MIT-CSAIL-TR-2010-035
CBCL-290

July 30, 2010

Learning and Invariance in a Family of Hierarchical Kernels

Andre Wibisono, Jake Bouvrie, Lorenzo Rosasco, and Tomaso Poggio

Learning and Invariance in a Family of Hierarchical Kernels

Andre Wibisono WIBISONO@MIT.EDU
CBCL, MIT, USA

Jake Bouvrie JVB@MATH.DUKE.EDU
Duke University and CBCL, MIT, USA

Lorenzo Rosasco LROSASCO@MIT.EDU
CBCL, MIT, USA

Tomaso Poggio TP@AI.MIT.EDU
CBCL, MIT, USA

Abstract

Understanding invariance and discrimination properties of hierarchical models is arguably the key to understanding how and why such models, of which the the mammalian visual system is one instance, can lead to good generalization properties and reduce the sample complexity of a given learning task. In this paper we explore invariance to transformation and the role of layerwise embeddings within an abstract framework of hierarchical kernels motivated by the visual cortex. Here a novel form of invariance is induced by propagating the effect of locally defined, invariant kernels throughout a hierarchy. We study this notion of invariance empirically. We then present an extension of the abstract hierarchical modeling framework to incorporate layer-wise embeddings, which we demonstrate can lead to improved generalization and scalable algorithms. Finally we analyze experimentally sample complexity properties as a function of architectural parameters.

1. Introduction

In recent years learning machines have shown to yield good generalization for a variety of tasks. Advances in designing general purpose learning machines are, however, constrained by at least two factors: typically many labeled data are needed to achieve good performance, and often substantial prior knowledge about the problem at hand must be used to obtain state of the art results on complex tasks such as image understanding. In the context of kernel machines, prior knowledge corresponds to specific data representations obtained by choosing kernels, or equivalently feature maps.

It is natural, however, to ask whether there are general principles which might allow one to learn data representations for a wide variety of problems. In this paper we adopt the perspective that such a principle can be a decomposability property which is satisfied in many domains: we will assume that the data is roughly describable by a hierarchy of parts, and work with a class of hierarchical "derived kernels" designed around this assumption introduced by Smale et al. (Smale et al., 2009). The derived kernel formalism generalizes and simplifies important aspects of several recent hierarchical architectures inspired by the visual cortex (Fukushima, 1980; LeCun et al., 1998; Wersing & Korner, 2003; Serre et al., 2007a;b; Jarrett et al., 2009), while preserving many of the key components.

We explore several important aspects of learning with hierarchical models. Our first, and primary, concern is that of invariance to transformation. It has been argued (e.g. (Zoccolan et al., 2007)) that it is the trade-off between invariance and discrimination properties which specifically underpins learning from extraordinarily small samples in the mammalian cortex. We first study a new kind of invariance to transformation induced by propagating local invariance, as given by invariant kernels defined on small image patches, bottom-up, throughout a hierarchy. A set of experiments and a discussion analyzing invariance to rotation of objects in grayscale images is given. We then show empirically that invariance to translation (as "built-in" to the model via spatial pooling) combined with the decomposability assumption inherent in the hierarchy leads to a reduction in sample complexity for a labeled image classification task. How and where attentional effects and context-dependent priors might be included is also discussed.

The second contribution of this paper is to extend the derived kernel framework to incorporate more general feature maps and pooling functions at each layer of the hierarchy. As an example of such a general feature map, we show that layer-wise KPCA embeddings (Schlkopf et al., 1998) can be profitably employed to reduce the computational complexity of the model and scale to large datasets while increasing classification accuracy. As we will describe in the following section, each layer of the derived kernel hierarchy is associated with image patches of a specific size. KPCA or other feature maps applied at each layer leads to a family of algorithms which rely on the geometry of image patches at different scales in an interesting and non-trivial way.

In Section 2 we describe the derived kernel framework, and in Section 3 extensions incorporating general feature maps and pooling functions are proposed. In Section 4 we provide a discussion comparing different notions of invariance found in the literature, and formalize the new notion of invariance. We then present layerwise KPCA embeddings and scalable low-rank approximations in Section 5. Section 6 gives a detailed empirical analysis of invariance discussed in Section 4, and natural low-rank approximations emerging from the discussion in Section 5. Our empirical work concludes with an experimental analysis of sample complexity. We then end with a few remarks in Section 7.

2. Setting & Hierarchical Framework

The use of hierarchical learning architectures to obtain complex data representations has recently received considerable interest in machine learning and computer vision (Wersing & Korner, 2003; Hinton & Salakhutdinov, 2006; Serre et al., 2007b; Jarrett et al., 2009; Lee et al., 2009). Hierarchical parts-based descriptions are ubiquitous among these publications, and date at least back to (Fukushima, 1980). The principle of decomposability of the input data is also at the heart of several Bayesian techniques (Geman & Johnson, 2002; Lee & Mumford, 2003), but has, most importantly, served the human visual system well.

