6. Random Vector Functional Link (RVFL) Networks

This chapter summarises the concept of the random vector functional link network (RVFL), that is, a multilayer perceptron (MLP) in which only the output weights are chosen as adaptable parameters, while the remaining parameters are constrained to random values independently selected in advance. It is shown that an RVFL is an efficient universal approximator that avoids the curse of dimensionality. The proof is based on an integral representation of the function to be approximated and subsequent evaluation of the integral by the Monte-Carlo approach. This is compared with the universal approximation capability of a standard MLP. The chapter terminates with a simple experimental illustration of the concept on a toy problem.

6.1 The RVFL theorem

Multilayer perceptrons (MLP) are universal approximators. However, due to the non-convexity of the error surface defined in the space of adaptable parameters, the training process can become rather awkward. It is well-known that local minima can lead to suboptimal network configurations, and 'flat' regions, that is regions corresponding to quasi-zero eigenvalues of the Hessian, normally result in extremely long convergence times. For this reason Pao, Park and Sobajic [48] studied a modification of the standard MLP, which they referred to as the *random vector functional link network* (RVFL). The architecture of an RVFL is identical to that of an MLP. Assume, for simplicity, that the function to be modelled is a scalar, then the output of the network is given by

$$f(\mathbf{x}) = \sum_{i=1}^{n} w_i g(\mathbf{u}_i^{\dagger}\mathbf{x} - b_i) \tag{6.1}$$

where $g(.)$ denotes a nonlinear, usually sigmoidal, transfer function. The difference between the standard MLP and the RVFL is that in the former all

D. Husmeier, *Neural Networks for Conditional Probability Estimation*

network parameters, i.e. the output weights w_k, hidden weights $\mathbf{u}_i$ and hidden thresholds b_i, are optimised in the training process, while in the latter only the output weights w_k can be adapted. The parameters of the hidden layer, $\mathbf{u}_i$ and b_i, are selected randomly and independently in advance, and then remain constrained to these values during training. In this way the error function E becomes quadratic in the remaining adaptable parameters $\mathbf{w}$, which solves the problem of local minima and allows fast optimisation using conjugate gradients or matrix inversion techniques. That is,

$$E(\mathbf{w}) = E(\hat{\mathbf{w}}) + \frac{1}{2}(\mathbf{w} - \hat{\mathbf{w}})^{\dagger}\mathbf{H}(\mathbf{w} - \hat{\mathbf{w}}), \tag{6.2}$$

$$\nabla E(\mathbf{w}) = \mathbf{H}(\mathbf{w} - \hat{\mathbf{w}}) \tag{6.3}$$

where the Hessian $\mathbf{H}$ is semi positive definite. If $\mathbf{H}$ is also strictly positive definite, that is, if there exists a well-defined global minimum $\hat{\mathbf{w}}$, then this minimum can, in principle, be found in a single step by application of Newton's rule:

$$\hat{\mathbf{w}} = \mathbf{w} - \mathbf{H}^{-1}\nabla E(\mathbf{w}) = -\mathbf{H}^{-1}\nabla E(0) \tag{6.4}$$

The question of interest is, of course, what are the consequences of imposing the constraints on the hidden parameters? Igelnik and Pao [32] showed that, if the unknown function f satisfies certain regularity conditions (which are the same as for the standard universal approximation theorem, namely that f is bounded, continuous, and defined on a compact set), then an arbitrarily close approximation of f can be attained by an RVFL of sufficient complexity. This universal approximation capability is not, in itself, of much value, though, since it can also be achieved with fixed basis functions, like polynomials. The latter, however, are inflicted by the *curse of dimensionality*, with an approximation error that cannot be made smaller than of order $\mathcal{O}(1/n^{2/m})$, where m is the dimension of the input vector $\mathbf{x}$ and n the number of basis functions [3]. Obviously, in a high-dimensional setting this approximation rate of $2/m$ becomes vanishingly small. Multilayer perceptrons, on the other hand, have been found to have an approximation error of order $\mathcal{O}(\frac{1}{n})$ irrespective of the dimension of the input space m [3], [43]. The important finding of Igelnik and Pao [32] is that, provided the function to be approximated satisfies certain smoothness conditions[1], then the RVFL shows the same rate of approximation error convergence of $\mathcal{O}(\frac{1}{n})$ as an MLP. Hence an RVFL is, like an MLP, an *efficient* universal approximator that is free from the curse of dimensionality.

