Notes and Comments on S. Mallat's Lectures at Collège de France (2018)

The challenge of learning in the face of the curse of high dimensionality"

J.E Campagne[*]

Apr. 2018; rév. 3 octobre 2023

[*]If you have any comments or suggestions, please send them to `jeaneric DOT campagne AT gmail DOT com`

Table des matières

1. Foreword

***Disclaimer**: What follows are my informal notes in French, translated into rough English, taken on the fly and reformatted with few personal comments ("NDJE" or dedicated sections). It is clear that errors may have crept in, and I apologize in advance for them. You can use the email address provided on the cover page to send me any corrections. I wish you a pleasant read.*

Please note that the Collège de France website has been redesigned. You can find all the course videos, seminars, as well as course notes not only for this year[1].

I would like to thank the entire Collège de France team for producing and editing the videos, without which the preparation of these notes would have been less convenient.

Also, note that S. Mallat[2] provides open access to chapters of his book *"A Wavelet Tour of Signal Processing"*, 3rd edition, as well as other materials on his ENS website.

This year, 2018, is the first in the cycle of S. Mallat's Data Science Chair. It concerns on the **Curse of high dimensionality**.

2. Introduction to the Lecture Series

The objective of this discipline is the extraction of knowledge from data. To see if there are patterns emerging that can be formulated as algorithms.

So, *a priori*, the Data contains **Information** (Model) that needs to be extracted using **Algorithms**. A guiding principle: "first understand the data!"

It is an emerging discipline; there used to be statisticians, but now it includes computer scientists and mathematicians. Mathematics encompasses not only statistics, probabilities, representations (analysis, Fourier, wavelets, etc.) and geometries (symmetry groups, etc.). On the computer science side, there is AI, databases, distributed/parallel computing.

1. https://www.college-de-france.fr/chaire/stephane-mallat-sciences-des-donnees-chaire-statutaire/events
2. https://www.di.ens.fr/~mallat/CoursCollege.html

Data comes in various forms: images, sounds, linguistics (text), but also physics, chemistry, and more.

2.1 Signal Processing

It's the estimation/approximation with the aim of recovering the cleanest possible signal $x(u)$ as follows:

$$x(u) \quad u \in \mathbb{Z}^\ell \text{ (discrete case)}, u \in \mathbb{R}^\ell \text{(analog case)}$$

The discrete case is not the simplest; in fact, in the continuous case, notions of regularity are more natural. But in practice, we end up computing in the discrete case. We encounter problems like:

$$z = Ax + b$$

Inverse problem (Is A invertible?)/comprehensive sensing (how much data do I need?)/data compression.

What is the dimension? It's the dimension of the variable u: d (e.g., time + typical volume in physics: 4). So here, we are in *low dimension*, whereas we will see that when dealing with an image, for example, it's the number of pixels that determines the problem's dimension (*high dimension*).

2.2 Modeling/Unsupervised Learning

Estimating a model for $x(u)$ is finding its representation in a space Ω, either **probabilistic** or **deterministic**. This has nothing to do with whether the underlying problem is stochastic or not. The **probabilistic** representation can be richer because we can associate a probability density $p(x)dx$ with the signal being in a small volume element of Ω. In the *deterministic* case, we can at best say that the probability density is uniform, so we seek models of maximum entropy.

So, a probabilistic model is a random vector X associated with a probability density $p(x)$, and we want to construct an estimator $\tilde{p}(x)$. For this, we use data (examples), $\{x_i\}_{1\leq i\leq n}$.

In 1D, we create a simple histogram. But the dimension is that of the signal d, typically the number of samples. And here it is of the order of 10^{6-9} if you think about image pixels, but it can be **much larger**, even 10^{23} in statistical mechanics.

So, simple, intuitive methods don't work at all in very high dimensions because there are almost never points $x(u)$ close to each other, or only at the cost of having boxes with volumes comparable to the entire space. Local calculations are impossible in high dimensions, and strong assumptions are needed to estimate $p(x)$.

The probabilistic representation, as we will see later, does not solve the curse of dimensionality problems.

If we have a model, we can do Signal Processing because we know "where the signal is", and coding is more meaningful (prediction). We can synthesize new signals/images. We can also address the explanation of the underlying phenomenon. Typically, statistical physics.

2.3 Prediction/Supervised Learning

Here we address the problem of estimating the response to a question (y) from the data x. If it's a *classification* problem, y belongs to an alphabet (class index), and there is no *a priori* topology on y. The challenge is to define distances between classes. If it's a *regression* problem, $y \in \mathbb{R}$ or $\mathbb{R}^c$, which can be seen as c real numbers. While regression might seem more challenging because it's like having an infinite number of classes, it's not the case, and the difficulty is equivalent. In classification, we calculate boundaries whose dimension is just $d-1$, so when d is large, finding a boundary or a representation of y makes no difference.

If there's a unique response, then $y = f(x)$ (this is the case in very high dimensions; there are no two identical samples), and the challenge is to estimate f as follows:

$$y = f(x) \to \tilde{y} = \tilde{f}(x)$$

where $\tilde{f}$ is an approximation of f with a large number of variables d.

But in supervised learning (classification or regression), we have examples $\{x_i, y_i\}_{1\leq i\leq n}$. We know a priori that $y_i = f(x_i)$, so we estimate $\tilde{y_i} = \tilde{f}(x_i)$, and we would like $\tilde{y_i} \sim y_i$. In the case of "unsupervised" learning, we don't have $p(x_i)$. Still, supervised learning has its own difficulty because the diversity of functions f is much greater than that of probability densities $p(x)$.

Problems tackled: speech recognition, vision, medical, physics, sociology, neurophysiology. Is there an equivalence class among all these problems? If so, algorithms developed in one domain can be recycled, and generic algorithms can also be developed.

2.4 Commonalities

A priori, there are models. The issue of the **regularity** of x is fundamental to understanding the representation. But regularity is not easy: there are singularities, and they're not everywhere (think of an image). So, in compression, signal processing, we need to understand regularity to sample cleverly (regular: linear case; singular: nonlinear case). What is the model's complexity?

In the case of $p(x)$, we are in high dimension. We approach the problem with notions of invariance (translation/rotation of images), so stationary processes with probabilities that don't change. Entropy measures complexity, and there are two notions of entropy that will interest us: Kolmogorov/deterministic entropy and Shannon/probabilistic entropy.

In prediction, the answer to the question $f(x_i)$ is the interpolation $f(x)$. The choice of the approximation function comes from the regularity that defines the model class/set $\mathcal{H}$. However, mathematics in signal processing (low dimension) is well understood (Sobolev space, etc.), but in high dimension, we lack tools currently. Regularity and sparsity are dual notions to each other.

Tool	Signal Processing	Modeling	Prediction (Supervised Learning)
Model	Regularity of $x(u)$	Regularity of $p(x)$	Regularity of $f(x)$, $f \in \mathcal{H}$
Complexity	Entropy (Kolmogorov/Shannon)	Invariance/Entropy	
	Deterministic/Probabilistic		

Tool	Signal Processing	Modeling	Prediction (Supervised Learning)
Representation $\Phi(x)$	Redundant Basis/Dictionary, Sparsity		
	$x = \sum_m \alpha_m g_m$ with $\Phi(x) = \{\alpha_m\}$ e.g., $\mathcal{D} = \{g_m\}$ (Fourier/Wavelets)	$\Phi(x)$ generalized moments (probabilistic): a family of expectations e.g., $\{E(\phi_k(x)\}_k$ with $\phi_k(x) = x^2$ and polynomials or not in general	$\Phi(x)$; discriminant pattern on y. Change of representation to reveal a linear problem. Invariant notion.
Computer Science	here, one tool	Notion of "associative" memory, virtual reality, information graphics (synthesis)	AI, GPUs, and Python but, in fact, it's not very easy to master what we're doing

While Signal Processing has developed since the 1930s, with a strong industrial presence and beautiful inverse problems, the mathematical framework is well understood: Harmonic Analysis (Fourier), Statistics, and Optimization.

Modeling: Signal Processing + Probability and concentration/deviation related to Entropy due to high dimension (law of large numbers), group theory, but not much. Mathematics has understood entropy since the 1970s.

Prediction: Signal Processing + Modeling; group theory is more present, and geometry is used to find distances (classification).

Computer science (AI + Python) is not very difficult to access, but the low dimension of Signal Processing is not well known to newcomers in Machine Learning. Deep Neural

Networks are not well understood, so one must start with low dimension. Another aspect to master is the bias/variance trade-off and the curse of high dimension.

3. The Problem of Overfitting

3.1 General Methodology

The first part of the course recalls the ingredients for tackling the proposed challenges. First and foremost, it is necessary to **familiarize oneself with the data**: understand their statistics, see if they need to be denoised before processing, etc. Split the samples into (3/4, 1/4) for training and internal testing.

Next, avoid the idea of using Neural Networks (abbreviated as NN or MLP) and, worse, Deep Neural Networks (DNN) right away. If the number of training samples is not very large, start with: linear classifiers/regressors, kernels, Decision Trees, Boosting (XGBoost on trees, Gradient Decision Tree). For high-dimensional cases (e.g., images, etc.), NNs and DNNs will be more suitable.

In supervised learning, y is the response to a question posed on data $f(x)$:

$$x \to \tilde{y} = \tilde{f}(x)$$

with $\tilde{f}$ selected from a class of functions $\mathcal{H}$. Q: How will it be selected? A: Through learning on data for which we know the answer:

$$\{x_i, y_i\}_{i \leq n}$$

and we want the estimation of the response $\tilde{f}$ to be close to the true response:

$$\tilde{y}_i = \tilde{f}(x_i) \simeq f(x_i) = y_i.$$

We provide a loss function that measures the error $r(y, \tilde{y})$, which differs depending on whether it is a "regression" or "classification" problem. Typically:

Regression	Classification
$y \in \mathbb{R}$	$y \in \mathcal{A}$
$(y - \tilde{y})^2$	1 if $y \neq \tilde{y}$, 0 otherwise

but there are other functions depending on the specific scenario.

So, on the training samples, we estimate an empirical risk calculated from the algorithm after learning:

$$\tilde{R}_e(\tilde{f}) = \frac{1}{n} \sum_{i=1}^{n} r(y_i, \tilde{f}(x_i))$$

and we want to minimize it to select $\tilde{f}$. But what really interests us is the generalization error that we want to minimize:

$$R(h) = E_{X,Y}(r(Y, h(X))))$$

In a challenge, we provide $\{x_i, y_i\}_{i \leq n}$, the empirical risk function $\tilde{R}_e$, and test examples $\{x_i^t\}_{i \leq n_t}$ for which the submission site calculates the score:

$$\tilde{R}_e^t(\tilde{f}) = \frac{1}{n_t} \sum_{i} r(y_i^t, \tilde{f}(x_i^t))$$

with y_i^t hidden. You can make only **2 submissions per day**.

The difference between $\tilde{R}_e(\tilde{f})$ and $\tilde{R}_e^t(\tilde{f})$ is that the first risk is "contaminated" by the training samples, while the second one is not; it provides a better estimate of the generalization risk $R(h)$. That being said, if you make many submissions on the site to improve your algorithm, eventually, the score $\tilde{R}_e^t(\tilde{f})$ will also be contaminated because the algorithm will adapt to the test samples.

Therefore, the protocol includes a first review of all submitted algorithms on a series of new samples that no one has seen before in **June (1st)**, and a second and final benchmarking will be done in **December** to conclude the challenges.

3.2 Linear and Kernel Algorithms: Regression/Classification

$$x = (x_1, \dots, x_d)$$

with d very large, so there's a very low chance that two samples are close to each other. Instead of finding a complex boundary to separate two classes, we'll try to adapt a representation $x \to \phi(x)$ to obtain:

$$\phi(x) = (\phi_1, \dots, \phi_m) \in \mathbb{R}^m$$

with a new dimension m, and we hope that the separation between the two classes is a **hyperplane**. We've flattened the boundary.

Suppose we know the parameterization of $\phi(x)$. The linear classifier is given by the sign of the distance from a sample to the hyperplane with normal w in m-dimension:

$$\tilde{y} = \operatorname{sgn}(\langle \phi(x), w \rangle - b) = \operatorname{sgn}\left[\sum_{k=1}^{m} \phi_k w_k - b\right]$$

We've selected an algorithm $\mathcal{A}$ with parameters (m+1): w, b.

We need to find (w, b) that minimizes the empirical risk of binary classification since we're testing ± 1. In the case of regression, y is continuous:

$$y = \langle \phi(x), w \rangle - b$$

and the empirical risk is the squared difference:

$$(y - (\langle \phi(x), w \rangle - b))^2$$

So, the challenge is in choosing the representation $\phi(x)$. The Boosting and Decision Trees algorithms can be seen in this linear framework.

The projection axis is the discriminant criterion between the two classes by a linear combination. The attributes ϕ_k are weak discriminants, but the aggregation of all these attributes can effectively separate the samples if their number is sufficiently large (i.e., the dimension m is large). In the case of Decision Trees and Gradient Decision Trees, we also perform an aggregation (voting) of weak discriminants.

In fact, neural networks will learn both (w, b) and the ϕ_k, so the representation is also acquired. But even there, prior information is required (in NN/MLP, it's the architecture that provides the prior). So, with learning, we minimize the empirical risk:

$$\tilde{f} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \ \tilde{R}_e(h)$$

But we want to minimize the generalization risk:

$$f_a = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \ R(h)$$

The difference between $\tilde{R}_e(h)$ and $R(h)$ is the main subject of overfitting, which occurs rapidly when conforming too closely to the data.

3.3 Bias-Variance

So, what is $R(\tilde{f})$, i.e., the true risk evaluated with my trained estimator? We can demonstrate that if we call f_I the "ideal" algorithm:

$$\boxed{R(f_I) \leq R(\tilde{f}) \leq R(f_I) + 2\underset{h \in \mathcal{H}}{\operatorname{Max}} \ |R(h) - R(\tilde{h})|}$$

(the proof is in the associated lecture notes).

What this says (especially the second inequality) is that the **generalization error** $R(\tilde{f})$ is bounded by a **bias term** because the class $\mathcal{H}$ is not perfect, even if we could ideally obtain the minimum risk:

$$R(f_I) = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \ R(h) = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \ E_{X,Y}(r(Y, h(X)))$$

and a **variance term** linked to the *variability of h within $\mathcal{H}$*. Schematically, we have:

$$Error \leq Bias + Variance$$

but we can't reduce both terms simultaneously. If the class $\mathcal{H}$ is large, bias decreases, but

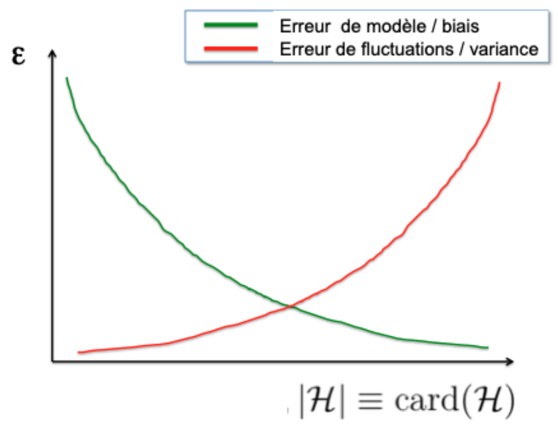

FIGURE 1 – Bias-Variance Error as a function of the size of the set $\mathcal{H}$.

fluctuations increase (see an illustration in Figure 1).

The key is to find model classes with small sizes (i.e., small $card(\mathcal{H})$) to control fluctuations (entropy) and to be good at approximating the problem to keep the bias as low as possible.

The "Probably Approximately Correct" (**PAC**) theorem states that we want *uniform results* regardless of the distribution of (X, Y), which is unknown *a priori*, and that for n tending towards ∞, the term of fluctuations tends to 0.

— **PAC Theorem**:
  If

$$R(h) \in [0,1](\text{bounded}), |\mathcal{H}| \equiv \text{card}(\mathcal{H}) < \infty$$

  then

$$\mathbb{P}\left(\underset{h\in\mathcal{H}}{Max}|R(h) - R(\tilde{h})| \leq \varepsilon\right) \geq 1 - \delta$$

  and for that, we will need a number of samples such that

$$n \geq \frac{\log(|\mathcal{H}|) + \log(2/\delta)}{2\varepsilon^2}$$

This means that if we want a small error ε with a reasonably high probability (i.e., a small δ), we also need to use a class $\mathcal{H}$ with small dimensions to avoid requiring an excessive number of examples. But can we have it all? Another way to look at this inequality, if n is fixed:

$$\boxed{\varepsilon^2 \geq \frac{\log(|\mathcal{H}|) + \log(2/\delta)}{2n} = \frac{\log(|\mathcal{H}|)}{2n} + \text{condition term}}$$

So, what I can roughly control is the size of $\mathcal{H}$, which is essentially the dimension of the parameter space.

What do these two terms correspond to? In linear regression:

$$h(x) = \sum_{k=1}^{m} \phi_k w_k - b$$

So, the parameters $(\{w_k\}_{k \leq m}, b)$ are in $\mathbb{R}^{m+1}$ and can, in principle, take an infinite number of values. However, if we assume $\phi_k < C$, and if we make small changes, the quantity $|R(h) - R(\tilde{h})|$ doesn't change much. We can therefore **quantize the parameters** like:

$$w_k = p_k \Delta$$

(similarly for b) with p_k having N possible values. Thus, $N^{m+1} = |\mathcal{H}|$ and

$$\frac{\log(|\mathcal{H}|)}{2n} = \frac{(m+1)\log N}{2n} \sim \left(\frac{m}{n}\right)$$

So, to be able to constrain m parameters, you need at least as many samples, and if you don't want too much fluctuation, you really need $n \gg m$. Paradoxically (*a priori*), for NN/MLP, the number of parameters is much larger than the number of training samples. So, be aware that in NN, there are contractive non-linearities.

Another interpretation is that $\log|\mathcal{H}|$ is related to entropy: it's the number of bits to describe $\mathcal{H}$. More generally, it's the notion of complexity.

The proof of this theorem is in the lecture notes. There's a lemma (*Hoeffding's inequality*) that controls **large deviations**.

Suppose $\{Z_i\}$, n variables i.i.d / $\forall i \leq n$, $Z_i \in [a, b]$ and $E(Z_i) = \mu$. If $\bar{\mu} = 1/n \sum Z_i$, then $\forall \varepsilon > 0$:

$$\mathbb{P}\left(|\mu - \bar{\mu}| \geq \varepsilon\right) \leq 2e^{-\frac{2n\varepsilon^2}{(b-a)^2}}$$

(Note: the estimator is unbiased $E(\bar{\mu}) = \mu$). The assumption of being « iid » (**independent**

and identically distributed) is strong, and errors in learning/generalization come from the bias (= lack of independence) between samples. This is a very important aspect for those who provide challenges.

In the proof of the PAC theorem, it appears that the upper bound is obtained by bounding the probability of the union of two sets through a sum of probabilities. This can be refined further.

4. Curse of High Dimensions (Part 1)

4.1 Recap

The algorithms $\tilde{f}$ that we select are those that minimize an empirical risk:

$$\tilde{f} = \underset{h\in\mathcal{H}}{\operatorname{argmin}}\ \tilde{R}_e(h) = \underset{h\in\mathcal{H}}{\operatorname{argmin}} \left\{ \frac{1}{n} \sum_i r(h(x_i), y_i) \right\}$$

and we would like this risk to be a good estimator of the average (generalization) risk:

$$f_I = \underset{h\in\mathcal{H}}{\operatorname{argmin}}\ E_{X,Y}[r(h(x), Y)]$$

We have established that the empirical risk of the trained model is bounded:

$$R(f_I) \leq \tilde{R}(\tilde{f}) \leq R(f_I) + 2\underset{h\in\mathcal{H}}{Max}|R(h) - R(\tilde{h})|$$

with an upper bound given by two terms: the **bias** ("Is the model adequate to answer the problem's question?") and the **variance** or fluctuations ("Is my model class too large?").

Furthermore, we have seen that if the risk is bounded, say [0,1], then the term for fluctuations is not too large, meaning:

$$\mathbb{P}\left(\underset{h\in\mathcal{H}}{Max}|R(h) - R(\tilde{h})| \leq \varepsilon \right) \geq 1 - \delta$$

if we have:

$$n \geq \frac{\log(|\mathcal{H}|) + \log(2/\delta)}{2\varepsilon^2}$$

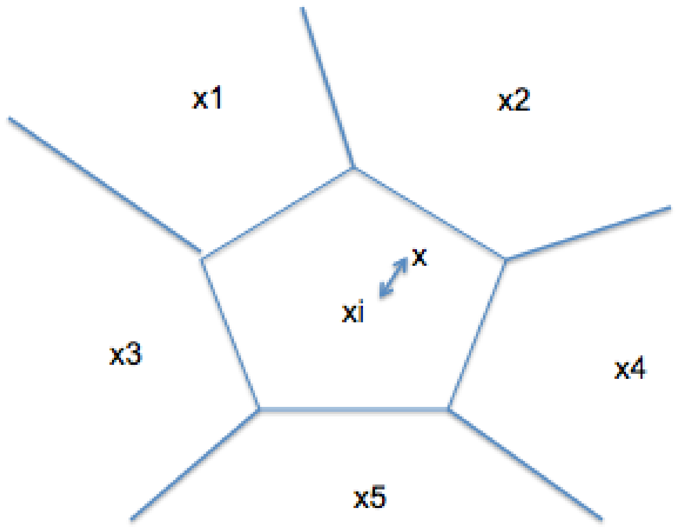

FIGURE 2 – Cells generated by the k-Nearest Neighbors algorithm.

or at fixed n, we deduce the fluctuations error:

$$\varepsilon^2 \geq \frac{\log(|\mathcal{H}|) + \log(2/\delta)}{2n}$$

Implicitly, the training sample base is a good reflection of the underlying probabilities (X, Y).

4.2 Case: k-Nearest Neighbors Algorithm

Let's assume that a reasonable *a priori* algorithm is to say that my approximation $\tilde{y}$ of the response y from x satisfies:

$$\tilde{f}(x) = \tilde{y} = y_i \quad \text{if} \quad ||x - x_i|| \leq ||x - x_j|| \text{ for } (j \neq i)$$

This draws cells within which x is associated with x_i (Figure 2), for which we know the response y_i.

But why doesn't it work? In fact, if I have m parameters, we have seen that the error is dominated by the ratio m/n with n being the number of samples. However, here the number of parameters m is essentially the number of points $m \sim n$, and therefore ε will never be small. We do not control the generalization error. It would be necessary to regularize it to reduce this number of free parameters [3].

4.3 Rate of Fluctuation Decay

The way to control generalization error is by controlling the rate at which fluctuation errors (variance) decrease: finding the right class $\mathcal{H}$ and the number of samples that should not explode.

Let's imagine, for example, that the risk of the ideal algorithm f_I decreases with the size of the class as:

$$R(f_I) \leq C(\log|\mathcal{H}|)^{-\alpha}$$

then:

$$\mathbb{P}(R(\tilde{f}) < 3\varepsilon) \geq 1 - 2e^{-(C/\varepsilon)^{1/\alpha}} \quad \text{if} \quad n \geq \frac{C^{1/\alpha}}{\varepsilon^{2+1/\alpha}}$$

So, if we are able to bound the risk, then with almost equal probability to 1, the error will be very small if n is large ($\sim (1/\varepsilon)^{2+1/\alpha}$). However, α will govern the kinetics, and in high dimensions, natural decay is very slow, so n should be absolutely enormous.

