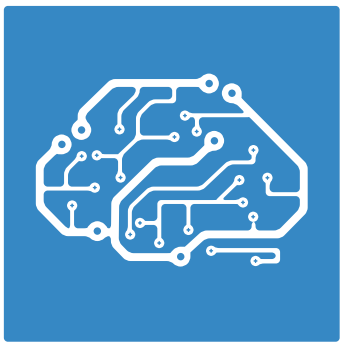

Center for Brains, Minds & Machines

CBMM Memo No. 35 **August 5, 2015**

Deep Convolutional Networks are Hierarchical Kernel Machines

by

Fabio Anselmi[1,2], **Lorenzo Rosasco**[1,2,3],**Cheston Tan**[4] **Tomaso Poggio**[1,2]

1 **Center for Brains, Minds and Machines, McGovern Institute, Massachusetts Institute of Technology, Cambridge, MA, USA.**

2 **Istituto Italiano di Tecnologia, Genova, Italy.**

3 **DIBRIS, Universita' degli studi di Genova, Italy.**

4 **Institute for Infocomm Research, Singapore, 138632.**

Abstract: In i-theory a typical layer of a hierarchical architecture consists of HW modules pooling the dot products of the inputs to the layer with the transformations of a few templates under a group. Such layers include as special cases the convolutional layers of Deep Convolutional Networks (DCNs) as well as the non-convolutional layers (when the group contains only the identity). Rectifying nonlinearities – which are used by present-day DCNs – are one of the several nonlinearities admitted by i-theory for the HW module. We discuss here the equivalence between group averages of linear combinations of rectifying nonlinearities and an associated kernel. This property implies that present-day DCNs can be exactly equivalent to a hierarchy of kernel machines with pooling and non-pooling layers. Finally, we describe a conjecture for theoretically understanding hierarchies of such modules. A main consequence of the conjecture is that hierarchies of trained HW modules minimize memory requirements while computing a selective and invariant representation.

This work was supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216.

Deep Convolutional Networks are Hierarchical Kernel Machines

Fabio Anselmi[1,2], Lorenzo Rosasco[1,2,3], Cheston Tan[4], and Tomaso Poggio[1,2,4]

[1]Center for Brains Minds and Machines, Massachusetts Institute of Technology, Cambridge, MA 02139.
[2]Laboratory for Computational Learning, Istituto Italiano di Tecnologia and Massachusetts Institute of Technology.
[3]DIBRIS, Universitá degli studi di Genova, Italy, 16146.
[4]Institute for Infocomm Research, Singapore, 138632.

August 5, 2015

Abstract

In i-theory a typical layer of a hierarchical architecture consists of HW modules pooling the dot products of the inputs to the layer with the transformations of a few templates under a group. Such layers include as special cases the convolutional layers of Deep Convolutional Networks (DCNs) as well as the non-convolutional layers (when the group contains only the identity). Rectifying nonlinearities – which are used by present-day DCNs – are one of the several nonlinearities admitted by i-theory for the HW module. We discuss here the equivalence between group averages of linear combinations of rectifying nonlinearities and an associated kernel. This property implies that present-day DCNs can be exactly equivalent to a hierarchy of kernel machines with pooling and non-pooling layers. Finally, we describe a conjecture for theoretically understanding hierarchies of such modules. A main consequence of the conjecture is that hierarchies of trained HW modules minimize memory requirements while computing a selective and invariant representation.

1 Introduction

The architectures now called Deep Learning Convolutional networks appeared with the name of convolutional neural networks in the 1990s – though the supervised optimization techniques used for training them have changed somewhat in the meantime. Such architectures have a history that goes back to the original Hubel and Wiesel proposal of a hierarchical architecture for the visual ventral cortex iterating in different layers the motif of simple and complex cells in V1. This idea led to a series of quantitative, convolutional cortical models from Fukushima ([1]) to HMAX (Riesenhuber and Poggio, [2]). In later versions (Serre et al., [3]) such models of primate visual cortex have achieved object recognition performance at the level of rapid human categorization. More recently, deep learning convolutional networks trained with very

Email addresses: anselmi@mit.edu; lrosasco@mit.edu; cheston-tan@i2r.a-star.edu.sg; corresponding author: tp@ai.mit.edu. The main part of this work was done at the Institute for Infocomm Research with funding from REVIVE

large labeled datasets (Russakovsky et al. [4], Google [5], Zeiler and Fergus [6]) have achieved impressive performance in vision and speech classification tasks. The performance of these systems is ironically matched by our present ignorance of why they work as well as they do. Models are not enough. A theory is required for a satisfactory explanation and for showing the way towards further progress. This brief note outlines a framework towards the goal of a full theory.

Its organization is as follows. We first discuss how i-theory applies to existing DLCNs. We then show that linear combinations of rectification stages can be equivalent to kernels. Deep Learning Convolutional Networks can be similar to hierarchies of HBFs ([7]).

2 DCNs are hierarchies of kernel machines

In this section, we review the basic computational units composing deep learning architectures of the convolution type. Then we establish some of their mathematical properties by using i-theory as described in [8, 9, 10].