[1] The function needs to be Lipshitz continuous, as will be discussed in the next section. See (6.24) for a definition.

6.2 Proof of the RVFL theorem

The following section combines the ideas of [32] with the mathematical approach adopted in [43]. The main objective is to summarise and explain the basic principles of the RVFL scheme. Mathematical rigour is therefore sacrificed in part in order to focus on the main aspects in a simplified, less opaque language. The mathematically interested reader is referred to the original work by Igelnik and Pao [32] and Murata [43].

Let f, the function to be approximated, be a bounded and continuous function defined on a compact set $K \subset \mathbb{R}^m$ (these are the same conditions as required for the standard universal approximation theorem). Let g be a bounded, differentiable function on $\mathbb{R}$, whose derivative is square integrable (i.e. $\int_{\mathbb{R}}[g'(x)]^2 dx < \infty$)[2] . For convenience it will be further assumed that $|g(x)| \leq 1 \forall\, x$. Examples for g are sigmoidal functions, like $\tanh(x)$ and $\varsigma(x) = [1+e^{-x}]^{-1}$, which are usually employed as transfer functions in neural networks. Then the following integral representation for f can be shown to hold [32], [43]:

$$f(\mathbf{x}) \quad = \quad \int_{\mathbb{R}^{m+1}} T(\mathbf{u}, b) g(\mathbf{u}^\dagger \mathbf{x} - b)\, d\mathbf{u}\, db. \tag{6.5}$$

Here $\mathbf{u} \in \mathbb{R}^m$, $b \in \mathbb{R}$, and $T : \mathbb{R}^{m+1} \rightarrow \mathbb{R}$ is given by an inverse transformation,

$$T(\mathbf{u}, b) \quad \propto \quad \int_{\mathbb{R}^m} \check{g}(\mathbf{u}^\dagger \mathbf{x} - b) f(\mathbf{x}) d\mathbf{x}. \tag{6.6}$$

The so-called decomposing kernel $\check{g}$ needs to satisfy certain *conjugacy* conditions listed in [43] to ensure that it is *conjugate* to the composing kernel g. However, for the purpose of this paragraph, these details are not important and will be omitted. Note that the above transformations have certain similarities with the Fourier transform, except that the Fourier transform and inverse Fourier transform give a one-to-one correspondence between functions defined on the same set $\mathbb{R}^m$, whereas (6.5) and (6.6) map between two different spaces of functions defined on $\mathbb{R}^m$ and $\mathbb{R}^{m+1}$. Hence (6.5) is more akin to a wavelet transform, allowing representation of f by different transforms $T_1(\mathbf{u}, b)$, $T_2(\mathbf{u}, b)$ with respect to different decomposing kernels $\check{g}_1$, $\check{g}_2$.

Let us define

[2] In [43], g itself is required to be square integrable, which excludes the sigmoidal function frequently employed in neural networks. A heuristic reasoning for the above generalisation is based on the fact that a bell-shaped function (which satisfies the stricter condition) can always be represented by a linear combination of two sigmoidals. A mathematical, though more cumbersome, proof can be found in [32]. Since the purpose of this section is to present ideas rather than maintain strict mathematical rigour, these subtleties will not be considered here.