We will look for theorems of this kind in the rest of the course to guide us on how to obtain minimal error. The proof starts from the PAC theorem by setting $\log|\mathcal{H}| = \log(2/\delta) = n\varepsilon^2$. So, the problem is to find classes of estimators whose decay is as fast as possible, where α is the largest. In this case, the statistical part is under control. But it's not simple! We will assume that y is unique, so $y = f(x)$ with f unknown, and therefore the risk given by the fact that we are in class $\mathcal{H}$ is:

$$R_I = \min_{h \in \mathcal{H}} E_X[r(h(X), f(X))]$$

This is therefore an **approximation problem** of the function f (unknown) by a function $h \in \mathcal{H}$; but we do not know the probability distribution of X. One way to protect ourselves

3. Note: The k-means algorithm is a case where the number of seeds chosen *a priori* is small compared to n

from this lack of knowledge of the data statistics is to say that the average is smaller than the maximum value. So, with a quadratic risk (regression type):

$$R_I \leq \min_{h\in\mathcal{H}} \sup_{x\in\Omega} |h(X) - f(X)|^2 = \min_{h\in\mathcal{H}} ||h - f||_\infty$$

where Ω is the support of the data (we do not know the probability density). We want to control $||h - f||_\infty$ with f unknown. So, not only do we need data samples, but we also need data models. We are looking to control the quantity:

$$\max_{f\in\mathcal{C}} R(f) = \max_{f\in\mathcal{C}} \min_{h\in\mathcal{H}} ||h - f||_\infty$$

The previous theorem tells us that if:

$$\max_{f\in\mathcal{C}} R(f) \leq C(\log|\mathcal{H}|)^{-\alpha}$$

then we have succeeded.

But what kind of **prior information** can we have? It's the **regularity** of functions. Let's think about a 1-dimensional approximation problem; if the underlying function varies slowly, we don't need many samples, but if the function is very irregular, the number of samples can be large or they will be localized at singularity points.

4.4 Regularity (Simple) in the Sense of Lipschitz

The problem $x \in \mathbb{R}^d$ with d very large.

— **Definition**: f is **locally Lipschitz** at x if:

$$\exists\ C_x / \ \forall x' \in \mathbb{R}^d \quad |f(x) - f(x')| \leq C_x ||x - x'||$$

(Note: $||x||^2 = \sum |x_i|^2$)

We say that f is **uniformly Lipschitz** if it is everywhere and all C_x are smaller than C. If f has a bounded derivative, then it is Lipschitz.

If f is Lipschitz on $\mathbb{R}$, then it is almost everywhere differentiable. So, there are points where the derivative is not determined. In dimension d, this generalizes with partial derivatives. **One can consider the notion of a Lipschitz function as « equivalent » to a function with a bounded derivative.**

— **Definition**: f is locally Lipschitz-α if:

$$\exists\ C_x, p_x(x') \text{ of degree } q < \alpha / \ \forall x' \in \mathbb{R}^d \quad |f(x) - p_x(x')| \leq C_x ||x - x'||^\alpha$$

Typically, we look at the residue of the **Taylor polynomial** at x of $f(x)$. If $C_x < C$, then we say that the function is Lipschitz-α uniform.

So, let's assume that we have a class of Lipschitz functions:

$$\mathcal{C} = \Big\{ f : \mathbb{R}^d \to \mathbb{R} \ / \text{ uniformly Lipschitz} \Big\}$$

How many samples do we need? We have seen that we need to control the rate of decrease of:

$$\max_{f \in \mathcal{C}} R(f) \leq C(\log |\mathcal{H}|)^{-\alpha}$$

Let's go back to the **k-Nearest Neighbors** algorithm. We know that the function is Lipschitz around the sample points $\{x_i\}$, then:

$$|f(x) - f(x_i)| \leq C_x ||x - x_i||$$

In other words, we have a good approximation of $f(x)$ by saying that it is equal to $f(x_i)$. This leads to a **piecewise approximation** (see Figure 3). We generalize to 2D and beyond. Using this classifier is not a good idea for fluctuations, but for modeling/approximation, let's see what happens from a decay perspective.

If f is uniformly Lipschitz-α, then $C > 0$:

$$||f - \tilde{f}||_\infty \leq C\varepsilon \quad \text{with} \quad \varepsilon = \max_x \min_i |x - x_i|$$

The error between the function and its piecewise approximation is governed by the maximum distance between any point x and its nearest neighbor in the sample. The worst case is when x is in a hole! So, we need to ensure that for every x, the maximum distance to a sample is not too large. How many samples n do we need to achieve this?

Let's imagine that $x \in \Omega$ with $\Omega \in [0,1]^d$, the balls of radius ε centered on the

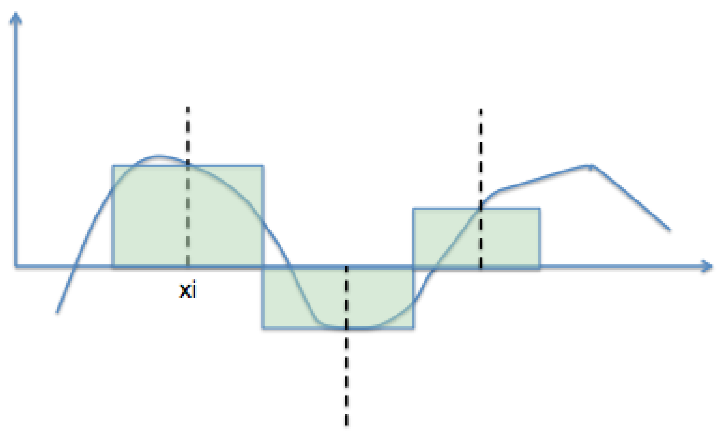

Figure 3 – Approximation of $f(x)$ by a piecewise constant function.

samples cover all of Ω, i.e.,

$$\Omega \subset \bigcup_{i=1}^{n} \mathcal{B}_\varepsilon(x_i)$$

This is a coding problem: the logarithm of the number of balls gives the number of bits of information.

— **Property**: the optimal distribution of balls in $[0,1]^d$ satisfies:

$$\frac{\sqrt{d}n^{-1/d}}{2} \geq \varepsilon \geq \frac{\sqrt{d}n^{-1/d}}{2}\sqrt{\frac{2}{\pi e}}\left[1 + O\left(\frac{d}{\log d}\right)\right]$$

so, by taking the second inequality, we have something like:

$$\boxed{n \gtrsim \varepsilon^{-d}\left[\frac{d}{2\pi e}\right]^{d/2}}$$

But we have a "double penalty": the term ε^{-d} explodes as soon as we want to constrain the error ε, and the constant term varies according to $d^{d/2}$. **So, very quickly, even in low dimensions, the number of samples becomes enormous if we assume that the function is Lipschitz-α.** In terms of approximation error decay, we can put the result in the form:

$$||f - \tilde{f}||_\infty \leq C\varepsilon \leq C\frac{\sqrt{d}n^{-1/d}}{2}$$

so the decay is extremely slow in $n^{-1/d}$, and we have something like:

$$R(f_I) \leq C\frac{\sqrt{d}}{2}(\log|\mathcal{H}|)^{-1/d}$$

This is the curse of high dimension because the space is essentially "empty" in Euclidean distance!

How to escape it? Find the right regularity of functions from a wide range of variability. You need to look at the nature of the data and go back to signal processing before doing machine learning.

We have x, y, and f in $y = f(x)$. The regularity of f for $x \in \Omega$ is supervised learning. For Ω, we said that its volume is 1, but if my data is in a volume/hypersurface of much lower dimension (it's feature selection)! What is the regularity of $x(u)$? For example, within an image, there are (ir)regularities that need to be captured. We will start with **low dimension**:

— Distinguish between **linear and non-linear** cases; non-linear does not mean complicated, but we only do non-linear if needed!

— Introduce **sparsity** (**sparse vector**), which is a dual concept to **regularity**.

5. Curse of High Dimensions (Part 2)

Using **Signal Processing** (representation study) for dimensionality reduction (linear) and sparsity (non-linear).

5.1 Orthogonal Bases: Linear/Non-linear (Introduction)

In an orthonormal basis $\mathcal{B} = \{g_m\}_{1\leq m\leq d}$, to represent a d-dimensional signal $x(u)$, we have:

$$x = \sum_m \langle x, g_m\rangle g_m$$

We change the representation x_i that are the initial components of x by introducing the inner products of x in the new basis. The question is whether, in this new basis, we can retain only a few coefficients while maintaining a good approximation of the signal. In the linear case, we can order them. So,

$$x_M = \sum_{m=1}^{M} \langle x, g_m \rangle g_m$$

The error made is:

$$\varepsilon_M = ||x - x_M||^2 = \sum_{k=M+1}^{d} |\langle x, g_m \rangle|^2$$

How does the decay of coefficients change with M?

If

$$|\langle x, g_k \rangle| \leq C k^{-\alpha} \quad (\alpha > 1/2)$$

then

$$\varepsilon_M < \frac{C^2}{1 - 2\alpha} M^{1-2\alpha}$$

So if the coefficients have a rapid decay (α large), then the error is correspondingly better. We have just followed a similar reasoning as with the learning algorithm, but instead of working on the function f, we are working on the data representation. The question becomes: What is the best basis that guarantees the greatest decay?

When we have no *a priori* knowledge of the signal representation in **the linear case**, it can be shown (will be shown) that **the best basis is the Karhunen-Loève** or **principal components** (diagonalized covariance operator) (see Figure 4).

However, as soon as we have information about the data, we can go further. For example, if there is **translation invariance** (image, sound) (no absolute zero), then the best basis is the **Fourier** basis. But by Shannon's theorem, there is a correspondence between Fourier and uniform sampling, which is suitable only if the data has no singularities. At that point, the Fourier basis is not optimal because it is clear that finer sampling is needed near singularities (it must be **adaptive**). This is a non-linear process because adaptive samplings on f and g do not fit well with that of $f + g$.

So, can we do better in non-linear? Let's reconsider the decomposition of x in the new orthonormal basis:

$$x = \sum_k \langle x, g_k \rangle g_k$$

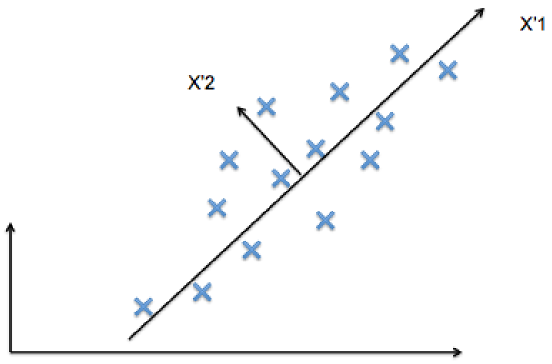

FIGURE 4 – Principal components of a sample cloud.

and now, we define the approximation by taking M coefficients, but not necessarily the first M (because it depends on the signal x):

$$x_M = \sum_{k \in I_M} \langle x, g_k \rangle g_k$$

and

$$\varepsilon_M = ||x - x_M||^2 = \sum_{k \notin I_M} |\langle x, g_k \rangle|^2$$

We select the indices for which the coefficients are the largest M:

$$I_M = \{k / |\langle x, g_k \rangle| \geq T_M\}$$

and **the threshold T_M (easy to implement) will define an approximation that adapts to the signal or, in other words, to the smoothness of the function.** This is how it works in **Wavelet** bases. Note that a neuron implements this nonlinear coefficient thresholding.

5.2 Denoising

If we denote x as the base signal and B as the noise, what we actually get is a noisy signal Z such that (capital letters = random):

$$Z(u) = x(u) + B(u)$$

We will construct a **deterministic model** of x and a **random model** of B. Indeed, it is quite simple to have random representations of noise (with some assumptions), but, by nature, the signal is more complicated, and contrary to popular belief, a « deterministic » model is simpler than a stochastic model. For a deterministic model, we say that $x \in \Omega$ (e.g., grayscale level of an image), while the stochastic (probabilistic) model adds that there are regions of Ω that are more or less populated (e.g., for noise, one often imagines Gaussians, so a simple representation).

So, eliminating noise means finding an operator L such that our estimator of X is:

$$\tilde{X}(u) = LZ(u)$$

and we want to minimize the mean square error (MSE):

$$R = \underset{B}{E}||x - LZ||^2$$

Two approaches:

1. To model the **signal**: A **linear** model means that x is concentrated on a **hyperplane** (H).
2. For noise: A Gaussian model (white noise) that populates all of Ω. So, it is sufficient to eliminate all noise components outside of H through projection (see Figure 5) [4].

So, let's consider a Gaussian model of the noise (**Gaussian White Noise or GWN**):

— **Zero mean**: $E[B(u)] = 0$ for all u (or we may have removed the mean).

— **Uncorrelated components**: $E[B(u)B(u')^*] = 0$ for all $u \neq u'$, and $u = u'$, then $E[|B(u)|^2] = \sigma^2$.

Therefore, the probability density of obtaining noise b is:

$$p(b) = c\ \exp\left(-\frac{1}{2}b^T\mathbf{\Sigma}^{-1}b\right) = c\ e^{-|b|^2/(2\sigma^2)}$$

with $\mathbf{\Sigma} = \sigma^2 I$ as the covariance matrix.

4. Note: conventionally in statistics, we model x as Gaussian, and it leads to the best linear estimator. But here, we take a deterministic model of x, which is less constrained. We will see that non-linear methods can be very effective in certain circumstances.

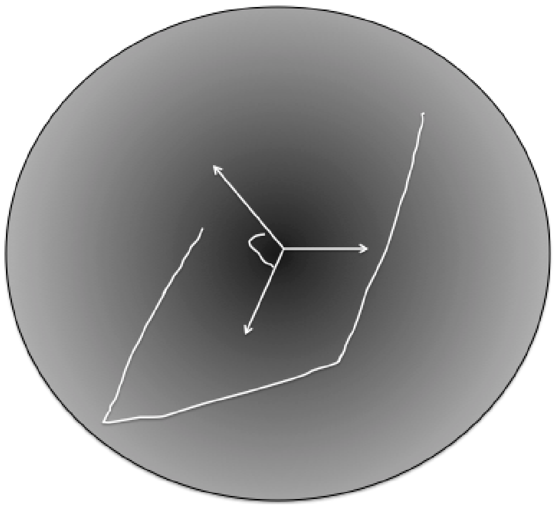

FIGURE 5 – The noisy signal $Z(u)$ consists of a clear signal $x(u)$ evolving in a hyperplane and noise $B(u)$. A projection of Z onto the hyperplane is needed to recover $x(u)$.

White noise remains white in all bases (it is isotropic), consider a basis $\{g_k\}_k$ of Ω:

$$\langle Z, g_k\rangle = \langle x, g_k\rangle + \langle B, g_k\rangle$$

If B is GWN:

$$E(\langle B, g_k\rangle, \langle B, g_k'\rangle^*) = \sigma^2\delta_{k,k'}$$

This is shown by expanding the dot product:

$$\langle B, g_k\rangle = \sum_u B(u)g_k^*(u)$$

where $B(u)$ are random variables and $g_k^*(u)$ are constants. Therefore,

$$\begin{aligned} E\left[\sum_{u,u'} B(u)B^*(u')g_k^*(u)g_{k'}(u')\right] &= \sum_{u,u'} E[B(u)B^*(u')]g_k^*(u)g_{k'}(u') \\ &= \sigma^2 g_k^*(u)g_{k'}(u') = \sigma^2\delta_{k,k'} \end{aligned}$$

The only difficulty is knowing what is random and what is not.

So, let's take the equivalent of the linear projection operator where we keep only M

coefficients, meaning our approximation is:

$$\tilde{X} = Proj_{V_M} Z$$

and we want to minimize:

$$R = \underset{B}{E} ||Proj_{V_M} Z - x||^2$$

Simply,

$$\begin{aligned} ||x - P_{V_M} Z||^2 &= ||(x - P_{V_M} x) - P_{V_M} B||^2 \\ &= ||(x - P_{V_M} x)||^2 + ||P_{V_M} B||^2 \end{aligned}$$

because $x - P_{V_M} x$ is orthogonal to V_M, while $P_{V_M} B$ is a vector in V_M by definition. If we now take the expectation **with respect to the noise**, the first term is a constant, and we need to calculate the second:

$$\underset{B}{E}[||P_{V_M} B||^2] = \sum_{k=1}^{M} \underset{B}{E}[|\langle B, g_k \rangle|^2] = M\sigma^2$$

The result shows that the risk consists of two terms: an approximation error and a fluctuation term:

$$\boxed{R = ||x - Proj_{V_M} x||^2 + M\sigma^2}$$

How can we achieve the optimum? If $M = d$, the first term is zero but the second is maximal, and vice versa. **So we choose the M that strikes a balance between the two terms**, and it's the **bias decay** that determines whether M will be smaller or larger between two linear models.

But if the signal has discontinuities, we must adapt to the signal with non-linear algorithms. So, as in the case of function approximation, we will choose the V_M space by thresholding the coefficients.

$$LZ = \sum_{k \in I_M} \langle Z, g_k \rangle g_k = \sum_{k \in I_M} \rho_T(\langle Z, g_k \rangle) g_k$$

with the threshold T defined as

$$T = \underset{1 \le k \le d}{max} |\langle B, g_k \rangle| \approx \sigma \sqrt{2 \log d}$$

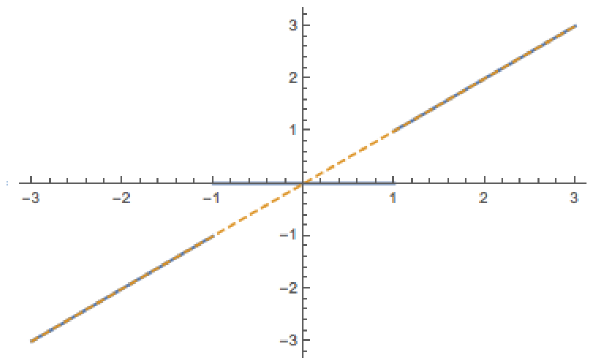

FIGURE 6 – Thresholding function $\rho_\alpha(x)$, here with $\alpha = 1$.

and the thresholding function defined as (see Figure 6):

$$\rho_\alpha(x) = \begin{cases} 0 & \text{if } |x| < \alpha \\ x & \text{if } |x| \geq \alpha \end{cases}$$

Under what condition is this thresholding effective? It will work if the representation (basis) is as sparse as possible (of course, if we knew x, we would have only one non-zero coefficient, but we don't know x!). **We want the energy of the signal to be concentrated on a small number of coefficients; it's a signal approximation problem**. For a compression problem, it's the same thing: M is the number of bits, and we want the smallest error.

6. Fourier Analysis

6.1 Recap

We started with **supervised learning**, where the answer to the question is unique, i.e., $y = f(x)$, and we train an algorithm h with n labeled samples $\{x_i, y_i\}$. In fact, we are trying to approximate f, which belongs to a family of functions with given regularity ($f \in \mathcal{C}$), using an algorithm/function in the class $\mathcal{H}$. **If we don't know anything about the probability distribution (X, Y), we have seen that if we want to control the error**

(pessimistic):

$$\underset{f\in\mathcal{C}}{sup}\ \underset{h\in\mathcal{H}}{min}||f-h||_\infty = \underset{x\in\mathbb{R}^d}{sup}|f(x-h(x))|$$

leads to the **curse of dimensionality** because $n \sim \varepsilon^{-d}$ potentially with a very large d. So, we need to **inject knowledge into the data**: signal processing and unsupervised learning. The data belongs to a subset of Ω with a much smaller dimension. We have two main ways to represent data: **the linear case and the non-linear case**. If we decompose x using an **orthonormal basis** of Ω, then:

$$x(u) = \sum_{k=1}^{d} \langle x, g_k\rangle g_k(u)$$

and: - **In the linear case**: we truncate the sum to the first M coefficients (e.g., the low frequencies in Fourier). Thus, an approximation $\tilde{x}_M$ is a **projection** of x onto a set of dimension M, and the induced error is:

$$\varepsilon_M = \sum_{k=M+1}^{d} |\langle x, g_k\rangle|^2$$

The rate of decay of coefficients will determine the value of M. The sampling of f (the positions of x_i that yield $f(x_i)$) is regularly spaced **independently of the function** f. So, if the underlying function is regular, everything is fine, but if it has singularities, then sampling introduces bias errors. - **In the non-linear case**: by setting a threshold T_M that reduces the number of coefficients to M as in the linear case, **it adapts to the signal**. In this case, the sampling **focuses on the singularities of** f and reduces the bias term.

6.2 Harmonic Analysis (Fourier)

We will revisit the Fourier transform because it appears everywhere, not only in signal processing associated with dimensionality reduction but also when we want to represent data by oscillating functions on a manifold. Here, we will consider it in one dimension and explore several themes, including **the regularity associated with sparsity**.

6.2.1 Convolution: Translation Covariant Operator

The Fourier transform is deeply linked to translation invariance: when we want to decompose an operator (e.g., the Laplacian) that respects translation invariance, we diagonalize it in the Fourier basis. Which operators commute with translation (translation covariant)?

$$L(x_\tau(u)) = L[x(u-\tau)] = (Lx)(u-\tau)$$

In the discrete case, we have:

$$\begin{aligned} Lx(u) &= L[\sum_v x(v)\delta(u-v)] = \sum_v x(v)L[\delta(u-v)] \\ &= \sum_v x(v)h(u-v) = \sum_v x(u-v)h(v) = (x*h)(u) = (h*x)(u) \end{aligned}$$

where h is the **impulse response** of L, i.e., $L\delta = h$, with δ as the Dirac operator. The first equality is the decomposition of $x(u)$ with δ, the second is due to the linearity of the operator L, the third represents the covariance of L with respect to translation, and the last equation is obtained by changing the variable. The notation "$*$" represents **convolution**.

In the continuous case, we proceed in a similar manner with the Dirac mass/distribution:

$$\int_{-\infty}^{+\infty} x(u)\delta(u)du = u(0)$$

for functions $x(u)$ with regularity properties. Here's a summary:

Analog	Discrete
$x(u), u \in \mathbb{R}$	$u \in \mathbb{Z}$
$x(u) = \int_{-\infty}^{+\infty} x(v)\delta(u-v)dv$	$x(u) = \sum_{v=-\infty}^{+\infty} x(v)\delta(u-v)$
$Lx(u) = \int x(v)h(u-v)dv = (x*h)(u)$	$Lx(u) = \sum_v x(v)h(u-v) = (x*h)(u)$

The linear operator that commutes with translation (either continuous or with integer jumps) is convolution. And this generalizes to any translation. Can we diagonalize

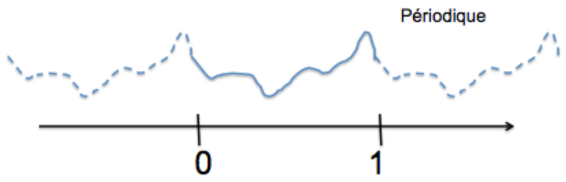

FIGURE 7 – Periodic extension of a function defined on the interval $[0, 1]$.

these convolution operators, and in which basis?

6.2.2 The Fourier Basis

Consider the function $e_\omega(u) = e^{i\omega u}$; this is an eigenvector of the convolution:

$$(e_\omega * h)(u) = \sum_v e^{i\omega(u-v)} h(v) = e^{i\omega u} \sum_v e^{-i\omega v} h(v) = e_\omega(u)\hat{h}(\omega)$$

where

$$\hat{h}(\omega) = \sum_v e^{-i\omega v} h(v)$$

is the **Fourier Transform** (FT) of h. In the continuous case, it is defined similarly [5]:

$$\hat{h}(\omega) = \int dv \; e^{-i\omega v} h(v)$$

We define the **Fourier Basis** according to the « analog » or « discrete » case, but keep in mind that we are thinking in « continuous » terms (e.g., notions of regularity) and computing in « discrete » terms (considering the sampling and as d increases, it converges to the continuous case), hence the importance of looking at both formulations.

