Mathematical Foundations of Data Sciences

Gabriel Peyré
CNRS & DMA
École Normale Supérieure
gabriel.peyre@ens.fr
https://mathematical-tours.github.io
www.numerical-tours.com

November 15, 2020

Chapter 15

Deep Learning

Before detailing deep architectures and their use, we start this chapter by presenting two essential computational tools that are used to train these models: stochastic optimization methods and automatic differentiation. In practice, they work hand-in-hand to be able to learn painlessly complicated non-linear models on large-scale datasets.

15.1 Multi-Layers Perceptron

In this section, we study the simplest example of non-linear parametric models, namely Multi-Layers Perceptron (MLP) with a single hidden layer (so they have in total 2 layers). Perceptron (with no hidden layer) corresponds to the linear models studied in the previous sections. MLP with more layers are obtained by stacking together several such simple MLP, and are studied in Section **??**, since the computation of their derivatives is very suited to automatic-differentiation methods.

15.1.1 MLP and its derivative

The basic MLP $a \mapsto h_{W,u}(a)$ takes as input a feature vector $a \in \mathbb{R}^p$, computes an intermediate hidden representation $b = Wa \in \mathbb{R}^q$ using q "neurons" stored as the rows $w_k \in \mathbb{R}^p$ of the weight matrix $W \in \mathbb{R}^{q \times p}$, passes these through a non-linearity $\rho : \mathbb{R} \to \mathbb{R}$, i.e. $\rho(b) = (\rho(b_k))_{k=1}^q$ and then outputs a scalar value as a linear combination with output weights $u \in \mathbb{R}^q$, i.e.

$$h_{W,u}(a) = \langle \rho(Wa),\, u \rangle = \sum_{k=1}^{q} u_k \rho((Wa)_k) = \sum_{k=1}^{q} u_k \rho(\langle a,\, w_k \rangle).$$

This function $h_{W,u}(\cdot)$ is thus a weighted sum of q "ridge functions" $\rho(\langle \cdot,\, w_k \rangle)$. These functions are constant in the direction orthogonal to the neuron w_k and have a profile defined by ρ.

The most popular non-linearities are sigmoid functions such as

$$\rho(r) = \frac{e^r}{1+e^r} \quad \text{and} \quad \rho(r) = \frac{1}{\pi}\mathrm{atan}(r) + \frac{1}{2}$$

and the rectified linear unit (ReLu) function $\rho(r) = \max(r, 0)$.

One often add a bias term in these models, and consider functions of the form $\rho(\langle \cdot,\, w_k \rangle + z_k)$ but this bias term can be integrated in the weight as usual by considering $(\langle a,\, w_k \rangle + z_k = \langle (a, 1),\, (w_k, z_k) \rangle$, so we ignore it in the following section. This simply amount to replacing $a \in \mathbb{R}^p$ by $(a, 1) \in \mathbb{R}^{p+1}$ and adding a dimension $p \mapsto p + 1$, as a pre-processing of the features.

Expressiveness. In order to define function of arbitrary complexity when q increases, it is important that ρ is non-linear. Indeed, if $\rho(s) = s$, then $h_{W,u}(a) = \langle Wa,\, u\rangle = \langle a,\, W^\top u\rangle$. It is thus a linear function with weights $W^\top u$, whatever the number q of neurons. Similarly, if ρ is a polynomial on $\mathbb{R}$ of degree d, then $h_{W,u}(\cdot)$ is itself a polynomial of degree d in $\mathbb{R}^p$, which is a linear space V of finite dimension $\dim(V) = O(p^d)$. So even if q increases, the dimension $\dim(V)$ stays fixed and $h_{W,u}(\cdot)$ cannot approximate an arbitrary function outside V. In sharp contrast, one can show that if ρ is not polynomial, then $h_{W,u}(\cdot)$ can approximate any continuous function, as studied in Section 15.1.3.

15.1.2 MLP and Gradient Computation

Given pairs of features and data values $(a_i, y_i)_{i=1}^n$, and as usual storing the features in the rows of $A \in \mathbb{R}^{n\times p}$, we consider the following least square regression function (similar computation can be done for classification losses)

$$\min_{x=(W,u)} f(W,u) \overset{\text{def.}}{=} \frac{1}{2}\sum_{i=1}^n (h_{W,u}(a_i) - y_i)^2 = \frac{1}{2}\|\rho(AW^\top)u - y\|^2.$$

Note that here, the parameters being optimized are $(W,u) \in \mathbb{R}^{q\times p} \times \mathbb{R}^q$.

