

### Solutions to Problem set 3

#### Problem 1. Exploratory data analysis

1. CHAS is the only binary attribute.

Attribute	Correlation
CRIM	-0.3883
ZN	0.3604
INDUS	-0.4837
CHAS	0.1753
NOX	-0.4273
RM	0.6954
AGE	-0.3770
DIS	0.2499
RAD	-0.3816
TAX	-0.4685
PTRATIO	-0.5078
B	0.3335
LSTAT	-0.7377

2.

LSTAT attribute has the highest negative and RM the highest positive correlation with the target attribute.

3. Attributes 9 and 10 have the largest mutual correlation in the dataset: 0.9102.

#### Problem 2. Linear regression.

**Part d.** The values of regression coefficients  $\mathbf{w} = [w_0 w_1 w_2 \dots w_{13}]^T$  obtained on the training set are:

$$\mathbf{w} = \begin{bmatrix} 39.584 & -0.10114 & 0.045894 & -0.0027304 & 3.072 & -17.225 & 3.711 \\ 0.0071586 & -1.599 & 0.37362 & -0.015756 & -1.0242 & 0.0096932 & -0.58597 \end{bmatrix}^T$$

For the linear model, we define the mean square error on  $N$  samples  $(\mathbf{x}_i, y_i), i = 1, \dots, N$  as:

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{x}_i^T \cdot \mathbf{w})^2$$

The mean squares error on the training set is  $MSE_{training} = 22.081$ . The mean squares error on the test set is  $MSE_{test} = 22.638$ . We could expect such results since assuming that data distribution on the training and on the test set is the same (has the same variances), the error on the training set is smaller than on the unseen data from the test set.

### Problem 3. Online linear regression

**Part c.** The weights and errors diverge until the learning crashes.

**Part d.** We used the following *LMS* rule for updating the weights:

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} + \alpha \cdot \epsilon \cdot \mathbf{x}^{(i)}$$

where  $\epsilon = y^{(i)} - \mathbf{x}^T \cdot \mathbf{w}^{(i)}$  and  $\alpha$  is a learning rate. We have performed a number of experiments varying initial weights and learning rate schedules.

In this assignment you were asked to do only an experiment with zero initial weights, learning schedule  $\alpha = 2/i$  and 1000 update steps. Figure 1 shows the progress of mean squares errors for this set of parameters. We see that the procedure is rather unstable at the beginning and errors are large. This behavior can be attributed to large weights fluctuations and corrections. However, after some number of steps (despite some error fluctuations) we see gradual improvement of error performance which is promising. Unfortunately, the errors after 1000 training steps are still very large (train: 1.524e+005, test: 1.7134e+005) are very large and thus not very encouraging.

**Part e.** The question that arises immediately is whether we did not stop the on-line procedure prematurely. Figure 2 illustrates the error progress for the experiment with 10000 steps. We observe significant error improvement (two orders of magnitude). Unfortunately, we also see that even after 10000 steps the errors (train:3.8096e+003 test:7.6902e+003) are not that good and still very far from the solution for Problem 2.

Since our error results are not that good we turn to the possibility of changing the parameters of the on-line learning procedure. First we try to change initial weights. Figure 3 illustrates the performance of the procedure on a random set of weights (every weight is selected randomly from the interval between 0 and 1). We observe that errors and curves changed from the previous experiment. The weights obtained after 1000 steps lead to (train:1.7310e+005 test:1.9450e+005) errors. Thus we can conclude that the on-line procedure is sensitive to the initial choice of weights. Unfortunately we also observe that errors are of about the same magnitude. Also, similarly to the above experiments we observe that weights tend to fluctuate widely over the experiment.

The above observations suggests that corrections of weights in the on-line update are too high leading to large errors. To dampen the corrections we try to modify the learning rate schedule and use  $\alpha = 0.01/i$ . Figure 4 illustrates the effect of this change on the errors. We see that squared errors are much better and closer to errors in part a. The errors for weights obtained after 1000 steps are (train: 508.7148, test: 406.7090).

