Institute of Computer Science
Academy of Sciences of the Czech Republic

Pruning algorithms for one-hidden-layer feedforward neural networks

Terezie Šámalová, Robert Šámal

Technical report No. 1022

July 2008

Pod Vodárenskou věží 2, 182 07 Prague 8, phone: +420 266 051 111, fax: +420 286 585 789,
e-mail:ics@cs.cas.cz

Institute of Computer Science
Academy of Sciences of the Czech Republic

Pruning algorithms for one-hidden-layer feedforward neural networks

Terezie Šámalová, Robert Šámal

Technical report No. 1022

July 2008

Abstract:

In the article we propose randomized algoritms based on a modification of probabilistic proof of theorem on rates of approximation in convex closures from [DDGS93]. On high-level we derive time complexities of the proposed algorithms and compare them to iterative algorithms in several concrete situations. Observing properties of one type of probabilistic algorithm we propose it as a solution for pruning a too big neural network. We derive theoretically bounds on error such introduced.

Too big networks arise for example from regularized minimization problems. We overview kernel-based regularization and propose use of the probabilistic algorithm. Further we discuss new types of composite kernels and their advantages. Experiments on use of probabilistic pruning on these schemas are running.

Keywords:
probabilistic algorithm, randomized pruning, error estimate, time complexity, regularization, kernel methods

Along the Barron-type investigation there are two established ways to estimate rates of approximation in Hilbert (and more generally Banach) spaces. The first one (pioneered by Jones [Jo92] and Barron [Ba93], and generalized to Banach spaces by Darken et al. [DDGS93]) analyzes the process of iteratively improving the approximating function. The intrinsic feature of this approach is, that to get the n-th approximating function, f_n, we first compute f_0, f_1, ..., f_{n-1}. This makes each of the iterative steps easier to manage, but it also may constitute unnecessary work.

The other approach (established by Maurey [Ps81], and also pursued by Barron [Ba93], Darken et al. [DDGS93], and Makovoz [Mk96]) is nonconstructive: it is possible to show that there exists a good approximant using the so-called probabilistic method; that is in appropriately chosen probability space, the probability of getting a good approximant is nonzero.

In first part of this article we extend this second approach – we suggest a randomized algorithm and compare it to the iterative approach. To this end, we also analyze complexity of the iterative algorithm, a task, which seems not to be addressed in existing literature.

By its nature the randomized algorithm is very well suited to prune networks with too many units to obtain smaller ones with still good approximation abilities (probability estimates on error with respect to error of original big network can be expressed). We propose how to use the randomized algorithm to prune any kernel-method attained network to obtain reasonable number of hidden units. Experiments towards this end are running in cooperation with Petra Kudov-Vidnerov on kernel-based neural network schemas.

We start by analyzing the iterative algorithm in Section 1. In Section 2 we propose and study the randomized algorithm. In Section 3 we briefly discuss the complexity of various "subprocedures" used in the algorithms under study. Finally, in Section 4 we compare the suggested algorithms in various situations. In Section 5 we briefly introduce theory of reproducing kernel Hilbert spaces (RKHS) following [Ar50]. This is utilized in further sections: In Section 6 learning from data using Tikhonov regularization employing kernels is presented. As has been mentioned in various detail in many works existence, uniqueness and form of solution to the basic problem can be proven. In Section 7 we present how to use the randomized algorithm to yield approximations that use less terms than the above derived one. Section 8 shows applications of some concrete regularization networks, part of the section comes from joint work with Petra Kudov-Vidnerov [KS05b], in cooperation with her we are also working on experiments employing randomized algorithm to these specific schemas.

1 Iterative Algorithms

In this section we analyze time complexity of the algorithm suggested by Jones [Jo92] and Barron [Ba93], and further analyzed by Darken et al. [DDGS93]. In these papers, the authors concentrate on the estimate of approximating abilities of some approximation schemas (in particular neural networks), without addressing the amount of computations needed to achieve such approximation. In this section we try to clarify this point. We start by specifying the algorithm under study – or, more precisely, the algorithms, as there are (at least) two distinct ways to organize the computations. Then we do a "high-level" estimate of complexity, postponing the discussion of how some standard tasks can be implemented to Section 3.

Let us present basic assumptions. We consider approximations of a function f using convex combinations of functions from $\mathcal{G}$, a subset of some function space X (which is a Banach space). We are given n, the upper bound on the number of elements of $\mathcal{G}$ we are allowed to use. The key idea is to use the following iterative transformation.

$$f_0 = 0$$
$$f_{k+1} = \alpha f_k + (1-\alpha) g_k, \qquad g_k \in \mathcal{G}, \alpha \in [0,1] \tag{1.1}$$

We will use a result that appears as Theorem 3.5 in [DDGS93]:

Theorem 1.1 (Iterative rates in smooth Banach spaces [DDGS93]) *Let X be a uniformly smooth Banach space having modulus of smoothness $\rho(u) \le \gamma u^t$ with $t > 1$. Let $\mathcal{G}$ be a bounded subset of X and let $f \in \operatorname{cl}\operatorname{conv}\mathcal{G}$ be given. Select $\rho > 0$ such that $\|f - g\| \le \rho$ for all $g \in \mathcal{G}$ and fix $\eta > 0$. We choose sequences $\{f_k\} \subset \operatorname{conv}\mathcal{G}$ and $\{g_k\} \subset \mathcal{G}$ recursively so that*

1. $f_1 \in \mathcal{G}$
2. $F_k(g_k - f) \le \frac{2\gamma}{k^{t-1}\|f_k-f\|^{t-1}}\left((\rho+\eta)^t - \rho^t\right) =: \delta_k$
3. $f_{k+1} = \frac{k}{k+1} f_k + \frac{1}{k+1} g_k$,

where F_k is peak functional for $f_k - f$ (we terminate the procedure if $f_k = f$). Then we have

$$\|f_k - f\| \le \frac{(2\gamma t)^{1/t}(\rho+\eta)}{k^{1-1/t}}\big(1 + \frac{(t-1)\log_2 k}{2tk}\big)^{1/t}. \tag{1.2}$$

Note that the error bound presented here holds in every step and thus also for an n given in advance by the "enemy". There are two natural ways to find an appropriate function g_k in (1.1).

1. Using repeated optimization. The first approach is to use (1.1) as a way to reduce dimensionality of an optimization problem (it was apparently meant so by Jones, who originally proposed this method). That is, instead of choosing all functions to form the approximation at the same time, we only decide about one function at a time. This amounts to n times solving a (multidimensional) optimization problem, to find good enough g_k: in view of Theorem 1.1 this means to find g_k for which $F_k(g_k - f) \le \delta_k$. Here F_k is the peak functional for $f_k - f$, that is a linear continuous functional of unit norm for which $F_k(f_k - f) = \|f_k - f\|$. We denote the time needed for finding such g_k by $t_{opt}(\mathcal{G}, \delta_k)$. We will discuss t_{opt} further in Section 3.2. In (1.3) a lower bound on δ_k is presented; it yields $\delta_k = \Omega(1/k^{1-1/t})$. Based on these considerations we have the following algorithm and time analysis.

Algorithm 1.2 (Iterative from scratch)

Given: f, $\mathcal{G}$, n

```
1. $f_0 = 0$
2. For $k = 0$ to $n-1$:
    • Let $F_k$ be a peak functional for $f_k - f$.
    • Find $g_k \in \mathcal{G}$ so that $F_k(g_k - f) < \delta_k$, where $\delta_k$ is as in Theorem 1.1.
    • Put $f_{k+1} = \frac{k}{k+1} f_k + \frac{1}{k} g_k$
```

Output: f_n
Time: $\sum_{k=1}^{n-1}(t_{opt}(\mathcal{G},\delta_k) + O(1)) = O(n \cdot t_{opt}(\mathcal{G}, \Omega(\frac{1}{n^{1-1/t}})))$

2. Using an auxiliary approximation. The second approach (inspired by the proofs in [Jo92, Ba93, DDGS93]) is to first find an auxiliary approximation of f, possibly using a large number of terms: For $i = 1, \dots, N$ we find $h_i \in \mathcal{G}$ and c_i ($c_i \geq 0$, $\sum_{i=1}^N c_i = 1$) and put

$$f_{apx} = \sum_{i=1}^{N} c_i h_i \,.$$

We let ε_{apx} denote the error of this approximation, that is $\varepsilon_{apx} = \|f - f_{apx}\|$. We will discuss the possibilities to obtain such approximation later, in Section 3.3. Meanwhile, we just let $t_{apx}(\mathcal{G}, \varepsilon_{apx}, N)$ denote the time needed to find such f_{apx}. Next, in each iteration of (1.1) we choose for g_k one of the functions h_i, $i = 1, \dots, N$. We will do so to make $F_k(g_k - f) < \delta_k$. Let us pause for a while to analyze when is this possible. Due to linearity of F_k we have

$$F_k(g_k - f) = F_k(g_k - f_{apx}) + F_k(f_{apx} - f)\,.$$

The second term is at most $\|F_k\| \cdot \|f_{apx} - f\| = \varepsilon_{apx}$. The first term is "zero in the average":

$$\sum_{i=1}^{N} c_i F_k(h_i - f_{apx}) = F_k(\sum_{i=1}^{N} c_i h_i) - \sum_{i=1}^{N} c_i F_k(f_{apx}) = 0\,.$$

Consequently, there is at least one h_i for which $F_k(h_i - f_{apx}) \leq 0$, therefore $F_k(h_i - f) \leq \varepsilon_{apx}$. We will let g_k be this h_i (or one of them, if there is more than one such h_i).

Theorem 1.1 states that we need $F_k(g_k - f) \leq \delta_k$, thus it is sufficient to ensure $\varepsilon_{apx} \leq \delta_k$. As we do not know in advance the size of $\|f - f_k\|$, we need a lower bound on δ_k, that can be evaluated before we start the algorithm. Using the estimate $(\rho + \eta)^t - \rho^t \geq \eta t \rho^{t-1}$ for $t > 1$ and the inductive bound on $\|f - f_k\|$, we get that

$$\begin{aligned}\delta_k &\geq 2\gamma t \eta \left(\frac{\rho}{k \frac{(2\gamma t)^{1/t}(\rho+\eta)\left(1+\frac{(t-1)\log_2 k}{2tk}\right)^{1/t}}{k^{1-1/t}}} \right)^{t-1} \\ &= (2\gamma t)^{1/t} \frac{\eta}{k^{1-1/t}} \big(\frac{\rho}{\rho+\eta}\big)^{t-1} \big(1 + \frac{(t-1)\log_2 k}{2tk}\big)^{1/t-1} =: d_k\,. \qquad (1.3)\end{aligned}$$

Thus, we will choose f_{apx} such as to make $\varepsilon_{apx} \leq d_{n-1}$. Hence the assumptions of Theorem 1.1 will be satisfied, and so we get approximation f_n as guaranteed by this theorem.

Algorithm 1.3 (Iterative from auxiliary approximation)

Given: f, $\mathcal{G}$, n

1. *Choose $\eta > 0$ (smaller η yields better approximation, as given by Theorem 1.1). Let the modulus of smoothness of the given space be $\varrho(u) \leq \gamma u^t$.*
2. *Put $\varepsilon_{apx} = (2\gamma t)^{1/t} \frac{\eta}{(n-1)^{1-1/t}} \big(\frac{\rho}{\rho+\eta}\big)^{t-1} \big(1 + \frac{(t-1)\log_2(n-1)}{2t(n-1)}\big)^{1/t-1}$*
3. *For $i = 1, \dots, N$ find functions $h_i \in \mathcal{G}$, and coefficients $c_i \geq 0$ with $\sum_{i=1}^N c_i = 1$ so that*

$$\big|f - \sum_{i=1}^{N} c_i h_i\big| < \varepsilon_{apx}\,.$$

4. Put $f_0 = 0$.

5. For k from 0 to $n-1$:

- *Find $i \in \{1, \dots, N\}$, so that $F_k(h_i - f) \leq \delta_k$, where δ_k is as in Theorem 1.1.*
- *Put $g_k = h_i$.*
- *Put $f_{k+1} = \frac{k}{k+1} f_k + \frac{1}{k+1} g_k$.*

Output: *f_n satisfying the estimate* (1.2)
Time: $t_{apx}(\mathcal{G}, \varepsilon_{apx}, N) + nNt_{int}$

Two remarks are in place.

How to compute peak functionals In a general Banach space, the existence of a peak functional for every nonzero element is only guaranteed nonconstructively, by means of Hahn-Banach theorem. In many concrete spaces, however, peak functionals are given by a simple integral formula (which is why we denote time to evaluate peak functionals by t_{int}, the time needed to enumerate an integral). For instance, in a Hilbert space, peak functional corresponding to f is given by scalar product with $f/\|f\|$. More generally, if f is an element of the $\mathcal{L}^p$ space, then we may put $\psi(t) = sgn(f(t)) \frac{|f(t)|^{p-1}}{\|f\|^{p-1}}$. It is easy to verify that F given by $F(g) = \int g(t)\psi(t)\,\mathrm{d}t$ is a peak functional corresponding to f.

