Polynomial Curve Fitting

We start by solving a simple problem in ML. Suppose we have a *training set* of N values, that is, a vector $\boldsymbol{x} = \{x_1, ..., x_N\}$ corresponding to $\boldsymbol{t} = \{t_1, ..., t_N\}$. Now given a new value x, we want to predict its corresponding value t.

Given a set of coefficients $\boldsymbol{w} = \{w_0, ..., w_M\}$ for some $M \in \mathbb{N}$, we have a polynomial of degree M expressed as $y(x, \boldsymbol{w}) = \sum_{i=0}^{M} w_i x^i$. Now we want to approximate the relation between $\boldsymbol{x}$ and $\boldsymbol{t}$ using a polynomial. We want to minimise an *error function* that tells us how good the approximation is given coefficients $\boldsymbol{w}$. The error function evaluates to 0 if and only if it passes through every $t \in \boldsymbol{t}$. Below is a simple way to do it.

Definition. The *sum of squares* error function E is given by

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, \boldsymbol{w}) - t_n)^2.$$

Deriving this gives us a linear map with a trivial kernel, so there is a unique $\boldsymbol{w}^*$ such that $E(\boldsymbol{w}^*)$ is minimal. We still need to choose M, if it's too large then we have the problem of over-fitting, if too little, we don't have enough flexibility to fit accurately to the training set.

We could rigorously test whether over-fitting is a problem by using our function $y(x, \boldsymbol{w}^*)$ against a set with way more datapoints. Hence, we require the following definition that generalises N.

Definition. The *root-means-square* error function is defined as follows.

$$E_{\text{RMS}}(\boldsymbol{w}) = \sqrt{\frac{1}{N} E(\boldsymbol{w})}$$