— **Analog**:

We consider functions $L^2[0, 1]$: a set of square-integrable functions, with finite energy over the support $[0, 1]$ (this is not restrictive, and the notion of a compact support is practical). We impose periodicity as shown in Figure 7:

5. Note: there is a normalization constant that depends on the convention.

The orthonormal Fourier basis is then defined as:

$$\boxed{g_k(u) = e^{i2\pi\ ku};\ k \in \mathbb{Z}, u \in [0,1]}$$

— **Discrete**:
In this case, we take d samples and also impose periodicity over $\mathbb{Z}$ except for $0, \ldots, d-1$. The Fourier basis is defined as:

$$\boxed{g_k(u) = e^{i2\pi k\ u/d}; 0 \leq k < d; 0 \leq u < d}$$

Now, in both cases, x is projected into the g_k basis with the associated inner product. For example, in the continuous (analog) case:

$$x(u) = \sum_k \frac{\langle x, g_k \rangle}{||g_k||^2} g_k(u); \quad ||x||^2 = \sum_k |\langle x, g_k \rangle|^2$$

The Fourier coefficient is equal to:

$$\langle x, g_k \rangle = \int_0^1 x(u) e^{-i2\pi\ ku} = \hat{x}(\omega = 2\pi k) \equiv \hat{x}(k)$$

Therefore, in the continuous case (with compact support), the reconstruction/synthesis of the signal is:

$$x(u) = \sum_{\omega = 2\pi k \in \mathbb{Z}} \hat{x}(\omega)\ e^{i\omega u}$$

In the discrete case with the normalization $||g_k(u)||^2 = d$, we have:

$$x(u) = \frac{1}{d} \sum_{k=1}^{d} \hat{x}\left(\frac{2\pi k}{d}\right) e^{i2\pi \frac{ku}{d}}$$

Note that in the discrete case, we need to calculate d Fourier coefficients for d components of x, so d^2 operations are required. Fortunately, we have found **algorithms (FFT) that only require $d \log d$ operations**. This is a fundamental point that is also relevant in data science, where mathematics and algorithms are closely linked.

The decomposition of any function with a compact support (finite energy) into sinusoids is not intuitive at all. For example, one must imagine decomposing the blue function in Figure 8 by weighting sinusoids, the first six of which are drawn. The very rapid

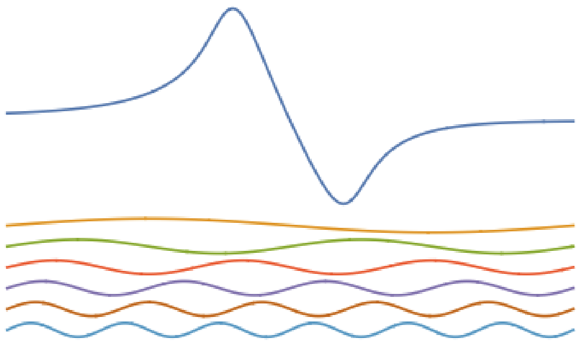

FIGURE 8 – Decomposing a function into sinusoids.

local variability involves a high-frequency sinusoid, but elsewhere, the rapid oscillations must be compensated exactly to leave room for a smoothly varying function represented by low-frequency sinusoids. While this can be seen in the mathematics, it is not intuitively obvious at first glance.

6.2.3 Filtering/Convolution and the Approximation-Regularity Tandem

The coefficients $\hat{x}(\omega)$ represent the different "weights" of weighting to reconstruct the signal, and thus **the decay of the Fourier coefficients is linked to the regularity of the function**. We want to keep only M coefficients and have a small approximation error. With $\int |x(u)|du < \infty$, which also holds for h, we have:

$$\boxed{z(u) = (x * h)(u) \Leftrightarrow \hat{z}(\omega) = \hat{x}(\omega)\hat{h}(\omega)}$$

Recall that convolution $(x * h)$ is nothing but the application of the linear translation-covariant operator L, i.e., $z = Lx$. The equivalence above is easily demonstrated by decomposing x on the Fourier basis and applying L to the basis vectors.

Filtering and Convolution are identical concepts, and the terminology changes depending on the field. For example, when we want to remove high frequencies to smooth a signal, we use a **low-pass filter**:

$$\hat{h}_M(\omega) = \begin{cases} 1 & -\frac{M}{2} \leq \omega \leq \frac{M}{2} \\ 0 & \text{elsewhere} \end{cases}$$

In the real space, this corresponds to a convolution with $h_M(u)$, which is a **cardinal sine** (periodized on $[0, 1]$).

So, in the context of function approximation, we want to project the signal x onto an orthonormal basis (here, the Fourier basis) and keep only the first M coefficients (i.e., filter out high frequencies). We define the projector onto the space V_M (a subspace of Ω) using convolution with the low-pass filter:

$$Proj_{V_M} x = x * h_M$$

We know that if the decay of the Fourier coefficients of x is rapid enough, then convolution with h_M will be a good approximation. And this decay is linked to the regularity of x. What exactly is the relationship? The derivative of $x(u)$, denoted $x'(u)$, commutes with translation, so it's easy to show (by integration by parts) that:

$$\widehat{x'}(\omega) = i\omega \ \hat{x}(\omega)$$

and for the k-th derivative:

$$\widehat{x^{(k)}}(\omega) = (i\omega)^k \ \hat{x}(\omega)$$

A **notion of regularity** can be introduced by the fact that the **derivatives of the function do not explode**, i.e.,

$$||x^{(q)}||^2 = \int_0^1 \left| \frac{d^q x}{du^q} \right|^2 du < \infty$$

This translates (via Plancherel) in the Fourier domain to the fact that:

$$\sum_{\omega = 2\pi k \in \mathbb{Z}} |\omega|^{2q} |\hat{x}(\omega)|^2 < \infty$$

For this to be true, it's necessary to impose a decay of $\hat{x}(\omega)$ as a function of ω (the small "o(a)" notation means negligible compared to "a"):

$$|\hat{x}(\omega)| = o(|\omega|^{-q}) = o(|k|^{-q});\ \omega = 2\pi k$$

and q can be a real number $\alpha \in \mathbb{R}$. These functions belong to Sobolev spaces of α-times differentiable functions. **So, the decay of the Fourier coefficients is linked to the notion of regularity through the concept of Sobolev differentiability.**

From the approximation theorem, we remember that if the coefficients in a basis ($\alpha > 1/2$) satisfy:

$$|\langle x, g_k \rangle| = O(k^{-\alpha}) \text{ then } \varepsilon_M^2 = ||x - x_M||^2 = O(M^{1-2\alpha})$$

This translates (in the continuous case with $g_k = e^{i2\pi ku}$) to the fact that the Fourier coefficients of a function α times differentiable in the Sobolev sense satisfy:

$$|\langle x, g_k \rangle| = |\hat{x}(\omega)| = o(|k|^{-\alpha})$$

So, as soon as we have regularity, we have possible approximation, and vice versa. However, as soon as there is a discontinuity, Sobolev regularity is not sufficient, and filtering in the Fourier basis introduces errors (bias), requiring a change in the type of regularity.

Sampling Theorem: Let the set of functions whose high-frequency Fourier coefficients are zero be denoted as:

$$V_M = \left\{ x/\text{support of } \hat{h}(\omega) \in [-M/2, M/2] \right\}$$

The projection of x onto V_M is such that:

$$x_M = Proj_{V_M} x = x * h_M; \quad \hat{h}(\omega) = \mathbb{I}_{[-M/2,M/2]}(\omega)$$

Assuming $x \in L^2[\mathbb{R}]$ (this is just to simplify the expression of h, otherwise, you need to periodicize the cardinal sine), then:

$$h_M(u) = \frac{\sin \pi x/T}{\pi x/T}; \quad \text{with} \quad T = 2\pi/M$$

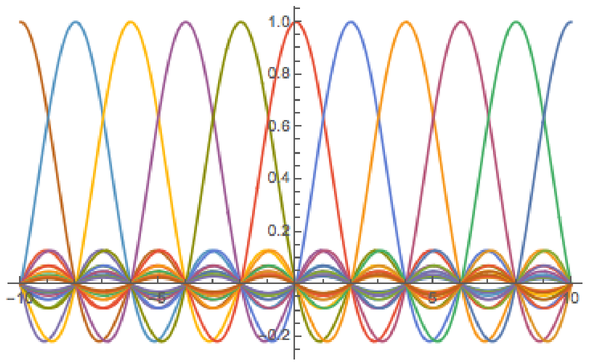

FIGURE 9 – Series of translated cardinal sines $h_{M,nT}(u)$, $T = 2\pi/M$.

So, if $h_M(u - nT) = h_{M,nT}(u)$ with $n \in \mathbb{Z}$, then its Fourier coefficients are:

$$\widehat{h_{M,nT}}(\omega) = \hat{h}_M(\omega)e^{-i2\pi\ n\ (\omega/M)}$$

where $\hat{h}_M(\omega) = 1$ on the support of ω in $[-M/2, M/2]$. Therefore, we have a new orthonormal basis (see Poisson's formula to show that any function in V_M can be decomposed into a sum of $h_{M,nT}(u)$):

$$\{h_M(u - nT)\}_{n\in\mathbb{Z}}$$

So, for all $x \in V_M$, we obtain the decomposition:

$$\boxed{\begin{aligned} x(u) &= \sum_{n\in\mathbb{Z}} c_n h_M(u - nT) \\ &= \sum_{n\in\mathbb{Z}} x(nT) h_M(u - nT) \end{aligned}}$$

The second equality comes from the fact that $h_M(mT - nT) = \delta_{m,n}$.

Thus, from the samples $x(nT)$, we can reconstruct the signal $x(u)$ using interpolations given by $h_M(u - nT)$. For example, with $T = 2$ and $n = -10, -9, \ldots, 9, 10$, the $h_M(u - nT)$ functions appear as translated cardinal sines (see Figure 9). The smaller T gets, the narrower the cardinal sines become, and the approximation of $x(u)$ becomes better, as shown in the example in Figure 10, where $T = 2$ is in yellow and $T = 1$ is in green, while $x(u)$ is in blue.

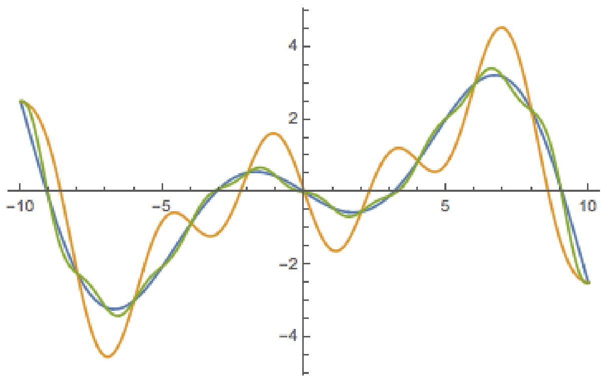

FIGURE 10 – Approximation of the function $x(u)$ (blue) using two functions obtained from sampling with $T = 2$ in yellow and $T = 1$ in green.

So, for uniformly regular functions, there's no problem. However, **as soon as there is a discontinuity, we cannot benefit (in terms of the number of samples) from continuity outside the discontinuity**. This is the major limitation of Fourier analysis. We will explore other types of regularities that adapt to the signal.

6.2.4 Interpolation and Sampling Theorem

The Sampling Theorem is more general; you should think of it in terms of the associated interpolation basis. Let $h(t) = 1$ over the support $[-T/2, T/2]$, and consider the vector space generated by the collection of translated h:

$$V_T = \text{Vect}\left\{h(t - nT)\right\}_{n\in\mathbb{Z}}$$

The Sampling Theorem associated with h then tells us:

$$x \in V_T \Leftrightarrow x(t) = \sum_n x(nT)h(t - nT)$$

We see that V_T is the space of **piecewise constant functions** of size T.

But we can do better! If we take the "triangle" function $h(t)$ from Figure 11 with support $[-T, T]$ and the value at 0 equal to 1, then V_T is the space of **linear splines** (or piecewise affine) functions that give approximations as shown in Figure 12.

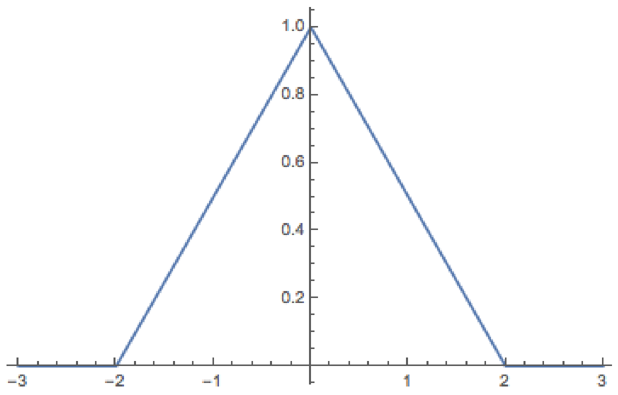

Figure 11 – Triangle function.

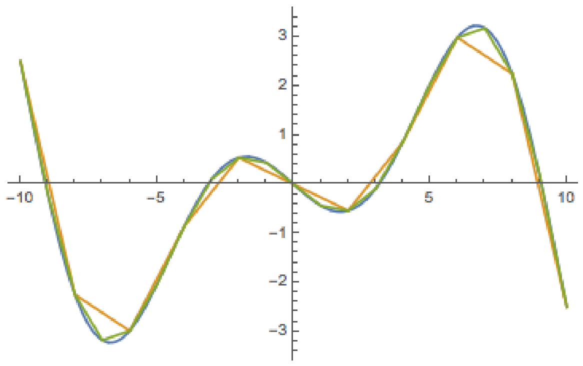

Figure 12 – Approximation of a function $f(x)$ (blue) by piecewise affine functions from the Sampling Theorem with the triangle function.

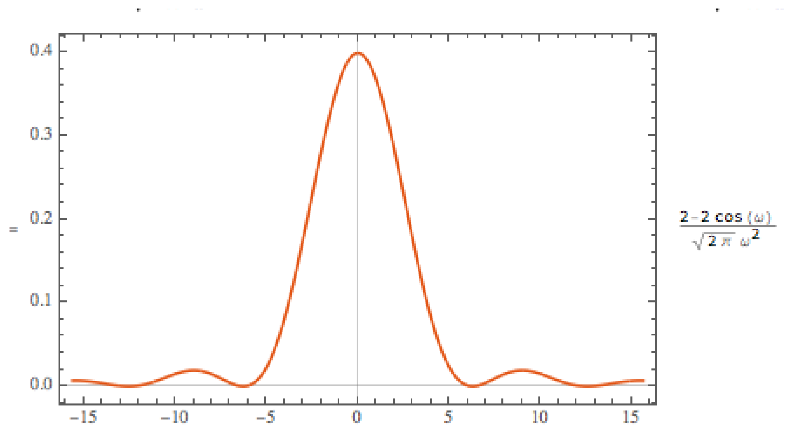

FIGURE 13 – Filter associated with the triangle function.

We can calculate the associated filter, which gives the function in Figure 13. So, we can obtain fine approximations of regular functions, but it's more general than that because we want to adapt to the function's regularity (or its discontinuities) and thus obtain a **Sampling Theorem in the nonlinear case: this is Wavelet analysis.**

6.2.5 Conventions for the Fourier Transform (NDJE)

Before concluding this chapter, let's address a practical point, which is the conventions for writing. Indeed, both in the literature and in the use of computer libraries, it's always good to know which convention is used. In Table 4, the Fourier Transform (FT) depends on 2 parameters (m, q) that influence the writing of the formulas.

7. Wavelet Analysis

Recall that in the context of learning, we aim to perform dimensionality reduction to understand the regularity patterns of the signal. In the case of Fourier analysis, by truncating the representation to the first M coefficients, we implicitly introduce a uniform (local) regularity. However, this approach is not suitable when the signal is "piecewise" regular with singularities at junctions.

FT	$\hat{f}(\omega) = \frac{1}{\sqrt{m}} \int f(x) e^{iq\omega x} dx$
Inverse FT	$f(x) = \frac{\sqrt{m}}{2\pi} \int \hat{f}(\omega) e^{-iq\omega x} dx$
Translation	$\widehat{f(x-h)}(\omega) = e^{iqwh} \hat{f}(\omega)$
Dilation	$\widehat{\frac{1}{s} f(\frac{x}{s})}(\omega) = \hat{f}(s\omega)$
Mean	$\hat{f}(0) = \frac{1}{\sqrt{m}} \int f(x) \mathrm{d}x$
Plancherel	$\int f(x) g^*(x) \mathrm{d}x = \frac{m}{2\pi} \int \hat{f}(\omega) \hat{g}^*(\omega) \mathrm{d}\omega$
Parseval	$\int \lvert f(x) \rvert^2 \mathrm{d}x = \frac{m}{2\pi} \int \lvert \hat{f}(\omega) \rvert^2 \mathrm{d}\omega$
Derivative	$\widehat{f^{(p)}(x)}(\omega) = (-iq\omega)^p \hat{f}(\omega)$
Convolution	$\widehat{f * g} = \sqrt{m} \hat{f}(\omega) \hat{g}(\omega)$
Multiplication by a phase	$\widehat{e^{i\xi x} f(x)}(\omega) = \hat{f}(\xi/q + \omega)$

Table 4 – Examples of properties of the Fourier transformation expressed according to a generic formulation in 1D. S. Mallat uses ($m = 1, q = -1$) which corresponds to the convention ($a = 1, b = -1$) in Mathematica; but in the literature, you can find formulations with $m = 1, 2\pi$ and $q = \pm 1, \pm 2\pi$.

It is in this extended context that **Wavelets** are introduced, with support localized in both time and frequency. Dennis Gabor (Nobel Prize for Holography, 1971) introduced the concept of coherent states in Quantum Mechanics in accordance with the uncertainty principle $\Delta x \Delta p \geq \hbar$. Gabor later applied this concept to sound processing, where time-frequency locality is associated with the notion of musical notes (Gabor spectrogram). This type of representation generalizes across various fields.

The fundamental idea of a wavelet is that it is a kind of "localized sine wave", denoted as ψ. Locality is expressed through the condition

$$\boxed{\int_{-\infty}^{\infty} \psi(u) du = 0}$$

To adapt to functions with varying degrees of regularity, two operations are performed:
— **Translation** $(b \in \mathbb{R})$
— **Dilation** $(s \in \mathbb{R}^{+*})$
applied to the base wavelet $\psi(u)$ to create the family

$$\boxed{\psi_{s,b}(u) = \frac{1}{\sqrt{s}} \psi\left(\frac{u-b}{s}\right)}$$

Figure 14 illustrates the effect of scaling on a wavelet: at the top: $s = 1$, in the middle: $s = 2$ (low frequency), at the bottom: $s = 1/2$ (high frequency) [6].

7.1 Wavelet Transform

For a signal $x(u)$, its wavelet transform is expressed as

$$Wx(s,u) = \langle x, \psi_{s,u} \rangle = \int dv\ x(v) \frac{1}{\sqrt{s}} \psi^*\left(\frac{v-u}{s}\right) \equiv \int x(v) \bar{\psi}_s(u-v) dv$$

Here, 1) we assume that ψ **is real**, and 2) we define $\bar{\psi}$ as follows:

$$\boxed{\bar{\psi}_s(u) \equiv \frac{1}{\sqrt{s}} \psi\left(-\frac{u}{s}\right)}$$

6. Note: The representation on the board by S. Mallat is a bit misleading; it appears to depict $\phi(x)$, also known as the "scaling function", centered at $x = 0$, rather than $\psi(x)$.

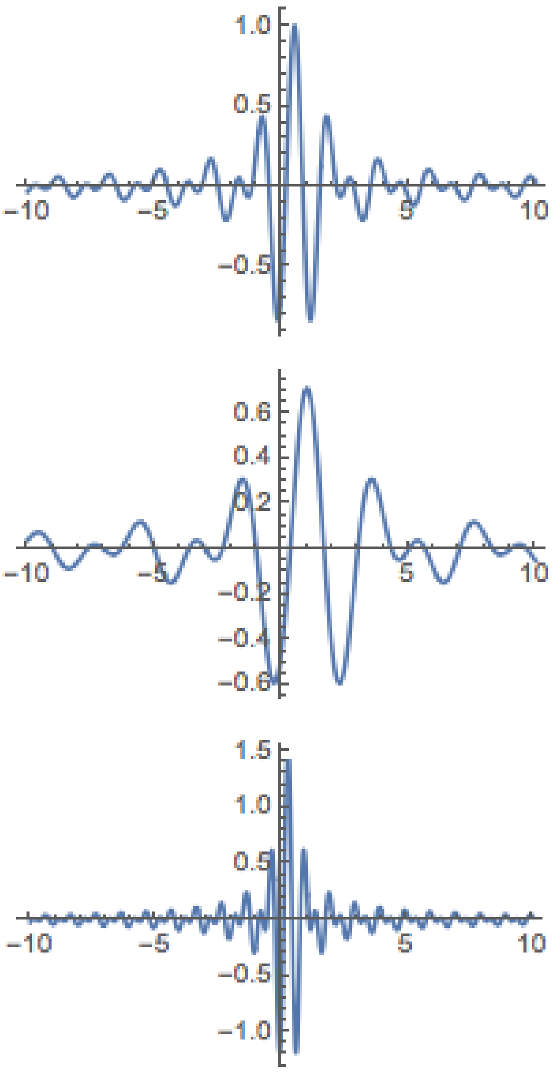

FIGURE 14 – (Top): Base wavelet; (Middle): $s = 2$ low frequency; (Bottom): $s = 1/2$ high frequency.

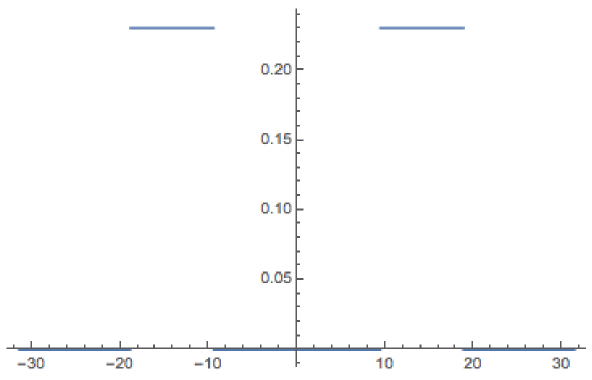

FIGURE 15 – Typical filter of a wavelet ψ.

So, we see that the **Wavelet Transform is a convolution**:

$$\boxed{Wx(s,u) = (x * \bar{\psi}_s)(u)}$$

This is effectively a **filtering** operation by $\bar{\psi}_s$.

The Fourier transform of the wavelet is given by

$$\hat{\psi}(\omega) = \int \psi(t)e^{-i\omega t}dt$$

Since, by definition, $\hat{\psi}(0) = 0$ and the wavelet is real, we have

$$\hat{\bar{\psi}}_s(\omega) = \sqrt{s}\hat{\psi}^*(s\omega)$$

The frequency support is of a **"bandpass" type**, as shown in the spectrogram $|\hat{\psi}(\omega)|$ of the wavelet in Figure 15 (being real, the spectrogram is symmetric about 0). If $s > 1$, the spectrogram shifts towards *low frequencies*, and conversely, if $s < 1$, it shifts towards *high frequencies*, consistent with the scaling of the wavelet with respect to s.

