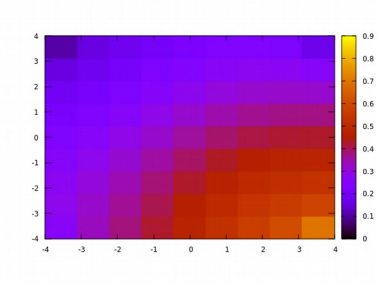
From this analysis we can conclude that each antecenents of any rule is a fuzzy set whose elements are elements of the domain D. This leads to conclusion that every additional rule in ANFIS system decomposes the domain D into subspaces which are approximated by best linear function learned in rules' consequnece part.
This is a reason why ANFIS with only 1 rule cannot learn any nonlinearities. He cannot decompose domain D into subspaces and than approximate each subspace with linearities.
This hypothesis is backed up with following heatmaps. Lighter color denotes elements whose membership function is activated.

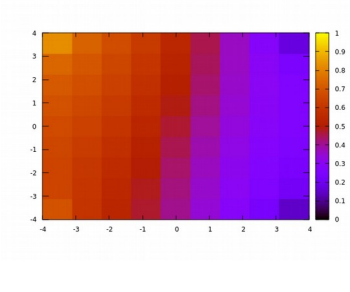










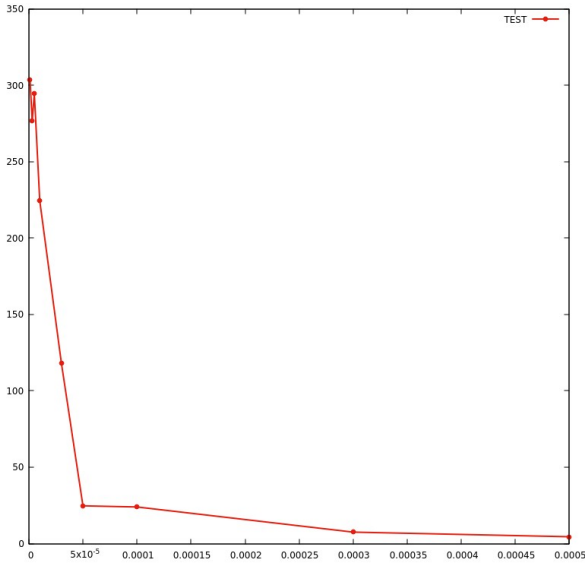


Every additonal neuron decomposes domain in more subspaces.

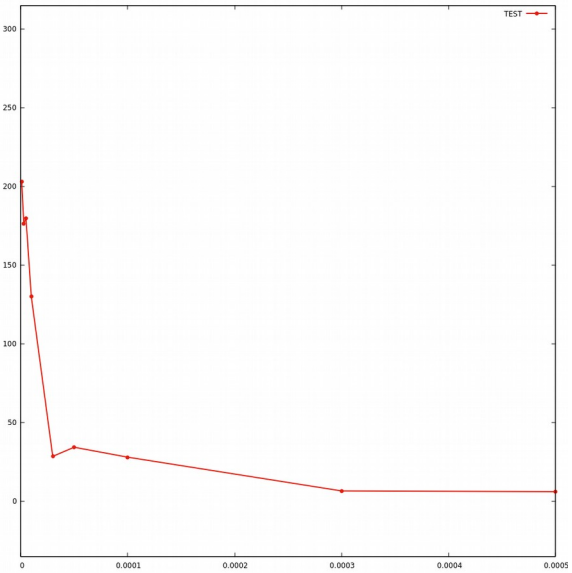
Learning rate

In following section we can see effect of learning rate of model test performance. We have tested ANFIS model with 4, 3 and 2 rules.

