8.6 Bayesian neural networks (BNN) [Book, Sect. 6.7]

While cross-validation allows one to find the weight penalty parameters which would give the model good generalization capability, the separation of the data record into training and validation segments is cumbersome, and prevents the full data record from being used to train the model.

Based on Bayes theorem, MacKay (1992b,a) introduced a Bayesian neural network (BNN) approach which gives an estimate of the optimal weight penalty parameter(s) without the need of validation data. Foresee and Hagan (1997) applied this approach to the MLP NN model using the Levenberg-Marquardt optimization algorithm, with their code implemented in the MATLAB neural network toolbox as trainbr.m.

8.7 Ensemble of models [Book, Sect. 6.8]

In weather forecasting, it is standard practice to run a numerical weather prediction model multiple times from slightly perturbed initial conditions, giving an ensemble of model runs.
The rationale is that the atmospheric models are very unstable to small perturbations in the initial conditions, i.e. a tiny error in the initial conditions would lead to a vastly different forecast a couple of weeks later.

From this ensemble of model runs, the averaged forecast over the individual ensemble members is usually issued as the forecast, while the spread of the ensemble members provides information on the uncertainty of the forecast.

In NN applications, one usually trains a number of models, e.g. to deal with the multiple minima in the objective function, to experiment varying the number of model parameters, etc.

One can test the models' skill over some validation data and simply select the best performer. However, the model skill is dependent on the noise in the validation data, i.e. if a different validation dataset is used, a different model may be selected as the best performer.

Hence, it is common to retain a number of good models to form an *ensemble* of models, and use the ensemble average of their outputs as the desired output. In machine learning jargon, an ensemble of models is called a *committee*.

One way to generate multiple models is through *bagging* (abbreviated from Bootstrap AGGregatING) (Breiman, 1996),

developed from the idea of *bootstrapping* (Efron, 1979; Efron and Tibshirani, 1993) in statistics.

Under bootstrap resampling, data are drawn randomly from a dataset to form a new training dataset, which is to have the same number of data points as the original dataset.
A data point in the original dataset can be drawn more than once into a training dataset.

Q2: For bootstrap resampling applied to a dataset with N observations, derive an expression for the fraction of data in the original dataset drawn in an average bootstrap sample. What is this fraction as $N \to \infty$? [Hint: $\left(1 - \frac{1}{N}\right)^N \to \mathrm{e}^{-1}, \quad \text{as } N \to \infty$]

This is repeated until a large number of training datasets are generated by this bootstrap procedure. During the random draws, predictor and predictand data pairs are of course drawn together.

In the case of autocorrelated data, data segments about the length of the autocorrelation time scale are drawn instead of individual data points — i.e. if monthly data is found to be autocorrelated over the whole season, then one would draw an entire season of monthly data altogether.

In the bagging approach, one model can be built from each training set, so from the large number of training sets, an ensemble of models is derived. By averaging the model output from all individual members of the ensemble, a final output is obtained.

[If the problem is nonlinear classification instead of regression, the final output is chosen by *voting*, i.e. the class most widely selected by the individual members of the ensemble is chosen as the final output.]

Incidentally, the data not selected during the bootstrap resampling are not wasted, as these out-of-bag (OOB) samples can be used as validation data. For instance, to avoid overfitting, these validation data can be used in the *early stopping* approach, i.e. NN model training is stopped when the model's error variance calculated using validation data started to increase.

Finally, from the distribution of the ensemble member solutions, statistical significance can be estimated easily — e.g. from the ensemble distribution, one can simply examine if at least 95% of the

ensemble members give a value greater than zero, or less than zero, etc.

Next compare the error of the ensemble average to the average error of the individual models in the ensemble.

Let $y_{\mathrm{T}}(\mathbf{x})$ denote the true relation, $y_m(\mathbf{x})$ the mth model relation in an ensemble of M models, and $y^{(M)}(\mathbf{x})$ the ensemble average.