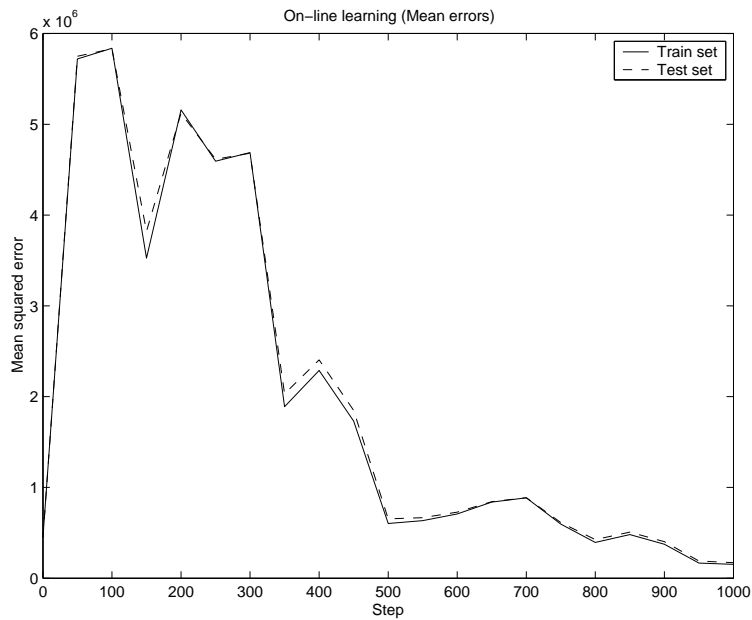


Figure 1: Zero initial weights,  $\alpha = 2/i$ , 1000 steps.

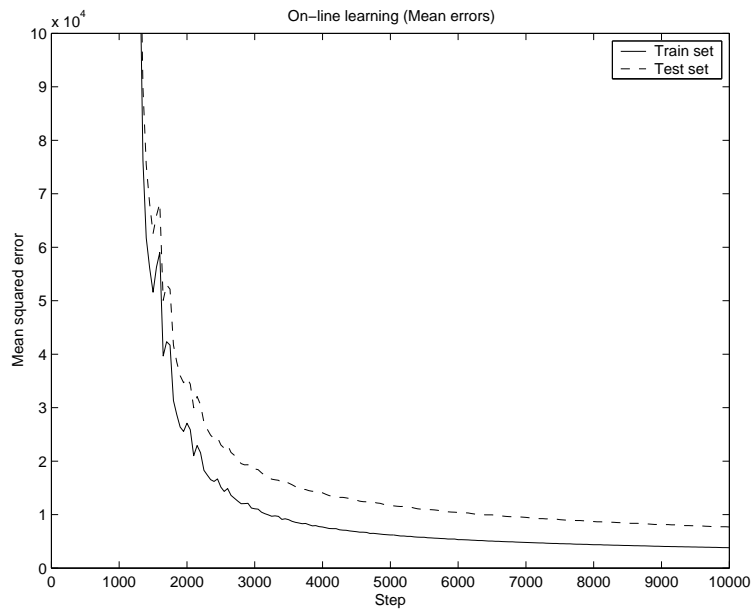


Figure 2: Zero initial weights,  $\alpha = 2/i$ , 10000 steps.

