CR12: Statistical Learning & Applications

Kernel-based Methods and Stochastic Learning

Lecturer: Zaid Harchaoui *Scribes: Massimiliano Fasi & Sébastien Maulat*

1 Kernel-based Methods

1.1 Introduction

The procedure of *lifting* we discussed at the end of the last lecture is powerful, and represents a workaround to get rid of the constraint of linear separability of the training samples, that constitutes one the main limitations of the *perceptron*, that is just a linear classifier. Nevertheless, even tough it allows us to work even with non separable training sets, on the other hand it suffers seriously the growth of dimensionality of the learning space that linearly depends on the cardinality of the training set. This drawback makes that trick impractical to use when dealing with most of the real scale problems, because of the performance loss introduced by working with high-dimensional object.

Nevertheless, the possibility of using a linear classifier with non-linearly separable data paying a – possibly slight – increase of domain's dimensionality, seems worth of attention. We will briefly discuss again the lifting technique later, under the light of what we will introduce in the following paragraphs.

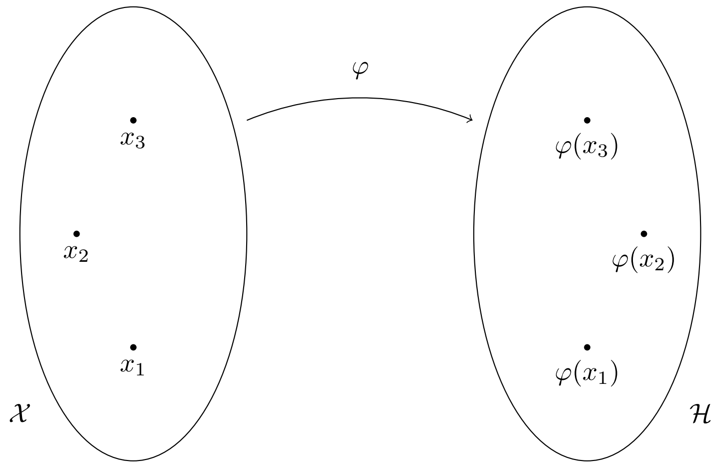

Figure 1: A simple representation of the situation

Let us begin with an informal introduction to the basic notions and ideas of the kernel-based learning methods. Let us suppose that we are working with a training set $\mathcal{D}_n \subseteq \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is a general set and $\mathcal{Y}$ the set of *targets*, i.e. the possible responses of the classification algorithm, that are usually -1 and $+1$ when the goal is a binary classification. It can happen that $\mathcal{X}$ is weakly structured or, as we have just seen, that data are not linearly separable there, and in such cases mapping the points of the training set into another space can be the key. For *kernel-based methods* we will discuss here, we will try to map points of $\mathcal{X}$ into a vector space $\mathcal{H}$, called *feature space* and equipped with a scalar product[1], through a function $\varphi : \mathcal{X} \to \mathcal{H}$ called *feature map*. In this context, we will require a map

$$K : \mathcal{X} \times \mathcal{X} \to \mathbb{R},$$

that will be called *kernel*, satisfying for all $x, x' \in \mathcal{X}$

$$K(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}}$$

where $\langle \cdot, \cdot \rangle_{\mathcal{H}} : \mathcal{X} \times \mathcal{X} \to \mathcal{H}$ denotes, and will denote from now on, the scalar product of $\mathcal{H}$. The main advantage of such approach, usually called *kernel trick*, is the possibility to work in the feature space $\mathcal{H}$ without having to know the value of $\varphi(x)$ on any of the $x \in \mathcal{X}$. Once the kernel is given, indeed, it will not be required to compute the images of the points: it will suffice to evaluate K for every pair of elements of the domain.

Now, it is easy to see that the lifting is nothing more than an *explicit* feature map, since in that case we defined an explicit way for computing a vector of the higher-dimensional space from a sample of the training set. This way of proceeding is precisely the one the use of a kernel is meant to avoid. Indeed, these kind of methods are based on the observation that some learning algorithms – and the preceptron is one of these, as we have seen – do not need to explicitly know the coordinates of a training point in the feature space, but just need a way to compute its scalar product with a given vector.

