CSC 242 Project 4 Report

By Jordan Altomare, Abdelmonem Khedr, Piyush Saini

Compile Instructions:
Be sure that you are in the folder called CSC242_Project4.
Type: javac learn/Test.java

Running Instructions:
Type java learn.Test {fileName} {classifier} {steps} {alpha}

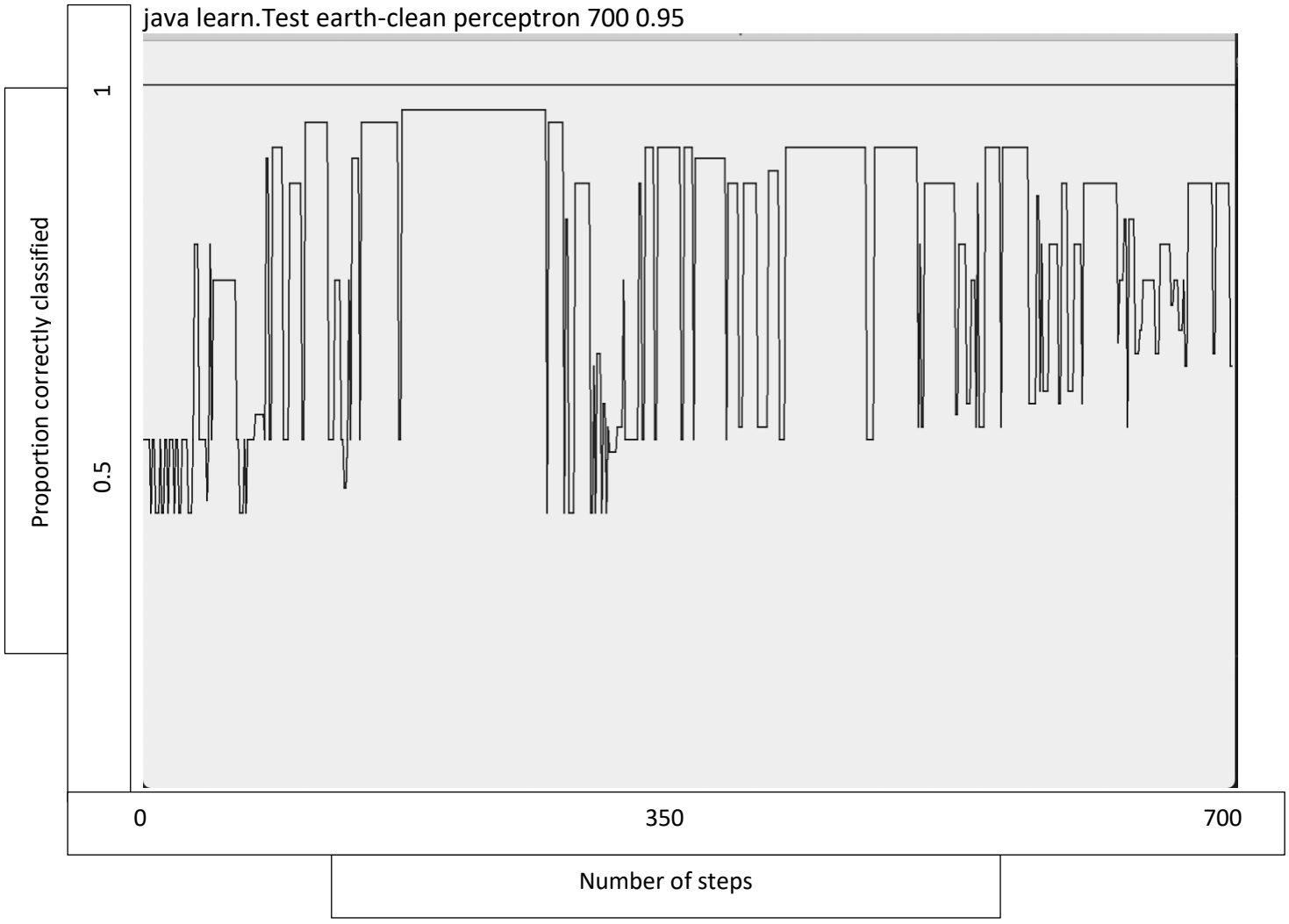
a. The file name can be any of the following:(make sure its lowercase)
   earth-clean
   earth-noisy
   house
b. The classifier can be any of the following:(make sure its lowercase)
   perceptron
   logistic
c. Enter any number of steps to run algorithm
d. For a constant alpha put a double between 0 and 1 and for a decaying alpha put 0

Example of how to run:
java learn.Test earth-clean perceptron 700 0.95

What we did:
We used an update learning rule to continually update weights for a perceptron, which allows us to find the weights for the line that best separates data into two separate sides. We used this update rule in combination with a threshold rule. For the perceptron, the threshold rule was if **W**dotprod**X** was greater than or equal to zero, return 1, else return 0. This was what is known as a hard threshold. We did similar things for a logistical linear classifier. We had a slightly different update rule for the weights. The threshold calculated with the following equation:
$1/(1 + e^{-\mathbf{w}\cdot\mathbf{x}})$
This is what is known as a soft threshold.