For wavelets, the support satisfies a Heisenberg-like uncertainty relation, i.e.,

$$Area \geq Const = 1/4$$

This support deforms along the frequency axis as schematically shown in Figure 16 when

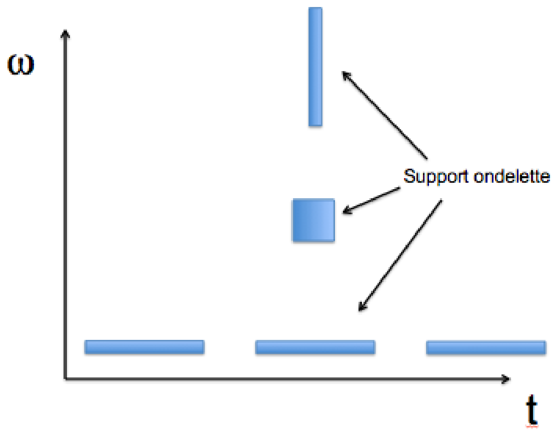

FIGURE 16 – Evolution of wavelet supports in relation to frequency and time. The area satisfies a Heisenberg-like uncertainty relation.

the wavelet is dilated; however, its shape remains unchanged when translated along the time axis.

7.2 Wavelets and Function Regularity/Singularity

Let's revisit the notion of Lipschitz-α regularity as follows:

$$|x(v) - p_u(v)| \leq C_u |v - u|^\alpha \ \forall v \in \mathbb{R}$$

Here, $p_u(v)$ is a polynomial in the vicinity of u of degree $q = \lfloor \alpha \rfloor$ (the largest integer less than or equal to $\alpha \geq 0$). If $\alpha = 0$, the function is bounded; if $\alpha \geq 1$, it is differentiable, and for $0 < \alpha < 1$, the function may exhibit a "pinch." All of this can be schematically represented as shown in Figure 17.

Proposition: If x is Lipschitz-α at u, then the wavelet coefficient (cf. inner product) satisfies the relation

$$\boxed{|\langle x, \psi_{s,u} \rangle| \leq C_u \ \beta \ s^{\alpha + \frac{1}{2}}}$$

if ψ is a wavelet with zero moments[7], meaning it oscillates slightly to *avoid seeing poly-*

7. Note: Ingrid Daubechies' wavelets have this property

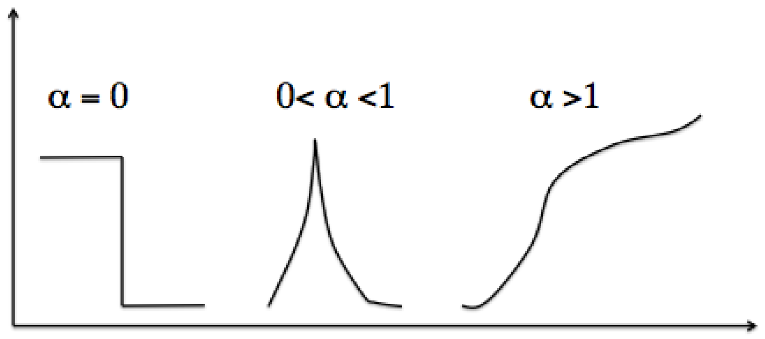

FIGURE 17 – Types of Lipschitz-α functions.

nomial regularity:

$$\int \psi(t)t^k dt = 0; \quad k \leq q$$

(this property holds for the scaled and translated versions as well).

The proposition states that if s approaches 0 (small support) and α is close to 1 (differentiable function), then the wavelet coefficient tends to 0. **So, the number of coefficients decreases as s approaches 0**, as illustrated in Figure 18. During the proof, we find the constant β which depends only on the wavelet:

$$\beta = \int |y|^\alpha |\psi(y)| dy$$

Therefore, the Lipschitz-α regularity of the function is "imprinted" in the decay of wavelet coefficients as a function of the scale change, following $s^{\alpha+1/2}$.

7.3 Orthonormal Wavelet Basis

In the 1940s, D. Gabor had found a family of functions based on the form:

$$\psi_\nu^{Gabor}(t) \propto e^{-x^2/2} e^{i\nu x}$$

These are "packets" of waves. Jean Morlet, an engineer at ELF Aquitaine, who was originally responsible for the term "wavelet", actually used the same family (only the real part)

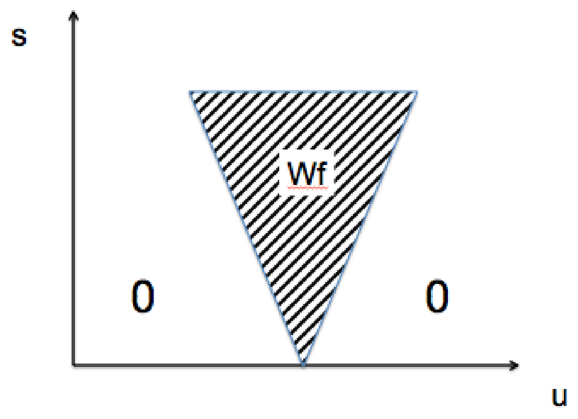

Figure 18 – The shaded area schematically indicates the evolution of the support of non-zero wavelet coefficients as a function of the scale s.

with $\nu = \sqrt{2/\log 2}$. However, these functions $\psi_{s,b}$ did not form an orthonormal basis. It was only in the 1980s that **Yves Meyer** demonstrated the existence of such orthonormal bases, against all expectations. This gave a new impetus to signal processing, and the use of wavelets diversified in various mathematical fields for classifying "regularities".

To construct an orthonormal basis, we will first discretize the scales. Indeed, we don't need all the wavelet coefficients, as can be seen in Figure 16 where the wavelet support extends in s, and there is redundancy in taking s continuous. **It is sufficient to take dyadic scales:** 2^j. In sound processing, intermediate values are used.

So, consider the family:

$$\boxed{\psi_j(u) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{u}{2^j}\right) \quad \forall j \in \mathbb{Z}}$$

The question then becomes: does knowing the wavelet coefficients for all j suffice to reconstruct the signal? Obtaining these coefficients involves performing convolutions:

$$Wx_j(u) \equiv \langle x, \psi_j \rangle(u) = (x * \overline{\psi_j})(u)$$

In fact, the question boils down to whether we can reconstruct the Fourier transform

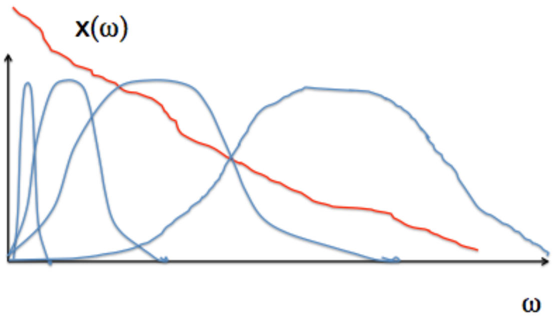

FIGURE 19 – Typical decay of the Fourier transform $\hat{x}(\omega)$ compared to wavelet support.

of the signal x from the Fourier transform of the wavelet coefficients. Note the property:

$$\widehat{\overline{\psi_j}}(\omega) = \sqrt{2^j}\hat{\psi}^*(2^j\omega)$$

So,

$$\widehat{Wx_j}(\omega) = \hat{x}(\omega)\widehat{\overline{\psi_j}}(\omega) = \sqrt{2^j}\ \hat{x}(\omega)\ \hat{\psi}^*(2^j\omega)$$

The product $\hat{x}(\omega)\ \hat{\psi}^*(2^j\omega)$ can be schematically illustrated for some values of j as shown in Figure 19. Therefore, to recover $\hat{x}(\omega)$, there must be a wavelet for every ω that yields a non-zero $\hat{\psi}^*(2^j\omega)$. In other words, there must be some overlap.

We impose an additional condition (Littlewood-Paley) of "power conservation" (*known since the 1930s with the Fourier transform in an attempt to achieve localization*):

$$\boxed{\forall\omega,\ \sum_{j\in\mathbb{Z}} |\hat{\psi}(2^j\omega)|^2 = 1}$$

With this property, we will show that $x(u)$ decomposes as:

$$\boxed{x(u) = \sum_{j\in\mathbb{Z}} 2^{-j}(Wx_j * \psi_j)(u) = \sum_{j\in\mathbb{Z}} 2^{-j}[(x * \overline{\psi_j}) * \psi_j](u)}$$

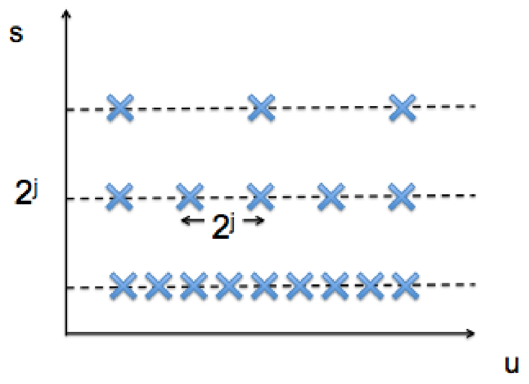

FIGURE 20 – Optimal sampling by dyadic discretization of the (s, u) space.

In Fourier terms, this means:

$$\begin{aligned}\hat{x}(\omega) &= \sum_j 2^{-j} \widehat{Wx_j}(\omega) \widehat{\psi_j}(\omega) \\ &= \sum_j 2^{-j} \sqrt{2^j}\ \hat{x}(\omega)\ \hat{\psi}^*(2^j\omega) \sqrt{2^j}\ \hat{\psi}(2^j\omega) \\ &= \hat{x}(\omega) \sum_j |\hat{\psi}(2^j\omega)|^2\end{aligned}$$

which is true according to the power conservation imposed above. It can also be shown that the energy of x is conserved:

$$||x||^2 = \sum_{j\in\mathbb{Z}} 2^{-j} ||Wx_j||^2$$

Now, regarding translation in u, we must also avoid leaving gaps. Referring to Figure 16, we can see that for low frequencies (small dilations: 2^{-j} with $j > 0$), the support is wide, so we don't need a fine sampling. Conversely, at high frequencies, the support narrows, so the sampling in u must be finer. We then define for $s = 2^j$, a sampling $u_n = n2^j$. The scheme in Figure 20 summarizes the dyadic sampling.

The complete family [8] of wavelets is therefore written using the base wavelet ψ:

$$\boxed{\left\{\psi_{j,n}(u) = \psi_{2^j,2^jn}(u) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{u-2^jn}{2^j}\right)\right\}_{j,n\in\mathbb{Z}}}$$

It turns out that A. Haar in 1910 found an orthonormal basis of $L^2[0,1]$ (square-integrable functions on $[0,1]$) defined as $\psi_{j,n}(u)$ from the function ψ shown in Figure 21. (To this collection of functions $\psi_{j,n}$, one must add the unit function 1). The Fourier transform of ψ is quite oscillatory (see Figure 22) due to the singularities at 0, 1/2, and 1 [9]. However, it is quite easy to show that the $\psi_{j,n}^{Haar}$ are orthogonal because either their supports are disjoint, or one support is in a constant part of the other, resulting in trivially zero inner products. It can be shown that for all (n,j) such that $0 \leq 2^jn \leq 1$ (i.e., $j \leq 0$ and $0 \leq n < 2^{-j}$), then:

$$||x(u) - \sum_{j=-J}^{0}\sum_{n=0}^{2^{-j}-1}\langle x, \psi_{j,n}\rangle\psi_{j,n}|| \underset{J\to\infty}{\longrightarrow} 0$$

(Note: x differs by a constant). The approximation of the function $x(u)$ is of the "piecewise constant" type up to the scale $1/2^J$.

We would like a better approximation, but it was believed that it was not possible at the same time to have an orthonormal basis of wavelets with much better regularity.

7.4 Wavelet vs. Filter

Let's take the Haar basis and a discretized signal. Calculating wavelet coefficients at scale 2^j in this basis amounts to taking the average over two consecutive intervals each of size 1/2 smaller and then averaging them. So, schematically, at step j, we proceed as shown in Figure 23. In the next step, we iterate with the obtained positions. We have a

8. Note: Just an aside. There is potentially an issue: what about the continuous (very low frequency) component of x? With only $\psi_{j,n}$, one would need an infinite number of wavelets with scale 2^{-j} for $j \to \infty$. S. Mallat introduced the "scaling function" ϕ (which can also be used to define ψ) which is nothing but a low-pass filter. So, one can reconstruct a signal with a finite series of dilated and translated ϕ and an infinite series of $\psi_{j,n}$. See the section at the end of this course.

9. Note: The Haar wavelet is a special case of the Daubechies wavelets denoted **Db1**

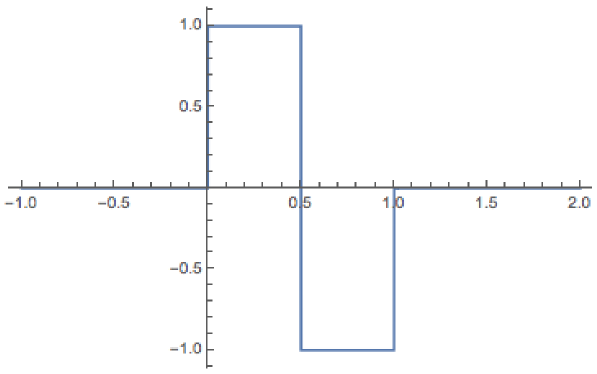

Figure 21 – Haar wavelet $\psi(x)$.

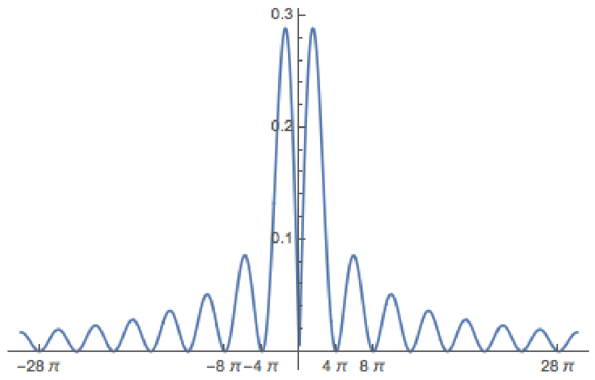

Figure 22 – Filter associated with the Haar wavelet ψ.

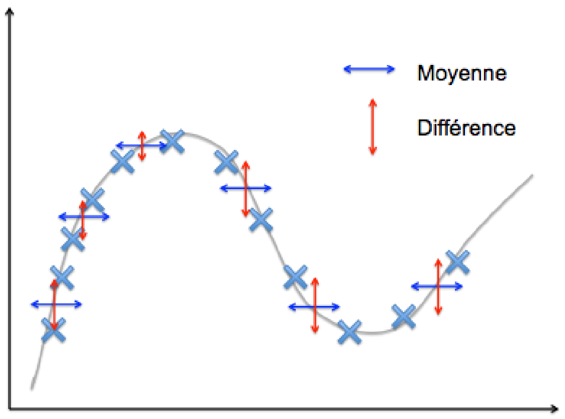

FIGURE 23 – Operation of the wavelet transform performed by the Haar wavelet.

cascade algorithm of the form:

$$(a,b) \rightarrow \left(\frac{a+b}{\sqrt{2}}, \frac{a-b}{\sqrt{2}}\right)$$

This naturally takes the form of a cascade of 2 filters (Figure 24), one being **low-pass** (the average in the case of Haar) and the other **high-pass** (the difference in the case of Haar). In fact, Mallat & Meyer showed that it is possible to completely characterize and construct wavelets from the **knowledge of these two types of filters**. Moreover, they showed that it is possible to construct a fast algorithm for calculating the wavelet transform [10].

Now, **neural networks** (deep neural networks, DNN) are ultimately cascades of filters, and learning involves **"learning" the filters**. So, one can focus on the filters for technical implementation aspects, but to understand the result, we want to know the **equivalent filter of the entire cascade**. **However, the significant difference between a cascade of filters and a DNN lies in the nonlinearities introduced in the neuron responses. This makes it much more complicated.**

10. Note: See, for example, http://cas.ensmp.fr/~chaplais/Wavetour_presentation/ondelettes/Biortho_Wave_and_filt.html

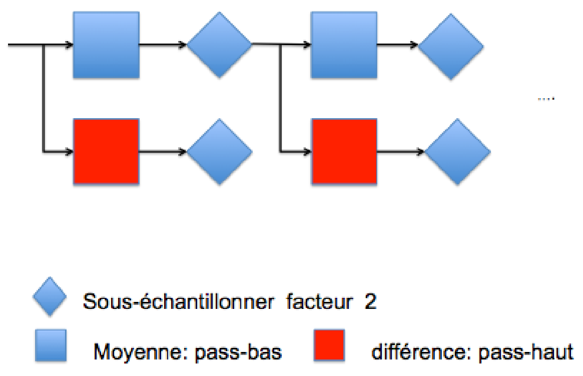

FIGURE 24 – Cascade of low-pass and high-pass filters involved in a Haar wavelet transform.

7.5 Sparsity

Consider a Lipschitz-α function. We know that the decay of wavelet coefficients will occur at $2^{-j(\alpha+1/2)}$ $(j > 0)$ as shown in Figure 25. The non-zero coefficients (above a threshold) will concentrate at the singularities of the function. The wavelet's support is 2^{-j} (for large j), and it shifts by this amount in translation. So, as long as the function is fairly uniform over the wavelet's support, the inner product is zero. The two regions of large variations in the function (singularities) will only affect a small number of coefficients. At a large scale (j small), remember that the wavelet has a large support, and the sampling is also large, so we only keep a few samples.

When we keep only the samples above a threshold, the sampling is adaptive (hence nonlinear), and the Gibbs phenomenon that affected the Fourier transform when we only kept the first M terms disappears.

The course concludes with some illustrations in 1D and 2D cases. Before concluding this chapter, I will allow myself to make a few additions.

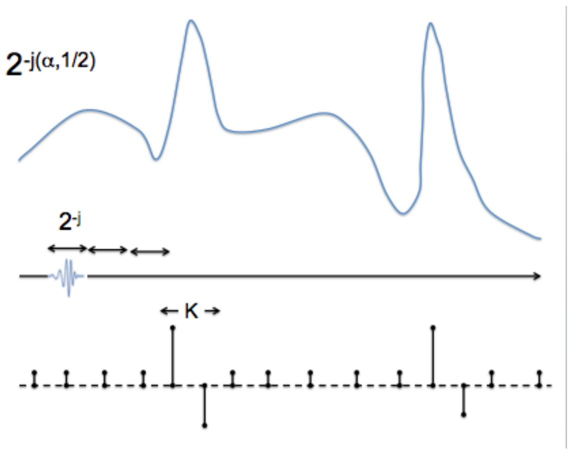

FIGURE 25 – Estimation of the location of significant wavelet coefficients near the singularities of the function.

7.6 Some Comments (NDJE)

In the course, S. Mallat introduced the wavelet ψ with bounded frequency support. It is well-suited for capturing signal discontinuities, which are more related to "high frequency" and thus associated with the "high-pass" filter. However, the signal also has regular components, which correspond to its "low-frequency" part. Therefore, a "low-pass" filter is needed to capture it. S. Mallat briefly mentioned this in the context of the cascade algorithm, alluding to these two "low-pass" filters for regularities and "high-pass" filters for discontinuities.

I intend to summarize (without proofs) some properties of the Multi-Resolution Analysis (MRA) developed by S. Mallat in the years 1987-89[11]. It is highly likely that in subsequent courses, he may delve into this topic further. Therefore, consider the following as supplementary information for the curious.

7.6.0.1 *The « scaling function » ϕ and its low-pass filter*

We consider functions with square (energy) summability over $\mathbb{R}$ (cf. $L^2(\mathbb{R})$). Let's examine a nested ensemble cascade of the form ($n \in \mathbb{Z}$):

11. See reference Mallat, S.G., 1989, "A theory for multiresolution signal decomposition: the wavelet representation, IEEE Trans. PAMI 11, 674-693."

$$\cdots \subset V_{-n} \subset \cdots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset V_n \subset \ldots$$

The intersection of V_i reduces to the function 0, while their union fills $L^2(\mathbb{R})$. If we denote that V_j is associated with an introspection scale s_j, then

$$\boxed{V_{j_1} \subset V_{j_2} \Leftrightarrow s_{j_2} < s_{j_1}}$$

(Note: The sign change of the index may appear in the literature, depending on the definition of the dilation operator in the following).

The "dyadic" choice for the scale associated with V_j has already been motivated. In fact, a function from V_0 and its "twin" version in V_j are related as:

$$f(2^j x) \in V_j \Leftrightarrow f(x) \in V_0$$

There exists a function $\phi \in V_0$ whose translations by integer steps cover the set V_0, meaning:

$$\forall f \in V_0, f(x) = \sum_{k\in\mathbb{Z}} c_k T_k[\phi](x) = \sum_{k\in\mathbb{Z}} c_k \phi(x-k)$$

And the $\{T_k[\phi], k \in \mathbb{Z}\}$ form an orthonormal basis for V_0. It can be shown that the integral of ϕ satisfies the relation:

$$\int_{-\infty}^{\infty} \phi(x)dx = 1$$

7.6.0.2 ***The « wavelet function » ψ and its high-pass filter: relation with ϕ***

In the course, S. Mallat shows that from a wavelet ψ, one can construct an orthonormal basis for $L^2(\mathbb{R})$ through dyadic dilation and translation (*Note: Be aware that here, the sign of j is opposite to that in the course*):

$$\left\{\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k) = D_j T_k[\psi](x), j, k \in \mathbb{Z}\right\}$$

For a fixed $j = j_0$, the family $\{D_{j_0}T_k[\psi](x), k \in \mathbb{Z}\}$ is **an orthonormal basis for the space** $W_{j_0} = V_{j_0+1} \ominus V_{j_0}$; **meaning that** V_{j_0+1} **is the orthogonal sum of** V_{j_0} **and** W_{j_0}**.**

In the case $j_0 = 0$, with the wavelet ψ being a member of V_1, we have a scaling relation similar to ϕ:

$$\boxed{\psi(x) = \sum_{k \in \mathbb{Z}} g_k \sqrt{2} \phi(2x - k)}$$

where the coefficients $\{g_k, k \in \mathbb{Z}\}$ define the discrete filter associated with ψ. Similar to m_0, we define the transfer function m_1 as:

$$\boxed{m_1(\omega) = \frac{1}{\sqrt{2}} G(\omega) = \frac{1}{\sqrt{2}} \sum_{k \in \mathbb{Z}} g_k e^{-i\omega k}}$$

Just as for ϕ, it can be shown that the Fourier transform of ψ is subject to the relation:

$$\boxed{\hat{\psi}(\omega) = m_1(\omega/2)\hat{\phi}(\omega/2)}$$

7.6.0.3 *The Daubechies Wavelets*

In the course, S. Mallat introduces Haar wavelets and provides arguments for using wavelets with the first N moments equal to zero, especially in relation to the rapid decay of wavelet coefficients for a Lipschitz-α signal x.

I. Daubechies (1992) invented a type of wavelet ψ with the first N moments equal to zero and the property that:

$$|\phi(x)| \leq C_2(1 + |x|)^{-\alpha}; \quad \alpha > N$$

Then, it can be shown that:

$$m_0(\omega) = \left(\frac{1 + e^{-i\omega}}{2}\right)^N \times \mathcal{P}(\omega)$$

where $\mathcal{P}(\omega)$ is a trigonometric polynomial.

