A Short Introduction to Hilbert Space Methods in Machine Learning

Risi Kondor

October 15, 2003

THIS IS AN INCOMPLETE DRAFT. CHECK BACK SOON FOR THE FINAL VERSION.

1 The Learning Problem

Machine Learning deals with the problem of inference from finite samples in high dimensional spaces. Typically, in **supervised learning** we are given m observations $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$, called the **training set**, and our job is to learn a rule, such that when we are presented with further x's we can do reasonably well in predicting the corresponding y's (**testing set**). The x's are called **inputs** and are assumed to live in some known space (or set) $\mathcal{X}$, unsurprisingly called the **input space**. In the simplest case $\mathcal{X}$ is just the Euclidean space $\mathbb{R}^n$ of n-dimensional vectors. The y's are called **outputs** and they come from the **output space** $\mathcal{Y}$. This framework can accomodate classification by setting $\mathcal{Y}=\{-1,+1\}$ and regression by setting $\mathcal{Y}=\mathbb{R}$. The (x, y) pairs from the training set are called **training examples** and those from the testing set are called **testing examples**.

Empirical error

The "rule" we are seeking, sometimes called the **hypothesis**, is just a function $f : \mathcal{X} \mapsto \mathcal{Y}$. To quantify how well f does on $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ we use the **empirical error**, sometimes called **empirical risk** or **training error**, defined

$$R_{\rm emp}[f] = \frac{1}{m}\sum_{i=1}^{m} L(f(x_i), y_i), \tag{1}$$

where $L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ is our **loss function** of choice. For classification the most immediate choice of loss function is the **zero-one loss**

$$L(\hat{y}, y) = \begin{cases} 1 & \text{if } \hat{y} \neq y \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

For regression we might use the **squared error loss**

$$L(\hat{y}, y) = (y - \hat{y})^2. \tag{3}$$

The choice of loss function is of critical importance not only in the evaluation, but also in the design of learning algorithms. We shall come back to this point in section **.

1.1 A simple-minded algorithm

A naive approach to learning might be to try and minimize the empirical error (1) directly. Assume that $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ are $\{-1, +1\}$ labeled points in the plane $\mathbb{R}^2$ and we are looking for $f : \mathbb{R}^2 \mapsto \{-1, +1\}$ minimizing the zero-one loss (2). If we can find a **linear decision function** $f(x) = \text{sgn}(b + w \cdot x)$ (where $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$) that correctly classifies the training data, then we are fine. If we cannot find such a function because the training data is not **linearly separable**, maybe we decide to look at second order functions $f(x) = \text{sgn}\left(b + x \cdot w + (x - x_0)^\top S\,(x - x_0)\right)$ where S is now a matrix. If that still doesn't do the job, we can go to even higher order polynomial discriminant functions, or some other family of functions. Ultimately, we might be looking at the entire class of continuous functions on $\mathbb{R}^2$ thresholded at zero, $\mathcal{F} = \left\{\, f = \text{sgn}(g) \mid g \in \mathcal{C}(\mathbb{R}^2) \,\right\}$.

Is this a sensible strategy? By choosing f from a large enough space of functions, we can certainly do really well on the training data (at least in the case that there is a **deterministic target hypothesis** f_0 relating y to x, i.e. $f_0(x) = y$). If we wish, we can always make the training error exactly zero. A simple hypothesis that does just this is

$$f(x) = \begin{cases} +1 & \text{if for some } \; i \in \{1, 2, \ldots m\} \;\; x_i = x \;\; \text{and} \;\; y_i = 1 \\ -1 & \text{otherwise.} \end{cases} \tag{4}$$

But is such a hypothesis likely to do well on future, unseen examples? Probably not. Consider a test example (x, y) where x is very close (even infinitesimally close) to one of the training examples with $+1$ output, but is not identical to any of them. Intuitively, it is clear that y is extremely likely to be $+1$, but (4) will stubbornly predict -1.