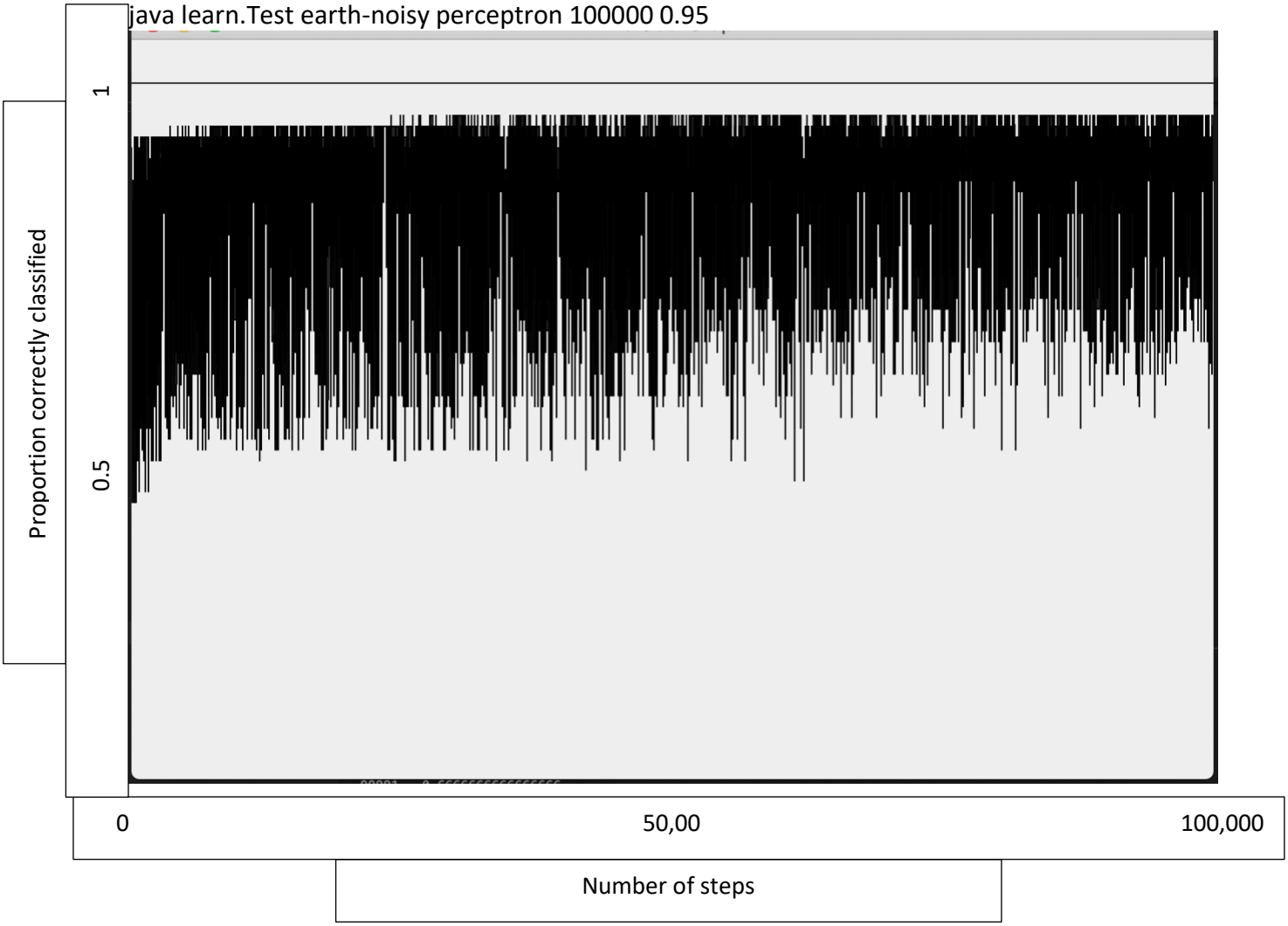
We ran our perceptron and logistic linear classifier on several data sets. For the perceptron, we printed out the step number and proportion correct. For the logistic linear classifier, we printed out the step# and the squared error. We also used Java to draw graphs for both types of linear classifiers. Below are the results of our project.

Perceptron run on the earthquake-clean data for 700 steps with alpha=0.95. The y-axis represents proportion correctly classified. It starts from the bottom of the picture and goes to the drawn horizontal line. Its range is from 0 to 1. The x-axis is number of steps. It ranges from 0 to 700. As can be seen, the graph starts with values around 0.4 and 0.5, then gradually improves. If you have already compiled, you can run this code using the following command:
java learn.Test earth-clean perceptron 700 0.95