$$V := [-\Omega, \Omega]^{m+1}, \qquad |V| := (2\Omega)^{m+1}, \tag{6.7}$$

where $\Omega \in \mathbb{R}$, and

$$\begin{aligned} f_\Omega(\mathbf{x}) &:= \int_V T(\mathbf{u}, b) g(\mathbf{u}^\dagger \mathbf{x} - b)\, d\mathbf{u}\, db \\ &= \int_{-\Omega}^{\Omega} \cdots \int_{-\Omega}^{\Omega} T(u_1, \ldots, u_m, b) g(\mathbf{u}^\dagger \mathbf{x} - b) du_1 \ldots du_m db. \end{aligned} \tag{6.8}$$

Obviously, from (6.5) and (6.8) it follows that

$$f(\mathbf{x}) = \lim_{\Omega \to \infty} f_\Omega(\mathbf{x}). \tag{6.9}$$

A two-stage approximation to the function f can now be attained. The first stage consists in approximating $f(\mathbf{x}) \approx f_\Omega(\mathbf{x})$, the second stage in obtaining an estimate of the integral (6.8) by using the Monte-Carlo method: $f_\Omega(\mathbf{x}) \approx \tilde{f}_n(\mathbf{x})$, where

$$\tilde{f}_n(\mathbf{x}) := \frac{|V|}{n} \sum_{i=1}^{n} T(\mathbf{u}_i, b_i) g(\mathbf{u}_i^\dagger \mathbf{x} - b_i) \tag{6.10}$$

and $\{(\mathbf{u}_1, b_1), \ldots, (\mathbf{u}_n, b_n)\}$ is a sample of size n drawn independently from the uniform distribution

$$P(\mathbf{u}_i, b_i) = \begin{cases} |V|^{-1} & \text{if } (\mathbf{u}_i, b_i) \in V \\ 0 & \text{otherwise} \end{cases} \tag{6.11}$$

If we define

$$w_i := \frac{|V|}{n} T(\mathbf{u}_i, b_i), \tag{6.12}$$

then

$$\tilde{f}_n(\mathbf{x}) = \sum_{i=1}^{n} w_i\, g(\mathbf{u}_i^\dagger \mathbf{x} - b_i) \tag{6.13}$$

has exactly the form of an RFVL network, where the hidden weights $\mathbf{u}_i$ and thresholds b_i are independently chosen subject to the probability distribution (6.11), and the output weights w_i are optimised in the training process. It will now be shown that

$$\left\langle \tilde{f}_n(\mathbf{x}) \right\rangle = f_\Omega(\mathbf{x}), \tag{6.14}$$

where $\langle . \rangle$ denotes the expectation value with respect to the probability distribution (6.11), and that the distance between f_Ω and $\tilde{f}_n$, defined as

$$d_K[f_\Omega, \tilde{f}_n] := \sqrt{\frac{1}{|K|} \left\langle \int_K \left[f_\Omega(\mathbf{x}) - \tilde{f}_n(\mathbf{x}) \right]^2 d\mathbf{x} \right\rangle} \tag{6.15}$$

(where $|K|$ denotes the volume of the compact set K) scales as $d_K \propto \frac{1}{\sqrt{n}}$. With definition (6.10) one obtains

$$\left\langle \tilde{f}_n(\mathbf{x}) \right\rangle = \left\langle \sum_{i=1}^{n} \frac{T(\mathbf{u}_i, b_i)|V|}{n} g(\mathbf{u}_i^\dagger \mathbf{x} - b_i) \right\rangle \tag{6.16}$$