In this paper we adopt the "derived kernel" framework proposed by (Smale et al., 2009), and test empirically several results presented therein. The framework describes, in functional analytical terms, a family of hierarchical kernels and associated feature maps, and seeks to provide an analytically tractable avenue for the analysis of complicated models. The framework also captures essential components of a broad class of architectures, including convolutional neural networks (LeCun et al., 1998) and the model of (Serre et al., 2007b). Here we recall derived kernels and their feature maps in the context of images.

Derived kernels are learned from unlabeled data and can be equivalently thought of as defining a hierarchical feature map which can be used as an unsupervised preprocessing step for a given learning problem. The ingredients needed to define the derived kernel consist of: (1) An architecture defined by a finite number of nested patches (for example subdomains of the square $Sq \subset \mathbb{R}^2$), (2) a set of transformations from a patch to the next larger one, (3) a suitable family of function spaces defined on each patch, and (4) a set of templates which connect the mathematical model to a real world setting.

We will first give the definition of the derived kernel in the case of an architecture composed of three layers of patches u, v and Sq in $\mathbb{R}^2$. The patches are nested, $u \subset v \subset Sq$, and are square, centered, and axis aligned. Assume that we are given a function space on Sq, denoted by $\mathrm{Im}(Sq)$, as well as the function spaces $\mathrm{Im}(u)$, $\mathrm{Im}(v)$ defined on subpatches u, v, respectively. Functions are assumed to take values in $[0, 1]$, and can be interpreted as grey scale images. Next, assume a finite set H_u of *transformations* that are maps from the smallest patch to the next larger patch $h : u \to v$, and similarly H_v with $h : v \to Sq$. Examples of transformations are translations, scalings and rotations, however we will consider only translations here. The transformations are embeddings of u in v and of v in Sq. A translation $h \in H_v$ can be thought of as moving the image over the "receptive field" v. The last and most fundamental ingredient are families of *template sets* $T_u \subset \mathrm{Im}(u)$ and $T_v \subset \mathrm{Im}(v)$, assumed here to be discrete, finite and endowed with the uniform probability measure. The templates are a key semantic component of the model and can be simply thought of as sets of patches (of different size) randomly sampled from images in some database.

In this paper "kernel" refers to reproducing kernels (Aronszajn, 1950), and we will deal primarily with inner product kernels which are known instances of reproducing kernels. We additionally always assume that $K(x, x) \neq 0$ for all $x \in X$ and denote with $\widehat{K}$ kernels normalized according to $\widehat{K}(x, x') = \frac{K(x,x')}{\sqrt{K(x,x)K(x',x')}}$. Clearly in this case $\widehat{K}$ is a reproducing kernel and $\widehat{K}(x,x) = 1$ for all $x \in X$. The kernel normalization provides a more interpretable as well as comparable quantity, as the similarity between two images should not depend on the size of the image.

2.1. The Derived Kernel

Given the above objects, we can describe the construction of the derived kernel in a bottom-up fashion. The process starts with a *normalized* initial reproducing kernel on $\mathrm{Im}(u) \times \mathrm{Im}(u)$ denoted by $\widehat{K}_u(f,g)$ that we assume to be non-negative valued.

Next, an object of central importance, the "neural response" feature map of f at t, is defined as:

$$N_v(f)(t) = \max_{h \in H} \widehat{K}_u(f \circ h, t), \qquad (1)$$

where $f \in \mathrm{Im}(v)$, $t \in T_u$ and $H = H_u$. The neural response of f is a map $N_v(f) : T_u \to [0,1]$. We can interpret the neural response as a vector in $\mathbb{R}^{|T_u|}$ with coordinates $N_v(f)(t)$, for $t \in T_u$. The corresponding inner product on $\mathbb{R}^{|T_u|}$ is defined as $\langle \cdot,\cdot \rangle_{L^2(T_u)}$. The derived kernel on $\mathrm{Im}(v) \times \mathrm{Im}(v)$ is then defined as

$$K_v(f,g) = \langle N_v(f), N_v(g) \rangle_{L^2(T_u)}, \qquad (2)$$

and can be normalized to obtain the kernel $\widehat{K}_v$. The process repeats by defining the second layer neural response as

$$N_{Sq}(f)(t) = \max_{h \in H} \widehat{K}_v(f \circ h, t), \qquad (3)$$

where in this case $f \in \mathrm{Im}(Sq), t \in T_v$ and $H = H_v$. The new derived kernel is now on $\mathrm{Im}(Sq) \times \mathrm{Im}(Sq)$, and is given by

$$K_{Sq}(f,g) = \langle N_{Sq}(f), N_{Sq}(g) \rangle_{L^2(T_v)}, \qquad (4)$$

where $\langle \cdot,\cdot \rangle_{L^2(T_v)}$ is the L^2 inner product with respect to the uniform measure $\frac{1}{|T_v|} \sum_{t \in T_v} \delta_t$. As before, K_{Sq} is normalized to obtain the final derived kernel $\widehat{K}_{Sq}$.