The structure of $\mathcal{X}$

The truth is, (4) is not a learning algorithm at all, it's just a label memorization procedure. Nevertheless, it does illustrate some important points. The first point is that it is impossible to learn without making some sort of assumption about what constitutes similarity in the input space $\mathcal{X}$. In particular, we need to have some a priori idea about how similarity in $\mathcal{X}$ is related to similarity of the outputs $y \in \mathcal{Y}$. We will come back to this crucial question later.

The distribution D

The second point is that we need to assume that the training examples are informative about the testing examples. The simplest way to formulate this is to assume that the target hypothesis behind the training examples is the same as the target hyothesis behind the testing examples. Additionally, the training set must be relevant to the testing set in the sense that the training inputs $x_1, x_2, \ldots, x_m$ should more or less cover the region of $\mathcal{X}$ where the test inputs will fall. If there is some remote region of input space where we have no training

examples, it will be very difficult for our algorithm to learn the shape of the discriminant function or regressor in that region.

To quantify this latter requirement we usually assume that the training and testing inputs are drawn from to the same distribution $D_{\mathcal{X}}$ over $\mathcal{X}$. The corresponding labels are determined according to the target hypothesis $f_0 : \mathcal{X} \mapsto \mathcal{Y}$, or, more generally, target probability distributions $p(y \mid x)$. This latter scenario is referred to as **learning with noise**. It is again crucial that the training and testing data share the same distributions $p(y \mid x)$ for all $x \in \mathcal{X}$. For simplicity, $D_{\mathcal{X}}$ and the conditionals $p(y \mid x)$ are usually folded into a single big distribution D over $\mathcal{X} \times \mathcal{Y}$. Our assumption is that training and testing examples are generated in an iid (independently and indentically dsitributed) fashion from this distribution D.

True error

The third and final lesson from the miserable failiure of our primitive label memorization procedure (4) is that it is important to bear in mind what our ultimate goal is: what we really want to do is to minimize our expected error on the testing set, not just the empirical error on the training set. With the notation we have just introduced, the ultimate measure of success for a hypothesis f is the so-called **true error** or **true risk**

$$R[f] = \mathbb{E}_D \left[L(f(x), y) \right],$$

where the expectation is taken over (x, y) pairs drawn from D. The art is in constructing algorithms that can (approximately) minimize the true risk, only knowing the finite training set $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$. This is called **generalization** ability. In contrast, what bad algorithms, such as (4), often end up doing is minimizing the empirical error at the expense of the true error, which is called **overfitting**. If there is a single unifying theme in learning theory, it is the constant battle against overfitting.

1.2 The formal model

The formal model

Our considerations so far give rise to the following formal model of the learning problem. Let $\mathcal{X}$ be the input space, $\mathcal{Y}$ be the output space, D be some unknown probablity distribution on $\mathcal{X} \times \mathcal{Y}$ and let $\mathcal{F}$ (called the **hypothesis space**) be a class of functions $f : \mathcal{X} \mapsto \mathcal{Y}$. Given a sample $S = (x_1, y_1), \ldots, (x_m, y_m)$ from D, the goal of learning is to find a hypothesis $f \in \mathcal{F}$ minimizing the true risk

$$R[f] = \mathbb{E}_D \left[L(f(x), y) \right]. \tag{5}$$

The advantage of working within such a well-defined model is it makes it possible to derive precise, yet general mathematical statements. In particular, we are interested in deriving theoretical guarantees of generalization performance and finding strategies for avoiding overfitting.

The main reason why we cannot minimize (5) directly is that we do not know D. Even after our algorithm has come up with a hypothesis f, we still cannot evaluate (5). This would actually be really valuable, since it could tell us how much we can trust f.

For many practical algorithms, however, it is often observed that the true error (5) is not too far from the empirical error (1). Real algorithms are not as ill behaved as the label memorization algorithm (4), which was an extreme example. The key observation is the dicrepancy between the R_D and R_{emp} is related to the size of $\mathcal{F}$, that is, how flexible our model is. What made the label memorization algorithm so bad was that the class of possible hypotheses was so huge, permitting us to fit anything we wanted. That is an invitation for disastrous overfitting. Learning algorithms used in practice usually have access to a much more limited set of hyoptheses (such as linear discriminators in the case of the perceptron, SVM etc..), so they have less opportunity to overfit.