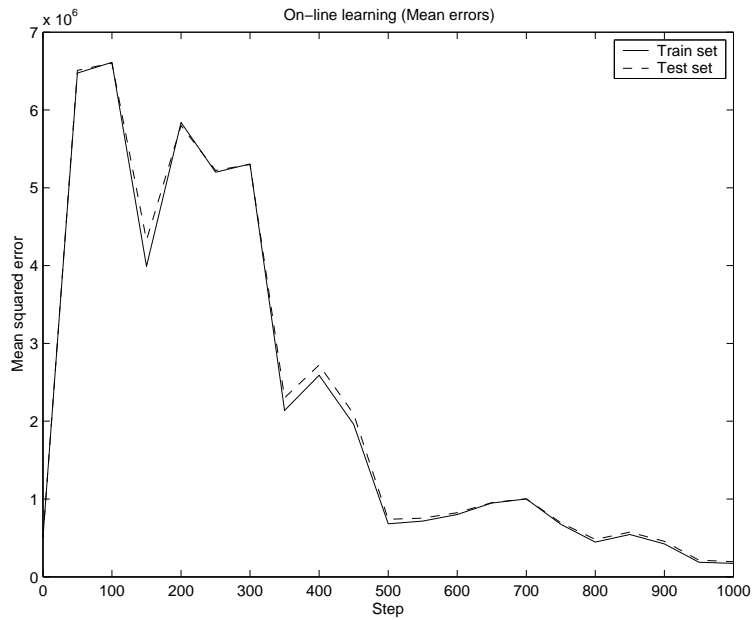


Figure 3: Random initial weights,  $\alpha = 2/i$ , 1000 steps.

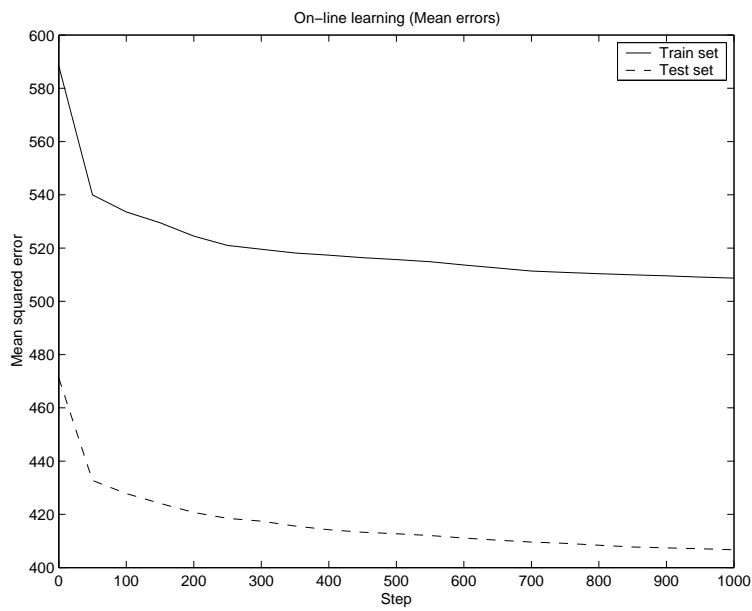


Figure 4: Zero initial weights,  $\alpha = 0.01/i$ , 1000 steps.

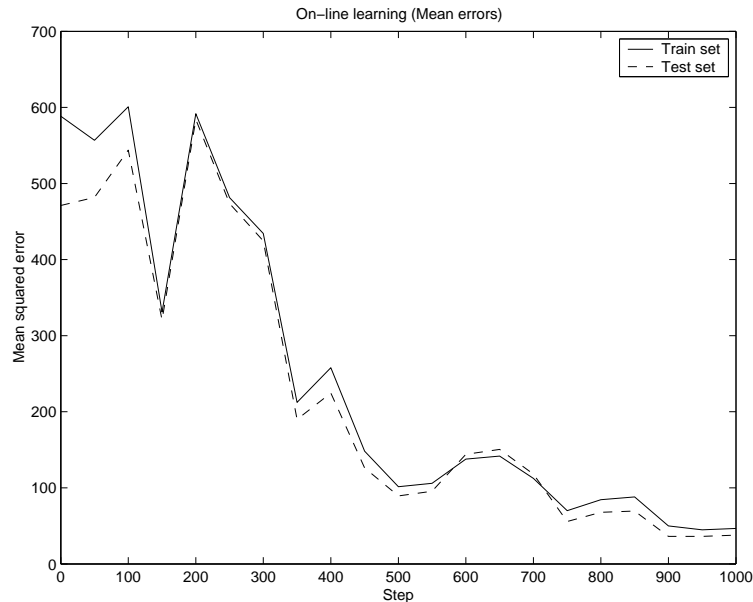


Figure 5: Zero initial weights,  $\alpha = 0.01$ , 1000 steps.

