Chap.8 Learning and generalization [Book, Chap. 6]

Avoid using a model with too little flexibility to model the underlying nonlinear relation adequately (*underfitting*), and using a model with too much flexibility, which readily fits to the noise (*overfitting*).

8.1 Mean squared error and maximum likelihood [Book, Sect.6.1]

In multi-layer perceptron (MLP) NN models, minimizing the objective function J involves minimizing the *mean squared error* (MSE) between the model outputs $\mathbf{y}$ and the target data $\mathbf{y}_{\mathrm{d}}$, i.e.

$$J = \frac{1}{N} \sum_{n=1}^{N} \left\{ \frac{1}{2} \sum_{k} [y_k^{(n)} - y_{\mathrm{d}k}^{(n)}]^2 \right\} , \tag{1}$$

where there are $k = 1, \ldots, M$ output variables y_k, and there are $n = 1, \ldots, N$ observations. While minimizing the MSE is quite intuitive and is used in many types of models besides NN (e.g. linear regression), it can be derived from the broader principle of *maximum likelihood* under the assumption of Gaussian noise distribution.

If we assume the multivariate target data $y_{\mathrm{d}k}$ are independent random variables, then the conditional probability distribution of $\mathbf{y}_{\mathrm{d}}$ given predictors $\mathbf{x}$ can be written as

$$p(\mathbf{y}_{\mathrm{d}}|\mathbf{x}) = \prod_{k=1}^{M} p(y_{\mathrm{d}k}|\mathbf{x}). \tag{2}$$

The target data are made up of noise ϵ_k plus an underlying signal (which we are trying to simulate by a model with parameters $\mathbf{w}$ and outputs y_k), i.e.

$$y_{\mathrm{d}k} = y_k(\mathbf{x}; \mathbf{w}) + \epsilon_k. \tag{3}$$

We assume that the noise ϵ_k obeys a Gaussian distribution with zero mean and standard deviation σ, with σ independent of k and $\mathbf{x}$, i.e.

$$p(\boldsymbol{\epsilon}) = \prod_{k=1}^{M} p(\epsilon_k) = \frac{1}{(2\pi)^{M/2}\sigma^M} \exp\left(-\frac{\sum_k \epsilon_k^2}{2\sigma^2}\right). \tag{4}$$

From (3) and (4), the conditional probability distribution

$$p(\mathbf{y}_\mathrm{d}|\mathbf{x};\mathbf{w}) = \frac{1}{(2\pi)^{M/2}\sigma^M} \exp\left[-\frac{\sum_k (y_k(\mathbf{x};\mathbf{w}) - y_{\mathrm{d}k})^2}{2\sigma^2}\right]. \tag{5}$$

The principle of *maximum likelihood* says: If we have a conditional probability distribution $p(\mathbf{y}_\mathrm{d}|\mathbf{x};\mathbf{w})$, and we have observed values $\mathbf{y}_\mathrm{d}$ given by the dataset D and $\mathbf{x}$ by the dataset X, then the parameters $\mathbf{w}$ can be found by maximizing the likelihood function $p(D|X;\mathbf{w})$, i.e. the parameters $\mathbf{w}$ should be chosen so that the likelihood of

observing D given X is maximized. Note that $p(D|X;\mathbf{w})$ is a function of $\mathbf{w}$ only as D and X are known.

The datasets X and D contain the observations $\mathbf{x}^{(n)}$ and $\mathbf{y}_{\mathrm{d}}^{(n)}$, with $n = 1, \ldots, N$. The likelihood function L is then

$$L = p(D|X;\mathbf{w}) = \prod_{n=1}^{N} p\left(\mathbf{y}_{\mathrm{d}}{}^{(n)}|\mathbf{x}^{(n)};\mathbf{w}\right). \tag{6}$$

Instead of maximizing the likelihood function, it is more convenient to minimize the negative log of the likelihood, as the logarithm

function is a monotonic function. From (5) and (6), we end up minimizing the following objective function with respect to $\mathbf{w}$:

$$\tilde{J} = -\ln L = \frac{1}{2\sigma^2} \sum_{n=1}^{N} \sum_{k=1}^{M} \left[y_k(\mathbf{x}^{(n)}; \mathbf{w}) - y_{\mathrm{d}k}^{(n)} \right]^2 + NM \ln \sigma + \frac{NM}{2} \ln(2\pi). \quad (7)$$

Since the last two terms are independent of $\mathbf{w}$, they are irrelevant to the minimization process and can be omitted. Other than a constant multiplicative factor, the remaining term in $\tilde{J}$ is the same as the MSE objective function J in (1). Hence minimizing MSE is equivalent to maximizing likelihood assuming Gaussian noise distribution.