Minimizing the empirical error might not be such a bad strategy after all *provided* we restrict our hypothesis class. After all, given finite training data, we cannot really do much better. We are still optimizing for the training data, so it is almost inevitable that we are going to do worse on the test data, but we can build in safeguards that will prevent us from doing *much worse*. In fact, it is possible to come up with bounds to quantify how much worse we can expect to be doing.

Uniform Convergence Bounds

So called **uniform convergence bounds** state that for all hypotheses f from a particular hypothesis class $\mathcal{H}$,

$$R[f] \leq R_{\text{emp}}[f] + \epsilon \tag{6}$$

where the **generalization error** $\epsilon_{\mathcal{H}}$ is usually a complicated function of several parameters, including the training set size m. Note that however we choose the hypothesis class and however clever our algorithm is, an unlucky draw of training examples can still lead us astray, so no bound like (6) in every case. The best we can say is that with probability at least $1-\delta$ over the choice of training set, where δ is a (hopefully small) positve real number (6) holds. This δ then becomes one of the parameters that ϵ depends on:

$$\mathbb{P}\left[R[f] - R_{\text{emp}}[f] \leq \epsilon \;\middle|\; \forall f \in \mathcal{F} \right] \geq 1 - \delta. \tag{7}$$

We can make the bound tighter by reducing ϵ but then the probability δ that it will fail goes up, or we can guard ourselves against failiure by reducing δ but then ϵ will be large.

Order of quantifiers

Manufacturing uniform convergence bounds is one of the longest standing and mathematically most challenging enterprises in Machine Learning. Note that since we don't know D, (7) has to hold for all possible distributions D on $\mathcal{X} \times \mathcal{Y}$. Furthermore, with probability $1-\delta$, the inequality must hold for all possible hypotheses simultaneously. In particular, (7) is *not* equivalent to

$$\mathbb{P}\left[R[f] - R_{\text{emp}}[f] \leq \epsilon \right] \geq 1 - \delta \qquad \forall f \in \mathcal{F}. \tag{8}$$

This latter equation roughly says that given any $f \in \mathcal{F}$, except for a fraction at most δ of "unlucky" draws of the training set, (6) will hold. Which training sets are bad and which are good can differ for different f's. In contrast, (7) suggests

that we draw the training set first, decide wheteher it is "lucky" or "unlucky" (which occurs at most δ fraction of the time) and if it is not unlucky, then the inequality will hold for all hypotheses simultaneuously. The set of lucky training sets has to be the same for every hypothesis. Equation (7) is a much stronger statement than (8).

To emphasize this point, it is preferable to write (7) as

$$\mathbb{P}\left[\sup_{f\in\mathcal{F}}\left[R[f]-R_{\text{emp}}[f]\,\right]\leq\epsilon\right]\;\geq\;1-\delta, \tag{9}$$

where sup, pronounced supremum, denotes the lowest upper bound: for our purposes, we can just regard it as the maximum.

The distinction between (9) and (8) is crucial for what we want to use our bounds for. What we are really interested is the error of the hypothesis f^* that minimizes the emprical risk, and the empirical risk, of course, heavily depends on what is in the training. The training set S and the hypothesis f^* are not independent random variables. In fact, given S, by defining f^* to be the minimizer of the emprical risk, we are actively seeking the worst case, in the sense that $R[f^*]-R_{\text{emp}}[f^*]$ will be atypically high. Conversely, given f^*, the distribution of generalization error amongst those training sets that can actually give rise to f^*, is heavily skewed towards large errors.

Bounds of the type (9) are called "uniform" bounds because they have to hold simultaneously for all f. It is also this property that makes deriving them difficult.