Analyzing the results for the smaller initial weight we observe that the progress of improvement is highest at the beginning and levels afterwards. This suggests that for larger step values the learning rate may be too small. To test this hypothesis we use the fixed learning rate  $\alpha = 0.01$ . The results of the experiment are shown in Figure 5. Using the fixed learning rate the errors after 1000 steps were (train: 46.5866 test: 37.8884) which is quite close to errors from part a.

Figure 6 illustrates the progress of errors for the fixed learning rate  $\alpha = 0.01$  and 10000 steps. We see that errors settle rather quickly (after about 1500 steps) in a reasonable range. Further fluctuations are due to the on-line corrections of weights. The final errors (train: 26.0923 test: 21.2000) are very good.

The fact that we can often use a constant learning rate to learn the weights is not surprising. However, this learning rate needs to be found. Choosing the correct fixed learning rate is not an easy task. If the choice is high, errors tend to fluctuate in a 'wide' region over time and they do not settle on a specific value. Even worse, when a fixed learning rate is too high the solution may diverge. On the other hand, small learning rate slows down the learning since corrections to weights can be very small. Finding the appropriate learning rate means to find a value in between the two extremes.

The advantage of the annealed learning rate is that it converges in the limit since the

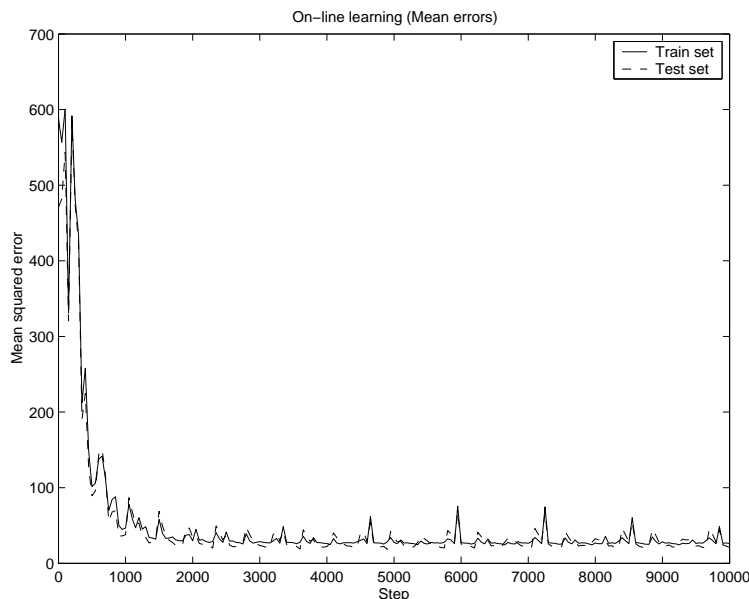


Figure 6: Zero initial weights,  $\alpha = 0.01$ , 10000 steps.

learning rate is changed slowly over iteration steps. However, the convergence may also take a long time since learning rates may get small rather quickly.

From the above analysis we see that the problem of tuning gradient descent algorithms is highly problem specific and not that straightforward. This problem is even more pronounced when used in conjunction with non-linear models (multilayer neural networks) with many local optima on the error function. Numerous solutions modifying and improving learning rates and on-line learning have been proposed. For example, one solution is to consider not only the current gradient value but also previous step gradient (so called momentum).

#### Problem 4. Regression with polynomials

**Part b.** The binary attribute was effectively replicated.

**Part d.** In this case, linear regression was performed by generating additional input attributes  $x_k = x_i x_j$  where  $k = f(i, j)$  and performing linear regression on new attributes. We obtained  $MSE_{training} = 5.4293$  and  $MSE_{test} = 36.26$ . Therefore, although training error was much smaller than for a linear model, the quadratic model was outperformed by a linear model on test data. We could expect such behavior since non-linear model with 105 parameters was fitted on only 432 samples. When very complex models are trained on rather small data sets, they typically cannot generalize well on test datasets.