Foundations of Deep Learning from a Kernel Point of View

Julien Mairal

Inria Grenoble

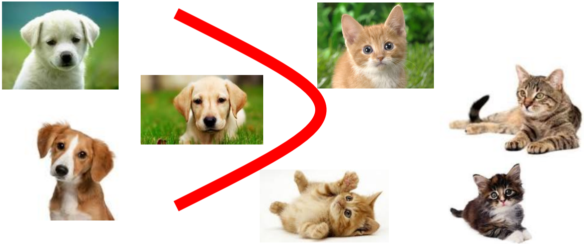
Nantes, Mascot-Num, 2018
Part I

Part I: Several Paradigms in Machine Learning

Common paradigm: optimization for machine learning

Optimization is central to machine learning. For instance, in supervised learning, the goal is to learn a **prediction function** $f : \mathcal{X} \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\ldots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}} .$$



[Vapnik, 1995, Bottou, Curtis, and Nocedal, 2016]...

Common paradigm: optimization for machine learning

Optimization is central to machine learning. For instance, in supervised learning, the goal is to learn a **prediction function** $f : \mathcal{X} \rightarrow \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\ldots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}} .$$

The scalars y_i are in

- $\{-1, +1\}$ for **binary** classification problems.
- $\{1, \ldots, K\}$ for **multi-class** classification problems.
- $\mathbb{R}$ for **regression** problems.
- $\mathbb{R}^k$ for **multivariate regression** problems.

Common paradigm: optimization for machine learning

Optimization is central to machine learning. For instance, in supervised learning, the goal is to learn a **prediction function** $f : \mathcal{X} \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\ldots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}} .$$

Example with linear models: logistic regression, SVMs, *etc.*

- assume there exists a linear relation between y and features x in $\mathbb{R}^p$.
- $f(x) = w^\top x + b$ is parametrized by w, b in $\mathbb{R}^{p+1}$;
- L is often a **convex** loss function;
- $\Omega(f)$ is often the squared ℓ_2-norm $\|w\|^2$.

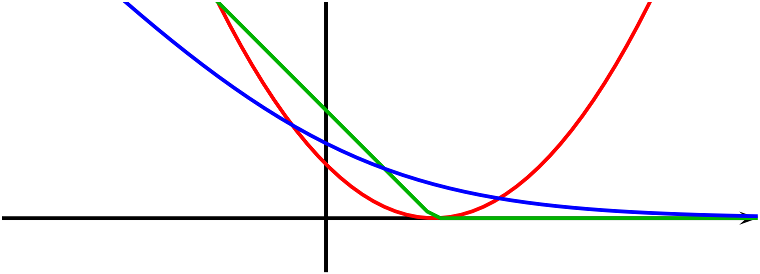
Common paradigm: optimization for machine learning

A few examples of linear models with no bias b:

Ridge regression: $$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2}(y_i - w^\top x_i)^2 + \lambda \|w\|_2^2.$$

Linear SVM: $$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i w^\top x_i) + \lambda \|w\|_2^2.$$

Logistic regression: $$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y_i w^\top x_i}\right) + \lambda \|w\|_2^2.$$



Common paradigm: optimization for machine learning

The previous formulation is called *empirical risk minimization*; it follows a classical scientific paradigm:

1. **observe** the world (gather data);
2. **propose models** of the world (design and learn);
3. **test** on new data (estimate the generalization error).

Common paradigm: optimization for machine learning

The previous formulation is called *empirical risk minimization*; it follows a classical scientific paradigm:

1. **observe** the world (gather data);
2. **propose models** of the world (design and learn);
3. **test** on new data (estimate the generalization error).

A general principle

It underlies many paradigms:

- **deep neural networks,**
- **kernel methods,**
- sparse estimation. (tomorrow's lecture)

Common paradigm: optimization for machine learning

The previous formulation is called *empirical risk minimization*; it follows a classical scientific paradigm:

1. **observe** the world (gather data);
2. **propose models** of the world (design and learn);
3. **test** on new data (estimate the generalization error).

Even with simple linear models, it leads to challenging problems in optimization: develop algorithms that

- **scale** both in the problem size n and dimension p;
- are able to **exploit the problem structure** (sum, composite);
- come with **convergence and numerical stability** guarantees;
- come with **statistical guarantees**.

Common paradigm: optimization for machine learning

The previous formulation is called *empirical risk minimization*; it follows a classical scientific paradigm:

1. **observe** the world (gather data);
2. **propose models** of the world (design and learn);
3. **test** on new data (estimate the generalization error).

It is not limited to supervised learning

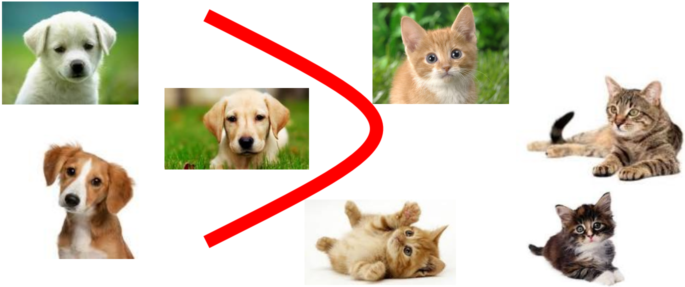
$$\min_{f \in \mathcal{F}} \quad \frac{1}{n} \sum_{i=1}^{n} L(f(x_i)) \; + \; \lambda \Omega(f).$$

- L is not a classification loss any more;
- K-means, PCA, EM with mixture of Gaussian, matrix factorization,... can be expressed that way.

Paradigm 1: Deep neural networks

The goal is to learn a **prediction function** $f : \mathbb{R}^p \to \mathbb{R}$ given labeled training data $(x_i, y_i)_{i=1,\ldots,n}$ with x_i in $\mathbb{R}^p$, and y_i in $\mathbb{R}$:

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}}.$$



Paradigm 1: Deep neural networks

The goal is to learn a **prediction function** $f : \mathbb{R}^p \to \mathbb{R}$ given labeled training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathbb{R}^p$, and y_i in $\mathbb{R}$:

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}}.$$

What is specific to multilayer neural networks?

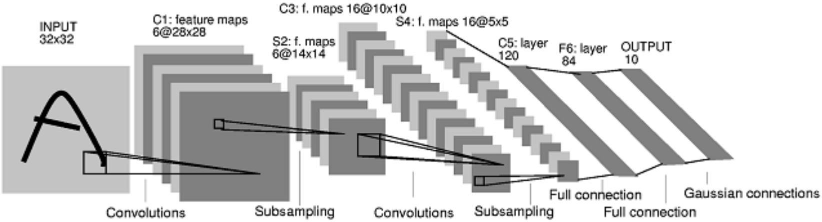
- The "neural network" space $\mathcal{F}$ is explicitly parametrized by:

$$f(x) = \sigma_k(\mathbf{A}_k \sigma_{k-1}(\mathbf{A}_{k-1} \dots \sigma_2(\mathbf{A}_2 \sigma_1(\mathbf{A}_1 x)) \dots)).$$

- Linear operations are either unconstrained (fully connected) or involve parameter sharing (e.g., convolutions).
- Finding the optimal $\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_k$ yields a **non-convex** optimization problem in **huge dimension.**

Paradigm 1: Deep neural networks

Picture from LeCun et al. [1998]

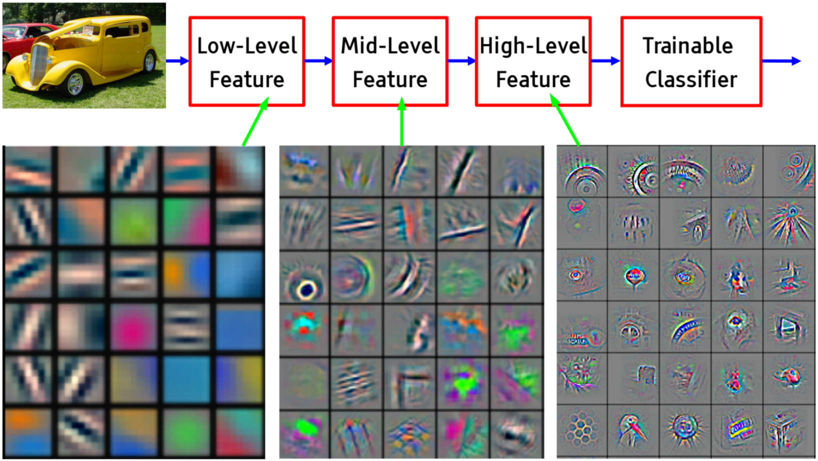


What are the main features of CNNs?

- they capture **compositional** and **multiscale** structures in images;
- they provide some **invariance**;
- they model **local stationarity** of images at several scales;
- they are **state-of-the-art** in many fields.

Paradigm 1: Deep neural networks

The keywords: **multi-scale, compositional, invariant, local features**.
Picture from Y. LeCun's tutorial:



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Paradigm 1: Deep neural networks

Picture from Olah et al. [2017]:

Edges (layer conv2d0) **Textures** (layer mixed3a) **Patterns** (layer mixed4a)

Paradigm 1: Deep neural networks

Picture from Olah et al. [2017]:

Patterns (layer mixed4a)   **Parts** (layers mixed4b & mixed4c)   **Objects** (layers mixed4d & mixed4e)

Paradigm 1: Deep neural networks

ImageNet: 1000 image categories, 10M hand-labeled images.
Picture from unknown source:

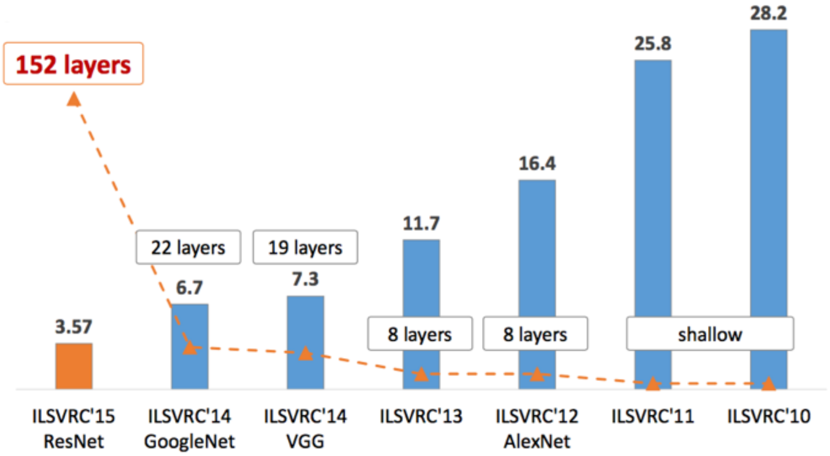


Figure: Top-5 error rate

Paradigm 1: Deep neural networks

What are current high-potential problems to solve?