An improvement for Hilbert spaces In Hilbert spaces we may utilize the simple form of the peak functionals to improve time complexity of the above algorithm. A simple way to calculate the required peak functionals (that is, scalar products) is to compute N scalar products at each iteration. This requires time nNt_{int}; we can do better, though.

Before we start our iteration process, we can compute the N scalar products $\langle h_i, f\rangle$ and also $\langle f, f\rangle$. We also store all the scalar products ($\langle h_i, f_k\rangle$ for $i = 1, \dots, N$ and for the "current value" of k. After each step, when f_{k+1} is found using (1.1), we recalculate these scalar products. We assume that all scalar products $\langle h_i, h_j\rangle$ are easy to compute, or precomputed, so the computation of $\langle g_i, f_k\rangle$ only takes $O(N)$ operations with numbers, but no scalar products. In this way, all the scalar products are computed in time $(N+1)t_{int} + O(nN)$.

2 Randomized Algorithms

In this section we are going to follow closely the proof of Theorem 2.1 as it appears in Darken et al. [DDGS93]. At the end of it, rather than using the basic observation "there is always something at least as good as the average", we use the Markov inequality. This extra step has useful algorithmic consequences, which we state as Corollary 2.3.

Recall that $\mathcal{L}^p$ spaces for $p \geq 1$ are of Rademacher type $\min\{p, 2\}$.

Theorem 2.1 (Probabilistic estimate of error in Banach spaces (based on [DDGS93])) *Let X be Banach space of Rademacher type t, $1 \leq t \leq 2$. Suppose $\mathcal{G} \subset X$, $f \in \operatorname{cl}\operatorname{conv}\mathcal{G}$ and let $\rho > 0$ be such that for every $g \in \mathcal{G}$ we have $\|g - f\| \leq \rho$.*

Assume we have an approximation of f of the form

$$f_{apx} = \sum_{i=1}^{N} c_i h_i, \qquad \sum_{i=1}^{N} c_i = 1, \quad c_i \geq 0, \quad h_i \in \mathcal{G}$$

so that $\|f - f_{apx}\| < \varepsilon$. We pick $g_1, \ldots, g_n$ from $h_1, \ldots, h_N$ independently at random, so that $\Pr[g_j = h_i] = c_i$ for each i, j.

Then the average

$$g_{apx} = \tfrac{1}{n} \sum_{i=1}^{n} g_i$$

is (with high probability) a reasonable approximation, namely

$$\Pr[\|g_{apx} - f\| > Q^{1/t} \cdot \big(\frac{C(\rho + \varepsilon)}{n^{1-1/t}} + \varepsilon\big)] < \frac{1}{Q}.$$

Here C is a constant depending on X but independent of n, and $Q > 1$ is a parameter specifying the trade-off between quality of approximation and time needed to find it.

Proof: We follow the proof of [DDGS93] altering it slightly at the end of the proof. Put $\Delta = f - f_{apx}$, so that $\|\Delta\| < \varepsilon$.

Take g_j to be a sequence of independent random variables on X every one of them taking value h_i with probability c_i. Then for any $\beta \in (0, 1)$

$$\begin{aligned} E\left\| f - \frac{1}{n}\sum_{j=1}^{n} g_j \right\|^t &= \frac{1}{n^t} E \left\| \sum_{j=1}^{n} (f - g_j - \Delta) + n\Delta \right\|^t \\ &\leq \frac{1}{n^t} E \left((1-\beta) \frac{\| \sum_{j=1}^{n} (f - g_j - \Delta) \|}{1-\beta} + \beta n \frac{\|\Delta\|}{\beta} \right)^t \\ &\leq \frac{1}{n^t} \left[(1-\beta) E \left(\frac{\| \sum_{j=1}^{n} (f - g_j - \Delta) \|}{1-\beta} \right)^t + \beta \left(\frac{n\|\Delta\|}{\beta} \right)^t \right] \\ &= \frac{1}{n^t (1-\beta)^{t-1}} E \left\| \sum_{j=1}^{n} (f - g_j - \Delta) \right\|^t + \frac{1}{\beta^{t-1}} \|\Delta\|^t \end{aligned} \tag{2.1}$$

This is true since $x \mapsto x^t$ is a convex function on $\mathbb{R}^+$ for $1 \leq t \leq 2$. Since the range of g_j has finitely many values, g_j is a Radon random variable. As the Rademacher type of X is t, by Proposition 9.11 in [LeTa91] we have that there is a constant $C = C(X)$ so that

$$E \left\| \sum_{j=1}^{n} (f - g_j - \Delta) \right\|^t \leq C^t \sum_{j=1}^{n} E \|f - g_j - \Delta\|^t. \tag{2.2}$$

It is also true that for every j

$$\begin{aligned} E\|f-g_j-\Delta\|^t &= \sum_{i=1}^{N} c_i\|f-h_i-\Delta\|^t \\ &\le \sum_{i=1}^{N} c_i(\|f-h_i\|+\|\Delta\|)^t \\ &< \sum_{i=1}^{N} c_i(\rho+\varepsilon)^t \\ &= (\rho+\varepsilon)^t. \end{aligned} \tag{2.3}$$

Combining (2.1), (2.2) and (2.3) we get

$$E\left\|f-\frac{1}{n}\sum_{j=1}^{n} g_j\right\|^t < \frac{C^t(\rho+\varepsilon)^t}{n^{t-1}(1-\beta)^{t-1}}+\frac{\varepsilon^t}{\beta^{t-1}}.$$

At this point we divert from the proof in [DDGS93]. The above equation is valid for every $\beta\in(0,1)$. A simple calculus shows that the right hand side is minimized when $\beta=(1+\frac{C(\rho+\varepsilon)}{\varepsilon n^{1-1/t}})^{-1}$. This value of β yields

$$E\left\|f-\frac{1}{n}\sum_{j=1}^{n} g_j\right\|^t < \left(\frac{C(\rho+\varepsilon)}{n^{1-1/t}}+\varepsilon\right)^t.$$

Thus, by Markov inequality we have

$$\begin{aligned} &\Pr\left[\left\|f-\frac{1}{n}\sum_{j=1}^{n} g_j\right\|^t \ge Q\cdot\left(\frac{C(\rho+\varepsilon)}{n^{1-1/t}}+\varepsilon\right)^t\right] \\ &\quad = \Pr\left[\left\|f-\frac{1}{n}\sum_{j=1}^{n} g_j\right\| \ge Q^{1/t}\cdot\left(\frac{C(\rho+\varepsilon)}{n^{1-1/t}}+\varepsilon\right)\right] \\ &\quad \le \frac{1}{Q}. \end{aligned}$$

□

Theorem 2.1 leads to the following algorithm.

Algorithm 2.2 (Probabilistic from auxiliary approximation)

Given: *f, $\mathcal{G}$, n*

1. *Pick $\eta>0$ and put $\varepsilon_{apx}=\frac{\eta}{n^{1-1/t}}$. The value of η affects the quality of approximation; as we will see below, for $\mathcal{L}_p$ spaces ($1\le p\le 2$) it is reasonable to make η comparable with ρ, or* $\sup\{\|g-f\|:g\in\mathcal{G}\}$.

2. *Find for $i=1,\dots,N$ functions $h_i\in\mathcal{G}$, and coefficients $c_i\ge 0$ with $\sum_{i=1}^{N} c_i=1$ so that*

$$\left|f-\sum_{i=1}^{N} c_i h_i\right| < \varepsilon_{apx}$$

3. *For j from 1 to n choose randomly $t_j \in \{1, \ldots, N\}$ with $\Pr[t_j = i] = c_i$.*
4. *Put $g_{apx} = \frac{1}{n}\sum_{i=1}^{n} h_{t_j}$*
5. *If $\|f - g_{apx}\| \geq Q^{1/t}\big(\frac{C(\rho+\varepsilon_{apx})}{n^{1-1/t}} + \varepsilon_{apx}\big)$ go back to step 2.*

Output: g_{apx}
Expected time: $t_{apx}(\mathcal{G}, \varepsilon_{apx}, N) + (O(n \log N) + nt_{int})\frac{Q}{Q-1}$.

Corollary 2.3 (Error of probabilistic algorithm) *Algorithm 2.2 gives an approximation of f with an error at most*

$$Q^{1/t}\frac{C\rho + \eta(1 - C/n^{1-1/t})}{n^{1-1/t}}$$

after an expected number of $\frac{Q}{Q-1}$ repetitions.

Proof: By Theorem 2.1 the probability of failure in Step 5 is at most $1/Q$, so we can bound the number of repetitions by a geometric random variable with probability of success $1 - 1/Q$. The expectation of this random variable is $\frac{Q}{Q-1}$. □

Remark We have kept C in the estimates to preserve generality. However, in the most interesting case (from the practical point of view), which is $\mathcal{L}_p$ with $p \in [1, 2]$, we have $C = 1$ (see [DDGS93], the discussion above Corollary 2.6).

Now we set out to estimate the running time of Algorithm 2.2. As in the previous section, we will postpone the discussion of some "sub-procedures" to Section 3.

The second step has time complexity $t_{apx}(\mathcal{G}, \varepsilon_{apx}, N)$. Then we need to pick a random number n times with the correct distribution. The standard way to do this is to define $b_i = \sum_{j \leq i} c_j$, so that $0 = b_0 < b_1 < \cdots < b_N = 1$. Then we pick a uniformly random number $u \in [0, 1]$ and find i such that $u \in (b_{i-1}, b_i)$; we let the next t_j be equal to i. To be able to tell which interval u belongs to, we need to sample it with sufficient precision. When we do this "adaptively", that is we only generate as many bits as needed to tell which interval will u end up in, we need on average

$$\sum_{i=1}^{N} c_i(1 + \log_2 \frac{1}{c_i}) \leq 1 + \log_2 \sum_{i=1}^{N} c_i \frac{1}{c_i} = 1 + \log_2 N$$

random bits (we have used concavity of log and the Jensen inequality for numbers $1/c_i$). Altogether Step 3 takes time $O(n \log N)$.

For Step 4 and 5 we need time $O(n \cdot t_{int})$, as we need to compute the sum in Step 4 for every point, where we will be evaluating f_{apx} when checking the norm $\|f - f_{apx}\|$ in Step 5.

Altogether, we obtain the expected time complexity

$$t_{apx}(\mathcal{G}, \varepsilon_{apx}, N) + (O(n \log N) + nt_{int})\frac{Q}{Q-1}.$$

In Section 4 we will discuss how this compares to the estimates obtained for the iterative algorithm.

Next we proceed with a variant of the above results for functions for which an integral representation is known. The following theorem is based on ideas from [DDGS93].

Theorem 2.4 (Probabilistic error in Banach spaces, int. repr.) *Let $H \subseteq \mathbb{R}^d$ be open and let $X = \mathcal{L}_p(H)$. Put $t = \min\{p, 2\}$. Let A be a subset of $\mathbb{R}^k$. Suppose $\varphi_a = \varphi(\cdot, a)$ is in X for each $a \in A$. Put $\mathcal{G} = \{\pm\varphi_a : a \in A\} \subset X$ and assume that we have an integral representation of f of form*

$$f(x) = \int_A w(a)\varphi(x,a)\,\mathrm{d}a \qquad \int_A |w(a)| = 1\,.$$

Let $\rho > 0$ be such that for every $g \in \mathcal{G}$ we have $\|g - f\| \leq \rho$.

We pick $g_1, \ldots, g_n$ from $\mathcal{G}$ independently at random, so that $|w(a)|$ is the density of the probability of choosing $\pm\varphi(\cdot, a)$. We take $+\varphi(\cdot, a)$ for $w(a) > 0$ and $-\varphi(\cdot, a)$ for $w(a) < 0$.

Then the average $g_{apx} = \frac{1}{n}\sum_{i=1}^n g_i$ is (with high probability) a reasonable approximation, namely

$$\Pr\left[\|g_{apx} - f\| > Q^{1/t} \cdot \frac{C\rho}{n^{1-1/t}}\right] < \frac{1}{Q}\,.$$

Here $C > 0$ is a constant depending on X but independent of n, and $Q > 1$ is a parameter specifying the trade-off between quality of approximation and time needed to find it.