Optimizing with respect to u. This function f is convex with respect to u, since it is a quadratic function. Its gradient with respect to u can be computed as in (13.8) and thus

$$\nabla_u f(W,u) = \rho(AW^\top)^\top(\rho(AW^\top)u - y)$$

and one can compute in closed form the solution (assuming $\ker(\rho(AW^\top)) = \{0\}$) as

$$u^\star = [\rho(AW^\top)^\top \rho(AW^\top)]^{-1}\rho(AW^\top)^\top y = [\rho(WA^\top)\rho(AW^\top)]^{-1}\rho(WA^\top)y$$

When $W = \mathrm{Id}_p$ and $\rho(s) = s$ one recovers the least square formula (13.9).

Optimizing with respect to W. The function f is non-convex with respect to W because the function ρ is itself non-linear. Training a MLP is thus a delicate process, and one can only hope to obtain a local minimum of f. It is also important to initialize correctly the neurons $(w_k)_k$ (for instance as unit norm random vector, but bias terms might need some adjustment), while u can be usually initialized at 0.

To compute its gradient with respect to W, we first note that for a perturbation $\varepsilon \in \mathbb{R}^{q\times p}$, one has

$$\rho(A(W+\varepsilon)^\top) = \rho(AW^\top + A\varepsilon^\top) = \rho(AW^\top) + \rho'(AW^\top)\odot(A\varepsilon^\top)$$

where we have denoted "$\odot$" the entry-wise multiplication of matrices, i.e. $U \odot V = (U_{i,j}V_{i,j})_{i,j}$. One thus has,

$$\begin{aligned} f(W+\varepsilon,u) &= \frac{1}{2}\|e + [\rho'(AW^\top)\odot(A\varepsilon^\top)]y\|^2 \quad \text{where} \quad e \overset{\text{def.}}{=} \rho(AW^\top)u - y \in \mathbb{R}^n \\ &= f(W,u) + \langle e,\, [\rho'(AW^\top)\odot(A\varepsilon^\top)]y\rangle + o(\|\varepsilon\|) \\ &= f(W,u) + \langle A\varepsilon^\top,\, \rho'(AW^\top)\odot(eu^\top)\rangle \\ &= f(W,u) + \langle \varepsilon^\top,\, A^\top \times [\rho'(AW^\top)\odot(eu^\top)]\rangle. \end{aligned}$$

The gradient thus reads

$$\nabla_W f(W,u) = [\rho'(WA^\top)\odot(ue^\top)] \times A \in \mathbb{R}^{q\times p}.$$

15.1.3 Universality

In this section, to ease the exposition, we explicitly introduce the bias and use the variable "$x \in \mathbb{R}^p$" in place of "$a \in \mathbb{R}^p$". We thus write the function computed by the MLP (including explicitly the bias z_k) as

$$h_{W,z,u}(x) \overset{\text{def.}}{=} \sum_{k=1}^{q} u_k \varphi_{w_k,z_k}(x) \quad \text{where} \quad \varphi_{w,z}(x) \overset{\text{def.}}{=} \rho(\langle x,\, w\rangle + z).$$

The function $\varphi_{w,z}(x)$ is a ridge function in the direction orthogonal to $\bar{w} \overset{\text{def.}}{=} w/\|w\|$ and passing around the point $-\frac{z}{\|w\|}\bar{w}$.

In the following we assume that $\rho : \mathbb{R} \to \mathbb{R}$ is a bounded function such that

$$\rho(r) \overset{r\to-\infty}{\longrightarrow} 0 \quad \text{and} \quad \rho(r) \overset{r\to+\infty}{\longrightarrow} 1. \tag{15.1}$$

Note in particular that such a function cannot be a polynomial and that the ReLu function does not satisfy these hypothesis (universality for the ReLu is more involved to show). The goal is to show the following theorem.