The relations between m_1 and m_0 given in the previous paragraph yield:

$$|m_0(\omega)|^2 + |m_0(\omega+\pi)|^2 = 1$$

If we define:

$$|\mathcal{P}(\omega)|^2 \equiv P_0(y); \quad y \equiv \sin^2 \frac{\omega}{2}$$

with P_0 being a polynomial, then:

$$(1-y)^N P_0(y) + y^N P_0(1-y) = 1$$

By using Bezout's theorem since y^N and $(1-y)^N$ are coprime, it can be shown that:

$$P_0(y) = \sum_{k=0}^{N-1} \begin{pmatrix} N+k-1 \\ k \end{pmatrix} y^k$$

Then, using a theorem from Fejér-Riesz, we can transition from P_0 to $\mathcal{P}$. In short, we can reformulate P_0 in terms of a polynomial in $\cos\omega$, and then in $z = e^{i\omega}$, as $\cos\omega = (z+1/z)/2$:

$$P_0(\cos\omega) = \frac{a_0}{2} + \sum_{k=1}^{N-1} a_k \cos^k \omega = \sum_{j=-(N-1)}^{N-1} c_j z^j = P_0(z)$$

It can then be shown that:

$$P_0(z) = c \prod_{j=1} r(z-\alpha_j)(1/z - \alpha_j^*)$$

with $c > 0$, and $|\alpha_j| \geq 1$, which leads to:

$$\mathcal{P}(\omega) = \sqrt{c}\prod_{j=1}^{r}(e^{i\omega} - \alpha_j)$$

Finally:

$$\boxed{m_0(\omega) = \sqrt{c}\left(\frac{1+e^{-i\omega}}{2}\right)^N \prod_{j=1}^{r}(e^{i\omega} - \alpha_j)}$$

Let's consider an example with $N = 2$ concerning the wavelet denoted as **Db2** (or DAUB4) in the literature. So,

$$\begin{aligned} P_0(y) &= 1 + 2y \\ &= 2 - \cos\omega \\ &= \frac{1}{2z}(-z^2 + 4z - 1) \\ &= \frac{2-\sqrt{3}}{2}(z - (2+\sqrt{3}))(1/z - (2+\sqrt{3})) \end{aligned}$$

This leads to

$$\mathcal{P}(\omega) = \frac{\sqrt{2-\sqrt{3}}}{\sqrt{2}}(e^{i\omega} - (2+\sqrt{3}))$$

Since $\sqrt{2-\sqrt{3}} = (\sqrt{3}-1)/\sqrt{2}$, finally, m_0 is given by

$$\boxed{m_0(\omega) = \frac{1}{8}\left(1+\sqrt{3} + (3+\sqrt{3})e^{-i\omega} + (3-\sqrt{3})e^{-i2\omega} + (1-\sqrt{3})e^{-i3\omega}\right)}$$

The graphs of $|m_0(\omega)|^2$ (low-pass) and $|m_1(\omega)|^2$ (high-pass) corresponding to this are shown in Figure 26.

From the development of m_0, we can extract the elements of the discrete low-pass filter h_k and the high-pass filter g_k:

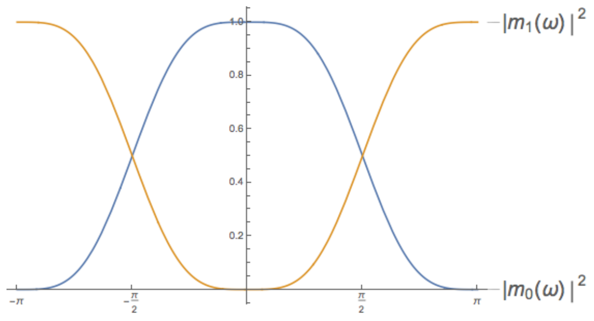

FIGURE 26 – High-pass filters (m_1) of ψ and low-pass filters (m_0 of ϕ for the Daubechies wavelet "Db2".

$$\begin{array}{rcrcccr} h_0 & = & -g_1 & = & \frac{1+\sqrt{3}}{4\sqrt{2}} & \approx & 0.482963 \\ h_1 & = & g_0 & = & \frac{3+\sqrt{3}}{4\sqrt{2}} & \approx & 0.836516 \\ h_2 & = & -g_{-1} & = & \frac{3-\sqrt{3}}{4\sqrt{2}} & \approx & 0.224144 \\ h_3 & = & g_{-2} & = & \frac{1-\sqrt{3}}{4\sqrt{2}} & \approx & -0.12941 \end{array}$$

In general, the N-th order wavelet has associated filters of length $2N$. To construct a representation of $\phi(x)$ with support in $[0, 2N-1]$ (for Db2, $N = 2$), Daubechies and Lagarias developed a simple algorithm, which consists of:

— Defining two matrices $(2N-1) \times (2N-1)$ as $T_0 = \sqrt{2}h_{2i-j-1}; 1 \leq i, j \leq 2N-1$ and $T_1 = \sqrt{2}h_{2i-j}; 1 \leq i, j \leq 2N-1$.
— For $x \in]0, 1]$, giving the list of binary digits of x truncated to m digits: e.g., for $m = 8$, $0.05_{10} = 0.00001100_2$ gives $\{0, 0, 0, 0, 1, 1, 0, 0\} = d_{1 \leq j \leq 8}$.
— For each d_j, associating T_{d_j} and constructing the matrix product

$$T_m(x) = \prod_{1 \leq j \leq m} T_{d_j}$$

This matrix converges to a $(2N-1) \times (2N-1)$ matrix, where each column converges to the vector:

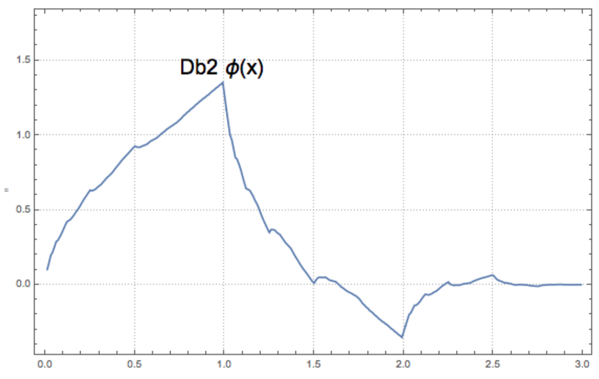

FIGURE 27 – Wavelet $\phi(x)$ for Db2.

$$T_m(x) \underset{m\to\infty}{\longrightarrow} \begin{pmatrix} \phi(x) \\ \phi(x+1) \\ \dots \\ \phi(x+2N-2) \end{pmatrix}$$

For Db2, this procedure yields $\phi(x)$, $\phi(x+1)$, and $\phi(x+2)$ by taking the row-wise averages of $T_m(x)$. The graph of $\phi(x)$ obtained in this way is shown in Figure 27.

The small structures are not due to numerical approximation artifacts. For $\psi(x)$, using the relation

$$\psi(x) = \sum_{k\in\mathbb{Z}} g_k \sqrt{2}\phi(2x-k)$$

and calculating $\phi(x)$ with the given high-pass filter coefficients, we obtain the graph shown in Figure 28 [12].

Before concluding this section, it's worth mentioning that the solution to the Bezout equation for P_0 chosen earlier can be extended if the degree of P_0 is not restricted to $N-1$. In this case, one can take $P(x) = P_0(x) + y^N R(1/2-y)$ with R being an odd polynomial

12. Note: There exists another algorithm for directly calculating $\psi(x)$.

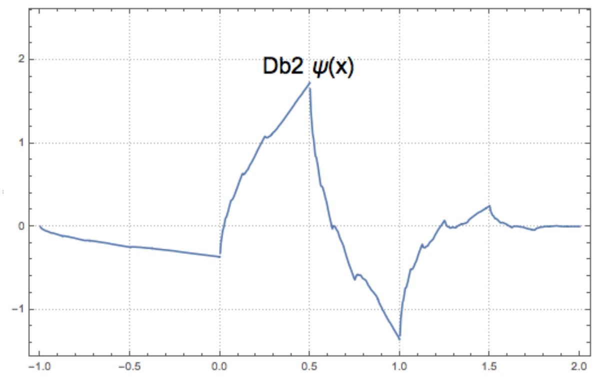

FIGURE 28 – Wavelet $\psi(x)$ for Db2.

that preserves the positivity of $P(x)$. Thus, the Daubechies wavelet family discussed in this section can be extended to include symlets, complex Daubechies wavelets, coiflets, and more.

Now, let's examine how to obtain wavelet coefficients of a discretized signal x_i and its synthesis.

7.6.0.4 *DWT (Discrete Wavelet Transform) and its Inverse IDWT (Version)*

Recalling the decomposition

$$L^2(\mathbb{R}) = \bigoplus_{j=-\infty}^{\infty} W_j = V_{j_0} \oplus \bigoplus_{j\geq j_0}^{\infty} W_j$$

, this can be visualized using the "Russian dolls" concept (Figure 29).

In his course, S. Mallat elaborated the first decomposition to highlight the discontinuities of a function. Instead, we have considered the second version where j_0 is the finest introspection scale.

If we take a signal f sampled at a fine introspection scale J, then $\tilde{f}_J \in V_J$ (note: $J \gg 0$), and we consider that the coarsest scale is j_0. So, as

$$V_J = V_{J-1} \oplus W_{J-1} = V_{j_0} \oplus \bigoplus_{j\geq j_0}^{J-1} W_j$$

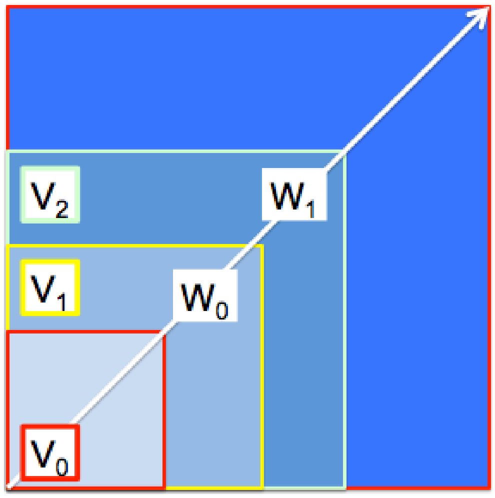

FIGURE 29 – Nesting sets of a multi-resolution analysis.

we have

$$\tilde{f}_J(x) = \overbrace{\sum_{k=-\infty}^{\infty} c_{j_0,k} 2^{j_0/2} \phi(2^{j_0}x - k)}^{V_{j_0}} + \sum_{j \geq j_0}^{J-1} \left\{ \overbrace{\sum_{k=-\infty}^{\infty} d_{j,k} 2^{j/2} \psi(2^j x - k)}^{W_j} \right\}$$

$$= \sum_k c_{j_0,k} \phi_{j_0,k}(x) + \sum_{j \geq j_0}^{J-1} \sum_k d_{j,k} \psi_{j,k}(x)$$

By applying the orthogonality of $\phi_{j,k}$ and $\psi_{j',k'}$, it follows that

$$c_{j_0,k} = \langle \tilde{f}_J, \phi_{j_0,k} \rangle = \sum_\ell h_{\ell-2k} \langle \tilde{f}_J, \phi_{j_0+1,\ell} \rangle$$

Using the equations defining the low-pass filters (h_k) and high-pass filters (g_k) involving the scaling equation for ϕ and the equivalent one for ψ, it can be easily shown that

$$\phi_{j,\ell} = \sum_k h_{k-2\ell}\phi_{j+1,k} \quad (1)$$
$$\psi_{j,\ell} = \sum_k g_{k-2\ell}\phi_{j+1,k} \quad (2)$$

All of this results in the recurrence relation (anonymizing j_0):

$$\boxed{c_{j-1,k} = \sum_\ell h_{\ell-2k}c_{j,\ell}}$$

Similarly, the relation $d_{j-1,k} = \langle \tilde{f}_J, \psi_{j-1,k}\rangle$ gives a recurrence relation:

$$\boxed{d_{j-1,k} = \sum_\ell g_{\ell-2k}c_{j,\ell}}$$

The **DWT** (Discrete Wavelet Transform) algorithm can be summarized as shown in Figure 30.

The inverse algorithm (**IDWT**) involves using $c_{j_0,.}, d_{j_0+1,.}, \ldots, d_{j,.}, \ldots, d_{J-1,.}$ to synthesize $c_{J,.}$. Locally, we would like to obtain $c_{j,.}$ from $(c_{j-1,.}, d_{j-1,.})$. However, the expansion of V_J can also be performed using the subdivision:

$$V_J = V_{J-1} \oplus W_{J-1} = V_{j_0-1} \oplus \bigoplus_{j\geq j_0-1}^{J-1} W_j$$

if we push the introspection scale one step further. Therefore,

$$\tilde{f}_J(x) = \sum_k c_{j_0-1,k}\phi_{j_0-1,k}(x) + \sum_{j\geq j_0-1}^{J-1}\sum_k d_{j,k}\psi_{j,k}(x)$$

By identifying the common and different parts, and then equating the coefficients of $\phi_{j_0,k}$, and finally anonymizing j_0, we find the recurrence relation:

$$\boxed{c_{j,k} = \sum_\ell c_{j-1,\ell}h_{k-2\ell} + \sum_\ell d_{j-1,\ell}g_{k-2\ell}}$$

Iterating this relation, it becomes clear that reconstructing c_J does not require the intermediate $c_{j,.}$ values except, of course, for $c_{j_0,.}$. The IDWT scheme is thus depicted in Figure 31.

However, if you pass through a DWT phase before proceeding to a new IDWT, for example, after "cleaning" the detail coefficients, then you have access to the intermediate $c_{j,.}$ values. The concrete implementation has some subtleties, which I will briefly touch upon here for the DWT phase. In principle, as shown above, we start by initializing $c_{J,k} = x_k$ for $k = 0, \dots, N_s - 1$ if we have N_s samples, and then we progressively calculate $(c_{J-1,.}, d_{J-1,.})$, $(c_{J-2,.}, d_{J-2,.})$, and so on. However, it is more natural to initialize $c_{0,k} = x_k$ to proceed with the calculation of pairs $(c_{1,.}, d_{1,.})$, $(c_{2,.}, d_{2,.})$, and so forth. This is just a simple index rearrangement, so the recurrence on the $c_{j,k}$ becomes:

$$c_{j+1,k} = \sum_{\ell} h_{\ell-2k} c_{j,\ell}; \quad j = 0, \dots$$

Next, if we consider Daubechies wavelets of order N, the filter h has a length of $2N$: $\{h_0, h_1, \dots, h_{2N-1}\}$. Thus, the sum over ℓ is constrained by $0 \leq \ell - 2k \leq 2N - 1$, which, through a change of index, translates to the relevant elements of the $c_{j,.}$ vector as:

$$c_{j+1,k} = \sum_{\ell=0}^{2N-1} h_\ell \ c_{j,\ell+2k}$$

This type of relation is typical of multiplying the $c_{j,.}$ vector by a circulant matrix with a shift of 2 between each row, and the first row being $\{h_0, h_1, \dots, h_{2N-1}, 0, \dots, 0\}$ of length N_s.

The question is to determine the dimension of the $c_{j+1,.}$ vector, given that of the $c_{j,.}$ vector. In fact, this question is related to the handling of boundary effects. If the signal has a length that is a power of 2, i.e., $N_s = 2^s$, one possibility is to construct successive matrices of dimension $2^{s-n} \times 2^{s-n+1}$ with a halving of the number of coefficients per iteration and an assumption of periodic signal. *Mathematica* and *pyWavelets (pywt)* have another strategy with a recurrence relation for the number of coefficients using $w_0 = N_s$

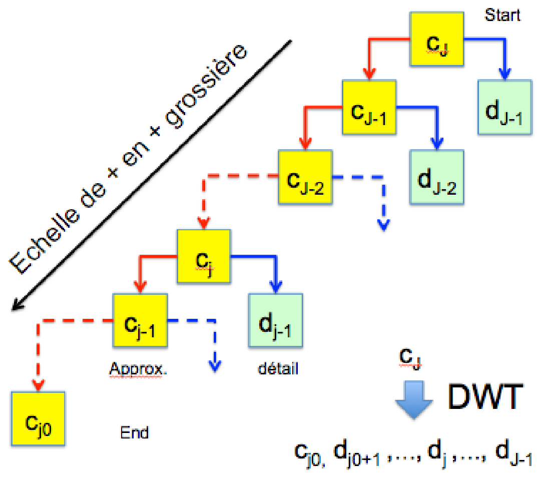

FIGURE 30 – Diagram of a wavelet decomposition (Discrete Wavelet Transform).

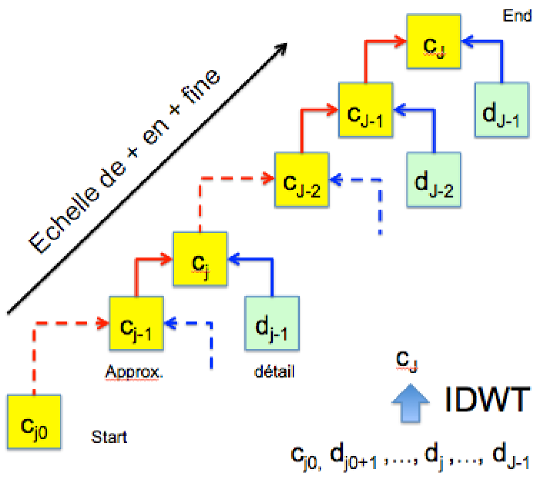

FIGURE 31 – Signal reconstruction using wavelet coefficients (Inverse Discrete Wavelet Transform).

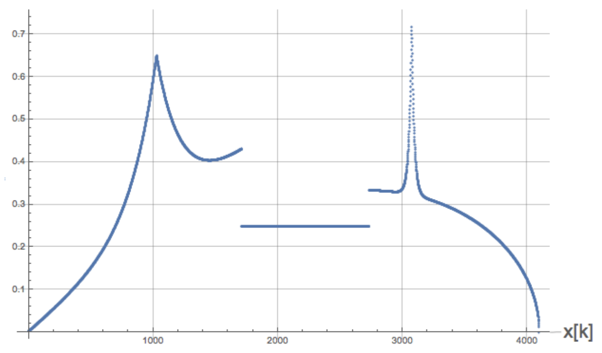

FIGURE 32 – Digitized test function (4096 samples) with various types of discontinuities.

and $FL = 2N$ (filter length):

$$\begin{aligned} w_j &= \left\lceil \frac{1}{2}(w_{j-1} + FL - 2) \right\rceil \quad \text{(Mathematica)} && (3) \\ \text{or} \quad w_j &= \left\lfloor \frac{1}{2}(w_{j-1} + FL - 1) \right\rfloor \quad \text{(pywt)} && (4) \end{aligned}$$

($\lceil x \rceil$: *ceil(x)*; $\lfloor x \rfloor$: *floor(x)*). Note that the two relations are equivalent.

Finally, there is the maximum admissible introspection depth, which is the maximum value of j in the recurrence relation for $c_{j,\cdot}$ (where $j = 0$ is the input signal state). There are different definitions depending on the libraries:

$$\begin{aligned} j_{\max} &= \left\lfloor \log_2(N_s) + \frac{1}{2} \right\rfloor = s \quad \text{(Mathematica)} && (5) \\ j_{\max} &= \left\lfloor \log_2\left(\frac{N_s}{FL - 1}\right) \right\rfloor < s \quad \text{(pywt)} && (6) \end{aligned}$$

(e.g., for *pywt* with Db2 wavelets, $j_{\max} = s - 1$). The definition used by *pywt* aims to minimize the impact of border effects on the $c_{j_{\max},\cdot}$ vector.

7.6.0.5 *Thresholding: Difference between FFT and DWT, Gibbs Phenomenon*

Let's consider the example of a signal like the one in Figure 32, sampled with $4096 = 2^{12}$ samples. The decomposition using Db2 wavelets, where we limited the introspection or refinement level to 6 (the maximum is 12), is shown in Figure 33. The coefficients $d_{j,k}$ of the high-frequency "details" are shown on each row, and the coefficients $c_{6,k}$ of

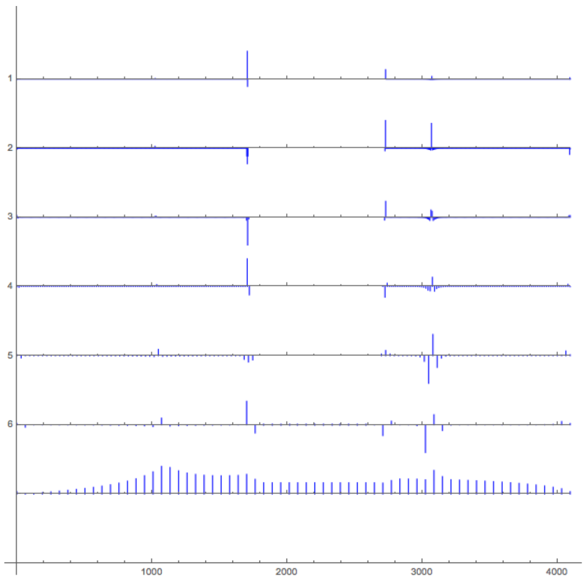

FIGURE 33 – Wavelet coefficients ψ at different detail scales. The bottom row corresponds to the low-frequency approximation obtained by the ϕ wavelet.

the low-frequency approximation are shown on the last row. The total number of detail coefficients is 4109, and there are 66 approximation coefficients. In the course, S. Mallat suggests keeping only the M most significant **detail** coefficients; let's take $M = 100$. So, there are 100+66=166 retained wavelet coefficients. Similarly, when performing an FFT, one can keep the first M low-frequency coefficients.

Firstly, S. Mallat mentions the Gibbs phenomenon, which was highlighted in 1848 by H. Wilbraham, an English mathematician, but popularized in 1898 by A. Michelson (of Michelson & Morlet). Michelson initially thought it was an artifact of his instruments, but it was ultimately explained by J.W. Gibbs as a mathematical phenomenon related to the Fourier transformation's approximation error near discontinuities. This phenomenon is clearly visible in our case of "thresholding", as shown in Figure 34. With the same number of coefficients, the wavelet reconstruction is much more faithful (see Figure 35). This demonstrates the importance of an adaptive algorithm (non-linear according to S. Mallat's notion). Of course, if you only had the FFT, you could get by with filtering (which would smooth out the irregularities).

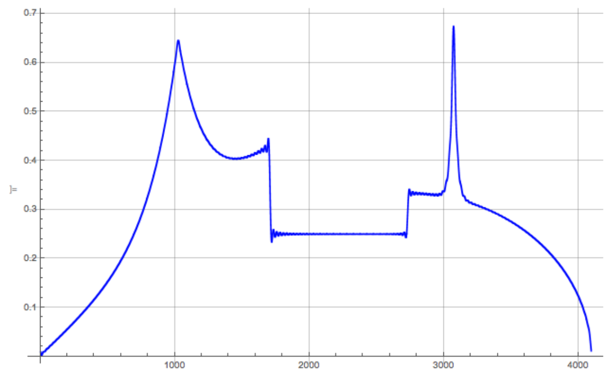

FIGURE 34 – Gibbs phenomenon near signal singularities when truncating the low-frequency components of the Fourier transform.

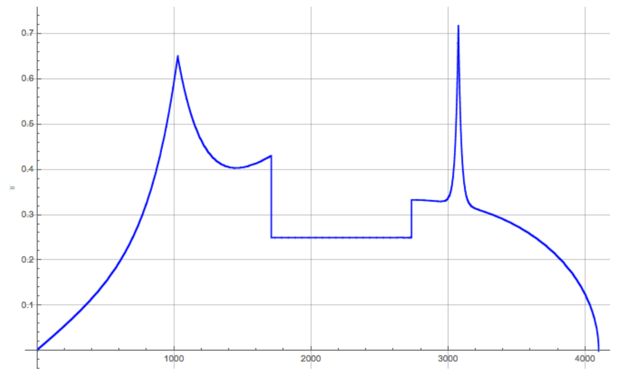

FIGURE 35 – Signal reconstruction using IDWT by retaining only the most significant detail coefficients, added to those of the low-frequency approximation, while keeping the same total number of coefficients as the inverse FFT in Figure 34.