1.2 Definitions

In the rest of this section, we will formally introduce the concepts, starting with the definition of a particular kind of kernel that, as we will see, has a nice characterization in the context of functional analysis.

Definition (Positive semi-definite kernel)**.** *Let $\mathcal{X}$ be an arbitrary set. A form $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive semi-definite kernel, often abbreviated as* ***psd****, if*

1. *it is symmetric, i.e. $\forall x, x' \in \mathcal{X}$ it holds that $K(x, x') = K(x', x)$;*
2. *it is positive definite, i.e $\forall n \in \mathbb{N}$, $\forall \mathbf{a} \in \mathbb{R}^n$ it holds that $\mathbf{a}^\intercal \mathbf{K}\, \mathbf{a} \geq 0$ where $k_{i,j} = K(x_i, x_j)$.*

According to the above definition, it is easy to see that a positive semi-definite kernel is just a semi-definite positive matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$, whose entries are given by the evaluation of K on each pair of vectors from a chosen subset $X \subset \mathcal{X}$ such that $|X| = n$.

A first – quite straightforward – example of kernel is represented by the *linear kernel*, that is simply given by the inner product of the domain. If we consider a vector space $\mathcal{Y}$ the linear kernel K has the form

$$K(y, y') = y^\intercal y' = \langle y, y' \rangle_{\mathcal{Y}},$$

where $y, y' \in \mathcal{Y}$.

The second object we will define is a functional Hilbert space of real-valued functions.

Definition (Reproducing Kernel Hilbert Space)**.** *Let $\mathcal{X}$ be an arbitrary set. The Hilbert space $\mathcal{H} = \{f : \mathcal{X} \to \mathbb{R}\}$ is a reproducing kernel Hilbert space, often abbreviated* ***rkhs*** *if and only if*

1. *$\forall x \in \mathcal{X}$, each form $K(x, \cdot) : t \mapsto K(x, t)$ is in $\mathcal{H}$*

[1]We will actually consider Hilbert spaces.

2. *$\forall x \in \mathcal{X}$ and for all $f \in \mathcal{H}$ we have that $f(x) = \langle f, k(x, \cdot) \rangle_{\mathcal{H}}$.*

According to the above definitions, in a **krhs** each functional $k(x, \cdot)$ is continuous with respect to the point-wise evaluation, i.e. if seen as an operator F_x as x varies in $\mathcal{X}$. Moreover, each evaluation of $f(x)$ with $x \in \mathcal{X}$ and $f \in \mathcal{H}$ can be computed as the scalar product of f and a functional of the form $k(x, \cdot)$.

1.3 Aronszajn's theorem

Now that we all both the ingredients, we can mix them up in the following theorem, due to Aronszajn, that gives us a neat way for characterizing a **rkhs**.

Theorem (Aronszajn, 1950)**.** *Let $\mathcal{X}$ be a compact set. Then $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a partial semi-definite kernel if and only in there exist a Hilbert space $\mathcal{H}$, a feature map such that for any $x, x' \in \mathcal{X}$ it holds that $K(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}}$.*

Proof. We well consider here just the proof of a very specific case, namely that of the set $\mathcal{X} = \{x_1, \ldots, x_n\}$ such that $|\mathcal{X}| = n < \infty$, but the proof can be extended to countable and also compact sets.