Vapnik-Chervonenkis bounds

For classification problems the first breakthrough in this regard was accomplished by Vapnik and Chervonenkis in the 1970's who proved that with probability $1-\delta$ over the choice of training set

$$\sup_{f\in\mathcal{F}}\big[R[f]-R_{\text{emp}}[f]\,\big]\leq\sqrt{\frac{h\left(\log\left(2m/h\right)+1\right)+\log\left(\delta/4\right)}{m}}$$

where h is the VC dimension of $\mathcal{F}$. Recall that h is the maximum number such that there is set $U\subset\mathcal{X}$ of size h with the property that for all subsets $V\subset U$, there is a $f\in\mathcal{F}$ such that $f(x)=1$ for all $x\in V$ and $f(x)=-1$ for all $x\in U\setminus V$.

Structural Risk Minimization

If somebody managed to come up with a good bound of the form (7) that was also suffciently general, the problem of learning would essentially be solved. We could try various hypothesis classes, see how well we can fit the training data with the best hypothesis in each class, and see what ϵ tells us about the likely generalization error. Chances are , for simple hypothesis classes, such as linear discriminants, the training error will not be so good, but ϵ will be relatively small because we are not likely to be overfitting. For complex hypothesis classes, like high order polynomials, we can fit the training data well, but ϵ is going to be huge. All we would have to do is try out a sufficient variety of hypothesis classes and choose the one with the lowest $R_{\text{emp}}[f]+\epsilon$ for the best f in that class. Once we have the class, we just stick with the best f in that class and return that as the final result.

Unfortunately, we are very far from being able to do SRM explicitly, mostly because uniform convergence bounds are so loose. Exactly because they must hold uniformly for all D and all $f \in \mathcal{H}$, the mathematics tends to get very involved, all sorts of loose inequalities have to be invoked along the way and the final result ends up even looser. More often than not, for real world problems uniform convergence bounds return an answer like "using this hypothesis, with .99 probability, the error rate on future examples will be less than $30,000$." In real applications, telling this a doctor who needs to know how much he can trust our algorithm, for example, such a result might not be so useful. For SRM where we are interested in fine variations in generalization capability, current bounds are even more useless. So is the whole business of laboriously proving uniform convergence bounds just a waste of time?

The answer is: not quite. While the numerical bottom line of most bounds known to date is generally very loose, and they might stay that way, too, the *form* of the bounds is often informative. For example, the whole idea of constructing classifiers with large **margins** came out of the studying bounds, when it was discovered that the presence of a large margin can reduce ϵ. More generally, it is important to bear in mind the form of Equation (6). Here we have two competing terms and we are trying to minimize their sum: the error suffered on the training set and a complexity penalty. In the next section we are going to construct algorithms explicitly minimizing an expression of this form.

2 Hilbert Space Methods

2.1 A space of hypotheses

In the previous section we saw that finding the hypothesis f minimizing the empirical error

$$R_{\text{emp}}[f] = \frac{1}{m}\sum_{i=1}^{m} L(f(x_i), y_i) \tag{10}$$

is a natural thing for a learning algorithm to be trying to do. We also saw that *just* minimizing the empirical error is suicidal, because it almost certainly leads to overfitting. Minimizing R_{emp} only makes sense if we simultaneously somehow restrict ourselves to hypotheses that are of just the right level of complexity.

One way to do this is by explicitly restricting the hypothesis space $\mathcal{F}$ to "simple" hypotheses, as in Structural Risk Minimization. Another way, inspired by the form of (6), is to introduce a penalty functional $\Omega[f]$ that somehow measures the complexity of each hypothesis f, and instead of (10) to minimize the sum

$$R_{\text{reg}}[f] = R_{\text{emp}}[f] + \Omega[f]. \tag{11}$$

The penalty $\Omega[f]$ is called the **regularizer** and R_{reg} is called the **regularized risk**.

Another thing we adopt from the previous sections is the function-based approach, concentrating on the abstract optimization problem rather than algorithms and efficiency. For now, we just worry about the loss function L, the regularizer Ω, and the hypothesis class $\mathcal{F}$. The development will be general in the sense of encompassing both classification and regression, but we find that concrete examples are now simplest to construct for regression with the squared error loss (3).

