Machine Learning Module

Week 2

Lecture Notes 3 & 4

Generalisation & Overfitting

Mark Girolami
girolami@dcs.gla.ac.uk
Department of Computing Science
University of Glasgow

January 5, 2006

1 Generalisation and Overfitting

In the laboratory session last week we saw how the Mean Squared Error (MSE) computed on the training set deviated from the MSE achieved on the test set for various model orders. The trend was that as the model complexity increased the training error decreased. There was however no such simple relation with the test error and it was observed that there was a model order for which the test error was a minimum with more or less complex models achieving higher test errors. In other words if we employ too simple a model then poor predictions will be made but if we use too complex a model the quality of our predictions will also be adversely affected. This week we will look at the underlying mechanisms which cause this phenomenon and we will be introduced to methods which allow us to estimate what our predictive performance or test error will be when no independent test data is available.

So far no assumptions regarding the manner in which the data is naturally distributed have been made, in this section however, we will begin to motivate such assumptions so that we can assess the possible performance that we can expect of our models.

As we have seen what is of importance in our models is how well they will predict the outcomes of new events, in our previous example the winning distance in the Olympic Long Jump. We do not stand to make much money by developing a model that can tell us with great accuracy what the winning distances were over the last 100 years, what will be of real value is to have a model that will provide *good* predictions of new and unseen events. In this sense we require a model that can *generalise* its performance beyond the available examples used for *training*. Consider again our averaged Loss-Function defined as

$$\frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(t_n, f(x_n; \mathbf{w})) \tag{1}$$

Now for any data modeling problem each *input-output* pair (x_n, t_n) can be assumed to follow a natural distribution which makes it more likely to observe certain *input-output* pairs than others in which case we can say that there is a *Probability Distribution* $p(x, t)$ which characterizes how likely it is to see (measure) any particular pair (x, t). Without exception in real data modeling this distribution will be unknown, if it were, then we could make optimal predictions based on the *true* distribution of what will ever be

observed. Life is not so simple.

Ideally what we would like to be able to do would be to minimise the average loss over all the possible *input-output* pairs that could possibly be observed. In other words we want to minimise the **Expected Loss**[1] under the distribution $p(x,t)$ such that

$$E\{\mathcal{L}\} = \int\int \mathcal{L}(t, f(x;\mathbf{w}))p(x,t)dxdt$$

As we have N examples drawn from $p(x,t)$ we can estimate the expected loss with the sample average Equation (1) which is often referred to as the *Empirical Loss*. So as the empirical loss is a sample estimate of the expected loss it is quite appropriate to seek to minimise this estimate as a surrogate for the expected loss.

2 Bias-Variance Decomposition

The expected squared error loss can be rewritten so that we can gain insight regarding the source of our modeling errors and as such this section will be devoted to what is referred to as the *Bias-Variance* decomposition in the machine learning literature.

We assume that the *true* model for our data is linear i.e. $w_0 + w_1 x$. Let us also assume that we had an infinite amount of data i.e. $N \to \infty$ then the MSE, which is based on a sample of data drawn from $p(x,t)$, will tend to the expected loss as follows. We denote $[1 \;\; x]^\mathsf{T}$ as $\mathbf{x}$ in what follows.

$$\begin{aligned}\lim_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N} |t_n - f(x_n;\mathbf{w})|^2 &= \int\int |t - f(x;\mathbf{w})|^2 p(x,t)dxdt \\ &= \int\int |t - \mathbf{w}^\mathsf{T}\mathbf{x}|^2 p(t|x)p(x)dxdt\end{aligned}$$

[1]The Expectation operator is defined as the population average of a function which for a continuous (real) random variable X which takes on values $x \in \mathbb{R}$ with probability density $p(x)$ is defined as $E\{f(X)\} = \int f(x)p(x)dx$. For example the expected value or population average of X is $E\{X\} = \int xp(x)dx$. If X takes on a number of K discrete values ($X = x_k$) then $E\{X\} = \sum_{k=1}^{K} x_k P(x_k)$

Now if we differentiate the expected loss with respect to the parameters $\mathbf{w} = [w_0 \;\; w_1]^\mathsf{T}$ and solve for $\mathbf{w}$ then we obtain

$$2 \int \int (t\mathbf{x} - \mathbf{x}\mathbf{x}^\mathsf{T}\mathbf{w}) p(t|x) p(x) dx dt = 0$$