8.2 Objective functions and robustness [Book, Sect. 6.2]

We examine where the model outputs converge to, under the MSE objective function (1) in the limit of infinite sample size N with a flexible enough model (Bishop, 1995).

The MSE is not the only way to incorporate information about the error between the model output y_k and the target data $y_{\mathrm{d}k}$ into J. We could minimize the *mean absolute error* (MAE) instead of the MSE, i.e. define

$$J = \frac{1}{N} \sum_{n=1}^{N} \sum_{k} \left| y_k^{(n)} - y_{\mathrm{d}k}^{(n)} \right| . \tag{8}$$

Any data point $y_{\mathrm{d}k}$ lying far from the mean of the distribution of $y_{\mathrm{d}k}$ would exert far more influence in determining the solution under the MSE objective function than in the MAE objective function. We will

show that unlike the MAE, the MSE objective function is not robust to *outliers* (i.e. data points lying far away from the mean, which might have resulted from defective measurements or from exceptional events).

First study the MSE objective function (1). With infinite N, the sum over the N observations in the objective function can be replaced by integrals, i.e.

$$J = \frac{1}{2} \sum_k \int \int [y_k(\mathbf{x}; \mathbf{w}) - y_{\mathrm{d}k}]^2 \, p(y_{\mathrm{d}k}, \mathbf{x}) \, \mathrm{d}y_{\mathrm{d}k} \, \mathrm{d}\mathbf{x} \,, \tag{9}$$

where $\mathbf{x}$ and $\mathbf{w}$ are the model inputs and model parameters respectively, and $p(y_{\mathrm{d}k}, \mathbf{x})$, a joint probability distribution. Since

$$p(y_{\mathrm{d}k}, \mathbf{x}) = p(y_{\mathrm{d}k}|\mathbf{x}) \, p(\mathbf{x}) \,, \tag{10}$$

where $p(\mathbf{x})$ is the probability density of the input data, and $p(y_{\mathrm{d}k}|\mathbf{x})$ is the probability density of the target data conditional on the inputs, we have

$$J = \frac{1}{2}\sum_k \int \int [y_k(\mathbf{x};\mathbf{w}) - y_{\mathrm{d}k}]^2 \, p(y_{\mathrm{d}k}|\mathbf{x}) \, p(\mathbf{x}) \, \mathrm{d}y_{\mathrm{d}k} \, \mathrm{d}\mathbf{x} \, . \tag{11}$$

Next introduce the following conditional averages of the target data:

$$\langle y_{\mathrm{d}k}|\mathbf{x}\rangle = \int y_{\mathrm{d}k} \; p(y_{\mathrm{d}k}|\mathbf{x}) \, \mathrm{d}y_{\mathrm{d}k} \, , \tag{12}$$

$$\langle y_{\mathrm{d}k}^2|\mathbf{x}\rangle = \int y_{\mathrm{d}k}^2 \; p(y_{\mathrm{d}k}|\mathbf{x}) \, \mathrm{d}y_{\mathrm{d}k} \, . \tag{13}$$

After some algebra (see Book), we can write

$$J = \frac{1}{2}\sum_k \int [y_k(\mathbf{x};\mathbf{w}) - \langle y_{\mathrm{d}k}|\mathbf{x}\rangle]^2 p(\mathbf{x})\mathrm{d}\mathbf{x} + \frac{1}{2}\sum_k \int [\langle y_{\mathrm{d}k}^2|\mathbf{x}\rangle - \langle y_{\mathrm{d}k}|\mathbf{x}\rangle^2]\, p(\mathbf{x})\mathrm{d}\mathbf{x}. \quad (14)$$