Constructing $\mathcal{F}$

We now set about constructing the hypothesis class $\mathcal{F}$. A natural requirement for $\mathcal{F}$ is that it be a linear function space in the sense that for any $f \in \mathcal{F}$ and any real number λ, λf is in $\mathcal{F}$; and also that for any $f_1, f_2 \in \mathcal{F}$ the sum $f_1 + f_2$ is in $\mathcal{F}$.

The next thing we want from $\mathcal{F}$ is that its structure be somehow related to the regularizer Ω. In particular, we define a **norm** $\| \cdot \|$ on $\mathcal{F}$ and set $\Omega[f] = \| f \|^2$. We have a fair amount of freedom in defining the norm, but we do require that it match the linear structure of $\mathcal{F}$ in the sense that $\| \lambda f \| = \lambda \| f \|$ for any $f \in \mathcal{F}$ and $\lambda \in \mathbb{R}$. In fact, we make the connection a little bit more stringent by requiring that the norm be derived from an **inner product**, $\| f \| = \langle f, f \rangle^{1/2}$. For the definition of a general inner product, see the Appendix.

Hilbert space

Linear inner product spaces (with a few additional mathematical niceitites) are called **Hilbert spaces**, and are very popular in applied mathematics, partly because due to their highly regular strucutre, they are very easy to work in (again, see the Appendix for the exact definition). A Hilbert space might be infinite dimensional, but just like in $\mathbb{R}^n$, we can talk about independent sets, orthogonality and bases. We can also project, decompose, linearly transform and

everything else we love to do in Euclidean space. Most practical computations in Hilbert spaces boil down to ordinary linear algebra.

Evaluation functionals

Another feature of Hilbert spaces we care about is that they are **self-dual**. Recall that the **dual** of a vector space $\mathcal{H}$ is the vector space of linear maps $g : \mathcal{H} \mapsto \mathbb{R}$. When $\mathcal{H}$ is a Hilbert space, any such g has a corresponding $f_g \in \mathcal{H}$ such that $g(f) = \langle f_g, g \rangle$. In particular, when $\mathcal{H}$ is a space of functions over $\mathcal{X}$, the **evaluation functional** $g_x(f) = f(x)$ for any $x \in \mathcal{X}$ is a linear map, hence for any $x \in \mathcal{X}$ we have a special function k_x in $\mathcal{H}$, called the **representer** of x, satisfying

$$f(x) = \langle k_x, f \rangle \quad \forall f \in \mathcal{F}. \tag{12}$$

Right now we don't know what the k_x are, but we do know they exist. The brilliant thing about this construction is that it makes a connection between the abstract structure of $\mathcal{F}$ and what the elements of $\mathcal{F}$ actually are (dicriminant functions or regressors). In particular, we can rewrite the entire regularized risk minimization problem as

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \left[\frac{1}{m} \sum_{i=1}^{m} L(\langle k_{x_i}, f \rangle, y_i) + \langle f, f \rangle \right]. \tag{13}$$

The hypothesis f only features in this equation in the form of inner products with other functions in $\mathcal{F}$. Once we know the form of the inner product and what the k_x are, we can do everything we want with simple mathematics.

Reproducing Kernel Hilbert Space

To this end we apply (12) to $k_{x'}$ for some $x' \in \mathcal{X}$ and find

$$k_{x'}(x) = \langle k_x, k_{x'} \rangle = k_x(x').$$

Specifying the inner product between the various k_x vectors prescribes what those vectors are as functions! Moreover, anything outside the span of the $\{k_x\}_{x \in \mathcal{X}}$ is uninteresting because it does not affect what $f \in \mathcal{F}$ evaluated at any point of the input space is, so we might as well just leave it out of the hyopthesis space altogether. We see that the whole construction, and hence the corresponding algorithm, is driven by the inner products