Theorem 25 (Cybenko, 1989)**.** *For any compact set $\Omega \subset \mathbb{R}^p$, the space spanned by the functions $\{\varphi_{w,z}\}_{w,z}$ is dense in $\mathcal{C}(\Omega)$ for the uniform convergence. This means that for any continuous function f and any $\varepsilon > 0$, there exists $q \in \mathbb{N}$ and weights $(w_k, z_k, u_k)_{k=1}^q$ such that*

$$\forall\, x \in \Omega, \quad |f(x) - \sum_{k=1}^{q} u_k \varphi_{w_k,z_k}(x)| \leqslant \varepsilon.$$

In a typical ML scenario, this implies that one can "overfit" the data, since using a q large enough ensures that the training error can be made arbitrary small. Of course, there is a bias-variance tradeoff, and q needs to be cross-validated to account for the finite number n of data, and ensure a good generalization properties.

Proof in dimension $p = 1$. In 1D, the approximation $h_{W,z,u}$ can be thought as an approximation using smoothed step functions. Indeed, introducing a parameter $\varepsilon > 0$, one has (assuming the function is Lipschitz to ensure uniform convergence),

$$\varphi_{\frac{w}{\varepsilon},\frac{z_k}{\varepsilon}} \overset{\varepsilon\to0}{\longrightarrow} 1_{[-z/w,+\infty[}$$

This means that

$$h_{\frac{W}{\varepsilon},\frac{z}{\varepsilon},u} \overset{\varepsilon\to0}{\longrightarrow} \sum_k u_k 1_{[-z_k/w_k,+\infty[},$$

which is a piecewise constant function. Inversely, any piecewise constant function can be written this way. Indeed, if h assumes the value d_k on each interval $[t_k, t_{k+1}[$, then it can be written as

$$h = \sum_k d_k (1_{[t_k,+\infty[} - 1_{[t_k,+\infty[}).$$

Since the space of piecewise constant functions is dense in continuous function over an interval, this proves the theorem.

Proof in arbitrary dimension p. We start by proving the following dual characterization of density, using bounded Borel measure $\mu \in \mathcal{M}(\Omega)$ i.e. such that $\mu(\Omega) < +\infty$.

Proposition 48. *If ρ is such that for any Borel measure $\mu \in \mathcal{M}(\Omega)$*

$$\left(\forall\,(w,z), \int \rho(\langle x,\, w\rangle + z)\mathrm{d}\mu(x) = 0\right) \quad \Longrightarrow \quad \mu = 0, \tag{15.2}$$

then Theorem 25 holds.

Proof. We consider the linear space

$$\mathcal{S} \stackrel{\text{def.}}{=} \left\{ \sum_{k=1}^{q} u_k \varphi_{w_k,z_k} \;;\; q \in \mathbb{N}, w_k \in \mathbb{R}^p, u_k \in \mathbb{R}, z_k \in \mathbb{R} \right\} \subset \mathcal{C}(\Omega).$$

Let $\bar{\mathcal{S}}$ be its closure in $\mathcal{C}(\Omega)$ for $\|\cdot\|_\infty$, which is a Banach space. If $\bar{\mathcal{S}} \neq \mathcal{C}(\Omega)$, let us pick $g \neq 0$, $g \in \mathcal{C}(\Omega)\backslash\bar{\mathcal{S}}$. We define the linear form L on $\bar{\mathcal{S}} \oplus \text{span}(g)$ as

$$\forall\, s \in \bar{\mathcal{S}},\, \forall\, \lambda \in \mathbb{R}, \qquad L(s + \lambda g) = \lambda$$

so that $L = 0$ on $\bar{\mathcal{S}}$. L is a bounded linear form, so that by Hahn-Banach theorem, it can be extended in a bounded linear form $\bar{L} : \mathcal{C}(\Omega) \to \mathbb{R}$. Since $L \in \mathcal{C}(\Omega)^*$ (the dual space of continuous linear form), and that this dual space is identified with Borel measures, there exists $\mu \in \mathcal{M}(\Omega)$, with $\mu \neq 0$, such that for any continuous function h, $\bar{L}(h) = \int_\Omega h(x)\mathrm{d}\mu(x)$. But since $\bar{L} = 0$ on $\bar{\mathcal{S}}$, $\int \rho(\langle \cdot, w\rangle + z)\mathrm{d}\mu = 0$ for all (w, z) and thus by hypothesis, $\mu = 0$, which is a contradiction. □

The theorem now follows from the following proposition.

