following sequential learning rule

$$f_i = f_{i-1} + \mathbf{Gain}(i)e(i) \tag{1–23}$$

where f_i denotes the estimate of the mapping at time i and $\mathbf{Gain}(i)$ is a function in general. This sequential learning, first studied by Goodwin and Sin [1984] for linear filters, is very attractive in practice since the current estimate consists of two additive parts, namely, the previous estimate and a correction term proportional to the prediction error on new data. This unique incremental nature distinguishes our methods from all the others. Although (1–23) appears very simple, the algorithm can in fact be motivated by many different objective functions. Also, depending on the precise meanings of $\mathbf{Gain}(i)$ and $e(i)$, the algorithm can take many different forms. We explore this in detail in the subsequent chapters. This amazing feature is achieved with the underlying linear structure of the reproducing kernel Hilbert space where the algorithms exist, as discussed next.

1.4 Reproducing Kernel Hilbert Spaces

A *pre-Hilbert space* is an *inner-product space* which has an *orthonormal basis* $\{\mathbf{x}_k\}_{k=1}^{\infty}$. Let $\mathbb{H}$ be the largest and most inclusive space of vectors for which the infinite set $\{\mathbf{x}_k\}_{k=1}^{\infty}$ is a basis. Then, vectors not necessarily lying in the original inner-product space represented in the form

$$\mathbf{x} = \sum_{k=1}^{\infty} a_k \mathbf{x}_k$$

are said to be spanned by the basis $\{\mathbf{x}_k\}_{k=1}^{\infty}$; the a_k are the coefficients of the representation. Define the new vector

$$\mathbf{y}_n = \sum_{k=1}^{n} a_k \mathbf{x}_k$$

Another vector $\mathbf{y}_m$ may be similarly defined. For $n > m$, we may express the squared Euclidean distance between the vectors $\mathbf{y}_n$ and $\mathbf{y}_m$ as

$$\begin{aligned} ||\mathbf{y}_n - \mathbf{y}_m||^2 &= ||\sum_{k=1}^{n} a_k \mathbf{x}_k - \sum_{k=1}^{m} a_k \mathbf{x}_k||^2 \\ &= ||\sum_{k=m+1}^{n} a_k \mathbf{x}_k||^2 \\ &= \sum_{k=m+1}^{n} a_k^2 \end{aligned}$$

where, in the last line, we invoked the orthonormality condition. Therefore, to make the definition of $\mathbf{x}$ meaningful, we need the following to hold:

1. $\sum_{k=m+1}^{n} a_k^2 \to 0$ as both n, $m \to \infty$.
2. $\sum_{k=1}^{m} a_k^2 < \infty$

In other words, a sequence of vectors $\{\mathbf{y}_k\}_{k=1}^{\infty}$ so defined is a *Cauchy sequence*. Consequently, a vector $\mathbf{x}$ can be expanded on the basis $\{\mathbf{x}_k\}_{k=1}^{\infty}$ if, and only if, $\mathbf{x}$ is a linear combination of the basis vectors and the associated coefficients $\{a_k\}_{k=1}^{\infty}$ are square summable. From this discussion, it is apparent that the space $\mathbb{H}$ is more "complete" than the starting inner-product space. We may therefore make the following important statement:

> *An inner-product space $\mathbb{H}$ is complete if every Cauchy sequence of vectors taken from the space $\mathbb{H}$ converges to a limit in $\mathbb{H}$; a complete inner-product space is called a* Hilbert space.

A *Mercer kernel* [Aronszajn, 1950] is a continuous, symmetric, positive-definite function $\kappa : \mathbb{U} \times \mathbb{U} \to \mathbb{R}$. $\mathbb{U}$ is the input domain, a subset of $\mathbb{R}^L$. The commonly used kernels include the Gaussian kernel (1–24) and the polynomial kernel (1–25):

$$\kappa(\mathbf{u}, \mathbf{u}') = \exp(-a||\mathbf{u} - \mathbf{u}'||^2) \tag{1–24}$$

$$\kappa(\mathbf{u}, \mathbf{u}') = (\mathbf{u}^T \mathbf{u}' + 1)^p \tag{1–25}$$

