CNS 187 Neural Computation - Problem Sheet 7

Ling Li, ling@cs.caltech.edu

November 17, 2000

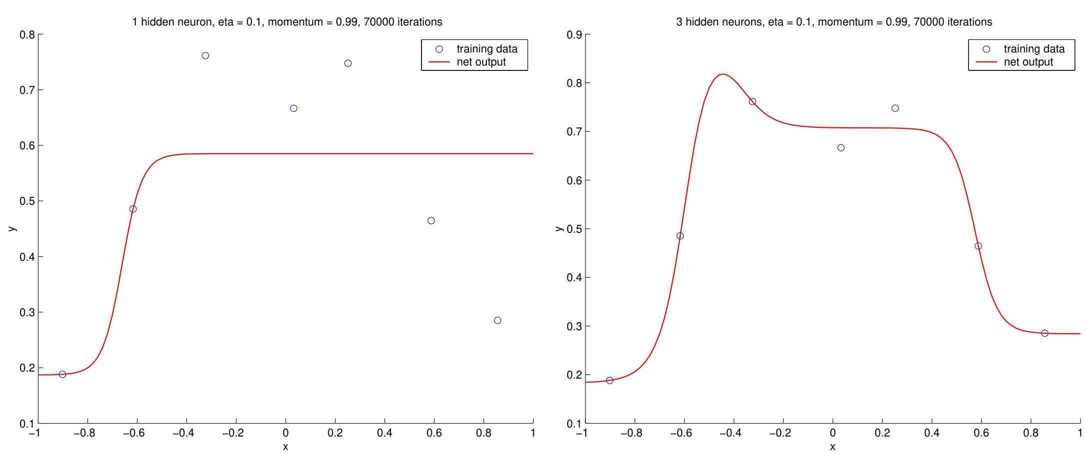
The mean-square-error

$$E = \frac{1}{N}\sum_{\mu=1}^{N} |d^\mu - y^\mu|^2 = \frac{1}{N}\sum_{\mu=1}^{N}\sum_{i=1}^{m} (d_i^\mu - y_i^\mu)^2$$

is used as the measure of errors in this homework. The notations follow the Problem Sheet 5.

7.1 Interpolation with multilayer networks

I used 2, 4, 8 and 16 hidden neurons to interpolate the given data `set3`.[1] Plots of them are shown in Figure 2. Here in Figure 1, I want to first display two plots of the results when using 1 and 3



(a) 1 hidded neuron

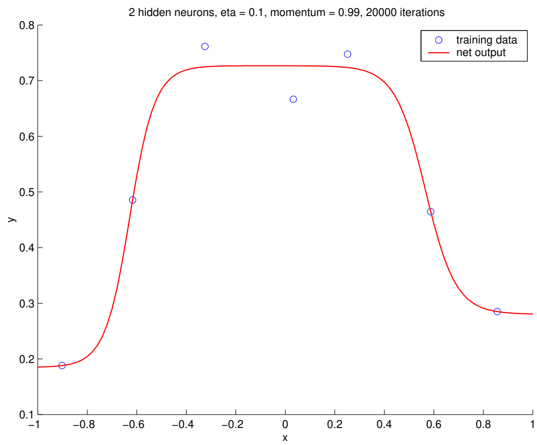
(b) 3 hidden neurons

Figure 1: Interpolate data `set3` with 1 and 3 hidden neurons.

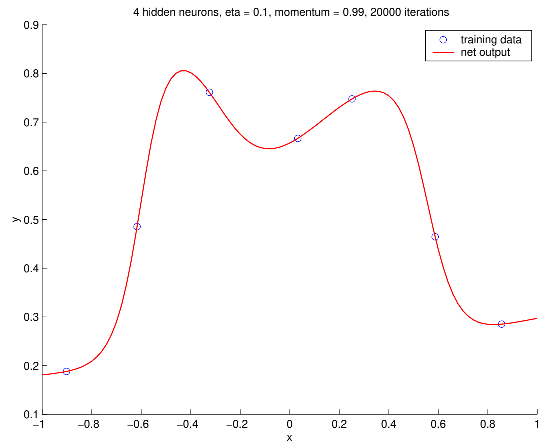
hidden neurons. With 70,000 iterations, the two networks still can not fit the training data well.