Proof: Take g_j to be a sequence of independent random variables on X every one of them taking value h_i with probability c_i.

$$E\left\|f - \frac{1}{n}\sum_{j=1}^n g_j\right\|^t = \frac{1}{n^t} E\left\|\sum_{j=1}^n (f - g_j)\right\|^t \tag{2.4}$$

The range of g_j is a subset of $\mathcal{L}^p(H)$, which is a separable Banach space. ($C_0(H)$ is dense in $\mathcal{L}^p(H)$ (for $1 \leq p < \infty$) – Corollary 19.20 in [Js03]. Then we use the Weierstrass theorem to approximate continuous functions by polynomials and we observe that it is enough to consider polynomials with rational coefficients.) Consequently, g_j is a Radon random variable [LeTa91, p.38]. As the Rademacher type of X is t, by Proposition 9.11 in [LeTa91] we have that there is a constant $C = C(X)$ so that

$$E\left\|\sum_{j=1}^n (f - g_j)\right\|^t \leq C^t \sum_{j=1}^n E\|f - g_j\|^t. \tag{2.5}$$

It is also true that for every j

$$\begin{aligned} E\|f - g_j\|^t &= \int_A |w(a)|\|f \pm \varphi(x,a)\|^t \\ &\leq \int_A |w(a)|\rho^t \\ &= \rho^t. \end{aligned} \tag{2.6}$$

Combining (2.4), (2.5) and (2.6) we get

$$E\left\|f - \frac{1}{n}\sum_{j=1}^n g_j\right\|^t < \frac{C^t\rho^t}{n^{t-1}}\,.$$

Thus, by Markov inequality we have

$$\begin{aligned}
&\Pr\left[\left\|f - \frac{1}{n}\sum_{j=1}^{n} g_j\right\|^t \geq Q \cdot \big(\frac{C\rho}{n^{1-1/t}}\big)^t\right] \\
&= \Pr\left[\left\|f - \frac{1}{n}\sum_{j=1}^{n} g_j\right\| \geq Q^{1/t} \cdot \big(\frac{C\rho}{n^{1-1/t}}\big)\right] \\
&\leq \frac{1}{Q}\,.
\end{aligned}$$

□

Theorem 2.4 leads to the following algorithm.

Algorithm 2.5 (Probabilistic from integral representation)

Given: f, φ, n

1. *Find an integral representation for f of form*

$$f(x) = \int_A w(a)\varphi(x,a)\,,$$

so that $\int_A |w(a)| = 1$.

2. *For j from 1 to n choose randomly $a_j \in A$ with the density of probability $|w(a)|$.*
3. *Put $g_j = \varphi_{a_j}$ if $w(a_j) > 0$, and $g_j = -\varphi_{a_j}$ if $w(a_j) \leq 0$.*
4. *Put $g_{apx} = \frac{1}{n}\sum_{i=1}^{n} g_j$*
5. *If $\|f - g_{apx}\| \geq Q^{1/t}\big(\frac{C\rho}{n^{1-1/t}}\big)$ go back to step 2.*

Output: g_{apx}
Expected time: $O(ns_{gen}(w)d \cdot t_{int}\frac{Q}{Q-1})$

For discussion of s_{gen} see below.

Corollary 2.6 (Error of integral probabilistic algorithm) *Algorithm 2.5 gives an approximation of f with an error at most*

$$Q^{1/t}\frac{C\rho}{n^{1-1/t}}$$

after an expected number of $\frac{Q}{Q-1}$ repetitions.

The proof is the same as for Corollary 2.3.

In Algorithm 2.5 we need to generate a according to a non-uniform distribution, the density of the probability being $|w(a)|$. A general approach to do this is the acceptance-rejection method (developed by von Neumann). The basic idea is as follows. We find C so that $|w(a)| \leq C$ for each a. For simplicity assume first that $vol(A) = 1$ and $\int_A |w(a)| = 1$. We generate $a \in A$ and $u \in [0,1]$ uniformly at random, and accept a if $u \leq \frac{|w(a)|}{C}$ (that is, we accept a with probability proportional to $|w(a)|$).

If we reject the generated a, we repeat the procedure. It is easy to see, that the expected number of steps needed to generate one a according to $|w(a)|$ is C: thus this basic approach is inefficient if $|w(a)|$ varies widely. In such case we try to find a simple upper bound $CW(a)$ on $|w(a)|$ (so that $\int_A W(a) = 1$) and generate a with density $W(a)$, rather than uniformly. For the details we refer the reader to Section 5.1.2 of [We00]. If we are given w such that $I = \int_A |w(a)| \neq 1$, we just find (estimate) I (for example by Monte Carlo integration, see Section 3.1) and then we will be generating a according to $|w(a)|/I$ and then put $g_i = \pm I\varphi(\cdot, a)$.

The conclusion is that our ability to efficiently generate with density $|w(a)|$ depends on our knowledge of the function w. We will use $s_{gen}(w)$ to denote the number of steps needed to generate one a with probability density $|w(a)|$.

The estimate of the running time of Algorithm 2.5 is derived similarly as that of Algorithm 2.2. There are two differences: instead of finding an approximating function f_{apx}, we need to get the integral representation – more precisely, we will be repeatedly evaluating $w(a)$.

When the approach of [KKK97] is used to obtain the integral representation, we may use the discussion in Section 3.3, where we estimate the time to evaluate $w(a)$ by $d \cdot t_{int}$. Also we need to estimate $\int_A |w(a)|$, which takes about $d \cdot (t_{int})^2$.

Altogether, we obtain the expected time complexity

$$d \cdot t_{int}^2 + (ns_{gen}(w)d \cdot t_{int} + n \cdot t_{int})\frac{Q}{Q-1} = O(ns_{gen}(w)d \cdot t_{int}\frac{Q}{Q-1}) .$$

In the above equation we needed to compare t_{int} and ns_{gen}: from Section 3.1 we have $t_{int} \sim \varepsilon^{-2}$, while $n \sim \varepsilon^{-\frac{1}{1-1/t}}$. We see that for $t = 2$ we have $t_{int} \sim n$, if t gets closer to 1 (arguably the most interesting case) n gets larger. Moreover s_{gen} is at least 1. This allows us to simplify the time complexity as above.

In Section 4 we will discuss how this compares to the other algorithms.

3 Numerical Issues

In this section we will discuss numerical issues that arose when estimating running times of the algorithms proposed in the previous two sections. We do not claim any originality in this section, and also, given the limited space, we hardly scratch the surface of the extensive field of research, which numerical analysis presents. Our purpose here is to shortly overview relevant known results in order to be able to somewhat understand the running times of the studied algorithms. We will deal with t_{int} in Section 3.1, t_{opt} in Section 3.2, and t_{apx} in Section 3.3.

Our model of computation is a simple one: we assume, that variables in our algorithms hold one real number, and that we can perform basic operations with real numbers at unit cost. Thus, we avoid discussing in detail numerical stability of the presented algorithms and also the effect of required precision is treated only partially.

3.1 Numerical Integration

In the discussed algorithms we need to compute numerically various integrals. In this section we start by an overview of known methods for estimating integrals. Then we show how to extend the idea of Monte-Carlo integration for the situation when the integral depends on parameter. We will include an estimate of time complexity of these methods. This will in the end lead to an estimate on t_{int}.

A. Classic methods Finding numerical estimates of definite integrals is a central topic in numerical analysis. The classical approach to estimate $I = \int_S h(x)\,\mathrm{d}x$ is as follows: We divide the region S into small parts (by a "dense regular grid"). On each of them we approximate h by polynomials and integrate those. The next claim summarizes such results.

Claim 3.1 (Classic numerical integration [Numerical Basics]) *Suppose $h \in C^r(S)$, where $S \subseteq \mathbb{R}^d$. We can estimate $\int_S h(x)$ using values $h(x_i)$ in some (carefully chosen) points $x_1, \dots, x_N$. with error at most*

$$O(N^{-\frac{r}{d}})$$

(the constant in $O(\cdot)$ depends on $\max_{x\in S} |f^{(r)}(x)|$ and on the volume of S).

We see another case of the "curse of dimensionality": to achieve error bounded by ε, we need to take $N = \Omega(\varepsilon^{-d/r})$. One way out of this would be to consider functions defined in $\mathbb{R}^d$ that are d-times continuously differentiable, but this may be a too strict requirement. Another common method is to use "sparse grids" – i.e., we use more points to sample the function h in that parts of S, where h varies more. This method yields error $O(\frac{(\log N)^{(d-1)(r+1)}}{N^r})$, where, again, N is the number of sample points and r the regularity of f. When d is large (as is usually the case in applications of neural networks), the standard approach is to use random sampling, which we discuss in the next section.

B. Monte-Carlo methods of integration We start by an overview of the basic Monte-Carlo method. For more details, see e.g. [We00]. In the same setting as above, let $x_1, \dots, x_N$ be chosen uniformly at random from S. Put

$$E = vol(S)\tfrac{1}{N}\sum_{i=1}^{N} h(x_i).$$

As expectation of $h(x_i)$ is $I/vol(S)$, we get that E is an unbiased estimator for I. To find how reliable results E yields, one uses tail estimates for sums of independent random variables. For example, by using the Chebyshev and Chernoff-Hoeffding bound we can easily get the following.

Claim 3.2 (Monte-Carlo integration, see, e.g., [CuSm01]) *Let h, S, and E be as above. Then the expected value of E is $\int_S h(x)$. Moreover:*

(a) If h is in $\mathcal{L}^2(S)$ and $\sigma^2 = \frac{1}{vol(S)} \int_S (h\ vol(S) - \int_S h(x))^2$, then for any $\eta > 0$ we have

$$\Pr[|E - \int_S h| > \frac{\eta}{\sqrt{N}}] \le \frac{\sigma^2}{\eta^2}\,.$$

(b) If h is in $\mathcal{L}^\infty(S)$ and M is such that $|h(x)vol(S) - \int_S h(s)\,\mathrm{d}s| \le M$ for almost every $x \in S$, then for any $\eta > 0$ we have

$$\Pr[|E - \int_S h| > \frac{\eta}{\sqrt{N}}] \le 2e^{-\frac{\eta^2}{2M^2}}\,.$$

As a side-remark, we note that in the literature about the application side of Monte-Carlo integration, the method is frequently justified by mentioning the Central Limit Theorem. Indeed, Central Limit Theorem gives that the distribution of $(E - \int_S h)\sqrt{N}$ converges to normal distribution. This yields bound similar to that in part (b) for any function h, that satisfies assumptions of the Central Limit Theorem, in particular the Lyapunov condition is satisfied for $h \in \mathcal{L}^{2+\varepsilon}(S)$ for any $\varepsilon > 0$. However, this only yields information about the limit distribution. If we want a bound for some particular N, we need to utilize the Berry–Esseen theorem: it estimates the speed of convergence to the normal distribution, but the guaranteed bound is only $O(1/\sqrt{N})$.

Claim 3.2 allows us to quickly estimate the various integrals we need for time complexity estimates for our algorithms. If we want to get the "probable" error ε, it suffices to use $N \geq C(1/\varepsilon)^2$, with the constant C depending on the function to be integrated (and also on the probability with which we want to get the desired precision). As we do not care about constants in our analysis, we summarize this as $t_{int}(\varepsilon) = O(1/\varepsilon^2)$.

3.2 Optimization

In this section we briefly touch the problem of optimization, as we are using it in one of our iterative algorithms. We can give only a glimpse of this interesting subject, for a more thorough treatment we suggest for instance [BoVa04].

In Algorithm 1.2 we needed to find a function for which a certain functional is small. It would be hard to be searching for an arbitrary function, fortunately we are looking for functions in specific forms. In the perceptron model we are searching for an optimal ridge function $\sigma(a \cdot x + b)$, where σ is the given activation function, $a, x \in \mathbb{R}^d$, $b \in \mathbb{R}$. This means that we are actually minimizing a function of $d+1$ real variables (a, b).

Generally, suppose we are seeking for a minimum of a function h, defined on $A \subseteq \mathbb{R}^{d'}$; we assume that d' is large. There is a plethora of branches of the optimization theory, where this question is studied under various assumptions (h is linear, quadratic, convex, ...). Unfortunately, none of these assumptions is applicable. The only general approach that guarantees finding the global optimum, is to sample h at all points of a dense enough grid, where what is dense enough depends on some bounds on derivatives of h (or its modulus of continuity). Obviously, this is not practical even for moderately large d', as the number of sample points scales as $m^{d'}$. (Here m is the number of division points in one dimension, which depends on the size of the region we are optimizing over, and on the modulus of continuity of the optimized function.) Thus, we need to leave the realm of guaranteed methods. Most methods used in practice are actually searching for local minimum. Then, a variety of methods (multiple runs of the algorithm with random starting points, simulated annealing, etc.) is used to get a chance of finding the global minimum. Obviously, without some knowledge of the function h, nothing can be said, and we can only attempt for a reasonably good heuristic.

A better understood question is how to search for the local minimum. The basic way is the gradient method:[1] we start at a random point $a = a_0$, compute the gradient $\nabla h(a)$ (assuming it exists), and move against the gradient, by a (carefully guessed) distance. In Algorithm 1.2 we need to optimize the peak functional $F_k(g)$. As we discussed after the algorithm $F_k(g) = \int g(x)\psi(x)\,\mathrm{d}x$ for some function ψ. In the notation of this section, the point $a = (a_1, \ldots, a'_d) \in A$ corresponds to parameters we use to describe the function g, $h(a)$ is the integral $\int g\psi$. Under mild assumptions the gradient $\nabla h(a)$ is the integral $\int (\nabla g)\psi$. (Here $\nabla g = (\frac{\partial}{\partial a_i} g)_i$ is a collection of functions that describe how g changes if we change the parameters a.) It follows that single step of gradient method will take time $d' \cdot t_{int}$.

