Introd. to Neural Networks Dr. S. Recksiegel WS 2022/23

Exercise Sheet 5 L. Rickler, E. Eberhard, J. Mayer-Steudte, L. Wicke

November 16, 2022 Moodle: https://www.moodle.tum.de/course/view.php?id=79979

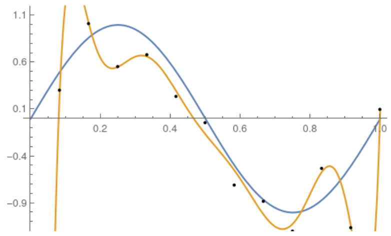
Fitting to an unknown function

1. **Generating data to fit to**
   To keep things simple, our "true" function will be $f(x) = \sin 2\pi x$ in the interval $[0, 1]$. Generate either a couple of random or uniformly spaced x values (e.g. 12) and calculate $f(x)$, but add an artificial error to each result. You can play with the type and size of the errors, we found that a Gaussian distribution with $\sigma = 0.2$ works fine (but you can also just use uniformly distributed random errors).

2. **Checking the generated data**
   Plot the function and the data that you generated. The data points should be distributed around the curve, but they should not be right on the curve.

3. **Fitting the data**
   Fit your data to polynomials of increasing order, i.e. first a_0, then $a_0 + x\,a_1$, etc. up to 9th order. You can use the χ^2 program that you (hopefully) wrote for the last exercise sheet. Plot your fits together with the "true" $f(x)$ and the data points.

4. **Analysing the fits**
   The value of χ^2 is considered a measure for the goodness of the fit. Study the χ^2 for each of the polynomials. Probably the highest polynomial gave you the smallest χ^2. Looking at the plots, would you agree that the highest polynomial gave you the "best" fit? (You should see some wild, unphysical oscillations in the higher order polynomials. If your higher order polynomials look smooth, you were probably lucky/unlucky with your random errors – generate some new data and try again.)



5. **Strategies against over-fitting**
   - One strategy is to keep a few (in our case 2-3) data points back, i.e. not using them for the initial fit but still including them in the final χ^2. If we found a decent description of the physical reality, our fitted curve should also describe the additional points reasonably well. If we over-fitted, on the other hand, we will see that adding these points makes χ^2 *much* larger.
   - Another strategy is to penalise the appearance of large coefficients because we know that those tend not to be natural. Instead of minimising our usual (but this time with $\sigma_i = 1$)

     $$\chi^2 = \sum_{i=1}^{n} (y_i - y(x_i, \vec{a}))^2 \quad \text{where} \quad y_m(x, \vec{a}) = \sum_{j=0}^{m} a_j\, x^j \,,$$

     let us now use

     $$\chi^2_{\text{ren}} = \sum_{i=1}^{n} (y_i - y(x_i, \vec{a}))^2 + \lambda\, |\vec{a}|^2 \;.$$

This way, we "encourage" our fit to avoid large coefficients, and you should observe that for suitable values of λ (try $\sim 10^{-6}$), the over-fitting does not occur anymore. Instead, the polynomials of orders $\sim 3-9$ all give almost identical results.

What happens (and why?) if λ is significantly larger or smaller than the suggested value?