Université Claude Bernard Lyon 1

Algorithme de descente du gradient stochastique

Laura GAY

Sous l'encadrement de Marianne CLAUSEL.

Décembre 2015

Table des matières

Introduction

L'algorithme de gradient de descente stochastique (SGD) est un algorithme simple mais constitue une approche efficace pour l'apprentissage de classifieurs linéaires sous des conditions de coûts convexes. C'est notamment le cas des machines à vecteurs support (SVM). Même si cet algorithme existe depuis quelques décennies, il a récemment de nouveau attiré les chercheurs dans le contexte de l'apprentissage à grande échelle pour lequel il a été appliqué avec succès.
Nous rappellerons tout d'abord quelques notions d'optimisation sous contraintes qui nous serviront à formuler notre problème et à l'analyser. Nous détaillerons ensuite la motivation de ce document, à savoir les classifieurs SVM, et nous les analyserons à l'aide des notions d'optimisation. Nous effectuerons ensuite la preuve de l'algorithme de descente du gradient stochastique ainsi que son utilisation dans le cas des SVM. Enfin, nous exploiterons les notions vues pour une application sur des données réelles (cancer du sein) sur le logiciel Python.

1 Notions d'optimisation sous contraintes utiles pour la suite

1.1 Définitions

Dans la suite, notre problème (P) de classification pourra être reformulé sous la forme

$$\begin{cases} \min\limits_{z\in\mathbb{R}^{p+1}} f(z) \\ \text{avec } \forall i \in [\![1,n]\!],\ g_i(z) \leqslant 0 \end{cases}$$

où g_i, f sont des fonctions $\mathcal{C}^1$ à valeurs réelles.

Définition 1 (*Lagrangien associé à* (P))

Il est défini comme suit :

$$L : \left| \begin{array}{ccc} \mathbb{R}^{p+1}\times\mathbb{R}^n_+ & \longrightarrow & \mathbb{R} \\ (z,\alpha) & \longmapsto & f(z)+\alpha^T g(z) = f(z)+\sum\limits_{i=1}^n \alpha_i g_i(z) \end{array} \right.$$

Définition 2 (*Point selle du Lagrangien associé à* (P))

$(z^*,\alpha^*) \in \mathbb{R}^{p+1}\times\mathbb{R}^n_+$ *est un point selle du Lagrangien associé à* (P) *si*

$$\max_{\alpha\in\mathbb{R}^n_+} L(z^*,\alpha) = L(z^*,\alpha^*) = \min_{z\in\mathbb{R}^{p+1}} L(z,\alpha^*)$$

Nous allons maintenant évoquer les résultats d'optimisation que l'on utilisera dans la partie suivante.

1.2 Résultats

Théorème 3 (*de Kuhn et Tucker*)

On suppose que f, g_1, ...,g_n sont convexes continues et dérivables sur $\mathbb{R}^{p+1}$.
Soit z^* un point où les contraintes sont QSL_{z^*}.
Alors z^* est une solution de (P) si et seulement si il existe $\alpha^* \in \mathbb{R}^n_+$ tel que (z^*,α^*) soit un point selle du Lagrangien. De manière équivalente,

$$g(z^*) \leqslant 0 \quad , \quad \alpha^{*T} g(z^*) = 0$$

$$\nabla f(z^*) + \sum_{i=1}^n \alpha_i \nabla g_i(z^*) = 0$$

Les α_i sont appelés multiplicateurs de Kuhn-Tucker.

Nous allons maintenant détailler la motivation de notre étude, et utiliser sur ce problème les notions et résultats que l'on vient d'exposer.

2 Support vector machines

2.1 Principe

Les machines à vecteurs support (en anglais Support Vector Machine, SVM) sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de classification (SVC) et de régression. Les SVM sont des types des classifieurs. On ne considérera ici que le cas des SVM linéaires.
Plus concrètement, voyons ce que cela représente sur un exemple. Des oranges et des pommes défilent sur un tapis roulant et on souhaite les séparer. Si on dispose par exemple de la hauteur et largeur de chacun de ces fruits, peut-on avoir un algorithme qui informe sur le type de fruit ?