Proposition 49. *If ρ is continuous and satisfies (15.1), then it satisfies (15.2).*

Proof. One has

$$\varphi_{\frac{w}{\varepsilon},\frac{u}{\varepsilon}+t}(x) = \rho\left(\frac{\langle x,\, w\rangle + u}{\varepsilon} + t\right) \stackrel{\varepsilon\to 0}{\longrightarrow} \gamma(x) \stackrel{\text{def.}}{=} \begin{cases} 1 & \text{if} \quad H_{w,u}, \\ \rho(t) & \text{if} \quad x \in P_{w,u}, \\ 0 & \text{if} \quad \langle w,\, x\rangle + u < 0, \end{cases}$$

where we defined $H_{w,u} \stackrel{\text{def.}}{=} \{x \;;\; \langle w,\, x\rangle + u > 0\}$ and $P_{w,u} \stackrel{\text{def.}}{=} \{x \;;\; \langle w,\, x\rangle + u = 0\}$. By Lebesgue dominated convergence (since the involved quantities are bounded uniformly on a compact set)

$$\int \varphi_{\frac{w}{\varepsilon},\frac{u}{\varepsilon}+t}\mathrm{d}\mu \stackrel{\varepsilon\to 0}{\longrightarrow} \int \gamma \mathrm{d}\mu = \varphi(t)\mu(P_{w,u}) + \mu(H_{w,u}).$$

Thus if μ is such that all these integrals vanish, then

$$\forall\,(w, u, t), \quad \varphi(t)\mu(P_{w,u}) + \mu(H_{w,u}) = 0.$$

By selecting (t, t') such that $\varphi(t) \neq \varphi(t')$, one has that

$$\forall\,(w, u), \quad \mu(P_{w,u}) = \mu(H_{w,u}) = 0.$$

We now need to show that $\mu = 0$. For a fixed $w \in \mathbb{R}^p$, we consider the function

$$h \in L^\infty(\mathbb{R}), \quad F(h) \stackrel{\text{def.}}{=} \int_\Omega h(\langle w,\, x\rangle)\mathrm{d}\mu(x).$$

$F : L^\infty(\mathbb{R}) \to \mathbb{R}$ is a bounded linear form since $|F(\mu)| \leqslant \|h\|_\infty \mu(\Omega)$ and $\mu(\Omega) < +\infty$. One has

$$F(1_{[-u,+\infty[} = \int_\Omega 1_{[-u,+\infty[}(\langle w,\, x\rangle)\mathrm{d}\mu(x) = \mu(P_{w,u}) + \mu(H_{w,u}) = 0.$$

By linearity, $F(h) = 0$ for all piecewise constant functions, and F is a continuous linear form, so that by density $F(h) = 0$ for all functions $h \in L^\infty(\mathbb{R})$. Applying this for $h(r) = e^{\mathrm{i}r}$ one obtains

$$\hat{\mu}(w) \stackrel{\text{def.}}{=} \int_\Omega e^{\mathrm{i}\langle x,\, w\rangle}\mathrm{d}\mu(x) = 0.$$

This means that the Fourier transform of μ is zero, so that $\mu = 0$. □

Quantitative rates. Note that Theorem 25 is not constructive in the sense that it does not explain how to compute the weights $(w_k, u_k, z_k)_k$ to reach a desired accuracy. Since for a fixed q the function is non-convex, this is not surprising. Some recent studies show that if q is large enough, a simple gradient descent is able to reach an arbitrary good accuracy, but it might require a very large q.

Theorem 25 is also not quantitative since it does not tell how much neurons q is needed to reach a desired accuracy. To obtain quantitative bounds, continuity is not enough, it requires to add smoothness constraints. For instance, Barron proved that if

$$\int \|\omega\| |\hat{f}(\omega)| \mathrm{d}\omega \leqslant C_f$$

where $\hat{f}(\omega) = \int f(x) e^{-\mathrm{i}\langle x, \omega \rangle} \mathrm{d}x$ is the Fourier transform of f, then for $q \in \mathbb{N}$ there exists $(w_k, u_k, z_k)_k$

$$\frac{1}{\mathrm{Vol}(B(0,r))} \int_{\|x\| \leqslant r} |f(x) - \sum_{k=1}^{q} u_k \varphi_{w_k, z_k}(x)|^2 \mathrm{d}x \leqslant \frac{(2rC_f)^2}{q}.$$

The surprising part of this Theorem is that the $1/q$ decay is independent of the dimension p. Note however that the constant involved C_f might depend on p.

