15 Generalization Prediction and Assessment

The following sections outline some approaches to predicting and estimating generalization ability, either a priori from static parameters such as network size, or after observing training performance. The problem of estimating the true performance of a prediction system trained on a limited data set is a basic statistics problem so it is not surprising that many of these methods are direct applications of statistical techniques. Techniques mentioned here are covered in more depth in [44, 389, 317].

15.1 Cross-Validation

A rather direct way to estimate the generalization ability of a system is to measure the error on a separate data set that the network has not seen during training. In simple cross-validation, that is, the holdout method, the available data is divided into two subsets: a training set used to train the network and a test set used to estimate the true error rate. To avoid obvious bias, both sets should be random samples of the same population.

Ideally, both sets should be large because the larger the training set, the more accurate the approximation learned by the trained system, and the larger the test set, the more accurate the estimate of the true error rate. When data are limited, these goals conflict. As a compromise, sets of roughly equal size are usually chosen.

With large sample sizes the holdout method can be accurate, but it has limitations when sample sizes are small. The test samples are unavailable for training, so the network must be trained on less data with more risk of overtraining or overfitting. The validation set is used to guard against this, but, with just a small amount of validation data, the error estimate has a large variance and may be unreliable; an uncharacteristic test set could give a bad estimate of the error. If the training error surface is distorted because of sampling deficiencies, the validation error surface is likely to be similarly distorted when the data sets have similar sizes. In order for the validation set to be a better predictor of the true generalization error than the training set, it will usually have to be several times larger but this limits the amount of data that can be used for training.

Another problem is that, depending on the training algorithm, the solution could be indirectly biased toward the validation set so a third, completely different, data set is needed to form an unbiased estimate of the error. That is, it is common to train a number of networks and choose the one that performs best on the validation set. If thousands of networks were generated, a few might coincidentally have low errors on the validation set but still not generalize well. Because the validation set is used, albeit indirectly, as part of the training process, there is a danger of obtaining a biased solution.

Simple cross-validation as described here uses a single holdout set to estimate the generalization error. Resampling techniques such as leave-one-out, k-fold cross-validation,

and bootstrapping [389, 115, 116] address limitations of the single holdout method by averaging over multiple holdout experiments using different partitions of the data into training and test sets. Once impractical, these sorts of methods have become feasible due to increasing computer processing power. Advantages are that the error estimates are generally more accurate, the network can be trained on almost all the data, and it can be tested on all of it. The drawback is an increased computational burden. It should be noted that these are nonparametric methods that do not make restrictive assumptions about data distributions and are not restricted to linear models.

Bootstrapping is one of the most accurate techniques, in general, but also one of the slowest. As noted, the estimate obtained from a single holdout set may have a large variance. Bootstrapping lowers the variance (at the expense of a slight increase in the bias) by averaging estimates obtained from many different partitions of the data [389]. It is common to use hundreds or thousands of subset estimates. If the method is used to obtain a more accurate one-time estimate of the generalization ability of a trained network, the computational burden may not be a critical factor because network training times are already long in most cases. This probably is not a practical way of comparing network architectures if each subset sample requires the training of a new network.

15.2 The Bayesian Approach

Bayesian methods provide ways to describe the effects of biases, sampling distributions, noise, and other uncertainties. The Bayesian approach incorporates external knowledge (or biases) about the target function in the form of prior probabilities of different hypothesis functions [251, 253]. The data set $D = \{(x_i, t_i),\ \ i = 1 \ldots m\}$, where x are the inputs and t the targets, is typically modeled as the sum of a deterministic function f and additive perturbations (noise) n representing prediction errors

$$t_i = f(x_i) + n_i. \tag{15.1}$$

Assuming the network output $y(x)$ is correct, the probability of the observed data is the probability that an error $t_i - y(x_i)$ is due entirely to the noise

$$P\left[t_i = f(x_i) + n_i \mid f = y\right] \quad = \quad P\left[n_i = t_i - y(x_i)\right]. \tag{15.2}$$

If the training cases are independent and the noise is independent and identically distributed, the probability of the entire training set given the assumption $f = y$ is

$$P\left[D \mid f = y\right] = \prod_{i=1}^{m} P\left[n_i = t_i - y(x_i)\right]. \tag{15.3}$$