In order to prove the first implication, let us suppose that K is a positive semi-definite kernel. Then the matrix $\mathbf{K} = [k(x_i, x_j)]_{i,j \in [\![1,n]\!]}$ is symmetric positive semi-definite and hence diagonalizable, i.e. there exist $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$ and $\mathbf{u}_1, \ldots, \mathbf{u}_n \in \mathbb{R}^n$ such that

$$K = \sum_{p=1}^{n} \lambda_p \mathbf{u}_p {\mathbf{u}_p}^{\mathsf{T}}.$$

Then, we can write each entry of $\mathbf{K}$ as follows

$$\begin{aligned} k_{i,j} &= K(x_i, x_j) \\ &= \sum_{p=1}^{n} \lambda_p u_{p_i} {u_{p_j}}^{\mathsf{T}} \\ &= \left(\sqrt{\lambda_1} u_{1_i}, \ldots \sqrt{\lambda_n} u_{n_i}\right)^{\mathsf{T}} \left(\sqrt{\lambda_1} u_{1_j}, \ldots \sqrt{\lambda_n} u_{n_j}\right), \end{aligned}$$

where u_{i_j} represents the j-th component of the vector $\mathbf{u}_i$. To conclude the proof, it suffices to take, as i varies in $[\![1, n]\!]$, the definition

$$\varphi(x_i) = \left(\sqrt{\lambda_1} u_{1_i}, \ldots \sqrt{\lambda_n} u_{n_i}\right).$$

For the converse, let us consider a kernel $k(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle_{\mathcal{H}}$ and prove that it is positive semi-definite. The symmetry is a direct consequence of the commutativity of the scalar product, while the positive definiteness descend from the fact that a Hilbert space is also an Euclidean space, with respect to to the metric induced by the norm $|| \, x \, ||_{\mathcal{H}} = \langle \varphi(x), \varphi(x) \rangle_{\mathcal{H}}$. □

1.4 The Kernel Perceptron

The following algorithm is a perceptron that, given a training set and a kernel function, is able to return a function f in some Hilbert space $\mathcal{H}$.

To convince ourselves that the above algorithm actually computes a function f, linear in $\mathcal{H}$, that acts as a non-linear classifier in $\mathcal{X}$ provided that it can works on some feature map $\varphi : \mathcal{X} \to \mathcal{H}$, let us suppose to

Algorithm 1 The Kernel Perceptron

```
Input: $\mathcal{D}_n = \{(x_1, y_1), \dots, (x_n, y_n)\} \subset \mathcal{X} \times \{-1, +1\}$ and $k(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
Output: $f_w(\dots) \in \mathcal{H}$
  $f_w = 0$
  while $\exists i \in [\![1, n]\!] : y_i f_w(x_i) \leq 0$ do
    $f_w(\cdot) \leftarrow f_w(\cdot) + y_i k(x_i, \cdot)$
  end while
  return $f_w$
```

run the perceptron in $\mathcal{H}$ on modified instance of the training set $\widetilde{\mathcal{D}_n} = \{(\varphi(x_1), y_1), \dots, (\varphi(x_n), y_n)\}$. Let $\mathbf{w} \in \mathcal{H}$ be the output of an execution of the algorithm on $\widetilde{\mathcal{D}_n}$, then for sure

$$\mathbf{w} = \sum_{i=1}^{n} \beta_i \varphi(x_i).$$

Looking at the standard algorithm presented in the last lecture, we can note that the only operation performed during its execution that involved the data points is the scalar product $\langle \mathbf{w}, \varphi(x_i) \rangle$, that can be rewritten as follows

$$\begin{aligned} \langle \mathbf{w}, \varphi(x_j) \rangle &= \left\langle \sum_{i=1}^{n} \beta_i \varphi(x_i), \varphi(x_j) \right\rangle \\ &= \sum_{i=1}^{n} \beta_i \langle \varphi(x_i), \varphi(x_j) \rangle \\ &= \sum_{i=1}^{n} \beta_i k(x_i, x_j) \end{aligned}$$

in order avoid explicitly calculating the feature map, thank to the use of the kernel function. By mean of that observation, we can say that the output of the algorithm 1.4 is

$$f_w(\cdot) = \sum_{i=1}^{n} \beta_i k(x_i, \cdot),$$

as we expected.