The output of the network with 1 hidden neuron *seems* to be a sigmoid function; The output of the network with 3 hidden neurons *seems* to be the concatenate of 3 sigmoid functions, or more precisely, the weighted sum of 3 different sigmoid functions. Look at plots (a) and (b) in Figure 2,

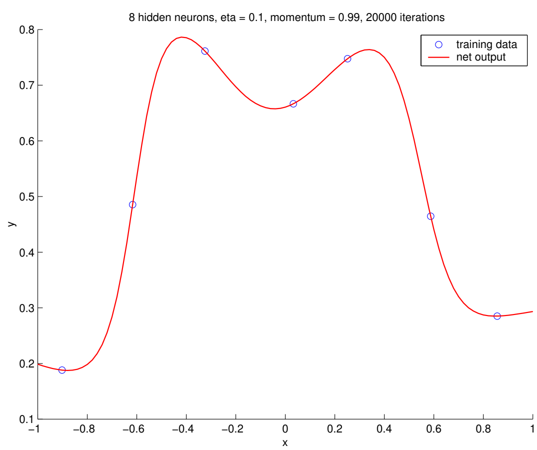
[1] From homework 3.3 of CS156a Learning Systems, I knew that for this kind of learning problems, the learning rate $\eta = 0.1$ and the momentum $\mu = 0.99$ is a good combination.



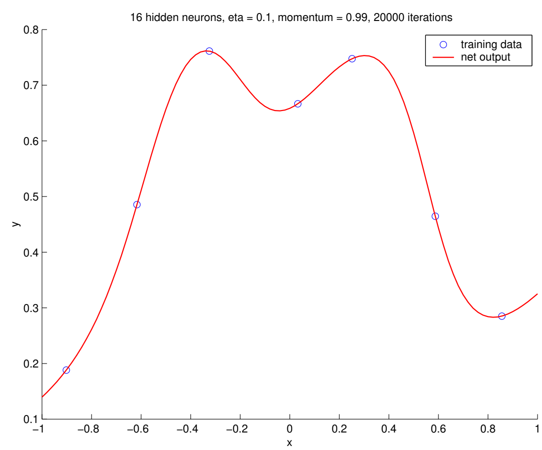
(a) 2 hidded neurons



(b) 4 hidden neurons



(c) 8 hidden neurons



(d) 16 hidden neurons

Figure 2: Interpolate data `set3` with 2, 4, 8, and 16 hidden neurons.

we may be more confident to say that the network *seems* to perform a weighted sum of several sigmoid functions. This should be approximately right when the training data aren neither too large nor too small, thus the output neuron would work approximately as a weighted adder. So, the interpolate becomes twistier when the number of hidden neurons increases.

From the results of 4, 8, and 16 hidden neurons (plots (b), (c) and (d) in Figure 2), I found out that 4 hidden neurons is sufficient for fitting the training data, and the networks with 8 or 16 hidden neurons did not show much advantage to that with 4 hidden neurons.

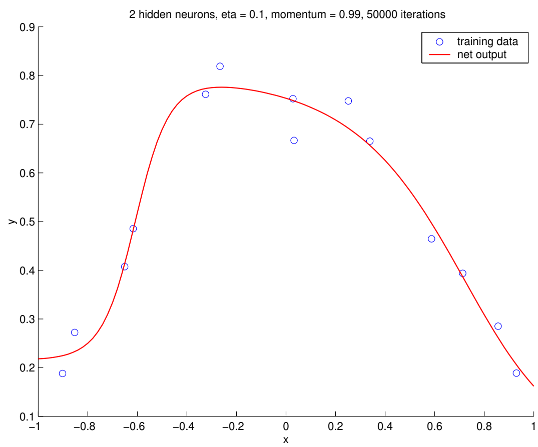
The training errors at iteration 20,000 (associated with Figure 2) are in Table 1. Though the errors vary according to the initial weights and network parameters, this set of errors shows that error E decreases with the number of hidden neurons which can be taken as a measure of the learning model complexity; The error drops dramatically from 2 to 4 hidden neurons, which is hinted by previous discussion that 4 hidden neurons begin to be sufficient for fitting the training data.

Table 1: Errors on `set3` and `set5` when trained with `set3`.