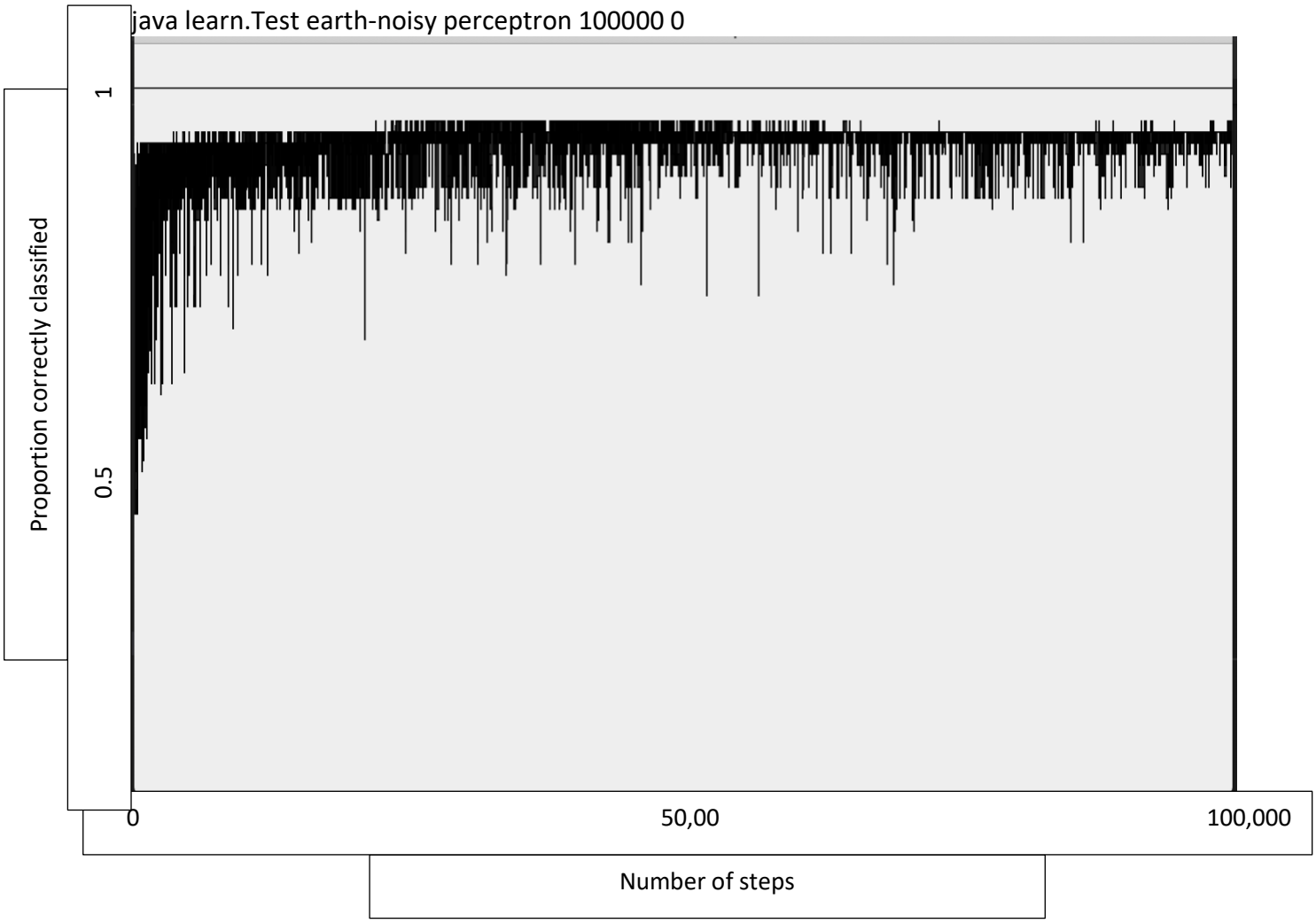


Perceptron run on the earthquake-noisy data for 100,000 steps with alpha=0.95. The y-axis represents proportion correctly classified. It starts from the bottom of the picture and goes to the drawn horizontal line. Its range is from 0 to 1. The x-axis is number of steps. It ranges from 0 to 100,000. As can be seen, the graph starts with values around 0.45 and 0.55, then improves a bit, but still fluctuates a lot due to the noise of the data. If you have already compiled, you can run this code using the following command:

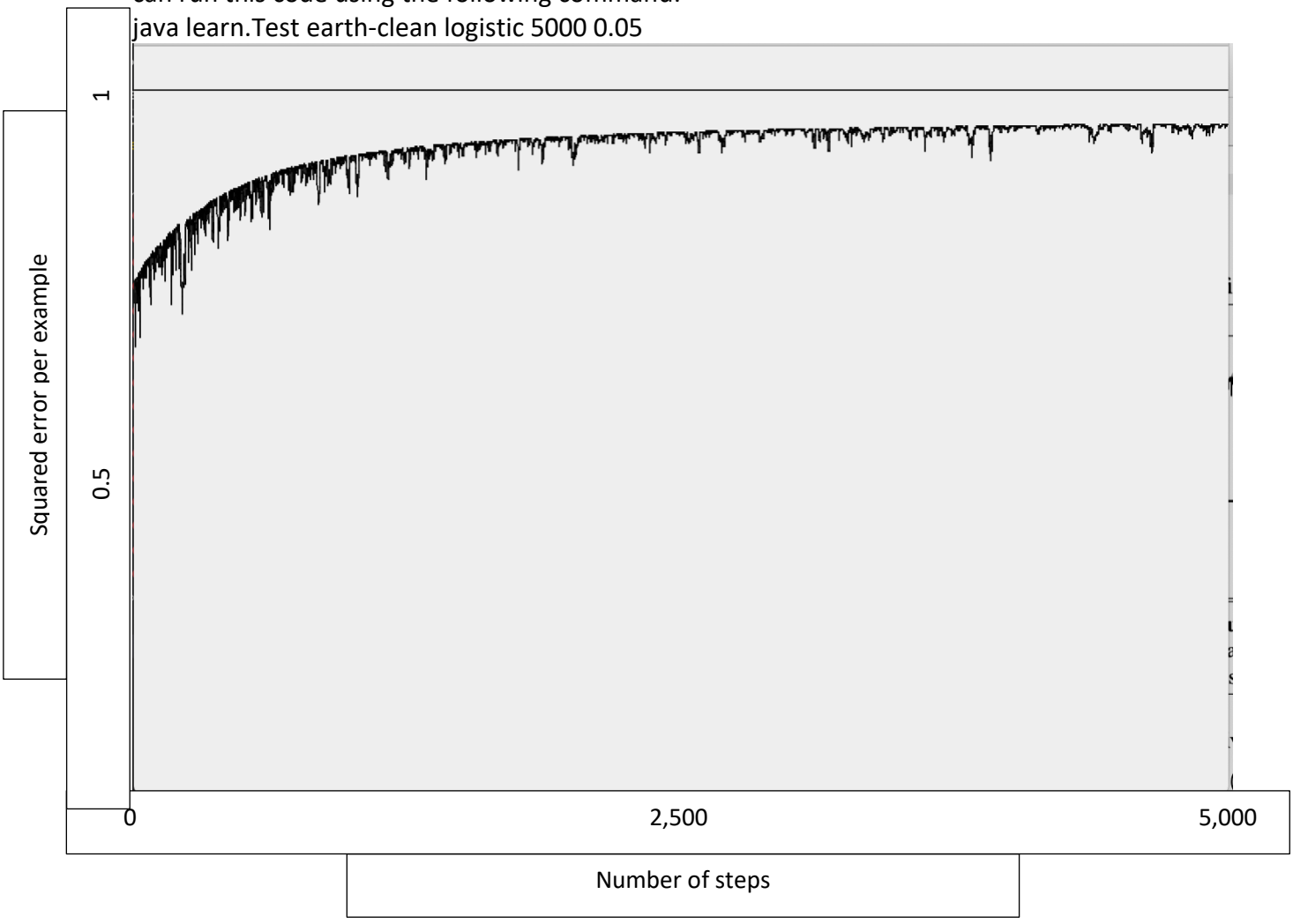
java learn.Test earth-noisy perceptron 100000 0.95