2.1 DCNs and i-theory

The class of learning algorithms called deep learning, and in particular convolutional networks, are based on a basic operation in multiple layers. We describe it using the notation of i-theory.

The operation is the inner product of an input with another point called a template (or a filter, or a kernel), followed by a non linearity, followed by a group average. The output of the first two steps can be seen to roughly correspond to the neural response of a so called simple cell [11]. The collection of inner products of a given input with a template and its *transformations* in i-theory corresponds to a so called convolutional layer in DCNs. More precisely, given a template t and its transformations gt, here $g \in \mathcal{G}$ is a finite set of transformations (in DCNs the only transformations presently used are translations), we have that each input x is mapped to $\langle x, gt \rangle, \quad g \in \mathcal{G}$. The values are hence processed via a non linear *activation* function, e.g. a sigmoid $(1 + e^{-s})^{-1}$, or a rectifier $|s + b|_+ = \max\{-b, s\}$ for $s, b \in \mathbb{R}$ (the rectifier nonlinearity was called ramp by Breiman[12]). In summary, the first operation unit can be described, for example by

$$x \mapsto |\langle x, gt \rangle + b|_+.$$

The last step, often called *pooling*, aggregates in a single output the values of the different inner products previously computed that correspond to transformations of the same template, for example via a sum

$$\sum_g |\langle x, gt \rangle + b|_+, \quad t \in \mathcal{T}, b \in \mathbb{R} \tag{1}$$

or a max operation

$$\max_g |\langle x, gt \rangle + b|_+, \quad t \in \mathcal{T}, b \in \mathbb{R} \tag{2}$$

This corresponds to the neural response of a so called complex cell in [11].

2.2 A HW module is a kernel machine

It is trivial that almost any (positive or negative defined) nonlinearity after the dot product in a network yields a kernel. Consider a 3-layers network with the first layer being the input layer

x. Unit i in the second layer (comprising N units) computes $|\langle t_i, x\rangle + b_i|_+ = \phi_i(x), i = 1, \cdots, N$. Thus each unit in the third layer performing a dot product of the vector of activities from the second layer with weights $\phi_i(y)$ (from layer two to three) computes $K(x,y) = \sum_j \phi_j(x)\phi_j(y)$ which is guaranteed to be well defined because the sum is finite. Key steps of the formal proof, which holds also in the general case, can be found in Rosasco and Poggio, [10] and references there; see also Appendix 5.1 and Appendix 5.2 for an example.

A different argument shows that it is "easy" to obtain kernels in layered networks. Assume that inputs $x \in \mathbb{R}^n$ as well as the weights t are normalized, that is $x, t \in S^n$ where S^n is the unit sphere.
In this case dot products are radial functions (since $\langle x, t\rangle = \frac{1}{2}(2-(|x-t|^2))$) of r^2. The kernel r^2 can be shaped by linear combinations (with bias) of rectifier units (linear combinations of ramps can generate sigmoidal units, linear combinations of which are equivalent to quasi-Gaussian, "triangular" functions)

Thus for normalized inputs dot products with nonlinearities can easily be equivalent to radial kernels. Pooling before a similarity operation (thus pooling at layer n in a DCN before dot products in layer $n+1$) maintains the kernel structure (since $\bar{K}(x,x') = \int dg \int dg' K(gx, g'x')$ is a kernel if K is a kernel). Thus *DCNs with normalized inputs are hierarchies of radial kernel machines, also called Radial Basis Functions (RBFs).*

Kernels induced by linear combinations of features such as

$$\sum_g |\langle x, gt\rangle + b|_+, \quad t \in \mathcal{T}, b \in \mathbb{R} \tag{3}$$

are selective and invariant (if G is compact, see Appendix 5.1). The max operation $\max_g |\langle x, gt\rangle + b|_+$ has a different form and does not satisfy the condition of the theorems in the Appendix. On the other hand the pooling defined in terms of the following soft-max operation (which approximates the max for "large" n)

$$\sum_g \frac{(\langle x, gt\rangle)^n}{\sum_{g'}(1 + \langle x, g't\rangle)^{n-1}}, \quad t \in \mathcal{T} \tag{4}$$

induces a kernel that satisfies the conditions of the theorems summarized in Appendix 5.1. It is well known that such an operation can be implemented by simple circuits of the lateral inhibition type (see [14]). On the other hand the pooling in Appendix 5.3 (see [15]) does not correspond in general to a kernel, does not satisfy the conditions of Appendix 5.1 and is not guaranteed to be selective.

Since weights and "centers" of the RBFs are learned in a supervised way, the kernel machines should be more properly called HyperBF, see Appendix 5.4.

2.3 Summary: every layer of a DCN is a kernel machine

Layers of a Deep Convolutional Network using linear rectifiers (ramps) can be described as

$$\sum_g |\langle x, gt\rangle + b|_+, \quad t \in \mathcal{T}, b \in \mathbb{R} \tag{5}$$

Notice that in one dimension the kernel $|x-y|$ can be written in terms of ramp functions as $|x-y| = |x-y|_+ + |-(x-y)|_+$. See Figure 1 and [13].