Now $\int \int t\mathbf{x} p(t|x) p(x) dx dt$ is simply the expected value of the cross term $t\mathbf{x}$ under the distribution $p(x, t)$ and so gives a description of how the *inputs* x and the *outputs* t are *correlated* with each other, in other words it is a measure of their *cross-covariance* and will be denoted by $E\{TX\}$, where the upper case is used to denote that these are random variables as opposed to the values which they may take on i.e. t & x. Likewise the right hand term is

$$\begin{aligned}
\int \int \mathbf{x}\mathbf{x}^\mathsf{T}\mathbf{w} p(t|x) p(x) dx dt &= \int p(t|x) dt \int \mathbf{x}\mathbf{x}^\mathsf{T}\mathbf{w} p(x) dx \\
&= 1 \times \int \mathbf{x}\mathbf{x}^\mathsf{T} p(x) dx \; \mathbf{w} \\
&= \int \begin{bmatrix} 1 & x \\ x & x^2 \end{bmatrix} p(x) dx \; \mathbf{w} \\
&= \begin{bmatrix} 1 & E\{X\} \\ E\{X\} & E\{X^2\} \end{bmatrix} \mathbf{w} \\
&= E\{XX^\mathsf{T}\} \; \mathbf{w}
\end{aligned}$$

Therefore in the limit of an infinite amount of data the *true* model parameters are obtained from

$$\mathbf{w} = \left(E\{XX^\mathsf{T}\}\right)^{-1} E\{TX\} \tag{2}$$

and comparing with the Least-Squares estimate we can see how $\widehat{\mathbf{w}}$ is an estimate of $\mathbf{w}$ based on the sample of data available. We would then expect to apportion some of the error observed to the sample based approximations to the expectations appearing in the above equation.

Consider the error made at a particular point x_*

$$\int |t - f(x_*; \mathbf{w})|^2 p(t|x_*) dt$$

differentiating with respect to $f(x_*; \mathbf{w})$ and setting to zero we find that

$$f(x_*; \mathbf{w}) \int p(t|x_*) dt = f(x_*; \mathbf{w}) = \int t p(t|x_*) dt = E\{T|x_*\} \tag{3}$$

So the best function estimate, in terms of expected squared error loss at a point x_* is given by the conditional expectation $E\{T|x_*\}$ in other words the expected value of t given that the *input* equals x_*. So this is the best that we can do. Now back to the expected loss, $\int\int |t - f(x;\mathbf{w})|^2 p(t|x)p(x)dxdt$, which can be written as follows

$$\begin{aligned}
&\int\int |t + E\{T|x\} - E\{T|x\} - f(x;\mathbf{w})|^2 p(t|x)p(x)dxdt = \\
&\int\int |t - E\{T|x\}|^2 p(t|x)p(x)dxdt + \\
&\int\int |E\{T|x\} - f(x;\mathbf{w})|^2 p(t|x)p(x)dxdt - \\
&2\int\int |E\{T|x\} - f(x;\mathbf{w})||t - E\{T|x\}|p(t|x)p(x)dxdt
\end{aligned}$$

It is straightforward to see that the third term above equals zero as

$$\begin{aligned}
&2\int\int |E\{T|x\} - f(x;\mathbf{w})||t - E\{T|x\}|p(t|x)p(x)dxdt = \\
&2\int\int |t - E\{T|x\}|p(t|x)dt|E\{T|x\} - f(x;\mathbf{w})|p(x)dx = \\
&2\int |E\{T|x\} - E\{T|x\}||E\{T|x\} - f(x;\mathbf{w})|p(x)dx = 0
\end{aligned}$$

Likewise the first term can be written as

$$\begin{aligned}
&\int\int |t - E\{T|x\}|^2 p(t|x)p(x)dxdt = \\
&\int\int \left(t^2 + E^2\{T|x\} - 2tE\{T|x\}\right) p(t|x)p(x)dxdt = \\
&\int \left(E\{T^2|x\} + E^2\{T|x\} - 2E^2\{T|x\}\right) p(x)dx = \\
&\int \left(E\{T^2|x\} - E^2\{T|x\}\right) p(x)dx
\end{aligned}$$

This expression gives the variance of the output (target) around the conditional mean value (which is the best estimate of the target value) and characterizes the intrinsic noise in the data and therefore the uncertainty in