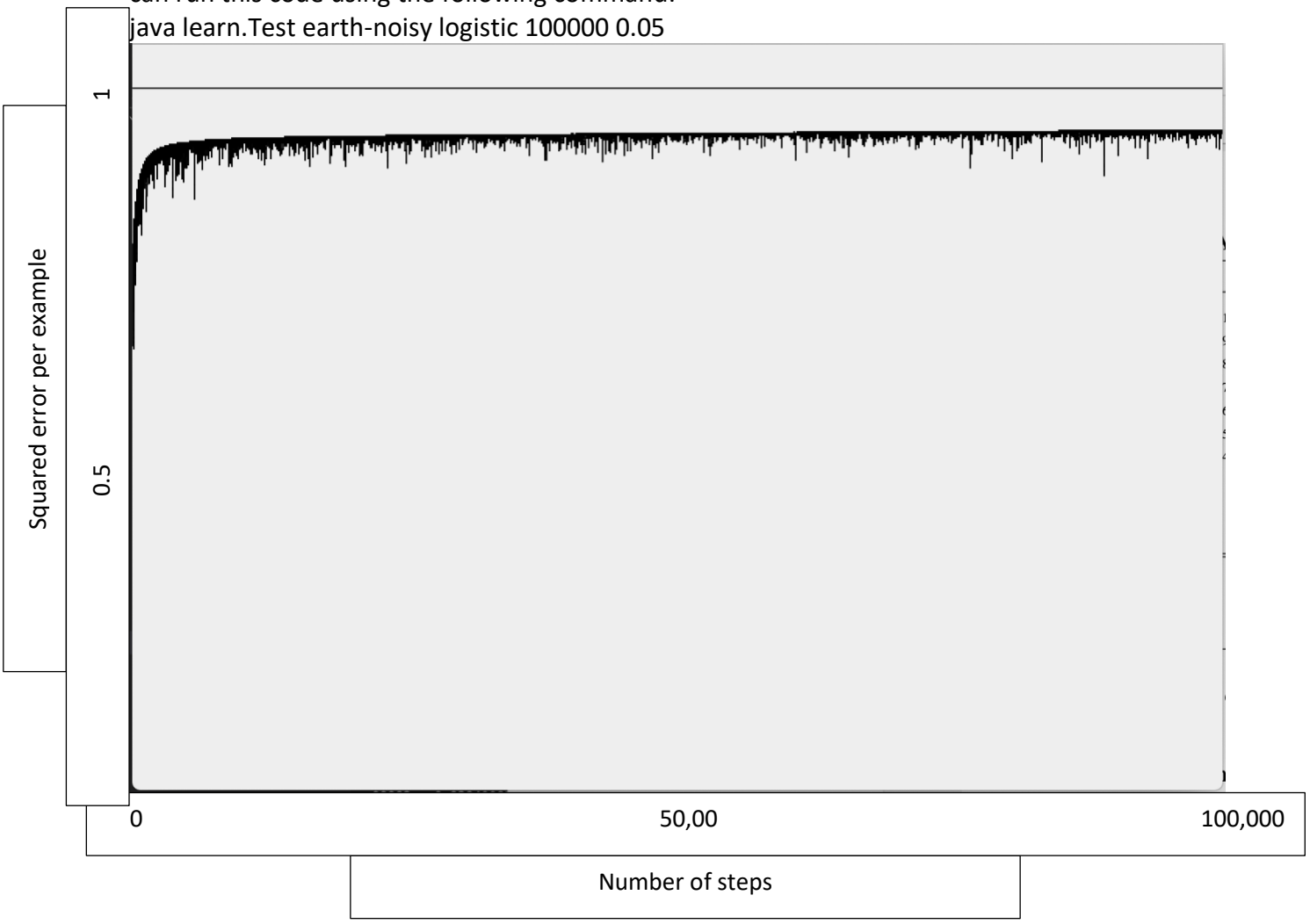
Perceptron run on the earthquake-noisy data for 100,000 steps with a decaying learning rate. The y-axis represents proportion correctly classified. It starts from the bottom of the picture and goes to the drawn horizontal line. Its range is from 0 to 1. The x-axis is number of steps. It ranges from 0 to 100,000. As can be seen, the graph starts with values around 0.45 and 0.55, then improves significantly because alpha is decaying rather than fixed. If you have already compiled, you can run this code using the following command:
java learn.Test earth-noisy perceptron 100000 0