If the noise is assumed to be Gaussian $N(0, \sigma)$, then

$$P\left[D \mid f = y\right] = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{(t_i - y(x_i))^2}{2\sigma^2}\right) \tag{15.4}$$

$$= \frac{1}{\sqrt{(2\pi)^m}\,\sigma^m} \exp\left(\frac{-E}{2\sigma^2}\right) \tag{15.5}$$

where $E = \sum_{i=1}^{m}(t_i - y(x_i))^2$ is the usual sum of squared errors. Minimization of the mean-squared-error is thus equivalent to selection of a maximum likelihood model under the assumption that the errors are Gaussian. Other error functions are appropriate under different assumptions about the error distribution; a number are reviewed by Rumelhart et al. [328].

By Bayes' rule, the evidence for a model $y(x)$ given the data is

$$P\left[y \mid D\right] = \frac{P\left[D \mid f = y\right] P\left[f = y\right]}{P\left[D\right]}. \tag{15.6}$$

External constraints (such as a bias toward smooth solutions) are reflected in the choice of model prior probabilities $P\left[f = y\right]$. The denominator $P\left[D\right]$ is the same for all models and can be ignored in comparing models.

Different model configurations can be compared by decomposing y into a choice of weights $\mathbf{w}$ and a network architecture H. (H specifies the number of layers, number of nodes, etc. and $\mathbf{w}$ specifies a set of weights in the given architecture.) The probability that a given set of weights is the correct choice given the data and the model H is

$$P\left[\mathbf{w} \mid D, H\right] = \frac{P\left[D \mid \mathbf{w}, H\right] P\left[\mathbf{w} \mid H\right]}{P\left[D \mid H\right]}. \tag{15.7}$$

(Note: this is different from the probability that some learning algorithm will produce a particular set of weights.) For a given H, the prior $P\left[\mathbf{w} \mid H\right]$ can reflect a bias in favor of small weight values, for example. The probability of different models H_i is given by [251]

$$P\left[H_i \mid D\right] \propto P\left[D \mid H_i\right] P\left[H_i\right]. \tag{15.8}$$

The priors $P\left[H_i\right]$ can reflect a bias in favor of models with small numbers of parameters, for example.

If all prior probabilities $P\left[H_i\right]$ are approximately equal, then the models can be compared based on the evidence [251]

$$P\left[D \mid H_i\right] = \int P\left[D \mid \mathbf{w}, H_i\right] P\left[\mathbf{w} \mid H_i\right]\, d\mathbf{w}. \tag{15.9}$$

If $\mathbf{w}$ is k-dimensional and if the posterior distribution is approximately Gaussian, then [251, 253]

$$P\left[D \mid H_i\right] \approx P\left[D \mid \mathbf{w}_{mp}, H_i\right] P\left[\mathbf{w}_{mp} \mid H_i\right] (2\pi)^{k/2} \det^{-1/2} \mathbf{A} \tag{15.10}$$

where $\mathbf{w}_{mp}$ is the maximum likelihood set of weights found by minimizing E and $\mathbf{A} = -\nabla\nabla \log P\left[\mathbf{w} \mid D, H_i\right]$ is the Hessian of E with respect to $\mathbf{w}$ evaluated at $\mathbf{w}_{mp}$. It has been argued [251, 253] that this approach has a built-in bias for simple models because the Occam factor $P\left[\mathbf{w}_{mp} \mid H_i\right] (2\pi)^{k/2} \det^{-1/2} \mathbf{A}$ is smaller for more complex models.

Remarks Perhaps in part because of its widespread success, criticisms of the Bayesian approach have been raised. From the viewpoint of the prediction system, approximation errors are random and unpredictable (otherwise it would be able to eliminate them) so errors are treated like noise and usually assumed to be independent and identically distributed. All network functions and most real target functions have structure, however, so errors may not be independent. The errors are often assumed to have some tractable distribution such as Gaussian (justified by the central limit theorem), but approximations are often made that hold only for large sample sizes. A common criticism of Bayesian approaches in general is that the prior probabilities may be subjective (i.e., biases rather than measured probabilities). This is not a major problem in cases where all the probabilities can be measured, or when the analysis is used for qualitative understanding, but may be a problem in quantitative predictions. Many of these criticisms are objections to the way the theory is applied, rather than defects of the theory itself.