The speed of convergence depends on both the chosen variant of this method, and on properties of h. One of the estimates for the number of repetitions (under the assumption of strong convexity of h) is as follows, see Chapter 9 in [BoVa04]. We let κ denote an upper bound on the condition number of the matrix of the second partial derivatives $\nabla^2 h(a)$. We let $a^* \in A$ be the point where the minimum

[1]This is the usual way to carry on the backpropagation algorithm, the common way to train a feedforward neural network.

of h is attained (for strongly convex h such point exists and is unique). Finally, $\varepsilon > 0$ is the required precision, that is we want to find a so that $h(a) - h(a^*) \le \varepsilon$. The number of iterations of the gradient method to achieve this is at most

$$\frac{\log \frac{h(a_0)-h(a^*)}{\varepsilon}}{\log(1-\frac{1}{\kappa})} . \tag{3.1}$$

Unfortunately, the condition number can be rather large and, what is worse, we need to minimize a function that is not convex. In this view, the time t_{opt} needed to find a minimum of a function, is hard to evaluate. A reasonable estimate is

$$t_{opt} \sim d' \cdot t_{int} \cdot s_{opt} ,$$

where $s_{opt} = s_{opt}(\varepsilon)$ (the required number of steps) is estimated by the formula (3.1).

3.3 Approximation

In this section we discuss several ways to obtain a "starting approximation" of a function $f(x)$ we are trying to estimate. Our point of view will be somewhat different from the one in Sections 1 and 2 in that we are not striving to get small number of approximating functions. Thus, we allow a somewhat larger number of functions (resulting in a larger time-complexity of obtaining the solution, but, perhaps, also making it easier to find a solution). Then we shall use algorithms in Sections 1 and 2 to get efficient approximations.

So suppose we are given a function $f(x)$, for $x \in H$. Let us assume that this function is given to us by means of values at sample points (the position of these sample points may or may not be under our control).

High-dimensional optimization One way to find the desired approximation is to choose N and optimize expression $\sum_{i=1}^{N} c_i g_i(x)$. We optimize by modifying the c_i's and the parameters describing the functions g_i. We choose N high enough, so that we can get good enough approximation, but small enough, so that the task is still computationally feasible. By choosing N larger, than will be the number n of the functions in the final approximation, we may hope to partially overcome one of the complications of the high-dimensional optimization – the fact, that optima can be hard to find. By choosing N large, we make sure that very good approximations exist, so our desired precision may not be so hard to achieve. If we use some variant of the gradient method to do the optimization, the amount of computation for one step is N times larger than when we optimize just one function. It is not clear how the number of steps is affected by having a large N. However, we may roughly estimate

$$t_{apx} \sim N \cdot t_{opt} .$$

Using integral representation It may be easier to first find an integral representation. (Then we will "sample from the integral", instead from the sum.) Two different approaches to obtain a relevant integral representation were pursued by Kůrková, Kainen and Kreinovich, and by Barron. We will concentrate on the approach of Kůrková, Kainen and Kreinovich, using their result, we express $f(x)$ as an integral of form

$$f(x) = \int_A w(a)\varphi(x,a)\,\mathrm{d}x .$$

Here, $\varphi(\cdot, a)$ is the Heaviside function in a direction and shift given by a.

We will discuss next, how computationally accessible this representation is. In Algorithm 2.5 we needed to evaluate $w(a)$ and $\int_A |w(a)|$. To get $w(a)$ we need to compute the d-th directional derivative of f and integrate it over a hyperplane. Numerical estimate of the d-th derivative takes time proportional to d, so we can evaluate $w(a)$ in time $O(dt_{int})$. To estimate $\int_A |w(a)|$, we will use Monte-Carlo integration, which takes time $O(d(t_{int})^2)$. In this formula, the two instances of t_{int} correspond to different integrals (one over x and the other over a), but in most applications these variables have similar dimension, so we neglect this small distinction.

Kernels Yet another way to obtain the starting approximation to the given function is using the Tikhonov regularization method and Reproducing Kernel Hilbert Spaces. In Section 7 we will see how the algorithms presented in this section may be used in such setting.

4 Comparison of proposed algorithms

In this section we will compare the algorithms under discussion: the iterative algorithms (either with repeated optimization, or using an auxiliary approximation) and the randomized algorithms (using an auxiliary approximation or an integral representation). In Table 4.1 we summarized what we found in the previous sections. However, the time complexities depend highly on the particular problem at hand. Indeed, the choice of functions used in the approximation, as well as the data we are trying to approximate affect, among else, the number of steps needed in the gradient method, which is usually the workhorse of the optimization "subprocedure". We make no attempt to analyze these subtleties; for this reason (and for the sake of brevity) we also suppress $\mathcal{G}$ from the expressions in Table 4.1. We will not try to provide universal conclusion, which of the algorithms is the best. Instead, we will try to address this question in different "scenarios". The scenarios will describe how we search for the "auxiliary" approximation f_{apx}.

Iterative 1 (rep. optimiz.)	$\sum_{k=1}^{n-1} t_{opt}(\frac{c_1}{k^{1-1/t}}) \sim n \cdot t_{opt}(\frac{c_1}{n^{1-1/t}})$
Itrative 2 (using aux. approx.)	$t_{apx}(\frac{c_2\eta}{n^{1-1/t}}, N) + nN \cdot t_{int}$
Randomized 1 (approx.)	$t_{apx}(\frac{c_3\eta}{n^{1-1/t}}, N) + \big(O(n \log N) + n \cdot t_{int}\big)\frac{Q}{Q-1}$
Randomized 2 (int.repr.)	$O(nd \cdot s_{gen}(w) \cdot t_{int} \cdot \frac{Q}{Q-1})$

Table 4.1: Time complexities of the discussed algorithms.

Disclaimer about N One piece of information that will be missing from our analysis is the size of N relative to n. The value of N depends on our choice, and in turns affects the complexity of the approximation. Roughly speaking, if N increases, it is more work to find the approximation, but the approximation will become an easier task (we know that the achievable error decreases with the number of terms in the approximation). As such, optimal choice of N depends on the task being solved and the used methods. From practical considerations, however, it is reasonable to expect that $n \leq N \leq n^k$ for some small constant k.

Iterative 2 vs. Randomized 1 In this paragraph we will (somewhat in contrary to the first paragraph) compare the algorithms Iterative 2 and Randomized 1 "universally". Recall that Q in the randomized algorithm describes the expected quality of the solution (we are ready to be satisfied with a solution

Q times worse than what is guaranteed to exist). As this is a constant greater than 1, we can see, that when using auxiliary approximation the Randomized algorithm is always superior to Iterative 2. In the following, we will only discuss Iterative 1 and Randomized.

1. scenario: auxiliary approximation is found by high-dimensional optimization As discussed in Section 3.3, $t_{apx}(\mathcal{G}, \varepsilon, N) \sim N \cdot t_{opt}(\mathcal{G}, \varepsilon)$. Consequently,

$$t_{Iter1} \leq n \cdot t_{opt} < N \cdot t_{opt} \sim t_{apx} \leq t_{Rand_1} .$$

This suggests, that in this case we should be using the iterative algorithm, especially if N is very large. This is probably the expected result: after all, the idea behind the Iterative algorithm as suggested by Jones [Jo92] is to reduce the dimension of the problem.

2. scenario: an auxiliary approximation is given In this second scenario, we assume that we are given *some* approximation of a function $f(x)$, possibly using many functions. Our task is to find an approximation using less functions and achieving a reasonable error. In this setting, the first iterative algorithm is wasteful, as it cannot anyhow use the given approximation. Indeed, we certainly have $t_{opt} \sim dt_{int}s_{opt} \gg t_{int}$. Also, for the typical range of N we have $t_{opt} \gg \log N$. Consequently, in this scenario the randomized 1 algorithm provides an efficient method to improve a given approximation by "pruning it at random".

$$t_{Rand1} < t_{Iter1}$$

This will be utilized in section 7:we will consider situations where a natural approach yields sums with large number of terms N (Theorem 6.1).

3. scenario: auxiliary approximation is obtained from an integral representation In many situations we may have access to some sort of integral representation of the function under study. We will use the second version of the rendomized algorithms in such case.

We will consider here an approach of this type, which was studied by Kůrková, Kainen and Kreinovich [KKK97]. As discussed in Section 3.3 and 2, the time complexity of our algorithm is $O(ns_{gen}(w)d \cdot t_{int}\frac{Q}{Q-1})$ To compare this to the iterative algorithm, recall from Section 3.2 that $t_{opt} \sim d \cdot t_{int}s_{opt}$. Thus we are comparing

$$nd \cdot t_{int}s_{gen}(w)\frac{Q}{Q-1} \qquad \text{with} \qquad nd \cdot t_{int}s_{opt} ,$$

or $s_{gen}(w)\frac{Q}{Q-1}$ with s_{opt}. Now s_{gen} can be as small as 1 (and even in the simplistic approach it is at most $(\max w(a))/\int_A |w(a)|)$. In particular, (in our simple analysis, at least) it does not depend on the required precision. On the other hand, s_{opt} is much larger than 1, and it depends on the required error (although only by a factor $\log 1/\varepsilon$). Most importantly, we may run into troubles with finding a local minimum that is not global. Thus, we may conclude that when the integral representation is available, the randomized algorithm that utilizes it (Algorithm 2.5) is better than the iterative one.

$$t_{Rand2} < t_{Iter1}$$

So far we proposed a new algorithm to find neural networks with small number of units and reasonable approximation error. (Equivalently, we are seeking an approximation of a given function by a sum with each term being one of a given set of approximating functions, where we want this sum to have a small number of terms.) Roughly speaking, the algorithm utilizes an auxiliary approximation (using a large number of units) and “prunes” this approximation by taking only some of these units. This selection process is done randomly, and, as shown in Corollary 2.3 it provides an approximation with close-to-optimal error in a small expected number of steps.

We analyzed two scenarios, where this algorithm is useful. One is when we are given an approximation and the goal is to use a smaller number of units without increasing the error too much. (This combines well with the techniques that we will discuss in section 7.)Another scenario, where the proposed algorithm is better than the usual one (which uses repeated optimization), is when our function admits an integral representation. We utilize a result of Kůrková, Kainen and Kreinovich [KKK97]: given $f \in C^d(\mathbb{R}^d)$, there is an integral representation using Heaviside units with explicitly given weights.

Finally, let us remark, that the suggested randomized algorithm works without modification for other models of neural networks (such as RBF networks).

In this part of the paper we show a natural way to use randomized algorithm for pruning a too rich network. We consider the approximation task formulated as regularized minimization problem with kernel-based stabilizer. These approximation tasks exhibit easy derivation of solution to the problem in the shape of linear combination of kernel functions (see for example [SchSl02]) that can be interpreted as one-hidden-layer feedforward network schemas. For basic quadratic error functional parameters of these networks can be computed from a linear system; however, these networks tend to have too many hidden units (in fact as many as was the number of data). There are many ways around this problem (see for example [SchWe06]). We propose a solution that uses randomized algorithm (2.2). This is probably the most interesting result of this section.

We also mention concrete applications of a special type of kernels proposed by the author – sum kernels and product kernels. As part of our joint work [KS05b], [KS06] Petra Kudov-Vidnerov tested these new schemas on real-life data and compared them to classical solutions. Experiments on pruning by randomized algorithm are running.

5 Reproducing Kernel Hilbert Spaces

Reproducing kernels proved themselves very useful in approximation theory. We introduce the basic concept and properties in Section 5.1. In Sections 5.2 and 5.3 we show two special types of composite kernels which we will use later in Section 8 to derive concrete approximation schemas.

5.1 Basic Properties

Reproducing Kernel Hilbert Spaces were formally defined by Aronszajn [Ar50], although related concepts were used even before. They find ample use in applications ranging from PDE’s to machine learning, see e.g. a survey [SchWe06]. In this section we present the reader with definitions and a brief overview of the properties needed for our purposes. We follow [Ar50], where we also kindly refer the reader for more details and all of the proofs.

Given a Hilbert space $\mathcal{H}$ of functions (real or complex) defined on $\Omega \subset \mathbb{R}^d$ we let $\mathcal{F}_x$ denote the evaluation functional, that is $\mathcal{F}_x(f) = f(x)$. If for each $x \in \Omega$ the evaluation functional $\mathcal{F}_x$ is continuous, we say that $\mathcal{H}$ is a *Reproducing Kernel Hilbert Space*, shortly an *RKHS*. The term is suggested by the following important property: In an RKHS as above there exists a positive definite symmetric function $k : \Omega \times \Omega \to \mathbb{R}$ (*reproducing kernel* corresponding to $\mathcal{H}$) such that

(i) for any $f \in \mathcal{H}$ and $y \in \Omega$ the following reproducing property holds

$$f(y) = \langle f(x), k(x,y) \rangle ,$$

where $\langle \cdot, \cdot \rangle$ is scalar product in $\mathcal{H}$ and

(ii) for every $y \in \Omega$, the function $k_y(x) = k(x,y)$ is an element of $\mathcal{H}$.

We bring here some basic properties of RKHS.