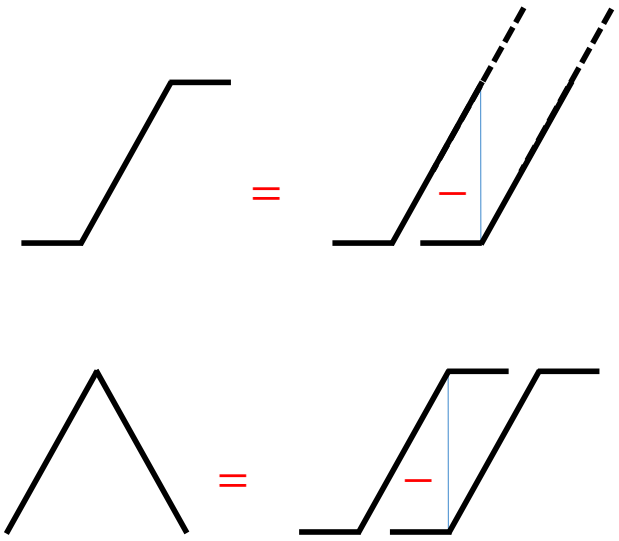

Figure 1: *A "Gaussian" of one variable can be written as the linear combination of ramps (e.g. rectifiers): a) a sigmoid-like function can be written as linear combinations of ramps b) linear combinations of sigmoids give gaussian-like triangular functions.*

where the range of pooling ($\sum_g$) may be degenerate (no pooling) in which case the layer is not a convolutional layer.

Such a layer corresponds to the kernel

$$\tilde{K}(x,x') = \int dg\, dg'\, K_0(x,g,g'). \tag{6}$$

with

$$K_0(x,g,g') = \int dt\, db\, |\langle gt,x\rangle + b|_+ |\langle g't,x'\rangle + b|_+. \tag{7}$$

An alternative pooling to 5 is provided by softmax pooling

$$\sum_g \frac{(\langle x,gt\rangle)^n}{\sum_{g'}(1+\langle x,g't\rangle)^{n-1}}, \quad t \in \mathcal{T} \tag{8}$$

Present-day DCNs seem to use equation 8 for the pooling layers and equation 5 for the non-convolutional layers. The latter is the degenerate case of equation 8 (when G contains only the identity element).

3 A conjecture on what a hierarchy does

The previous sections describe an extension of i-theory that can be applied exactly to any layer of a DLCN and any of the nonlinearities that have been used: pooling, linear rectifier, sigmoidal units. In this section we suggest a framework for understanding hierarchies of such modules, which we hope may lead to new formal or empirical results.

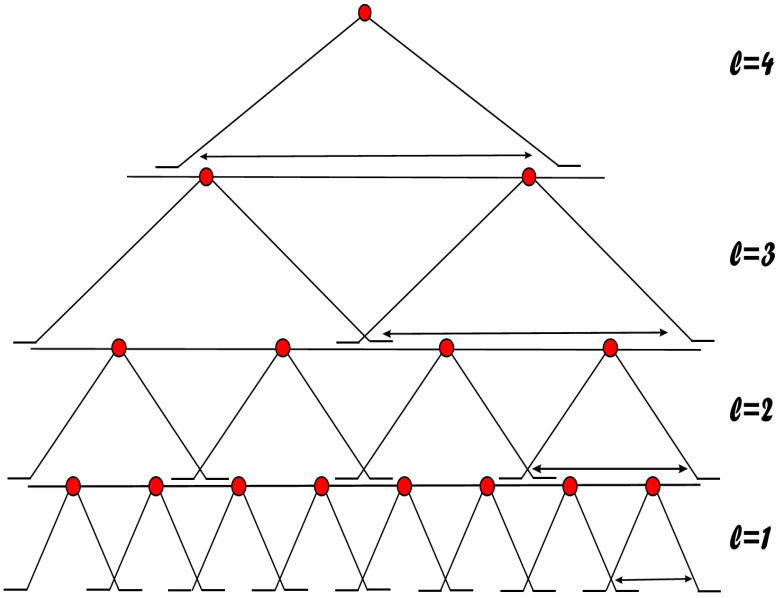

Figure 2: *A hierarchical, supervised architecture built from eHW-modules. Each red circle represents the signature vector computed by the associated module (the outputs of complex cells) and double arrows represent its receptive fields – the part of the (neural) image visible to the module (for translations this is also the pooling range). The "image" is at level* 0*, at the bottom.*

3.1 DLCNs and HVQ

We consider here an HW module with pooling (no pooling corresponds to 1 pixel stride convolutional layers in DLN). Under the assumption of normalized inputs, this is equivalent to a HBF module with Gaussian-like radial kernel and "movable" centers (we assume standard Euclidean distance, see Poggio, Girosi,[16]; Jones, Girosi and Poggio,[13]).

Notice that one-hidden-layer HBF can be much more efficient in storage (e.g. bits used for all the centers) than classical RBF because of the smaller number of centers (HBFs are similar to a multidimensional free-knots spline whereas RBFs correspond to classical spline).