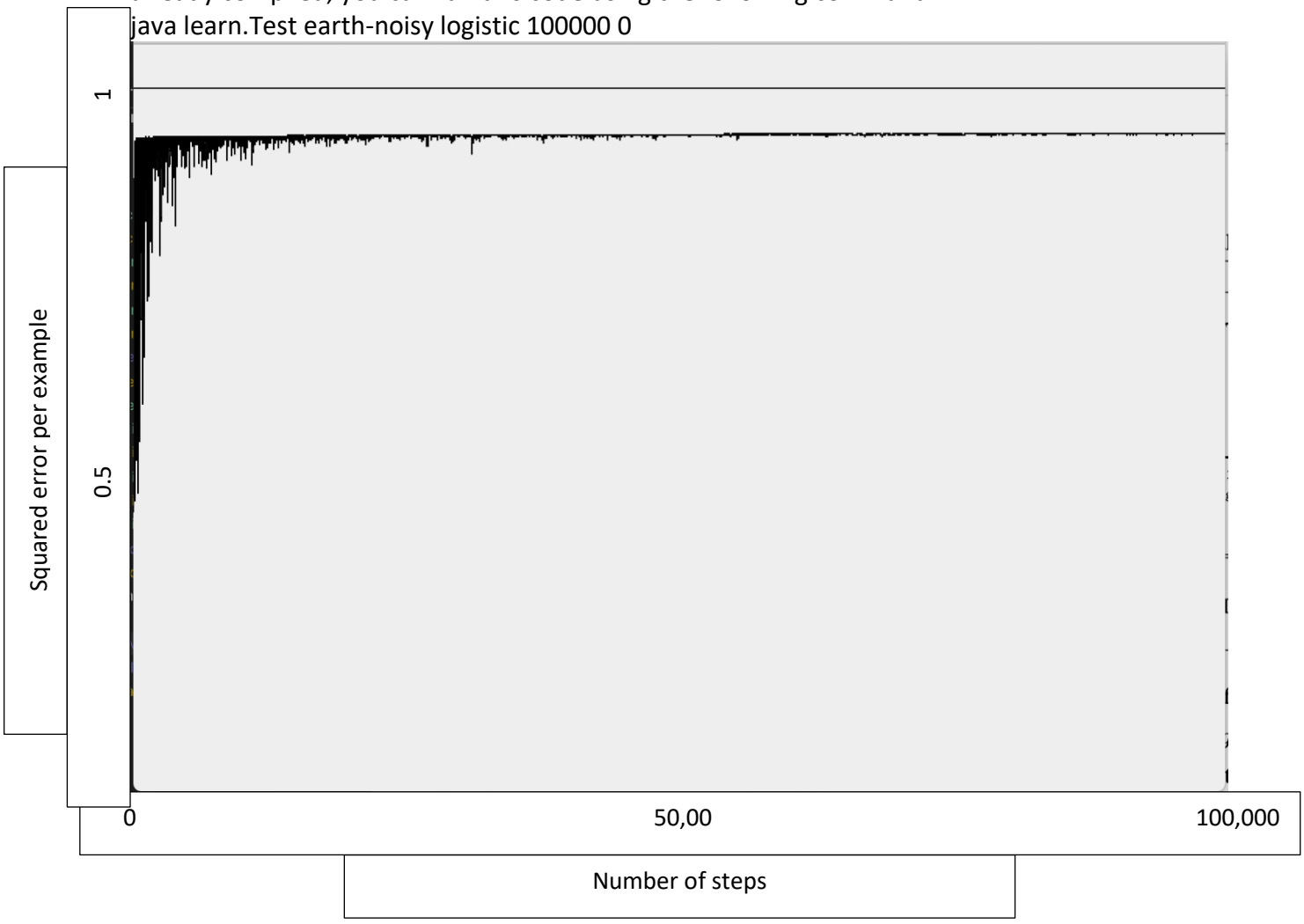
Logistic linear classifier run on the earthquake-clean data for 5,000 steps with alpha = 0.05.
The y-axis represents squared error per example. It starts from the bottom of the picture and goes to the drawn horizontal line. Its range is from 0 to 1. The x-axis is number of steps. It ranges from 0 to 5,000. As can be seen, the graph looks like a logistic curve, as expected, starting with steep improvement which gradually levels out. If you have already compiled, you can run this code using the following command:
java learn.Test earth-clean logistic 5000 0.05