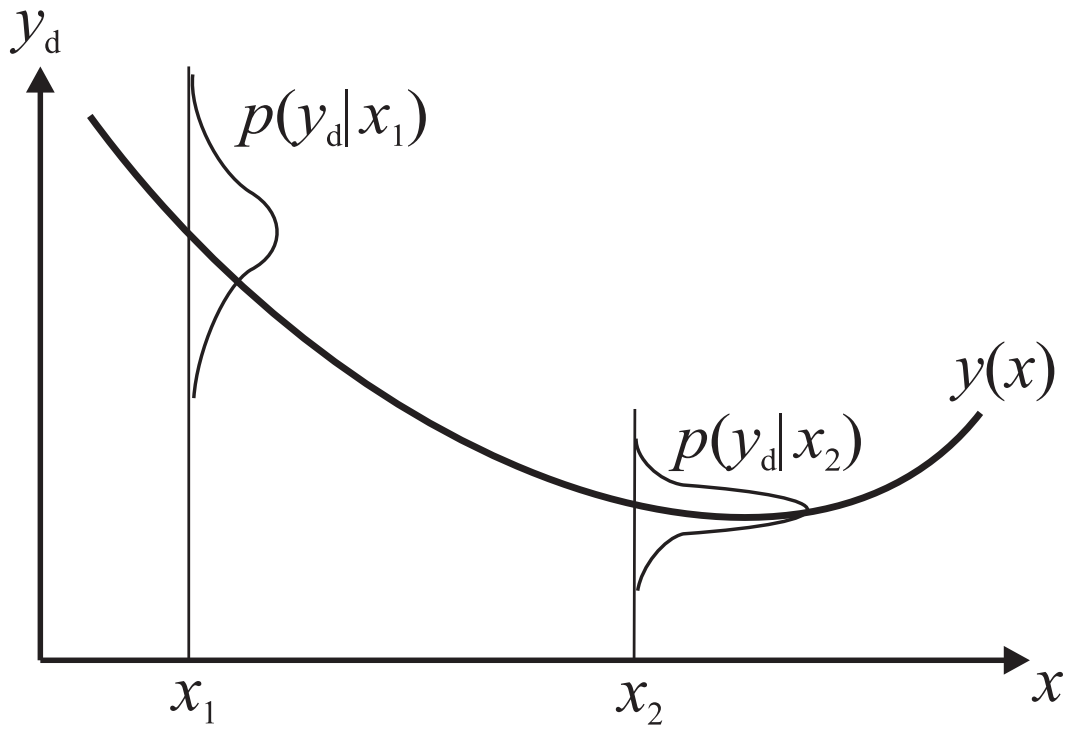
The second term does not depend on the model output y_k, hence it is independent of the model weights $\mathbf{w}$. Thus during the search for the optimal weights to minimize J, the second term can be ignored.

In the first term of (14), the integrand cannot be negative, so the minimum of J occurs when this first term vanishes, i.e.

$$y_k(\mathbf{x};\mathbf{w}_{\mathrm{opt}}) = \langle y_{\mathrm{d}k}|\mathbf{x}\rangle, \quad (15)$$

where $\mathbf{w}_{\mathrm{opt}}$ denotes the weights at the minimum of J. This is a very important result as it shows that the model output is simply the conditional mean of the target data. Thus in the limit of infinite number of observations in the dataset, and with the use of a flexible enough model, the model output y_k for a given input $\mathbf{x}$ is the conditional mean of the target data at $\mathbf{x}$.

Diagram below shows the model output y as the conditional mean of the target data y_{d}, with the conditional probability distribution $p(y_{\mathrm{d}}|x)$ displayed at x_1 and at x_2.



The derivation of this result is quite general, as it does not actually require the model mapping $y_k(\mathbf{x};\mathbf{w})$ to be restricted to NN models. This result also shows that in nonlinear regression problems, in the limit of infinite sample size, overfitting cannot occur, as the model output converges to the conditional mean of the target data. In

practice, in the absence of outliers, overfitting ceases to be a problem when the number of independent observations is much larger than the number of model parameters.