Lemma 5.1 (Uniqueness of reproducing kernel [Ar50]) *Let $\mathcal{H}$ be a Hilbert space with a reproducing kernel k. Then k is unique.*

Proof: We include here this short proof (due to [Ar50]) to illustrate the reproducing property of reproducing kernels. Suppose we have two reproducing kernels k, k' and $k \neq k'$. Then for some y we have $0 < \|k_y - k'_y\|^2 = \langle k_y - k'_y, k_y - k'_y \rangle = \langle k_y - k'_y, k_y \rangle - \langle k_y - k'_y, k'_y \rangle$. By the property (i) in the definition of kernel, this equals $(k(y,y) - k'(y,y)) - (k(y,y) - k'(y,y)) = 0$, which is a contradiction. $\square$

Lemma 5.2 (Existence of reproducing kernel [Ar50]) *Let $\mathcal{H}$ be a Hilbert space of functions on $\Omega \subset \mathbb{R}^d$. Suppose that all evaluation functionals $\mathcal{F}_x$ are continuous. Then there exists a reproducing kernel satisfying properties (i) and (ii) and from the reproducing property it follows that the kernel is positive definite. On the other hand from (i) and (ii) we conclude that evaluation functionals are continuous.*

Lemma 5.3 (Uniqueness of RHKS [Ar50]) *To every $k(x,y)$ satisfying the properties (i) and (ii) there corresponds one and only one Hilbert space $\mathcal{H}$ admitting k as a reproducing kernel.*

5.2 Sum of Reproducing Kernels

In this section we will present the notion of a sum of RKHS's, as described in [Ar50]. This will be utilized later, in Section 8.2. The sum of two RKHS's is defined simply as the space containing all sums of two functions, one from each of the RKHS's. Some care has to be taken when defining the norm (and scalar product) as there may be many sums that yield the same function.

For $i = 1,2$ let F_i be an RKHS of functions on Ω, let k_i be the corresponding kernel and $\|.\|_i$ the corresponding norm. Consider the space $H = \{\{f_1, f_2\} \mid f_1 \in F_1, f_2 \in F_2\}$. with norm given by $\|\{f_1, f_2\}\|^2 = \|f_1\|_1^2 + \|f_2\|_2^2$.

We have to deal with duplicities. Consider the class F_0 of all functions f belonging to $F_1 \cap F_2$. We define $H_0 := \{\{f, -f\}; f \in F_0\}$. H_0 is a closed subspace of H and thus we can write $H = H_0 \oplus H'$, where H' is the subspace complementary to H_0. Now to every element $\{f_1, f_2\}$ of H there corresponds a function $f(x) = f_1(x) + f_2(x)$. This is a linear correspondence transforming H into a linear class of functions F. Elements of H_0 are transformed into zero functions and thus the correspondence between H' and F is one-to-one and has an inverse (for $f \in F$ we let $\{g_1(f), g_2(f)\}$ be the corresponding element in $H' \subseteq H$). We define norm on F by

$$\|f\|^2 = \|\{g_1(f), g_2(f)\}\|^2 = \|g_1(f)\|_1^2 + \|g_2(f)\|_2^2 . \tag{5.1}$$

Here we present a result from [Ar50] showing that to the class F with the above defined norm there corresponds a reproducing kernel $k = k_1 + k_2$.

Theorem 5.4 (Sum-kernel RKHS [Ar50]) *Let F_i be RKHS and k_i and $\|.\|_i$ the corresponding kernels and norms. Let F be as above with norm given by* (5.1)*. Then*

$$k(x,y) = k_1(x,y) + k_2(x,y)$$

is the kernel corresponding to F.

The same holds when F is defined as the class of all functions $f = f_1 + f_2$ with f_i in F_i and norm $\|f\|^2 = \min \|f_1\|_1^2 + \|f_2\|_2^2$, the minimum taken over all decompositions $f = f_1 + f_2$ with f_i in F_i.

It is easy to extend this theorem to sum of more than two spaces: $k(x,y) = \sum_{i=1}^n k_i(x,y)$. We call F a Sum-Kernel Reproducing Hilbert Space.

5.3 Product of Reproducing Kernels

Here we will consider product of Reproducing Kernel Hilbert Spaces, again following [Ar50]. For $i = 1,2$ let F_i be an RKHS of functions on Ω_i, let k_i be the corresponding kernel. Consider the set of functions on $\Omega = \Omega_1 \times \Omega_2$ defined by

$$F' = \{\sum_{i=1}^n f_{1,i}(x_1) f_{2,i}(x_2) \mid n \in \mathbb{N}, f_{1,i} \in F_1, f_{2,i} \in F_2\}\,.$$

Clearly, F' is a vector space. We define a scalar product on F': Let f, g be elements of F' expressed as $f(x_1,x_2) = \sum_{i=1}^n f_{1,i}(x_1) f_{2,i}(x_2)$, and $g(x_1,x_2) = \sum_{j=1}^m g_{1,j}(x_1) g_{2,j}(x_2)$. We define

$$\langle f, g\rangle = \sum_{i=1}^n \sum_{j=1}^m \langle f_{1,i}, g_{1,j}\rangle_1 \langle f_{2,i}, g_{2,j}\rangle_2$$

(with $\langle\cdot,\cdot\rangle_i$ denoting the scalar product in F_i). It is a routine to check that this definition does not depend on the particular form in which f and g are expressed and that the properties of scalar product are satisfied. We define norm on F' by $\|f\| = \sqrt{\langle f, f\rangle}$. Finally, let F be the completion of F'. It can be shown [Ar50] that the completion exists not only as an abstract Hilbert space but that F is in fact a space of functions on Ω. We call F the product of F_1 and F_2 and write $F = F_1 \otimes F_2$.

Theorem 5.5 (Product-kernel RKHS [Ar50]) *For $i = 1,2$ let F_i be an RKHS on Ω_i with kernel k_i. Then the product $F = F_1 \otimes F_2$ on $\Omega_1 \times \Omega_2$ is an RKHS with kernel given by*

$$k((x_1,x_2),(y_1,y_2)) = k_1(x_1,y_1) k_2(x_2,y_2)\,, \tag{5.2}$$

where $x_1, y_1 \in \Omega_1$, $x_2, y_2 \in \Omega_2$.

6 Learning from Data in RKHS Spaces

As advertised at the beginning of this section we are interested in learning from data, i.e., we are given a set of data $z = \{(x_i,y_i)\}_{i=1}^N \subseteq \mathbb{R}^d \times \mathbb{R}$ and we want to "learn" from it how to obtain y from a given x. This means we want to find a function f so that $f(x_i) \doteq y_i$. We do not require exact equality as we may assume the given data are not precise.

However, even if we require $f(x_i) = y_i$ exactly, the problem to find f is still *ill-posed* – there are many solutions to it (see Figure 6.1). This issue is addressed in a variety of ways (Lagrange interpolation,

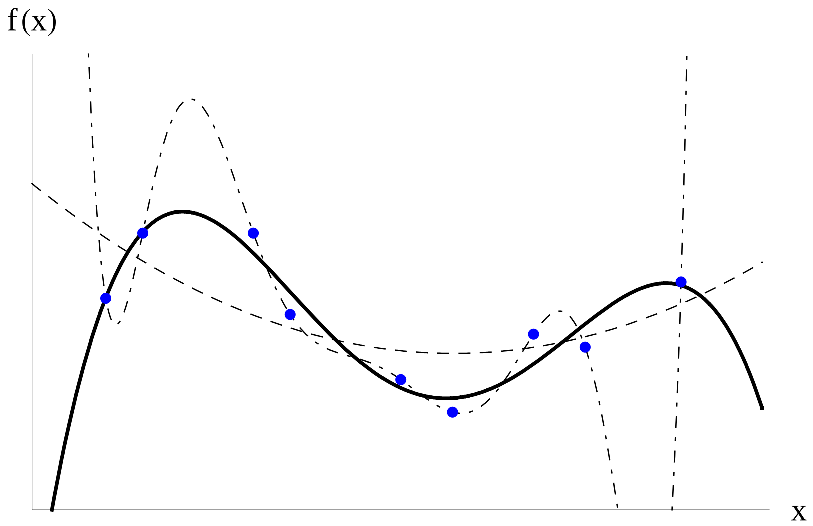

Figure 6.1: Illustration of the basic problem in learning. Data, an over-fitted function, a well-fitted function and a too much generalised function

least square interpolation, ...). The approach to the approximate problem we describe here was suggested by Tikhonov [TiAr77] and studied by a plethora of researchers (see for example [PoSm03, SchSl02, CuSm01, Gi98, PaSa93, Wa90]). We follow the most basic exposition of [GJP95].

We combine the goal of fitting the data with the goal to find a function with good "global properties" or complying with some a-priori knowledge. This can be partially achieved by appropriate choice of the space over which we are minimizing but we can do more. The common approach constructs a functional that quantifies the precision to which a given function fits the data and also the global properties; then we seek to minimize this functional over appropriately chosen set of functions.

Let us be more precise now. We are looking for a function $f : \Omega \to \mathbb{R}$ as a member of some function space X. (Thus the given data z are subset of $\Omega \times \mathbb{R}$ and we also assume $\Omega \subseteq \mathbb{R}^d$.) We let the functional $\mathcal{F}$ be defined by

$$\mathcal{F}(f) = \mathcal{E}_z(f) + \gamma\Phi(f) \tag{6.1}$$

where $\mathcal{E}_z$ is the error functional depending on the data $z = \{(x_i, y_i)\}_{i=1}^N$ and penalizing distance from the data, Φ is the regularization part – the *stabilizer* – penalizing "remoteness from the global property" and γ is the regularization parameter giving the trade-off between the two terms of the functional to be minimized.

The error functional is usually of the form $\mathcal{E}_z(f) = \sum_{i=1}^N V(f(x_i), y_i)$. We can easily infer sufficient conditions on V (for solvability) from proof of Theorem 6.1, but we shall not elaborate on it here. A typical example of the empirical error functional is the classical mean square error: $\mathcal{E}_z(f) = \frac{1}{N}\sum_{i=1}^N (f(x_i) - y_i)^2$ and we will restrict ourselves to it as it exhibits easy derivation of solution. In order for this problem to be well-defined, we would have to have the function f defined pointwise – which is not always the case (consider for example $X = \mathcal{L}^2(\Omega)$ where we may change the values of f on any set of measure 0). In the sequel we will assume, that the *evaluation functionals* $f \mapsto f(x)$ are not only well-defined but also continuous. In the important case when we want X to be a Hilbert space as well, we will be working with the reproducing kernel Hilbert spaces, as introduced in Section 5.

If X is an RKHS with norm $\|\cdot\|_k$, a useful choice for the regularization functional is $\Phi(f) = \|f\|_k^2$ (square of the norm), yielding

$$\mathcal{F}(f) = \mathcal{E}_z(f) + \gamma\|f\|_k^2 \,. \tag{6.2}$$

For such $\mathcal{F}$ it is possible to show the existence and uniqueness of the minimum and derive a simple form for this minimum. Many authors have addressed this topic for general or specific settings (see for example [Wa90, SchSl02, Gi98, PaSa93, PoSm03] and others).

Let $\mathcal{H}$ be an RKHS over $\Omega \subseteq \mathbb{R}^d$ with kernel k and norm $\|.\|_k$. We construct the minimization functional composing of error part $\mathcal{E}_z(f)$ based on data $z = \{(x_i, y_i); i = 1, \dots, N\} \subseteq \mathbb{R}^d \times \mathbb{R}$ and let the regularization part be $\Phi(f) = \|f\|_k^2$ forming

$$\mathcal{F}(f) = \mathcal{E}_z(f) + \gamma\|f\|_k^2, \tag{6.3}$$

where $\gamma \in \mathbb{R}^+$.

To prove existence and uniqueness of solution to such a problem we will use some results from mathematical analysis. Error part of our functional does not have sufficiently nice properties, so the regularization part has to do the job. We employ RKHS in such a way that we easily obtain existence, uniqueness and even form of the solution.

Derivation of the shape of the solution to the regularized minimization problem is referred to as Representer theorem.

6.1 Representer Theorem

In this section we will discuss the solution to the problem of minimizing the functional in (6.2). This is a well-known question addressed for instance in [Wa90]. Our treatment follows the one sketched in [GJP95], which seems to be applicable to a wider class of functionals (6.1). Some details are omitted there, though, in particular it is not verified that the minimum exists. (Incidentally, the existence of minimum is not verified in [Wa90], either, although the proof presented there is easy to fix.) We present the proof (without any claim on being original) for the reader's convenience, as this result is the basis of the results in the later parts of this section. Also, the usage of RKHS's enables the proof to be short and iluminating.

Theorem 6.1 (Representer Theorem – basic version [KiWa71, SchSl02]) *Let $z = \{(x_i, y_i)\}_{i=1}^N \subseteq \mathbb{R}^d \times \mathbb{R}$ be given, let X be an RKHS with kernel k and norm $\|\cdot\|_k$. Let the functional $\mathcal{F}(f)$ be given by*

$$\mathcal{F}(f) = \frac{1}{N}\sum_{i=1}^N (f(x_i) - y_i)^2 + \gamma\|f\|_k^2$$

for a function $f \in X$ ($\gamma > 0$ is a constant).