Let $\mathbb{H}$ be any vector space of all real-valued functions of $\mathbf{u}$ that are generated by the kernel $\kappa(\mathbf{u}, \cdot)$. Suppose now two functions $h(\cdot)$ and $g(\cdot)$ are picked from the space $\mathbb{H}$ that are respectively represented by

$$h = \sum_{i=1}^{l} a_i \kappa(\mathbf{c}_i, \cdot)$$

and

$$g = \sum_{j=1}^{m} b_j \kappa(\widetilde{\mathbf{c}}_j, \cdot)$$

where the a_i and the b_j are expansion coefficients and both $\mathbf{c}_i$ and $\widetilde{\mathbf{c}}_j \in \mathbb{U}$ for all i and j. The *bilinear form* defined as

$$< h, g >= \sum_{i=1}^{l} \sum_{j=1}^{m} a_i \kappa(\mathbf{c}_i, \widetilde{\mathbf{c}}_j) b_j$$

satisfies the following properties:

1. **Symmetry**

$$< h, g >=< g, h >$$

2. **Scaling and distributive property**

$$< (cf + dg), h >= c < f, h > +d < g, h >$$

3. **Squared norm**

$$||f||^2 =< f, f >\geq 0$$

By virtue of these facts, the bilinear term $< h, g >$ is indeed an *inner product*. There is one additional property that follows directly. Specifically, setting $g(\cdot) = \kappa(\mathbf{u}, \cdot)$, we obtain

$$\begin{aligned} < h, \kappa(\mathbf{u}, \cdot) > &= \sum_{i=1}^{l} a_i \kappa(\mathbf{c}_i, \mathbf{u}) \\ &= h(\mathbf{u}) \end{aligned}$$

This property is known as the *reproducing property*. The kernel $\kappa(\mathbf{u}, \mathbf{u}')$, representing a function of the two vectors $\mathbf{u}$, $\mathbf{u}' \in \mathbb{U}$, is called a reproducing kernel of the vector space $\mathbb{H}$ if it satisfies the following two conditions:

1. For every $\mathbf{u} \in \mathbb{U}$, $\kappa(\mathbf{u}, \mathbf{u}')$ as a function of the vector $\mathbf{u}'$ belongs to $\mathbb{H}$.

2. It satisfies the reproducing property.

These two conditions are indeed satisfied by the Mercer kernel, thereby endowing it with the designation "reproducing kernel". If the inner-product space $\mathbb{H}$, in which the reproducing kernel space is defined, is also *complete*, then it is called a *reproducing kernel Hilbert space*, for which we use the acronym RKHS hereafter.

The analytic power of RKHS is expressed in an important theorem called the Mercer theorem. The Mercer theorem [Aronszajn, 1950, Burges, 1998] states that any reproducing kernel $\kappa(\mathbf{u}, \mathbf{u}')$ can be expanded as follows:

$$\kappa(\mathbf{u}, \mathbf{u}') = \sum_{i=1}^{\infty} \varsigma_i \phi_i(\mathbf{u}) \phi_i(\mathbf{u}') \tag{1–26}$$

where ς_i and ϕ_i are the eigenvalues and the eigenfunctions respectively. The eigenvalues are non-negative. Therefore, a mapping $\boldsymbol{\varphi}$ can be constructed as

$$\begin{aligned} &\boldsymbol{\varphi} : \mathbb{U} \rightarrow \mathbb{F} \\ &\boldsymbol{\varphi}(\mathbf{u}) = [\sqrt{\varsigma_1}\phi_1(\mathbf{u}), \sqrt{\varsigma_2}\phi_2(\mathbf{u}), ...] \end{aligned} \tag{1–27}$$

By construction, the dimensionality of $\mathbb{F}$ is determined by the number of strictly positive eigenvalues, which can be infinite in the Gaussian kernel case.

In the machine learning literature, $\boldsymbol{\varphi}$ is usually treated as the feature mapping and $\boldsymbol{\varphi}(\mathbf{u})$ is the transformed feature vector lying in the feature space $\mathbb{F}$ (which is an inner product space). By doing so, an important implication is

$$\boldsymbol{\varphi}(\mathbf{u})^T \boldsymbol{\varphi}(\mathbf{u}') = \kappa(\mathbf{u}, \mathbf{u}') \tag{1–28}$$

It is easy to check that $\mathbb{F}$ is essentially the same as the RKHS induced by the kernel by identifying $\boldsymbol{\varphi}(\mathbf{u}) = \kappa(\mathbf{u}, \cdot)$, which are the bases of the two spaces respectively. By slightly abusing the notation, we do not distinguish $\mathbb{F}$ and $\mathbb{H}$ in this book if no confusion is involved.