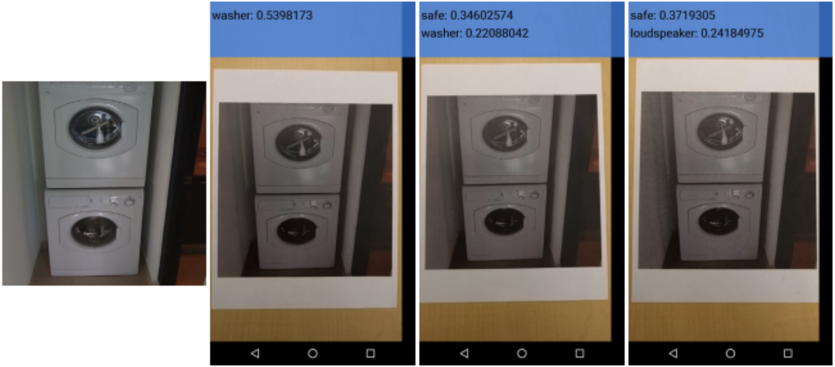
1. lack of **stability** (see next slide).
2. learning with **few labeled data**.
3. learning with **no supervision** (see Tab. from Bojanowski and Joulin, 2017).

Method	Acc@1
Random (Noroozi & Favaro, 2016)	12.0
SIFT+FV (Sánchez et al., 2013)	55.6
Wang & Gupta (2015)	29.8
Doersch et al. (2015)	30.4
Zhang et al. (2016)	35.2
[1]Noroozi & Favaro (2016)	38.1
BiGAN (Donahue et al., 2016)	32.2
NAT	36.0

Table 3. Comparison of the proposed approach to state-of-the-art unsupervised feature learning on ImageNet. A full multi-layer perceptron is retrained on top of the features. We compare to several self-supervised approaches and an unsupervised approach.

Paradigm 1: Deep neural networks

Illustration of instability. Picture from Kurakin et al. [2016].



(a) Image from dataset (b) Clean image (c) Adv. image, $\epsilon = 4$ (d) Adv. image, $\epsilon = 8$

Figure: Adversarial examples are generated by computer; then printed on paper; a new picture taken on a smartphone fools the classifier.

Paradigm 1: Deep neural networks

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda\Omega(f)}_{\text{regularization}}.$$

The issue of regularization

- today, heuristics are used (DropOut, weight decay, early stopping)...
- ...but they are not sufficient.
- how to **control variations of prediction functions**?

$$|f(x) - f(x')| \text{ should be close if } x \text{ and } x' \text{ are "similar"}.$$

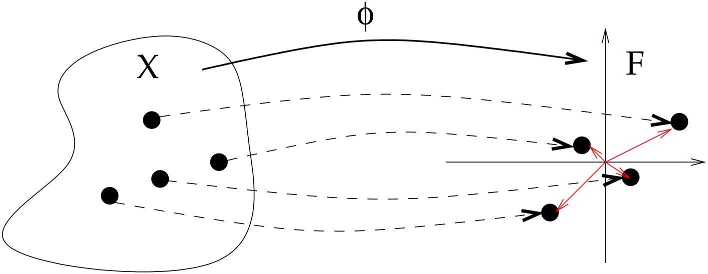
- what does it mean for x and x' to be "similar"?
- what should be a good **regularization function** Ω?

Paradigm 2: Kernel methods

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \|f\|_{\mathcal{H}}^2.$$

- **map** data x in $\mathcal{X}$ to a Hilbert space and work with **linear forms**:

$$\varphi : \mathcal{X} \to \mathcal{H} \quad \text{and} \quad f(x) = \langle \varphi(x), f \rangle_{\mathcal{H}}.$$



[Shawe-Taylor and Cristianini, 2004, Schölkopf and Smola, 2002]...

Paradigm 2: Kernel methods

$$\min_{f \in \mathcal{H}} \quad \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) \; + \; \lambda \|f\|_{\mathcal{H}}^2.$$

First purpose: embed data in a vectorial space where

- many **geometrical operations** exist (angle computation, projection on linear subspaces, definition of barycenters....).
- one may learn potentially **rich infinite-dimensional models**.
- **regularization** is natural (see next...)

Paradigm 2: Kernel methods

$$\min_{f \in \mathcal{H}} \quad \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) \; + \; \lambda \|f\|_{\mathcal{H}}^2.$$

First purpose: embed data in a vectorial space where

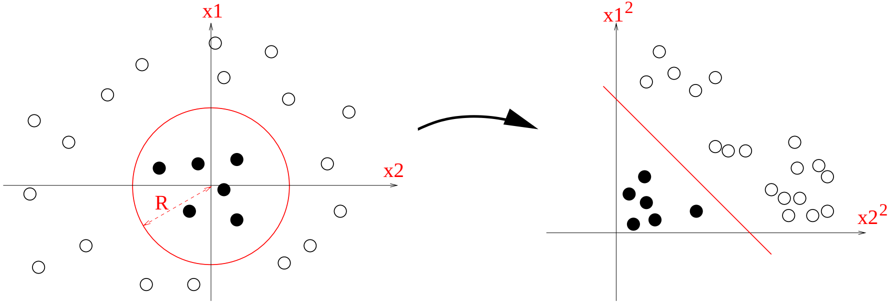
- many **geometrical operations** exist (angle computation, projection on linear subspaces, definition of barycenters....).
- one may learn potentially **rich infinite-dimensional models**.
- **regularization** is natural (see next...)

The principle is **generic** and does not assume anything about the nature of the set $\mathcal{X}$ (vectors, sets, graphs, sequences).

Paradigm 2: Kernel methods

Second purpose: unhappy with the current Euclidean structure?

- lift data to a higher-dimensional space with **nicer properties** (e.g., linear separability, clustering structure).
- then, the **linear** form $f(x) = \langle \varphi(x), f \rangle_{\mathcal{H}}$ in $\mathcal{H}$ may correspond to a **non-linear** model in $\mathcal{X}$.

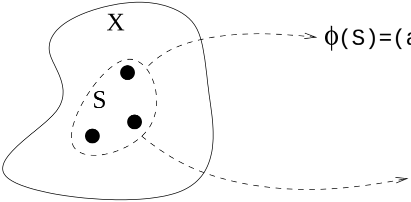


Paradigm 2: Kernel methods

How does it work? representation by pairwise comparisons

- Define a "comparison function": $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$.
- Represent a set of n data points $\mathcal{S} = \{x_1, \ldots, x_n\}$ by the $n \times n$ **matrix**:

$$\mathbf{K}_{ij} := K(x_i, x_j).$$



ϕ(S)=(aatcgagtcac,atggacgtct,tgcactact)

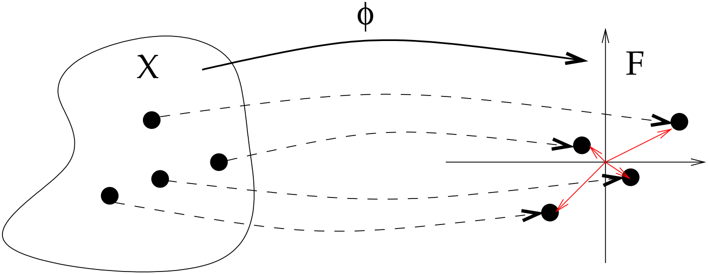
$$K = \begin{pmatrix} 1 & 0.5 & 0.3 \\ 0.5 & 1 & 0.6 \\ 0.3 & 0.6 & 1 \end{pmatrix}$$

Paradigm 2: Kernel methods

Theorem (Aronszajn, 1950)

$K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite kernel if and only if there exists a Hilbert space $\mathcal{H}$ and a mapping $\varphi : \mathcal{X} \to \mathcal{H}$, such that

$$\text{for any } x, x' \text{ in } \mathcal{X}, \qquad K(x,x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}}.$$



Paradigm 2: Kernel methods

Mathematical details

- the only thing we require about K is **symmetry** and **positive definiteness**

$$\forall x_1, \ldots, x_n \in \mathcal{X}, \alpha_1, \ldots, \alpha_n \in \mathbb{R}, \quad \sum_{ij} \alpha_i \alpha_j K(x_i, x_j) \geq 0.$$

- then, there exists a Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathbb{R}$, called the **reproducing kernel Hilbert space (RKHS)** such that

$$\forall f \in \mathcal{H}, x \in \mathcal{X}, \qquad f(x) = \langle \varphi(x), f \rangle_{\mathcal{H}},$$

and the mapping $\varphi : \mathcal{X} \to \mathcal{H}$ (from Aronszajn's theorem) satisfies

$$\varphi(x) : y \mapsto K(x, y).$$

Paradigm 2: Kernel methods

Why mapping data in $\mathcal{X}$ to the functional space $\mathcal{H}$?

- it becomes feasible to learn a prediction function $f \in \mathcal{H}$:

$$\min_{f \in \mathcal{H}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \|f\|_{\mathcal{H}}^2}_{\text{regularization}} .$$

  (why? the solution lives in a finite-dimensional hyperplane).