15.2 Deep Discriminative Models

15.2.1 Deep Network Structure

Deep learning are estimator $f(x, \beta)$ which are built as composition of simple building blocks. In their simplest form (non-recursive), they corresponds to a simple linear computational graph as already defined in (14.20) (without the loss $\mathcal{L}$), and we write this as

$$f(\cdot, \beta) = f_{L-1}(\cdot, \beta_1) \circ f_{L-2}(\cdot, \beta_2) \circ \ldots \circ f_0(\cdot, \beta_0)$$

where $\beta = (\beta_0, \ldots, \beta_{L-1})$ is the set of parameters, and

$$f_\ell(\cdot, \beta_\ell) : \mathbb{R}^{n_\ell} \to \mathbb{R}^{n_{\ell+1}}$$

While it is possible to consider more complicated architecture (in particular recurrent ones), we restrict here out attention to these simple linear graph computation structures (so-called feedforward networks).

The supervised learning of these parameters β is usually done by empirical risk minimization (12.11) using SGD-type methods as explained in Section 14.2. Note that this results in highly non-convex optimization problems. In particular, strong convergence guarantees such as Theorem 24 do not hold anymore, and only weak convergence (toward stationary points) holds. SGD type technics are however found to work surprisingly well in practice, and it now believe that the success of these deep-architecture approaches (in particular the ability of these over-parameterized model to generalize well) are in large part due to the dynamics of the SGD itself, which induce an implicit regularization effect.

For these simple linear architectures, the gradient of the ERM loss (14.13) can be computed using the reverse mode computation detailed in Section **??**. In particular, in the context of deep learning, formula (15.4). One should however keep in mind that for more complicated (e.g. recursive) architectures, such a simple formula is not anymore available, and one should resort to reverse mode automatic differentiation (see Section **??**), which, while being conceptually simple, is actually implementing possibly highly non-trivial and computationally optimal recursive differentiation.

In most successful applications of deep-learning, each computational block $f_\ell(\cdot, \beta_\ell)$ is actually very simple, and is the composition of

- an affine map, $B_\ell \cdot + b_\ell$ with a matrix $B_\ell \in \mathbb{R}^{n_\ell \times \tilde{n}_\ell}$ and a vector $b_\ell \in \mathbb{R}^{\tilde{n}_\ell}$ parametrized (in most case linearly) by β_ℓ,
- a fixed (not depending on β_ℓ) non-linearity $\rho_\ell : \mathbb{R}^{\tilde{n}_\ell} \to \mathbb{R}^{n_{\ell+1}}$

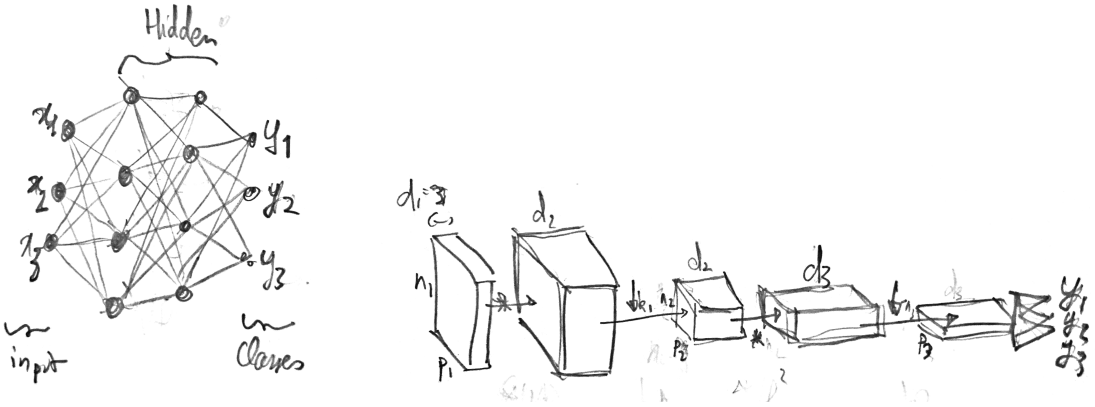

Figure 15.1: Left: example of fully connected network. Right: example of convolutional neural network.