A concrete example helps here. Let [Cherkassky and Mulier, 1998]

$$\kappa(\mathbf{u}, \mathbf{c}) = (1 + \mathbf{u}^T \mathbf{c})^2 \tag{1–29}$$

with $\mathbf{u} = [u_1, u_2]^T$ and $\mathbf{c} = [c_1, c_2]^T$. By expressing the polynomial kernel in terms of monomials of various orders, we have

$$\kappa(\mathbf{u}, \mathbf{c}) = 1 + u_1^2 c_1^2 + 2u_1 u_2 c_1 c_2 + u_2^2 c_2^2 + 2u_1 c_1 + 2u_2 c_2$$

Therefore, the image of the input vector $\mathbf{u}$ in the feature space may be written as

$$\boldsymbol{\varphi}(\mathbf{u}) = [1, u_1^2, \sqrt{2}u_1 u_2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2]^T$$

And similarly we have

$$\boldsymbol{\varphi}(\mathbf{c}) = [1, c_1^2, \sqrt{2}c_1 c_2, c_2^2, \sqrt{2}c_1, \sqrt{2}c_2]^T$$

It is easy to verify that

$$\boldsymbol{\varphi}(\mathbf{u})^T \boldsymbol{\varphi}(\mathbf{c}) = \kappa(\mathbf{u}, \mathbf{c})$$

However, it is hard in general to explicitly express $\boldsymbol{\varphi}$ even for simple polynomial kernels, because the dimensionality of $\boldsymbol{\varphi}$ scales with $O(L^p)$ where L is the dimension of input vectors and p is the order of the polynomial kernel.

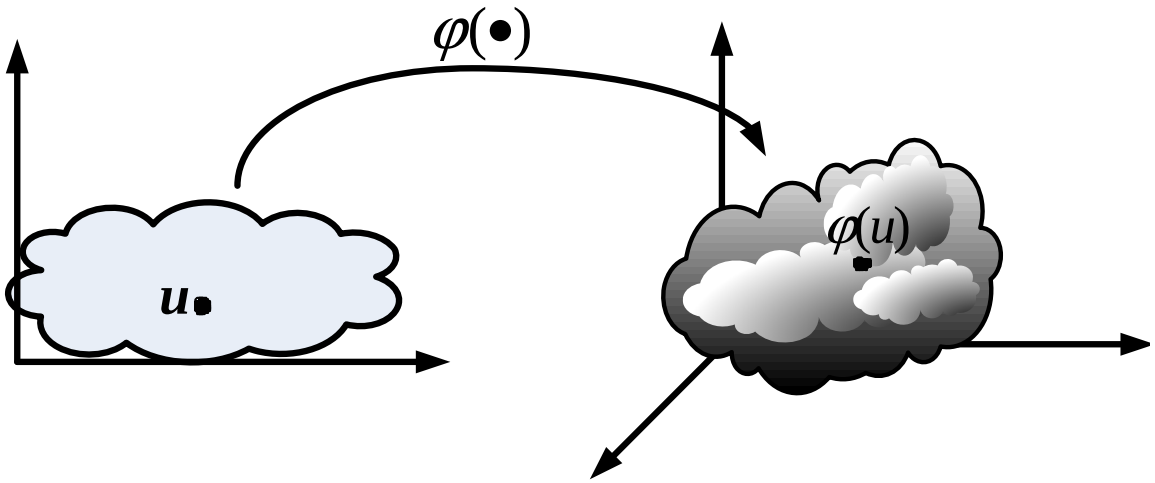

Figure 1-4. Nonlinear map $\boldsymbol{\varphi}(\cdot)$ from the input space to the feature space

1.5 Kernel Adaptive Filters

Kernel method is a powerful nonparametric modeling tool. The main idea can be summarized as follows: transform the input data into a high-dimensional feature space via a reproducing kernel such that the inner product operation in the feature space can be computed efficiently through the kernel evaluation (1–28). Then appropriate linear methods are subsequently applied on the transformed data. As long as an algorithm can be formulated in terms of inner products (or equivalent kernel evaluation), there is no need to perform computations in the high dimensional feature space. While this methodology is called the "kernel trick", we have to point out that the underlying reproducing kernel Hilbert space plays a central role to provide linearity, convexity, and universal approximation capability. Successful examples of this methodology include support vector machines, kernel principal component analysis, Fisher discriminant analysis, and many others[8].