Since the $\mathbf{u}_i$ and b_i are drawn independently from the probability distribution (6.11), this gives

$$\begin{aligned}
\left\langle \tilde{f}_n(\mathbf{x}) \right\rangle &= n \left\langle \frac{T(\mathbf{u}, b)|V|}{n} g(\mathbf{u}^\dagger \mathbf{x} - b) \right\rangle = \left\langle T(\mathbf{u}, b)|V|\, g(\mathbf{u}^\dagger \mathbf{x} - b) \right\rangle \\
&= \int T(\mathbf{u}, b)|V|\, g(\mathbf{u}^\dagger \mathbf{x} - b) P(\mathbf{u}, b)\, d\mathbf{u}\, db \\
&= \int_V T(\mathbf{u}, b)|V|\, g(\mathbf{u}^\dagger \mathbf{x} - b) \frac{1}{|V|}\, d\mathbf{u}\, db \\
&= \int_V T(\mathbf{u}, b) g(\mathbf{u}^\dagger \mathbf{x} - b)\, d\mathbf{u}\, db = f_\Omega(\mathbf{x})
\end{aligned} \tag{6.17}$$

in which the last step follows from (6.8). In the same way, the variance is obtained as

$$\begin{aligned}
\mathrm{Var}[\tilde{f}_n(\mathbf{x})] &= \mathrm{Var}\left[\sum_i \frac{T(\mathbf{u}_i, b_i)|V|}{n} g(\mathbf{u}_i^\dagger \mathbf{x} - b_i) \right] \\
&= n\, \mathrm{Var}\left[\frac{T(\mathbf{u}, b)|V|}{n} g(\mathbf{u}^\dagger \mathbf{x} - b) \right] \\
&= \frac{1}{n} \mathrm{Var}\left[T(\mathbf{u}, b)|V|\, g(\mathbf{u}^\dagger \mathbf{x} - b) \right] \\
&= \frac{1}{n} \left\{ \left\langle \left(T(\mathbf{u}, b)|V|\, g(\mathbf{u}^\dagger \mathbf{x} - b) \right)^2 \right\rangle - \left\langle T(\mathbf{u}, b)|V|\, g(\mathbf{u}^\dagger \mathbf{x} - b) \right\rangle^2 \right\} \\
&= \frac{1}{n} \left\{ \left\langle \left(T(\mathbf{u}, b)|V|\, g(\mathbf{u}^\dagger \mathbf{x} - b) \right)^2 \right\rangle - \left[f_\Omega(\mathbf{x}) \right]^2 \right\}
\end{aligned}$$

where the last step follows from (6.17). The first term can be written as

$$\begin{aligned}
\left\langle \left(T(\mathbf{u}, b)|V|\, g(\mathbf{u}^\dagger \mathbf{x} - b) \right)^2 \right\rangle &= \int \left(T(\mathbf{u}, b)|V|\, g(\mathbf{u}^\dagger \mathbf{x} - b) \right)^2 P(\mathbf{u}, b)\, d\mathbf{u}\, db \\
&= \int_V \left(T(\mathbf{u}, b)|V|\, g(\mathbf{u}^\dagger \mathbf{x} - b) \right)^2 \frac{1}{|V|}\, d\mathbf{u}\, db \\
&= |V| \int_V \left(T(\mathbf{u}, b)\, g(\mathbf{u}^\dagger \mathbf{x} - b) \right)^2 d\mathbf{u}\, db
\end{aligned}$$

Inserting this expression into the above equation for the variance yields

$$\mathrm{Var}[\tilde{f}_n(\mathbf{x})] = \frac{1}{n} \left\{ |V| \int_V \left(T(\mathbf{u}, b)\, g(\mathbf{u}^\dagger \mathbf{x} - b) \right)^2 d\mathbf{u}\, db - \left[f_\Omega(\mathbf{x}) \right]^2 \right\}. \tag{6.18}$$

Inserting (6.14) into the expression for the distance between f_Ω and $\tilde{f}_n$, Equation (6.15), gives

$$d_K^2[f_\Omega, \tilde{f}_n] = \frac{1}{|K|}\left\langle \int_K \left[f_\Omega(\mathbf{x}) - \tilde{f}_n(\mathbf{x})\right]^2 d\mathbf{x}\right\rangle = \frac{1}{|K|}\int_K \left\langle \left[f_\Omega(\mathbf{x}) - \tilde{f}_n(\mathbf{x})\right]^2 \right\rangle d\mathbf{x}$$

$$= \frac{1}{|K|}\int_K \left\langle \left[\left\langle \tilde{f}_n(\mathbf{x})\right\rangle - \tilde{f}_n(\mathbf{x})\right]^2 \right\rangle d\mathbf{x} = \frac{1}{|K|}\int_K \mathrm{Var}[\tilde{f}_n(\mathbf{x})]d\mathbf{x} \quad (6.19)$$