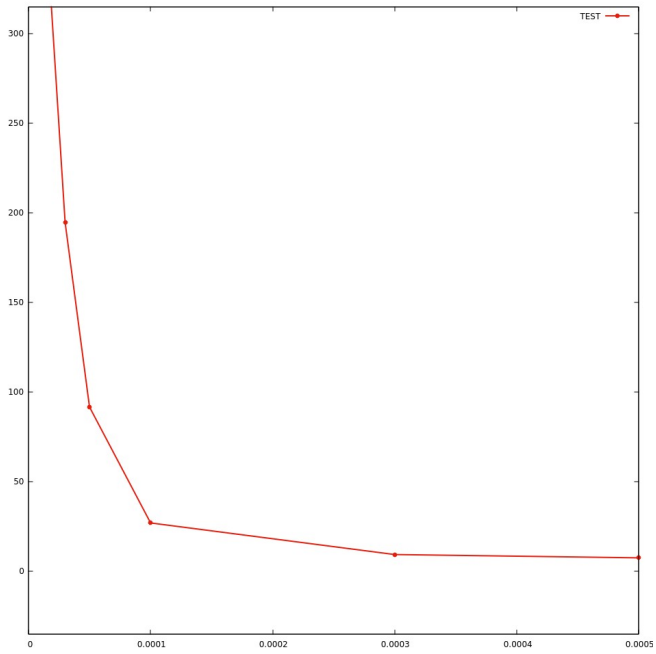
3 RULES



2 RULES

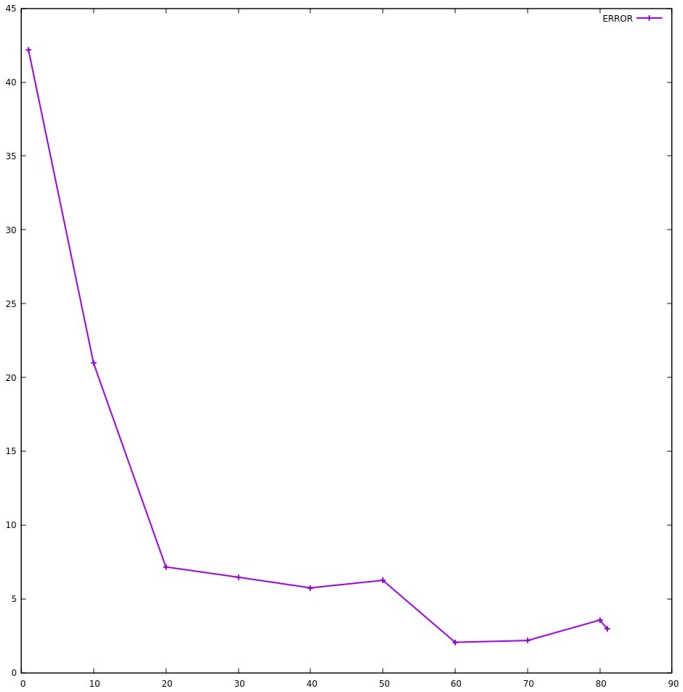


4 RULES

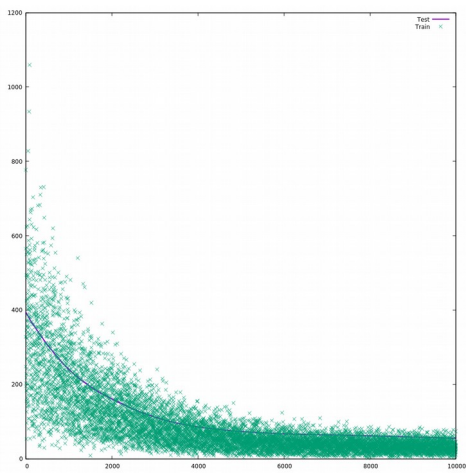


BATCH SIZE

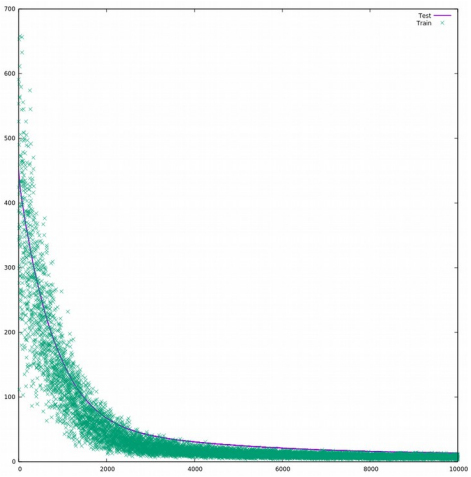
Graph show us influence of different bath sizes on model errror. Obviously, models which learn on bigger batch sizes have smaller error because their gradient is more precise, but his method is computationaly more expansive and can lead to overfitting. Stocahstic gradient descent resolves this problem by moving in direction of gradient approximation and because of that we have more noise in learning.