- **non-linear** operations in $\mathcal{X}$ become **inner-products** in $\mathcal{H}$ since

$$\forall f \in \mathcal{H}, x \in \mathcal{X}, \quad f(x) = \langle \varphi(x), f \rangle_{\mathcal{H}}.$$

- the norm of the RKHS is a **natural regularization function**:

$$|f(x) - f(x')| \leq \|f\|_{\mathcal{H}} \|\varphi(x) - \varphi(x')\|_{\mathcal{H}}.$$

Paradigm 2: Kernel methods

What are the main features of kernel methods?

- builds **well-studied functional spaces** to do machine learning;
- **decoupling** of data representation and learning algorithm;
- typically, **convex optimization problems** in a supervised context;
- **versatility**: applies to vectors, sequences, graphs, sets,...;
- **natural regularization function** to control the learning capacity;

[Shawe-Taylor and Cristianini, 2004, Schölkopf and Smola, 2002, Müller et al., 2001]

Paradigm 2: Kernel methods

What are the main features of kernel methods?

- builds **well-studied functional spaces** to do machine learning;
- **decoupling** of data representation and learning algorithm;
- typically, **convex optimization problems** in a supervised context;
- **versatility**: applies to vectors, sequences, graphs, sets,...;
- **natural regularization function** to control the learning capacity;

But...

- **decoupling** of data representation and learning may not be a good thing, according to recent **supervised** deep learning success.
- requires **kernel design**.
- $O(n^2)$ **scalability problems**.

[Shawe-Taylor and Cristianini, 2004, Schölkopf and Smola, 2002, Müller et al., 2001]

Kernels and deep learning

What is the relation?

- it is possible to design functional spaces $\mathcal{H}$ where deep neural networks live [Mairal, 2016].

$$f(x) = \sigma_k(\mathbf{A}_k \sigma_{k-1}(\mathbf{A}_{k-1} \ldots \sigma_2(\mathbf{A}_2 \sigma_1(\mathbf{A}_1 x)) \ldots)) = \langle f, \varphi(x) \rangle_{\mathcal{H}}.$$

- we call the construction "**convolutional kernel networks**" (in short, replace $u \mapsto \sigma(\langle a, u \rangle)$ by a kernel mapping $u \mapsto \varphi_k(u)$.

Why do we care?

- $\varphi(x)$ is related to the **network architecture** and is **independent of training data**. Is it stable? Does it lose signal information?

Kernels and deep learning

What is the relation?

- it is possible to design functional spaces $\mathcal{H}$ where deep neural networks live [Mairal, 2016].

$$f(x) = \sigma_k(\mathbf{A}_k \sigma_{k-1}(\mathbf{A}_{k-1} \ldots \sigma_2(\mathbf{A}_2 \sigma_1(\mathbf{A}_1 x)) \ldots)) = \langle f, \varphi(x) \rangle_{\mathcal{H}}.$$

- we call the construction "**convolutional kernel networks**" (in short, replace $u \mapsto \sigma(\langle a, u \rangle)$ by a kernel mapping $u \mapsto \varphi_k(u)$.

Why do we care?

- $\varphi(x)$ is related to the **network architecture** and is **independent of training data**. Is it stable? Does it lose signal information?
- f is a **predictive model**. Can we control its stability?

$$|f(x) - f(x')| \leq \|f\|_{\mathcal{H}} \|\varphi(x) - \varphi(x')\|_{\mathcal{H}}.$$

Part II: Convolutional Kernel Networks

Challenges of deep kernel machines

- **Build functional spaces for deep learning**, where we can quantify **invariance and stability** to perturbations, **signal recovery** properties, and the **complexity** of the function class.
- do deep learning with a **geometrical interpretation** (learn collections of linear subspaces, perform projections).
- exploit kernels for **structured objects** (graph, sequences) within deep architectures.
- show that **end-to-end** learning is natural with kernel methods.
- build models that are **stable** by design?

Convolutional Kernel Networks

The (happy?) **marriage** of kernel methods and CNNs

1. **a multilayer convolutional kernel for images**: A hierarchy of kernels for local image neighborhoods (aka, receptive fields).
2. **unsupervised scheme for large-scale learning**: the kernel beeing too computationally expensive, the Nyström approximation at each layer yields a new type of unsupervised deep neural network.
3. **end-to-end learning**: learning subspaces in the RKHSs can be achieved with a supervised loss function.

Convolutional Kernel Networks

The (happy?) **marriage** of kernel methods and CNNs

1. **a multilayer convolutional kernel for images**: A hierarchy of kernels for local image neighborhoods (aka, receptive fields).
2. **unsupervised scheme for large-scale learning**: the kernel beeing too computationally expensive, the Nyström approximation at each layer yields a new type of unsupervised deep neural network.
3. **end-to-end learning**: learning subspaces in the RKHSs can be achieved with a supervised loss function.

First proof of concept with unsupervised learning

- J. Mairal, P. Koniusz, Z. Harchaoui and C. Schmid. Convolutional Kernel Networks. NIPS 2014.

Application to image retrieval

- M. Paulin, J. Mairal, M. Douze, Z. Harchaoui, F. Perronnin, and C. Schmid. Convolutional Patch Representations for Image Retrieval: an Unsupervised Approach. IJCV. 2017.

Convolutional Kernel Networks

The (happy?) **marriage** of kernel methods and CNNs

1. **a multilayer convolutional kernel for images**: A hierarchy of kernels for local image neighborhoods (aka, receptive fields).
2. **unsupervised scheme for large-scale learning**: the kernel beeing too computationally expensive, the Nyström approximation at each layer yields a new type of unsupervised deep neural network.
3. **end-to-end learning**: learning subspaces in the RKHSs can be achieved with a supervised loss function.

Conceptually better model, with supervised learning

- J. Mairal. End-to-End Kernel Learning with Supervised Convolutional Kernel Networks. NIPS 2016.

Application to biological sequences

- D. Chen, L. Jacob, and J. Mairal. Predicting Transcription Factor Binding Sites with Convolutional Kernel Networks. preprint BiorXiv. 2017.

Convolutional Kernel Networks

The (happy?) marriage of kernel methods and CNNs

1. **a multilayer convolutional kernel for images**: A hierarchy of kernels for local image neighborhoods (aka, receptive fields).
2. **unsupervised scheme for large-scale learning**: the kernel beeing too computationally expensive, the Nyström approximation at each layer yields a new type of unsupervised deep neural network.
3. **end-to-end learning**: learning subspaces in the RKHSs can be achieved with a supervised loss function.

Theory of stability and invariance

- A. Bietti and J. Mairal. Group Invariance, Stability to Deformations, and Complexity of Deep Convolutional Representations. preprint arXiv 2017.
- A. Bietti and J. Mairal. Invariance and Stability of Deep Convolutional Representations. NIPS 2017.

Convolutional Kernel Networks

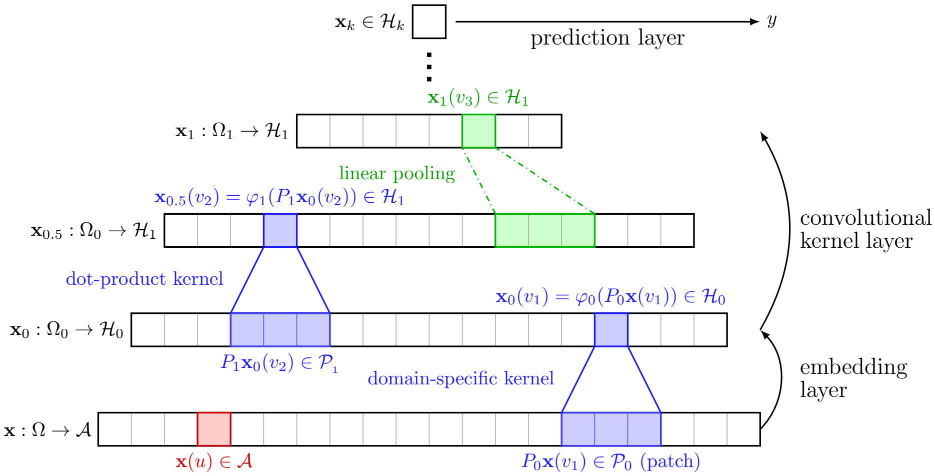


Illustration of multilayer convolutional kernel for 1D discrete signals.

Convolutional Kernel Networks

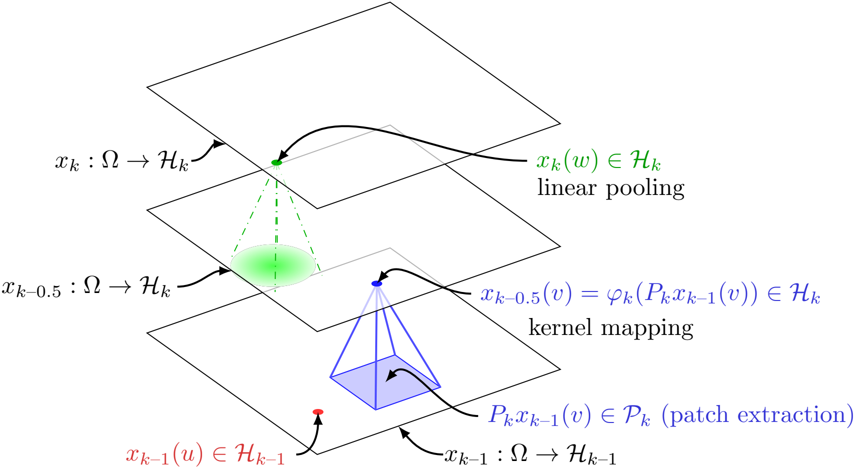
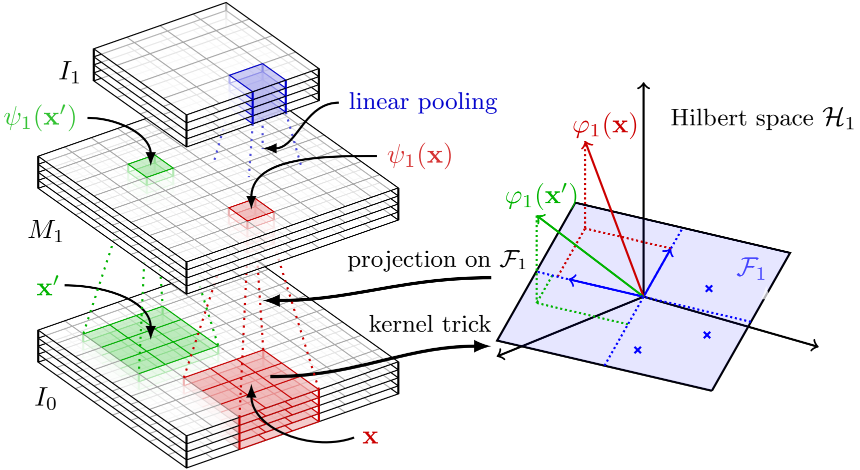


Illustration of multilayer convolutional kernel for 2D continuous signals.