As an example of use of the algorithm let us consider the problem of binary classification of graphs: we are given n graphs $x_1, \dots, x_n$ such that each of them has at least a path of length 2, and we want to classify them using a similarity metric. Since we cannot run a perceptron on set of graphs, we need another way to describe them, and the one we are going to give reveals itself to be very suitable to define a positive semi-definite kernel. Let us consider, for each graph $x_k = (\mathcal{V}_k, \mathcal{E}_k)$, the enumeration $\mathcal{Z}_k = \{p_1, \dots p_{n_k}\} \in \mathcal{P}\left(\{0,1\}^3\right)$ of all the paths of length 2 and the following kernel function

$$k(x_i, x_j) = \frac{1}{n_i \, n_j} \sum_{l=1}^{N_i} \sum_{m=1}^{N_j} h(p_l, p_m) \tag{1}$$

where $h(w_1, w_2)$ is the function that computes the Hamming distance between two words $w_1, w_2 \in \Sigma^*$ for some alphabet Σ. Is it straightforward to see that (1) is a positive definite kernel, since the symmetry is consequence of the commutative and associative properties of the sum, and the positive definiteness comes directly from the non-negativity of the used metric.

2 Stochastic learning

Stochastic learning is a way to implement machine learning algorithms using stochastic gradient descent.

- Input : $(x_1, y_1), \dots, (x_n, y_n) = z_1, \dots, z_n$, with $z_i \sim \mathbb{P}_z$ ($\mathbb{P}_z$ unknown).
- The loss function : $l(x, y; f(x))$ measures the cost incurred when predicting $f(x)$ whereas the right label is y.

Learning is solving the stochastic optimization problem :

$$\min_f \mathbb{E}_{x,y \sim \mathbb{P}_{x,y}} l(x, y; f(x))$$

We shall index the functionals f as f_w over a finite-dimensional space $w \in \mathbb{R}^n$, and set practical conditions on l for the optimization problem to be feasible (f convex).

Example. *For classification, the loss functions considered*[2] *write as:*

- *the misclassification loss :* $l(x, y; w) = \mathbb{1}\{y \cdot w^\intercal x \leq 0\}$
- *the linear hinge loss :* $l(x, y; w) = \max(0, 1 - y \cdot w^\intercal x)$ *also denoted by* $(1 - y \cdot w^\intercal x)_+$
- *the squared hinge loss :* $l(x, y; w) = (1 - y \cdot w^\intercal x)_+^p$ *with* $p = 2$

The aim of this section is to provide generic algorithms for minimizing convex (but not necessarily differentiable) functions l. For this, let us recall some convexity results.

Convexity and (sub)gradients.

Definition (Convexity). *Given a convex subset* $A \subseteq \mathbb{R}^d$*, a function* $f : A \to \mathbb{R}$ *is convex if it satisfies :* $\forall u, v \in A, \forall \alpha \in [0, 1], \quad f(\alpha u + (1 - \alpha)v) \leq \alpha f(u) + (1 - \alpha) f(v)$.

Definition (gradient). *For differentiable functions* $f : \mathbb{R}^d \to \mathbb{R}$*, we can define the gradient* $\nabla f(w)$ *also written* $f'(w)$ *such as :*

$$f(u) = f(w) + \nabla f(w)^\intercal (u - v) + \underbrace{o(...)}_{\geq 0 \text{ if } f \text{ is convex}} \tag{2}$$

This property is far easier to use in practice to compute a gradient (by hand) than the explicit expression : $\nabla f = (\frac{\partial f}{\partial w_1}, \dots, \frac{\partial f}{\partial w_d})$.

For not differentiable functions, we introduce a similar notion of subgradient, which intuitively denotes "any vector satisfying the inequality (with $\geq$) corresponding to equation (2)".

Definition (subgradient). *Formally, the subgradient* $\partial f(w)$ *is the set of vectors* v *such that* $\forall u \in A, f(u) - f(w) \geq v^\intercal (u - w)$.