The expected squared error of the ensemble average is

$$\begin{aligned}
\mathrm{E}[(y^{(M)} - y_{\mathrm{T}})^2] &= \mathrm{E}\left[\left(\frac{1}{M}\sum_{m=1}^{M} y_m - y_{\mathrm{T}}\right)^2\right] \\
&= \mathrm{E}\left[\left(\frac{1}{M}\sum_{m}(y_m - y_{\mathrm{T}})\right)^2\right] \\
&= \frac{1}{M^2}\,\mathrm{E}\left[\left(\sum_{m}\epsilon_m\right)^2\right], \qquad (1)
\end{aligned}$$

where $\epsilon_m \equiv y_m - y_{\mathrm{T}}$ is the error of the mth model. Cauchy inequality says:

$$\left(\sum_{m=1}^{M}\epsilon_m\right)^2 \leq M\sum_{m=1}^{M}\epsilon_m^2\,. \qquad (2)$$

Q3: Prove Cauchy's inequality for the special case $M = 2$, i.e. prove that

$$(\epsilon_1 + \epsilon_2)^2 \leq 2(\epsilon_1^2 + \epsilon_2^2).$$

$$\frac{1}{M^2}\,\mathrm{E}\left[\left(\sum_{m=1}^{M}\epsilon_m\right)^2\right] \leq \frac{1}{M}\,\mathrm{E}\left[\sum_{m=1}^{M}\epsilon_m^2\right] = \frac{1}{M}\sum_{m=1}^{M}\mathrm{E}[\epsilon_m^2]\,. \qquad (3)$$

$$\mathrm{Eq.}(1) \;=>\quad \mathrm{E}[(y^{(M)} - y_{\mathrm{T}})^2] \leq \frac{1}{M}\sum_{m=1}^{M}\mathrm{E}[\epsilon_m^2]. \qquad (4)$$

This proves that the expected squared error of the ensemble average is less than or equal to the average expected squared error of the individual models in the ensemble, thereby providing the rationale for using ensemble averages instead of individual models.

Note this is a general result, as it applies to an ensemble of dynamical models (e.g. general circulation models) as well as an ensemble of empirical models (e.g. NN models), or even to a mixture of completely different dynamical and empirical models. Perhaps this result is not so surprising, since in social systems we do find that, on average, democracy is better than the average dictatorship!

Next we *restrict to a single model* for generating the ensemble members, e.g. by training the model with various bootstapped resampled datasets, or performing nonlinear optimization with random initial parameters.

Repeat the variance and bias error calculation for an ensemble of models. Again, let $\mathcal{E}[\cdot]$ denote the expectation or ensemble average over all datasets D (or over all random initial weights). Note $\mathcal{E}[\cdot]$ is

distinct from E[·], the expectation over **x**. Since all members of the ensemble were generated from a single model, we have for all the $m = 1, \ldots, M$ members,

$$\mathcal{E}[y_m] = \mathcal{E}[y] \equiv \bar{y}. \tag{5}$$

The expected squared error of the ensemble average $y^{(M)}$ is

$$\begin{aligned}
\mathcal{E}[(y^{(M)} - y_{\mathrm{T}})^2] &= \mathcal{E}[(y^{(M)} - \bar{y} + \bar{y} - y_{\mathrm{T}})^2], \\
&= \mathcal{E}[(y^{(M)} - \bar{y})^2] + \mathcal{E}[(\bar{y} - y_{\mathrm{T}})^2] + 2\mathcal{E}[(y^{(M)} - \bar{y})(\bar{y} - y_{\mathrm{T}})], \\
&= \mathcal{E}[(y^{(M)} - \bar{y})^2] + (\bar{y} - y_{\mathrm{T}})^2 + 2(\bar{y} - y_{\mathrm{T}})\,\mathcal{E}[y^{(M)} - \bar{y}], \\
&= \mathcal{E}[(y^{(M)} - \bar{y})^2] + (\bar{y} - y_{\mathrm{T}})^2,
\end{aligned} \tag{6}$$

as $\mathcal{E}[y^{(M)} - \bar{y}] = 0$.

The first term, $\mathcal{E}[(y^{(M)} - \bar{y})^2]$, is the variance error, while the second term, $(\bar{y} - y_{\mathrm{T}})^2$, is the bias error. Note that the variance error depends on M, the ensemble size, whereas the bias error does not.

Let us examine the variance error:

$$\mathcal{E}[(y^{(M)} - \bar{y})^2] = \mathcal{E}\left[\left(\frac{1}{M}\sum_{m=1}^{M} y_m - \bar{y}\right)^2\right]$$

$$= \mathcal{E}\left[\left(\frac{1}{M}\sum_{m=1}^{M}(y_m - \bar{y})\right)^2\right] = \frac{1}{M^2}\,\mathcal{E}\left[\left(\sum_{m}\delta_m\right)^2\right], \qquad (7)$$

where $\delta_m = y_m - \bar{y}$.