Convolutional Kernel Networks



Learning mechanism of CKNs between layers 0 and 1.

Convolutional Kernel Networks

Main principles

- A multilayer kernel, which builds upon similar principles as a convolutional neural net (**multiscale, local stationarity**).

Convolutional Kernel Networks

Main principles

- A multilayer kernel, which builds upon similar principles as a convolutional neural net (**multiscale, local stationarity**).
- When going up in the hierarchy, we represent **larger neighborhoods with more invariance**;

Convolutional Kernel Networks

Main principles

- A multilayer kernel, which builds upon similar principles as a convolutional neural net (**multiscale, local stationarity**).
- When going up in the hierarchy, we represent **larger neighborhoods with more invariance**;
- The first layer may encode **domain-specific knowledge**;

Convolutional Kernel Networks

Main principles

- A multilayer kernel, which builds upon similar principles as a convolutional neural net (**multiscale, local stationarity**).
- When going up in the hierarchy, we represent **larger neighborhoods with more invariance**;
- The first layer may encode **domain-specific knowledge**;
- We build a sequence of functional spaces and data representations that are **decoupled from learning**...

Convolutional Kernel Networks

Main principles

- A multilayer kernel, which builds upon similar principles as a convolutional neural net (**multiscale, local stationarity**).
- When going up in the hierarchy, we represent **larger neighborhoods with more invariance**;
- The first layer may encode **domain-specific knowledge**;
- We build a sequence of functional spaces and data representations that are **decoupled from learning**...
- But, we learn **linear subspaces** in RKHSs, where we project data, providing a new type of CNN with a **geometric interpretation**.

Convolutional Kernel Networks

Main principles

- A multilayer kernel, which builds upon similar principles as a convolutional neural net (**multiscale, local stationarity**).
- When going up in the hierarchy, we represent **larger neighborhoods with more invariance**;
- The first layer may encode **domain-specific knowledge**;
- We build a sequence of functional spaces and data representations that are **decoupled from learning**...
- But, we learn **linear subspaces** in RKHSs, where we project data, providing a new type of CNN with a **geometric interpretation**.
- Learning may be **unsupervised** (reduce approximation error) or **supervised** (via backpropagation).

Basic component: dot-product kernels

A simple link between kernels and neural networks can be obtained by considering dot-product kernels.

A classical old result [Schoenberg, 1942]

Let $\mathcal{X} = \mathbb{S}^{d-1}$ be the unit sphere of $\mathbb{R}^d$. The kernel $K : \mathcal{X}^2 \to \mathbb{R}$

$$K(x, y) = \kappa(\langle x, y \rangle_{\mathbb{R}^d})$$

is positive definite for any dimension p if and only if κ is smooth, non-zero, and its Taylor expansion coefficients are non-negative.

Remark

- the proposition holds if $\mathcal{X}$ is the unit sphere of some Hilbert space and $\langle x, y \rangle_{\mathbb{R}^d}$ is replaced by the corresponding inner-product.

[Smola, Ovari, and Williamson, 2001]...

Basic component: dot-product kernels

linear kernel	$\langle z, z' \rangle$
exponential kernel	$e^{\alpha(\langle z,z' \rangle - 1)}$
inverse polynomial kernel	$\frac{1}{2-\langle z,z' \rangle}$
polynomial kernel of degree p	$(c + \langle z, z' \rangle)^p$
arc-cosine kernel of degree 1	$\frac{1}{\pi}\left(\sin(\theta) + (\pi - \theta)\cos(\theta)\right)$ with $\theta = \arccos(\langle z, z' \rangle)$
Vovk's kernel of degree 3	$\frac{1}{3}\left(\frac{1-\langle z,z' \rangle^3}{1-\langle z,z' \rangle}\right) = \frac{1}{3}\left(1 + \langle z, z' \rangle + \langle z, z' \rangle^2\right)$

Basic component: dot-product kernels

linear kernel	$\langle z, z' \rangle$
exponential kernel	$e^{\alpha(\langle z, z' \rangle - 1)}$
inverse polynomial kernel	$\frac{1}{2 - \langle z, z' \rangle}$
polynomial kernel of degree p	$(c + \langle z, z' \rangle)^p$
arc-cosine kernel of degree 1	$\frac{1}{\pi}\left(\sin(\theta) + (\pi - \theta)\cos(\theta)\right)$ with $\theta = \arccos(\langle z, z' \rangle)$
Vovk's kernel of degree 3	$\frac{1}{3}\left(\frac{1 - \langle z, z' \rangle^3}{1 - \langle z, z' \rangle}\right) = \frac{1}{3}\left(1 + \langle z, z' \rangle + \langle z, z' \rangle^2\right)$

Remark

if $\|z\| = \|z'\| = 1$, the exponential kernel recovers the Gaussian kernel

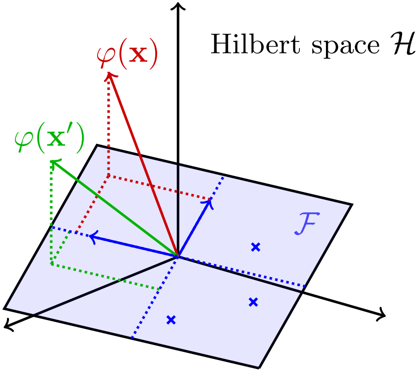
$$\kappa_{\text{exp}}(\langle z, z' \rangle) = e^{\alpha(\langle z, z' \rangle - 1)} = e^{-\frac{\alpha}{2}\|z - z'\|^2},$$

Basic component: dot-product kernels + Nyström

The Nyström method consists of replacing any point $\varphi(x)$ in $\mathcal{H}$, for x in $\mathcal{X}$ by its orthogonal projection onto a finite-dimensional subspace

$$\mathcal{F} = \text{span}(\varphi(z_1), \ldots, \varphi(z_p)),$$

for some anchor points $Z = [z_1, \ldots, z_p]$ in $\mathbb{R}^{d \times p}$



Basic component: dot-product kernels + Nyström

The projection is equivalent to

$$\Pi_{\mathcal{F}}[x] \triangleq \sum_{j=1}^{p} \beta_j^\star \varphi(z_j) \quad \text{with} \quad \beta^\star \in \arg\min_{\beta \in \mathbb{R}^p} \left\| \varphi(x) - \sum_{j=1}^{p} \beta_j \varphi(z_j) \right\|_{\mathcal{H}}^2,$$

Then, it is possible to show that with $K(x,y) = \langle \varphi(x), \varphi(y) \rangle_{\mathcal{H}}$,

$$K(x,y) \approx \langle \Pi_{\mathcal{F}}[x], \Pi_{\mathcal{F}}[y] \rangle_{\mathcal{H}} = \langle \psi(x), \psi(y) \rangle_{\mathbb{R}^p},$$

with

$$\psi(x) = \kappa(Z^\top Z)^{-1/2} \kappa(Z^\top x),$$

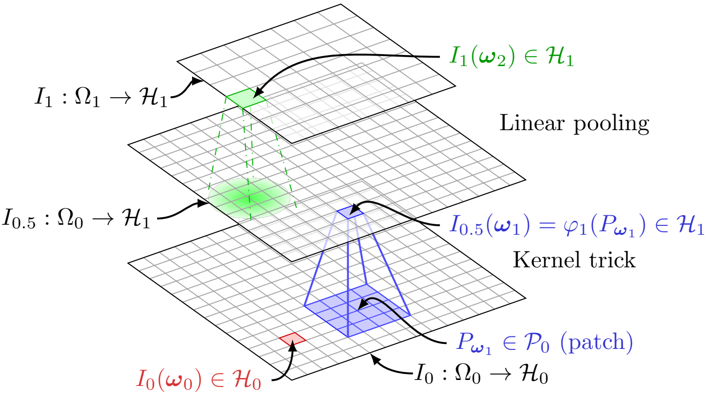
where the function κ is applied pointwise to its arguments. The resulting ψ can be interpreted as a neural network performing (i) linear operation, (ii) pointwise non-linearity, (iii) linear operation.

[Williams and Seeger, 2001, Smola and Schölkopf, 2000, Fine and Scheinberg, 2001].

The multilayer convolutional kernel

Definition: image feature maps

An image feature map is a function $I : \Omega \rightarrow \mathcal{H}$, where Ω is a 2D grid representing "coordinates" in the image and $\mathcal{H}$ is a Hilbert space.



The multilayer convolutional kernel

Definition: image feature maps

An image feature map is a function $I : \Omega \rightarrow \mathcal{H}$, where Ω is a 2D grid representing "coordinates" in the image and $\mathcal{H}$ is a Hilbert space.

Motivation and examples

- Each point $I(\omega)$ carries information about an image neighborhood, which is motivated by the **local stationarity** of natural images.
- We will construct a sequence of maps $I_0, \ldots, I_k$. Going up in the hierarchy yields **larger receptive fields** with **more invariance**.
- I_0 may simply be the input image, where $\mathcal{H}_0 = \mathbb{R}^3$ for RGB.