[2] written with Euclidean dot-products here, but everything can readily be extended with Hilbertian dot-products using the kernel trick

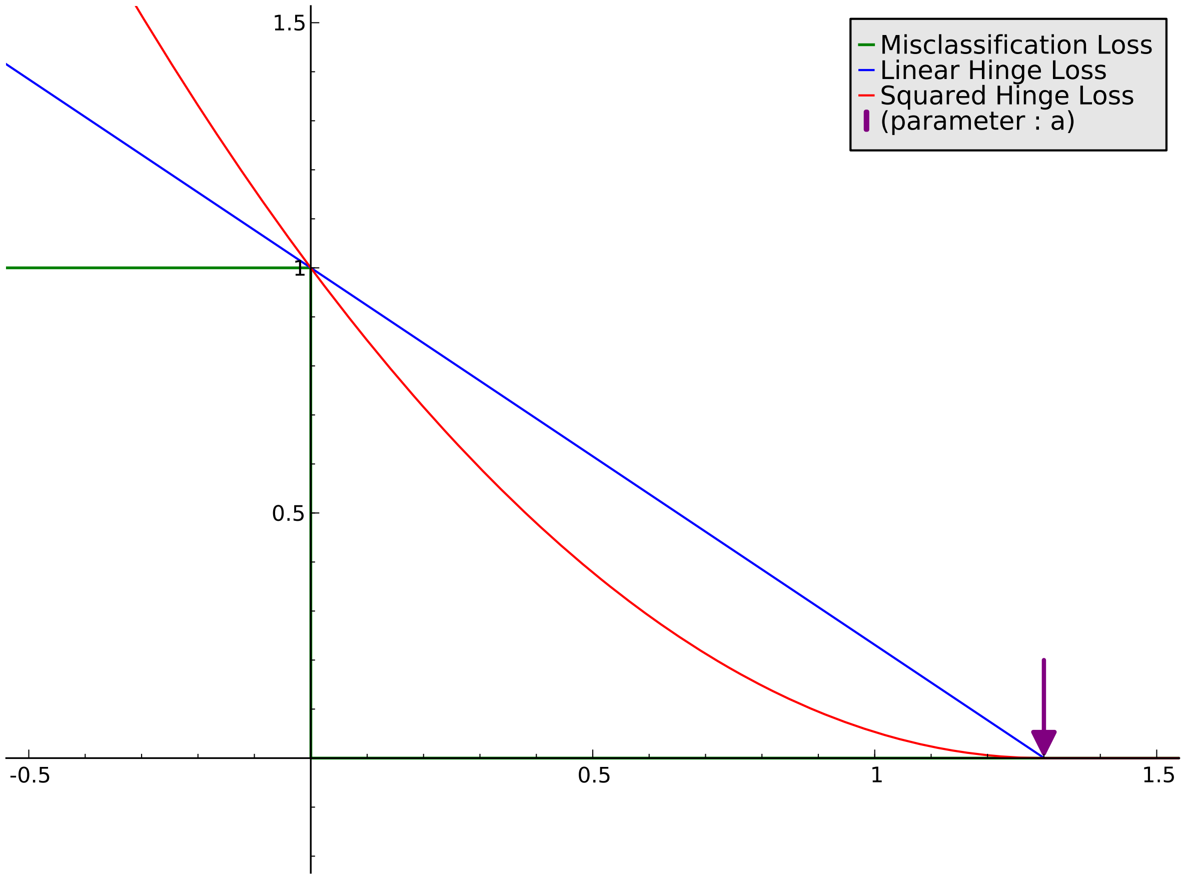

Figure 2: Usual Loss Functions

Proposition.

- *If f is differentiable at w (and convex), then $\partial f(w) = \{\nabla f(w)\}$.*
- *If $f = \max_{1\leq i\leq n} g_i(w)$ where the g_i are convex, then $\forall i, \nabla g_i(w) \in \partial f(w)$.*

Lemma. *The following properties imply the convexity of f :*

- *f is differentiable and satisfies : $\forall w, \nabla f(w) \in \partial f(w)$.*
- *f admits a Hessian $\nabla^2 f(w)$ that is positive definite.*
- *$f = g \circ h$ where g is convex and h linear.*
- *f is the maximum of linear functions.*

Example.