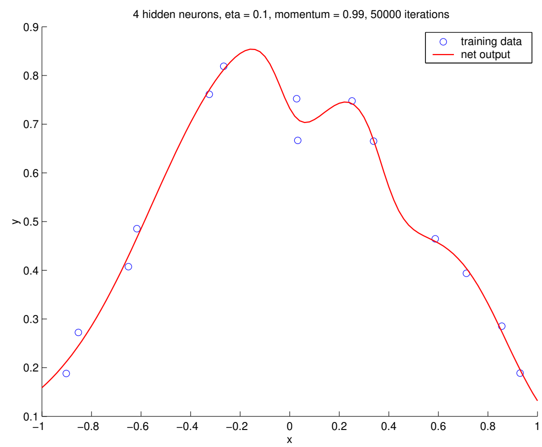
# of hidden neurons	2	4	8	16
errors on `set3`	7.8835×10^{-4}	6.6582×10^{-32}	4.6663×10^{-32}	1.0235×10^{-32}
errors on `set5`	0.007490	0.014171	0.013291	0.012307

However, I can not tell which network is the best, though those with 4 or more hidden neurons definitely fit the training data much better than the one with 2 hidden neurons. The networks with 4, 8, and 16 hidden neurons are equally good (according to this training problem).

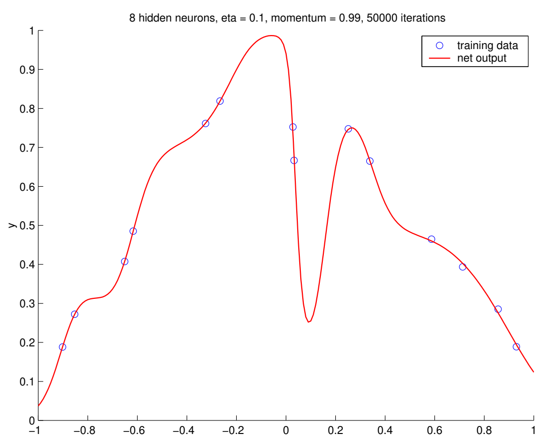
7.2 Generalization



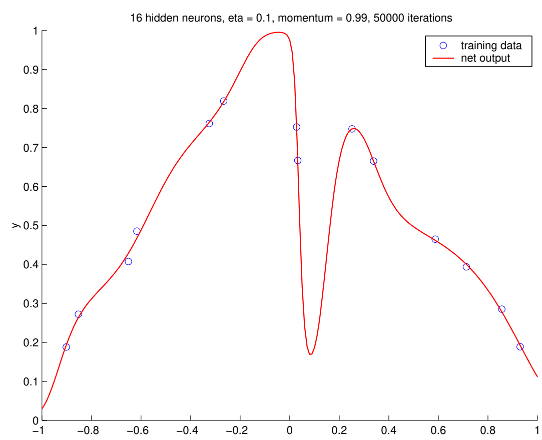
(a) 2 hidded neurons



(b) 4 hidden neurons



(c) 8 hidden neurons



(d) 16 hidden neurons

Figure 3: Interpolate data `[set3,set4]` with 2, 4, 8, and 16 hidden neurons.

1. The errors those 4 networks (2, 4, 8, and 16 hidden neurons) generated for data `set5` are also

listed in Table 1. The network with 2 hidden neurons generalized the best.

2. With set3 and set4 together as the training data, we get the plots in Figure 3 and the errors

Table 2: Errors on `[set3,set4]` and `set5` when trained with `[set3,set4]`.

# of hidden neurons	2	4	8	16
errors on `[set3,set4]`	0.001188	4.3465×10^{-4}	4.6942×10^{-5}	7.0296×10^{-5}
errors on `set5`	0.007984	0.007722	0.015241	0.017886

in Table 2. Even with the number of iterations being increased to 50,000, the errors here are much larger than those in Table 1.

3. Now we can see in the Table 2 that the network with 4 hidden neurons generalized the best. However, the network with 2 hidden neurons did the same good work.

   So, as I said before, the number of hidden neurons is a measure of the model complexity and thus for more complex data, more hidden neurons are needed. However, increasing the model complexity will decrease the capability of generalization of the network. As we can see from plots (c) and (d) in Figure 3, the over-fitting to the training data degrades the performance on out-of-sample data. These may explain why the network with 4 neurons became the best one on generalization.

   Thus, if some *a priori* assumption is known about the data, constructing a network with model complexity just in the same level of the data is a wise choice. Not too complex, not too simple. In this problem, I guess the underlying function is some parabola. Thus a network with 2–4 hidden neurons should be suitable to this problem.