If the errors of the members are uncorrelated, i.e. $\mathcal{E}[\delta_m \delta_n] = 0$ if $m \neq n$, then

$$\frac{1}{M^2}\,\mathcal{E}\left[\left(\sum_m \delta_m\right)^2\right] = \frac{1}{M^2}\,\mathcal{E}\left[\sum_m \delta_m^2\right] = \frac{1}{M}\,\mathcal{E}[\delta^2], \tag{8}$$

as $\mathcal{E}[\delta_m^2] = \mathcal{E}[\delta^2]$, for all m.

Thus if the member errors are uncorrelated, the variance error of the ensemble average is

$$\mathcal{E}[(y^{(M)} - \bar{y})^2] = \frac{1}{M}\,\mathcal{E}[(y - \bar{y})^2], \tag{9}$$

where the right hand side is simply the variance error of a single member divided by M.

Hence, the variance error of the ensemble average $\to 0$ as $M \to \infty$. Of course, the decrease in the variance error of the ensemble average will not be as rapid as M^{-1} if the errors of the members are correlated.

In summary, for the ensemble average, the variance error can be decreased by increasing the ensemble size M, but the bias error is unchanged.

This suggests that one should use models with small bias errors, and then rely on the ensemble averaging to reduce the variance error. In other words, one would prefer using models which overfit slightly to models which underfit, as ensemble averaging can alleviate the overfitting.

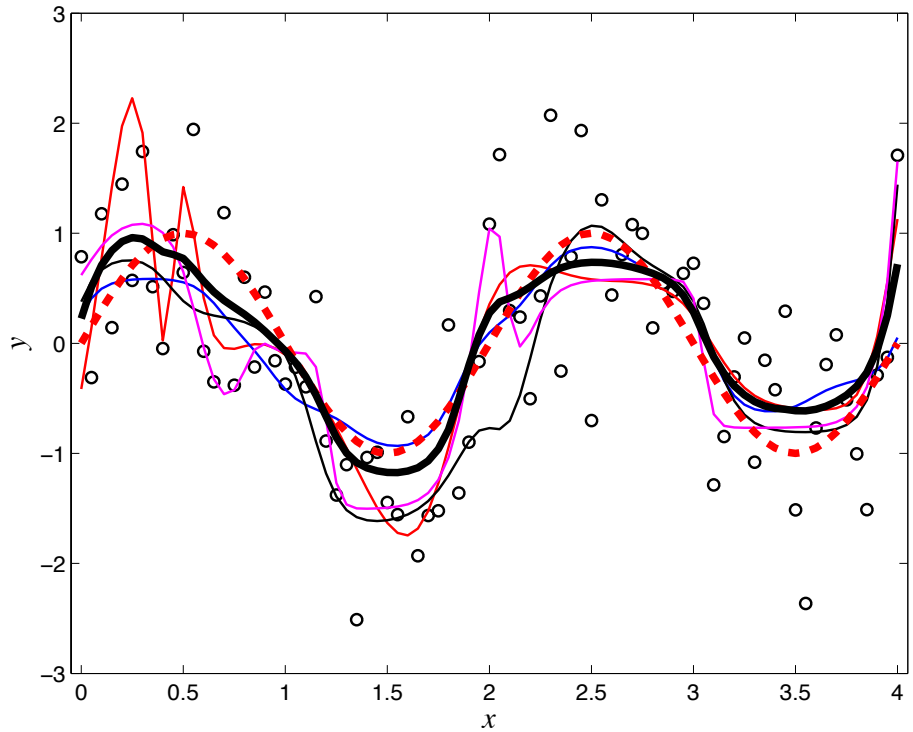

Figure : The data (circles) are the signal, $y = \sin x$ (thick dash curve), plus Gaussian noise (with the same standard deviation as the signal). MLP NN from Matlab (using 1 hidden layer of 10 neurons, and Levenberg-Marquardt optimization without regularization) was run 30 times from random initial weights. Four individual runs are shown as thin curves, and the ensemble average over 30 runs as the thick curve.

Q4: You have 50 years of data, and you want to do a 25-fold cross-validation of your MLP NN model. You also want to run an ensemble of 30 runs with random initial weights for the NN model. How many NN model runs do you have to perform?

So far, all the members are equally weighted in forming the ensemble average. A more sophisticated way to form the ensemble average is to use an MLP NN model to perform nonlinear ensemble averaging.