Then there exists a unique function $f_0 \in X$ that minimizes $\mathcal{F}$. Moreover, f_0 is of form

$$f_0(x) = \sum_{i=1}^N c_i k(x, x_i)\,, \tag{6.4}$$

for real c_i's, where these can be found as

$$c_i = (K[x] + \gamma N I)^{-1} y_i,$$

where $K[x]_{i,j} = k(x_i, x_j)$ is Gramm matrix of kernel k with respect to vector x, I is $N \times N$ identity matrix.

Proof: We will use several basic results in approximation theory, see e.g. [Dn71, p. 7–15]:

Lemma 6.2 (Existence of minimizing function [Dn71]) *Let $\mathcal{F}$ be a weakly sequentially lower semicontinuous functional defined on a weakly sequentially compact set E. Then $\mathcal{F}$ attains its minimum: there is $f_0 \in E$ such that $\mathcal{F}(f_0) = \inf_{f \in E} \mathcal{F}(f)$.*

Lemma 6.3 (Uniqueness of minimizing function [Dn71]) *A strictly quasiconvex functional $\mathcal{F}$ can attain minimum over a convex set C at no more than one point.*

Lemma 6.4 (Necessary condition for minimum [Functional analysis]) *Let the functional $\mathcal{F}$ defined on a set E in a Banach space X be minimized at a point $f_0 \in E$, with f_0 an interior point in the norm topology. For any $h \in X$, if $\mathcal{F}$ has a derivative $\mathrm{D}_h\mathcal{F}(f_0)$ at f_0 in direction h, then $\mathrm{D}_h\mathcal{F}(f_0) = 0$.*

Now we can prove easily existence, uniqueness and form of solution to the kernel-based minimization problem (6.2). Once more we would like to stress that this is not our result nor is it in any sense new (see for example [SchWe06, PoSm03, SchSl02, CuSm01, Gi98, Wa90] and others). The reason for stating it including the proof is that the ideas are simple and we consider them illuminating for understanding the role of kernel in learning and we will also need the theorem in our considerations (Theorem 7.1).

We start by proving that $\mathcal{F}$ attains minimum and to achieve this we will use Lemma 6.2. Pick any $g \in X$. Certainly, any f for which $\mathcal{F}(f) \le \mathcal{F}(g)$ needs to have $\|f\|_k \le \sqrt{\mathcal{F}(g)/\gamma}$, so we only need to minimize $\mathcal{F}$ over the ball in X with radius $\sqrt{\mathcal{F}(g)/\gamma}$. As X is a Hilbert space, this ball is weakly compact (see, e.g., Theorem 7 in [Lax02]), thus also weakly sequentially compact. Thus all we need to do to assure existence of minimum is to prove that $\mathcal{F}$ is weakly sequentially lower semicontinuous. (Perhaps confusingly, weak continuity is stronger property than continuity, so although $\mathcal{F}$ is continuous, that is not enough.)

So let us consider a sequence of functions f_n weakly converging to some f. From the basic properties of RKHS's (Theorem 5.2) we know that the evaluation functionals $\Lambda_i : f \mapsto f(x_i)$ are continuous (and, obviously, linear). Thus, by definition of weak convergence, $\lim_{n\to\infty} \Lambda_i(f_n) = \Lambda_i(f)$. It follows, that the data part of the functional $\mathcal{F}$ is weakly sequentially lower semicontinuous.

The second part of $\mathcal{F}$ is a square of a norm. A norm in any Banach space is weakly sequentially lower semicontinuous (see, e.g., Theorem 5 in [Lax02]). It easily follows that so is the square of the norm and thus $\mathcal{F}$ itself. Consequently, $\mathcal{F}$ attains its minimum.

To show uniqueness of the minimum, we will use Lemma 6.3. The functional $\mathcal{F}$ is actually strictly convex, but verifying strict quasiconvexity is somewhat easier.

The first part of $\mathcal{F}$ is a sum of N elements, each of which is a convex functional, as the (real) function $z \mapsto \frac{1}{N}(z - y_i)^2$ is convex.

For the second part, let f, g be two distinct elements of X, $s \in (0,1)$ and $t = 1 - s$. Put $M = \max\{\|f\|_k^2, \|g\|_k^2\}$. We have

$$\begin{aligned}\langle sf + tg, sf + tg\rangle &= s^2\langle f, f\rangle + st(\langle f, g\rangle + \langle g, f\rangle) + t^2\langle g, g\rangle \\ &\le M^2(s^2 + 2st + t^2) \\ &= M^2\end{aligned}$$

(we have used Cauchy inequality to estimate $\langle f, g\rangle$ and $\langle g, f\rangle$). This proves quasiconvexity of the mapping $f \mapsto \|f\|_k^2$, to obtain strict quasiconvexity, we analyze when equality is achieved in the above formula. We need to have $\|f\|_k = \|g\|_k$, moreover (to have equality in the Cauchy inequality) f and g are collinear. This implies $f = \pm g$; $f = -g$ does not produce equality and $f = g$ is false by assumption.

Altogether, $\mathcal{F}$ is a sum of a convex functional and a strictly quasiconvex functional, so it is strictly quasiconvex itself. Thus we may apply Lemma 6.3 (with $C = X$).

So we have shown that there exists a unique f_0 at which the minimum of $\mathcal{F}$ is attained. As a final step we derive the form of f_0. Lemma 6.4 implies that for each h we have $\mathrm{D}_h\mathcal{F}(f_0) = 0$. A routine computation yields

$$\mathrm{D}_h\mathcal{F}(f) = \frac{1}{N}\sum_{i=1}^{N} 2(f(x_i) - y_i)h(x_i) + \gamma(\langle f, h\rangle + \langle h, f\rangle)\,.$$

We put $h(y) = k(y, x)$. The reproducing property yields $\langle f_0, h\rangle = f_0(x)$, and so $\langle h, f_0\rangle = \overline{f_0(x)} = f_0(x)$ (as f_0 was assumed to be real). Putting $c_i = -(f_0(x_i) - y_i)/(\gamma N)$ we obtaine the desired form (recall that k is symmetric)

$$f_0(x) = \sum_{i=1}^{N} c_i k(x, x_i)\,.$$

And we easily obtain the form of c_i: $c = (K[x] + \gamma N I)^{-1} y$, where $K[x]_{i,j} = k(x_i, x_j)$ is Gramm matrix of kernel k with respect to vector x, I is $N \times N$ identity matrix, c, y are vectors. □

Notice that here we consider only the simplest case where existence and uniqueness of minima can be proven easily. For more detail see for example [SchSl02] or [DRCDO05, Ku04], where (as opposed to proof presented), theory of inverse problems is used.

We remark that the same proof comes through also when the error part of the functional is $\frac{1}{N}\sum_{i=1}^{N} V(f(x_i) - y_i)$ for any continuous convex function V, however derivation of form of the solution will use derivative of V.

The form of solution was derived for example in [GJP95], we will use it in Section 8. Solutions have been derived also for schemas where uniqueness cannot be proven (regularization part Φ is not strictly quasiconvex). In that case we have to add functions from null space of Φ to the solution (see e.g. [GJP95]).

The solution derived above is very nice, since it resembles a neural network with $k(x, x_i)$ as the activation functions parameterized by the data points x_i.

The problem of the number of hidden units being too large to be implemented can be solved by variable-basis approximation using the obtained shape of the activation functions (see [KuSa05b]). Another approach is proposed and discussed in Section 7.

7 Randomized Pruning

In this section we will show how to combine results of algorithmic sections and kernel methods. We saw how to apply regularization to find a function f that strikes a good balance between fitting the given data $z = \{(x_i, y_i)\}_{i=1}^{N}$ and possessing good global properties – we find f_0 so that $\mathcal{F}(f_0)$ is as small as possible ($\mathcal{F}$ is given by (6.2)). Theorem 6.1 claims that the (unique) optimal function with respect to these criteria is of a simple form (6.4), the only drawback being that the number of terms in the sum is equal to the number of data – which may be too many. We will show, how to "prune" this expression to get an expression using lower number of terms.

The basic idea is simple. We approximate the minimizing function f_0 using Algorithm 2.2. We only need to show that this does not change the value of $\mathcal{F}$ too much (here we will use again, that we are searching for an approximating function as a member of a certain Hilbert space). The regularization paradigm then suggests, that we will get a reasonable solution to the original problem.

Theorem 7.1 (Randomized pruning of kernel-based neural networks) *Let X be an RKHS with kernel k and norm $\|\cdot\|_k$ and according to Theorem 6.1 let $f_0 = \sum_{i=1}^N c_i k(x, x_i)$ be the unique solution of minimization problem*

$$\mathcal{F}(f) = \frac{1}{N}\sum_{i=1}^N (f(x_i) - y_i)^2 + \gamma\|f\|_k^2$$

based on data $z = \{(x_i, y_i)\}_{i=1}^N \subseteq \mathbb{R}^d \times \mathbb{R}$ and some a-priori knowledge. Then for any $\delta > 0$ there is $n = O(\frac{1}{\delta^2})$ for which we can find

$$g_{apx}^n = \tfrac{1}{n}\sum_{j=1}^n k(x, x'_j),$$

so that

$$\sqrt{\mathcal{F}(g_{apx}^n)} \leq \sqrt{\mathcal{F}(f_0)} + \delta\,.$$

Each x'_j $(j = 1, \ldots, n)$ is one of x_i $(i = 1, \ldots, N)$ from the data z.

Proof: We use Algorithm 2.2 to obtain g_{apx}^n. We will utilize our analysis of the algorithm to derive the claim. To do so we introduce an auxiliary norm on X:

$$\|g\|_{aux} = \sqrt{\frac{1}{N}\sum_{i=1}^N (g(x_i))^2 + \gamma\|g\|_k^2}\,.$$

This is indeed a norm, and it corresponds to the scalar product

$$\langle g, h\rangle_{aux} = \frac{1}{N}\sum_{i=1}^N g(x_i)h(x_i) + \gamma\langle g, h\rangle_k\,.$$

For every $g \in X$ we have $\|g\|_{aux} \geq \sqrt{\gamma}\|g\|_k$. Consequently, a Cauchy sequence in $\|\cdot\|_{aux}$ is also Cauchy in $\|\cdot\|_k$, thus the completeness of $(X, \|\cdot\|_k)$ implies completeness of $(X, \|\cdot\|_{aux})$ – so we again have a Hilbert space. Consequently, by Corollary 2.3 we may efficiently obtain a simpler function g_{apx}^n so that $\|f_0 - g_{apx}^n\|_{aux} < \delta$ and g_{apx}^n uses $n = O(1/\delta^2)$ terms in the sum. Now by the triangle inequality for $\|\cdot\|_k$ and for the $\mathcal{L}_2$ norm we get that

$$\begin{aligned}
\sqrt{\mathcal{F}(g^n_{apx})} =& \sqrt{\sum_i (g^n_{apx}(x_i) - y_i)^2 + \gamma \|g^n_{apx}\|^2_k} \\
=& \sqrt{\sum_i \big((g^n_{apx}(x_i) - f_0(x_i)) + (f_0(x_i) - y_i)\big)^2 + \gamma \|(g^n_{apx} - f_0) + f_0\|^2_k} \\
\leq& \sqrt{\sum_i (g^n_{apx}(x_i) - f_0(x_i))^2 + \gamma \|g^n_{apx} - f_0\|^2_k} \\
&+ \sqrt{\sum_i (f_0(x_i) - y_i)^2 + \gamma \|f_0\|^2_k} \\
=& \|f_0 - g^n_{apx}\|_{aux} + \sqrt{\mathcal{F}(f_0)} \\
\leq& \sqrt{\mathcal{F}(f_0)} + \delta \,.
\end{aligned}$$

□

We see that $\mathcal{F}(g^n_{apx})$ is close to the optimal value $\mathcal{F}(f_0)$, so g^n_{apx} is a reasonable function to be learnt from the original data assuming that $\mathcal{F}$ is the right measure.

In the following remark we will try to clarify the quality of g^n_{apx} as an approximation to the original underlying function.

Remark 7.2 (Kernel-based networks with feasible number of units) *Let $z = \{(x_i, y_i)\}_{i=1}^N \subseteq \mathbb{R}^d \times \mathbb{R}$ be given data sampled from an unknown underlying function f_{orig}, (that is, $f_{orig}(x_i) = y_i$ for $i = 1, \ldots, N$), let X be an RKHS with kernel k and norm $\|\cdot\|_k$ derived from a-priori knowledge on f_{orig}, in particular we assume that $f_{orig} \in X$.*

Let again the functional $\mathcal{F}(f)$ be given by

$$\mathcal{F}(f) = \frac{1}{N} \sum_{i=1}^N (f(x_i) - y_i)^2 + \gamma \|f\|^2_k$$

for a function $f \in X$ ($\gamma > 0$ is a constant). Let $f_0 \in X$ be the unique function that minimizes $\mathcal{F}$; according to Theorem 6.1 it is of the form

$$f_0(x) = \sum_{i=1}^N c_i k(x, x_i)\,, \tag{7.1}$$

for some real c_i's.