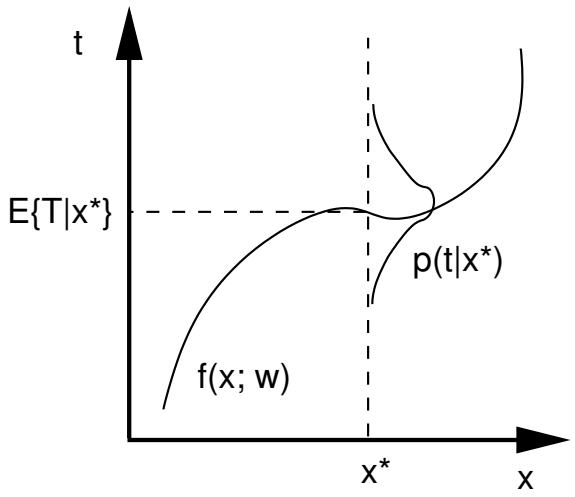

Figure 1: Diagram illustrating the irreducible component of error. The true function to be estimated is $f(x; \mathbf{w})$ and the best estimate in the mean square sense is the conditional mean $E\{T|x^*\}$ however we also see that the conditional distribution $p(t|X^*)$ will have a finite variance $E\{T^2|x^*\} - E^2\{T|x^*\}$ which contributes to the overall error.

the target value estimates, which like taxes and death cannot be avoided. A little diagram is given below to illustrate this component of the overall error.

The second term, $\int\int |E\{T|x\} - f(x; \mathbf{w})|^2 p(t|x)p(x)dxdt$, now needs further consideration. This can be considered as an *approximation* error as it measures the mismatch between our model parameters identified with an infinite amount of data and the parameters estimated from a finite sample.

Now the parameters of the model $f(x; \mathbf{w})$ are estimated from a particular data set $\mathcal{D} = (x_n, t_n)_{n=1,\cdots,N}$ and if we repeated our experiment or measurements[2] and obtained another data set $\mathcal{D}'$ then our function estimate would differ somewhat from that obtained from data set $\mathcal{D}$. If there were a sampling distribution for our data sets $P(\mathcal{D})$ then the expected value of our estimated function would be the model of choice i.e. $\int f(x; \mathbf{w})P(\mathcal{D})d\mathcal{D} = E_{P(\mathcal{D})}\{f(x; \mathbf{w})\}$.

[2]It of course may not be possible to repeat an experiment or make a repeat observation, take the Olympic Games Long Jump example, we won't be seeing a re-run of any of the events held over the last hundred years. In cases like this it seems nonsensical to talk about collections of repeated observations, however the Bayesian approach to such problems has no problems in this respect and we shall be looking at this methodology in later lectures.

Let's just recap here and note that each $f(x;\mathbf{w})$ is estimated from a data set $\mathcal{D}$ via the least squares estimator. Therefore averaging our models over multiple data sets ensures that we have, on average over data sets, a mean-square optimal model. So back to the second term in our error criterion, we can employ the same trick as previous and so

$$\int\int |E\{T|x\} - f(x;\mathbf{w})|^2 p(t|x)p(x)dxdt =$$
$$\int\int |E\{T|x\} - E_{P(\mathcal{D})}\{f(x;\mathbf{w})\} + E_{P(\mathcal{D})}\{f(x;\mathbf{w})\} - f(x;\mathbf{w})|^2 p(t|x)p(x)dxdt =$$
$$\int\int |E\{T|x\} - E_{P(\mathcal{D})}\{f(x;\mathbf{w})\}|^2 p(t|x)p(x)dxdt +$$
$$\int\int |E_{P(\mathcal{D})}\{f(x;\mathbf{w})\} - f(x;\mathbf{w})|^2 p(t|x)p(x)dxdt -$$
$$2\int\int |E\{T|x\} - E_{P(\mathcal{D})}\{f(x;\mathbf{w})\}||E_{P(\mathcal{D})}\{f(x;\mathbf{w})\} - f(x;\mathbf{w})|p(t|x)p(x)dxdt$$

Now we average this over all possible data sets and we find that, as before the third term is zero and all that remains is

$$\int |E\{T|x\} - E_{P(\mathcal{D})}\{f(x;\mathbf{w})\}|^2 p(x)dx +$$
$$\int E_{P(\mathcal{D})}\left\{|E_{P(\mathcal{D})}\{f(x;\mathbf{w})\} - f(x;\mathbf{w})|^2\right\} p(x)dx$$