The MLP NN has M inputs and 1 output. The M inputs simply receive the outputs from the M trained ensemble members, while the output is trained towards the same target data used in training the individual ensemble members.

Krasnopolsky (2007) used an ensemble of 10 MLP NN models to emulate sea level height anomalies using state variables from an ocean model as input. The output of the 10 ensemble members were then nonlinearly averaged by another MLP NN and compared with simple averaging.

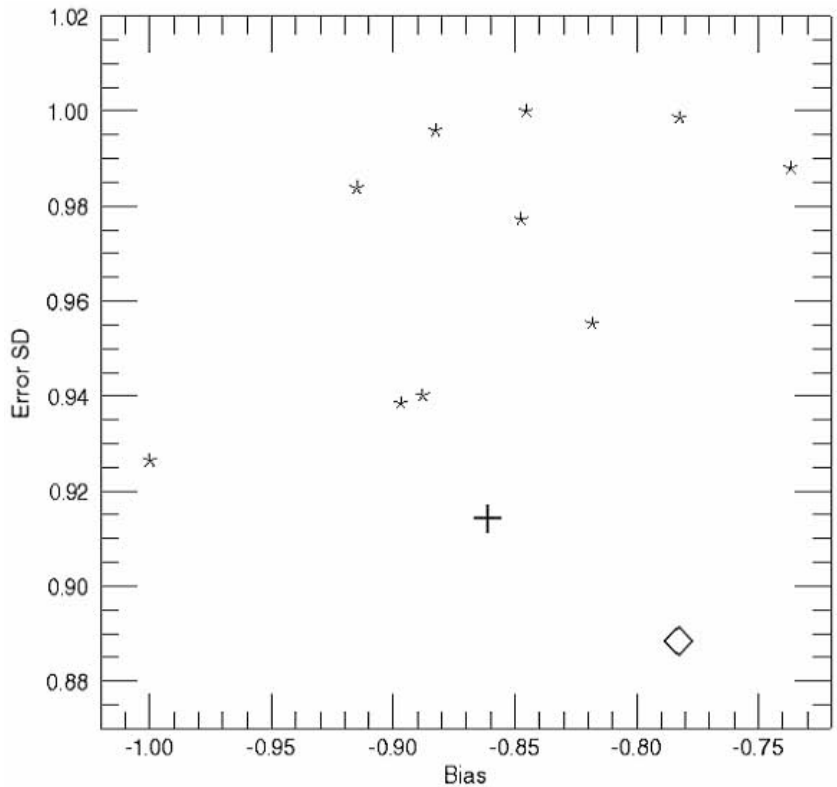

Figure : Scatter plot of the model bias versus the error standard deviation (SD) for the 10 individual ensemble members (asterisks), the simple ensemble average (cross) and the nonlinear ensemble average by NN (diamond). [Reproduced from Krasnopolsky (2007)]

The simple ensemble average has smaller error standard deviation than all 10 individuals, but its bias is just the average of the bias of the individuals. In contrast, the nonlinear ensemble average by NN results in even smaller error standard deviation plus a considerable reduction in the bias.

Another ensemble/committee approach called boosting differs from other ensemble methods such as bagging in that the models in the ensemble are trained in sequence, with "improvement" from one

model to the next, and the final output of the ensemble being a weighted sum of the output from all its members.

The most popular boosting algorithm is AdaBoost (Freund and Schapire, 1997), developed originally for classification problems, but also extended to regression problems.

The key idea is that in the mth model there are some data points which are not well predicted, so when we train the next model, we increase the weighting of these difficult data points in our objective function.
This type of learning approach is used by students, e.g. if a student does poorly in some courses, he will put more effort into the difficult courses in order to improve his overall grade.

Since boosting tries to improve on predicting the difficult data points, we must ensure that the difficult data points are of sound quality, i.e. the data are not simply wrong!

The outline of the boosting approach is as follows:
Let $w_n^{(m)}$ denote the weight placed on the nth data point $(n = 1, \ldots, N)$ in the objective function of the mth model $(m = 1, \ldots, M)$. For the first model, we use uniform weight, i.e. $w_n^{(1)} = 1/N$.

We next generate a sequence of models: For model m, the weights $w_n^{(m)}$ are increased relative to $w_n^{(m-1)}$ for a data point n if this point was poorly predicted by model $m - 1$.