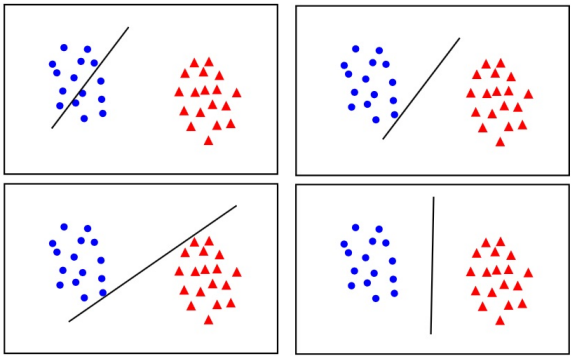

FIGURE 1 – Différents hyperplans séparateurs possibles

Considérons un exemple de problème en 2D où l'on cherche à séparer les croix des ronds. On voit que plusieurs droites permettent de séparer nos deux "zones" mais que une est plus appropriée que les autres si de nouveaux points "limites" se rapprochent. Essayons de formaliser cette pensée.

On considère un problème binaire où à chaque vecteur $x \in \mathbb{R}^p$ est associé une valeur $y \in \{\pm 1\}$. Cela forme un couple (x, y). On suppose ensuite que l'on a à disposition un échantillon de tels couples c'est à dire $\{(x_i, y_i), i \in [\![1, n]\!]\}$. Cet ensemble sera appelé ensemble d'apprentissage.

L'idée est alors de construire une fonction s qui au vecteur d'entrée x fait correspondre la sortie $y = s(x)$. Dans notre cas (le cas simple), on prend pour notre classification une fonction discriminante linéaire, obtenue par combinaison linéaire du vecteur d'entrée x, avec un vecteur de poids v :

$$h(x) = v^T x + a$$

Il est alors décidé que x est de classe 1 si $h(x) \geq 0$ et de classe -1 sinon. Autrement dit, s n'est rien d'autre que le signe de h. C'est un classifieur linéaire. L'hyperplan séparateur est alors

$$\Delta = \{x \in \mathbb{R}^p \ / \ h(x) = 0\}$$

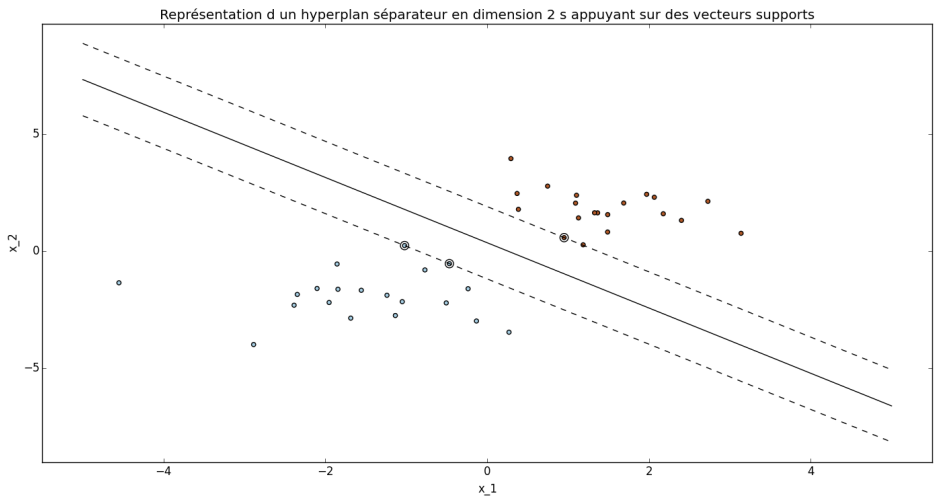

FIGURE 2 – Un exemple de problème séparable en dimension 2 d'espace

Afin de classer au mieux nos données, on cherche le "meilleur" hyperplan. En effet, il existe en général une infinité de frontières de décision linéaires séparant un échantillon. Nous allons prendre celui qui maximise la marge géométrique

$$m = \min_{i \in [\![1,n]\!]} \text{dist}(x_i, \Delta) = \min_{i \in [\![1,n]\!]} \frac{|v^T x_i + a|}{\|v\|}$$

Cette marge s'appuie sur les "vecteurs supports", visibles sur la figure ci-contre, qui définissent le plus grande séparation possible entre deux classes.