The above construction can be easily generalized to an n layer architecture given by sub-patches $v_1 \subset v_2 \subset \cdots \subset v_n = Sq$. In this case we use the notation $K_n = K_{v_n}$ and similarly $H_n = H_{v_n}$, $T_n = T_{v_n}$, and the definition is given formally using induction:

Definition 2.1. Given a non-negative valued, normalized, initial reproducing kernel $\widehat{K}_1$, the m-layer derived kernel $\widehat{K}_m$, for $m = 2, \ldots, n$, is obtained by normalizing

$$K_m(f,g) = \langle N_m(f), N_m(g) \rangle_{L^2(T_{m-1})}$$

where

$$N_m(f)(t) = \max_{h \in H_{m-1}} \widehat{K}_{m-1}(f \circ h, t), \qquad t \in T_{m-1}.$$

In our discussion, subscripts will be dropped when the statement holds for any layer m within an architecture. Note that the normalized neural response is the feature map associated to the derived kernel, and provides a natural *representation* for any function f.

The derived kernel definitions can also be written compactly as

$$N_{Sq}(f) = \max_{h \in H} \Big\{ \Pi_v \widehat{N}_v(f \circ h) \Big\}, \qquad (5)$$

where the max operation is assumed to apply component-wise, and the operator $\Pi_v : L^2(T_u) \to L^2(T_v)$ is defined as $(\Pi_v)_{t,t'} = \widehat{N}_v(t)(t')$ with $t \in T_v$ and $t' \in T_u$. The operator Π can be seen as a $|T_v| \times |T_u|$ matrix so that each step in the recursion amounts to matrix-vector multiplications ("filtering") followed by max operations ("pooling"). The construction of the operator Π is the unsupervised learning step in the model; a simple way to learn it is by randomly sampling image patches, and this idea hinges on the assumption that the space $\mathrm{Im}(Sq)$ is endowed with a "mother" probability measure ρ.

If the transformation spaces H_i are endowed with probability measures ρ_{H_i}, the measures mark a natural entry point for incorporating notions of attention and context-dependent priors. Such a measure can be used to bias the template sampling mechanism towards exploring particular regions of an image, such as in the case of a rudimentary attention-like mechanism whereby prior information guides the search to interesting parts of an image.

3. General Feature Maps and Pooling Functions

We extend the basic derived kernel defined in the previous section in two ways that will lead to improved generalization and scalability properties: we consider general pooling functions and more sophisticated procedures for defining the feature maps at each layer, while preserving the overall architecture. The generalized framework is perhaps best illustrated by way of the following diagram:

$$\begin{tikzcd}
\hat{K}_1 \arrow[d, "\Phi_1"'] & \hat{K}_2 \arrow[d, "\Phi_2"'] & \cdots & \hat{K}_{n-1} \arrow[d, "\Phi_{n-1}"'] & \hat{K}_n \\
\tilde{K}_1 \arrow[ur, "\text{pool}"] & \tilde{K}_2 \arrow[ur, "\text{pool}"] & \cdots & \tilde{K}_{n-1} \arrow[ur, "\text{pool}"] &
\end{tikzcd}$$

Figure 1. General feature maps and pooling can be defined at each layer.

The pooling operation can be generalized by considering different positive, bounded functionals acting on functions on H. For a fixed function $f \in \mathrm{Im}(v_i)$ and a template $t \in T_{i-1}$ we can consider the positive real valued function on $H = H_{i-1,i}$ defined by $F(h) = F_{f,t}(h) = \widehat{K}_{i-1}(f \circ h, t)$. If $\widehat{K}$ and hence F are sufficiently regular and in particular $F \in L^p(H, \rho_H)$, different pooling operations can be defined by considering L^p norms of F. The original definition of the neural response simply takes the uniform norm in $L^\infty(H)$ $\|F\|_\infty = \sup_{h\in H} F(h)$. Another natural choice is the L^1 norm $\int_H F(h) d\rho_H(h)$, which corresponds to an average. More generally one can consider $\|F\|_p = \left(\int_H F(h)^p d\rho_H(h)\right)^{1/p}$. The neural response for an arbitrary pooling function Ψ is given by

$$N(f)(t) = \Psi(F), \quad \text{with} \quad F(h) = \widehat{K}_{m-1}(f \circ h, t).$$

where $F : H \to \mathbb{R}_+$. As we argue below in Section 4, the pooling function plays an important role in establishing invariance properties so that consideration of different pooling possibilities is an essential part of the modeling process.

For a generalized architecture as shown in Figure 1, compact recursive definitions for the generalized neural response and derived kernel can be given.