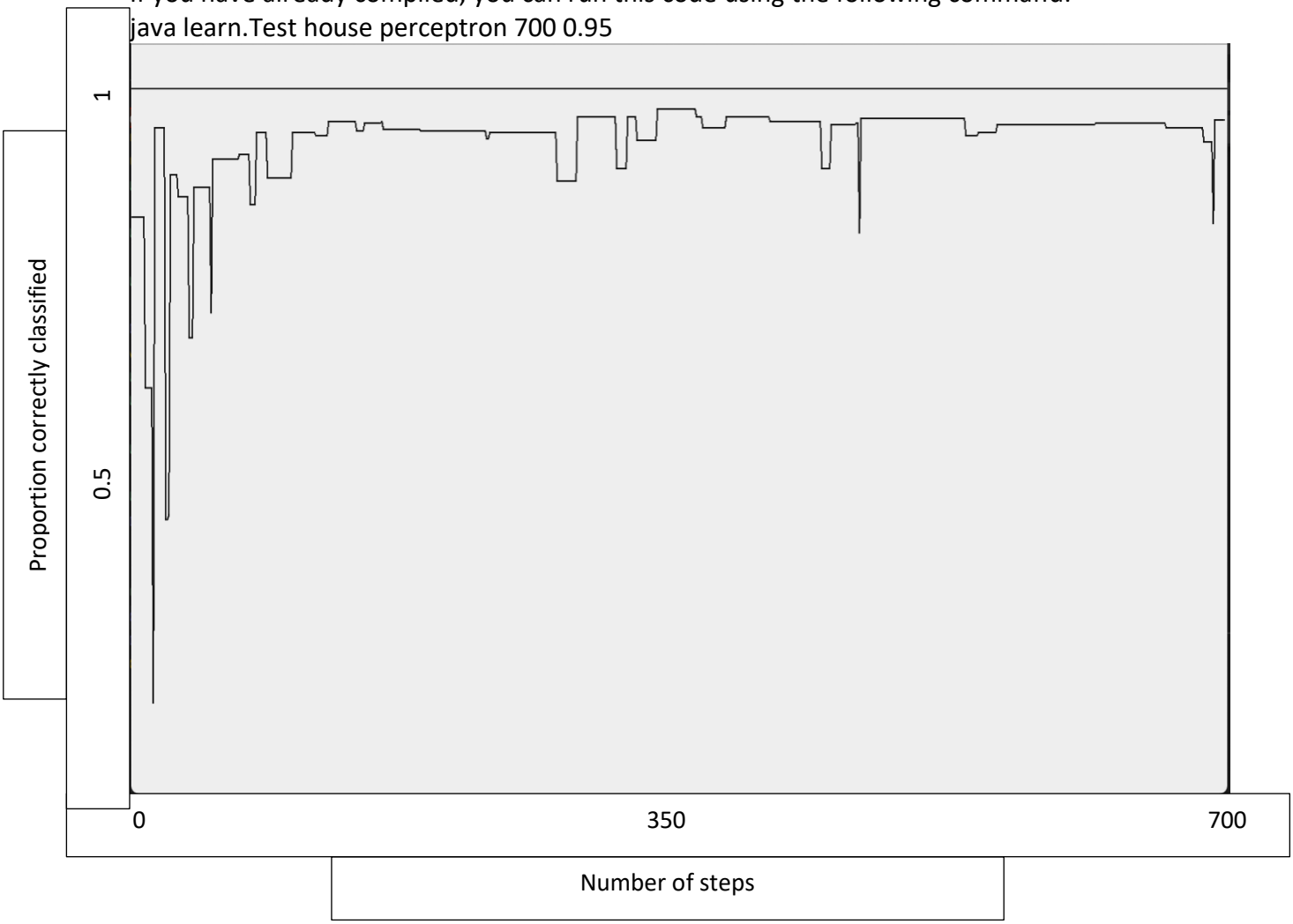
Logistic linear classifier run on the earthquake-noisy data for 100,000 steps with alpha = 0.05. The y-axis represents squared error per example. It starts from the bottom of the picture and goes to the drawn horizontal line. Its range is from 0 to 1. The x-axis is number of steps. It ranges from 0 to 100,000. As can be seen, the graph looks like a logistic curve, as expected, starting with steep improvement which gradually levels out. If you have already compiled, you can run this code using the following command:
java learn.Test earth-noisy logistic 100000 0.05



Logistic linear classifier run on the earthquake-noisy data for 100,000 steps with decaying alpha. The y-axis represents squared error per example. It starts from the bottom of the picture and goes to the drawn horizontal line. Its range is from 0 to 1. The x-axis is number of steps. It ranges from 0 to 100,000. As can be seen, the graph looks like a logistic curve, as expected, starting with steep improvement which gradually levels out. Note the decrease in fluctuation that results from having a decaying rather than fixed learning rate. If you have already compiled, you can run this code using the following command:
java learn.Test earth-noisy logistic 100000 0

