## 3.1

**First question**. For RBFs I divide them uniformely around the input space (means pi/n and std deviation (mean\_n-mean\_n-1)/2

To reduce the error, we need:

|  |  |  |
| --- | --- | --- |
|  | Sin(2x) | Sq(2x) |
| 0.1 | 38-40 RBF | 38-40 RBF |
| 0.01 | 48-50 RBF | 60-63 RBF |
| 0.001 | 60-63 RBF | NOP (only 63) RBF |

¿Why validation error increases after one point (around RBF = nsamples/3)? Maybe because we don’t very many data compared to the number of RBFs

**Second question**. It can be solved like a classification problem, where all positive mapped to 1 and all negative mapped to 0 (threshold = 0). It can be solved with a single layer perceptron. We can do it with **at least 8 nodes**, since with less nodes some values in which square(2x) = 1 are negative with the approximation (SOMETIMES, DEPENDING OF THE NUMBER OF RBFS. **However, it works well (gets 0 errors) only with multiples of 4 (I AM TAKING INTO ACCOUNT TRAINING + VALIDATION DATA).** This transformation could be useful, for example, when you need to classify data, but input samples are noisy.