The final output of the M models is a weighted sum of the individual model outputs, with the weights a_m being larger for the better models.

8.8 Linearization from time-averaging [Book, Sect. 6.10]

Time-averaging is widely used to reduce noise in the data; however, it also linearizes the relations in the dataset.

In a study of the nonlinear relation between the precipitation rate (the predictand) and 10 other atmospheric variables (the predictors $\mathbf{x}$) in the NCEP/NCAR reanalysis data (Kalnay et al., 1996), Yuval and Hsieh (2002) examined the daily, weekly and monthly averaged data by nonlinear multiple regression using the MLP NN model.

They discovered that the strongly nonlinear relations found in the daily data became dramatically reduced by time-averaging to the almost linear relations found in the monthly data.

Central limit theorem from statistics tells us that by averaging data, the data distribution approaches the (multivariate) Gaussian distribution.

To visualize this effect, consider the synthetic dataset

$$y = x + x^2 + \epsilon, \tag{10}$$

where x is a Gaussian variable with unit standard deviation and ϵ is Gaussian noise with a standard deviation of 0.5. Averaging this 'daily' data over 7 days and over 30 days reveals a dramatic weakening of the nonlinear relation:

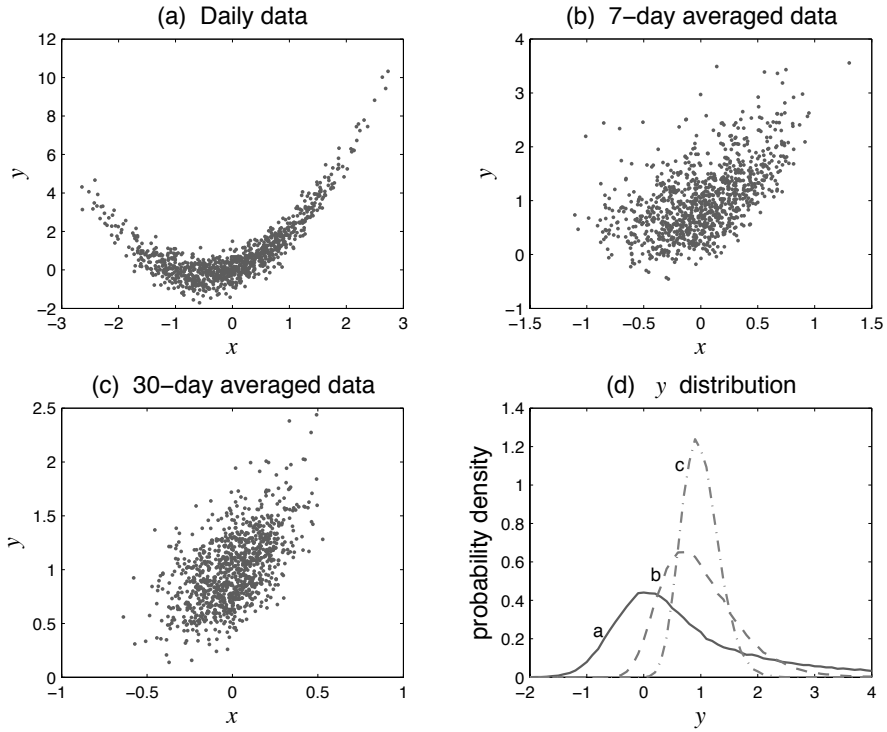

Figure : Effects of time-averaging on a nonlinear relation. (a) Synthetic 'daily' data from a quadratic relation between x and y, and the data time-averaged over (b) 7 observations and (c) 30 observations. The probability density distribution of y is shown in (d) for cases (a), (b) and (c).

With real data, there is autocorrelation in the time series, so the monthly data will be effectively averaging over far fewer than 30 independent observations as done in this synthetic dataset.

If the data has strong autocorrelation, so that the integral time scale from the autocorrelation function is not small compared to the time-averaging window, then there are actually few independent observations used during the time-averaging, and the central limit theorem does not apply.

While time-averaging tends to reduce the nonlinear signal, it also smooths out the noise. Depending on the type of noise (and perhaps on the type of nonlinear signal), it is possible that time-averaging may nevertheless enhance the detection of a nonlinear signal above the noise for some datasets.

Be aware that time-averaging could have a major impact on the modelling or detection of nonlinear empirical relations, and that a nonlinear machine learning method often outperforms a linear method in weather applications, but fails to do so in climate applications.