Definition 3.1 (Generalized Derived Kernels)**.** Given the feature maps $\Phi_m : L^2(T_{m-1}) \to \mathcal{F}_m$, $2 \le m \le n$, a pooling function Ψ, and a non-negative valued, normalized, initial reproducing kernel $\widetilde{K}_1$, the m-layer derived kernel $\widehat{K}_m$, for $m = 2, \dots, n$, is obtained by normalizing $K_m(f,g) := \langle \widetilde{N}_m(f), \widetilde{N}_m(g) \rangle_{L^2(T_{m-1})}$, and the m-layer generalized derived kernel $\widetilde{K}_m$, for $m = 2, \dots, n$, is given by

$$\widetilde{K}_m(f,g) := \frac{\langle (\Phi_m \circ \widetilde{N}_m)(f), (\Phi_m \circ \widetilde{N}_m)(g) \rangle_{\mathcal{F}_m}}{\|(\Phi_m \circ \widetilde{N}_m)(f)\|_{\mathcal{F}_m} \|(\Phi_m \circ \widetilde{N}_m)(g)\|_{\mathcal{F}_m}}$$

where

$$\widetilde{N}_m(f)(t) = \Psi\Big[\big(\widetilde{K}_{m-1}(f \circ h, t)\big)_{h\in H_{m-1}}\Big] \qquad (6)$$

for $f \in \mathrm{Im}(v_m)$, $t \in T_{m-1}$.

As we will discuss below in Section 5 and demonstrate experimentally in Section 6, this generalized neural response leads to flexible, scalable algorithms.

4. Invariance in Hierarchical Models

A goal of central importance in the study of hierarchical architectures and the visual cortex alike is that of understanding the invariance-selectivity tradeoff, and how invariance and selectivity contribute towards providing an improved representation useful for learning from data. In this section we discuss invariance properties of the derived kernel, and compare the notion of propagated invariance explored in this paper with other forms appearing in the literature. The discussion is complemented by a set of experiments in Section 6, which verify invariance properties and confirm that assumptions built into the current class of hierarchical models apply to and are useful for practical supervised classification tasks.

4.1. Types of Invariance Found in Hierarchical Models

Invariance in hierarchical models can arise in predominantly one of three ways. The simplest case corresponds to pooling over specific transformations (e.g. (Serre et al., 2007a; Lee et al., 2009; Jarrett et al., 2009)). For example, we can look for the best match of a template in an image by comparing translations and rotations of the template to all patches of the image and taking the largest correlation. In this case the transformations are "built-in" to the model, leading to high computational cost. Although it is often the case that the modeler will integrate domain knowledge directly into an architecture by imposing this form of invariance, it is not always obvious what the effect will be in a complex hierarchy involving nested analysis of images and their sub-patches. A preliminary empirical study of this kind of invariance can be found in (Goodfellow et al., 2009). A second form of invariance, prevalent in slow subspace learning (Wiskott & Sejnowski, 2003) and similarity learning (Bar-Hillel et al., 2005), arises from training. In this case the type of invariance is not always known explicitly, but is adapted to the modes of variation in the dataset in accordance with an objective function which may or may not involve data labels. Deep belief networks (Hinton & Salakhutdinov, 2006) and models that effectively compress the data have also been known to achieve small amounts of invariance, but still much less than that of algorithms where a particular invariance is integrated directly.

The first type of invariance is appealing because it allows one to include specific domain knowledge when available, but is brute-force and can be prohibitive computationally. The second form is advantageous in the absence of knowledge about the problem, but requires training and often lacks interpretability. A third means for imposing invariance, which is our focus here, involves enforcing invariance of a similarity metric defined on small patches of images, and allowing local invariance to propagate up the hierarchy. This new method has the benefit that the computational cost does not increase, and the particular local invariance can be chosen to reflect problem-specific knowledge. We discuss this type of invariance in more detail below.

4.2. Invariance Properties of the Neural Response Feature Map

We will consider invariance with respect to some set of transformations $\mathcal{R} = \{r \mid r : v \to v\}$. We say that the neural response or derived kernel is invariant to the transformation $r \in \mathcal{R}$ whenever $\widehat{N}(f) = \widehat{N}(f \circ r)$ (or equivalently $\widehat{K}_n(f \circ r, f) = 1$). The following important assumption relates the transformations $\mathcal{R}$ and the translations H "built-in" to the model:

Assumption 1. *Fix any $r \in \mathcal{R}$. Then for each $h \in H$, there exists a unique $h' \in H$ such that $r \circ h = h' \circ r$, and the map $h \mapsto h'$ is surjective.*

Note that r on the left hand side of the Assumption maps v_{m+1} to itself, while on the right hand side r maps v_m to itself. As an example, let $\mathcal{R}$ be rotations about the origin and let H be translations so that $f \circ h$ is an image patch obtained by restricting an image f to a sub-patch. The assumption says that rotating an image and then taking a restriction (patch) is equivalent to first taking a (different) restriction and then rotating the resulting image patch.

Given this assumption, we recall the following invariance result:

Proposition 4.1 ((Smale et al., 2009)). *If the initial kernel satisfies $\widehat{K}_1(f, f \circ r) = 1$ for all $r \in \mathcal{R}$, $f \in \mathrm{Im}(v_1)$, then*

$$\widehat{N}_m(f) = \widehat{N}_m(f \circ r),$$

for all $r \in \mathcal{R}$, $f \in \mathrm{Im}(v_m)$ and $m \leq n$.

