1. **Training Dataset and Validation Dataset**
   1. Partitioning of the dataset into training and validation portions must be done randomly to maintain the probability distribution of the original sample. That is the set of all possible data D is defined by some probability distribution P(y | x). The dataset provided D’ is hopefully selected uniformly at random from D, implying that D’ should have approximately the same P(y | x) as D. When we further partition the dataset into two pieces, we want to ensure, to the best of our ability, that we again maintain the same probability distribution in both of the partitions. The way to accomplish this is to select from the dataset uniformly at random.
2. **Plot (Learning Rate = 1, max iterations only termination criteria)**
   1. Do you see any pattern in the graph? Do you see any trends in each plot? Do you see any relation between the two plots?
      1. Linear:
         1. As expected due to the convergence proof in class, the training error of perceptron on the linear dataset always converges to a single value and then remains constant no matter how many more iterations are performed after the convergence point.
         2. For both the training and validation error plots, the error trends downward.
         3. The training error is seems to be an excellent estimate for the validation error in all cases, trending almost perfectly alongside the training error as the iterations increase.
         4. As expected the validation error is slightly higher than the training error. Also the training error reaches 0 when the learning rate = 1.
      2. Non-Linear:
         1. As expected the perceptron fails to converge on the non-linear dataset.
         2. Interestingly, the performance on the training data remains a great predictor for the validation data. The rises and falls in both plots are directly and nearly perfectly correlated with one another.
         3. It’s an intriguing property that the minimum training error and validation error is found in the first and second iterations respectively.
         4. After those first two iterations the perceptron reaches a state where its error rate just oscillates continuously between some bounds.
            1. This is clear evidence of the phenomena discussed in class. The perceptron is trying to fix its prediction of various points in the dataset, and each time it fixes its classification of some it begins to misclassify for others due to the fact that the data is not linearly separable.
3. **Learning Rate**
   1. How does varying the learning rate affect the graph? Do they have any effect on how fast/slow your perceptron converges?
      1. Linear:
         1. As the learning rate increases, the number of iterations it takes for the perceptron to converge drops rapidly. Consider the following iteration numbers of the minimum training error for the learning rates tested
            1. 0.01: 536
            2. 0.10: 53
            3. 0.20: 26
            4. 0.50: 13
            5. 0.80: 6
            6. 1.00: 5
         2. In all of these cases the perceptron converges to a training error of 0. Interestingly, the highest 3 learning rates of 0.5, 0.8, and 1.0 actually achieve a slightly better validation error (0.0103) than the lower learning rates (0.0206). This seems counterintuitive given that the slower learning rates should allow a slower and more accurate search for the line that linearly separates the data and minimizes error. This difference is equivalent to misclassifying 5 more examples (5/488 = 0.0103). Likely this is just a side effect of the starting weights used that the training error reaches 0 training error before its able to move the line just a bit further to correctly classify those 5 validation examples.
      2. Non-Linear:
         1. For the nonlinear dataset we see much the same effect as in the linear dataset, except instead of delaying the time it takes to converge, the smaller learning rates delay the time it takes to reach the point where the error rates begin to oscillate continuously. As noted above, with a learning rate of 1, the perceptron begins to oscillate after just 2 iterations. When the learning rate is 0.01, this point is delayed all the way to approximately the 825th iteration. And at the learning rates in between it takes less and less iterations for the error to begin to oscillate. This makes sense since that initial period is the perceptron moving from the initial weights closer and closer to the best possible weights. Once it reaches the best possible weight vector it can no longer make steps in the right direction, any time it tries to correct its classification of one data point, it begins to misclassify another.
         2. In addition to delaying the point of error oscillation, the lower learning rates actually achieve a lower minimum error rate compared to the higher learning rates. This is what we would expect, by forcing the perceptron to take smaller steps when updating the weight vector, it is able to fit the line more effectively to data.
4. **Termination Condition**
   1. Alternate Termination Condition: Terminate early if the validation error does not improve for 50 iterations and the error rate is below 0.2. Keep track of the weight vector that produced the best validation error seen, upon termination set the weight vector back to this optimal value. (So the last iteration and final perceptron will always achieve the minimum validation error seen)
   2. Does your training stop eventually?
      1. Linear:
         1. Yes, for all the learning rates tested the perceptron algorithm terminates before reaching the maximum of 1000 iterations.
      2. Non-Linear:
   3. Does your training stop prematurely?
      1. Linear:
         1. For the learning rate of 0.01, the perceptron terminates prematurely with a training error rate of 0.0026. For all other learning rates the perceptron terminates exactly 50 iterations after reaching the minimum validation error which is always after the perceptron reaches 0% training error.
      2. Non-Linear:
   4. Does your training stop when the training error is at the minimum?
      1. Linear:
         1. Yes for all learning rates except for 0.01.
      2. Non-Linear:
   5. Does your training stop when the validation error is at the minimum?
      1. Yes, the termination condition ensures this.
   6. Do you have any other observations?
      1. Linear:
         1. A nice improvement to this termination criterion would be to simply cut things off once 0% training error is achieved.
         2. Also it would make sense to scale the maximum number of iterations without improvement inversely with the learning rate. For smaller learning rates you will inherently require more iterations to make improvements, thus the maximum iterations without improvement should be increased to compensate for this.

**Linear Dataset: Terminate only due to reaching 1000 total iterations, for 8 different Learning Rates**