15.3 Akaike's Final Prediction Error

A standard estimate of the test set error for a linear system is Akaike's final prediction error (FPE) [6, 7]:

$$\hat{E}_{test} = \frac{p + N}{p - N} E_{train} \tag{15.11}$$

where p is the number of training samples and N is the number of parameters in the model.

A related estimate, Akaike's information criterion (AIC) [7], has been used to compare linear models with different numbers of parameters

$$AIC(\theta) = (-2) \log(\text{maximum likelihood}) + 2k \tag{15.12}$$

where θ is a model with k parameters. If equation 15.5 is valid, then

$$AIC(\theta) = 2E + 2k \tag{15.13}$$

where E is the usual mean-squared error. With this cost function, simple models are preferred over complex models if the increased cost of the additional parameters in the complex models do not result in corresponding decreases in the error. There are a number of similar criteria developed for linear systems.

The Effective Number of Parameters A problem with these estimates is that they are asymptotic approximations valid only for linear systems with large sample sets. The assumptions are invalid for small sample sizes [61]. An extension to linear systems with finite sample sizes is considered by Hansen [152]. Some work has been done to extend this to nonlinear systems by estimating the *effective number of parameters* from derivatives of the error with respect to the weights [273]. One form, for a single-hidden-layer network, is [365]

$$\hat{E}_{test} = \frac{p + N_{eff}}{p - N_{eff}} E_{train} \tag{15.14}$$

where

$$N_{eff} = \sum_{ij}^{N_w} \left(\frac{\lambda_{ij}}{\lambda_{ij} + 2\alpha_w / p} \right)^2 + \sum_{j}^{N_W} \left(\frac{\Lambda_{ij}}{\Lambda_{ij} + 2\alpha_W / p} \right)^2 . \tag{15.15}$$

Here N_w and N_W are the number of weights (including thresholds) in the hidden and output units, respectively, and the λ's are second derivatives $\lambda_{ij} \equiv \partial^2 E_{train} / \partial w_{ij}^2$, $\Lambda_{ij} \equiv \partial^2 E_{train} / \partial W_{ij}^2$. This estimate is used to determine when to stop a pruning algorithm in [365].

15.4 PAC Learning and the VC Dimension

Valiant's PAC (probably approximately correct) learning model and the Vapnik-Chervonenkis (VC) dimension have been used to study the problem of learning binary valued functions from examples, for example, [376, 49, 1]. These relate the complexity of a learning system to the number of examples required for it to learn a particular function from a given class of functions. Briefly, if the number of examples is small relative to the complexity of the system, the generalization error is expected to be high.

Abu-Mostafa [1] provides a brief tutorial on which the following paragraphs are based. More complete descriptions can be found in several texts, [282] for example. A learning algorithm samples points $x \in X$, observes the target values $f(x)$, and tries to find a hypothesis function $g(x)$ that matches f everywhere. The examples are assumed to be

drawn independently from some fixed distribution, which can be arbitrary as long as the same distribution is used for learning and testing. The hypothesis functions are drawn from a restricted class G. The algorithm chooses among hypothesis functions based on their performance on the samples. If the number of samples is too small, the estimated performance ν_g (the frequency of error on the test set) could differ significantly from the actual performance π_g and the algorithm could be fooled. A condition for uniform convergence [379] is

$$P\left[\sup_{g\in G}\left|\nu_g-\pi_g\right|>\epsilon\right]\leq 4m(2N)\,e^{-\epsilon^2N/8} \tag{15.16}$$

where m is a function which depends on G. For ν_g to approach π_g as the number of samples N becomes large, the righthand term must approach 0. The $e^{-\epsilon^2N/8}$ term decays exponentially with N so convergence is possible if the function $m(2N)$ does not grow too fast. This is satisfied when $m(N)$ is polynomial in N, for example [1].

The growth function $m(N)$ measures the number of ways that G can label N arbitrary but independent points. The VC dimension d of class G measures the maximum number of points N for which a function in G can always be found that will fit the points no matter how they are labeled. For $N<d$, $m(N)$ grows exponentially with N (i.e. 2^N). For $N>d$, G cannot realize some labelings of the points and $m(N)$ ceases to grow exponentially. Thus $m(N)\leq 2^d+1$.