- *The logistic loss (also called logit) is defined by :*

$$l(x, y; w) = \log\left(1 + \exp(-y \cdot w^\intercal x)\right)$$

It is convex since $\log(1+e^{-az})$ is convex in z independently from a (check that $f''(z) = \dfrac{a}{1+\exp(az)}\exp(az)a$), and $w^\intercal x$ is linear (in w).

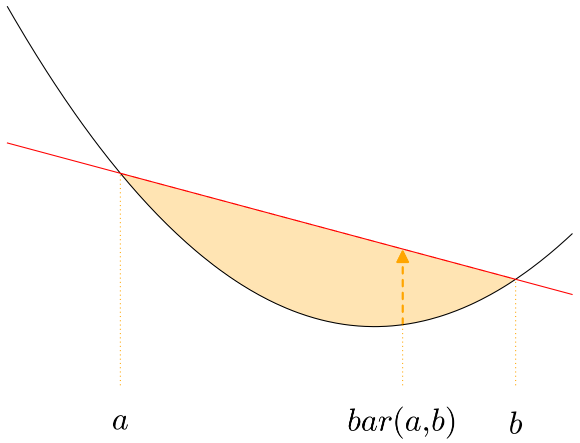

Figure 3: Convexity

- $l(x, y; w) = (1 - y \cdot w^\intercal x)_+ = \max(\underbrace{0, \dots}_{linear})$ *is convex. Still, it is not differentiable at one point (see on figure 2). We have* $\begin{cases} 1 - y \cdot w^\intercal x > 0 & \Rightarrow -y \cdot x \in \partial l \\ 1 - y \cdot w^\intercal x < 0 & \Rightarrow 0 \in \partial l \end{cases}$*. Show as an exercise that*

$$\partial l(w) = \begin{cases} \{-y \cdot x\} & \textit{if } l(x, y; w) > 0 \\ \{0\} & \textit{if } l(x, y; w) < 0 \\ \{-\alpha y \cdot x; 0 \leq \alpha \leq 1\} & \textit{if } l(x, y; w) = 0 \end{cases}$$

We now want to investigate the consequences of further regularity of f on the subgradient.

Proposition (Lipschitz and subgradients)**.** *If f is ρ-Lipschitz ($\forall u, v \in A, \quad |f(u) - f(v)| \leq \rho \|u - v\|$) then we have : $\forall w, \forall v \in \partial f(w), \quad \|v\| \leq \rho$.*

2.1 The subgradient stochastic descent.

We start with the case of f differentiable.

Definition (Gradient descent)**.** *The gradient descent algorithm (Algo. 2) for a differentiable, convex function $f(w) = \sum_{i=1}^{n} l(x_i, y_i; w)$ aims at finding $\arg\min_w f(w)$, in "going down" (like hiking down a mountain) incrementally in steps towards the opposite direction of the gradient, with step sizes given in advance.*

In practice, taking $\eta_t \sim 1/t$ gives good results (but the coefficient depends on the Lipschitz constant).

2.2 Algorithm for the general case : f only convex.

To adapt this algorithm to non differentiable convex functions, we can rely on fundamental algorithm in machine learning : the Stochastic Subgradient Descent Algorithm (Algo.3). The aim of this algorithm is to find a $w^\star$ minimizing $\mathbb{E}_{x,y \sim \mathbb{P}_{x,y}} l(x, y; w)$ over $w \in A$ (A being a closed convex set). Roughly speaking, it is

Algorithm 2 Gradient Descent

```
Input: $f$ differentiable (and convex), $T > 0$ , $(\eta_t)_{t=0...T-1} \in (\mathbb{R}_+^*)^T$
Output: $w$, probably around a minimum of $f$
  $w_0 \leftarrow 0$
  for $t = 0$ to $T - 1$ do
    $w_{t+1} \leftarrow w_t - \eta_t \nabla f(w_t)$
  end for
  return $w_T$
```

done in adapting the gradient descent with subgradients picked randomly instead of gradients. Stochastic gradient descent iterates could correspond to the sequences of moves of a drunk (stochastic) bug who tries to go back home (the minimum).