which we write as

$$\forall\, x_\ell \in \mathbb{R}^{n_\ell}, \quad f_\ell(x_\ell, \beta_\ell) = \rho_\ell(B_\ell x_\ell + b_\ell) \in \mathbb{R}^{n_{\ell+1}}. \tag{15.3}$$

In the simplest case, the so-called "fully connected", one has $(B_\ell, b_\ell) = \beta_\ell$, i.e. B_ℓ is a full matrix and its entries (together with the bias b_ℓ) are equal to the set of parameters β_ℓ. Also in the simplest cases ρ_ℓ is a pointwise non-linearity $\rho_\ell(z) = (\tilde{\rho}_\ell(z_k))_k$, where $\tilde{\rho}_\ell : \mathbb{R} \to \mathbb{R}$ is non-linear. The most usual choices are the rectified linear unit (ReLu) $\tilde{\rho}_\ell(s) = \max(s, 0)$ and the sigmoid $\tilde{\rho}_\ell(s) = \theta(s) = (1 + e^{-s})^{-1}$.

The important point here is that the interleaving of non-linear map progressively increases the complexity of the function $f(\cdot, \beta)$.

The parameter $\beta = (B_\ell, b_\ell)_\ell$ of such a deep network are then trained by minimizing the ERM functional (12.11) using SGD-type stochastic optimization method. The gradient can be computed efficiently (with complexity proportional to the application of the model, i.e. $O(\sum_\ell n_\ell^2)$) by automatic differentiation. Since such models are purely feedforward, one can directly use the back-propagation formula (14.20).

For regression tasks, one can can directly use the output of the last layer (using e.g. a ReLu non-linearity) in conjunction with a ℓ^2 squared loss L. For classification tasks, the output of the last layer needs to be transformed into class probabilities by a multi-class logistic map (**??**).

An issue with such a fully connected setting is that the number of parameters is too large to be applicable to large scale data such as images. Furthermore, it ignores any prior knowledge about the data, such as for instance some invariance. This is addressed in more structured architectures, such as for instance convolutional networks detailed in Section 15.2.3.

15.2.2 Perceptron and Shallow Models

Before going on with the description of deep architectures, let us re-interpret the logistic classification method detailed in Sections 12.4.2 and 12.4.3.

The two-class logistic classification model (12.21) is equal to a single layer ($L = 1$) network of the form (15.3) (ignoring the constant bias term) where

$$B_0 x = \langle x, \beta \rangle \quad \text{and} \quad \tilde{\lambda}_0(u) = \theta(u).$$

The resulting one-layer network $f(x, \beta) = \theta(\langle x, \beta \rangle)$ (possibly including a bias term by adding one dummy dimension to x) is trained using the loss, for binary classes $y \in \{0, 1\}$

$$L(t, y) = -\log(t^y (1-t)^{1-y}) = -y \log(t) - (1-y) \log(1-t).$$

In this case, the ERM optimization is of course a convex program.

Multi-class models with K classes are obtained by computing $B_0 x = (\langle x, \beta_k \rangle)_{k=1}^K$, and a normalized logistic map

$$f(x, \beta) = \mathcal{N}((\exp(\langle x, \beta_k \rangle))_k) \quad \text{where} \quad \mathcal{N}(u) = \frac{u}{\sum_k u_k}$$

and assuming the classes are represented using vectors y on the probability simplex, one should use as loss

$$L(t, y) = -\sum_{k=1}^K y_k \log(t_k).$$

15.2.3 Convolutional Neural Networks

In order to be able to tackle data of large size, and also to improve the performances, it is important to leverage some prior knowledge about the structure of the typical data to process. For instance, for signal, images or videos, it is important to make use of the spacial location of the pixels and the translation invariance (up to boundary handling issues) of the domain.