Note that the expectation does not appear in the first term as it is independent of data set and as both terms are independent of the target values then $\int p(t|x)dt = 1$ thus the integral with respect to t drops out. At long and weary last we can now look at the overall expression for the expected loss and here we also take expectations with respect to the data sets.

$$\int\int E_{P(\mathcal{D})}\{|t - f(x;\mathbf{w})|^2\} p(t|x)p(x)dxdt =$$
$$\int \left(E\{T^2|x\} - E^2\{T|x\}\right) p(x)dx + \tag{4}$$
$$\int |E\{T|x\} - E_{P(\mathcal{D})}\{f(x;\mathbf{w})\}|^2 p(x)dx + \tag{5}$$
$$\int E_{P(\mathcal{D})}\left\{|E_{P(\mathcal{D})}\{f(x;\mathbf{w})\} - f(x;\mathbf{w})|^2\right\} p(x)dx \tag{6}$$

As discussed previously the first term above defines the irreducible error caused by noise in the observations. The second expression is referred to as the square of the bias. This gives a measure of structural miss-match between the model and the underlying data generating process or function. Adopting a functional class for our model which is too simple, in other words it is insufficiently flexible and expressive, will mean that the averaged estimate $E_{P(\mathcal{D})}\{f(x;\mathbf{w})\}$ will be biased away from the conditional-mean $E\{T|x\}$. Now the bias can be reduced by employing appropriately expressive functional classes so we have a lever which we can adjust to reduce our overall loss. However the third term is referred to as the variance and this gives a measure of how much our predictions between data sets will vary. This is something which we must control carefully as highly variable predictions are unreliable. Whilst a more complex model will reduce the bias there may be a corresponding increase in the variance and it is this trade-off between the two competing criteria that is the focus of much attention in devising predictive models for real applications.

2.1 Illustrative Toy Example

We now try out the theory presented on an example. Assume that we are able to make noisy measurements of a function which is defined as a simple polynomial of order three such that $f(x;\mathbf{w}) = 5x^3 - x^2 + x$ and so $\mathbf{w} = [0 \;\; 1 \;\; -1 \;\; 5]^\mathsf{T}$. Now x is randomly sampled with uniform probability over the limited range from -5 to +5, we denote this as $x \sim \mathcal{U}(-5,+5)$ and reads as 'x is uniformly distributed in the range -5 to +5', so $p(x) = \mathcal{U}(-5,+5)$. Now to inject some realism into our playtime our measurements actually have noise on them such that our target values t are actually a noisy realisation of the function of interest i.e. $t(x) = f(x;\mathbf{w}) + \epsilon$. The noise ϵ will be distributed with a Gaussian distribution[3] such that $\epsilon \sim \mathcal{N}(0, 50)$. We will employ a linear model of the form $\sum_{k=0}^{K} w_k x^k$ and varying values of K will be considered.

We will estimate the bias^2 and variance for this model by using sample averages for the required expectations. Let's start with the expression for $\mathsf{bias}^2 = \int |E\{T|x\} - E_{P(\mathcal{D})}\{f(x;\mathbf{w})\}|^2 p(x)dx$, now we draw 1000 *test* samples from $p(x)$ and use these samples to compute the sample average so that $\frac{1}{1000}\sum_{n=1}^{N} |E\{T|x_n\} - E_{P(\mathcal{D})}\{f(x_n;\mathbf{w})\}|^2 \approx \mathsf{bias}^2$. We require $E\{T|x_n\}$ for

[3]The Gaussian probability density for a single variable is defined as $p(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2\sigma^2}|x-\mu|^2\right)$ and is denoted in shorthand as $\mathcal{N}_x(\mu,\sigma)$.

each x_n, and being in the fortunate, and highly unlikely, position of knowing the actual *true* function then we can compute $E\{T|x_n\} = 5x_n^3 - x_n^2 + x_n$ so we are making progress. The final element we need for our estimate of bias^2 is $E_{P(\mathcal{D})}\{f(x_n; \mathbf{w})$ and this can be easily obtained by drawing 100 data samples $(\mathcal{D}_m)_{m=1,\cdots,100}$, where each sample $\mathcal{D}_m$ comprises of 50 *training sample pairs* $(x_l, t_l)_{l=1,\cdots,50}$, and these are used to obtain the corresponding least-squares estimate $\widehat{\mathbf{w}}(m)$ for each sampled data set. With these we can then obtain the estimate $E_{P(\mathcal{D})}\{f(x_n; \mathbf{w}) \approx \frac{1}{100}\sum_{m=1}^{100} f(x_n; \mathbf{w}(m)$. These can then be plugged together to obtain our estimate of the bias^2.