Next, we turn to the MAE objective function (8). Under infinite N, (8) becomes

$$J = \sum_k \int \int |y_k(\mathbf{x}; \mathbf{w}) - y_{\mathrm{d}k}| \; p(y_{\mathrm{d}k}|\mathbf{x})\, p(\mathbf{x})\, \mathrm{d}y_{\mathrm{d}k}\, \mathrm{d}\mathbf{x}\,. \tag{16}$$

This can be rewritten as

$$J = \sum_k \int \tilde{J}_k(\mathbf{x})\; p(\mathbf{x})\, \mathrm{d}\mathbf{x}\,, \tag{17}$$

where

$$\tilde{J}_k(\mathbf{x}) \equiv \int |y_k(\mathbf{x}; \mathbf{w}) - y_{\mathrm{d}k}| \; p(y_{\mathrm{d}k}|\mathbf{x})\, \mathrm{d}y_{\mathrm{d}k}\,. \tag{18}$$

$\tilde{J}_k(\mathbf{x}) \geq 0$ since the integrand of (18) is non-negative. Also J in (17) is minimized when $\tilde{J}_k(\mathbf{x})$ is minimized. To minimize $\tilde{J}_k(\mathbf{x})$ with respect to the model output y_k, we set

$$\frac{\partial \tilde{J}_k}{\partial y_k} = \int \operatorname{sgn}(y_k(\mathbf{x};\mathbf{w}) - y_{\mathrm{d}k})\ p(y_{\mathrm{d}k}|\mathbf{x})\,\mathrm{d}y_{\mathrm{d}k} = 0\,, \qquad (19)$$

where the function $\operatorname{sgn}(z)$ gives $+1$ or -1 depending on the sign of z. For this integral to vanish, the equivalent condition is

$$\int_{-\infty}^{y_k} p(y_{\mathrm{d}k}|\mathbf{x})\,\mathrm{d}y_{\mathrm{d}k} - \int_{y_k}^{\infty} p(y_{\mathrm{d}k}|\mathbf{x})\,\mathrm{d}y_{\mathrm{d}k} = 0\,, \qquad (20)$$

which means that $y_k(\mathbf{x};\mathbf{w})$ has to be the conditional *median*, so that the conditional probability density integrated to the left of y_k equals that integrated to the right of y_k.

The median is robust to outliers whereas the mean is not. Thus in the presence of outliers, the MSE objective function can produce solutions which are strongly influenced by outliers, whereas the MAE objective function can largely eliminate this undesirable property (since in practice an infinite N is not attainable, therefore using MAE does not completely eliminate this problem). However, a disadvantage of the MAE objective function is that it is less sensitive than the MSE objective function, so it may not fit the data as closely.

8.3 Variance and bias errors [Book, Sect. 6.3]

Two types of errors when fitting a model to a dataset — variance error and bias error.

To simplify the discussion, assume the model output is a single variable $y = f(\mathbf{x})$. The true relation is $y_{\mathrm{T}} = f_{\mathrm{T}}(\mathbf{x})$.

The model was trained over a dataset D. Let $\mathcal{E}[\cdot]$ denote the expectation or ensemble average over all datasets D. Note that $\mathcal{E}[\cdot]$ is not the expectation $\mathrm{E}[\cdot]$ over $\mathbf{x}$, so $\mathcal{E}[y] \equiv \bar{y}$ is still a function of $\mathbf{x}$.

Thus the error of y is

$$\begin{aligned}
\mathcal{E}[(y - y_{\mathrm{T}})^2] &= \mathcal{E}[(y - \bar{y} + \bar{y} - y_{\mathrm{T}})^2] \\
&= \mathcal{E}[(y - \bar{y})^2] + \mathcal{E}[(\bar{y} - y_{\mathrm{T}})^2] + 2\mathcal{E}[(y - \bar{y})(\bar{y} - y_{\mathrm{T}})] \\
&= \mathcal{E}[(y - \bar{y})^2] + (\bar{y} - y_{\mathrm{T}})^2 + 2(\bar{y} - y_{\mathrm{T}})\,\mathcal{E}[y - \bar{y}] \\
&= \mathcal{E}[(y - \bar{y})^2] + (\bar{y} - y_{\mathrm{T}})^2, \qquad (21)
\end{aligned}$$

since $\mathcal{E}[y - \bar{y}] = 0$.