The multilayer convolutional kernel

Definition: image feature maps

An image feature map is a function $I : \Omega \to \mathcal{H}$, where Ω is a 2D grid representing "coordinates" in the image and $\mathcal{H}$ is a Hilbert space.

Motivation and examples

- Each point $I(\omega)$ carries information about an image neighborhood, which is motivated by the **local stationarity** of natural images.
- We will construct a sequence of maps $I_0, \ldots, I_k$. Going up in the hierarchy yields **larger receptive fields** with **more invariance**.
- I_0 may simply be the input image, where $\mathcal{H}_0 = \mathbb{R}^3$ for RGB.

How do we go from $I_0 : \Omega_0 \to \mathcal{H}_0$ to $I_1 : \Omega_1 \to \mathcal{H}_1$?

The multilayer convolutional kernel

Definition: image feature maps

An image feature map is a function $I : \Omega \to \mathcal{H}$, where Ω is a 2D grid representing "coordinates" in the image and $\mathcal{H}$ is a Hilbert space.

Motivation and examples

- Each point $I(\omega)$ carries information about an image neighborhood, which is motivated by the **local stationarity** of natural images.
- We will construct a sequence of maps $I_0, \ldots, I_k$. Going up in the hierarchy yields **larger receptive fields** with **more invariance**.
- I_0 may simply be the input image, where $\mathcal{H}_0 = \mathbb{R}^3$ for RGB.

How do we go from $I_0 : \Omega_0 \to \mathcal{H}_0$ to $I_1 : \Omega_1 \to \mathcal{H}_1$?

First, define a p.d. kernel on patches of I_0!

The multilayer convolutional kernel

Going from I_0 to $I_{0.5}$: kernel trick

- Patches of size $e_0 \times e_0$ can be defined as elements of the **Cartesian product** $\mathcal{P}_0 \triangleq \mathcal{H}_0^{e_0 \times e_0}$ endowed with its natural inner-product.
- **Define a p.d. kernel on such patches**: For all x, x' in $\mathcal{P}_0$,

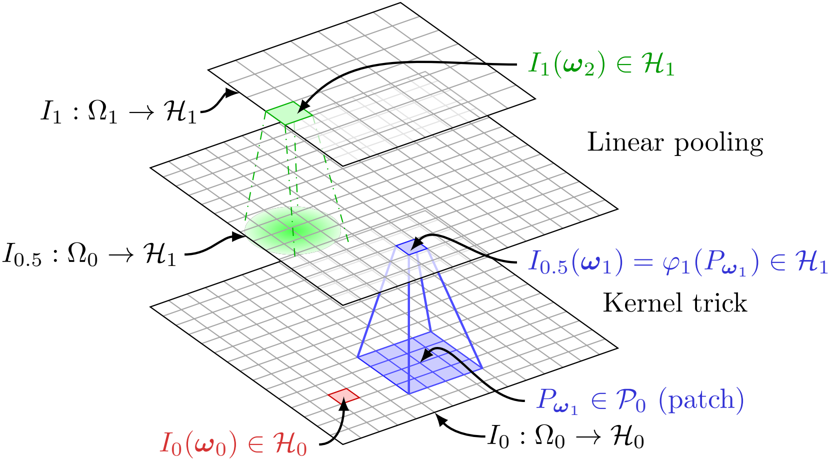
$$K_1(x,x') = \|x\|_{\mathcal{P}_0}\|x'\|_{\mathcal{P}_0}\kappa_1\left(\frac{\langle x, x'\rangle_{\mathcal{P}_0}}{\|x\|_{\mathcal{P}_0}\|x'\|_{\mathcal{P}_0}}\right) \text{ if } x, x' \neq 0 \text{ and 0 otherwise.}$$

  Note that for y, y' normalized, we may choose

$$\kappa_1\left(\langle y, y'\rangle_{\mathcal{P}_0}\right) = e^{\alpha_1\left(\langle y,y'\rangle_{\mathcal{P}_0}-1\right)} = e^{-\frac{\alpha_1}{2}\|y-y'\|^2_{\mathcal{P}_0}}.$$

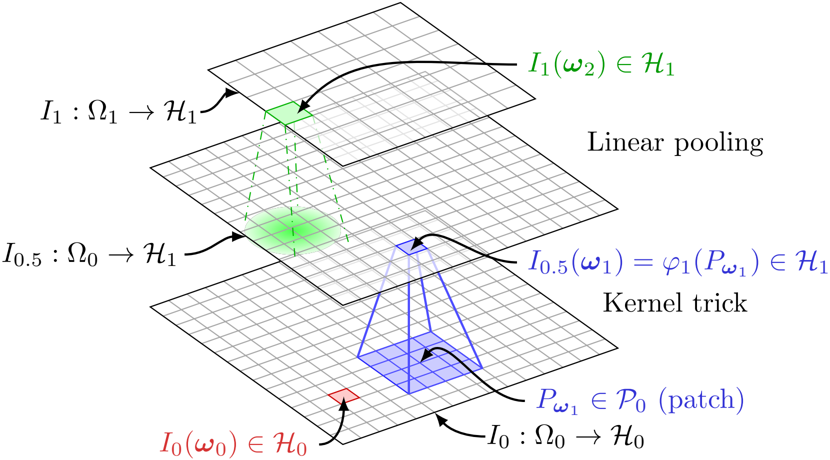
- **We call** $\mathcal{H}_1$ the RKHS and define a **mapping** $\varphi_1 : \mathcal{P}_0 \to \mathcal{H}_1$.
- Then, we may define the map $I_{0.5} : \Omega_0 \to \mathcal{H}_1$ that carries the representations in $\mathcal{H}_1$ of the patches from I_0 at all locations in Ω_0.

The multilayer convolutional kernel



How do we go from $I_{0.5} : \Omega_0 \to \mathcal{H}_1$ to $I_1 : \Omega_1 \to \mathcal{H}_1$?

The multilayer convolutional kernel



How do we go from $I_{0.5} : \Omega_0 \rightarrow \mathcal{H}_1$ to $I_1 : \Omega_1 \rightarrow \mathcal{H}_1$?

Linear pooling!

The multilayer convolutional kernel

Going from $I_{0.5}$ to I_1: linear pooling

- For all ω in Ω_1:

$$I_1(\omega) = \sum_{\omega' \in \Omega_0} I_{0.5}(\omega') e^{-\beta_1 \|\omega' - \omega\|_2^2}.$$

- The Gaussian weight can be interpreted as an anti-aliasing filter for downsampling the map $I_{0.5}$ to a different resolution.
- Linear pooling is compatible with the kernel interpretation: linear combinations of points in the RKHS are still points in the RKHS.

Finally,

- We may now repeat the process and build $I_0, I_1, \ldots, I_k$.
- and obtain the **multilayer convolutional kernel**

$$K(I_k, I'_k) = \sum_{\omega \in \Omega_k} \langle I_k(\omega), I'_k(\omega) \rangle_{\mathcal{H}_k}.$$

The multilayer convolutional kernel

In summary

- The multilayer convolutional kernel builds upon similar principles as a convolutional neural net (**multiscale, local stationarity**).
- Invariance to local translations is achieved through **linear pooling** in the RKHS.
- It remains a **conceptual object** due to its high complexity.
- **Learning and modelling are still decoupled**.

Let us first address the second point (scalability).

Unsupervised learning for convolutional kernel networks

Learn linear subspaces of finite-dimensions where we project the data

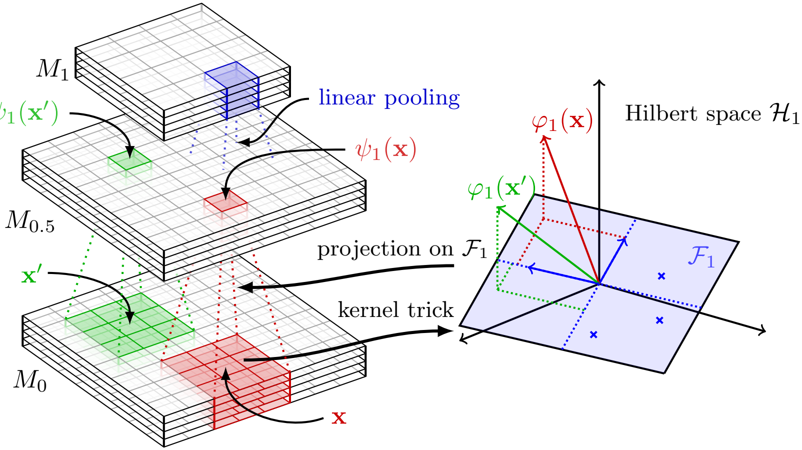


Figure: The convolutional kernel network model between layers 0 and 1.

Unsupervised learning for convolutional kernel networks

Formally, this means using the Nyström approximation

- We now manipulate **finite-dimensional maps** $M_j : \Omega_j \to \mathbb{R}^{p_j}$.
- Every linear subspace is parametrized by anchor points

$$\mathcal{F}_j \triangleq \text{Span}\left(\varphi(z_{j,1}), \ldots, \varphi(z_{j,p_j})\right),$$

where the $z_{1,j}$'s are in $\mathbb{R}^{p_{j-1}e_{j-1}^2}$ for patches of size $e_{j-1} \times e_{j-1}$.

- The encoding function at layer j is

$$\psi_j(x) \triangleq \|x\|\kappa_j(Z_j^\top Z_j)^{-1/2}\kappa_1\left(Z_j^\top \frac{x}{\|x\|}\right) \text{ if } x \neq 0 \text{ and } 0 \text{ otherwise,}$$

where $Z_j = [z_{j,1}, \ldots, z_{j,p_j}]$ and $\|.\|$ is the Euclidean norm.

- The interpretation is **convolution** with filters Z_j, **pointwise non-linearity**, 1×1 **convolution**, **contrast normalization**.

Unsupervised learning for convolutional kernel networks

- The pooling operation keeps points in the linear subspace $\mathcal{F}_j$, and pooling $M_{0.5} : \Omega_0 \rightarrow \mathbb{R}^{p_1}$ is equivalent to pooling $I_{0.5} : \Omega_0 \rightarrow \mathcal{H}_1$.