The same procedure can be followed to obtain the estimate of $\mathsf{variance}$ and follows as below.

$$\int E_{P(\mathcal{D})}\left\{|E_{P(\mathcal{D})}\{f(x; \mathbf{w})\} - f(x; \mathbf{w})|^2\right\} p(x)dx \approx$$
$$\frac{1}{1000}\sum_{n=1}^{1000}\frac{1}{100}\sum_{m=1}^{100}\left(\frac{1}{100}\sum_{l=1}^{100} f(x_n; \mathbf{w}(l)) - f(x_n; \mathbf{w}(m))\right)^2$$

A simple implementation of the above procedure in Matlab yields the bias^2 & $\mathsf{variance}$ estimates for polynomial models of order $K = 1$ to $K = 8$ (Figure 2). The theory predicts that as the model becomes more flexible the bias^2 will decrease however there is a corresponding increase in the $\mathsf{variance}$ of the models, giving an estimate of the overall expected loss which finds a minimum at a model complexity of $K = 3$, which happily is the correct order of the actual function being estimated.

Before we sit back feeling that this little piece of analysis provides us with a means of assessing the predictive performance of models of varying orders of representational complexity, lets have a brief sanity check. The bias-variance decomposition is an important theoretical analytic tool as it demonstrates that despite more complex models being able to better describe the available data the variation, in terms of generalisation capability, (how will predictions made on new and previously unseen data be adversely affected by this flexibility?) will increase. Given that in real modeling situations the chances that the *true* model lies in the functional class selected (we have been using the additive polynomial class so far) are almost negligble it is then clear that driving the model bias as low as possible is clearly an unwise strategy to follow. The Least-Squares estimator happens to be an unbiased estimator and

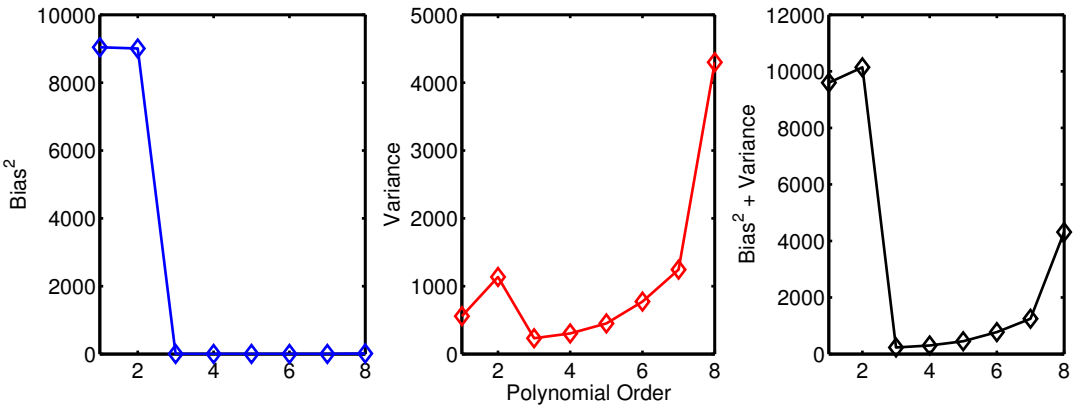

Figure 2: The leftmost plot shows the estimated bias^2 for a polynomial model indexed on the horizontal axis by polynomial order, the middle plot shows the corresponding estimated $\mathsf{variance}$, whilst the rightmost plot gives the cumulative effect of both bias^2 + $\mathsf{variance}$. The details to note are that as the complexity of the model increase the bias^2 continually decrease providing an increasingly superior fit to the data. On the other hand the $\mathsf{variance}$ increase with model complexity with the net effect being that the minimum of bias^2 + $\mathsf{variance}$ (the expected loss minus the constant term) is achieved at $K = 3$ which is the correct complexity for the function being approximated.

indeed of all unbiased estimators it is the one with the smallest variance[4]. This of course makes the rather huge assumption that the model structure is correct and in most applications within Machine Learning this is very seldom the case[5].