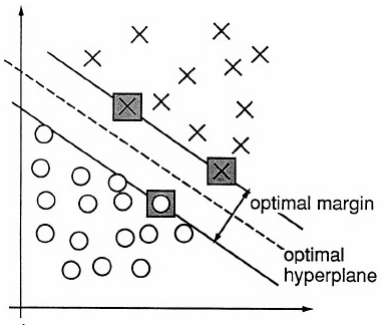

FIGURE 3 – Un exemple de problème séparable en dimension 2 d'espace. Les vecteurs supports sont marqués par des carrés gris.
Source : [3]

2.2 Formulation du problème par optimisation

On est donc face à un problème d'optimisation sous contrainte. En effet, nous souhaitons maximiser la marge, mais en n'oubliant pas que les distances de tous nos points à l'hyperplan doivent rester supérieures à la marge. Autrement dit, notre problème se réécrit :

$$\begin{cases} \text{Trouver } v \text{ et } a \text{ qui réalisent} \\ \max\limits_{v,a} m \\ \text{avec } \forall i \in [\![1,n]\!],\ \text{dist}(x_i, \Delta) \geqslant m \end{cases}$$

L'hyperplan est alors défini à constante près. En effet, si (v, a) solution, tout multiple de (v, a) l'est aussi. On fait alors le changement de variable $w = \dfrac{v}{m\,\|v\|}$ et $b = \dfrac{a}{m\,\|v\|}$. En notant $z = (w, b)^T \in \mathbb{R}^{p+1}$, on peut réécrire notre problème :

$$\begin{cases} \text{Trouver } z \text{ qui réalise} \\ \max\limits_{z} \dfrac{1}{\|w\|} \\ \text{avec } \forall i \in [\![1,n]\!],\ y_i(w^T x_i + b) \geqslant 1 \end{cases}$$

Or, maximiser l'inverse de la norme revient à minimiser le carré de celle-ci. On reformule alors notre problème, noté (P_{SVM})

$$\begin{cases} \text{Trouver } z \text{ qui réalise} \\ \min\limits_{z} \overbrace{\dfrac{1}{2}\|w\|^2}^{f(z)} \\ \text{avec } \forall i \in [\![1,n]\!],\ \underbrace{1 - y_i(w^T x_i + b)}_{g_i(z)} \leqslant 0 \end{cases}$$

On se retrouve bien dans le cas de notre problème (P) vu en partie **1.** avec g_i affine donc toutes nos contraintes sont qualifiées.
Le Lagrangien de (P_{SVM}) est

$$L : \left| \begin{array}{ccc} \mathbb{R}^p \times \mathbb{R} \times \mathbb{R}_+^n & \longrightarrow & \mathbb{R} \\ (w, b, \alpha) & \longmapsto & \dfrac{1}{2}\|w\|^2 + \sum\limits_{i=1}^{n} \alpha_i (1 - y_i(w^T x_i + b)) \end{array} \right.$$

On va raisonner par analyse synthèse. Supposons qu'il existe (w^*, b^*) solution de (P_{SVM}) tel que $\sup\limits_{\alpha \in \mathbb{R}_+^n} L(w^*, b^*, \alpha) < \infty$ alors il existe $\alpha^* \in \mathbb{R}_+^n$ tel que (w^*, b^*, α^*) soit un point selle du Lagrangien et

$$\nabla f(z^*) + \sum_{i=1}^{n} \alpha_i \nabla g_i(z^*) = 0$$

Autrement dit on a

$$w^* - \sum_{i=1}^{n} \alpha_i^* y_i x_i = 0 \tag{1}$$

$$\sum_{i=1}^{n} \alpha_i^* y_i = 0 \tag{2}$$

Ce point étant un point selle, on a également, pour tout $i \in [\![1, n]\!]$,

$$\alpha_i^* (1 - y_i({w^*}^T x_i + b^*)) = 0 \tag{3}$$

Si on veut que $\alpha_i^* > 0$ pour tout i, on se place alors sur $\mathcal{A} = \{i \in [\![1, n]\!] \ / \ y_i({w^*}^T x_i + b^*) = 1\}$.
La solution du problème, si elle existe, s'écrit donc, avec (1)

$$w^* = \sum_{i \in \mathcal{A}} \alpha_i y_i x_i$$

3 Descente stochastique du gradient

Nous allons tout d'abord montrer des résultats généraux puis les appliquer aux SVM.