This result says that if Assumption 1 holds (at all layers), and an appropriate *local invariance* holds via $\widehat{K}_1$ – the kernel defined on the smallest patches – then the neural responses at all layers are invariant to the transformations in $\mathcal{R}$. In particular, the final representation will be invariant to *global* transformations of the input. Thus if invariance of the initial kernel is enforced, then this invariance is seen to propagate to higher layers and impose additional invariance properties. These invariances collectively lead to rich equivalence classes of inputs, the members of which get mapped to the same representation by the hierarchy.

5. Layer-wise KPCA Embeddings and Scalability

We provide an example illustrating the feature map extension proposed in Section 3 and consider the case of a canonical feature map associated to the Mercer expansion of the derived kernel at a given layer. This example corresponds to applying KPCA (but without centering in this case) to the neural responses of the templates, and projecting the responses of templates and input images presented to the hierarchy onto a principal subspace. By projecting the neural response vectors onto only the first $k < |T|$ principal components, there is hope that for some small k one can achieve a computational speedup without sacrificing performance on the task. Section 6 suggests that indeed only a small number of components k are needed in practice.

Note that each space $\mathrm{Im}(v)$ can be endowed with a probability measure ρ_v. Starting from the mother measure on $\mathrm{Im}(Sq)$ and a given measure ρ_H on $H = H_{v_{n-1},Sq}$, the product space $\mathrm{Im}(Sq) \times H$ can be endowed with the corresponding product measure P. The measure $\rho_{v_{n-1}}$ on $\mathrm{Im}(v_{n-1})$ is then defined as the pushforward measure $\rho_v = P \circ \pi^{-1}$ defined by the map $\pi = \pi_v : \mathrm{Im}(Sq) \times H \to \mathrm{Im}(v)$ mapping (f, h) to $f \circ h$. The construction then repeats in a similar fashion for the previous layer, proceeding from the top downwards. At any layer we can then consider the integral operator

$$L_{\widehat{K}} F(f) = \int_{\mathrm{Im}(v)} \widehat{K}(f, g) F(g) d\rho_v(g)$$

whose spectrum we denote by $(\sigma_j, \phi_j)_{j=1}^{p}$, with $p \leq \infty$. In practice the measure ρ_v can be replaced by the empirical measure underlying the corresponding template set T. Now consider the map $\Phi : \mathrm{Im}(v) \to \ell^2$ such that $f \mapsto \Phi(f) = \big(\sqrt{\sigma_1}\phi_1(f), \sqrt{\sigma_1}\phi_2(f), \ldots, \sqrt{\sigma_N}\phi_N(f)\big)$, with $N < p$. Recalling the Mercer expansion of a kernel, one can see that the above feature map corresponds to replacing the derived kernel with a truncated derived kernel of the form

$$\widetilde{K}(f, g) = \sum_{j=1}^{N} \sigma_j \phi_j(f) \phi_j(g).$$

Alternatively, using the above truncated kernel can be seen as equivalent to working with low-rank approximations of the Π matrices appearing at each layer, as defined in Equation (5). In this case the approximation scheme is exactly the truncated SVD.

The above reasoning is not specific to KPCA, however. Other feature maps, for example incorporating further geometric information or sparsity constraints may also be used. In the context of deep belief networks, (Weston et al., 2008) has previously explored using Laplacian embeddings at each layer of a DBN.

6. Empirical Analysis

In this section we present experimental results verifying the properties and performance of the derived kernel architecture, in particular in the setting of the generalized derived kernel that was developed in Sections 3 and 5. The purpose of this empirical evaluation is twofold. First, we verify the extent to which the theoretical invariance properties of the derived kernel hold in practice, given that we must necessarily work in a finite, discrete setting where the theoretical results might not apply completely. It is therefore interesting to quantify the extent to which the theoretical and empirical results agree. Second, we compare the classification performance of the generalized derived kernel with embeddings given by KPCA to the performance of the original derived kernel. Here the goal is not to achieve the best possible classification accuracies, but rather to check the goodness of the data representation that the neural response encoding provides. In particular we will be working in an impoverished regime, in which we only use a small number of training images. Moreover, the classifications are performed using a 1-nearest neighbor rule defined by the distance induced by the derived kernel. This simple choice of classifier highlights the role of neural response as a data representation.

6.1. Experimental Setup

The experiments in this section use two databases of images: the MNIST dataset of handwritten digits (LeCun et al., 1998) and the Caltech101 dataset (Fei-Fei et al., 2006).