Convolutional neural networks are obtained by considering that the manipulated vectors $x_\ell \in \mathbb{R}^{n_\ell}$ at depth ℓ in the network are of the form $x_\ell \in \mathbb{R}^{\bar{n}_\ell \times d_\ell}$, where $\bar{n}_\ell$ is the number of "spatial" positions (typically along a 1-D, 2-D, or 3-D grid) and d_ℓ is the number of "channels". For instance, for color images, one starts with $\tilde{n}_\ell$ being the number of pixels, and $d_\ell = 3$.

The linear operator $B_\ell : \mathbb{R}^{\bar{n}_\ell \times d_\ell} \to \mathbb{R}^{\bar{n}_\ell \times d_{\ell+1}}$ is then (up to boundary artefact) translation invariant and hence a convolution along each channel (note that the number of channels can change between layers). It is thus parameterized by a set of filters $(\psi_{\ell,r,s})_{s=1,\dots,d_\ell}^{r=1,\dots,d_{\ell+1}}$. Denoting $x_\ell = (x_{\ell,s,\cdot})_{s=1}^{d_\ell}$ the different layers composing x_ℓ, the linear map reads

$$\forall\, r \in \{1, \dots, d_{\ell+1}\}, \quad (B_\ell x_\ell)_{r,\cdot} = \sum_{s=1}^{d_\ell} \psi_{\ell,r,s} \star x_{\ell,s,\cdot}$$

and the bias term $b_\ell \in \mathbb{R}$ is contant (to maintain translation invariance).

The non-linear maps across layers serve two purposes: as before a pointwise non-linearity is applied, and then a sub-sampling helps to reduce the computational complexity of the network. This is very similar to the construction of the fast wavelet transform. Denoting by m_k the amount of down-sampling, where usually $m_k = 1$ (no reduction) or $m_k = 2$ (reduction by a factor two in each direction). One has

$$\lambda_\ell(u) = \left(\tilde{\lambda}_\ell(u_{s,m_k \cdot})\right)_{s=1\dots,d_{\ell+1}}.$$

In the literature, it has been proposed to replace linear sub-sampling by non-linear sub-sampling, for instance the so-called max-pooling (that operate by taking the maximum among groups of m_ℓ successive values), but it seems that linear sub-sampling is sufficient in practice when used in conjunction with very deep (large L) architectures.

The intuition behind such model is that as one moves deeper through the layers, the neurons are receptive to larger areas in the image domain (although, since the transform is non-linear, precisely giving sense to this statement and defining a proper "receptive field" is non-trivial). Using an increasing number of channels helps to define different classes of "detectors" (for the first layer, they detect simple patterns such as edges and corner, and progressively capture more elaborated shapes).

In practice, the last few layers (2 or 3) of such a CNN architectures are chosen to be fully connected. This is possible because, thanks to the sub-sampling, the dimension of these layers are small.

The parameters of such a model are the filters $\beta = (\psi_{\ell,r,s})_{\ell,s,r}$, and they are trained by minimizing the ERM functional (12.11). The gradient is typically computed by backpropagation. Indeed, when computing the gradient with respect to some filter $\psi_{\ell,r,s}$, the feedforward computational graph has the form (14.20). For simplicity, we re-formulate this computation in the case of a single channel per layer (multiple layer can

be understood as replacing convolution by matrix-domain convolution). The forward pass computes all the inner coefficients, by traversing the network from $\ell = 0$ to $\ell = L-1$,

$$x_{\ell+1} = \lambda_\ell(\psi_\ell \star x_\ell)$$

where $\lambda_\ell(u) = (\tilde{\lambda}_\ell(u_i))_i$ is applied component wise. Then, denoting $\mathcal{E}(\beta) = \mathcal{L}(\beta, y)$ the loss to be minimized with respect to the set of filters $\beta = (\psi_\ell)_\ell$, and denoting $\nabla_\ell \mathcal{E}(\beta) = \frac{\partial \mathcal{E}(\beta)}{\partial \psi_\ell}$ the gradient with respect to ψ_ℓ, one computes all the gradients by traversing the network in reverse order, from $\ell = L-1$ to $\ell = 0$

$$\nabla_\ell \mathcal{E}(\beta) = [\lambda'_\ell(\psi_\ell \star x_\ell)] \odot [\bar{\psi}_\ell \star \nabla_{\ell+1} \mathcal{E}(\beta)], \tag{15.4}$$

where $\lambda'_\ell(u) = (\tilde{\lambda}'_\ell(u_i))_i$ applies the derivative of $\tilde{\lambda}_\ell$ component wise, and where $\bar{\psi}_\ell = \psi_\ell(-\cdot)$ is the reversed filter. Here, $\odot$ is the pointwise multiplication of vectors. The recursion is initialized as $\nabla \mathcal{E}_L(\beta) = \nabla \mathcal{L}(x_L, y)$, the gradient of the loss itself.