The next step in the argument is the observation that a network of radial Gaussian-like units become in the limit of $\sigma \to 0$ a look-up table with entries corresponding to the centers. The network can be described in terms of *soft Vector Quantization* (VQ) (see section 6.3 in Poggio and Girosi, [7]). Notice that hierarchical VQ (dubbed HVQ) can be even more efficient than VQ in terms of storage requirements (see e.g. [17]). This suggests that a hierarchy of HBF layers may be similar (depending on which weights are determined by learning) to HVQ. Note that *compression is achieved when parts can be reused in higher level layers as in convolutional networks*. Notice that the center of one unit at level n of the "convolutional" hierarchy of Figure 2 is a combinations of parts provided by each of the lower units feeding in it. This may even happen without convolution and pooling as shown in the following extreme example.

Example Consider the case of kernels that are in the limit delta-like functions (such as Gaussian with very small variance). Suppose as in Figure 3 that there are four possible quantizations of the input x: x_1, x_2, x_3, x_4. One hidden layer would consist of four units $\delta(x - x_i), i = 1, \cdots, 4$. But suppose that the vectors x_1, x_2, x_3, x_4 can be decomposed in terms of two smaller parts or

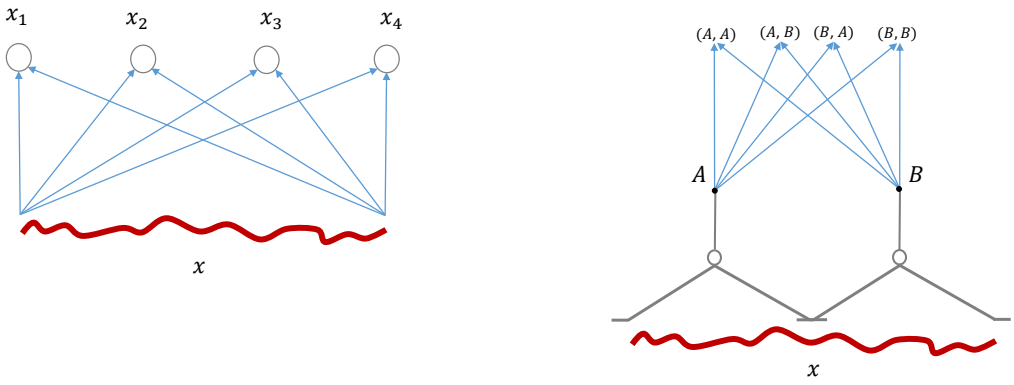

Figure 3: *See text, Example in section 3.1.*

features x' and x'', e.g. $x_1 = x' \oplus x''$, $x_2 = x' \oplus x'$, $x_3 = x'' \oplus x''$ and $x_4 = x'' \oplus x'$. Then a two layer network could have two types of units in the first layer $\delta(x - x')$ and $\delta(x - x'')$; in the second layer four units will detect the conjunctions of x' and x'' corresponding to x_1, x_2, x_3, x_4. The memory requirements will go from $4N$ to $2N/2 + 8$ where N is the length of the quantized vectors; the latter is much smaller for large N. Memory compression for HVQ vs VQ – that is for multilayer networks vs one-layer networks – increases with the number of (reusable) parts. Thus for problems that are *compositional*, such as text and images, hierarchical architectures of HBF modules minimize memory requirements.

Classical theorems (see refrences in [18, 19] show that one hidden layer networks can approximate arbitrarily well rather general classes of functions. A possible advantage of multilayer vs one-layer networks that emerges from the analysis of this paper is memory efficiency which can be critical for large data sets and is related to generalization rates.

4 Remarks

- Throughout this note, we discuss the potential properties of multilayer networks, that is the properties they have with the "appropriate" sets of weights when supervised training is involved. The assumption is therefore that greedy SGD using very large sets of labeled data, can find the "appropriate sets of sets of weights".

- Recently, several authors have expressed surprise when observing that the last hidden unit layer contains information about tasks different from the training one (e.g. [20]). This is in fact to be expected. The last layer of HBF is rather independent of the training target and mostly depends on the input part of the training set (see theory and gradient

descent equations in Poggio and Girosi, [7] for the one-hidden layer case). This is exactly true for one-hidden layer RBF networks and holds approximatively for HBFs. The weights from the last hidden layer to the output are instead task/target dependent.

- The result that linear combinations of rectifiers can be equivalent to kernels is *robust* in the sense that it is true for several different nonlinearities such as rectifiers, sigmoids etc. Ramps (e.g. rectifiers) are the most basic ones. *Such robustness is especially attractive for neuroscience.*

5 Appendices

5.1 HW modules are equivalent to kernel machines (a summary of the results in [10])

In the following we summarize the key passages of [10] in proving that HW modules are kernel machines:

1. The feature map
$$\phi(x,t,b) = |\langle t,x\rangle + b|_+$$
(that can be associated to the output of a simple cell, or the basic computational unit of a deep learning architecture) can also be seen as a kernel in itself. The kernel can be a universal kernel. In fact, under the hypothesis of normalization of the vectors x,t we have that $2(|1-\langle t,x\rangle|_+ + |\langle t,x\rangle - 1|_+) = 2|1-\langle t,x\rangle| = \|x-t\|_2^2$ which is a universal kernel (see also th 17 of [21]).
The feature ϕ leads to a kernel
$$K_0(x,x') = \phi^T(x)\phi(x') = \int db\, dt\, |\langle t,x\rangle + b|_+ |\langle t,x'\rangle + b|_+$$
which is a universal kernel being a kernel mean embedding (w.r.t. t,b, see [22]) of the a product of universal kernels.