MNIST. Each image in the MNIST dataset is 28×28 pixels in grayscale. Experiments using the MNIST dataset are performed using a three-layer architecture with patch sizes $u = 12\times12$, $v = 20\times20$, and $Sq = 28\times 28$, except for the low-rank approximation experiment in Section 6.4 in which a two-layer architecture is also used with patch sizes $u = 12 \times 12$ and $Sq = 28 \times 28$. The template sets are constructed by randomly extracting 500 image patches (of size u and v) from images which are not used in the train or test sets. The transformations at each stage are taken to be all possible translations.

Caltech101. We use a subset of 8 categories from the Caltech101 dataset: `binocular`, `cup`, `gramophone`, `grand_piano`, `headphone`, `inline_skate`, `laptop`, and `umbrella`. The images are resized to be 100×100 pixels. Experiments using the Caltech101 dataset are performed using a three-layer architecture with patch sizes $u = 24 \times 24$, $v = 60 \times 60$, and $Sq = 100 \times 100$, with 500 templates per layer. The transformations at each layer are taken to be translations with a 3 pixel hop size.

6.2. Transformation Invariance

In this section we investigate the experimental invariance of the derived kernel when the transformation is a rotation. For the initial kernel at the first layer, we use the **histogram kernel**:

$$\widehat{K}^{\text{hist}}(f, g) = \frac{\langle \text{hist}(f), \text{hist}(g)\rangle}{\|\text{hist}(f)\|\|\text{hist}(g)\|},$$

where $\text{hist}(z)$ denotes the histogram representation of the image z with 101 bins centered at $0, 0.01, \ldots, 0.99, 1$. In the ideal setting, this initial kernel is invariant to rotation. However, Proposition 4.1 might not apply in practice because we are working in a finite and discrete setting, and Assumption 1 might not hold since the set of translations H is not exhaustive. Furthermore, a rotated image might become distorted or have different pixel intensity distributions because of cropping and zero-padding. The rotation invariant architecture is then compared to the usual architecture built from normalized inner product kernels, which should not be rotation invariant. For Proposition 4.1 to be meaningful, the architecture using histogram kernel should exhibit some degree of rotation invariance, while the architecture using inner product kernel should be sensitive to rotation. We present the results of two experiments that confirm this hypothesis.

6.2.1. Distribution Comparison

In this experiment we compare the distribution of the derived kernel values between rotated images $K(f, f \circ r)$ with the derived kernel values between distinct images $K(f, g)$. We sample one image per class and rotate each image 12 times, with rotation angles multiples of 30°, and compute the pairwise derived kernel values between the resulting images. We then compare the distribution of the "within-class" values $K(f, f \circ r)$ with the "between-class" values $K(f, g)$. If the architecture is rotation invariant, then we expect to see that the within-class distribution is strongly peaked at 1. In practice, we observe that the kernel values are close to 1, so in this experiment we compare the distributions of $1 - K(\cdot, \cdot)$ instead.

Figure 2 shows the results of this experiment on the MNIST and Caltech101 (subset) datasets. The left panel of Figure 2(a) shows the comparison of the Q-Q (quantile-quantile) plots of the within- and between-class values across the histogram and inner product kernel for the MNIST dataset. The Q-Q plot measures the degree of similarity between two distributions. The diagonal line in each plot is the line $y = x$, and the shape of the plot with respect to this line indicates the relative dispersion of the two probabilities. In our case, the Q-Q plot of the histogram kernel (left) is vertical in the beginning, which indicates strong concentration of the within-class values at 0. In contrast, the plot for the inner product kernel (right) lies below the line $y = x$, which indicates that the within-class values is more dispersed than the between-class values.

The right panel of Figure 2(a) provides another view of this result. The diagram in the figure shows the mean and standard deviation of each empirical distribution. The ranges of the kernel values of the histogram and inner product kernel are rescaled to be the same to allow us to present a meaningful visual comparison in the diagram. From the figure it is clear that in the histogram architecture the within-class values are very concentrated compared to the between-class values, while in the inner product architecture, the within- and between-class values are almost the same.

Figure 2(b) shows the results of the same experiment on the Caltech101 (subset) dataset. In the left panel, although the two Q-Q plots lie above the $y = x$ line, the plot for the histogram kernel is much steeper, and even vertical in the beginning, which indicates strong concentration at 0. Moreover, this is also confirmed by the diagram in the right panel. On the other hand, while the within-class values in the inner product kernel is also more compressed compared to the between-class values, it is not concentrated at 0, and the compression is not too severe.

Together the results presented in this section confirm our hypothesis that the architecture with histogram kernel exhibits invariance to rotation, while the architecture with inner product kernel lacks it.

6.2.2. Image Identification and Classification

The second experiment is to identify or classify a randomly rotated image using the derived kernel. We sample 30 images per class from a given dataset, and for each image we encode both the original image and a randomly rotated version of it. Then for every rotated image $f \circ r$, we use the derived kernel K to find an original image g among those sampled that maximizes $K(f \circ r, g)$, and count how many times we find f and g to be equal ("identification" task) or to be in the same class ("classification" task). Table 1 shows the accuracy results of the experiments, averaged over 50 independent trials, on the MNIST (9 digit classes, 1s through 9s) and Caltech101 (subset) datasets. The accuracy of the experiment using the L_2 distance is also presented for comparison. From the table, we see that the histogram kernel yields significantly better performances compared to the inner product kernel and the L_2 distance, in both datasets and both tasks. This result confirms that the histogram kernel architecture exhibits a degree of rotation invariance, while the inner product architecture is sensitive to it.