3.1 Principe

On considère, comme lors de l'exemple des SVM, des couples entrée-sortie $z = (x, y)$. On considère également une fonction de perte $l(\hat{y}, y)$ qui va mesurer l'écart entre la variable prédite $\hat{y}$ et l'observée y. Notre but est de trouver une fonction f_w paramétrée par un poids w qui minimise le coût total $C(w) := \mathbb{E}_z[Q(z, w)]$, où Q est la fonction de coût définie par $Q(z, w) := l(f_w(x), y)$.
Si l'on connaissait la loi d$\mathbb{P}$ de z, nous pourrions estimer (et donc minimiser) le risque moyen (aussi appelé erreur de généralisation)

$$E(f_w) = C(w)$$

Cependant, comme nous ne disposons que d'un échantillon $(z_1, \cdots, z_n)$ dont on ne connait pas la loi, on s'intéresse plutôt à la minimisation du risque empirique (aussi appelé erreur d'entrainement)

$$E_n(f_w) = \frac{1}{n} \sum_{i=1}^{n} l(f_w(x_i), y_i)$$

Un algorithme de descente du gradient déterministe reviendrait à obtenir une estimation du meilleur poids w en effectuant

$$w_{k+1} = w_k - \underbrace{\gamma}_{\text{taux bien choisi}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \nabla_w Q(z_i, w_i)}_{\nabla_w E_n(f_w)}$$

Cependant, selon la taille des données, cet algorithme peut être très long.
L'algorithme de descente stochastique du gradient (algorithme SGD) est une simplification de l'algorithme de descente du gradient.

$$w_{k+1} = w_k - \gamma_k \nabla_w Q(z_k, w_k)$$

où z_k est choisi aléatoirement, et $\gamma_k \geqslant 0$.

3.2 Preuve

On suppose que C est différentiable. Posons H tel que $\nabla_w C(w) = \mathbb{E}_z H(z, w)$.

Proposition 4

Sous les hypothèses :

— *C admet un unique minimum* w^*
— $\forall \varepsilon > 0, \inf_{\|w-w^*\|^2>\varepsilon} (w-w^*)\nabla_w C(w) > 0$
— $\sum_{k=1}^{\infty} \gamma_k^2 < \infty$
— $\sum_{k=1}^{\infty} \gamma_k = \infty$
— $\exists A, B \geqslant 0, \mathbb{E}_z[H(z,w)^2] \leqslant A + B(w-w^*)^2$

Alors, l'algorithme se réécrit, par différentiabilité de C :

$$w_{k+1} = w_k - \gamma_k H(z_k, w_k)$$

On a convergence presque-sûre de l'algorithme vers w^*.

Étape 1 : On définit le processus de Lyapunov $h_k := (w_k - w^*)^2$ où w^* est notre solution de (P_{SVM}) précédemment définie.
On a

$$\begin{aligned} h_{k+1} - h_k &= (w_{k+1} - w^*)^2 - (w_k - w^*)^2 \\ &= (w_k - w^* - \gamma_k H(z_k, w_k))^2 - (w_k - w^*)^2 \\ &= \gamma_k^2 H(z_k, w_k)^2 - 2\gamma_k H(z_k, w_k)(w_k - w^*) \end{aligned}$$

On aimerait montrer la convergence de h_k pour avoir la convergence de l'algorithme. Cependant, la dépendance en l'aléatoire z_k gêne. On va pour pallier à cela utiliser le théorème ci-après.

Étape 2 : On note $\mathcal{P}_k := \sigma\{z_0, \cdots, z_{k-1}, w_0, \cdots, w_k\}$. On note également $\mu_k := \prod_{k=1}^{t} \frac{1}{1+\gamma_k^2 B}$ et $h'_k := \mu_{k-1} h_k$.

Théorème 5

Soit (u_k) *une suite positive.*
On définit $\delta_k := \begin{cases} 1 & \text{si } \mathbb{E}[u_{k+1} - u_k \mid \mathcal{P}_k] > 0 \\ 0 & \text{sinon} \end{cases}$.
Si $\sum_{k=1}^{\infty} \mathbb{E}[\delta_k(u_{k+1} - u_k)] < \infty$
alors (u_k) *converge presque-sûrement.*

Dans le but d'appliquer ce théorème, on regarde

$$\begin{aligned} \mathbb{E}[h_{k+1} - h_k \mid \mathcal{P}_k] &= \gamma_k^2 \mathbb{E}[H(z_k, w_k)^2 \mid \mathcal{P}_k] - 2\gamma_k (w_k - w^*) \mathbb{E}[H(z_k, w_k) \mid \mathcal{P}_k] \\ &= \gamma_k^2 \mathbb{E}_z[H(z, w_k)^2] - 2\gamma_k (w_k - w^*) \underbrace{\mathbb{E}_z[H(z, w_k)]}_{\nabla_w C(w_k)} \\ &\leqslant A\gamma_k^2 + B\gamma_k^2 \underbrace{(w_k - w^*)^2}_{h_k} - 2\gamma_k (w_k - w^*) \nabla_w C(w_k) \end{aligned}$$