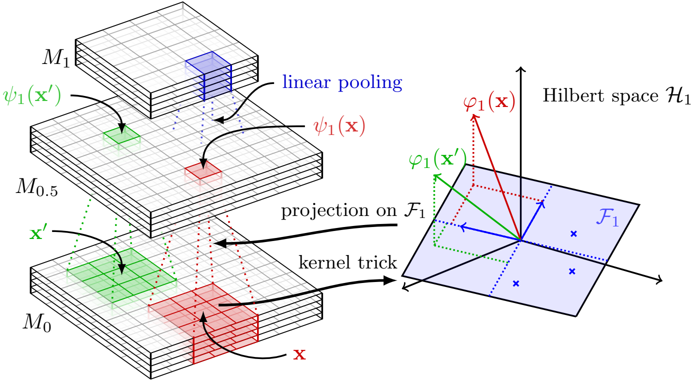


Figure: The convolutional kernel network model between layers 0 and 1.

Unsupervised learning for convolutional kernel networks

How do we learn the filters **with no** supervision?

we learn one layer at a time, starting from the bottom one.

- we **extract a large number**—say $100\,000$ patches from layers $j-1$ computed on an image database and normalize them;
- perform a **spherical K-means algorithm** to learn the filters Z_j;
- **compute the projection matrix** $\kappa_j(Z_j^\top Z_j)^{-1/2}$.

Remarks

- with kernels, we map **patches in infinite dimension**; with the projection, we **manipulate finite-dimensional objects**.
- we obtain an **unsupervised** convolutional net with a **geometric interpretation**, where we perform projections in the RKHSs.

Convolutional kernel networks with supervised learning

How do we learn the filters **with** supervision?

- Given a kernel K and RKHS $\mathcal{H}$, the ERM objective is

$$\min_{f \in \mathcal{H}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\frac{\lambda}{2}\|f\|_{\mathcal{H}}^2}_{\text{regularization}}.$$

- here, we use the parametrized kernel

$$K_{\mathcal{Z}}(I_0, I_0') = \sum_{\omega \in \Omega_k} \langle M_k(\omega), M_k'(\omega) \rangle = \langle M_k, M_k' \rangle_{\mathsf{F}},$$

- and we obtain the simple formulation

$$\min_{W \in \mathbb{R}^{p_k \times |\Omega_k|}} \frac{1}{n}\sum_{i=1}^{n} L(y_i, \langle W, M_k^i \rangle_{\mathsf{F}}) + \frac{\lambda}{2}\|W\|_{\mathsf{F}}^2. \tag{1}$$

Convolutional kernel networks with supervised learning

How do we learn the filters **with** supervision?

- we **jointly optimize** w.r.t. $\mathcal{Z}$ (set of filters) and W.
- we **alternate** between the optimization of $\mathcal{Z}$ and of W;
- for W, the problem is strongly-convex and can be tackled with recent algorithms that are much faster than SGD;
- for $\mathcal{Z}$, we derive **backpropagation rules** and use classical tricks for learning CNNs (SGD+momentum);

The only tricky part is to differentiate $\kappa_j(Z_j^\top Z_j)^{-1/2}$ w.r.t Z_j, which is a non-standard operation in classical CNNs.

Convolutional kernel networks

In summary

- a multilayer kernel for images, which builds upon similar principles as a convolutional neural net (**multiscale, local stationarity**).
- A new type of convolutional neural network with a geometric interpretation: **orthogonal projections in RKHS**.
- Learning may be unsupervised: **align subspaces with data**.
- Learning may be supervised: **subspace learning in RKHSs**.

Related work on deep kernel machines

Related work

- proof of concept for combining kernels and deep learning [Cho and Saul, 2009];
- hierarchical kernel descriptors [Bo et al., 2011];
- other multilayer models [Bouvrie et al., 2009, Montavon et al., 2011, Anselmi et al., 2015];
- deep Gaussian processes [Damianou and Lawrence, 2013].
- multilayer PCA [Schölkopf et al., 1998].
- old kernels for images [Scholkopf, 1997].
- RBF networks [Broomhead and Lowe, 1988].

Related work on deep kernel machines

Composition of feature spaces

Consider a p.d. kernel $K_1 : \mathcal{X}^2 \to \mathbb{R}$ and its RKHS $\mathcal{H}_1$ with mapping $\varphi_1 : \mathcal{X} \to \mathcal{H}_1$. Consider also a p.d. kernel $K_2 : \mathcal{H}_1^2 \to \mathbb{R}$ and its RKHS $\mathcal{H}_2$ with mapping $\varphi_2 : \mathcal{H}_1 \to \mathcal{H}_2$. Then, $K_3 : \mathcal{X}^2 \to \mathbb{R}$ below is also p.d.

$$K_3(x, x') = K_2(\varphi_1(x), \varphi_1(x')),$$

Related work on deep kernel machines

Composition of feature spaces

Consider a p.d. kernel $K_1 : \mathcal{X}^2 \to \mathbb{R}$ and its RKHS $\mathcal{H}_1$ with mapping $\varphi_1 : \mathcal{X} \to \mathcal{H}_1$. Consider also a p.d. kernel $K_2 : \mathcal{H}_1^2 \to \mathbb{R}$ and its RKHS $\mathcal{H}_2$ with mapping $\varphi_2 : \mathcal{H}_1 \to \mathcal{H}_2$. Then, $K_3 : \mathcal{X}^2 \to \mathbb{R}$ below is also p.d.

$$K_3(x, x') = K_2(\varphi_1(x), \varphi_1(x')),$$

Examples

$$K_3(x, x') = e^{-\frac{1}{2\sigma^2}\|\varphi_1(x)-\varphi_1(x')\|^2_{\mathcal{H}_1}}.$$

$$K_3(x, x') = \langle \varphi_1(x), \varphi_1(x') \rangle^2_{\mathcal{H}_1} = K_1(x, x')^2.$$

Related work on deep kernel machines

Remarks on the composition of feature spaces

- we can iterate the process many times.
- the idea appears early in the literature of kernel methods [see Schölkopf et al., 1998, for a multilayer variant of kernel PCA].

Is this idea sufficient to make kernel methods more powerful?

Related work on deep kernel machines

Remarks on the composition of feature spaces

- we can iterate the process many times.
- the idea appears early in the literature of kernel methods [see Schölkopf et al., 1998, for a multilayer variant of kernel PCA].

Is this idea sufficient to make kernel methods more powerful?

Probably not:

- K_2 is doomed to be a simple kernel (dot-product or RBF kernel).
- K_3 and K_1 operate **on the same type of object**; it is not clear why desining K_3 is easier than designing K_1 directly.

Related work on deep kernel machines

Remarks on the composition of feature spaces

- we can iterate the process many times.
- the idea appears early in the literature of kernel methods [see Schölkopf et al., 1998, for a multilayer variant of kernel PCA].

Is this idea sufficient to make kernel methods more powerful?

Probably not:

- K_2 is doomed to be a simple kernel (dot-product or RBF kernel).
- K_3 and K_1 operate **on the same type of object**; it is not clear why desining K_3 is easier than designing K_1 directly.

CKNs rely on this principle, but exploit the multi-scale and spatial structure of the signal to operate on more and more complex objects.

Related work on deep kernel machines: infinite NN

A large class of kernels on $\mathbb{R}^p$ may be defined as an expectation

$$K(x, y) = \mathbb{E}_w[s(w^\top x)s(w^\top y)],$$

where $s : \mathbb{R} \to \mathbb{R}$ is a nonlinear function. The encoding can be seen as a **one-layer neural network with infinite number of random weights**.

Related work on deep kernel machines: infinite NN

A large class of kernels on $\mathbb{R}^p$ may be defined as an expectation

$$K(x,y) = \mathbb{E}_w[s(w^\top x)s(w^\top y)],$$

where $s : \mathbb{R} \to \mathbb{R}$ is a nonlinear function. The encoding can be seen as a **one-layer neural network with infinite number of random weights**.

Examples

- random Fourier features

$$\kappa(x-y) = \mathbb{E}_{w\sim q(w), b\sim \mathcal{U}[0,2\pi]}\left[\sqrt{2}\cos(w^\top x + b)\sqrt{2}\cos(w^\top y + b)\right]$$

- Gaussian kernel

$$e^{-\frac{1}{2\sigma^2}\|x-y\|_2^2} \propto \mathbb{E}_w\left[e^{\frac{2}{\sigma^2}w^\top x} e^{\frac{2}{\sigma^2}w^\top y}\right] \quad \text{with} \quad w \sim \mathcal{N}(0, (\sigma^2/4)I).$$

Related work on deep kernel machines: infinite NN

Example, arc-cosine kernels

$$K(x,y) \propto \mathbb{E}_w \left[\max\left(w^\top x, 0\right)^\alpha \max\left(w^\top y, 0\right)^\alpha \right] \quad \text{with} \quad w \sim \mathcal{N}(0, I),$$

for x, y on the hyper-sphere $\mathbb{S}^{m-1}$. Interestingly, the non-linearity s are **typical ones from the neural network literature**.

- $s(u) = \max(0, u)$ (rectified linear units) leads to $K_1(x,y) = \sin(\theta) + (\pi - \theta)\cos(\theta)$ **with** $\theta = \cos^{-1}(x^\top y)$;
- $s(u) = \max(0, u)^2$ (squared rectified linear units) leads to $K_2(x,y) = 3\sin(\theta)\cos(\theta) + (\pi - \theta)(1 + 2\cos^2(\theta))$;

Remarks

- infinite neural nets were discovered by Neal, 1994; then revisited many times [Le Roux, 2007, Cho and Saul, 2009].
- the concept does not lead to more powerful kernel methods...

Image classification

Experiments were conducted on classical **"deep learning" datasets**, on CPUs with no model averaging and no data augmentation.