Table 1. Accuracy of the identification and classification tasks via randomly rotated images.

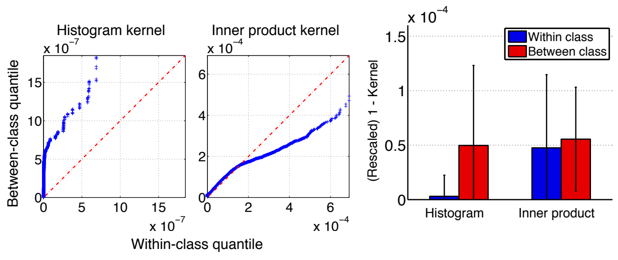
Dataset: MNIST			
Task	Histogram	Inner product	L_2
identify	37.39%	8.84%	8.05%
classify	47.40%	27.13%	27.64%

Dataset: Caltech101			
Task	Histogram	Inner product	L_2
identify	88.62%	21.98%	9.94%
classify	91.35%	34.06%	24.64%

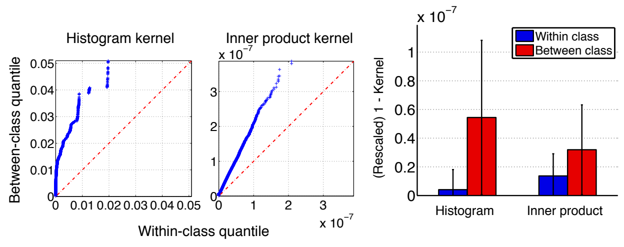
6.3. Sample Complexity

In this experiment we confirm empirically that, when used as an unsupervised data preprocessing step, the neural response provides a representation which leads to lower sample complexity of a supervised task. We apply two- and three-layer derived kernels to an 8-class MNIST classification task (digits 2s through 9s), with 3-pixels of random, artificial translation applied to the images. We use varying numbers of images per class for training and 30 images per class for testing. Figure 3 shows the classification accuracies as functions of the number of training images per class, averaged over 50 random test sets while holding the training and template sets fixed. An L_2 baseline is also given for comparison.

Figure 3 shows that three-layer architecture achieves higher accuracy with fewer training examples compared to the other two models. We find, for example, that in order to obtain 65% accuracy the 2-layer derived kernel based classifier needs about 11 examples per class, while the 3-layer derived kernel based classifier requires only 5 examples. The overall behavior confirms that: (1) the hierarchical assumption holds for this task, and (2) in terms of sample complexity, two layers is better than none, and three is better than two.



(a) Results for MNIST.



(b) Results for Caltech101.

Figure 2. Results for rotation invariance tests. In each subfigure the left panel shows the Q-Q plot and the right panel shows the comparison of the mean value and standard deviation.

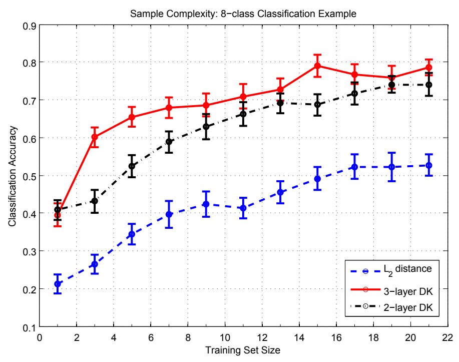


Figure 3. Empirical sample complexities of the digit classification task using two and three layer hierarchical models.

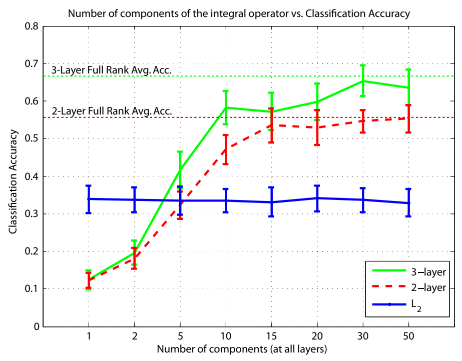


Figure 4. Classification accuracy with layer-wise low rank approximations to the integral operator.

6.4. Scalability and Layer-wise Low-rank Approximations

In this section we investigate the classification performance of the derived kernel architecture using layer-wise low-rank approximations to the integral operator associated with the derived kernel, as developed in Section 5. Figure 4 shows the accuracies for the 8-class MNIST classification task with 5 training images and 30 test images per class, when taking different numbers of components in the approximation at all layers. Although there are originally 500 templates at each layer, the experiments show that only 20-25 components gives similar accuracy as the full rank case. The computational advantage of working with such an approximation is substantial.