The first term, $\mathcal{E}[(y - \bar{y})^2]$, is the *variance error*, as it measures the departure of y from its expectation $\bar{y}$.

The second term, $(\bar{y} - y_{\mathrm{T}})^2$, is the *bias error*, as it measures the departure of $\bar{y}$ from the true value y_{T}.

The variance error tells us how much the y estimated from a given dataset D can be expected to fluctuate about $\bar{y}$, the expectation over all datasets D. Even with this fluctuation caused by the sampling for a particular dataset D removed, one has the bias error indicating the departure of the model expectation from the true value.

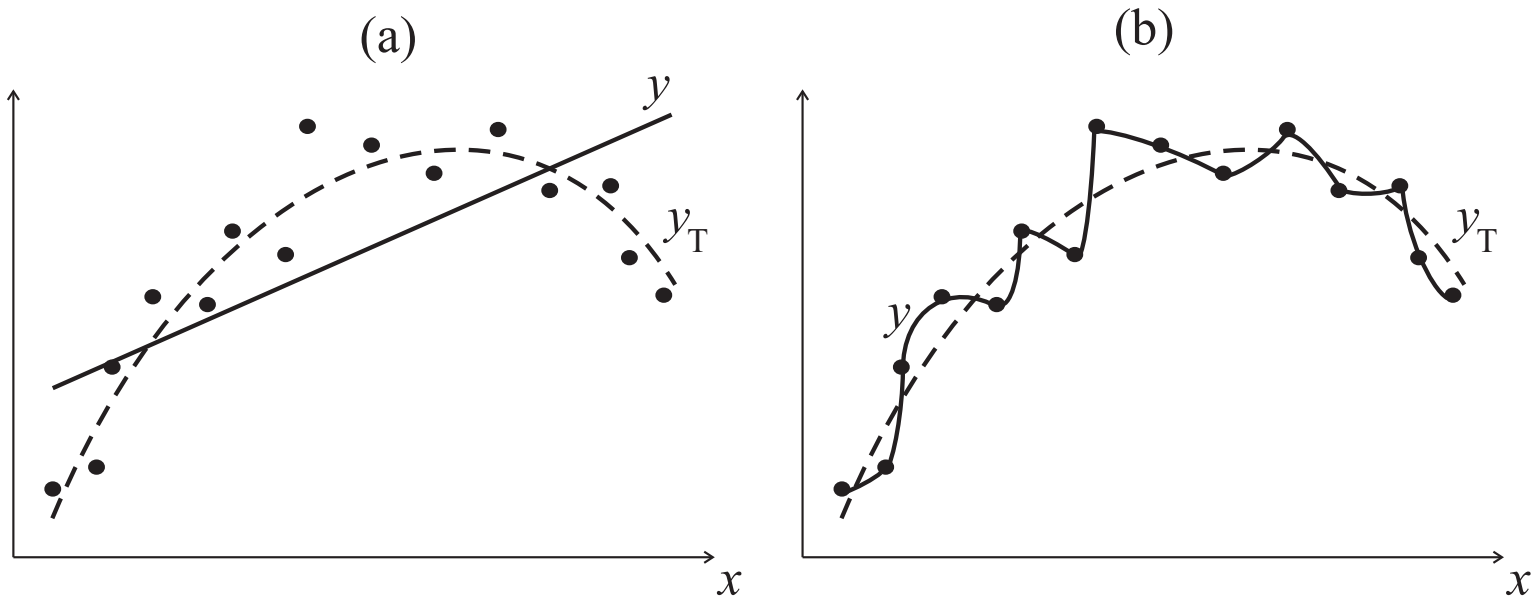


Figure : A schematic diagram illustrating the results from using a model with (a) few adjustable parameters and (b) many adjustable parameters to fit the data. The model fit y is shown by a solid line and the true relation y_T by the dashed line. In (a), the bias error is large as y from a linear model is a poor approximation of y_T, but the variance error is small. In (b), the bias error is small but the variance error is large, since the model is fitting to the noise in the data.