We start by an example to show why projecting the input into a feature space helps in learning. Consider the target function of a two dimensional input $\mathbf{u} = [u_1, u_2]^T$.

$$f(u_1, u_2) = a_1 u_1 + a_2 u_2 + a_3 u_1^2 + a_4 u_2^2 \tag{1–30}$$

where a_1, a_2, a_3, a_4 are some constant coefficients. Apparently a linear system trying to approximate f by a linear combination of u_1 and u_2 could not exactly model it as written. However, by using the kernel (1–29) and its mapping $\boldsymbol{\varphi}$, we have a new representation of the input

$$(u_1, u_2) \overset{\boldsymbol{\varphi}}{\mapsto} (x_1, x_2, x_3, x_4, x_5, x_6) = (1, u_1^2, \sqrt{2}u_1 u_2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2)$$

Now f can be represented by a linear system of $(x_1, x_2, x_3, x_4, x_5, x_6)$

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = 0 \cdot x_1 + a_3 x_2 + 0 \cdot x_3 + a_4 x_4 + \frac{a_1}{\sqrt{2}} x_5 + \frac{a_2}{\sqrt{2}} x_6$$

The fact that mapping the input into a feature space can simplify the learning task has been well known for a long time in machine learning as exemplified by polynomial regression[9] and Volterra series. The problem is really how to construct this mapping. One may be tempted to add as many features as possible since it is more likely the target functions can be represented using a standard learning algorithm in high dimensional feature spaces, but this may run into the danger of *overfitting*. Overfitting is a phenomenon where a good fit to the training data is achieved but the over-learned system performs badly when making test predictions. The difficulties with high dimensional feature spaces are mainly:

1. the computational complexity explodes with the dimensionality;
2. the generalization performance degrades as the dimensionality increases.

There are two approaches to overcome the problem. One is called *feature selection* where only useful features are selected and other features are pruned so that the dimensionality of the feature space is constrained. In our previous example, apparently x_1 and x_3 are two *redundant features* which should be pruned. The other workaround is the kernel method. Since the features are only implicitly constructed and we don't need to work directly in the feature space, the explosion of computational complexity is avoided. And the overfitting problem is taken care of by the use of regularization. These properties will become clear when we develop the kernel adaptive filters in the subsequent chapters.

It has been proved [Steinwart, 2001] that in the case of the Gaussian kernel, for any continuous input-output mapping $f : \mathbb{U} \rightarrow \mathbb{R}$ and any $\varsigma > 0$, there exist parameters $\{\mathbf{c}_i\}_{i=1}^m$ in $\mathbb{U}$ and real numbers $\{a_i\}_{i=1}^m$ such that

$$||f - \sum_{i=1}^{m} a_i \kappa(\cdot, \mathbf{c}_i)||_2 < \varsigma \tag{1–31}$$

If we denote a vector $\boldsymbol{\omega}$ in $\mathbb{F}$ as

$$\boldsymbol{\omega} = \sum_{i=1}^{m} a_i \boldsymbol{\varphi}(\mathbf{c}_i)$$

then by (1–28) and (1–31), we have

$$||f - \boldsymbol{\omega}^T \boldsymbol{\varphi}||_2 < \varsigma$$

This equation implies that the linear model in $\mathbb{F}$ has the *universal approximation property*. Clearly, this property is established from the viewpoint of strict function approximation.

Furthermore, if our problem is to minimize a regularized cost function over a finite data set $\{(\mathbf{u}(i), d(i))\}_{i=1}^N$, we write

$$\min_f J(f) = \sum_{i=1}^{N} (d(i) - f(\mathbf{u}(i)))^2 + \lambda ||f||_2^2$$

It has been shown that the optimal solution can be expressed as

$$f = \sum_{i=1}^{N} a_i \kappa(\cdot, \mathbf{u}(i))$$

for suitable a_i. This result is called the *representer theorem* [Schölkopf et al., 2001]. In other words, although we did consider functions which were expansions in terms of arbitrary points $\mathbf{c}_i$ (see (1–31)), it turns out that we can always express the solution in terms of the training points $\mathbf{u}(i)$ only. Hence the optimization problem over an arbitrarily large number of variables is transformed into one over N variables, where N is the number of training points.