2. If we explicitly introduce a group of transformations acting on the feature map input i.e. $\phi(x,g,t,b) = |\langle gt,x\rangle + b|_+$ the associated kernel can be written as
$$\tilde{K}(x,x') = \int dg\, dg' \int dt\, db\, |\langle gt,x\rangle + b|_+ |\langle g't,x'\rangle + b|_+ = \int dg\, dg'\, K_0(x,g,g').$$
$\tilde{K}(x,x')$ is the group average of K_0 (see [23]) and can be seen as the mean kernel embedding of K_0 (w.r.t. g,g', see [22]).

3. The kernel $\tilde{K}$ is invariant and, if G is compact, selective i.e.
$$\tilde{K}(x,x') = 1 \;\Leftrightarrow\; x \sim x'.$$
The invariance follows from the fact that any G–group average function is invariant to G transformations. Selectivity follows from the fact that $\tilde{K}$ a universal kernel being a kernel mean embedding of K_0 which is a universal kernel (see [22]).

Remark 1. *If the distribution of the templates t follows a gaussian law the kernel K_0, with an opportune change of variable, can be seen as a particular case of the nth order arc-cosine kernel in [24] for $n = 1$.*

5.2 An example of an explicit calculation for an inner product kernel

Here we note how a simple similarity measure between functions of the form in eq. (3) correspond to a kernel in the case when the inner product between two HW module outputs at the first layer, say $\mu(I), \mu(I')$, is calculated using a step function nonlinearity. Note first that a heaviside step function can be approximated by a sigmoid like function derived by a linear combination of rectifiers of the form:
$$H(x) \sim \alpha(|x|_+ - |x - \frac{1}{\alpha}|_+)$$

for very large values of α. With this specific choice of the nonlinearity we have that the inner product of the HW modules outputs (for fixed t) at the first layer is given by:

$$\langle \mu(I), \mu(I') \rangle = \int \int dg\, dg' (\int db H(b - \langle I, gt \rangle) H(b - \langle I', g't \rangle)).$$

with

$$\mu_b^t(I) = \int dg\, H(b - \langle I, gt \rangle).$$

Assuming that the scalar products, $\langle I, gt \rangle, \langle I', g't \rangle$, range in the interval $[-p, p]$ a direct computation of the integral above by parts shows that, $x = \langle I, gt \rangle, x' = \langle I', gt \rangle$:

$$\begin{aligned}
&\int_{-p}^{p} db H(b-x) H(b-x') \\
&= H(b-x)\Big((b-x')H(b-x') - (x-x')H(x-x')\Big)\Big|_{-p}^{p} \\
&= p - \frac{1}{2}(x + x' + |x - x'|)
\end{aligned}$$

and being

$$\begin{aligned}
\max\{x, x'\} &= \max\left\{x - \frac{1}{2}(x+x'), x' - \frac{1}{2}(x+x')\right\} + \frac{1}{2}(x+x') \\
&= \max\left\{\frac{1}{2}(x'-x), \frac{1}{2}(x-x')\right\} + \frac{1}{2}(x+x') \\
&= \max\left\{-\frac{1}{2}(x-x'), \frac{1}{2}(x-x')\right\} + \frac{1}{2}(x+x') \\
&= +\left|\frac{1}{2}(x-x')\right| + \frac{1}{2}(x+x') \\
&= \frac{1}{2}(x + x' + |x - x'|).
\end{aligned}$$

we showed that

$$K(\langle I', t \rangle, \langle I, t \rangle) = C - \max(\langle I', t \rangle, \langle I, t \rangle)$$

which defines a kernel. If we include the pooling over the transformations and templates we have

$$\tilde{K}(I, I') = p - \int d\lambda(t)\, dg\, dg'\, \max(\langle I', gt \rangle, \langle I, g't \rangle). \tag{9}$$

where $\lambda(t)$ is a probability measure on the templates t. $\tilde{K}$ is again a kernel.
A similar calculation can be repeated at the successive layer leading to kernels of kernels structure.

5.3 Mex

Mex is a generalization of the pooling function. From [15] eq. 1 it is defined as:

$$Mex_{(\{c_i\},\xi)} = \frac{1}{\xi} \log\left(\frac{1}{n} \sum_{i=1}^{n} \exp(\xi c_i)\right) \tag{10}$$

We have

$$Mex_{(\{c_i\},\xi)} \xrightarrow[\xi\to\infty]{} Max_i(c_i)$$
$$Mex_{(\{c_i\},\xi)} \xrightarrow[\xi\to 0]{} Mean_i(c_i)$$
$$Mex_{(\{c_i\},\xi)} \xrightarrow[\xi\to-\infty]{} Min_i(c_i).$$

We can also choose values of ξ in between the ones above, the interpretation is less obvious. The Mex pooling does not define a kernel since is not positive definite in general (see also Th 1 in [15])