Dataset	♯ classes	im. size	n_{train}	n_{test}
CIFAR-10	10	32×32	$50\,000$	$10\,000$
SVHN	10	32×32	$604\,388$	$26\,032$

	Stoch P. [29]	MaxOut [9]	NiN [17]	DSN [15]	Gen P. [14]	SCKN (Ours)
CIFAR-10	15.13	11.68	10.41	9.69	**7.62**	10.20
SVHN	2.80	2.47	2.35	1.92	**1.69**	2.04

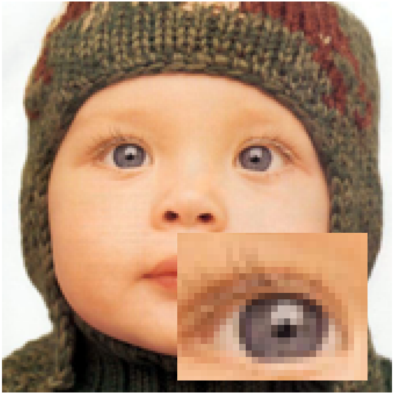
Figure: Figure from the NIPS'16 paper. Error rates in percents.

Remarks on CIFAR-10

- 10% is the standard "good" result for single model with no data augmentation.
- the best **unsupervised** architecture has two layers, is wide (1024-16384 filters), and achieves 14.2%;

Image super-resolution

The task is to predict a high-resolution y image from low-resolution one x. This may be formulated as a **multivariate regression problem**.

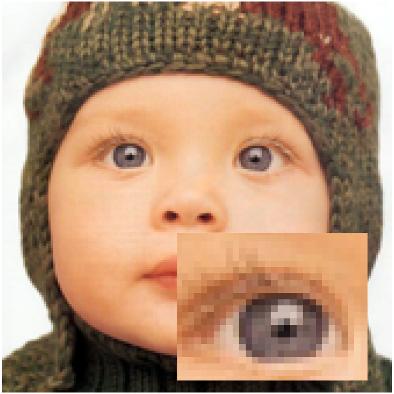


(a) Low-resolution y

(b) High-resolution x

Image super-resolution

The task is to predict a high-resolution y image from low-resolution one x. This may be formulated as a **multivariate regression problem**.



(c) Low-resolution y

(d) Bicubic interpolation

Image super-resolution

Fact.	Dataset	Bicubic	SC	CNN	CSCN	SCKN
x2	Set5	33.66	35.78	36.66	36.93	**37.07**
	Set14	30.23	31.80	32.45	32.56	**32.76**
	Kodim	30.84	32.19	32.80	32.94	**33.21**
x3	Set5	30.39	31.90	32.75	**33.10**	33.08
	Set14	27.54	28.67	29.29	29.41	**29.50**
	Kodim	28.43	29.21	29.64	29.76	**29.88**

Table: Reconstruction accuracy for super-resolution in PSNR (the higher, the better). All CNN approaches are without data augmentation at test time.

Remarks

- CNN is a "vanilla CNN" [Dong et al., 2016];
- Very recent work does better with very deep CNNs and residual learning [Kim et al., 2016];
- CSCN combines ideas from sparse coding and CNNs;

[Zeyde et al., 2010, Dong et al., 2016, Wang et al., 2015, Kim et al., 2016].

Image super-resolution

Bicubic | Sparse coding | CNN | SCKN (Ours)

Figure: Results for x3 upscaling.

Image super-resolution

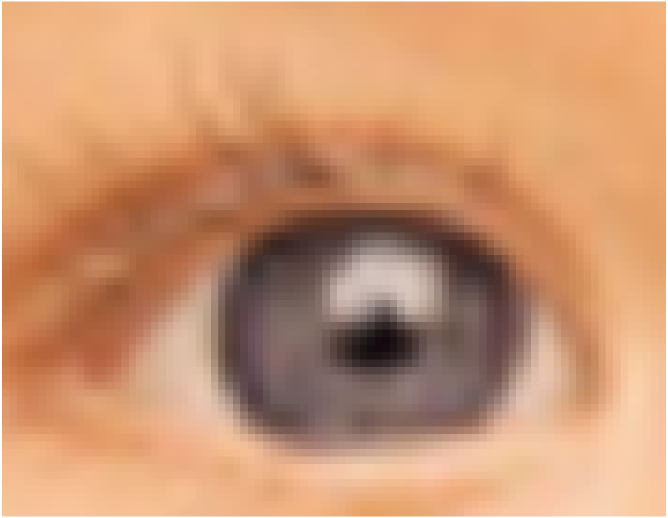


Figure: Bicubic

Image super-resolution

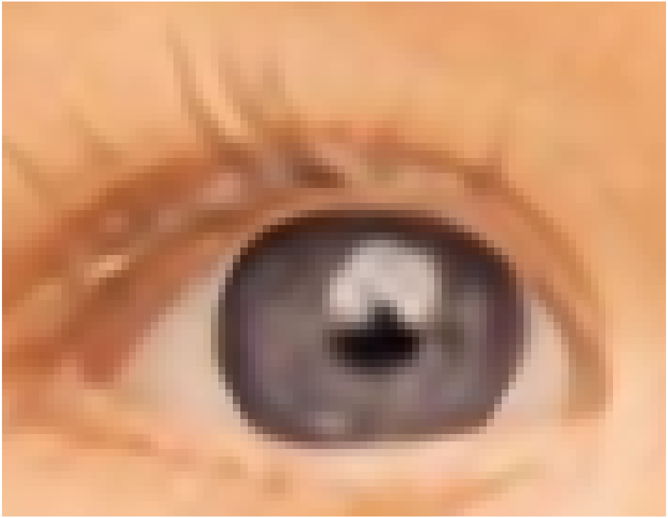
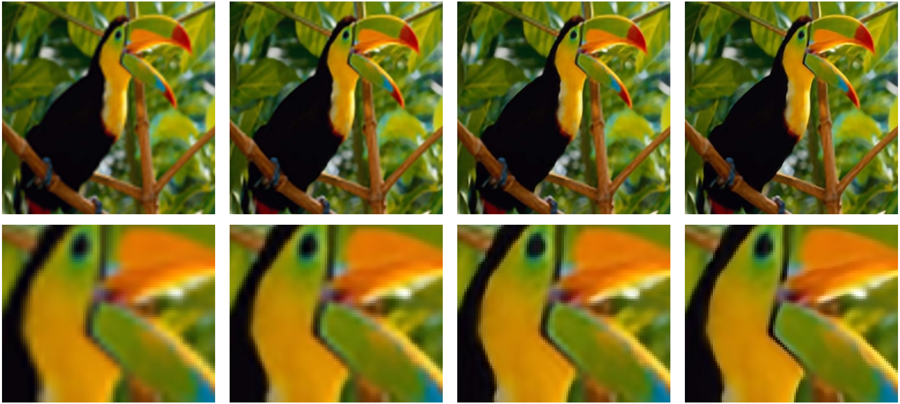


Figure: SCKN

Image super-resolution



Bicubic | Sparse coding | CNN | SCKN (Ours)

Figure: Results for x3 upscaling.

Image super-resolution

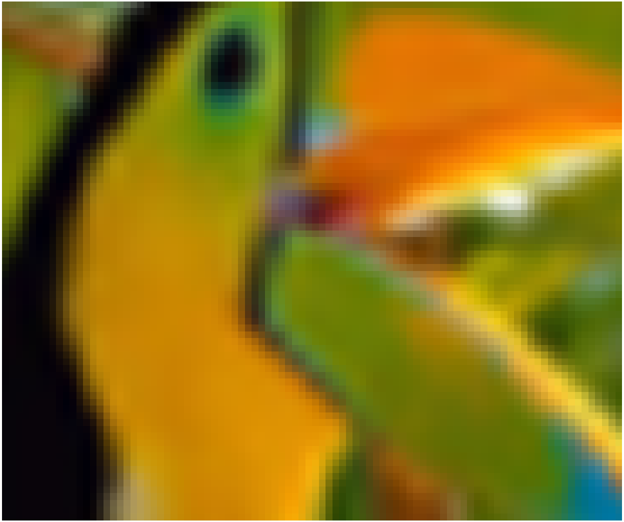
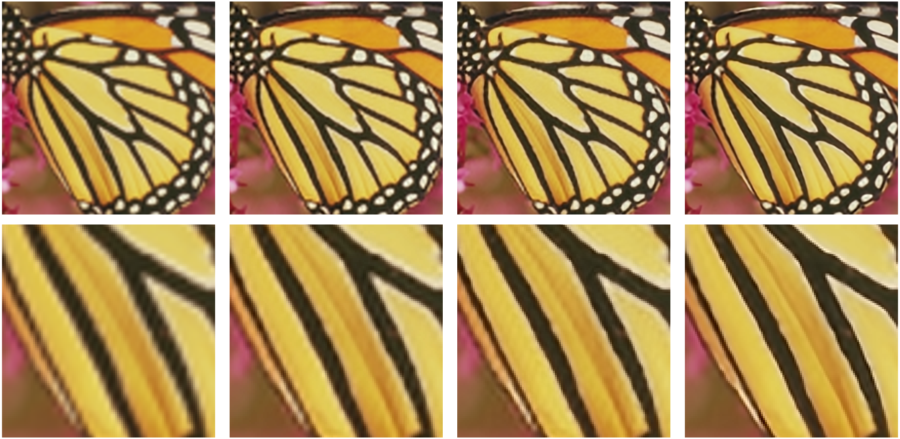


Figure: Bicubic

Image super-resolution

Figure: SCKN

Image super-resolution



Bicubic	Sparse coding	CNN	SCKN (Ours)

Figure: Results for x3 upscaling.

Image super-resolution

Figure: Bicubic

Image super-resolution

Figure: SCKN

Image super-resolution

Bicubic CNN SCKN (Ours)

Figure: Results for x3 upscaling.