Recently it has also been shown that Volterra series and Wiener series can be treated just as a special case of a kernel regression framework [Franz and Schölkopf, 2006]. By formulating the Volterra and Wiener series as a linear regression in RKHS, the complexity is now independent of the input dimensionality and the order of nonlinearity.

Based on these advantages and arguments, our strategy is clear: to formulate the classic adaptive filters in RKHS such that we are iteratively solving a convex least-squares problem there. As long as we can formulate these algorithms in terms of inner products we obtain nonlinear adaptive filters which have the universal approximation property and

convexity at the same time. Convexity is a very important feature which prevents the algorithms from being stuck in local minima. (See Table 1-1)

Table 1-1. Comparison of different nonlinear adaptive filters

Algorithms	Modeling capacity	Convexity	Complexity
Linear adaptive filters	Linear only	Yes	Very simple
Hammerstein, Wiener models	Limited nonlinearity	No	Simple
Volterra, Wiener series	Universal	Yes	Very high
Time-lagged neural networks	Universal	No	Modest
Recurrent neural networks	Universal	No	High
Kernel adaptive filters	Universal	Yes	Modest
Recursive Bayesian estimation	Universal	No	Very high

In recent years, there have been many efforts of "kernelizing" adaptive filters in the literature. T.-T.Frieb and Harrison [1999] first used this idea to derive the kernel ADALINE, which is formulated as a deterministic gradient method based on all the training data (not online). Then, Kivinen et al. [2004] proposed an algorithm called NORMA by directly differentiating a regularized functional cost to get the stochastic gradient. While the derivation involves advanced mathematics, the results are actually equivalent to a kernel version of the leaky least-mean-square algorithm. At almost the same time, Engel et al. [2004] studied the case of kernel recursive least squares by utilizing the matrix inversion lemma. Later on, Liu et al. [2008] investigated the kernel least mean square algorithm and pointed out that the algorithm possesses a self-regularization property. Kernel affine projection algorithms were studied from different perspectives by Liu and Príncipe [2008b], Slavakis and Theodoridis [2008] and Richard et al. [2009]. More recently, the extended kernel recursive least squares algorithm was presented in [Liu et al., 2009], which studied the general state estimate problem in RKHS. After a decade of efforts from many researchers, kernel adaptive filtering is rapidly evolving into an important field of signal processing. This book serves to address previous works and investigate them in a unifying framework (see Figure 1-5). We present, in detail, the kernel least mean square algorithm, the kernel affine projection algorithms, the kernel recursive least squares algorithm and the extended kernel recursive least squares algorithm. Relations among

these algorithms, illustrated in Figure 1-5, will become clear when we present them in the subsequent chapters.