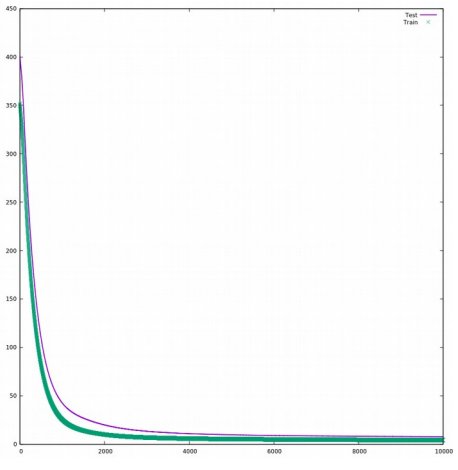
BATCH SIZE = 10

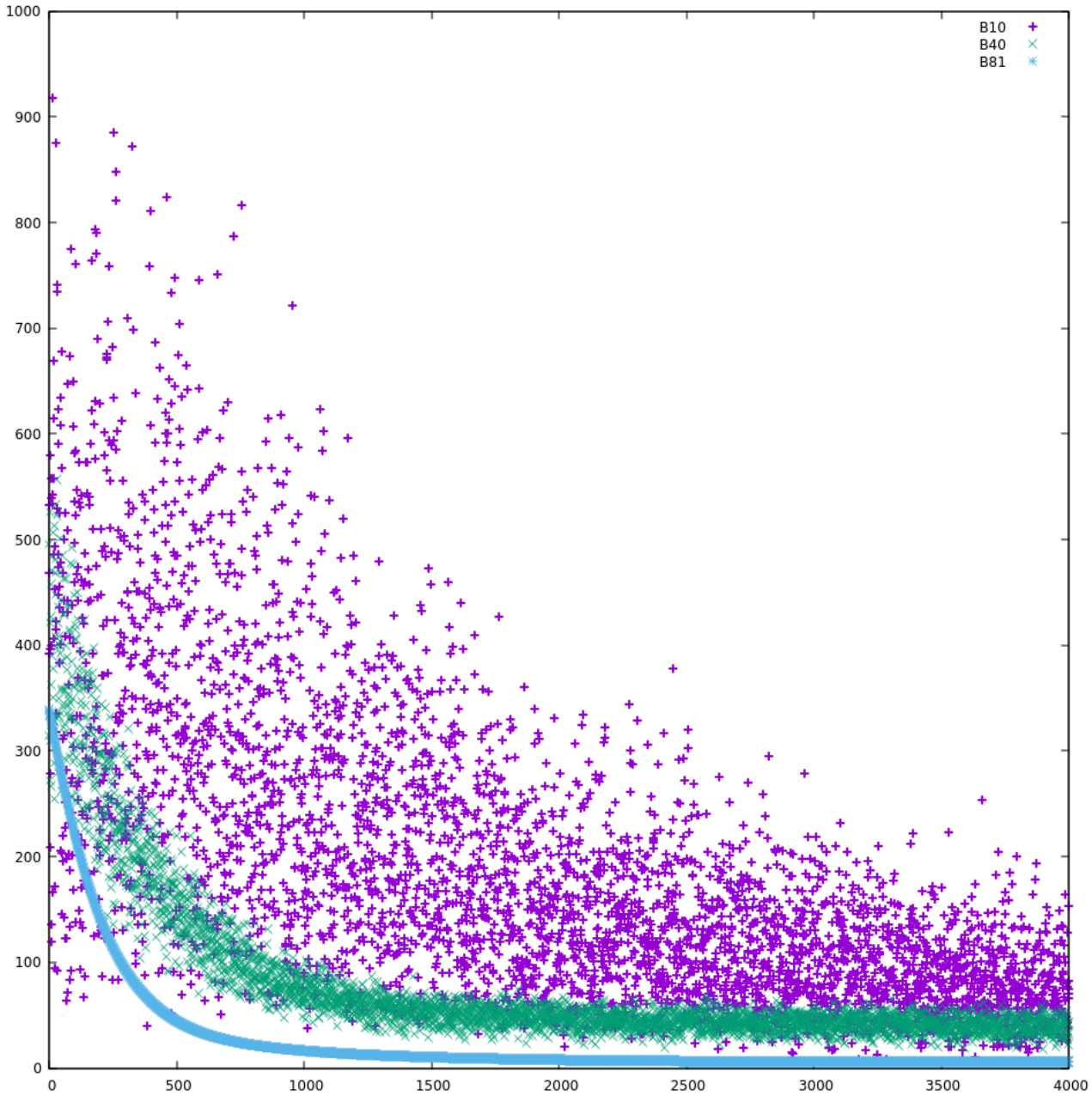


BATCH SIZE = 20



BATCH SIZE = 56 (real gradient)




B10
B40
B81
1000
900
800
700
600
500
400
300
200
100
0
0
500
1000
1500
2000
2500
3000
3500
4000