We can now substitute the expression for the variance, equation (6.18), to obtain

$$d_K^2[f_\Omega, \tilde{f}_n] = \frac{1}{n|K|}\int_K d\mathbf{x}\left\{|V|\int_V \left(T(\mathbf{u},b)\, g(\mathbf{u}^\dagger\mathbf{x} - b)\right)^2 d\mathbf{u}\, db - \left[f_\Omega(\mathbf{x})\right]^2\right\} \quad (6.20)$$

Since $|g(x)| \leq 1$, this can be bounded by

$$\begin{aligned} d_K^2[f_\Omega, \tilde{f}_n] &\leq \frac{1}{n|K|}\int_K d\mathbf{x}\left\{|V|\int_V \left(T(\mathbf{u},b)\, g(\mathbf{u}^\dagger\mathbf{x} - b)\right)^2 d\mathbf{u}\, db\right\} \\ &\leq \frac{|V|}{n|K|}\int_K d\mathbf{x}\int_V T^2(\mathbf{u},b)\, d\mathbf{u}\, db = \frac{|V||K|}{n|K|}\int_V T^2(\mathbf{u},b)\, d\mathbf{u}\, db \end{aligned}$$

and hence

$$d_K[f_\Omega, \tilde{f}_n] \leq \frac{C}{\sqrt{n}} \quad (6.21)$$

where

$$C := \sqrt{|V|\int_V T^2(\mathbf{u},b)\, d\mathbf{u}\, db} = \sqrt{(2\Omega)^{m+1}\int_V T^2(\mathbf{u},b)\, d\mathbf{u}\, db} \quad (6.22)$$

Let us now consider the distance between $\tilde{f}_n$ and the actual function f, which can be bounded by

$$d_K[f, \tilde{f}_n] \leq \sup_{\mathbf{x}\in K} |f(\mathbf{x}) - f_\Omega(\mathbf{x})| + d_K[f_\Omega, \tilde{f}_n]. \quad (6.23)$$

Because of (6.9), the first term can be made arbitrarily small by choosing large enough values for the parameter Ω. In this case, however, the second term becomes very large and tends to infinity as $\Omega \to \infty$ (as seen from (6.21)). It is therefore necessary to restrict the class of continuous functions f to the class of smoother functions satisfying the *Lipshitz* condition

$$\exists \kappa > 0: \quad |f(\mathbf{x}) - f(\mathbf{y})| \leq \kappa \sum_{i=1}^{m} |x_i - y_i| \quad \forall\, \mathbf{x}, \mathbf{y} \in K. \quad (6.24)$$

Then the first term on the right of (6.23) becomes negligibly small for finite Ω (see [32]) so that (6.21) can be re-written as

$$d_K[f, \tilde{f}_n] \leq \frac{C}{\sqrt{n}}; \qquad C = \sqrt{(2\Omega)^{m+1} \int_V T^2(\mathbf{u}, b)\, d\mathbf{u}\, db}; \qquad \Omega < \infty \tag{6.25}$$

In practical applications the parameter Ω, which determines the distribution of the random parameters $\mathbf{u}_i$ and b_i, needs to be optimised in the training process. This will be discussed in more detail in Chapter 8.