5.4 Hyper Basis Functions: minimizing memory in Radial Basis Function networks

We summarize here an old extension by Poggio and Girosi [25] of the classical kernel networks called Radial Basis Functions (RBF). In summary (but see the paper) they extended the theory by defining a general form of these networks which they call Hyper Basis Functions. They have two sets of modifiable parameters: *moving centers* and *adjustable norm-weights*. Moving the centers is equivalent to task-dependent clustering and changing the norm weights is equivalent to task-dependent dimensionality reduction.
A classical RBF has the form

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i K(\|\mathbf{x}-\mathbf{x}_i\|^2), \tag{11}$$

which is a sum of radial functions, each with its *center* $\mathbf{x}_i$ on a distinct data point. Thus the number of radial functions, and corresponding centers, is the same as the number of examples. Eq. (11) is a minimizer solution of

$$H[f] = \sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda \|Pf\|^2 \quad \lambda \in \mathbb{R}^+ \tag{12}$$

where P is a constrain operator (usually a differential operator).
HBF extend RBF in two directions:

1. The computation of a solution of the form (11) has a complexity (number of radial functions) that is independent of the dimensionality of the input space but is on the order of the dimensionality of the training set (number of examples), which can be very high. Poggio and Girosi showed how to justify an approximation of equation (11) in which the number of centers is much smaller than the number of examples and the positions of the centers are modified during learning. The key idea is to consider a specific form of an approximation to the solution of the standard regularization problem.

2. Moving centers are equivalent to the free knots of nonlinear splines. In the context of networks they were first suggested as a potentially useful heuristics by Broomhead and Lowe [26] and used by Moody and Darken [27].

Poggio and Girosi called *Hyper Basis Functions*, in short *HyperBFs*, the most general form of regularization networks based on these extensions plus the use of a weighted norm.

5.4.1 Moving Centers

The solution given by standard regularization theory to the approximation problem can be very expensive in computational terms when the number of examples is very high. The computation of the coefficients of the expansion can become then a very time consuming operation: its complexity grows polynomially with N, (roughly as N^3) since an $N \times N$ matrix has to be inverted. In addition, the probability of ill-conditioning is higher for larger and larger matrices (it grows like N^3 for a $N \times N$ uniformly distributed random matrix) [28]. The way suggested by Poggio and Girosi to reduce the complexity of the problem is as follows. While the exact regularization solution is equivalent to generalized splines with *fixed* knots, the approximated solution is equivalent to generalized splines with *free* knots.

A standard technique, sometimes known as Galerkin's method, that has been used to find approximate solutions of variational problems, is to expand the solution on a finite basis. The approximated solution $f^*(\mathbf{x})$ has then the following form:

$$f^*(\mathbf{x}) = \sum_{i=1}^{n} c_i \phi_i(\mathbf{x}) \tag{13}$$

where $\{\phi_i\}_{i=1}^n$ is a set of linearly independent functions [29]. The coefficients c_i are usually found according to some rule that guarantees a minimum deviation from the true solution. A natural approximation to the exact solution will be then of the form:

$$f^*(\mathbf{x}) = \sum_{\alpha=1}^{n} c_\alpha G(\mathbf{x}; \mathbf{t}_\alpha) \tag{14}$$

where the parameters $\mathbf{t}_\alpha$, that we call "centers", and the coefficients c_α are unknown, and are in general fewer than the data points ($n \leq N$). This form of solution has the desirable property of being an universal approximator for continuous functions [7] and to be the only choice that guarantees that in the case of $n = N$ and $\{\mathbf{t}_\alpha\}_{\alpha=1}^n = \{\mathbf{x}_i\}_{i=1}^n$ the correct solution (of equation (12)) is consistently recovered. We will see later how to find the unknown parameters of this expansion.

5.4.2 How to learn centers' positions

Suppose that we look for an approximated solution of the regularization problem of the form

$$f^*(\mathbf{x}) = \sum_{\alpha=1}^{n} c_\alpha G(\|\mathbf{x} - \mathbf{t}_\alpha\|^2) \tag{15}$$

We now have the problem of finding the n coefficients c_α, the $d \times\ n$ coordinates of the centers $\mathbf{t}_\alpha$. We can use the natural definition of optimality given by the functional H. We then impose the condition that the set $\{c_\alpha, \mathbf{t}_\alpha | \alpha = 1, ..., n\}$ must be such that they minimizes $H[f^*]$, and the following equations must be satisfied:

$$\frac{\partial H[f^*]}{\partial c_\alpha} = 0\ , \qquad \frac{\partial H[f^*]}{\partial \mathbf{t}_\alpha} = 0, \quad \alpha = 1, ..., n\ .$$