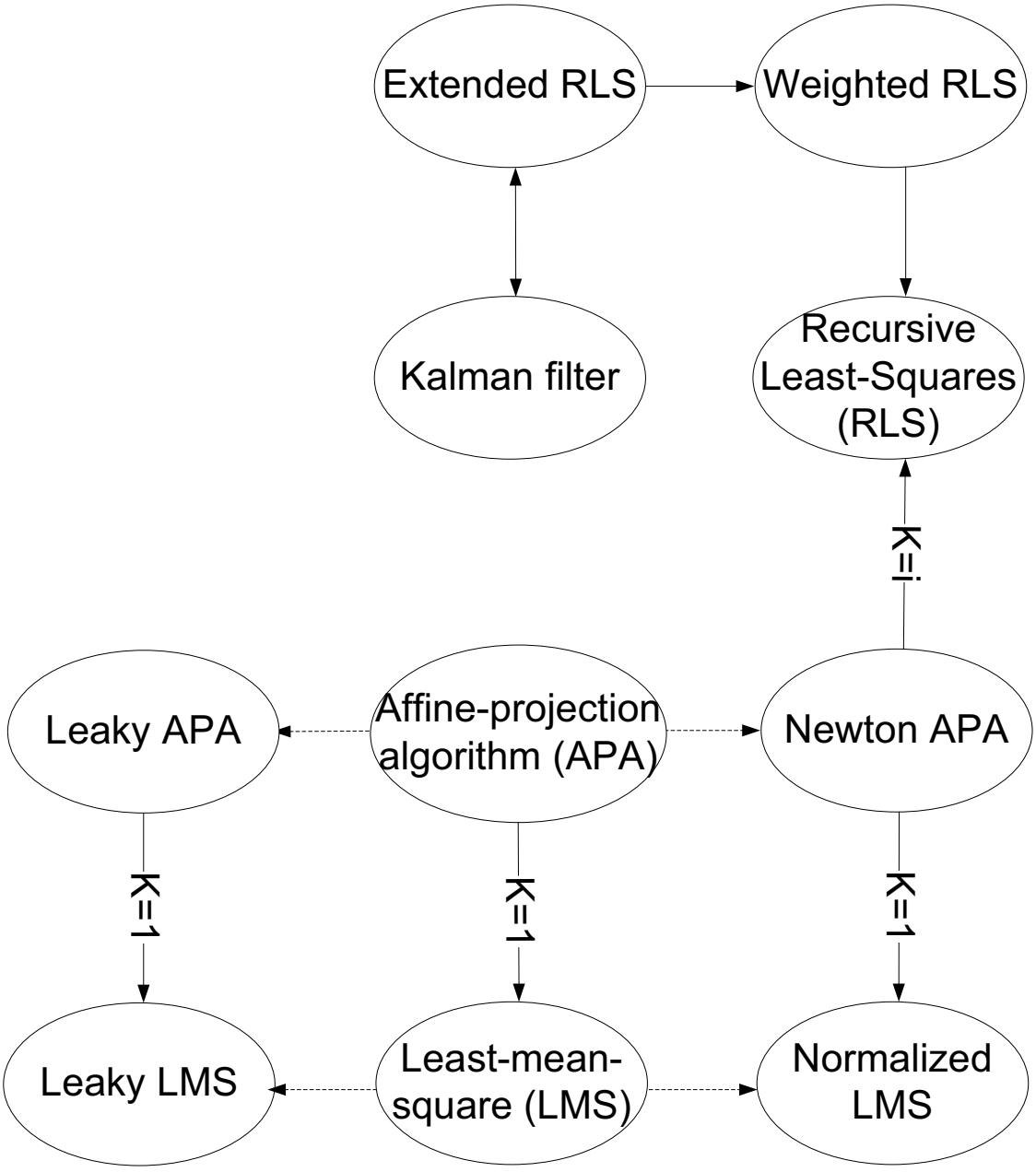

Figure 1-5. Relation between different adaptive filtering algorithms

Kernel adaptive filters provide a generalization of linear adaptive filters since these become a special case of the former when expressed in the dual space. Kernel adaptive filters exhibit a growing memory structure embedded in the filter weights. They naturally

create a growing radial-basis function network, learning the network topology and adapting the free parameters directly from data at the same time. The learning rule is a beautiful combination of the error-correction and memory-based learning, and potentially will have a deep impact on our understanding about the essence of kernel-learning theory.

Historically, most of the kernel methods use block adaptation and are computationally very expensive using a large Gram matrix of dimensionality given by the number of data points; therefore, the efficient online algorithms provide the useful flexibility for trading off performance with complexity. And in nonstationary environments, the tracking ability of online algorithms provides an extra advantage.

The combination of sequential learning and memory-based learning requires, and at the same time, enables the network to select informative training examples instead of treating all examples equally. Empirical evidence shows that selecting informative examples can drastically reduce the training time and produce much more compact networks with equivalent accuracy. Therefore, in the case of a large and redundant data set, performing kernel online learning algorithms provides a big edge over batch mode methods in terms of efficiency.

The widely used active data selection methods for kernel adaptive filters include the novelty criterion [Platt, 1991] and approximate linear dependency test [Engel et al., 2004]. Both are based on heuristic distance functions while we present a principled and unifying approach. Our criterion is based on a subjective information measure called surprise. It quantifies how informative the candidate exemplar is relative to the knowledge of the learning system. It turns out that the approximate linear dependency test is a special case and the novelty criterion is some approximation in this information theoretic framework.

1.6 Summarizing Remarks

To put the introductory material covered in this Chapter into a historical context, it is noteworthy that in a classic paper on the separability of patterns published in 1965, Cover proved that, given a nonlinearly separable pattern-classification problem, there