8. High-Dimensional Classification/Regression (Part I)

8.1 A Brief Recap

Recall that the central issue is the **"curse of dimensionality"** when trying to estimate/approximate functions in very high dimensions. To address this problem, we have seen that it is necessary to **reduce the dimensionality** of the problem by identifying forms of **regularity**. We returned to lower dimensions to understand the topic of (Lipschitz) regularity and explored **Fourier Transform** (*linear case*) and **Wavelet Transform** (*non-linear case*), as well as more generally, **sparse representations**. These tools will allow us to construct representations within the framework of **classification and regression**.

We will revisit these concepts in connection with **algorithms**, especially in this course, **linear classification/regression techniques** with representations like **kernel classifiers**.

8.2 Deterministic vs. Stochastic Models: It's Not the Problem!

When addressing classification/regression problems, we can have two perspectives:

	Deterministic	Stochastic
Model	$y = f(x)$	y is a random function of x
Question	We want to estimate f	Find the most probable y (Bayesian)

However, deciding between the two perspectives ("one is better than the other") is not the problem! The real question is: how can we capture the regularity of the problem in order to determine the model parameters with as few samples as possible.

8.2.1 Bayesian and Deterministic Perspectives

Let's examine **the Bayesian perspective** in the context of supervised learning in d dimensions ($x \in \mathbb{R}^d$), where we have n samples $\{x_i, y_i\}_{i \leq n}$. Let $\tilde{y}$ **be the estimator of** y

such that

$$\tilde{y} = \tilde{f}(x)$$

To do this, we define a **cost or risk** (the term that will be used later) of making an error when estimating y with $\tilde{y}$: $r(y, \tilde{y})$. We seek $\tilde{f}$ that **minimizes the average risk**, which is given by

$$\tilde{f} = \underset{f \in \mathcal{H}}{\text{argmin}}\; E_{X,Y}[r(Y, f(X))]$$

where (X, Y) are the probability distributions that « generate » x and y. Recall that the « classical » risk functions are:

— **Classification**, where $y \in \{-1, 1\}$ (generalizable to K classes), using a binary risk defined as

$$r(y, \tilde{y}) = \begin{cases} 0 & \text{if } y = \tilde{y} \\ 1 & \text{if } y \neq \tilde{y} \end{cases}$$

— **Regression**, where $y \in \mathbb{R}$, using, for example, a quadratic risk

$$r(y, \tilde{y}) = (y - \tilde{y})^2$$

In this probabilistic approach, we need to calculate the average risk, which in the case of regression is given by

$$\begin{aligned} E_{X,Y}[r(Y, f(X))] &= \int\int p(x, y) r(y, f(x)) dx dy \\ &= \int_{\mathbb{R}^d} p(x) \int_{\mathbb{R}} p(y|x) r(y, f(x)) dy dx \end{aligned}$$

In classification, we would replace $\int dy$ with a discrete sum over classes $\sum_{y=1}^{K}$, and thus

$$\begin{aligned} E_{X,Y}[r(Y, f(X))] &= \int_{\mathbb{R}^d} p(x) \sum_{y=1}^{K} p(x, y) r(y, f(x)) dx \\ &= \int_{\mathbb{R}^d} p(x) \sum_{y=1}^{K} p(x, y) \times \mathbb{I}(y \neq f(x)) dx \end{aligned}$$

So, when x is fixed, we want to minimize

$$\sum_{y=1}^{K} p(x, y) \times \mathbb{I}(y \neq f(x))$$

which means that when $p(y,x)$ is large, we want $f(x) \approx y$. In other words,

$$\boxed{p(\tilde{f}(x)|x) = \max_f(p(f(x)|x))}$$

This is the well-known **Bayesian classifier** that selects the most probable class given x. Is this the end of the story since we have a formula/a scheme?

In fact, there is an underlying assumption of **regularity on** $p(y|x)$ that can be formulated quite trivially: *when x varies only slightly, the probability $p(y|x)$ does not change much.* This simply means that when x is « close » to a training sample x_i, we consider $p(y|x_i) = y_i$ (a constant). This is the **k-nearest neighbors algorithm** that we encountered in the lecture on 31/1/18 (Part 1). However, **the space between samples in high dimensions is enormous**, so we need many samples n to reasonably estimate $p(y|x)$. In other words, we require a very high sample density to consider that the labeling error ε is low. We have seen that

$$n \approx \varepsilon^{-d}$$

For the **deterministic perspective**, f is a unique function to be determined for which $y = f(x)$. In fact, we can define a conditional probability

$$p(y|x) = \delta(y - f(x))$$

The fact that f is unique is not surprising, especially in the context of high dimensionality d, where we have a lot of information to distinguish, for example, images of cats, dogs, cars, etc., or sound samples for speech recognition (of course, this is the idea...). Thus, by identifying $p(y|x)$, we are reduced to the previous discussion.

Consider the case of regression with y as a continuous variable

$$E_{X,Y}[r(Y, f(X))] = \int_{\mathbb{R}^d} p(x) \int_{\mathbb{R}} p(y|x)(y - f(x))^2 dy\ dx$$

In fact, we want to minimize $(y - f(x))^2$ given x. Therefore, the solution is nothing else but

$$\boxed{\tilde{f}(x) = E[Y|X] = \int_{\mathbb{R}} y\ p(y|x)\ dy}$$

This is a specific result of the general result

$$\min_{\mu} E[(Y-\mu)^2] \Leftrightarrow \mu = E[Y]$$

The problem is that in high dimensions, to estimate $E[Y|X]$ around a fixed x, either the ball for collecting samples is huge, and then it is likely that $p(y|x) = Cte$ is not correct, or you need a high sample density to make the ball's radius small, but then $n \approx \varepsilon^{-d}$. We always encounter the same problem of the curse of dimensionality, which makes these conditional expectations very difficult to calculate in high dimensions (see MCMC and other quasi-Monte Carlo integration techniques).

So, whether from a deterministic point of view or from a probabilistic perspective that initially provides a scheme to solve the problem, the curse of dimensionality remains.

The solution (perhaps temporary) is to circumvent the problem and **impose strong regularity** either on $f(y)$ (deterministic perspective) or on $p(y|x)$ (probabilistic perspective). We no longer assume solely that the function is locally Lipschitz or Lipschitz-α, but we drastically reduce the dimensionality d of the problem and reduce it to **only 1 variable**. Generalizing this (very rough) idea of going from d variables to 1 variable will encompass almost all learning algorithms other than NN (MLP).

8.3 Dimensionality Reduction: Similarity Kernel and Hyperplane

For example, let's consider **binary classification** $y = \pm 1$. The idea is as follows: to separate the 2 classes in d dimensions, we use a transformation

$$z = \phi(x)$$

such that the boundary between the 2 classes is a **hyperplane: thus, we have linearized the boundary**. Figure 36 illustrates the process.

In the new space, to classify a sample, you only need to know its position relative to the hyperplane, i.e., know the sign of the distance between the sample and the hyperplane.

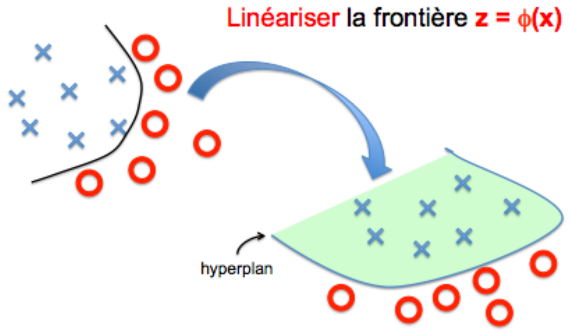

FIGURE 36 – Linearization of the problem by changing the representation $z = \phi(x)$. We then look for the hyperplane that separates the 2 classes of samples.

This is expressed by defining an estimator for y, denoted $\tilde{y}$, as follows

$$\boxed{\tilde{y} = \operatorname{sgn}\left\{\langle w, \phi(x)\rangle + b\right\}}$$

where w is the normal vector to the hyperplane, $\langle w, \phi(x)\rangle$ is the vector projection of $\phi(x)$ onto the normal, and b is the position of the affine plane. **The parameters** (w, b) **must be estimated: the components of vector** w **and the scalar** b**, i.e.,** $d+1$ **variables.** We introduce the concept of **similarity** to characterize if two samples are « close » to each other, as in the case of the k-nearest neighbors algorithm, which is natural in low dimensions (it is known not to work well in high dimensions). So, let's take the Euclidean distance in the new space.

$$||\phi(x) - \phi(x')||^2 = ||\phi(x)||^2 + ||\phi(x')||^2 - 2\langle\phi(x), \phi(x')\rangle$$

Therefore, two samples are similar if their distance is close to 0, meaning that their inner product is large. So, studying similarity in the new space is equivalent to studying the **kernel function**

$$\boxed{k(x, x') = \langle\phi(x), \phi(x')\rangle}$$

Finally, in this linearization example, we have the choice (and equivalence): **either define the function** ϕ **as the change of variables, or define directly the similarity kernel between two points in the new space**. So, the problem is to find the « right » change of variables to linearize the boundary, or the « right » kernel that best aggregates the samples into clusters.

Another way to view this problem is by expanding the dot product:

$$\phi(x) = (\varphi_k(x))_{k \leq d'} \to \langle w, \phi(x) \rangle = \sum_k w_k \varphi_k(x)$$

We then obtain a weighted average (where w is normalized to 1) among d' **"patterns" or "structures"** $\varphi(x)$, or in other words, we perform a **"vote"** among the structures, and w is the **discriminant vector**[13].

Now that we have linearized the boundary to be sought to separate the two classes, are we free from the curse of dimensionality? We need to answer two types of questions (not necessarily in this order):

— Q1: How to optimize $(w, b) \in \mathbb{R}^{d+1}$?

— Q2: How to optimize the change of variables or **representation** $\phi(x)$?

Q2 is the more critical one, and to address it, two strategies are considered:

— **Either we fix *a priori*** the representation $\phi(x)$, which can be replaced by the choice of the **similarity kernel** $k(x_1, x_2)$;

— **Or the algorithm is capable of learning the representation by itself** because we do not have enough elements/information to fix it *a priori*. This is the domain of **neural networks (NN/MLP)**.

8.3.1 Finding the hyperplane (w, b): Necessary Regularization

For now, we'll forget about $\phi(x)$. In the new space, we denote it as x, which is a vector in d' dimensions, but we'll use d for simplicity. We are looking for a classifier in the form of

$$\text{sgn}\left(\sum_k w_k x_k + b\right)$$

The space of classifiers $\mathcal{H}$ is defined by the set of (w, b)

$$\mathcal{H} = \left\{(w, b) \in \mathbb{R}^{d+1}\right\}$$

13. Note: S. Mallat shows in a simple example that counting the number of "red" pixels in an image containing either fire trucks or regular trucks can be a simple discriminant pattern for recognizing fire trucks.

However, we have seen in the course of 24/1/18 that the *fluctuation error* related to the size of $\mathcal{H}$ is given by

$$\varepsilon^2 \sim \frac{\log|\mathcal{H}|}{n}$$

which is the ratio between the size of $\mathcal{H}$ and the number of examples/samples. Here, even if $\mathbb{R}^{d+1}$ is infinite, we can quantify it, but we would need an **infinite number of bins**, which leads to an infinite set of « possibilities », resulting in the **overfitting** effect. Therefore, we need to **impose a condition to reduce the dimensionality** of $\mathcal{H}$ to consider and reduce overfitting.

Notation: If $x' = (x_1, \dots, x_d, b)$ and $w' = (w_1, \dots, w_d, 1)$, then $\sum_k w_k x_k + b = \sum_\ell w'_\ell x'_\ell$. From now on, we'll forget the $'$.

Let's consider the new **regularized empirical risk** (note that x_i here represents the i-th sample and not the i-th component):

$$\boxed{\tilde{R}_r(h) = \frac{1}{n}\sum_{i=1}^{n} r(y_i, h(x_i)) + \lambda||w||^2} \quad \text{with } h(x_i) = \text{sgn}\langle w, x_i\rangle$$

With the new term, we reduce $\mathcal{H}$ to a **compact set**. In general, the concepts of **overfitting, prediction stability, and convexity of $\mathcal{H}$ are closely related**.

Let's examine for a moment the case *without regularization* for a regression problem (we will address classification afterward, which adds complexity). We have an empirical risk $\tilde{R}(h)$ such that

$$\tilde{R}(h) = \tilde{R}(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - h(x_i))^2$$

The minimization with respect to the components of w leads to a linear equation

$$\frac{1}{n}\sum_i \langle w, x_i\rangle x_i = \frac{1}{n}\sum_i y_i x_i \equiv c$$

(Note: Here, x_i is a vector in $d+1$ dimensions, even though d is the dimension of the space after the change of representation.) If we define the matrix $\mathbf{A}$ as

$$\mathbf{A} = \frac{1}{n}\sum_i x_i x_i^T$$

then we need to solve a linear system of the form

$$\mathbf{A}w = c$$

whose solution, if $\mathbf{A}$ is invertible, becomes:

$$w = \mathbf{A}^{-1}c$$

But, there are two « buts »:

— In general, $\mathbf{A}$ is non-invertible. This can typically be handled using the concept of the Moore-Penrose pseudo-inverse;

— $\mathbf{A}^{-1}$ is **unstable** (branch of « inverse problems »), and an error on c (changing the response y_i for an x_i) leads to an amplification of errors. This naturally results in **overfitting**.

Let's take a closer look. The definition of $\mathbf{A}$ tells us that it is symmetric, so it is diagonalizable in the form:

$$\mathbf{A} = \mathbf{O}\mathbf{D}\mathbf{O}^T$$

where $\mathbf{O}$ is an orthogonal matrix, and $\mathbf{D}$ is the diagonal matrix of eigenvalues of $\mathbf{A}$. If the space spanned by the n samples (V) is of dimension d, then $\mathbf{A}$ is invertible ($\mathbf{D}$ has no zero values). If the space spanned is of lower dimension, then we need to construct the pseudo-inverse denoted $\mathbf{A}^+$. The $\mathbb{R}^d$ space is divided into two vector subspaces, V and $V^\perp$, defined by

$$\begin{cases} \forall x \in V^\perp & \mathbf{A}^+x = 0 \\ \forall x \in V & x = \mathbf{A}w \Rightarrow \mathbf{A}^+x = w \end{cases}$$

Note that the singular value decomposition (SVD) can be used to define the pseudo-inverse. The solution to the linear problem

$$\mathbf{A}w = c$$

consists of a general solution to the homogeneous equation $\mathbf{A}w = 0$ and a particular solution given by applying the pseudo-inverse if (and only if) $\mathbf{A}\mathbf{A}^+c = c$. Therefore, in general, we have:

$$w_{sol} = \mathbf{A}^+c + \mathbf{P_0}w_0$$

where $\mathbf{P_0} = \mathbf{I} - \mathbf{A}^+\mathbf{A}$ is the orthogonal projector onto the kernel of $\mathbf{A}$; $\mathbf{AP_0} = 0$.

8.3.2 How Does Regularization Help Stabilize the Response?

We have found a way to calculate the parameters of the hyperplane (w, b) that separates the two populations of labeled samples $\{x_i, y_i\}$. But does this solve the problem of **generalization error**, i.e., when we take **test samples**? So, after training, for any x, we have an estimator of the response $\tilde{f}(x)$ (*recall: b is implicitly contained in x*),

$$\tilde{f}(x) = \langle \tilde{w}, x \rangle$$

Depending on the type of x, we obtain different empirical risks:

$$\begin{cases} x_i \in \text{"Training set"} & \to \tilde{R}(\tilde{f}) \\ x_i \in \text{"Test set"} & \to R^{test}(\tilde{f}) \end{cases}$$

Imagine that the « Test set » is identical to the « Training » set except for a single sample $(x_k, \bar{y}_k)$. If the estimator $\tilde{f}$ is stable, then a small change of this kind has no influence on the risk, and there is no overfitting. In fact, we wonder what would happen when we calculate w after changing the value of c (denoted $\bar{c}$) when changing y_k to $\bar{y}_k$? It follows that $\mathbf{A}\bar{w} = \bar{c}$ and

$$||w - \bar{w}|| = ||\mathbf{A}^{-1}(c - \bar{c})|| \leq ||\mathbf{A}^{-1}|| \times ||c - \bar{c}||$$

In principle, $||c-\bar{c}||$ is small because we made a small change to only one sample. However,

$$||\mathbf{A}^{-1}|| = \max(\sigma_\ell^{-1})_\ell$$

So, if the matrix A has a very small eigenvalue, there will be instability. To guard against small eigenvalues in « inverse problems », regularization is employed! This is a topic discovered in several fields in the 1940s and is known as Tikhonov-Miller regularization, as mentioned by S. Mallat.

How does this regularization actually stabilize things in practice? Recall that the empirical risk is equal to

$$\tilde{R}_r(h) = \frac{1}{n}\sum_{i=1}^{n}(\langle \tilde{w}, x_i\rangle - y_i)^2 + \lambda w.w^T$$

The gradient with respect to w gives the following regularized equation:

$$(\mathbf{A} + \lambda \mathbf{I})w = c$$

However, the matrix $\mathbf{A} + \lambda\mathbf{I}$ is always invertible because, using the decomposition of $\mathbf{A}$, we realize that the new diagonal matrix is given by:

$$\mathbf{D}_r = \mathbf{D} + \lambda\mathbf{I}$$

and thus the new eigenvalues are always $\sigma'_\ell \geq \lambda$. So, when we consider the effect of a small change as before, we get the inequality

$$||w - \bar{w}|| \leq ||(\mathbf{A} + \lambda\mathbf{I})^{-1}|| \times ||c - \bar{c}|| = \frac{1}{\lambda}||c - \bar{c}||$$

Therefore, if $||c - \bar{c}|| \approx \varepsilon$, then $||w - \bar{w}|| \approx \varepsilon$. **We have effectively eliminated/reduced overfitting.**

8.3.3 Convexity

Another way to see things is that problem (Pb1) boils down to minimizing

$$\min\, \tilde{R}_r(w) = \min\left(\frac{1}{n}\sum_{i=1}^{n}(\langle w, x_i\rangle - y_i)^2 + \lambda||w||_2^2\right)$$

which is equivalent to problem (Pb2) of constrained minimization, namely

$$\min\, \tilde{R}(w) = \min\left(\frac{1}{n}\sum_{i=1}^{n}(\langle w, x_i\rangle - y_i)^2\right) \quad \text{with} \quad ||w||_2^2 < \beta$$

Now, $\tilde{R}(w)$ is a **convex function** that we want to minimize **with a constraint**. We can interpret this problem as a Lagrangian problem, where a λ (*Lagrange multiplier*) corresponds to β, which bounds the L2 norm of w. In this context, solving problem Pb2 amounts to ensuring that the space of w ($\mathcal{H}_\beta$) has a smaller dimension than $\mathcal{H}$ without constraints. **Once again, we combat dimensionality by reducing the space of functions in which we seek the solution.**

Different types of regularization can be imposed, but we essentially consider L2 and L1 norms:

— **L2 Norm**: $||w||_2^2 = \sum_k |w_k|^2 < \beta$, easier to handle;

— **L1 Norm**: $||w||_1 = \sum_k |w_k| < \beta$, which favors a « sparse » w with many 0s (sparse representation) and a few large components, used for feature selection.

8.3.4 Risk in terms of the dual variables of w

We quickly realize that $w \in V$, the space generated by the training samples x_i. Indeed, we can linearly decompose w into a V component (w_1) and a $V^\perp$ component (w_2), so the regularized empirical risk is

$$\min \tilde{R}_r(w) = \min \left(\frac{1}{n} \sum_{i=1}^{n} (\langle w, x_i \rangle - y_i)^2 + \lambda(||w_1||^2 + ||w_2||^2) \right)$$

The regularization term will make component w_2 tend toward 0, which constrains $w \in V$, and mechanically, this implies that

$$\boxed{w = \sum_{i=1}^{n} \alpha_i x_i}$$

(this is the *representer theorem*, which has practical applications when $n < d$). The α_i are the **dual variables** of w. The regularized empirical risk to be minimized has the expression

$$\boxed{\tilde{R}_r(\alpha) = \frac{1}{n} \sum_{i=1}^{n} (\sum_{k=1}^{n} \alpha_k \langle x_k, x_i \rangle - y_i)^2 + \lambda \sum_{k,k'} \alpha_k \alpha_{k'} \langle x_k, x_{k'} \rangle}$$

$\tilde{R}_r(\alpha)$ is thus a **convex quadratic form** in α_k with coefficients that are the inner products $\langle x_k, x_i \rangle$. But let's remember that in fact, these are the inner products *after* *changing the representation*, so they should be seen as $\langle \phi(x_k), \phi(x_i) \rangle$. **In the end, it is not necessary to know ϕ, but rather the values of the similarity kernel $k(x_k, x_i)$.**

We have approximately n^2 coefficients (inner products) independently of the dimension of $\phi(x)$ ([14]). So, if I have to compute them from $\phi(x)$, it can be prohibitive if the dimension of $\phi(x)$ is large. Especially for very good reasons, we want to have a very large dimension of $\phi(x)$ (e.g., several thousand). Indeed, in very high dimensions, you always find a feature (in the broad sense) that is discriminative and can separate the two classes. This was the hope in the 2000s because at that time we thought (in a caricatured way): « we just have to work in very high-dimensional structures/features! ». Furthermore, if we calculate the inner products directly using an analytical formula for the kernel, there is no impact on the computation time.

But: **everything will work, but if the dimension of $\phi(x)$ is too large, we will fall into overfitting.** So, in practice, we come back to very down-to-earth aspects of how to find the right $\phi(x)$, or how to find the right features.

9. High-Dimensional Classification/Regression (Part II)

9.1 Regression (Kernel Model): Bias-Variance Trade-off

Recall that in the case of the « curse of dimensionality », the fluctuation error and the number of samples behave as $n \approx \varepsilon^{-d}$. The work on changing the representation $\phi(x)$ (cf. Part I) aimed to drastically reduce the dimension of the space in which we look for an estimator $\tilde{f}$ (in the deterministic case where $y = f(x)$), keeping only the $d+1$ components of $w \in \mathbb{R}^{d+1}$ as parameters. Therefore, $n \approx d\varepsilon^{-2}$, and there is no explosion as d becomes large. But that doesn't mean we're out of the woods yet; we also need to find ϕ and avoid overfitting.

If we define $x \in \Omega = \{x \in \mathbb{R}^d / ||x|| \leq 1\}$ (bounded) and similarly, the response y is bounded by $[-1, 1]$ (in regression), the minimization problem $\tilde{R}_r(w)$ and

14. ndje: imagine that the components of $\phi(x)$ are all monomials of a certain degree obtained from the components of x. Ex. $(x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1^2x_2, x_1^2x_3, x_2^2x_1, x_2^2x_3, x_3^2x_1, x_3^2x_2, \dots)$

$\tilde{w} = \underset{w}{\operatorname{argmin}} \tilde{R}_r(w)$, which means $\tilde{w}$ is the best discriminative direction (minimizing the regularized empirical risk). If we define the « true » risk as

$$R(w) = E_{X,Y}[r(Y, \langle w, X \rangle)]$$

that we want to compare to $E(\tilde{R}_r(\tilde{w}))$, which means

$$E(\tilde{R}_r(\tilde{w})) \leq \min_{w \in \mathcal{H}} R(w) + \varepsilon^2$$

we would like ε^2 to decrease rapidly as the number of samples (n) increases.