Gradient-descent is probably the simplest approach for attempting to find the solution to this problem, though, of course, it is not guaranteed to converge. Several other iterative methods, such as versions of conjugate gradient and simulated annealing [30] may be more efficient than gradient descent and should be used in practice. Since the function $H[f^*]$ to minimize is

in general non-convex, a stochastic term in the gradient descent equations may be advisable to avoid local minima. In the stochastic gradient descent method the values of c_α, $\mathbf{t}_\alpha$ and $\mathbf{M}$ that minimize $H[f^*]$ are regarded as the coordinates of the stable fixed point of the following stochastic dynamical system:

$$\dot{c}_\alpha = -\omega \frac{\partial H[f^*]}{\partial c_\alpha} + \eta_\alpha(t), \quad \alpha = 1, \ldots, n$$

$$\dot{\mathbf{t}}_\alpha = -\omega \frac{\partial H[f^*]}{\partial \mathbf{t}_\alpha} + \boldsymbol{\mu}_\alpha(t), \quad \alpha = 1, \ldots, n$$

where $\eta_\alpha(t)$, $\boldsymbol{\mu}_\alpha(t)$ are white noise of zero mean and ω is a parameter determining the microscopic timescale of the problem and is related to the rate of convergence to the fixed point. Defining

$$\Delta_i \equiv y_i - f^*(\mathbf{x}) = y_i - \sum_{\alpha=1}^{n} c_\alpha G(\|\mathbf{x}_i - \mathbf{t}_\alpha\|^2)$$

we obtain

$$H[f^*] = H_{\mathbf{c},\mathbf{t}} = \sum_{i=1}^{N} (\Delta_i)^2.$$

The important quantities – that can be used in more efficient schemes than gradient descent – are

- for the c_α

$$\frac{\partial H[f^*]}{\partial c_\alpha} = -2 \sum_{i=1}^{N} \Delta_i G(\|\mathbf{x}_i - \mathbf{t}_\alpha\|^2) \ ; \tag{16}$$

- for the centers t_α

$$\frac{\partial H[f^*]}{\partial \mathbf{t}_\alpha} = 4 c_\alpha \sum_{i=1}^{N} \Delta_i G'(\|\mathbf{x}_i - \mathbf{t}_\alpha\|^2)(\mathbf{x}_i - \mathbf{t}_\alpha) \tag{17}$$

Remarks

1. Equation (16) has a simple interpretation: the correction is equal to the sum over the examples of the products between the error on that example and the "activity" of the "unit" that represents with its center that example. Notice that $H[f^*]$ is quadratic in the coefficients c_α, and if the centers are kept fixed, it can be shown [7] that the optimal coefficients are given by

$$\mathbf{c} = (G^T \ G + \lambda g)^{-1} G^T \mathbf{y} \tag{18}$$

   where we have defined $(\mathbf{y})_i = y_i$, $(\mathbf{c})_\alpha = c_\alpha$, $(G)_{i\alpha} = G(\mathbf{x}_i; \mathbf{t}_\alpha)$ and $(g)_{\alpha\beta} = G(\mathbf{t}_\alpha; \mathbf{t}_\beta)$. If λ is let go to zero, the matrix on the right side of equation (18) converges to the pseudo-inverse of G [31] and if the Green's function is radial the approximation method of [26] is recovered.

2. Equation (17) is similar to task-dependent clustering [7]. This can be best seen by assuming that Δ_i are constant: then the gradient descent updating rule makes the centers move as a function of the majority of the data, that is of the position of the clusters. In this case a technique similar to the k-means algorithm is recovered, [32, 27]. Equating

$\frac{\partial H[f^*]}{\partial \mathbf{t}_\alpha}$ to zero we notice that the optimal centers $\mathbf{t}_\alpha$ satisfy the following set of nonlinear equations:

$$\mathbf{t}_\alpha = \frac{\sum_i P_i^\alpha \mathbf{x}_i}{\sum_i P_i^\alpha} \quad \alpha = 1, \ldots, n$$

where $P_i^\alpha = \Delta_i G'(\|\mathbf{x}_i - \mathbf{t}_\alpha\|^2)$. The optimal centers are then a weighted sum of the data points. The weight P_i^α of the data point i for a given center $\mathbf{t}_\alpha$ is high if the interpolation error Δ_i is high there *and* the radial basis function centered on that knot changes quickly in a neighborhood of the data point. This observation suggests faster update schemes, in which a suboptimal position of the centers is first found and then the c_α are determined, similarly to the algorithm developed and tested successfully by Moody and Darken [27].

5.4.3 An algorithm

It seems natural to try to find a reasonable initial value for the parameters $\mathbf{c}, \mathbf{t}_\alpha$, to start the stochastic minimization process. In the absence of more specific prior information the following heuristics seems reasonable.

- Set the number of centers and set the centers' positions to positions suggested by cluster analysis of the data (or more simply to a subset of the examples' positions).
- Use matrix pseudo-inversion to find the c_α.
- Use the $\mathbf{t}_\alpha$, and c_α found so far as initial values for the stochastic gradient descent equations.

Experiments with movable centers and movable weights have been performed in the context of object recognition (Poggio and Edelman, [33]; Edelman and Poggio, [34]) and approximation of multivariate functions.

5.4.4 Remarks

1. Equation (17) is similar to a clustering process.
2. In the case of N examples, $n = N$ fixed centers, there are enough data to constrain the N coefficients c_α to be found. Moving centers add another nd parameters (d is the number of input components). Thus the number of examples N must be sufficiently large to constrain adequately the free parameters – n d-dimensional centers, n coefficients c_α. Thus

$$N >> n + nd.$$

Acknowledgment

This work was supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF – 1231216. This work was also supported by A*STAR JCO VIP grant #1335h00098. Part of the work was done in Singapore at the Institute for Infocomm under REVIVE funding. TP thanks A*Star for its hospitality.