Our last experiment compares the performance of PCA and KPCA in the 8-class MNIST classification task with three-layer architectures and 30 training and testing images per class. For KPCA we use the Gaussian kernel $K_G(x, y) = e^{-\gamma(x-y)^2}$, where the γ parameter at each layer is estimated using the average nearest neighbors distances from the encoded templates. Figure 5 shows the classification accuracies for the PCA and Gaussian PCA architectures as the number of components varies. The accuracies are averaged over 50 trials. The accuracy for the original three-layer architecture with no extension method is also given for reference. We make the following observations from Figure 5:

1. Classification accuracies for both PCA and Gaussian PCA architectures tend to increase as the number of components increases.
2. PCA reaches its stable accuracy at 30 components, but this accuracy is slightly lower than the accuracy of the original architecture.
3. Gaussian PCA reaches its stable accuracy at 150 components, and this accuracy is the same as that of the original architecture.
4. For small number (< 50) of components, PCA outperforms Gaussian PCA. Therefore, PCA provides a substantial computational speed-up at the cost of slightly lower accuracy. On the other hand, If our goal is to reach the best accuracy, then Gaussian PCA with 150 components also provides computational benefits.

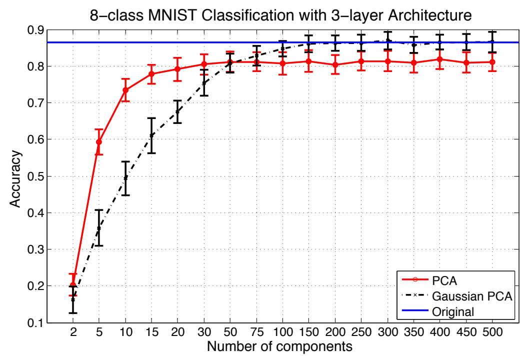


Figure 5. Accuracy of PCA and Gaussian PCA on 8-class MNIST classification task.

7. Conclusions

We have explored a new form of invariance and the impact of layer-wise feature maps in a family of biologically-inspired hierarchical "derived" kernels. We extended the derived kernel framework of (Smale et al., 2009) to include more general layer-wise embeddings and pooling functions, and verified that theoretical notions of bottom-up, "propagated" invariance hold empirically when applied to real-world data. We suggested that the application of layerwise feature maps, in particular KPCA, can lead to scalable algorithms, and a class of methods more generally which might merit further study. The assumption that images can be decomposed into parts was also shown to hold experimentally, and furthermore led to representations which reduced the sample complexity of a corresponding classification task.

References

Aronszajn, N. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68:337–404, 1950.

Bar-Hillel, A., Hertz, T., Shental, N., and Weinshall, D. Learning a Mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6:937–965, 2005.

Fei-Fei, L., Fergus, R., and Perona, P. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):594, 2006.

Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cyb.*, 36:193–202, 1980.

Geman, S. and Johnson, M. Probabilistic grammars and their applications. *International Encyclopedia of the Social & Behavioral Sciences*, pp. 12075–12082, 2002.

Goodfellow, I., Le, Q., Saxe, A., Lee, H., and Ng, A. Measuring invariances in deep networks. In *Advances in Neural Information Processing Systems (NIPS) 22*, 2009.

Hinton, G.E. and Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science*, 313 (5786):504–507, 2006.

Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. What is the best multi-stage architecture for object recognition? In *Proc. International Conference on Computer Vision (ICCV'09)*. IEEE, 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, November 1998.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the Twenty-Sixth International Conference on Machine Learning*, 2009.

Lee, T.S. and Mumford, D. Hierarchical bayesian inference in the visual cortex. *J Opt Soc Am A Opt Image Sci Vis*, 20(7):1434–1448, July 2003.

Schlkopf, B., Smola, A., and Mller, K. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

Serre, T., Oliva, A., and Poggio, T. A feedforward architecture accounts for rapid categorization. *Proceedings of the National Academy of Science*, 104:6424–6429, 2007a.

Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., and Poggio, T. Robust object recognition with cortex-like mechanisms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29:411–426, 2007b.

Smale, S., Rosasco, L., Bouvrie, J., Caponnetto, A., and Poggio, T. Mathematics of the neural response. *Foundations of Computational Mathematics*, June 2009. DOI:10.1007/s10208-009-9049-1.

Wersing, H. and Korner, E. Learning optimized features for hierarchical models of invariant object recognition. *Neural Comput.*, 7(15):1559–1588, July 2003.

Weston, J., Ratle, F., and Collobert, R. Deep learning via semi-supervised embedding. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pp. 1168–1175, New York, NY, USA, 2008. ACM.

Wiskott, T. and Sejnowski, T. Slow feature analysis: unsupervised learning of invariances. *Neural Computation*, 14:715–770, 2003.

Zoccolan, D., Kouh, M., Poggio, T., and DiCarlo, J. J. Trade-off between object selectivity and tolerance in monkey inferotemporal cortex. *J. Neurosci.*, 27(45): 12292–12307, 2007.