The art of machine learning hinges on a balanced trade-off between variance error and bias error.

8.4 Regularization [Book, Sect. 6.5]

To prevent overfitting in MLP NN models, the most common approach is via *regularization* of the objective function, i.e. by adding *weight penalty* (also known as *weight decay*) terms to the objective function.

The objective function is now

$$J = \frac{1}{N} \sum_{n=1}^{N} \left\{ \frac{1}{2} \sum_{k} \left[y_k^{(n)} - y_{\mathrm{d}k}^{(n)} \right]^2 \right\} + P \frac{1}{2} \sum_{j} w_j^2 \,, \qquad (22)$$

where w_j presents all the weight (and offset) parameters, and P, a positive constant, is the *weight penalty parameter* or *regularization parameter*.

P is also referred to as a *hyperparameter* as it exerts control over the weight parameters — during nonlinear optimization of the objective function, P is held constant while the optimal values of the other parameters are being computed.

With a positive P, the selection of larger $|w_j|$ during optimization would increase the value of J, so larger $|w_j|$ values are penalized. Thus choosing a larger P will more strongly suppress the selection of larger $|w_j|$ by the optimization algorithm.

For sigmoidal activation functions such as tanh, the effect of weight penalty can be illustrated as follows: For $|wx| \ll 1$, the leading term of the Taylor expansion gives

$$y = \tanh(wx) \approx wx\,, \tag{23}$$

i.e. the nonlinear activation function tanh is approximated by a linear activation function when the weight $|w|$ is penalized to be small and x is reasonably scaled.

Using a larger P to penalize weights diminishes the nonlinear modelling capability of the model, thereby avoiding overfitting.

With the weight penalty term in (22), it is essential that the input variables have been scaled to similar magnitudes. The reason is that if e.g. the first input variable is much larger in magnitude than the second, then the weights multiplied to the second input variable will

have to be much larger in magnitude than those multiplied to the first variable, in order for the second input to exert comparable influence on the output as the first input. However the same weight penalty parameter P acts on both sets of weights, thereby greater reducing the influence of the second input variable since the associated weights are not allowed to take on large values.

Similarly, if there are multiple output variables, the target data for different variables should be scaled to similar magnitudes.

Hence when dealing with real unbounded variables, *standardize* the data first, i.e. each variable has its mean value subtracted, and then divided by its standard deviation.

After the NN model has been applied to the standardized variables, each output variable is rescaled to the original dimension, i.e.

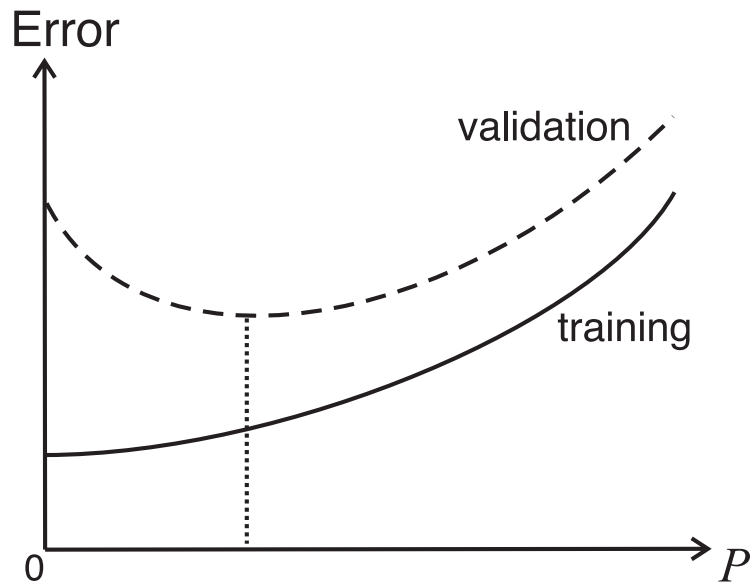
multiply by the original standard deviation and add back the original mean value.

What value should one choose for the weight penalty parameter? A common way to select P is by *validation*. The dataset is divided into training data and validation data. Models are trained using the training data for a variety of P values, e.g. $P = 3, 1, 0.3, 0.1, 0.03, 0.01, \ldots$, or $P = 2^{-p}$ ($p = -1, 0, 1, 2, \ldots$).