References

[1] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193–202, Apr. 1980.

[2] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition," *Nature Neuroscience*, vol. 3,11, 2000.

[3] T. Serre, A. Oliva, and T. Poggio, "A feedforward architecture accounts for rapid categorization," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 104, no. 15, pp. 6424–6429, 2007.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, B. M., A. Berg, and F.-F. L., "Large scale visual recognition challenge," *ImageNet, arXiv:1409.0575*, 2014.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," *ArXiv e-prints*, Sept. 2014.

[6] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks.," *CoRR*, vol. abs/1311.2901, 2013.

[7] T. Poggio and F. Girosi, "A theory of networks for approximation and learning," *Laboratory, Massachusetts Institute of Technology*, vol. A.I. memo n1140, 1989.

[8] F. Anselmi, J. Z. Leibo, L. Rosasco, J. Mutch, A. Tacchetti, and T. Poggio, "Unsupervised learning of invariant representations in hierarchical architectures," *arXiv preprint arXiv:1311.4158*, 2013.

[9] F. Anselmi, L. Rosasco, and T. Poggio, "On invariance and selectivity in representation learning," *arXiv:1503.05938 and CBMM memo n 29*, 2015.

[10] L. Rosasco and T. Poggio, "Convolutional layers build invariant and selective reproducing kernels," *CBMM Memo, in preparation*, 2015.

[11] D. Hubel and T. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, no. 1, p. 106, 1962.

[12] L. Breiman, "Hinging hyperplanes for regression, classification, and function approximation," Tech. Rep. 324, Department of Statistics University of California Berkeley, California 94720, 1991.

[13] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Computation*, vol. 7, pp. 219–269, 1995.

[14] M. Kouh and T. Poggio, "A canonical neural circuit for cortical nonlinear operations," *Neural computation*, vol. 20, no. 6, pp. 1427–1451, 2008.

[15] N. Cohen and A. Shashua, "Simnets: A generalization of convolutional networks," *CoRR*, vol. abs/1410.0781, 2014.

[16] T. Poggio and F. Girosi, "Regularization algorithms for learning that are equivalent to multilayer networks," *Science*, vol. 247, pp. 978–982, 1990.

[17] J. Mihalik, "Hierarchical vector quantization. of images in transform domain.," *ELEKTROTECHN. CA5, 43, NO. 3. 92,94.*, 1992.

[18] F. Girosi and T. Poggio, "Representation properties of networks: Kolmogorov's theorem is irrelevant," *Neural Computation*, vol. 1, no. 4, pp. 465–469, 1989.

[19] F. Girosi and T. Poggio, "Networks and the best approximation property," vol. 63, pp. 169–176, 1990.

[20] D. Yamins, H. Hong, C. Cadieu, E. Solomon, D. Seibert, and J. DiCarlo, "Performance-optimized hierarchical models predict neural responses in higher visual cortex," *Proceedings of the National Academy of Sciences*, 2014.

[21] C. A. Micchelli, Y. Xu, and H. Zhang, "Universal kernels.," *Journal of Machine Learning Research*, vol. 6, pp. 2651–2667, 2006.

[22] B. K. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schölkopf, and G. R. G. Lanckriet, "Hilbert space embeddings and metrics on probability measures," *Journal of Machine Learning Research*, vol. 11, pp. 1517–1561, 2010.

[23] B. Haasdonk and H. Burkhardt, "Invariant kernel functions for pattern analysis and machine learning," *Mach. Learn.*, vol. 68, pp. 35–61, July 2007.

[24] Y. Cho and L. K. Saul, "Kernel methods for deep learning," in *Advances in Neural Information Processing Systems 22* (Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, eds.), pp. 342–350, Curran Associates, Inc., 2009.

[25] T. Poggio and F. Girosi, "Extensions of a theory of networks for approximation and learning: dimensionality reduction and clustering," *Laboratory, Massachusetts Institute of Technology*, vol. A.I. memo n 1167, 1994.

[26] D. S. Broomhead and D. Lowe, "Multivariable Functional Interpolation and Adaptive Networks," *Complex Systems 2*, pp. 321–355, 1988.

[27] J. Moody and C. J. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Computation*, vol. 1, no. 2, pp. 281–294, 1989.

[28] J. Demmel, "The geometry of ill-conditioning," *J. Complexity*, vol. 3, pp. 201–229, 1987.

[29] S. Mikhlin, *The problem of the minimum of a quadratic functional*. San Francisco, CA: Holden-Day, 1965.

[30] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 219–227, 1983.

[31] A. E. Albert, *Regression and the Moore-Penrose pseudoinverse*. Mathematics in science and engineering, New York, London: Academic Press, 1972.

[32] J. MacQueen, "Some methods of classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symposium on Math., Stat., and Prob.* (L. LeCam and J. Neyman, eds.), p. 281, Berkeley, CA: U. California Press, 1967.

[33] T. Poggio and S. Edelman, "A network that learns to recognize 3D objects," *Nature*, vol. 343, pp. 263–266, 1990.

[34] S. Edelman and T. Poggio, "Bringing the grandmother back into the picture: a memory-based view of object recognition," a.i. memo 1181, mitai, 1990.