Autrement dit

$$\mathbb{E}[h_{k+1} - (1 + B\gamma_k^2) h_k \mid \mathcal{P}_k] \leqslant A\gamma_k^2 \underbrace{-2\gamma_k (w_k - w^*) \nabla_w C(w_k)}_{\leqslant 0 \text{ par hypothèse sur } C} \tag{4}$$

En multipliant cette relation par μ_k, on obtient

$$\mathbb{E}[h'_{k+1} - h'_k \mid \mathcal{P}_k] \leqslant A\gamma_k^2 \mu_k$$

Finalement

$$\begin{aligned} \mathbb{E}[\delta_k(h'_{k+1} - h'_k)] &= \mathbb{E}\left[\delta_k \left[\mathbb{E}[h'_{k+1} - h'_k \mid \mathcal{P}_k]\right]\right] \\ &= \mathbb{E}[h'_{k+1} - h'_k \mid \mathcal{P}_k] \underbrace{\mathbb{P}(\mathbb{E}[h'_{k+1} - h'_k \mid \mathcal{P}_k] > 0)}_{\leqslant 1} + 0 \\ &\leqslant A\gamma_k^2 \mu_k \end{aligned}$$

Comme $\sum \gamma_k^2$ converge, et $|\mu_k| \leqslant 1$, on a $\sum \gamma_k^2 \mu_k$ converge. De plus, (h'_k) est positive, donc par théorème, (h'_k) converge presque sûrement .
Enfin, μ_k converge (passer au ln puis utiliser un développement limité et la convergence de $\sum \gamma_k^2$).
Donc (h_k) converge également presque sûrement.

Étape 3 : Il ne nous reste plus qu'à prouver la convergence ps de l'algorithme à l'aide de la convergence ps de (h_k).
En utilisant (4), la convergence ps de (h_k) et la convergence de $\sum \gamma_k^2$, on a [1], presque sûrement,

$$\sum_{k=1}^{\infty} \underbrace{\gamma_k (w_k - w^*) \nabla_w C(w_k)}_{\geqslant 0} < \infty$$

Donc $\gamma_k (w_k - w^*) \nabla_w C(w_k) \xrightarrow[i \to \infty]{\text{p.s.}} 0$.
Posons $r = \liminf_{k \to +\infty} (w_k - w^*) \nabla_w C(w_k)$. Supposons par l'absurde $r > 0$.
Alors, à partir d'un certain rang, $(w_k - w^*) \nabla_w C(w_k) > \frac{r}{2}$.
Donc $\gamma_k (w_k - w^*) \nabla_w C(w_k) > \frac{r}{2} \gamma_k$. La somme de droite diverge et celle de gauche converge, ce qui est absurde. Donc $r = 0$.
Posons maintenant $\Omega_0 = \{\omega \ / \exists H,\ h_k(\omega) \xrightarrow[t \to \infty]{\text{p.s.}} H(\omega)\}$. Cette ensemble est de mesure pleine car (h_k) converge presque sûrement.
Supposons par l'absurde qu'il existe $\omega \in \Omega_0$ tel que $H(\omega) > 0$.
On peut alors poser $H(\omega) = 2\varepsilon^2$ avec $\varepsilon > 0$.
A partir d'un certain rang, $h_k(\omega) \geqslant \varepsilon^2$ donc $(w_k(\omega) - w^*) \geqslant \varepsilon$.
Par continuité du gradient de C et par l'existence d'un unique minimum, on aurait alors

$$\liminf_{k \to +\infty} (w_k(\omega) - w^*) \nabla_w C(w_k(\omega)) > 0$$

Cette dernière proposition étant absurde par ce que l'on a montré plus haut.
Il n'existe donc pas de tel ω. Et ainsi $h_k \xrightarrow[t \to \infty]{\text{p.s.}} 0$. Autrement dit $w_k \xrightarrow[t \to \infty]{\text{p.s.}} w^*$.

Nous allons maintenant regarder comment notre algorithme s'applique au SVM en regardant notamment la fonction de coût.