Due to recent interest in the climate of extremes, climate is described by more than just the averaging of daily data. E.g. annual indices of 10th percentile of daily minimum temperature, 90th percentile of daily maximum temperature, etc. Nonlinearity is better preserved in such extreme climate indices.

8.9 Regularization of linear models (Hastie et al., 2009, Sect.3.4)

Regularization (weight penalty) is widely used to prevent overfitting in nonlinear models like MLP NN. It is also used in linear regression in place of stepwise regression to alleviate overfitting from having too many predictors. I.e. instead of deleting some predictors, regularization assigns very small (or zero) weights to some predictors.

First remove the mean for each predictor variable x_l $(l = 1, \ldots, m)$ and for the response variable y. Multiple linear regression is

$$y_i = \sum_{l=1}^{m} x_{il} w_l + e_i, \qquad i = 1, \ldots, n, \tag{11}$$

with n observations, e_i the errors or residuals, and w_l the regression coefficients (no constant coefficient w_0 due to the mean removal). Rewrite as

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \mathbf{e}\,. \tag{12}$$

Add weight penalty (with parameter p) to objective function J:

$$J = \sum_{i=1}^{n} \left(y_i - \sum_{l=1}^{m} x_{il} w_l \right)^2 + p \sum_{l=1}^{m} w_l^2 \,. \tag{13}$$

Setting gradient of J to zero yields [Book, Eqs.(7.10)-(7.14)]:

$$\mathbf{w} = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + p\mathbf{I})^{-1}\,\mathbf{X}^{\mathrm{T}}\mathbf{y}\,, \tag{14}$$

with identity matrix $\mathbf{I}$. This is called ridge regression (or Tikhonov regularization). If $p = 0$, one gets back multiple linear regression.

Even better is lasso (least absolute shrinkage and selection operator), where the mean absolute norm is used (instead of the mean squared norm) for weight penalty

$$J = \frac{1}{2} \sum_{i=1}^{n} \left(y_i - \sum_{l=1}^{m} x_{il} w_l \right)^2 + p \sum_{l=1}^{m} |w_l| \,. \tag{15}$$

Lasso will set more weights to zero than ridge regression, and tends to predict better (Hastie et al., 2009, Sect.3.4). Delete predictors with zero weights, hence useful in predictor selection.

Matlab codes

For ridge regression:
`http://www.mathworks.com/help/stats/ridge.html`
For lasso:
`http://www.mathworks.com/help/stats/lasso.html`

For Bayesian NN:
`www.mathworks.com/help/toolbox/nnet/ref/trainbr.html`

An alternative for BNN is the Netlab library written in Matlab:

`www1.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/downloads`
See book "Netlab" by Nabney, Chap.9, esp. Sect. 9.6.1.

References:

Breiman, L. (1996). Bagging predictions. *Machine Learning*, 24:123–140.

Efron, B. (1979). Bootstrap methods: another look at the jackknife. *Annals of Statistics*, 7:1–26.

Efron, B. and Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. CRC Press, Boca Raton, Florida.

Foresee, F. D. and Hagan, M. T. (1997). Gauss-Newton approximation to Bayesian regularization. In *Proceedings of the 1997 International Joint Conference on Neural Networks*.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretical generalization of on-line learning and an application to boosting. *Journal of Computer System Sciences*, 55:119–139.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2nd edition.

Kalnay, E., Kanamitsu, M., Kistler, R., Collins, W., Deaven, D., Gandin, L., et al. (1996). The NCEP/NCAR 40-year reanalysis project. *Bulletin of the American Meteorological Society*, 77(3):437–471.

Krasnopolsky, V. M. (2007). Neural network emulations for complex multidimensional geophysical mappings: Applications of neural network techniques to atmospheric and oceanic satellite retrievals and numerical modeling. *Reviews of Geophysics*, 45(3). RG3009, doi:10.1029/2006RG000200.

MacKay, D. J. C. (1992a). A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472.

MacKay, D. J. C. (1992b). Bayesian interpolation. *Neural Computation*, 4(3):415–447.

Nabney, I. T. (2002). *Netlab: Algorithms for Pattern Recognition.* Springer, London.

Yuval and Hsieh, W. W. (2002). The impact of time-averaging on the detectability of nonlinear empirical relations. *Quarterly Journal of the Royal Meteorological Society*, 128:1609–1622.