The main problem with the practical use of the bias-variance decomposition is that it cannot really be used in practice as we do not have access to the true function nor do we have any way of sampling from $p(x, t)$ or $P(\mathcal{D})$ as these are also unknown. However, there is a technique which provides an estimate of model performance on unseen data that is straightforward to implement and of great practical importance and usefulness and we present

[4]The famous Gauss-Markov theorem proves that the Least-Squares estimator is BLUE - Best Linear Unbiased Estimator, proofs are fairly straightforward and if you are loosing sleep regarding such a proof come and see me.

[5]George Box the chemist and statistician has said *All Models are Wrong but Some are Useful* which then suggests that choosing unbiased BLUE estimators of model parameters is perhaps not the wisest of choices. Further lectures will explore this point further.

this in the following section.

3 Cross-Validation

We require a measure of the *expected loss* to provide an indication of the generalisation ability of our predictive models and the *empirical loss* based on a finite sample is a useful estimate of *expected loss*. As the previous section has highlighted via the bias-variance decomposition increasing model complexity will reduce model bias reflected in a lower *training error*. However the *training error* is obtained from the same data that is used for parameter estimation and hence will provide an optimistic estimate of the achievable *test error*. A number of methods have been proposed to provide a more pessimistic and hopefully realistic indicator of test error such as the *Akaike Information Criterion* (AIC), *Minimum Description Length* (MDL) and *Vapnik-Chernovenkis Dimension* (VC). We will not focus on these methods as they are of limited practical use, instead we will focus on the simplest and possibly most useful method of estimating test error, that is cross-validation.

Cross-validation directly estimates the generalisation (test) error quite simply by holding out a fraction of the available training data and using this to obtain a prediction error. Computing a **Leave-One-Out** cross-validation (LOOCV) error is quite simply computed as follows.

3.1 Leave-One-Out Cross Validation

Given a data set $\mathcal{D} = (x_1, t_1), \cdots, (x_N, t_N)$ comprising of N *input-target* pairs we remove one input and target pair, say (x_i, t_i) from the available sample so creating the data sample $\mathcal{D}_{-i}$. We now use $\mathcal{D}_{-i}$ to induce our learning machine whatever it happens to be, at the moment the only learning machine we have met is our Least-Squares based Linear Regression, in which case

$$\widehat{\mathbf{w}}_{-i} = \left(\mathbf{X}_{-i}^{\mathsf{T}}\mathbf{X}_{-i}\right)^{-1}\mathbf{X}_{-i}^{\mathsf{T}}\mathbf{t}_{-i}$$

where the $(N-1)\times(K+1)$ matrix with the ith row removed is denoted by $\mathbf{X}_{-i}$, the $(N-1)\times 1$ dimensional column vector with the ith element removed is denoted by $\mathbf{t}_{-i}$ and $\widehat{\mathbf{w}}_{-i}$ denotes our least-squares parameter estimate based on the data sample $\mathcal{D}_{-i}$.

For the held-out *input-target* pair (x_i, t_i) we can compute the corresponding loss $\mathcal{L}(t_i, f(x_i; \widehat{\mathbf{w}}_{-i}))$ which in our case here would be the squared-error loss $|t_i - \widehat{\mathbf{w}}_{-i}^{\mathsf{T}}\mathbf{x}_i|^2$ where $\mathbf{x}_i$ is the ith row of $\mathbf{X}$.

Now we can perform this procedure N times cycling through all the data and leaving each one out in turn and so our LOO estimate of the generalisation error or expected loss will simply be

$$\begin{aligned}\mathcal{L}_{cv} &= \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(t_i, f(x_i; \widehat{\mathbf{w}}_{-i})) \\ &= \frac{1}{N}\sum_{i=1}^{N}|t_i - \widehat{\mathbf{w}}_{-i}^{\mathsf{T}}\mathbf{x}_i|^2\end{aligned}$$

Cross-Validation is entirely general with regard to the loss function for which it can estimate the expectation.

3.1.1 Experimental Demonstration

Fifty input-target pairs from a noisy third-order polynomial function are sampled and these are used to learn a polynomial regression function. A further 1000 input-target pairs are used as an independent test set with which to compute the overall test error. In addition we use the LOOCV estimator as described above to estimate the expected test-error. A range of polynomial orders are considered from order 1 (linear model) up to 10th order (highly flexible model and for each model-order the training error, test error and LOOCV error are computed.