Algorithm 3 Stochastic Subgradient Descent

```
Input: $f$ convex, $T > 0$ , $\eta_0 > 0$
Output: $w$, probably around a minimum of $f$
1: $w_0 \leftarrow 0$
2: for $t = 1$ to $T$ do
3:   choose $v_t$ s.t. $\mathbb{E}(v_t) \in \partial f(w_{t-1})$
4:   $\eta_t \leftarrow \frac{\eta_1}{\sqrt{t}}$
5:   $w'_t \leftarrow w_{t-1} - \eta_t v_t$
6:   $w_t \leftarrow \Pi_A(w'_t)$ // "slap the drunk bug" back to $A$ if it goes too far away from its hometown
7: end for
8: return $\bar{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$
```

Note that in line 6, we project the new vector onto A to prevent the stochastics from taking us too far away from the base set A (see figure 2.2). Returning a mean value of all the computed vectors counterbalances the stochastic choices of v_t that do not necessarily decrease $\mathbb{E}_{x,y\sim\mathbb{P}_{x,y}} l(x, y; w)$.

And here is the central theorem on this algorithm :

Theorem. *Assume that $f = l(x, y; w)$ is convex (in w), and $\mathbb{E}\|v_t\|^2 \leq \rho^2$,*
let $w^\star \in \arg\min_{w\in A} f(w)$, and u a real number such that $\sup\{\|w^\star - w\|, w \in A\} \leq u$,
then $\mathbb{E}(f(\bar{w}) - f(w^\star)) \leq \frac{1}{\sqrt{T}}\left(\frac{u^2}{2\eta_1} + \rho^2\eta_1\right)$

Note that in the particular case of $\eta_1 = \frac{u}{\rho\sqrt{2}}$ we have $\mathbb{E}\left[f(\bar{w}) - f(w^\star)\right] \leq u\rho\sqrt{\frac{2}{T}}$.

Proof. Convexity and Jensen's inequalities give :

$$f(\bar{w}) \leq \frac{1}{T}\sum_{t=1}^{T} f(w_t)$$

$$\mathbb{E}\left[f(\bar{w})\right] \leq \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[f(w_t)\right]$$

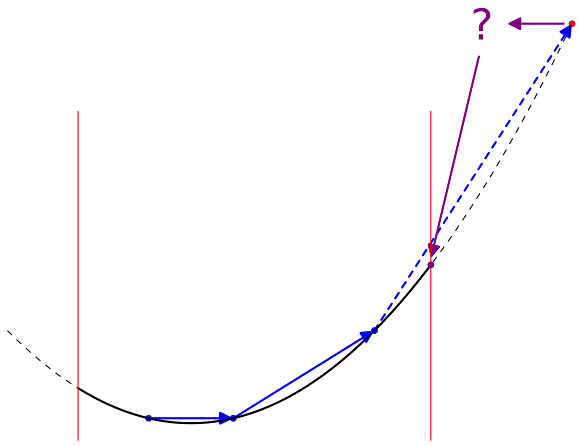

Figure 4: Slapping the bug

So it suffices to prove that :

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[f(w_t)\right]-f(w^\star)\leq\frac{1}{\sqrt{T}}\left(\frac{u^2}{2\eta_1}+\rho^2\eta_1\right)$$

The idea is to prove that for any t,

$$\mathbb{E}f(w_t)-f(w^\star)\leq\mathbb{E}\left(\frac{\|w_t-w^\star\|^2-\|w_{t+1}-w^\star\|^2}{2\eta_t}\right)+\frac{\eta_t}{2}\rho^2 \tag{3}$$

Which will telescope into :

$$\sum_{t=1}^{T}\left\{\mathbb{E}f(w_t)-f(w^\star)\right\}\leq\mathbb{E}\left\{\frac{\|w_1-w^\star\|^2}{2\eta_1}+\sum_{t=2}^{T}\|w_t-w^\star\|^2\left(\frac{1}{2\eta_t}-\frac{1}{2\eta_{t-1}}\right)\underbrace{-\frac{\|w_{T+1}-w^\star\|^2}{\eta_t}}_{\leq 0}\right\}+\sum_{t=1}^{T}\frac{\eta_t}{2}\rho^2$$