6.3 Comparison with the multilayer perceptron

In order to provide additional insight into the universal approximation capability of the RVFL, it is useful to look over the corresponding proof for the universal approximation capability of the standard multilayer perceptron (MLP), as given e.g. by Murata [43]. First define

$$\mathcal{N} := \int_{\mathbb{R}_{m+1}} |T(\mathbf{u}, b)|\, d\mathbf{u}\, db, \tag{6.26}$$

where the integral is assumed to be finite[3], and rewrite the integral representation (6.5) in the form

$$f(\mathbf{x}) = \int_{\mathbb{R}^{m+1}} \operatorname{sign}\big[T(\mathbf{u}, b)\big] |T(\mathbf{u}, b)| \frac{\mathcal{N}}{\mathcal{N}} g(\mathbf{u}^\dagger \mathbf{x} - b)\, d\mathbf{u}\, db. \tag{6.27}$$

With the definition of the probability distribution

$$P(\mathbf{u}, b) := \frac{1}{\mathcal{N}} |T(\mathbf{u}, b)|, \tag{6.28}$$

equation (6.27) leads to

$$f(\mathbf{x}) = \mathcal{N} \int_{\mathbb{R}^{m+1}} \operatorname{sign}\big[T(\mathbf{u}, b)\big]\, g(\mathbf{u}^\dagger \mathbf{x} - b)\, P(\mathbf{u}, b)\, d\mathbf{u}\, db. \tag{6.29}$$

Similar to (6.10), this integral can be approximated with the Monte-Carlo method $f(\mathbf{x}) \approx \tilde{f}_n(\mathbf{x})$, where

$$\tilde{f}_n(\mathbf{x}) := \frac{\mathcal{N}}{n} \sum_{i=1}^{n} \operatorname{sign}\big[T(\mathbf{u}_i, b_i)\big] g(\mathbf{u}_i^\dagger \mathbf{x} - b_i) \tag{6.30}$$

and the $\{(\mathbf{u}_1, b_1), \ldots, (\mathbf{u}_n, b_n)\}$ are independently chosen subject to the probability distribution (6.28). Define

$$w_i := \frac{\mathcal{N}}{n} \operatorname{sign}\big[T(\mathbf{u}_i, b_i)\big], \tag{6.31}$$

[3] See [43] for a discussion of the divergent case.

then

$$\tilde{f}_n(\mathbf{x}) \quad = \quad \sum_{i=1}^{n} w_i \, g(\mathbf{u}_i^\dagger \mathbf{x} - b_i) \tag{6.32}$$

has a form similar to (6.13), where the w_i can be interpreted as output weights, the $\mathbf{u}_i$ as hidden weights, and the b_i as hidden thresholds. However, whereas in (6.13) the $\mathbf{u}_i$ and b_i are drawn *randomly* from the domain V (RVFL), the probability distribution considered here includes information about the unknown function f via $T(\mathbf{u}, b)$, as seen from (6.6) and (6.28), and thus needs to be learned in the training process (MLP). In other words, whereas the RVFL uses a simple mechanism for random selection of the parameters $\mathbf{u}_i$, b_i (uniform distribution) and includes all information about the unknown function f in the output weights to be learned, w_i, the MLP includes further information about f in the mechanism of random selection. This corresponds to the variance reduction method of *importance sampling* (see [32], [51], pp.316-317) and allows a closer approximation of the integral (6.5). In fact, repeating the steps (6.16), (6.18), and (6.19) for the distribution defined in (6.28) leads to (see [43])

$$d_K[f, \tilde{f}_n] \leq \frac{C_{MLP}}{\sqrt{n}}; \qquad C_{MLP} = \int_{\mathbb{R}^{m+1}} |T(\mathbf{u}, b)| \, d\mathbf{u} \, db. \tag{6.33}$$