Theorem: If we constrain w to be sought in the set $\mathcal{H}_{B\sqrt{d}} = \{w/||w||^2 < B^2 d\}$ then

$$\boxed{\varepsilon^2 = 150\frac{B^2 d}{n}, \quad \text{and} \quad \lambda = \frac{\varepsilon}{3dB^2}}$$

Let's see what we can expect with this result. On one hand, if we regularize with a value of λ, the result on w is that its norm is constrained, and this is all the more true as λ increases. Therefore, the size of class $\mathcal{H}$ decreases ($\propto B$) as λ increases. Thus, we understand the relationship $\lambda \leftrightarrow B$. On the other hand, the first relation states that $\varepsilon^2 \propto$ (size of $\mathcal{H})/n$; this is what we expected from the general formula. Finally, we realize that if we want ε to be small, we cannot choose λ too large because it would introduce a too significant bias term.

Thus, as soon as the number of samples n is much larger than the size d of the ϕ vector, we guarantee **a small fluctuation error** ε but now there is no guarantee that **the bias term** $\min_{w \in \mathcal{H}_{B\sqrt{d}}} R(w)$ is small, meaning that I will approximate the training data with a linear combination with $w \in \mathcal{H}_{B\sqrt{d}}$. **This is the limit of such an estimator where classification/regression relies on only one discriminative direction w.**

9.2 The Counterpart for Classification

Compared to the regression discussed earlier, there is an added complication in classification: the **non-convexity of the problem** at first glance. In regression, we have a convex function (because the risk is convex) to minimize under constraints, which can be

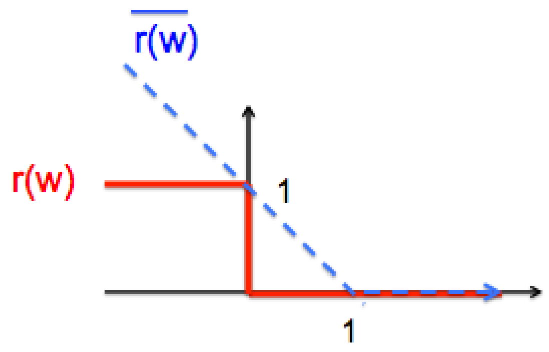

FIGURE 37 – Convexification of the risk $r(w)$ through the upper bound $\bar{r}(w)$.

solved perfectly with gradient descent. However, this is not the case in classification, as we will need to go through a phase to make the problem convex.

In classification, we typically have a risk $r(y, \tilde{y})$ with the estimator $\tilde{y}$ given by:

$$r(y, \tilde{y}) = \begin{cases} 0 & \text{if } y = \tilde{y} \\ 1 & \text{if } y \neq \tilde{y} \end{cases} ; \tilde{y} = \operatorname{sgn}(\langle w, x \rangle + b)$$

If we take $y_i = 1$, as long as $\tilde{y}_i = -1$ (I'm on the wrong side of the hyperplane), the risk is 1, and as soon as $\tilde{y}_i = 1$ (I'm on the right side), the risk is 0.

A few reminders about convexity: - A set Ω is convex if

$$\forall x, y \in \Omega, \alpha \in [0, 1], \alpha x + (1 - \alpha) y \in \Omega$$

- A function $f(x)$ is convex on Ω if

$$\forall x, y \in \Omega, \alpha \in [0, 1], f(\alpha x + (1 - \alpha) y) \leq \alpha f(x) + (1 - \alpha) f(y)$$

- The set Ω_c of x such that $f(x) \leq c$ is a convex set if f is convex. - Finally, if f has a minimum f_{min} on a convex Ω, then

$$\Omega_{min} = \{x / f(x) = f_{min}\}$$

is convex, and it reduces to a single point if f is *strictly convex*.

So **the risk is not convex** (see the red curve in the figure). Indeed, taking $x_1 = -1$ and $x_2 = 1$, we should have the property

$$\alpha \in [0,1],\ f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha$$

which is not satisfied for $\alpha = 1/2 - \epsilon$ ($\epsilon \ll 1$) since the left-hand side is equal to 1.

To make it convex (by simplifying it) and ensure a small ε of fluctuations, as we saw in regression, we use an upper bound for the risk r like the one shown in blue in the figure (originating from the Hinge loss, the basis of Support Vector Machines by V. N. Vapnik & A. Chervonenkis (1963)). Thus, when we obtain an upper bound for the new risk ($\bar{r}$), it effectively becomes an upper bound for the old risk (r).

9.3 Kuhn & Tucker's Saddle Point Condition (1950)

This concerns the risk minimization with a constraint on $||w||^2$. Let's consider problem 1:

$$\min_{x\in\Omega} f(x) \quad \text{and} \quad C_k(x) \leq 0 \quad \text{(Prob. 1)}$$

where $C_k(x)$ represents a series of constraints. Now, let's introduce the Lagrangian ($\lambda = (\lambda_1, \ldots, \lambda_K)$, and $\lambda_k \geq 0$):

$$L(x,\lambda) = f(x) + \sum_{k=1}^{K} \lambda_k C_k(x)$$

A **sufficient condition** (Proposition 2) can be expressed as follows:

if there exists a saddle point $(\bar{x}, \bar{\lambda})$ (Figure 38), defined as:

$$\forall (x,\lambda) \in \Omega \times (\mathbb{R}^+)^K \quad L(\bar{x},\lambda) \leq L(\bar{x},\bar{\lambda}) \leq L(x,\bar{\lambda}) \quad \text{(Prop. 2)}$$

then we have the following properties:

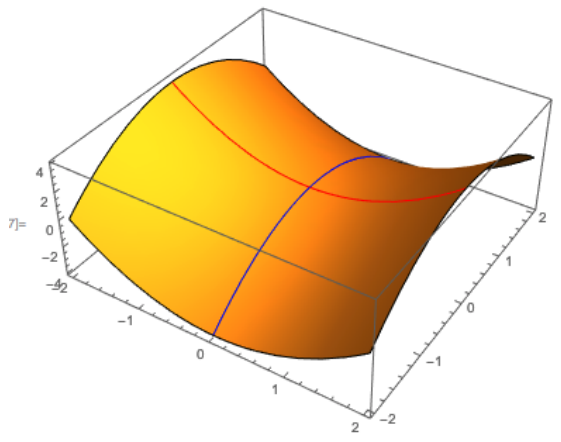

Figure 38 – Saddle point at the intersection of the blue (max) and red (min) curves.

$$\begin{cases} \bar{x} \text{ is a solution of Prob. 1} \\ \text{and } \forall k, \bar{\lambda}_k C_k(\bar{x}) = 0 \end{cases}$$

The second property implies that if the constraint $C_k(\bar{x}) < 0$, then $\bar{\lambda}_k = 0$, and at the constraint boundary, $\bar{\lambda}_k \neq 0$. **This result is fundamental in optimization.** The proof is as follows:

So, if we have a saddle point, taking the first inequality and expressing the Lagrangian explicitly:

$$f(\bar{x}) + \sum_k \lambda_k C_k(\bar{x}) \leq f(\bar{x}) + \sum_k \bar{\lambda}_k C_k(\bar{x})$$

thus:

$$\forall \lambda_k \geq 0, \quad \sum_k (\lambda_k - \bar{\lambda}_k) C_k(\bar{x}) \leq 0$$

We can then choose, for a fixed k_0:

$\lambda_{k_0} = \bar{\lambda}_{k_0} + 1$ and $\forall k \neq k_0$ $\lambda_k = \bar{\lambda}_k$, which translates to:

$$C_{k_0}(\bar{x}) \leq 0$$

and this holds for all k_0. Therefore, the constraints of Prob. 1 are satisfied.

Another particular case is when we fix $\lambda_{k_0} = 0$ for k_0, and $\forall k \neq k_0\ \lambda_k = \bar{\lambda}_k$. This results in the relation:

$$\bar{\lambda}_{k_0} C_{k_0}(\bar{x}) \geq 0$$

Now, as we have shown that $\forall k,\ C_k(\bar{x}) \leq 0$, and $\bar{\lambda}_{k_0} \geq 0$, then $\bar{\lambda}_{k_0} C_{k_0}(\bar{x}) = 0$ for all k_0.

Now, let's take the second inequality concerning the Lagrangian and express it:

$$f(\bar{x}) + \sum_k \bar{\lambda}_k C_k(\bar{x}) \leq f(x) + \sum_k \bar{\lambda}_k C_k(x)$$

Since $\forall k,\ \bar{\lambda}_k C_k(\bar{x}) = 0$, and also $C_k(x) \leq 0$ and $\bar{\lambda}_k \geq 0$, we conclude that:

$$f(\bar{x}) \leq f(x)$$

So, $\bar{x}$ is indeed a minimum, and it satisfies Prob. 1. $\blacksquare$

When the problem is **convex**, i.e., when $f(x)$, the constraints C_k, and the set Ω in which we seek x are all convex, then:

$$\text{If } \exists x_0 \in \Omega / C_k(x_0) \leq 0 \Rightarrow \text{then Prop. 2 is necessary.}$$

and we can express the result in the form of a differential, which is useful in convex optimization. **Thus, the necessary and sufficient conditions whenever we have a convex risk with convex constraints (and a solution in a convex space)** are given by:

$$\begin{aligned}
\partial_x L(\bar{x}, \bar{\lambda}) &= 0 = \partial_x f(\bar{x}) + \sum_k \bar{\lambda}_k \partial_x C_k(\bar{x}) && (7) \\
\partial_\lambda L(\bar{x}, \bar{\lambda}_k) &= C_k(\bar{x}) \leq 0 && (8) \\
\sum_k \bar{\lambda}_k C_k(\bar{x}) &= 0 && (9)
\end{aligned}$$

The goal hereafter is to work within this framework of convexity, even if it means convexifying (by upper-bounding) the risk, as in the case of classification.

10. Support Vector Machine Classification

This refers to research conducted from the 1990s to 2005. These are algorithms that work well (not necessarily those used in practice), but there are interesting concepts: how to linearize a problem, how to make it convex, and thus solve it with simple and efficient algorithms. Moreover, the mathematical framework allows for control up to the risk of estimators.

10.1 The Margin Criterion

So, we are in the case of a binary classification $y = \pm 1$ with $x \in \mathbb{R}^d$.

The **linear estimator/classifier** is determined by the sign of x relative to the hyperplane (w, b) (Figure 39). **If the problem is separable**, then the hyperplane exists, and for all training data:

$$\forall (x_i, y_i)_{i \leq n}, \quad y_i \times (\langle w, x_i \rangle + b) \geq 0$$

($= 0$ on the hyperplane itself). However, the hyperplane is not unique, as shown in Figure 40.

To obtain an optimal solution, we use Vapnik's idea: we should choose the hyperplane that is the most robust to small changes in the training data. In other words, choose the hyperplane that is the "farthest" from both classes: x_i^+ and x_i^- (Figure 41).

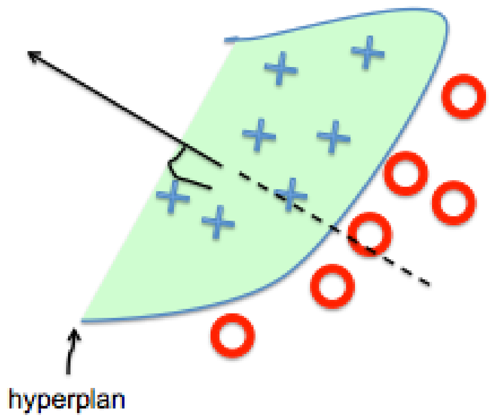

Figure 39 – Normal to the separating hyperplane.

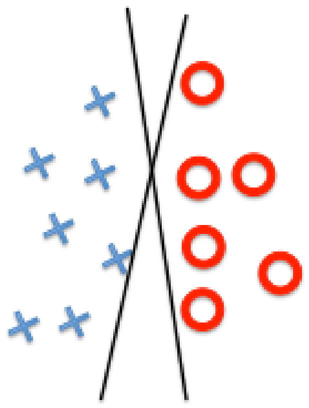

Figure 40 – Which line (or hyperplane in dimension d) to choose?

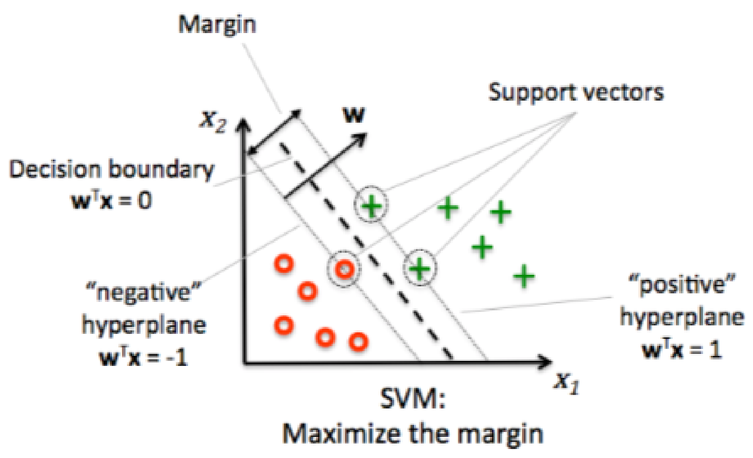

FIGURE 41 – The concept of margin and support vectors.

Note that on the hyperplane $\langle w, x\rangle + b = 0$ and outside it $\langle w, x\rangle + b = c \neq 0$. On either side of the plane, we can find the points x_1 and x_2 closest to the hyperplane, belonging to the two classes, and such that the hyperplane is in the middle (i.e., the point $(x_1 + x_2)/2$ is on the hyperplane). Therefore:

$$\begin{cases} \langle w, x_1\rangle + b & = m||w|| \\ \langle w, x_2\rangle + b & = -m||w|| \end{cases}$$

With $m||w||$ being the distance from x_1 (x_2) to the plane. We can rescale x such that the distance between x_1 and x_2 is equal to 2, so:

$$\boxed{||w|| = \frac{1}{m}}$$

Maximizing the margin « $2m$ » (and thus m) is equivalent to minimizing $||w||$. Knowing that w is a linear combination of x_i, the problem is reformulated as follows:

$$\boxed{y_i \times (\langle w, x_i\rangle + b) \geq 1; \text{ and } \quad \min(||w||^2)}$$

10.2 Finding the Best Possible Hyperplane: Penalization

The problem is of the type seen in the previous class, with a function to minimize, here the norm of w, and n constraints $(C_i(w, b))$. The Lagrangian is then written as:

$$\boxed{L(w, b, \alpha) = \frac{1}{2}||w||^2 + \sum_i \alpha_i(1 - y_i \times (\langle w, x_i\rangle + b))}$$

Now, the minimization will indeed give the best hyperplane, provided it exists. Most of the time, there will be no linear classifier (here, the hyperplane) that perfectly separates all training samples[15]. In this case, we seek a hyperplane that minimizes the classification error. Vapnik suggests defining the error as how much we need to move the misclassified samples to place them beyond the margin to reclassify them correctly. So, if we denote $\xi_i \geq 0$ as the distance for sample i that needs to be moved, then:

$$y_i \times (\langle w, x_i\rangle + b) \geq 1 - \xi_i \quad \text{(Soft SVM)}$$

(if $\xi_i = 0$, it's called « Hard SVM »). Of course, we want to penalize these rearrangements. So, we reformulate the problem as follows:

$$\boxed{\begin{cases} \text{Min}(\frac{1}{2}||w||^2 + C\sum_i \xi_i) \\ y_i \times (\langle w, x_i\rangle + b) \geq 1 - \xi_i \end{cases}}$$

10.3 Regularization

We have already seen that to ensure that fluctuations do not introduce too much error, we need to constrain the size of the class (space) in which we search for parameters. How does this translate into our problem? If we fix (w, c) then:

$$\xi_i \geq 1 - y_i \times (\langle w, x_i\rangle + b)$$

15. Note: except in a sufficiently high-dimensional space.

and we want $\xi_i \geq 0$, so:

$$\xi_i = \max(1 - y_i \times (\langle w, x_i \rangle + b), 0)$$

So, the minimization can be written by posing $\bar{y}_i = \langle w, x_i \rangle + b$ (linear regression):

$$\frac{1}{2}||w||^2 + C \sum_i \max(1 - y_i \bar{y}_i, 0)$$

Now, the function:

$$r(x, y, w, b) = r(y, \bar{y}) = \max(1 - y\bar{y}, 0)$$

is nothing but a convexified version of the binary classification risk $r(y, \tilde{y})$ with $\tilde{y} = \text{sgn}(\bar{y})$ as seen in the previous class. Therefore, the global risk:

$$\tilde{R}(w, b) = \sum_{i=1}^{n} r(x_i, y_i, w, b)$$

is convex, and the minimization problem appears as:

$$\text{Min}(\tilde{R}(w, b) + \frac{1}{2}||w||^2)$$

That is, the minimization of a **convex risk regularized by the norm of** $||w||^2$ which restricts the size of the hypothesis class on (w, b) and thus minimizes fluctuations. The problem is well-posed, and the calculation proceeds correctly.

10.4 Saddle Point Method

If we consider the case of Hard SVM ($\xi_i = 0$), then the Lagrangian is given by:

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 + \sum_i \alpha_i (1 - y_i \times (\langle w, x_i \rangle + b))$$

The saddle point is the point where we seek a minimum in the space (w, c) and a maximum for α_i with $\alpha_i \geq 0$. The respective gradients of the Lagrangian give:

$$\begin{cases} \sum_i \alpha_i y_i = 0 & (R1) \\ w = \sum_i (\alpha_i y_i) x_i & (R2) \\ 1 - y_i \times (\langle w, x_i \rangle + b) = 0 \quad (\forall \alpha_i \neq 0) & (R3) \end{cases}$$

The second equation shows that w is indeed a linear combination of x_i (cf. $\alpha_i y_i = \pm\alpha_i$). If $\alpha_i = 0$, then $1 - y_i \times (\langle w, x_i \rangle + b) < 0$, meaning the sample is beyond the margin, so the constraint does not apply. Thus, the number of « active » constraints ($\alpha_i > 0$) depends on the number of points that lie on the two boundary hyperplanes.

In the end, the algorithm will be driven not by the total number of samples but only by those that are closest to the hyperplane for which $\alpha_i > 0$, and thus:

$$\boxed{w = \sum_{i \in \mathcal{I}} (\alpha_i y_i) x_i}$$

Hence the term « Support Vectors of the samples in $\mathcal{I}$ ».

To solve the problem, we work in the dual space of the Lagrange multipliers (α_i). Taking into account $R1$ and $R2$, the Lagrangian expansion yields:

$$L(w, b, \alpha) = -\frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle + \sum_{i=1}^{n} \alpha_i$$

which needs to be minimized for the α_i. This can be done using conjugate gradient and stochastic gradient techniques.

For the case of « Soft SVM », we must minimize:

$$\frac{1}{2} ||w||^2 + C \sum_i \xi_i$$

which translates into an additional constraint on the α_i:

$$0 \leq \alpha_i \leq C$$

10.5 Generalization to the Non-Linear Case

We have seen that optimizing the margin is equivalent to convexifying the risk and regularization, and that ultimately, the problem depends only on the dot products $\langle x_i, x_j \rangle$, resulting in a linear boundary (an hyperplane). To find a non-linear boundary, we need to find a representation $z = \phi(x)$ that linearizes the problem. We have seen that everything comes down to the transformation:

$$\langle x_i, x_j \rangle \rightarrow \langle \phi(x)_i, \phi(x)_j \rangle = k(x_i, x_j)$$

with $k(x, x')$ being a measure of similarity between x and x' (a kernel).

The optimization is the same as in the linear case, and the solution for the estimator y_i is:

$$\begin{aligned}\tilde{f}(x) &= \text{sgn}(\langle w, \phi(x)_i \rangle + b) = \text{sgn}(\sum_j \alpha_j y_j \langle \phi(x), \phi(x)_j \rangle + b) \\ &= \text{sgn}(\sum_j \alpha_j y_j k(x_j, x) + b)\end{aligned}$$

In the 1990s-2000s, there was an optimistic wave. It was understood that not only could we find a very complex boundary, but we could also characterize the complexity of the boundary, where the dimension of the vector $\phi(x)$ can be very large. Let's look at two examples: **polynomial kernels and Gaussian kernels**.

10.5.1 Polynomial Kernels

Let's take an example of a kernel ($x_0 \equiv 1$):

$$\boxed{k(x, x') = (1 + \langle x, x' \rangle)^p = \left(\sum_{j=0}^{d} x_j x'_j \right)^p}$$

What are the $\Phi(x)$? In fact, if $\vec{j} = \{j_1, \ldots, j_p\}$ with all permutations:

$$\Phi(x) = \left\{\varphi_{\vec{j}}(x)\right\}_{\vec{j}=(0,\ldots,d)^p}$$

and

$$k(x,x') = \sum_{\vec{j}=(0,\ldots,d)^p} \prod_{i=1}^{p} x_{j_i} x'_{j_i} = \sum_{\vec{j}=(0,\ldots,d)^p} \prod_{i=1}^{p} x_{j_i} \prod_{i=1}^{p} x'_{j_i} = \langle \Phi(x), \Phi(x') \rangle$$

So, $\dim\Phi(x) = (1+d)^p = d'$, and

$$\boxed{\varphi_{\vec{j}}(x) = \prod_{i=1}^{p} x_{j_i}}$$

which means **all monomials of degree** p. The estimator $\tilde{f}(x)$ (in regression) can be expressed as:

$$\tilde{f}(x) = \sum_{\ell=1}^{n} \alpha_\ell y_\ell \sum_{\vec{j}=(0,\ldots,d)^p} \prod_{i=1}^{p} x_{j_i}^{(\ell)} \prod_{i=1}^{p} x_{j_i}$$

So, $\tilde{f}(x)$ is an arbitrary polynomial of degree p.

The trick is that it was not necessary to fit these monomials of degree p (we fitted the dual variables, i.e., α_i), and thus p can be very large even in dimension d, so $d' = (d+1)^p$ terms! Imagine with $d = 10^6$ for an image and $p = 5\ldots$ Then, in fact, all sums are bounded by n, so the number of scalar products is $n(n+1)/2$. **So the technique appears to be very powerful.** Is it a miracle, or is there a catch? Where is the price to pay? As one can imagine, it's in the fluctuations: **overfitting is quickly a problem as we have too many features and too few samples.** The reason for success is not only that the algorithm is stable but also that we can **always** put ourselves in a case where the hyperplane answers the problem on the training samples (fitting in regression or separation in classification). This comes from the following proposition:

Proposition: If the $\{x_i\}_{i\leq n}$ are linearly independent (and $d' \geq n$), then

$$\forall y_i,\ \exists w/\ \langle w, x_i \rangle = y_i$$

The matrix whose rows are the vectors x_i is invertible, so w exists. **So it is sufficient to place oneself in a sufficiently high dimension.**

We can even do a little better. If we take a sample at random (denoted x_1) as a reference for the coordinates, then

$$\{x_1 - x_i\}_{2 \leq i \leq n}$$

are linearly independent ($d' \geq n-1$), then

$$\forall y_i,\ \exists (w,b)/\ \langle w, x_i \rangle + b = y_i$$

Indeed, we reduce ourselves to the case of the theorem:

$$\begin{cases} \langle w, x_i - x_1 \rangle = y_i - y_1 \\ \langle w, x_1 \rangle + b = y_1 \end{cases}$$

The first equality indicates that w exists, and the second fixes b.

So in fact, we can always find $\phi(x)$ of sufficient dimension to find a suitable hyperplane. Returning to the original space of x, the boundary becomes non-linear.

However, the fluctuation error is

$$\varepsilon^2 \approx \frac{\log |\mathcal{H}|}{n} = \frac{d'}{n} = \frac{(1+d)^p}{n}$$

leading to a catastrophe, as the error can potentially be enormous. To control it, d' must be much less than n. So either d is small from the start, or feature reduction must be performed, even if the dimension of d is slightly increased to d'.