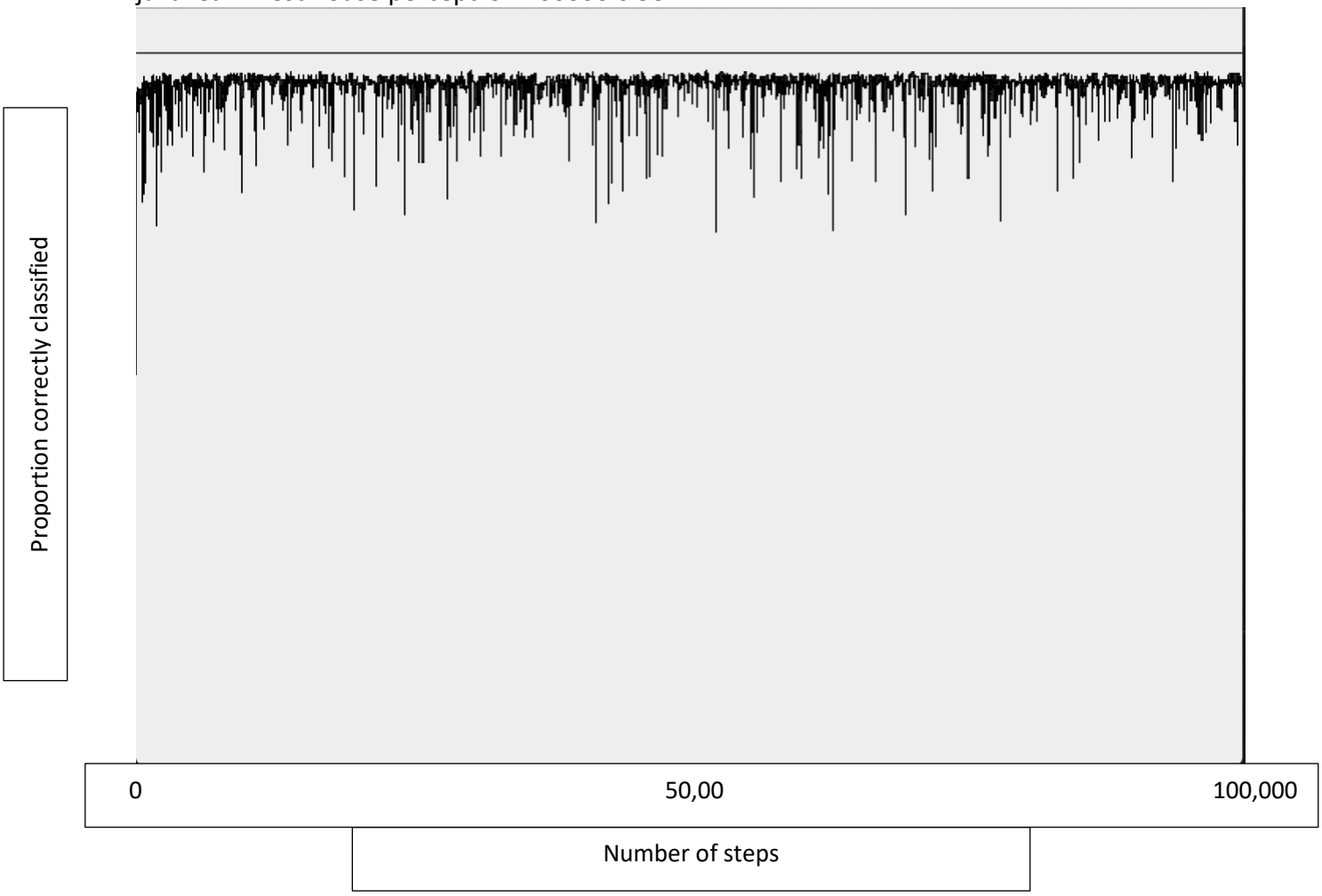


Perceptron run on the house-votes-84 data for 700 steps with alpha=0.95. The y-axis represents proportion correctly classified. It starts from the bottom of the picture and goes to the drawn horizontal line. Its range is from 0 to 1. The x-axis is number of steps. It ranges from 0 to 700. As can be seen, the graph starts with a huge range of values, then gradually improves. If you have already compiled, you can run this code using the following command:
java learn.Test house perceptron 700 0.95



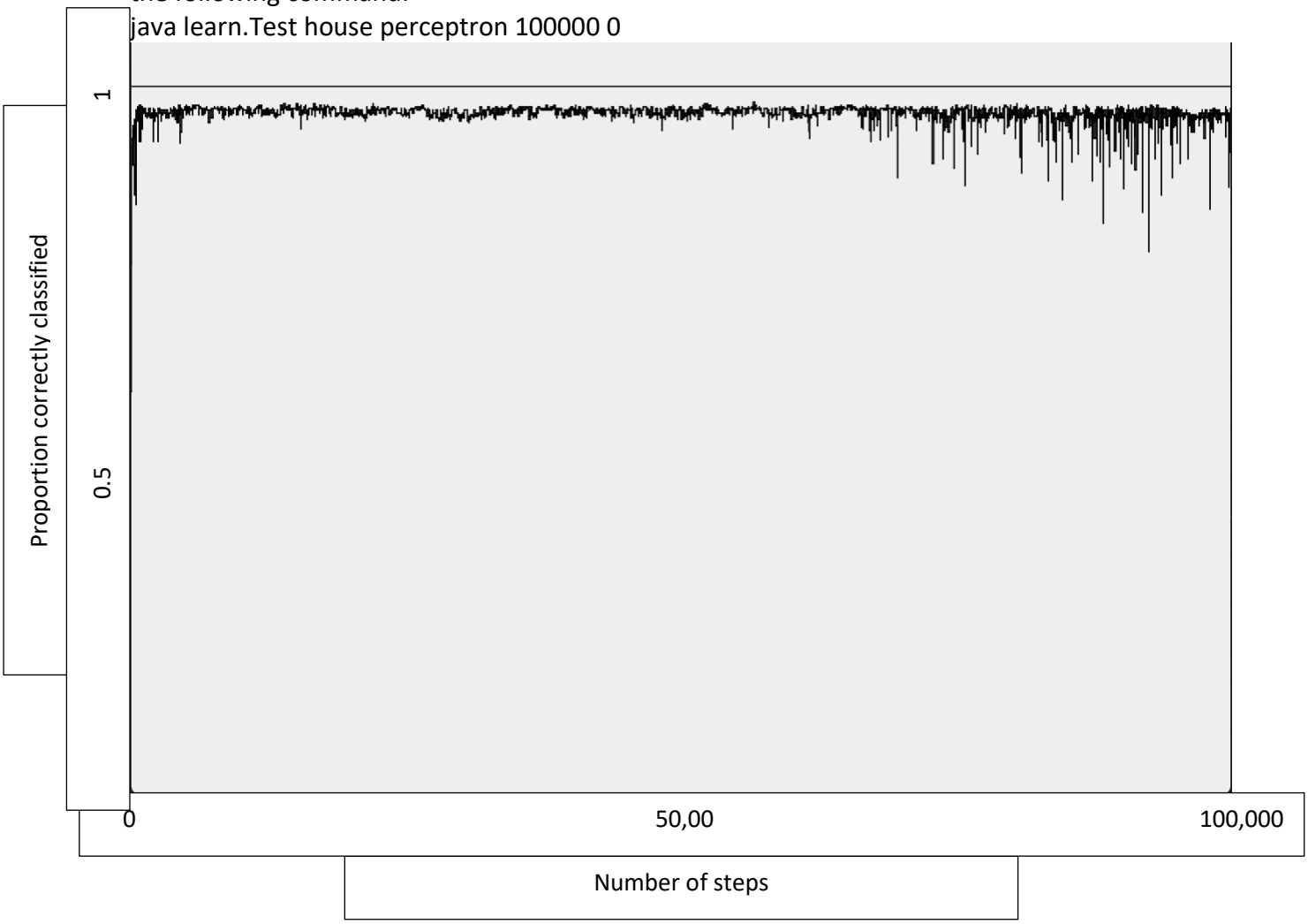
Perceptron run on the house-votes-84 data for 100,000 steps with alpha=0.95. The y-axis represents proportion correctly classified. It starts from the bottom of the picture and goes to the drawn horizontal line. Its range is from 0 to 1. The x-axis is number of steps. It ranges from 0 to 100,000. As can be seen, the graph does improve from the start, though not significantly due to the noise in the data. If you have already compiled, you can run this code using the following command:

java learn.Test house perceptron 100000 0.95

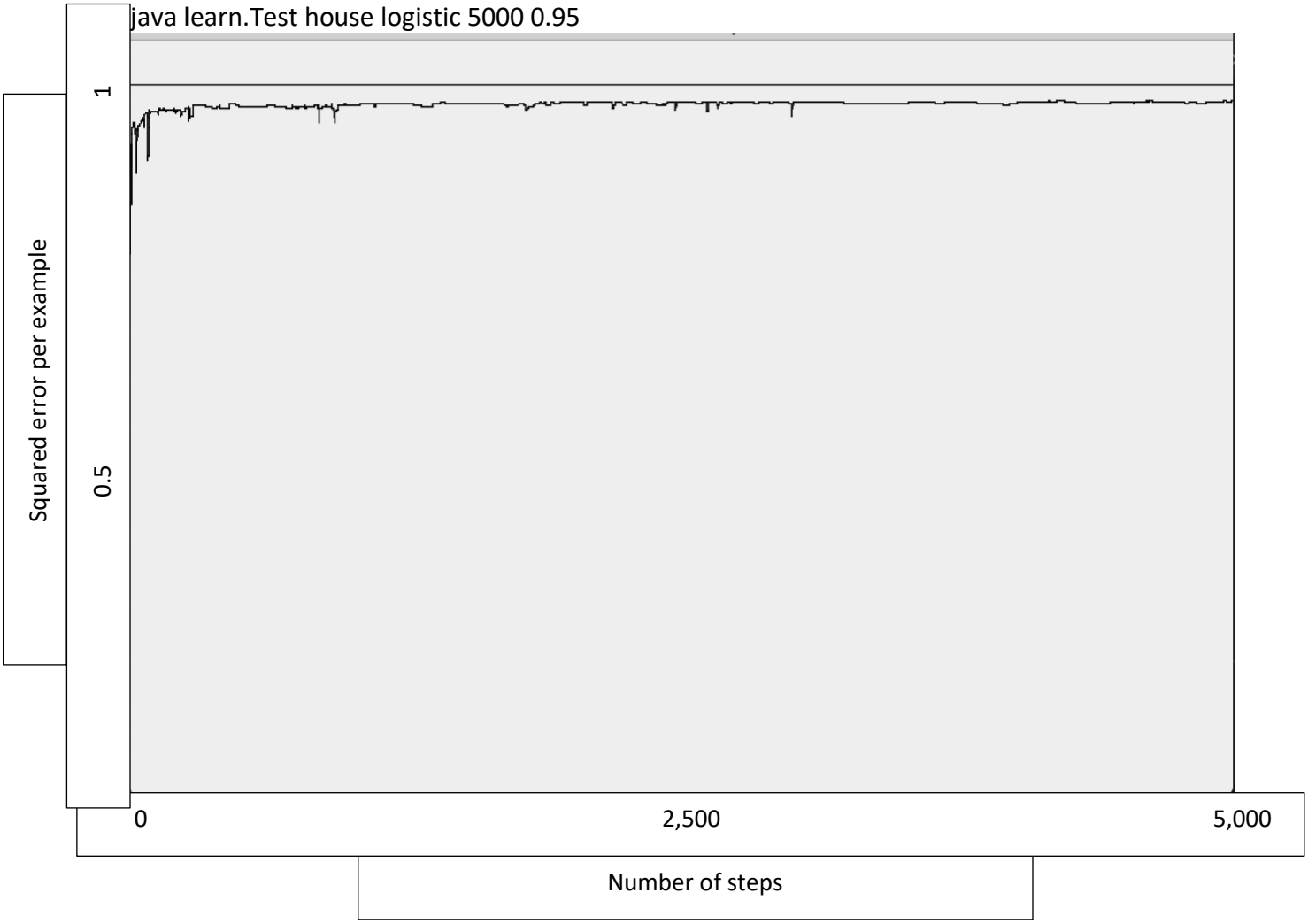


Perceptron run on the house-votes-84 data for 100,000 steps with decaying alpha. The y-axis represents proportion correctly classified. It starts from the bottom of the picture and goes to the drawn horizontal line. Its range is from 0 to 1. The x-axis is number of steps. It ranges from 0 to 100,000. As can be seen, the graph does improve from the start, levels out, and then actually fluctuates more at the end. If you have already compiled, you can run this code using the following command:

java learn.Test house perceptron 100000 0



Logistic Linear Classifier run on the house-votes-84 data for 5000 steps with alpha=0.95. The y-axis represents squared error per example. It starts from the bottom of the picture and goes to the drawn horizontal line. Its range is from 0 to 1. The x-axis is number of steps. It ranges from 0 to 5,000. As can be seen, the graph improves rapidly from the start then levels out. If you have already compiled, you can run this code using the following command:
java learn.Test house logistic 5000 0.95



Logistic Linear Classifier run on the house-votes-84 data for 100,000 steps with alpha=0.95.
The y-axis represents squared error per example. It starts from the bottom of the picture and goes to the drawn horizontal line. Its range is from 0 to 1. The x-axis is number of steps. It ranges from 0 to 100,000. As can be seen, the graph improves rapidly from the start then levels out. It is practically a straight line the whole time. If you have already compiled, you can run this code using the following command:
java learn.Test house logistic 100000 0.95

Logistic Linear Classifier run on the house-votes-84 data for 100,000 steps with decaying alpha. The y-axis represents squared error per example. It starts from the bottom of the picture and goes to the drawn horizontal line. Its range is from 0 to 1. The x-axis is number of steps. It ranges from 0 to 100,000. As can be seen, the graph improves rapidly from the start then levels out. It becomes completely straight with little fluctuation due to the decaying learning rate. If you have already compiled, you can run this code using the following command:

java learn.Test house logistic 100000 0