This recursion (15.4) is the celebrated backpropagation algorithm put forward by Yann Lecun. Note that to understand and code these iterations, one does not need to rely on the advanced machinery of reverse mode automatic differentiation exposed in Section **??**. The general automatic differentiation method is however crucial to master because advanced deep-learning architectures are not purely feedforward, and might include recursive connexions. Furthermore, automatic differentiation is useful outside deep learning, and considerably eases prototyping for modern data-sciences with complicated non-linear models.

15.2.4 Scattering Transform

The scattering transform, introduced by Mallat and his collaborators, is a specific instance of deep convolutional network, where the filters $(\psi_{\ell,r,s})_{\ell,s,r}$ are not trained, and are fixed to be wavelet filters. This network can be understood as a non-linear extension of the wavelet transform. In practice, the fact that it is fixed prevent it to be applied to arbitrary data (and is used mostly on signals and images) and it does not lead to state of the art results for natural images. Nevertheless, it allows to derives some regularity properties about the feature extraction map $f(\cdot, \beta)$ computed by the network in term of stability to diffeomorphisms. It can also be used as a set of fixed initial features which can be further enhanced by a trained deep network, as shown by Edouard Oyallon.

Bibliography

[1] Amir Beck. *Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with MATLAB*. SIAM, 2014.

[2] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] E. Candès and D. Donoho. New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities. *Commun. on Pure and Appl. Math.*, 57(2):219–266, 2004.

[5] E. J. Candès, L. Demanet, D. L. Donoho, and L. Ying. Fast discrete curvelet transforms. *SIAM Multiscale Modeling and Simulation*, 5:861–899, 2005.

[6] A. Chambolle. An algorithm for total variation minimization and applications. *J. Math. Imaging Vis.*, 20:89–97, 2004.

[7] Antonin Chambolle, Vicent Caselles, Daniel Cremers, Matteo Novaga, and Thomas Pock. An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery*, 9(263-340):227, 2010.

[8] Antonin Chambolle and Thomas Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319, 2016.

[9] S.S. Chen, D.L. Donoho, and M.A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1999.

[10] Philippe G Ciarlet. Introduction à l'analyse numérique matricielle et à l'optimisation. 1982.

[11] P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *SIAM Multiscale Modeling and Simulation*, 4(4), 2005.

[12] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commun. on Pure and Appl. Math.*, 57:1413–1541, 2004.

[13] D. Donoho and I. Johnstone. Ideal spatial adaptation via wavelet shrinkage. *Biometrika*, 81:425–455, Dec 1994.

[14] Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

[15] M. Figueiredo and R. Nowak. An EM Algorithm for Wavelet-Based Image Restoration. *IEEE Trans. Image Proc.*, 12(8):906–916, 2003.

[16] Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing*, volume 1. Birkhäuser Basel, 2013.

[17] Stephane Mallat. *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.

[18] D. Mumford and J. Shah. Optimal approximation by piecewise smooth functions and associated variational problems. *Commun. on Pure and Appl. Math.*, 42:577–685, 1989.

[19] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[20] Gabriel Peyré. *L'algèbre discrète de la transformée de Fourier*. Ellipses, 2004.

[21] J. Portilla, V. Strela, M.J. Wainwright, and Simoncelli E.P. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Trans. Image Proc.*, 12(11):1338–1351, November 2003.

[22] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, 1992.

[23] Otmar Scherzer, Markus Grasmair, Harald Grossauer, Markus Haltmeier, Frank Lenzen, and L Sirovich. *Variational methods in imaging*. Springer, 2009.

[24] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[25] Jean-Luc Starck, Fionn Murtagh, and Jalal Fadili. *Sparse image and signal processing: Wavelets and related geometric multiscale analysis*. Cambridge university press, 2015.