Since it is assumed that the integral on the right is finite, it is valid for sufficiently large Ω to make the approximation

$$C_{MLP} \approx \int_V |T(\mathbf{u}, b)| \, d\mathbf{u} \, db = \frac{|V|}{|V|} \int_V |T(\mathbf{u}, b)| \, d\mathbf{u} \, db = |V| \Big\langle |T(\mathbf{u}, b)| \Big\rangle \tag{6.34}$$

where the expectation value $\langle . \rangle$ is taken with respect to the uniform distribution defined in (6.11). Compare this with the constant C defined in (6.21), which will be denoted C_{RVFL} henceforth,

$$C_{RVFL}^2 = |V| \int_V T^2(\mathbf{u}, b) \, d\mathbf{u} \, db = |V|^2 \frac{1}{|V|} \int_V T^2(\mathbf{u}, b) \, d\mathbf{u} \, db = |V|^2 \Big\langle T^2(\mathbf{u}, b) \Big\rangle \tag{6.35}$$

(where again the expectaion value is taken with respect to the uniform distribution (6.11)). Combining (6.34) and (6.35) gives

$$C_{RVFL}^2 - C_{MLP}^2 = |V|^2 \left(\Big\langle T^2(\mathbf{u}, b) \Big\rangle - \Big\langle |T(\mathbf{u}, b)| \Big\rangle^2 \right) = |V|^2 \mathrm{Var}|T(\mathbf{u}, b)| \geq 0 \tag{6.36}$$

and hence

$$C_{RVFL} \geq C_{MLP}. \tag{6.37}$$

Consequently, for a given value of n, the MLP gives a closer approximation to f than the RVFL, as expected. However, the functional dependence of the approximation error on the complexity of the model is in both cases of the form $\propto 1/\sqrt{n}$, so both models are *efficient* universal approximators that avoid an exponential increase in the number of hidden units.

6.4 A simple illustration

It was proved in the previous section that the RVFL network, as distinct from a network with fixed basis functions, is an efficient universal approximator that avoids the curse of dimensionality. The purpose of this section is to give a simple, plausible illustration for this fact. Consider the problem of determining the volume of an m-dimensional (hyper-)sphere by numerical integration. First, define the binary indicator function

$$I(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad ||\mathbf{x}|| \leq R \\ 0 & \text{if} \quad ||\mathbf{x}|| > R \end{cases} \tag{6.38}$$

where R is the radius of the sphere. The true volume is known (see e.g. [7], pp. 28-29) to be

$$V_{Sphere} = \int_{\mathbb{R}^m} I(\mathbf{x}) d^m\mathbf{x} = \frac{2\pi^{m/2}R^m}{m\Gamma(m/2)}. \tag{6.39}$$

A standard numerical integration approximates the integral by a simple Riemann sum. That is, first introduce a hypercube $K = [-R, R]^m$ of side $L = 2R$ and divide each side into k equi-distant fragments of length $l = \frac{L}{k}$. This leads to a division of the whole space into a grid of $n = k^m$ equally-sized cells or sub-cubes with associated volume

$$V_1 = \ldots V_n = l^m = \left(\frac{L}{k}\right)^m = \frac{L^m}{n} = \frac{(2R)^m}{n}. \tag{6.40}$$

Next, define $\mathbf{x}_i$ to be the centre of the ith sub-cube and approximate the integral (6.38) by

$$V_{Sphere}^{Grid} = \sum_{i=1}^{n} I(\mathbf{x}_i)V_i = \frac{(2R)^m}{n}\sum_{i=1}^{n} I(\mathbf{x}_i). \tag{6.41}$$

An alternative method is the Monte-Carlo technique. Here we introduce the uniform distribution

$$P(\mathbf{x}_i) = \begin{cases} |K|^{-1} = (2R)^{-m} & \text{if} \quad \mathbf{x}_i \in K \\ 0 \quad \text{otherwise} \end{cases} \tag{6.42}$$

The Monte-Carlo approximation to (6.39) is given by

$$V_{Sphere}^{MC} = \frac{|K|}{n}\sum_{i=1}^{n} I(\mathbf{x}_i) = \frac{(2R)^m}{n}\sum_{i=1}^{n} I(\mathbf{x}_i), \tag{6.43}$$

where the $\mathbf{x}_i$ are selected independently from the hypercube according to the distribution (6.42). Note that the essential difference between equations

(6.41) and (6.43), which formally look identical, is the different choice of the vectors $\mathbf{x}_i$.