3.3 Application de l'algorithme au cas des SVM

Souvent, les données ne sont pas parfaitement linéairement séparables lorsque certaines données ne sont pas dans le bon domaine. Il est possible de suivre la même démarche adoptée dans le cas séparable en introduisant variables d'écart. Dans le cas où un point est "bien" positionné, c'est à dire on a bien $y_i(w^T x_i + b) \geqslant 1$ alors la variable d'écart ξ_i est nulle. Sinon, la variable d'écart ξ_i est $\xi_i = 1 - y_i(w^T x_i + b) > 0$.
Finalement

$$\xi_i = \max(0, 1 - y_i(w^T x_i + b))$$

On cherche bien évidemment à minimiser ces variables d'écart également. Le problème vu en partie **1.** devient donc (on reprend les notations vectorielles)

$$\begin{cases} \min\limits_{w,b} \dfrac{1}{2} \|w\|^2 \\ \min\limits_{w,b} \max(0, 1 - y(w^T x + b)) \\ \text{avec } \forall i \in [\![1, n]\!],\ \underbrace{1 - y_i(w^T x_i + b)}_{g_i(z)} \leqslant 1 - \xi_i \ \text{ et } \xi_i \geqslant 0 \end{cases}$$

1. Comme (h_k) converge presque sûrement, (h_k) est bornée, disons par un certain K, alors

$$\sum_{k=1}^{\infty} \underbrace{\gamma_k (w_k - w^*) \nabla_w C(w_k)}_{\geqslant 0} \leqslant A \underbrace{\sum_{k=1}^{\infty} \gamma_k^2}_{<\infty} + \underbrace{\sum_{k=1}^{\infty} (h_{k+1} - h_k)}_{h_\infty - h_1 < \infty} + B \underbrace{\sum_{k=1}^{\infty} \gamma_k^2 h_k}_{\leqslant K \sum_{k=1}^{\infty} \gamma_k^2 < \infty}$$

La résolution de ce problème s'effectue grâce à l'introduction d'un terme d'équilibrage fixé , l'hyperparamètre $\lambda > 0$ de sorte que l'on cherche

$$\min_{w,b} \lambda \|w\|^2 + \max(0, 1 - y(w^T x + b)$$

La fonction de coût pour les SVM est donc

$$Q = \lambda \|w\|^2 + \max\{0, 1 - y(w^T x + b)\}$$

L'algorithme devient

$$w \leftarrow w - \gamma_k \begin{cases} \lambda w & \text{si } y_k(w^T x_k + b) > 1 \\ \lambda w - y_k x_k & \text{sinon} \end{cases}$$

Le tracé ci-dessous illustre l'effet du paramètre λ sur la ligne de séparation. Une grande valeur de λ sera utilisée lors d'un échantillon de données où il peut y avoir des valeurs anormales et nous permettra de considérer seulement les points proches de l'hyperplan séparateur. A l'inverse, une petite valeur de λ inclura la plupart des ou toutes les observations, permettant à la marge d'être calculée en utilisant toutes les données possibles.

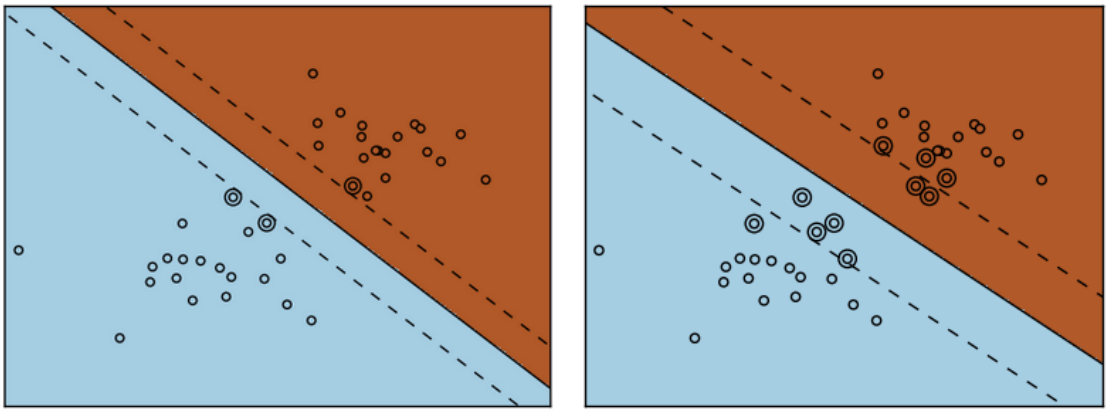

FIGURE 4 – Evolution des marges selon l'hyperparamètre. $\lambda = 1$ à gauche et $\lambda = 0.05$ à droite

Nous allons dorénavant exploiter Python pour classifier un jeu de données réels à l'aide de l'algorithme du gradient stochastique et des classifieurs SVM.