Image super-resolution

Figure: Bicubic

Image super-resolution

Figure: SCKN

Part III: Invariance, Stability, and Complexity of Deep Convolutional Representations

Understanding deep convolutional representations

Questions

- Are they **stable to deformations?**
- How can we achieve **invariance to transformation groups?**
- Do they **preserve signal information?**
- How can we measure **model complexity?**

- A. Bietti and J. Mairal. Group Invariance, Stability to Deformations, and Complexity of Deep Convolutional Representations. arXiv:1706.03078. 2017.
- A. Bietti and J. Mairal. Invariance and Stability of Deep Convolutional Representations. NIPS. 2017.

Construct a functional space for deep learning

Main ideas

1. use the kernel construction of CKNs;
2. notice that the functional space contains some CNNs;
3. derive theoretical results for CKNs and CNNs.

Why? Separate learning from representation: $f(x) = \langle f, \Phi(x) \rangle$

- $\Phi(x)$: CNN **architecture** (stability, invariance, signal preservation)
- f: CNN **model**, learning, generalization through $\|f\|$

$$|f(x) - f(x')| \leq \|f\| \cdot \|\Phi(x) - \Phi(x')\|.$$

- $\|f\|$ **controls both stability and generalization**!

  $\rightarrow$ discriminating small deformations requires large $\|f\|$

  $\rightarrow$ learning stable functions is "easier"

Construct a functional space for deep learning

Which CNNs live in the RKHS of CKNs?

The RKHS construction provides a **linearization** of some CNNs:

$$f(x) = \sigma_k(\mathbf{A}_k\sigma_{k-1}(\mathbf{A}_{k-1}\ldots\sigma_2(\mathbf{A}_2\sigma_1(\mathbf{A}_1 x))\ldots)) = \langle f, \varphi(x)\rangle_{\mathcal{H}}.$$

Remember that the patch kernels are defined as

$$K_k(z, z') = \|z\|\|z'\|\kappa_k\left(\frac{\langle z, z'\rangle}{\|z\|\|z'\|}\right), \qquad \kappa_k(u) = \sum_{j=0}^{\infty} b_j u^j$$

The RKHS for patches contains activation functions σ that are

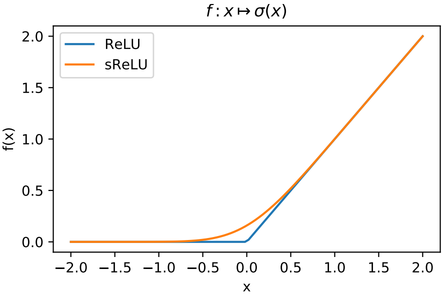
- **homogeneous**: $\sigma : z \mapsto \|z\|\tilde{\sigma}(\langle g, z\rangle/\|z\|)$
- **smooth**: $\tilde{\sigma}(u) = \sum_{j=0}^{\infty} a_j u^j$
- with norm: $\|\sigma\|_{\mathcal{H}_k}^2 \leq C_\sigma^2(\|g\|^2) = \sum_{j=0}^{\infty} \frac{a_j^2}{b_j}\|g\|^{2j} < \infty.$

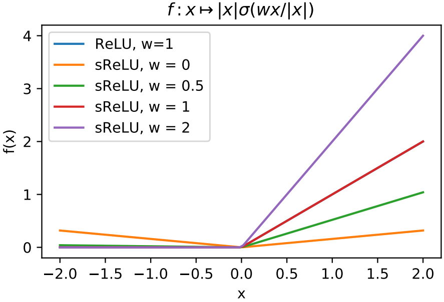
Homogeneous version of [Zhang et al., 2017]

Construct a functional space for deep learning

Examples

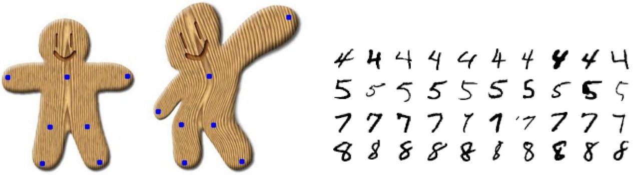
- $\sigma(u) = u$ (linear): $C_\sigma^2(\lambda^2) = O(\lambda^2)$
- $\sigma(u) = u^p$ (polynomial): $C_\sigma^2(\lambda^2) = O(\lambda^{2p})$
- $\sigma \approx$ sin, sigmoid, smooth ReLU: $C_\sigma^2(\lambda^2) = O(e^{c\lambda^2})$





Stability

- $\tau : \Omega \to \Omega$: C^1-diffeomorphism
- $L_\tau x(u) = x(u - \tau(u))$: action operator
- Much richer group of transformations than translations



- Representation $\varphi(\cdot)$ is **stable** [Mallat, 2012] if:

$$\|\varphi(L_\tau x) - \varphi(x)\| \leq (C_1 \|\nabla \tau\|_\infty + C_2 \|\tau\|_\infty)\|x\|$$

- $\|\nabla \tau\|_\infty = \sup_u \|\nabla \tau(u)\|$ controls deformation
- $\|\tau\|_\infty = \sup_u |\tau(u)|$ controls translation

Stability and signal recovery

Proposition [Bietti and Mairal, 2017]

if $\|\nabla\tau\|_\infty \leq 1/2$ and Φ_n is the representation at layer n,

$$\|\Phi_n(L_\tau x) - \Phi_n(x)\| \leq \left(C_1\,(1+n)\,\|\nabla\tau\|_\infty + \frac{C_2}{\sigma_n}\|\tau\|_\infty\right)\|x\|$$

Remarks and additional results

- The result requires the receptive field sizes at layer k to be of the same order or smaller than the pooling bandwidth of layer $k-1$.
- To **preserve information** when discretizing, we need subsampling factors that are smaller than the patch sizes at layer k.

This points to small patches, small subsampling factors, and several layers, as in recent architectures.

Group invariance

- Convolutions and pooling provides translation invariance
- We encode more general **transformation groups** in the architecture (e.g. rotations, roto-translations, rigid motion)
- Related work: [Cohen and Welling, 2016, Mallat, 2012, Sifre and Mallat, 2013, Raj et al., 2016]

Model complexity and generalization

- How do we measure **model complexity** of CKNs and CNNs?
- Can we get meaningful bounds on generalization error?
- Summary of results:
  - Some CNNs are contained in the RKHS of CKNs.
  - we may control the RKHS norm of a generic CNN
  - The choice of activation function is important.
  - Same norm also controls stability ("stable functions generalize better")
- Related work: [Zhang et al., 2017]

Spoiler: should classical CNNs be regularized with products of spectral norms [Bartlett et al., 2017]?

Conclusion and Perspectives

Stability and generalization are related through **regularization**. There are two types of perpectives for this approach:

For existing deep networks

- new regularization functions, along with algorithmic tools to learn with **less labeled data**, and obtain **more stable** models?
  $\Rightarrow$ on-going work with spectral regularization.

For designing new deep models

- design deep models that are **stable by design**?
  $\Rightarrow$ We already have models that are stable w.r.t hyper-parameter choices. Are they robust to adversarial perturbations?

Mark the date! July 2-6th, Grenoble

Along with Naver Labs, Inria is organizing a summer school in Grenoble on artificial intelligence. Visit https://project.inria.fr/paiss/.

Among the distinguished speakers

- Lourdes Agapito (UCL)
- Kyunghyun Cho (NYU/Facebook)
- Emmanuel Dupoux (EHESS)
- Martial Hebert (CMU)
- Hugo Larochelle (Google Brain)
- Yann LeCun (Facebook/NYU)
- Jean Ponce (Inria)
- Cordelia Schmid (Inria)
- Andrew Zisserman (Oxford/Google DeepMind).
- ...

References I

Fabio Anselmi, Lorenzo Rosasco, Cheston Tan, and Tomaso Poggio. Deep convolutional networks are hierarchical kernel machines. *arXiv preprint arXiv:1508.01084*, 2015.

Peter Bartlett, Dylan J Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1706.08498*, 2017.

Alberto Bietti and Julien Mairal. Group invariance and stability to deformations of deep convolutional representations. *arXiv preprint arXiv:1706.03078*, 2017.

L. Bo, K. Lai, X. Ren, and D. Fox. Object recognition with hierarchical kernel descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, 2016.

J. V. Bouvrie, L. Rosasco, and T. Poggio. On invariance in hierarchical models. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

References II

David S Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document, 1988.

Y. Cho and L. K. Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning (ICML)*, 2016.

A. Damianou and N. Lawrence. Deep Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2013.

C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 38(2):295–307, 2016.

Shai Fine and Katya Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research (JMLR)*, 2:243–264, 2001.

References III

Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

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

J. Mairal. End-to-end kernel learning with supervised convolutional kernel networks. In *Adv. NIPS*, 2016.

Stéphane Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

GrÃŠgoire Montavon, Mikio L Braun, and Klaus-Robert MÃÅ§ller. Kernel analysis of deep networks. *Journal of Machine Learning Research*, 12(Sep): 2563–2581, 2011.

References IV

K-R Müller, Sebastian Mika, Gunnar Ratsch, Koji Tsuda, and Bernhard Scholkopf. An introduction to kernel-based learning algorithms. *IEEE transactions on neural networks*, 12(2):181–201, 2001.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. 2017.

Anant Raj, Abhishek Kumar, Youssef Mroueh, P Thomas Fletcher, and Bernhard Scholkopf. Local group invariant representations via orbit embeddings. *preprint arXiv:1612.01988*, 2016.

I. Schoenberg. Positive definite functions on spheres. *Duke Math. J.*, 1942.

B. Scholkopf. *Support Vector Learning*. PhD thesis, Technischen Universität Berlin, 1997.

Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10 (5):1299–1319, 1998.

References V

John Shawe-Taylor and Nello Cristianini. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2004.

Laurent Sifre and Stéphane Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2013.

Alex J Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2000.

Alex J Smola, Zoltan L Ovari, and Robert C Williamson. Regularization with dot-product kernels. In *Advances in neural information processing systems*, pages 308–314, 2001.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1995.

Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

References VI

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, pages 711–730. 2010.

Y. Zhang, P. Liang, and M. J. Wainwright. Convexified convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2017.