$$k(x, x') = \langle k_x, k_{x'} \rangle \tag{14}$$

called the **kernel**. The only requirement on the kernel is that it give rise to a valid inner product, that is, it must be a symmetric function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ such that for any finite collection $x_1, x_2, \ldots, x_n \in \mathcal{X}$ and real coefficients $c_1, c_2, \ldots, c_n$,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i \, c_j \, k(x_i, x_j) \geq 0 \tag{15}$$

to guarantee that

$$\langle \textstyle\sum_{i=1}^{n} c_i k_{x_i} \, , \, \sum_{i=1}^{n} c_i k_{x_i} \rangle \geq 0.$$

This latter requirement is expressed by saying that k is a **positive definite** function. Note that it is not equivalent to pointwise positivity, $k(x, x') \geq 0$.

In fact, we can reverse the whole procedure and construct $\mathcal{F}$ starting from the kernel. Given a positive definite function k on the input space $\mathcal{X}$, we define $\mathcal{F}$ to be the minimal complete space of functions that includes all $\{k_x\}_{x\in\mathcal{X}}$ and that has an inner product defined by (14). This defines $\mathcal{F}$ uniquely. Formally, $\mathcal{F}$ is called the **Reproducing Kernel Hilbert Space** (**RKHS)**) associated with k.

We have reduced the learning problem to that of defining the loss function L and the kernel k. In the next section we look at how to choose k. Before that, we note one more imortant feature of the current framework which is already apparent. Just by looking at (13) it is clear that $\hat{f}$ is going to be in the span of the representers of the training data, $k_{x_1}, k_{x_2}, \ldots, k_{x_m}$. We can tell this because the loss term only depends on the inner products of f with $k_{x_1}, k_{x_2}, \ldots, k_{x_m}$, while the regularization term penalizes f "in all directions". If f has any component orthogonal to the subspace spanned by $k_{x_1}, k_{x_2}, \ldots, k_{x_m}$, the loss term is not going to be sensitive to that component, but the regularization term will still penalize it. Hence, the optimal f will be entirely contained in the subspace spanned by the representers. This is called the **representer theorem** and it means that the optimal hypothesis $\hat{f}$ can be expressed as

Representer theorem

$$\hat{f} = b + \sum_{i=1}^{m} \alpha_i \, k_{x_i} \tag{16}$$

for some real coefficients $\alpha_1, \alpha_2, \ldots, \alpha_m$ and bias b. Equivalently,

$$\hat{f}(x) = b + \sum_{i=1}^{m} \alpha_i \, k(x_i, x). \tag{17}$$

Plugging (17) back into (13) reduces our problem to elementary computations: all that our learning algorithm will have to do is figure out the values of the coefficients $\alpha_1, \alpha_2, \ldots, \alpha_i$ and b.

On a more

2.2 The kernel

The way we introduced the kernel as part of the regularization scheme makes it clear that the choice of k really ought to be motivated by the Ω it gives rise to. In particular, Ω should penalize overly complex functions that are likely to have bad generalization performance (i.e. to overfit). Looking at a simple regression problem in, say, $\mathbb{R}^n$, this translates into penalizing functions that are too "wiggly" (not smooth enough). If we ask any physicist or practising engineer how to quantify smoothness, the answer will most likely be: Fourier analysis.

Recall that the **Fourier transform** of a square integrable function $f : \mathbb{R}^n \mapsto \mathbb{C}$ is

Frequency space regularization

$$\tilde{f}(\omega) = \int_{\mathbb{R}^n} f(x) e^{-i\omega \cdot x} dx. \tag{18}$$

The higher the frequency $\| \omega \|$, the more rapidly the corresponding component oscillates. Functions with a great deal of energy ($\| \tilde{f}(\omega) \|^2$) in their high frequency components tend bo be "rough" or have sudden jumps, while smooth functions have most of their energy concentrated in low frequency components. A scheme familiar from many branches of Physics is to regularize with the exponent of the squared norm of the frequency:

$$\Omega[f] = \int \tilde{f}(\omega)\, e^{\sigma^2 \| \omega \|^2/2}\, d\omega \tag{19}$$

for some constant σ. This will heavily penalize "wiggly" functions with large amounts of energy at high frequencies. In Fourier space, this regularization scheme can just be characterized by the simple function

$$\tilde{\Omega}(\omega) = e^{\sigma^2 \| \omega \|^2/2}.$$

From an operational point of view, we are not interested in finding pretty regularizers, though, but in finding a simple the kernel.