10.5.2 Gaussian Kernel

Let's imagine that we have a problem like the one shown in Figure 42.

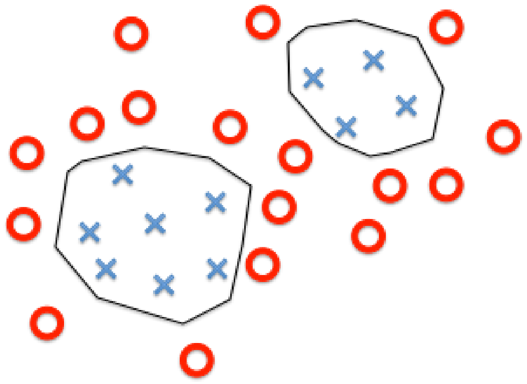

FIGURE 42 – Isolated non-convex clusters, non-linear and non-convex boundary.

The boundary is clearly non-convex and cannot be generated by a polynomial kernel. The kernel we will consider is defined by the Radial Basis Functions (**RBF**):

$$\boxed{k(x_1, x_2) = e^{-\frac{1}{2\sigma^2}||x_1 - x_2||^2}}$$

What is $\Phi(x)$ in this case? It is shown that one solution can be written as:

$$\varphi_{\vec{k}}(x) = e^{-\frac{1}{2\sigma^2}||x||^2} \prod_{\vec{j}=(0,\dots,d)^k} x_{j_i}$$

but there are infinitely many k, so $\Phi(x)$ **is of infinite dimension**. However, we notice that this kernel is calculated very simply, whereas if we went through the $\varphi_{\vec{k}}(x)$ step, it would have required an infinite number of operations!

If $||x_1 - x_2|| \ll \sigma$, then

$$k(x_1, x_2) \approx 1 - ||x_1 - x_2||^2/(2\sigma^2)$$

so the kernel behaves like a local Euclidean distance, and we can determine a local linear classifier (regression). Gradually, we will define a boundary that can be of any structure.

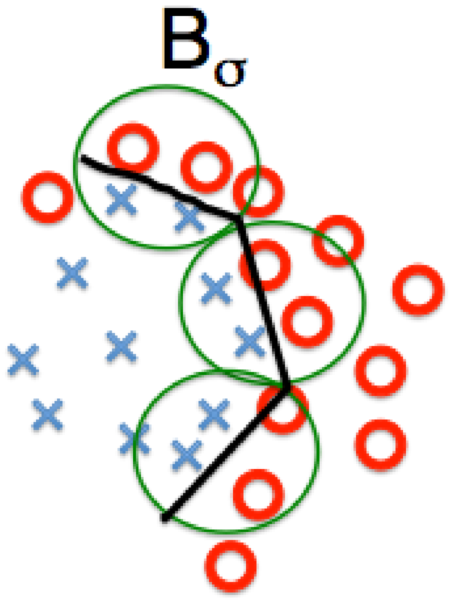

Figure 43 – Through progressive application of a filter of size σ, we linearize locally, and ultimately obtain a complex boundary.

But what is the price to pay? Always the bias-variance trade-off. If σ is too small, there are not enough samples (**remember: we are in high dimensions, the 2D drawing is very misleading**). So σ must be large enough so that the volume of the ball is of the same order of magnitude as the size of the space, and the Gaussian classifier becomes a linear classifier! Then we cannot account for details. σ **is determined by cross-validation** in practice, and we can obtain not linear classifiers but slightly curved ones locally. But there is no miracle.

10.6 NDJE. Comments

Before concluding this course, I wanted to revisit the SVM technique from a completely different perspective. So, consider $\{x_i, y_i\}_{i\leq n}$ with $y_i \in \{0,1\}$. The idea is a linear parametric probabilistic model (I'm trying to keep the notations related to S. Mallat's course):

$$h_w(x) = P(y=1|x;w) = 1 - P(y=0|x;w)$$

So, the probability that y is 0 or 1 is given by:

$$P(y|x;w) = h_w(x)^y(1-h_w(x))^{1-y}$$

We then form the likelihood as follows, assuming that all samples are independent:

$$\mathcal{L}(w) = \prod_{i=1}^{n} P(y_i|x_i;w)$$

and seek a maximum to find the value of w. To use minimization algorithms, we instead use $-\log\mathcal{L}(w)$ normalized by the number of samples n. Thus, we form the function:

$$J(w) = -\frac{1}{n}\sum_{i=1}^{n}\{y_i\log(h_w(x_i)) + (1-y_i)\log(1-h_w(x_i))\}$$

Now, we need to provide an expression (model) for $h_w(x)$ (i.e., $P(y=1|x;w)$). In principle, in connection with S. Mallat's course, we would look for something like:

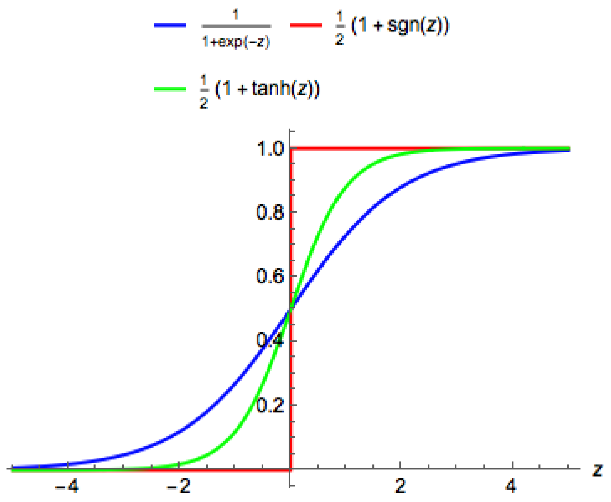

FIGURE 44 – Several functions $h_w(z)$ for likelihood minimization.

$$h_w(x) = \frac{1}{2}(1 + \mathrm{sgn}(\langle x, w \rangle))$$

(Note that b is integrated into the dot product.)

But this expression is not suitable as an argument for log. This is why D. Cox (1958) introduces the *logistic function* (Figure 44) in this type of problem:

$$h_w(x) = \frac{1}{1 + e^{-\langle x, w \rangle}}$$

Alternatively, we could introduce the *"tanh"* function:

$$h_w(x) = \frac{1}{2}(1 + \tanh(\langle x, w \rangle))$$

To conclude on **logistic regression**, we add regularization to correct overfitting by introducing an $L2$ norm penalty, so $J(w)$ becomes:

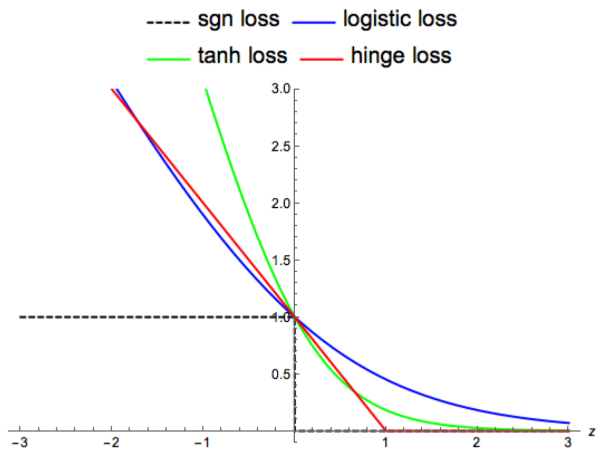

FIGURE 45 – Candidate risk/cost functions.

$$\boxed{J_r(w) = -\frac{1}{n}\sum_{i=1}^{n}\{y_i \log(h_w(x_i)) + (1-y_i)\log(1-h_w(x_i))\} + \lambda\sum_{j=1}^{d}|w_i|^2}$$

For the *"non-linear"* step with kernels, I pointed out that the previous expression could be generalized as follows (in the literature, λ is taken out of the $\{\}$ and $1/(2\lambda) = C$):

$$\boxed{J_r(w) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\{y_i\mathrm{cost}_1(\langle x_i, w\rangle) + (1-y_i)\mathrm{cost}_0(\langle x_i, w\rangle)\}}$$

The functions $\mathrm{cost}_1(z)$ for the logistic and tanh expressions are shown in blue and green, respectively, in Figure 45.

Let's consider the case of the red curve, which represents a **threshold effect**. In S. Mallat's course, this is referred to as *"hinge loss"* associated with the margin, given the geometric interpretation:

$$\mathrm{cost}_1^{hinge}(\langle x_i, w\rangle) = \begin{cases} 1 - \langle x_i, w\rangle & \langle x_i, w\rangle < 1 \\ 0 & \langle x_i, w\rangle \geq 1 \end{cases}$$

These three functions are upper bounds of the *"sign"* function shown in black in Figure 45, as in S. Mallat's course. They serve to convexify the problem [16].

In the case of minimizing $J_r(w)$, to interpret the choice of cost_1^{hinge}, I considered the case where $C \gg 1$. The problem then becomes a minimization:

$$\min_w \left(\frac{1}{2}||w||^2\right)$$

under **the constraints** (which nullify the C term) as follows:

$$\begin{cases} \langle x_i, w\rangle \geq 1 & \text{if } y_i = 1 \\ \langle x_i, w\rangle \leq -1 & \text{if } y_i = 0 \end{cases}$$

This problem is identical to the one mentioned by S. Mallat, adapted for $y_i = \{0, 1\}$. **So, minimizing $J_r(w)$ yields the same result as SVM in the case $C \gg 1$.**

11. Gradient Descent Method and an Introduction to Neural Networks

11.1 Gradient Descent Optimization

Minimization algorithms are not just tools; they are conditions of possibility. They make learning possible. So, we have the following problem:

$$\min_{w \in \mathcal{H}} R(w)$$

To achieve this, we calculate gradients:

$$\vec{\nabla} R(w) = \left(\frac{\partial R(w)}{\partial w_k}\right)_{k \leq d} \equiv \partial_w R(w)$$

16. Note that we rescaled the logistic/tanh cost functions by $1/\log(2)$ to do this, which does not contribute to the minimization.

and through iteration, starting from an initial value $w = w_0$, we obtain:

$$\boxed{w_{k+1} = w_k - \eta \vec{\nabla} R(w_k)}$$

In fact, the derivative of $R(w)$ with respect to any unit vector $\vec{n}$ is given by:

$$\frac{\partial R(w)}{\partial \vec{n}} = \vec{\nabla} R(w).\vec{n}$$

So, the direction of the gradient maximizes the derivative, which is the steepest local ascent.

The existence of gradients everywhere is guaranteed by the smoothness (at least Lipschitz) of $R(w)$, and the presence or absence of local minima is related to the convexity of $R(w)$. The convergence rate is determined by the Hessian (second derivative).

11.1.1 Quadratic Risk

$$R(w) = \frac{1}{2}\langle \mathbf{A}w, w\rangle + \langle b, w\rangle$$

with $\mathbf{A}$ being symmetric and positive definite (*typically a covariance matrix*). Therefore,

$$\partial_w R(w) = \mathbf{A}w + b$$

A necessary condition for w^* to be the solution is that

$$\partial_w R(w^*) = 0 \Rightarrow w^* = -\mathbf{A}^{-1}b$$

Hence, the convergence rate can be studied by looking at the deviation at each step between w_k and w^*:

$$w_{k+1} - w^* = (\mathbf{I} - \eta\mathbf{A})(w_k - w^*)$$

Thus, in terms of norm:

$$||w_{k+1} - w^*|| \leq ||(\mathbf{I} - \eta\mathbf{A})|| \times ||w_k - w^*||$$

If we denote $||(\mathbf{I} - \eta\mathbf{A})|| = \rho$, then:

$$||w_{k+1} - w^*|| \leq \rho^k ||w_0 - w^*||$$

Convergence is ensured if $\rho < 1$, and we want it to be as small as possible by tuning η. Now,

$$||(\mathbf{I} - \eta\mathbf{A})|| = \max(|1 - \eta\sigma_i|) = \max(|1 - \eta\sigma_{\text{min}}|, |1 - \eta\sigma_{\text{max}}|)$$

So, by taking:

$$\eta = \frac{2}{\sigma_{\text{max}}^2 + \sigma_{\text{min}}^2}$$

we achieve the best ρ:

$$\rho = \frac{\sigma_{\text{max}}^2 - \sigma_{\text{min}}^2}{\sigma_{\text{max}}^2 + \sigma_{\text{min}}^2}$$

We can see that if we condition the matrix A such that $\sigma_{\text{max}} \approx \sigma_{\text{min}}$, we get rapid convergence. All of this discussion generalizes by taking $\mathbf{A}$ as the Hessian matrix (second derivatives) of the risk (making the problem locally quadratic).

11.1.2 Batch vs. Stochastic Gradient

In the case of learning with n samples, the empirical risk is calculated as the average risk over all samples, and n can be very large. However, at each step w_{k+1} requires the calculation:

$$\partial_w \tilde{R}(w) \propto \sum_{i=1}^{n} \partial_w r(y_i, x_i; w)$$

which involves n operations. This is called a **Batch of n elements**.

For **stochastic gradient**, the strategy is to calculate an estimate of the stepping direction using **one element**:

$$w_{k+1} = w_k - \eta \partial_w \tilde{r}(y_i, x_i; w_k)$$

We still want to minimize the true risk (*independently of the chosen technique*), which is:

$$\min_w R(w) = \min_w E_{Y,X}(r(Y, X, w))$$

and both methods guarantee this.

What is the advantage? If there are redundancies in the samples, the Batch method is not efficient because it performs unnecessary calculations. In the Stochastic method, redundant samples will be used at different steps (k) of the algorithm. However, Stochastic Gradient is very noisy compared to Batch Gradient, resulting in a different convergence rate. In a strongly convex case (where the Hessian is well-conditioned):

$$E(\tilde{R}(w_k) - \min_w \tilde{R}(w)) \approx \begin{cases} O(1/k) & \text{(StochasticGradient)} \\ O(\rho^k) & \text{(BatchGradient)} \end{cases}$$

So, the « Stochastic Gradient » method is faster with a small number of iterations, but the « Batch Gradient » converges faster with a large number of iterations. This is schematically represented in Figure 46.

Some methods attempt to combine the advantages of both methods. See, for example, the work of F. Bach et al. (SAG method) [17]

17. See https://www.di.ens.fr/%7Efbach/Defazio_NIPS2014.pdf.

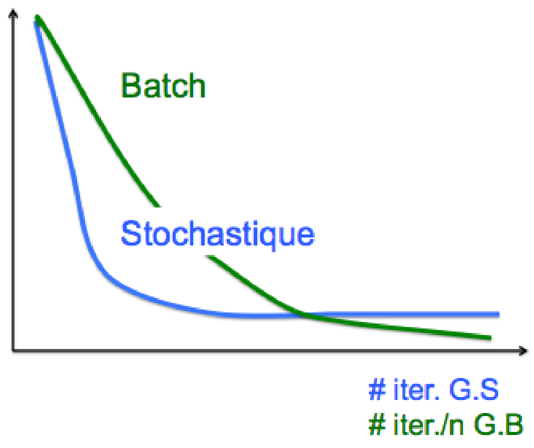

Figure 46 – Diagram illustrating the convergence speed of stochastic gradient descent and batch gradient descent.

11.2 Data Representation $\phi(x)$ and Introduction to Neural Networks

11.2.1 Introduction: What Does One Neuron and One Neural Network Do?

So far, the **representation** $\phi(x)$ has been **separate** from the **classification/regression** operation. We make a choice *a priori*, for example, the type of kernel (e.g., polynomial, Gaussian) and its internal parameters (e.g., degree p, width σ). Then, in the case of classification, we provide a linear estimator:

$$\tilde{h}(x) = \langle w, \phi(x) \rangle + b$$

The lingering question is: how do we choose $\phi(x)$? There are two possibilities: either we have information about the data, and we can make an "informed" choice, or we let the algorithm learn the right representation. **Neural networks (referred to hereafter as NN: Neural Net, or MLP: Multi-Layer Perceptron) learn both the representation and the classifier simultaneously.**

The representation is a **cascade of linear classifiers** passed through **non-linear filters** (Figure 47).

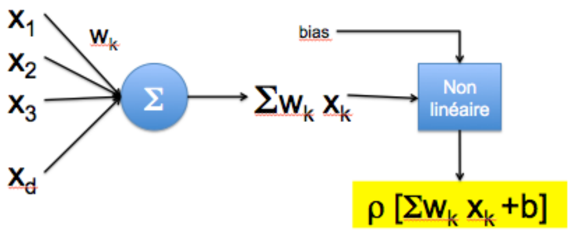

FIGURE 47 – Operations performed by one neuron.

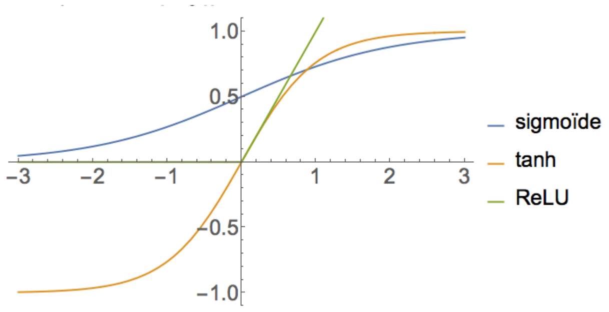

FIGURE 48 – Activation functions (ρ) of a neuron.

Regarding the non-linear function $\rho(x)$, several choices have been/are being used. Three typical ones (with ReLU: rectified linear unit = $max(0, x)$) are presented in Figure 48.

The result of a linear classifier (w, b) separates into two classes, and the function ρ in the case of sigmoid/tanh reaffirms the class choice with a "fuzzy" transition zone. With ReLU, below the hyperplane, we have classification, and above, we have linear regression. In both cases, the neuron performs a separation of the space.

An MLP (Figure 49) constructs a representation $\phi(x)$ and a final linear regression, possibly with classification.

If x_0 is the input, with 2 hidden layers and 1 final regression, and we differentiate the non-linear operations at each activation step of the neurons:

$$\tilde{y} = \rho_3 \left[W_3 \rho_2 \left[W_2 \rho_1 [W_1 x_0 + b_1] + b_2\right] + b_3\right]$$

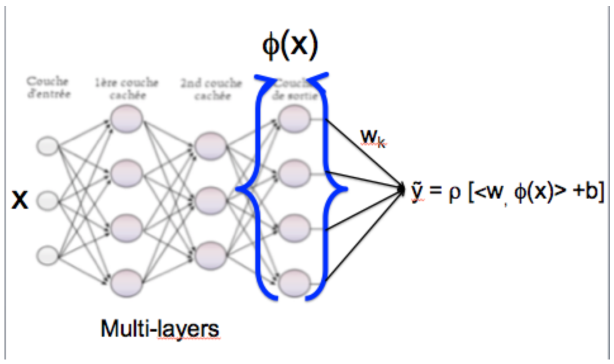

FIGURE 49 – The network produces both the representation $\phi(x)$ and the final classification.

The challenge is to perform **optimization for all neurons**, meaning learning the representation $\phi(x)$ while performing the final regression. We will minimize a risk as we have done so far, but now the number of parameters will be very large. How can we understand the architectures of MLPs? This is what Y. LeCun, G. Hinton, Y. Bengio, and others have done using convolutions and input information to reduce connections.

11.2.2 Single Hidden Layer Network

In the case of a single hidden layer, we have an important result. So, consider the network in Figure 50.

$$\tilde{h}(x) = \tilde{y} = \rho\left[\sum_{k=1}^{M} \alpha_k \, \rho\left[\sum_{m=1}^{d} w_{k,m} x_m + b_k\right] + b\right] \approx f(x)$$

A quick reminder: when performing classification, we define an approximation class:

$$\mathcal{H} = \{h_w / \text{algorithm parameters (w)}\}$$

and minimize an empirical risk, possibly regularized:

$$\min_{h \in \mathcal{H}} \tilde{R}(h_w)$$

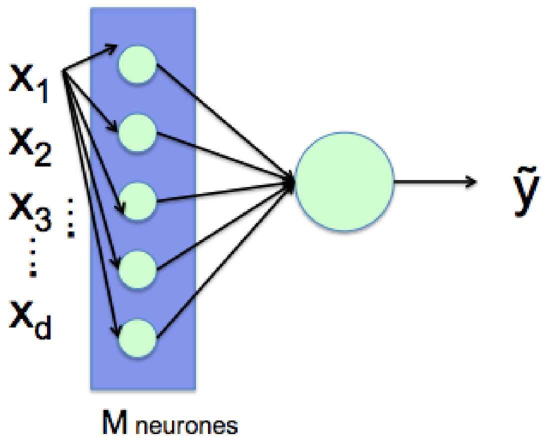

Figure 50 – Single hidden layer network with M neurons.

But is $\tilde{y} = \tilde{h}(x)$ close to $y = f(x)$? The class $\mathcal{H}$ must be large enough to fit any function but not too large to avoid overfitting.

In the case of a single hidden layer, do we have enough parameters? In fact, we can ask what family of functions is generated by this type of architecture. What is the regularity of $f(x)$? In this case, the last non-linearity does not change the class (regularity), so we can focus on the representation $\phi(x)$ such that:

$$\phi(x) = \sum_{k=1}^{M} \alpha_k \; \rho \left[\sum_{m=1}^{d} w_{k,m} x_m + b_k \right] = \sum_k \alpha_k \; \rho \left[\langle w_k, x \rangle + b_k \right]$$

In other words, if we leave (α_k, w_k, b_k) free, what space do we generate? The answer is simple and spectacular: **any continuous function can be approximated with a single hidden layer NN!** [18].

Theorem (1989-93): If $\rho \in C(\mathbb{R})$ (continuous) and **is not a polynomial**, then if $f \in C(\mathbb{R}^d)$, f can be approximated with arbitrary precision by a single hidden layer NN (NN-1 hidden layer) to say that:

18. Note, S. Mallat talks about a 2-layer NN

$$\forall \varepsilon > 0, \forall K(\text{compact}) \subset \mathbb{R}^d \Rightarrow \exists \phi \ (\text{NN} - 1 \text{ hidden layer}) \ / \ \max_{x \in K} |f(x) - \phi(x)| < \varepsilon$$

The proof will be presented next year. However, this result, while spectacular, does not provide a general solution to the dimensionality problem because imagine that **the size of the hidden layer M explodes in dimension d**, then we have not gained anything: **the fluctuation error explodes** as the size of $\mathcal{H}$ grows.

Let's take an example to understand why the theorem does not provide a general solution to the dimensionality problem. Consider $\rho(x) = e^{ix}$ (or $\cos(x)$), i.e., the kernel of Fourier series.

$$\phi(x) = \sum_{k=1}^{M} \alpha_k e^{i(\langle w_k, x\rangle + b_k)} = \sum_{k=1}^{M} \alpha_k e^{ib_k} e^{i\langle w_k, x\rangle}$$

If we consider the compact $x \in [0,1]^d$, we find a Fourier series decomposition that we have seen before:

$$\phi(x) = \sum_{w_k \in \mathbb{Z}^d} \hat{\phi}(w_k) e^{i2\pi\langle w_k, x\rangle}$$

If the function's regularity is Lipschitz, it guarantees a decay of Fourier coefficients, and we can truncate the series:

$$0 \leq |w_k| \leq C$$

But the number of w_k grows as C^d, **we fall back into the curse of dimensionality**.

This result generalizes to any continuous function ρ (except polynomials), and it shows that the number of parameters (neurons) explodes. So, a single hidden layer NN can approximate any continuous function, but potentially at the expense of an explosion of parameters in high dimension d.

The challenge of architectures is therefore to solve the dimensionality problem while being able to approximate a wide class of functions with hidden regularity that the neural network can capture. Why it works completely is not yet understood.