Model performance over the validation data is then used to select the optimal P value.

Figure below illustrates the model error (e.g. the MSE) for the training data (solid curve) and for the validation data (dashed curve) as a function of the weight penalty parameter P. The

minimum in the dashed curve gives the optimal P value (as marked by the vertical dotted line).



For an MLP NN model with a single hidden layer, where there are m_1 inputs, m_2 hidden neurons and m_3 output neurons, we have assumed that m_2 is large enough so that the model has enough flexibility to accurately capture the underlying relation in the dataset. In practice, we may not know what m_2 value to use. Hence instead of a single loop of model runs using a variety of P values, we may also need a second loop with $m_2 = 1, 2, 3, \ldots$. The run with the smallest validation error gives the best P and m_2 values.

8.5 Cross-validation [Book, Sect. 6.6]

When there are plentiful data, reserving some data for validation poses no problem. Unfortunately, data are often not plentiful. Cross-validation is a technique which allows the entire dataset to be used for validation.

Given a data record, *K*-fold *cross-validation* involves dividing the record into K (approximately equal) segments. One segment is reserved as validation data, while the other $K-1$ segments are used for model training. This process is repeated K times, so that each segment of the data record has been used as validation data.

Thus a validation error can be computed for the whole data record. A variety of models are run, with different number of model parameters and different weight penalty. Based on the lowest validation error over the whole data record, one can select the best model among the many runs.

E.g. the data record is 50 years long. In 5-fold cross-validation, the record is divided into 5 segments, i.e. years 1-10, 11-20, 21-30, 31-40, 41-50.

First, reserve years 1-10 for validation, and train the model using data from years 11-50.
Next reserve years 11-20 for validation, and train using data from years 1-10 and 21-50.
This is repeated until the final segment of 41-50 is reserved for validation, with training done using the data from years 1-40.

Q1: After you completed the 5-fold cross-validation, some new data have become available. How would you make predictions with the new data?

If one has more computing resources, one can try 10-fold cross-validation, where the 50 year record is divided into ten 5-year segments.

At the extreme, one arrives at the *leave-one-out* cross-validation, where the validation segment consists of a single observation. E.g., if the 50 year record contains monthly values, then there are 600 monthly observations, and a 600-fold cross-validation is the same as the leave-one-out approach.

With time series data, the neighbouring observations in the data record are often not independent of each other due to autocorrelation.

If the dataset has a decorrelation time scale of 9 months, then leaving a single monthly observation out for independent validation would make little sense since it is well correlated with neighbouring observations already used for training.
When there is autocorrelation in the time series data, then the validation segments should be longer than the decorrelation time

scale, i.e. in this example, the validation segments should exceed 9 months.
Even then, at the boundary of a validation segment, there is still correlation between the data immediately to one side which are used for training and those to the other side used for validation.
Thus under cross-validation, autocorrelation can lead to an underestimation of the model error over the validation data, especially when using small validation segments.

Because validation data are used for model selection, i.e. for choosing the best number of model parameters, weight penalty value, etc., the model error over the validation data cannot be considered accurate model forecast error, since the validation data have already been involved in deriving the model.

To accurately assess the model forecast error, the model error needs to be calculated over independent data not used in model training or model selection. Thus the data record needs to be divided into training data, validation data and "*testing*" (or *verification*) data for measuring the true model forecast error.

One then has to do a *double cross-validation*, which can be quite expensive computationally:
Again consider the example of a 50-year data record, where we want to do a 10-fold cross-testing. We first reserve years 1-5 for testing data, and use years 6-50 for training and validation. We then implement 9-fold cross-validation over the data from years 6-50 to select the best model, which we use to forecast over the testing data.
Next, we reserve years 6-10 for testing, and perform a 9-fold cross-validation over the data from years 1-5 and 11-50 to select the

best model (which may have different number of model parameters and different weight penalty value than the model selected in the previous cross-validation).
The process is repeated until the model error is computed over test data covering the entire data record.

References:

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Clarendon Pr., Oxford.