Then as $\forall t, \|w_t-w^\star\|^2\leq u^2$ we have the result of the theorem :

$$\sum_{t=1}^{T}\left\{\mathbb{E}f(w_t)-f(w^\star)\right\}\leq u^2\left\{\frac{1}{2\eta_1}+\sum_{t=2}^{T}\left(\frac{1}{2\eta_t}-\frac{1}{2\eta_{t-1}}\right)\right\}+\frac{\eta_1\rho^2}{2}\qquad\underbrace{\sum_{t=1}^{T}\frac{1}{\sqrt{t}}}_{\leq\int_0^T\frac{dt}{t}=2\sqrt{T}}$$

$$\leq u^2\left(\frac{1}{2\eta_T}+\eta_1\rho^2\sqrt{T}\right)=u^2\sqrt{T}\left(\frac{1}{2\eta_1}+\eta_1\rho^2\right)$$

Let's now prove inequality (3) :

$$\forall t, \mathbb{E}f(w_t)-f(w^\star)\leq\mathbb{E}\left(\frac{\|w_t-w^\star\|^2-\|w_{t+1}-w^\star\|^2}{2\eta_t}\right)+\frac{\eta_t}{2}\rho^2$$

First, the following technical lemma enables to bound the effect of projection (on A convex) :

$$\forall w, \forall x\in A, \|w-x\|^2-\|\Pi_A(w)-x\|^2\geq 0$$

Let us consider $\mathbb{E}\|w_t - w^\star\|$. By definition, $\|w_0 - w^\star\|^2 = \|w^\star\|^2$. Then, for any $t \geq 0$. Recall that w_t' is the "value" of w_t before the projection, so that $w_t' = w_{t-1} - \eta_t v_t$, and :

$$\begin{aligned}
\|w_{t-1} - w^\star\|^2 - \|w_t - w^\star\|^2 &= \|w_{t-1} - w^\star\|^2 - \|\Pi_A(w_t') - w^\star\|^2 \\
&= \Big(\|w_{t-1} - w^\star\|^2 - \|w_t' - w^\star\|^2\Big) - \Big(\underbrace{\|\Pi_A(w_t') - w^\star)\|^2 - \|w_t' - w^\star\|^2}_{\leq 0 \text{ (projection lemma)}}\Big) \\
&\geq \|w_{t-1} - w^\star\|^2 - \|(w_{t-1} - w^\star) - \eta_t v_t\|^2 \\
&\geq 2\eta_t (w_{t-1} - w^\star)^\intercal v_t - \eta_t^2 \|v_t\|^2
\end{aligned}$$

but by independance of the v_is :

$$\mathbb{E}_{v_1,\dots,v_t}\left(\|w_{t-1} - w^\star\|^2 - \|w_t' - w^\star\|^2\right) = \mathbb{E}_{v_1,\dots,v_{t-1}}\left\{2\eta_t (w_{t-1} - w^\star)^\intercal (\mathbb{E}_{v_t} v_t) - \eta_t^2 \mathbb{E}_{v_t}(\|v_t\|^2)\right\}$$

and as $\mathbb{E}_{v_t}(v_t)$ is a subgradient of f at w_{t-1},

$$(w_{t-1} - w^\star)^\intercal (\mathbb{E}_{v_t} v_t) \geq f(w_{t-1}) - f(w^\star)$$

and as we assumed that $\mathbb{E}\|v_t\|^2 \leq \rho$, we finally get :

$$\mathbb{E}\left(\|w_{t-1} - w^\star\|^2 - \|w_t' - w^\star\|^2\right) \geq 2\eta_t \mathbb{E}\left(f(w_t) - f(w^\star)\right) - \eta_t^2 \rho^2$$

which is exactly the inequality (3). □