The Gaussian kernel

We call a kernel k **stationary** if $k(x, x') = k(x + a, x' + a)$ for all $a \in \mathcal{X}$. Slightly overloading the notation, stationary kernels can be described just by the function $k(x - x') = k(x, x')$. For reasons that we here do not have space to go into, for regularizers expressible in terms of a Fourier transform as a function of $\| \omega \|$ only, the corresponding kernel will be stationary, and $\tilde{\Omega}$ will be related to the Fourier transform of the $k(x)$ by the very simple relation

$$\tilde{k}(\omega) = \frac{1}{\tilde{\Omega}(\omega)}\ .$$

In our case this means that

$$\tilde{k}(\omega) = e^{-\sigma^2 \| \omega \|^2/2}$$

which we recognize as the Fourier transform (up to a constant factor) of the Gaussian

$$k(x) = e^{-x^2/(2\sigma^2)}$$

giving rise to the kernel

$$k(x, x') = e^{-\| x - x' \|^2/(2\sigma^2)}. \tag{20}$$

This is called the **Gaussian Radial Basis Function** (RBF) kernel, or just **Gaussian kernel** for short, and was historically the first kernel to be used in learning algorithms. It is probably the most commonly used kernel to this day.

Using the Gaussian kernel really makes sense because the regularizater it gives rise to heavily penalizes functions that are not smooth. Historically it was not such a sophisticated regularization theory based argument that motivated its introduction, though.

Appendix

Definition 1. *An **inner product** on a (real) vector space $\mathcal{H}$ is a mapping $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \mapsto \mathbb{R}$ satisfying for all vectors $x, y, z \in \mathcal{H}$ and all scalars $\mu, \nu \in \mathbb{R}$*

1. $\langle \mu x + \nu y, z \rangle = \mu \langle x, z \rangle + \nu \langle y, z \rangle$
2. $\langle x, \mu y + \nu z \rangle = \mu \langle x, z \rangle + \nu \langle y, z \rangle$
3. $\langle x, y \rangle = \langle y, x \rangle$
4. $\langle x, x \rangle \geq 0$ *with equality only if* $x = 0$.

Definition 2. *A sequence $x_1, x_2, \ldots$ in a normed space $\mathcal{H}$ is said to be **convergent** if there is a $x \in \mathcal{H}$ with the property that for any $\epsilon > 0$ there is a corresponding k such that $\| x_i - x \| < \epsilon$ for all $i \geq k$. We say that x is the **limit point** of the sequence $x_1, x_2, \ldots$ and notate it $\lim_{i \to \infty} x_i$.*

Definition 3. *A **Cauchy sequence** in a normed space is a sequence $x_1, x_2, \ldots$ with the property for any $\epsilon > 0$ there is a corresponding k such that $\| x_i - x_j \| < \epsilon$ for all $i, j \geq k$.*

Definition 4. *A space $\mathcal{H}$ endowed with a norm is said to be **complete** if every Caucy sequence in $\mathcal{H}$ is convergent in $\mathcal{H}$, i.e. if for every Cauchy sequence $x_1, x_2, \ldots$ in $\mathcal{H}$, the limit point $\lim_{i \to \infty} x_i$ is also in $\mathcal{H}$.*

Definition 5. *A vector space $\mathcal{H}$ endowed with an inner product $\langle \cdot, \cdot \rangle$ and corresponding norm $\| x \| = \langle x, x \rangle^{1/2}$ such that $\mathcal{H}$ is complete with respect to this norm is called a **Hilbert space**.*

Index