Let $\varepsilon_0 > 0$ be such that

$$\|f_{orig} - f_0\|_k < \varepsilon_0\,.$$

If our choice of X and γ capture well the properties of f_{orig} the we can assume that ε_0 is small.

Then we can claim the following estimate

$$\|g^n_{apx} - f_{orig}\|_k < Q^{1/2} \cdot \Big(\frac{\rho + \varepsilon_0}{\sqrt{n}} + \varepsilon_0\Big).$$

Here $g^n_{apx} = \frac{1}{n}\sum_{j=1}^n k(x, x'_j)$, each x'_j ($j = 1, \ldots, n$) is one of x_i ($i = 1, \ldots, N$) from the data z; $Q > 0$, ρ is a positive constant depending on X and f_{orig} but not on n.

Now it is appropriate to comment on the constant ρ that appears in the above bound. By [DDGS93](upon which Corollary 2.3 is based) ρ is defined so that it holds: for every $g \in \mathcal{G}$ (here $\mathcal{G} = \{k(\cdot, x_1), \ldots, k(\cdot, x_N)\}$ we have $\|g - f_0\|_k \leq \rho$ which can be further expressed in terms of $\mathcal{G}$-variation ($\|f_0\|_{\mathcal{G}} \leq \sum_{i=1}^{N} |c_i|$) and $s_{\mathcal{G}} = \max k(x_i, x_i)$. Heuristically ρ expresses how good our chosen kernel-based approximation schema is for approximating given f_{orig} and how good was our a-priori knowledge of f_{orig} that helped us create kernel k and further also the functional $\mathcal{F}$.

The proposed method seems to be of practical interest, in cooperation with Petra Vidnerov we work on experiments showing applicability in practical settings.

8 Specific Types of Kernels

In this section we try to come closer to practical results by presenting a few specific examples of kernels used in regularized minimization and mainly by proposing their natural combinations that may aid to tailor regularized minimization schemas better to specific situations.

8.1 Simple Kernels

Lef f be an $\mathcal{L}^1$ function on $\mathbb{R}^d$. In [GJP95] a special stabilizer based on the Fourier Transform was proposed:

$$\Phi_K(f) = \int_{\mathbb{R}^d} \frac{|\hat{f}(s)|^2}{\hat{K}(s)} \, \mathrm{d}m_d(s),$$

where $\hat{K} : \mathbb{R}^d \to \mathbb{R}_+$ is a symmetric function ($\hat{K}(s) = \hat{K}(-s)$) tending to zero as $\|s\| \to \infty$, $\hat{f}$ is Fourier transform of f and m_d is the normalized d-dimensional Lebesgue measure (m_d on $\mathbb{R}^d$ is definded as $\mathrm{d}m_d(x) = (2\pi)^{-d/2} \, \mathrm{d}\lambda(x)$). In this setting $1/\hat{K}$ is a low-pass filter.

Thus the functional $\mathcal{F}$ to be minimized is of the form:

$$\mathcal{F}(f) = \mathcal{E}_z(f) + \Phi_K(f) = \frac{1}{N} \sum_{i=1}^{N} (f(x_i) - y_i)^2 + \gamma \int_{\mathbb{R}^d} \frac{|\hat{f}(s)|^2}{\hat{K}(s)} \, \mathrm{d}m_d(s),$$

where $\gamma \in \mathbb{R}^+$. Now we show how to build an RKHS corresponding to the regularization part of our functional:

Let us define

$$k(x, y) = K(x - y) = \int_{\mathbb{R}^d} \hat{K}(t) e^{it \cdot x} e^{-it \cdot y} \, \mathrm{d}m_d(t).$$

For $k \in \mathcal{S}(\mathbb{R}^{2d})$ symmetric positive definite we obtain an RKHS X (using the classic construction, see [SchSl02], [Gi98], [Wa90]). We put $\langle f, g \rangle_X = \int_{\mathbb{R}^d} \frac{\hat{f}(s)\overline{\hat{g}(s)}}{\hat{K}(s)} \, \mathrm{d}m_d(s)$ and obtain the norm

$$\|f\|_X^2 = \int_{\mathbb{R}^d} \frac{|\hat{f}(s)|^2}{\hat{K}(s)} \, \mathrm{d}m_d(s).$$

This enables us to put X to be the completion of the set $\mathrm{span}\{k(x, .), x \in \mathbb{R}^d\}$ in the above norm. It is easy to check the reproducing property of k, resp. K on X, that is $\langle f(x), K(x - y) \rangle_X = f(y)$.

Special types of reproducing kernels and following RKHS are for example the well known

- Gaussian kernel $k_1(x, y) = e^{-\|x-y\|^2}$ with Fourier transform $\hat{K_1}(s) = e^{-\frac{\|s\|^2}{2}}$
or in one dimension

- kernel $k_2(x,y) = e^{-|x-y|}$ with Fourier transform $\hat{K_2}(s) = (1+s^2)^{-1}$.

The norm for this RKHS is of the form $\|f\|_{k_2}^2 = \int \frac{|\hat{f}|^2}{(1+s^2)^{-1}} = \|f\|_2^2 + \|f'\|_2^2$. So we see we obtain a Sobolev space $\mathcal{W}^{1,2}$. These and many more specific instances of kernels are presented for example in [SchWe06]. Here also feature maps for deriving specifically tailored kernels are discussed.

8.2 Composite Kernels

This section discusses two types of composite kernels proposed by the author and proposes their possible use in practical situations. Experiments have been done by Petra Vidnerov, se for example [KS06].

Sum of Kernels Here we will consider a more sophisticated type of kernels – sum of kernels introduced in Section 5.2. We will study two cases. First suppose that a-priori knowledge or analysis of data suggests to look for a solution in the form of a sum of two functions (for example data is generated from function influenced by two sources differing in frequency). We will use a kernel summed of two parts (employing Theorem 5.4) corresponding to high and low frequencies, in the easiest case two Gaussians of different widths:

$$k(x,y) = k_1(x,y) + k_2(x,y) = e^{-\frac{\|x-y\|^2}{d_1}} + e^{-\frac{\|x-y\|^2}{d_2}}$$

In this case we will consider regularized minimization schema of the form:

$$\mathcal{F}(f) = \frac{1}{N}\sum_{i=1}^{N}(f(x_i)-y_i)^2 + \gamma \int_{\mathbb{R}^d} \frac{|\hat{f}(s)|^2}{(\widehat{K_1+K_2})(s)}\, \mathrm{d}m_n(s). \tag{8.1}$$

Since we operate in an RKHS we can employ Representer theorem (for this case the simplified version of it 6.1) and obtain solution in the form of

$$f_0(x) = \sum_{i=1}^{N} c_i \left(e^{-\frac{\|x-x_i\|^2}{d_1}} + e^{-\frac{\|x-x_i\|^2}{d_2}} \right). \tag{8.2}$$

This schema has been tested by Petra Vidnerov in [KS06] on data from [Proben], (see table 8.1, that shows also experiments done on product kernel networks discussed in the next paragraph). Figure 8.1 shows sum kernel as derived for specific task of `cancer1` from [Proben].

The experiments proved the schema to be promising for special types of tasks. Further experiments on improved schema (see section 7) are running.

Another conceivable task would be to approximate data with different distribution in the input space. Here again sum of kernels might be helpful if we use different kernels for different parts of the input space.

First, we present the following auxiliary lemma from [Ar50].

Lemma 8.1 (Restriction of kernel [Ar50]) *Let X be an RKHS of real-valued functions on Ω with k as kernel. Then function k_A defined by*

$$k_A(x,y) = \begin{cases} k(x,y) & \text{if } x,y \in A, \\ 0 & \text{otherwise;} \end{cases}$$

is a kernel for a space $X_A(\Omega) = \{f_A; f \in X \text{ and } f_A(x) = f(x) \text{ if } x \in A \text{ and } f_A(x) = 0 \text{ otherwise}\}$.

	RN		SKRN$_1$		PKRN		SKRN$_2$	
Task	E_{train}	E_{test}	E_{train}	E_{test}	E_{train}	E_{test}	E_{train}	E_{test}
cancer1	2.28	**1.75**	0.00	1.77	2.68	1.81	2.11	1.93
cancer2	1.86	3.01	0.00	**2.96**	2.07	3.61	1.68	3.37
cancer3	2.11	2.79	0.00	**2.73**	2.28	2.81	1.68	2.95
card1	8.75	**10.01**	8.81	10.03	8.90	10.05	8.55	10.58
card2	7.55	**12.53**	0.00	12.54	8.11	12.55	7.22	13.03
card3	6.52	12.35	6.55	**12.32**	7.01	12.45	6.22	12.86
diabetes1	13.97	16.02	14.01	**16.00**	16.44	16.75	12.92	16.66
diabetes2	14.00	**16.77**	13.78	16.80	15.87	18.14	13.64	17.33
diabetes3	13.69	16.01	13.69	**15.95**	16.31	16.62	12.85	16.34
flare1	0.36	0.55	0.35	**0.54**	0.36	**0.54**	0.35	0.59
flare2	0.42	0.28	0.44	**0.26**	0.42	0.28	0.41	0.28
flare3	0.38	0.35	0.42	**0.33**	0.40	0.35	0.38	0.34
glass1	3.37	6.99	2.35	**6.15**	2.64	7.31	2.56	6.78
glass2	4.32	7.93	1.09	**6.97**	2.55	7.46	3.27	7.29
glass3	3.96	7.25	3.04	**6.29**	3.31	7.26	3.48	6.44
heart1	9.61	**13.66**	0.00	13.91	9.56	13.67	9.51	13.79
heart2	9.33	13.83	0.00	**13.82**	9.43	13.86	8.52	14.31
heart3	9.23	15.99	0.00	**15.94**	9.15	16.06	8.30	16.75
hearta1	3.42	4.38	0.00	**4.37**	3.47	4.39	3.20	4.45
hearta2	3.54	4.07	3.51	**4.06**	3.28	4.29	3.17	4.34
hearta3	3.44	4.43	0.00	4.49	3.40	4.44	3.37	**4.40**
heartac1	4.22	**2.76**	0.00	3.26	4.22	**2.76**	3.68	3.37
heartac2	3.50	3.86	0.00	**3.85**	3.49	3.87	2.99	3.97
heartac3	3.36	**5.01**	3.36	**5.01**	3.26	5.18	3.14	5.13
heartc1	9.99	16.07	0.00	**15.69**	10.00	16.08	6.50	16.07
heartc2	12.70	**6.13**	0.00	6.33	12.37	6.29	11.06	6.69
heartc3	8.79	12.68	0.00	12.38	8.71	12.65	9.91	**11.74**
horse1	7.35	**11.90**	0.20	**11.90**	14.25	12.45	7.66	12.62
horse2	7.97	15.14	2.84	**15.11**	12.24	15.97	6.84	15.70
horse3	4.26	**13.61**	0.18	14.13	9.63	15.88	8.56	15.24
soybean1	0.12	0.66	0.11	0.66	0.13	0.86	0.12	**0.64**
soybean2	0.24	**0.50**	0.25	0.53	0.23	0.71	0.19	0.54
soybean3	0.23	0.58	0.22	**0.57**	0.21	0.78	0.15	0.72

Table 8.1: Comparisons of errors on training and testing set for RN with Gaussian kernels and SKRN and PKRN.

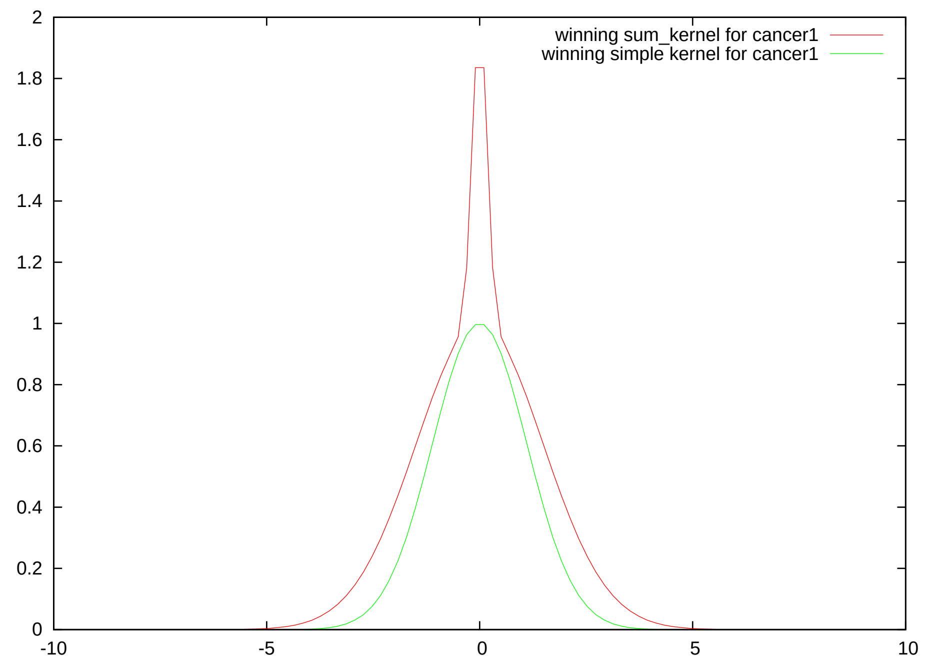

Figure 8.1: An example of the sum kernel (an optimal combination of widths for `cancer1 data`).