The importance of this to learning is that if the number of examples is large compared to the VC dimension of the target function class, then equation 15.16 promises uniform convergence. The estimated error rates will then be close to the actual rates and the learning algorithm has a reliable method to choose the best hypothesis.

The sample complexity $m(\epsilon,\delta)$ of a class G is the smallest sample size that guarantees uniform convergence for all target concepts in G and all sampling distributions. An upper bound is [49]

$$m(\epsilon,\delta)=\max\left\{\frac{4}{\epsilon}\log_2\frac{2}{\delta},\ \frac{8d}{\epsilon}\log_2\frac{13}{\epsilon}\right\}. \tag{15.17}$$

A lower bound is [117]

$$\Omega\left(\frac{1}{\epsilon}\ln\frac{1}{\delta}+\frac{d}{\epsilon}\right). \tag{15.18}$$

This is relevant to neural network training in that a network is capable of representing a certain class of concepts and so has some particular VC dimension (e.g., the VC dimension of a simple perceptron with k inputs is $k+1$). If the network can be trained on a number

$m(\epsilon, \delta)$ of examples achieving an error no greater than ϵ on the training set, then, with probability $1 - \delta$, one expects that the true error is no greater than 2ϵ and similar average error can be expected on novel examples drawn from the same distribution.

By assuming a uniform sampling distribution, the VC dimension of a feedforward network with N nodes and W weights has been estimated as [33, 32, 34]

$$d_{VC} \leq 2W \log_2(eN). \tag{15.19}$$

This has been used to put an upper bound on the number of examples that might be needed to achieve a given generalization error rate. If the network can be trained with

$$m \geq O\left(\frac{W}{\epsilon} \log_2 \frac{N}{\epsilon}\right) \tag{15.20}$$

randomly selected training examples achieving an error rate of less than $\epsilon/2$, then a generalization error rate of at most ϵ can be expected for examples drawn from the same distribution (for $0 < \epsilon \leq 1/8$). This agrees with the rule of thumb that roughly $O(W/\epsilon)$ examples are needed to achieve a generalization error less than ϵ.

For a network with N inputs and one hidden layer of H units [33],

$$d_{VC} \geq 2\lfloor \frac{H}{2} \rfloor N. \tag{15.21}$$

This is approximately equal to the number of weights W for large H, also suggesting that $\Omega(W/\epsilon)$ examples are needed to achieve a generalization error less than ϵ.

Problems Because the theory is very general, the estimated bounds are loose. Its main value to neural network design seems to be to indicate that if there are enough examples, then the training error should be a good predictor of the generalization error. This allows broad statements to be made about the appropriate size of a network given a particular amount of training data. These bounds, however, do not apply to networks with multiple continuous outputs and they do not say how to choose a suitable network given a particular set of examples to be learned. Other concerns are that the analysis is asymptotic, whereas data are often finite, and the bounds are worst-case (over any data distribution) and appear to be overly pessimistic; the number of examples required to satisfy PAC requirements is often very high. For practical problems, the average-case behavior may be more important. Numerical tests [85, 172] show that the average behavior can be better than the VC bounds in many cases.

The basic theory also ignores peculiarities of specific learning algorithms. Techniques such as pruning and regularization may add constraints that prevent full exploitation of the intrinsic network complexity. Large networks can realize complex functions, but they

can also mimic simple (e.g., linear) functions. A network is usually initialized with small weights and the resulting input-output relationship is very smooth, almost linear. As the network is trained, the weights become larger and the transfer function more complex. The complexity of the transfer function thus depends on other factors in addition to network size. Some work has been done to estimate the *effective* number of parameters based on the network response function [378, 147]. Normally, the effective dimension cannot be calculated analytically but it may be estimated from network performance. This approach may be able to account for dynamic changes in network complexity as a result of training.

This is an active research area, however, and new results continue to appear. It is known, for example, that networks with continuous activation functions are more powerful than networks of threshold units. Recent work suggests that networks with continuous unit activations can have VC dimension at least as large W^2, where W is the number of weights [213]. This means that it may be very hard to constrain a network with reasonable numbers of examples.