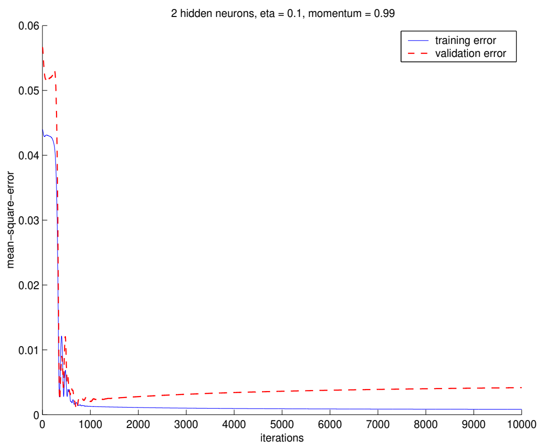
4. When training with validation, the error curves are shown in Figure 4. While the training errors decrease with the number of iterations, the validation errors begin to increase, after some 2000 iterations. The errors of early-stopping on `set5` is shown in Table 3. I also list the

Table 3: Errors on `set5` with and without early-stopping.

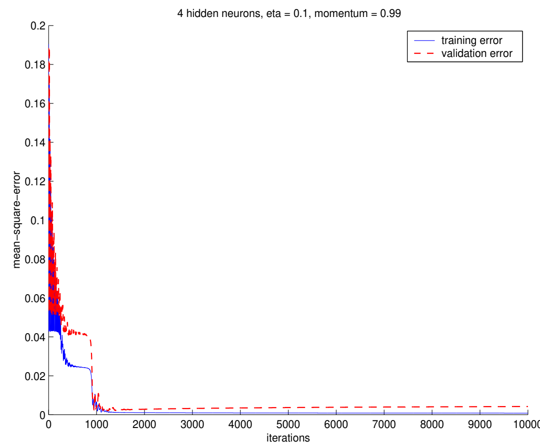
# of hidden neurons	2	4	8	16
errors on `set5`	0.005700	0.006954	0.008814	0.005649
errors at the end	0.007218	0.007279	0.010039	0.013383

errors on `set5` at the end of the training, which is under the same condition with errors in Table 1 though with different number of iterations.

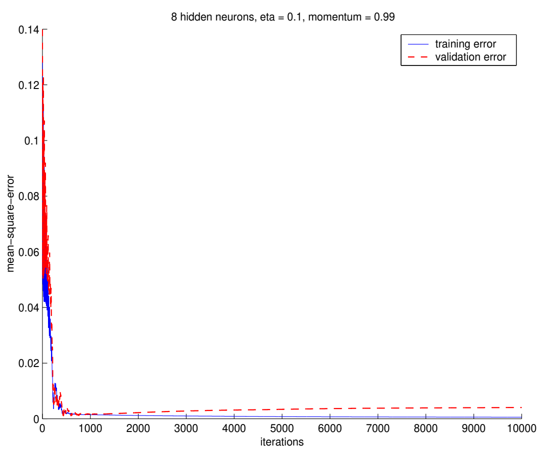
   Early-stopping stops the exploration of the weights space at an early time, thus confine the complexity of the learning model. This is why the out-of-sample errors with early-stopping are usually smaller than those without early-stopping.



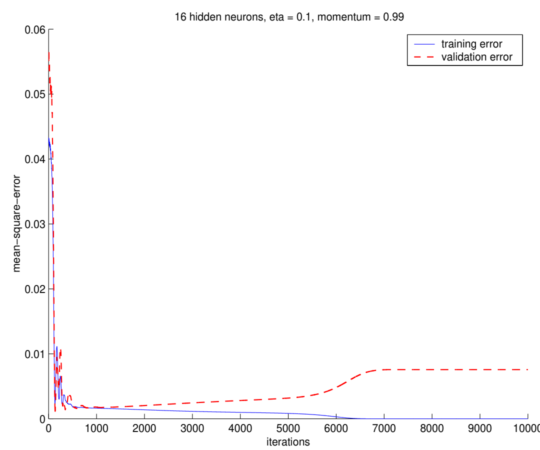
(a) 2 hidden neurons



(b) 4 hidden neurons



(c) 8 hidden neurons



(d) 16 hidden neurons

Figure 4: Training with validation. The training data is `set3` and the validation data is `set4`.