We may use this lemma for different sets $A \subseteq \Omega$. Then we can apply Theorem 5.4 for kernels gained in this way. Consequently, our kernels may look as follows:

Choose constants $d_1, \ldots, d_r > 0$ and partition $\mathbb{R}^d$ to sets $A_1, \ldots, A_r$. Then define

$$k_j(x,y) = K_j(x-y) = \begin{cases} e^{-d_j\|x-y\|^2} & x, y \in A_j \\ 0 & \text{otherwise.} \end{cases}$$

The outcomming sum kernel then is $k(x,y) = \sum_{j=1}^{r} k_j(x,y)$ and we can chose the basic regularization schema

$$\mathcal{F}(f) = \frac{1}{N}\sum_{i=1}^{N}(f(x_i)-y_i)^2 + \gamma \int_{\mathbb{R}^d} \frac{|\hat{f}(s)|^2}{\widehat{\sum K_j}(s)} \,\mathrm{d}m_d(s).$$

Now by 6.1 again we obtain a form in which we will expect the solution:

$$f_0(x) = \sum_{i=1}^{N} c_i \sum_{j=1}^{r} k_j(x-x_i).$$

Experimets towards this end have been done by Petra Vidnerov citePdiz. The approach doesn't provide significantly better errors but the errors are comparable and in some cases lower. However, it proves to be clearly far better for big data sets providing significantly lower time complexity. This is enabled in a simple way - this method in fact divides the big task into several small ones.

Let us proceed with another type of composite kernels:

Product of Kernels We consider the product of kernels introduced in Section 5.3. Suppose that a-priori knowledge of our data suggests to look for the solution as a member of product of two function spaces. In one dimension the data may be clustered faraway thus being suitable for approximation via narrow Gaussian kernels, in the other dimension the data is smooth, hence we will use broader Gaussian kernel. Employing Theorem 5.5 we obtain a kernel for the product space of the form: $k((x_1, x_2), (y_1, y_2)) = k_1(x_1, y_1) \cdot k_2(x_2, y_2) = e^{-\|x_1-y_1\|^2} \cdot e^{-\|x_2-y_2\|}$, where $x_1, y_1 \in \Omega_1$, $x_2, y_2 \in \Omega_2$.

Regularized minimization schema in this case is of the form:

$$\mathcal{F}(f) = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2 + \gamma \int_{\mathbb{R}^d} \frac{|\hat{f}(s)|^2}{\widehat{k_1 k_2}(s)}\, \mathrm{d}m_d(s). \tag{8.3}$$

Taking advantage of this being an RKHS we have the form of the solution to such a type of minimization:

$$f_0(x_1, x_2) = \sum_{i=1}^{N} c_i e^{-\|x_1 - x_{i,1}\|^2} \cdot e^{-\|x_2 - x_{i,2}\|}. \tag{8.4}$$

Product kernel network of this type was used to predict the flow rate on the Czech river Ploučnice [KS06]. Our goal was to predict the current flow rate from the flow rate and total rainfall from the previous date, i.e., we were approximating a function $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$.

We have chosen the Gaussian function $e^{-\left(\frac{\|c-x\|}{d_i}\right)^2}$ for both kernel functions (k_1 and k_2), the kernels differ in the width d_i of the Gaussian. All parameters γ, d_1 and d_2 were estimated by crossvalidation. Outcomes of the experiments done by Petra Vidnerov are presented in Figure 8.2 and in Table 8.2.

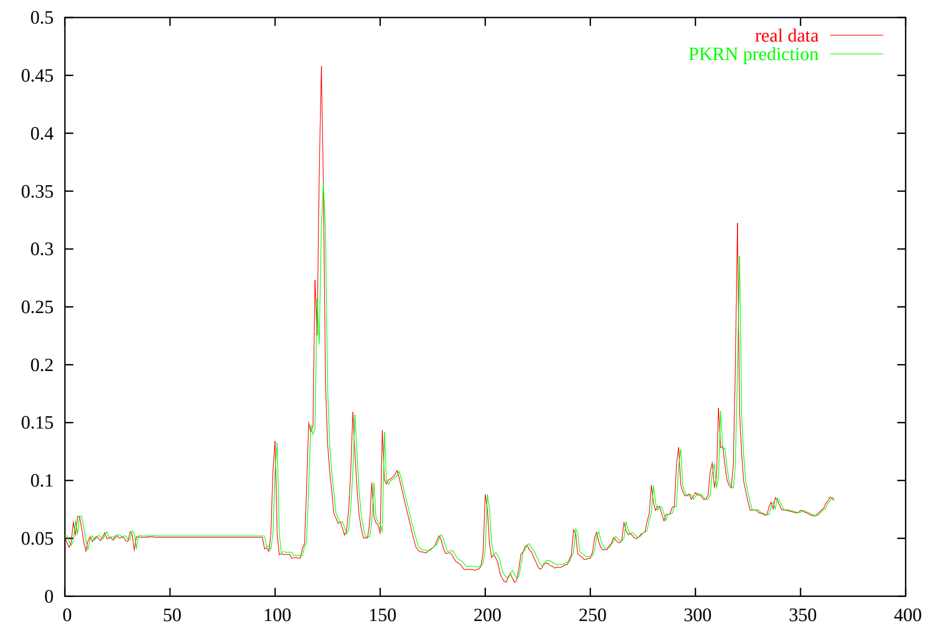

Figure 8.2: Prediction of flow rate on the river Ploucnice.

The Table 8.2 compares the resulting errors of Product Kernel Regularization Network, Regularization Network (for detailed discussion see [K06]) and *conservative predictor*. Conservative predictor is a predictor saying that the value will be the same as it was yesterday, and in spite of its simplicity it is very successful on some tasks, including this one. We can see that the PKRN overperforms both

	PKRN	**RN**	**CP**
E_{train}	0.057	0.008	0.093
E_{test}	0.048	0.056	0.054
network parameters	$\gamma = 10^{-4}$ $d_1 = 0.8$ $d_2 = 1.9$	$\gamma = 1.48 \cdot 10^{-5}$ $d = 0.5$	

Table 8.2: Comparison of error of Product Kernel Regularization Network (PKRN), Regularization Network (RN) and Conservative Predictor (CP) on training set and testing set.

the Regularization Network and Conservative Predictor. We can also see that PKRN shows higher parameter γ which suggest better generalisation.

Results of Section 7 on pruning were not used in these experiments. They are, however, the topic of our current cooperation with Petra Vidnerov.

9 Conclusion

Based on theoretical results of [DDGS93]and their modifications done by the authors we proposed probabilistic algorithms for obtaining trained neural networks. We high-level analysed time complexity of these algorithms and compared them to known iterative algorithms in specific situations. We showed that probabilistic algorithm based on known precise approximation is very well suited for pruning a too big neural network and we derived theoretical bounds on error such introduced.

We have shown how to use randomized algorithm in the specific case of neural networks derived from kernel-based regularized minimization schemas.

We overviewed briefly basic theory regarding RKHS spaces and regularized minimization that is applicable to our objective. Based on this theory and on results of algorithmic sections we show how to overcome the intrinsic problem of regularization-inferred neural networks – i.e. too high number of hidden units. We have derived bounds on errof of approximation by so derived networks with a fixed (smaller) number of units.

We showed several special types of kernels used widely in practice and their combinations proposed in [KS06]. We added a few illustrations of experiments done on these schemas by Petra Vidnerov. Unfortunately we cannot bring experiments on networks pruned by randomized algorithm as these are currently running.

Bibliography

[Ar50] N. Aronszajn: *Theory of Reproducing Kernels*, Transactions of the AMS, **68** (1950), no. 3, 337–404.

[Ba93] A. R. Barron: *Universal approximation bounds for superposition of a sigmoidal function*, IEEE Transactions on Information Theory, **39** (1993), 930–945.

[BoVa04] S. Boyd, L. Vandenberghe: *Convex Optimization*, Cambridge University Press, (2004).

[CuSm01] F. Cucker, S. Smale: *On the Mathematical Foundations of Learning.* Bulletin of the American Mathematical Society **39** (2001), 1–49.

[Dn71] J. W. Daniel: *The Approximate Minimization of Functionals*, Prentice-Hall, (1971).

[DDGS93] C. Darken, M. Donahue, L. Gurvits, E. Sontag: *Rate of Approximation Results Motivated by Robust Neural Network Learning*, Proceedings of the 6th Annual ACM Conference on Computational Learning Theory, Santa Cruz, CA, (1993), 303–309.

[DRCDO05] E. De Vito, L. Rosasco, A. Caponnetto, U. De Giovannini, F. Odone: *Learning from Examples as an Inverse Problem*, MIT Press, The Journal of Machine Learning Research, **6** (2005), 883–904.

[Gi98] F. Girosi: *An Equivalence between Sparse Approximation and Support Vector Machines*, Neural Computation **10** (1998), 1455–1480 (A.I. Memo No. 1606, MIT, 1997).

[GJP95] F. Girosi, M. Jones, T. Poggio: *Regularization Theory and Neural Networks Architectures*, Neural Computation, **7** (1995), 219–269.

[Js03] J. Jost: *Postmodern Analysis*, Springer Verlag, (2003).

[Jo92] L. K. Jones: *A Simple Lemma on Greedy Approximation in Hilbert Space and Convergence Rates for Projection Pursuit Regression and Neural Network Training*, Annals of Statistics, **20** (1992), no. 1., 608–613.

[KiWa71] G. Kimeldorf, G. Wahba: *Some results on Tchebycheffian spline functions*, J. Math. Anal. Applic., **33** (1971), 82–95.

[K06] P. Kudová: *Learning with Regularization Networks*, PhD. Thesis, MFF UK, Prague, (2006).

[KS06] P. Kudová, T. Šámalová: *Sum and Product Kernel Regularization Networks*, Lecture Notes in Artificial Intelligence, **4029** (2006), 56–65.

[KS05b] P. Kudová, T. Šámalová: *Product Kernel Regularization Networks*, Icannga 2005, Coimbra, (2005).

[Ku04] V. Krkov: *Learning from Data as an Inverse Problem*, proceedings of COMPSTAT 2004, Prague (CZ), Computational Statistics, Physica Verlag, (2004), 1377–1384.

[KKK97] V. Kůrková, P.C. Kainen, V. Kreinovich: *Estimates of the Number of Hidden Units and Variation with Respect to Half-Spaces*, Neural Networks, **10** (1997), 1061–1068.

[KuSa05b] V. Kůrková, M. Sanguinetti: *Learning with generalization capability by kernel methods with bounded complexity*, Journal of Complexity **21** (2005), 350–367.

[Lax02] P. D. Lax: *Functional Analysis*, J. Wiley, (2002).

[LeTa91] M. Ledoux, M. Talagrand: *Probability in Banach spaces*, Springer-Verlag, Berlin, (1991).

[Mk96] Y. Makovoz: *Random approximants and neural networks*, Journal of Approximation Theory **85** (1996), 98–109.

[Ps81] G. Pisier: *Remarques sur un resultat non publi'e de B. Maurey*, in Seminaire D'Analyse Fonctionnelle, 1980-1981, 'Ecole Polytechnique, Centre de Math'ematiques, Palaiseau, France (1981).

[PaSa93] J. Park, I. W. Sandberg: *Approximation and radial-basis-function networks*, Neural Computation, **5** (1993), 305–316.

[Proben] L. Prechelt: *PROBEN1 – A Set of Benchmarks and Benchmarking Rules for Neural Network Training Algorithms*, Universitaet Karlsruhe, 21/94, (1994).

[PoSm03] T. Poggio, S. Smale: *The Mathematics of Learning: Delaing with Data*, Notices of the AMS, **50** (2003), no. 5, 536–544.

[SchWe06] R. Schaback, H. Wendland: *Kernel techniques: From machine learning to meshless methods*, Acta Numerica, (2006), 543–639.

[SchSl02] B. Schölkopf, A.J. Smola: *Learning with Kernels*, MIT Press, Cambridge, Massachusetts, (2002).

[S04a] T. Šidlofová: *Existence and Uniqueness of Minimization Problems with Fourier Based Stabilizers*, Compstat 2004, Prague, (2004).

[TiAr77] A. N. Tikhonov and V. Y. Arsenin. Solutions of Ill-posed Problems. W. H. Winston, Washington, D.C., (1977).

[Wa90] G. Wahba: *Spline Models for Observational Data*, Series in Applied Mathematics, **59**, SIAM, Philadelphia, (1990).

[We00] S. Weinzierl: *Introduction to Monte Carlo methods*, arXiv:hep-ph/0006269v1, (2000).