The first method can be likened to a neural network with fixed basis functions, with each cell or sub-cube corresponding to one particular basis function. A sufficient approximation accuracy requires cells with small side $l = L/k$, and hence large values of k. On the other hand, the total number of cells, n, is given by $n = k^m$. This leads to an exponential increase in the number of cells with the input dimension m.

The second method is akin to the RVFL approach. As the $\mathbf{x}_i$ are chosen at random rather than systematically, the integration space K can be explored with a much smaller number of 'basis functions' n. The approximation error can be estimated by

$$\epsilon \approx \frac{C}{\sqrt{n}} \tag{6.44}$$

where

$$C = (2R)^m \sqrt{\langle I^2 \rangle - \langle I \rangle^2}, \quad \langle I^2 \rangle = \frac{1}{n} \sum_{i=1}^{n} I^2(\mathbf{x}_i), \quad \langle I \rangle = \frac{1}{n} \sum_{i=1}^{n} I(\mathbf{x}_i) \tag{6.45}$$

(see e.g. [51], p.305). Its functional dependence on n is thus independent of m (i.e. m does not occur in the exponent of n, but only enters indirectly via C), which avoids the curse of dimensionality.

Table 6.1 contrasts the relative approximation error $\varepsilon = \frac{100|V_{predict} - V_{Sphere}|}{V_{Sphere}}$ for the two methods, where $V_{predict}$ is the predicted volume of the sphere, given either by (6.41) or by (6.43), and V_{Sphere} is the true volume (6.39). The computations were repeated for different dimensions of the hyper-sphere, m, and different numbers of 'basis functions', n. In each case, the Monte-Carlo scheme was repeated ten times (starting from different random number generator seeds), and the mean error was determined. It can be seen that the errors obtained with the Monte-Carlo method are significantly smaller than for the grid method, especially in the case of the large dimension $m = 10$.

6.5 Summary

The proof of the universal approximation capability for both the MLP and the RVFL is based on an integral representation of the function to be approximated and subsequent evaluation of the integral by the Monte-Carlo approach. In both cases the approximation error is of the order $\frac{C}{\sqrt{n}}$, where n is the number of nodes in the hidden layer, and C a constant independent of n. Hence it follows that both the MLP and the RVFL are free of the *curse*

dimension	number of kernels	Grid: relative error	MC: relative error
3	27	34.4	12.8
3	125	23.8	7.8
3	1000	5.4	2.1
10	1024	100.0	38.5
10	59049	36.7	4.9

Table 6.1. Simple Monte Carlo. The table shows, for the simple case of determining the volume of an m-dimensional sphere, the dependence of the relative error, $\varepsilon = \frac{100|V_{predict} - V_{Sphere}|}{V_{Sphere}}$ (where $V_{predict}$ is the predicted volume and V_{Sphere} the correct one), on the number of 'kernels' or 'basis functions' for the grid and the Monte Carlo (MC) method.

of dimensionality, as opposed to a network with fixed basis functions. The difference between the RVFL and the MLP is that the former approximates the integral with a *simple* Monte-Carlo method, whereas the latter follows the more sophisticated scheme of *importance sampling* for variance reduction. The advantage of the MLP is therefore the possibility of a closer approximation of f with the same model complexity. Its disadvantage, however, is the non-convexity of the optimisation scheme, which renders the training process slow and susceptible to local minima.

This chapter has been concerned with learning in a conventional neural network for point predictions. The following chapter will show how the RVFL approach can be applied to GM networks for predicting conditional probabilities. The first section will review the EM algorithm, a powerful parameter adaptation scheme known from statistics. It will be shown, however, that its direct application to the GM model is not immediately possible, and that only in combination with the RVFL approach can a significant speed-up of the training scheme be achieved.