The results are shown in Figure (3) and it can be seen that the training error continues to decrease as the model complexity increases i.e. we are obtaining a better fit to the training data which is a mixture of the underlying functional response and the additive measurement noise. When we look at the test error then it is clear that once the model becomes more flexible there is corresponding decrease in test error followed by a slow increase as we should now be expecting. However if we look at the LOOCV error then we see that it tends to reflect the obtained test error at the various model orders which is exactly what we would hope for. Note that the error-bars are one standard error which is the standard deviation of the LOOCV estimate divided by $\sqrt{N}$.

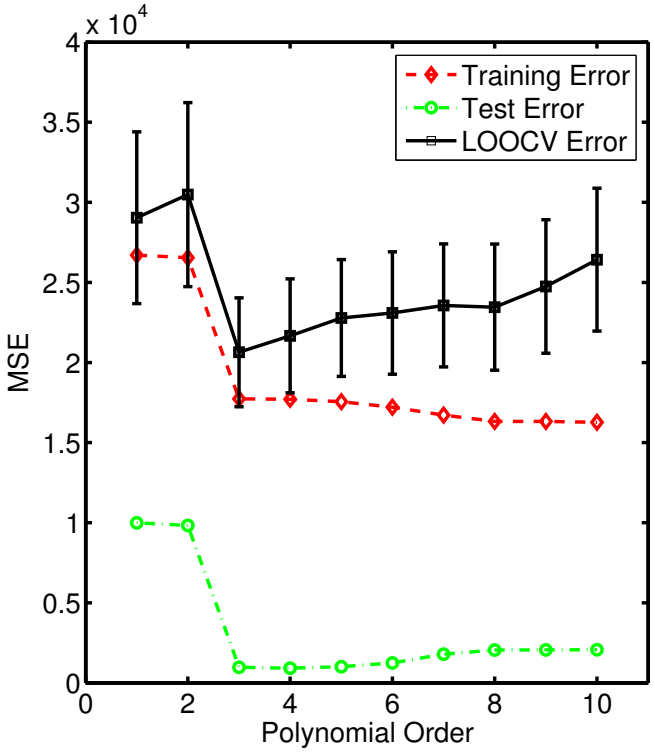

Figure 3: The Training, Testing and Leave-One-Out error curves obtained for a noisy cubic function where a sample size of 50 is available for training and LOOCV estimation. The test error is computed using 1000 independent samples.

So we now have a useful surrogate measure for expected loss in our LOOCV and this is a tool which we shall make regular use of throughout the rest of this course.

3.1.2 Computational Scaling & K-Fold Validation

The good news is that we have a means of estimating our expected test error from the training data alone and this provides a useful way to explore and assess various alternative models for a particular prediction task. However, let us have a look at a simple Matlab function for computing the LOOCV error for a Least-Squares linear regression model.

```
function [cv_err, cv_std] = cross_val(x,f)
N = size(x,1);
CV = [];

for n=1:N
    X = x;
    t = f;
    X(n,:) = [];
    t(n) = [];
    Xt = x(n,:);
    tt = f(n);
    w_hat = inv(X'*X)*X'*t;
    f_t = Xt*w_hat;
    CV  = [CV; (f_t - tt).^2];
end

cv_err = mean(CV);
cv_std = std(CV);
```

The first thing to note is that we are looping N times and within the loop we have to perform our *training* method which in this case is obtaining the Least-Squares solution which requires a matrix inversion that scales as $\mathcal{O}((K+1)^3)$ where $K+1$ is the dimension of the matrix being inverted. Matrix multiplications will contribute $\mathcal{O}(N(K+1)^2 + 2N(K+1)^3)$ scaling so the overall dominant scaling for LOOCV is $\mathcal{O}(N^2(K+1)^3)$. As either K or N become large we can see that LOOCV can become rather expensive computationally. Indeed some of the methods which we will meet later in the course are dominated by $\mathcal{O}(N^3)$ scaling where N is the training set size so LOOCV for these methods will cost a staggering $\mathcal{O}(N^4)$.

To alleviate this problem somewhat rather than using leave-one-out we can leave out 10% of the data as test data and use the remaining 90% for training, this means that only 10-folds of the data and so 10 training repeats are required which will be far less costly than 1% & 99% data splits.