4 Utilisation de Python pour application sur un jeu de données

Nous allons utiliser scikit-learn [2], une librairie pour Python spécialisée dans le machine learning.

4.1 Données sur le cancer du sein : introduction

Le but est d'évaluer le caractère malin ou bénin d'une tumeur au sein. Les données ont été relevées au Wisconsin [3] en 1995. Nous disposons des données de 568 individus,avec le diagnostique (malin ou bénin) et 32 attributs pour chacun. Ces derniers sont donnés pour chaque noyau de cellules et sont par exemple le rayon, la texture, le périmètre. En reprenant les notations de la partie 2., les attributs forment le vecteur x et le diagnostique forme le vecteur binaire de sortie $y \in \{-, 1\}$

Il est cependant difficile de visualiser les données dont on dispose lorsque les vecteurs attributs sont de dimension supérieure à 2. Pour cela, nous allons effectuer une analyse en composantes principales.

2. http ://scikit-learn.org/stable/
3. http ://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

4.2 Analyse en Composantes Principales (ACP)

Une première étape de visualisation des nos données est l'ACP. Il s'agit de trouver des axes privilégiés pour représenter nos attributs. Ces axes seront des combinaisons linéaires des attributs [4]. L'ACP peut être définie comme la recherche de combinaisons linéaires des variables initiales de plus grande variance. L'idée est ici de trouver deux "axes" privilégiés, qu'on appellera facteurs ou composantes principales. Ce sont les nouvelles variables de plus grande dispersion. Ces 2 facteurs restituent globalement la dispersion de notre nuage de points, ce qui permet de négliger les autres pour une première représentation.

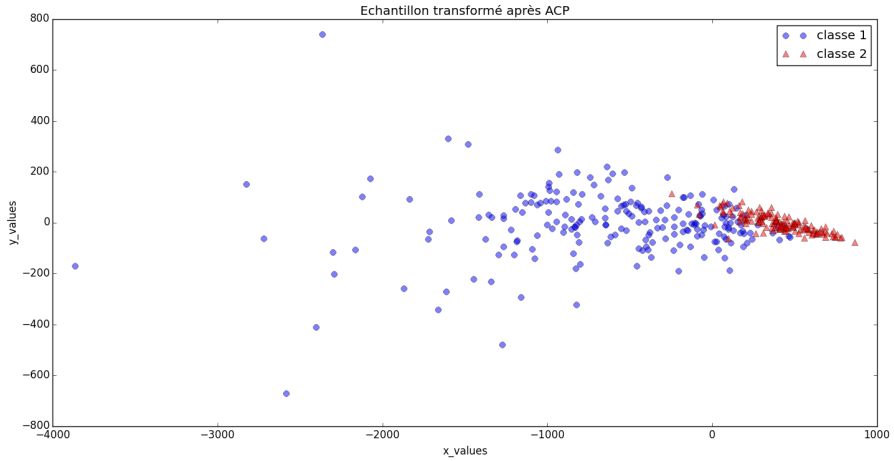

FIGURE 5 – Représentation des données "Wisconsin breast cancer" à l'aide de la fonction pca de Python

Par conséquent, les graphiques en dimension 2 présentés ci-dessus résument presque parfaitement la configuration réelle des données qui se trouvent en dimension 32 : l'objectif (résumé pertinent des donnée en petite dimension) est donc atteint.

4.3 Classification des données et performance

Sur les données mises en dimension 2 à l'aide de l'ACP, nous pouvons utiliser SGDClassifier qui utilise par défaut la fonction de coût des SVM pour représenter un hyperplan de séparation de ces deux classes. Cette fonction optimise la fonction de coût des SVM en utilisant le gradient stochastique comme vu en partie **2.**. Nous cherchons maintenant à effectuer la classification des données par vecteurs supports en utilisant l'ensemble des données. L'algorithme derrière la fonction svm.SVC utilisée est l'algorithme de descente stochastique du gradient.

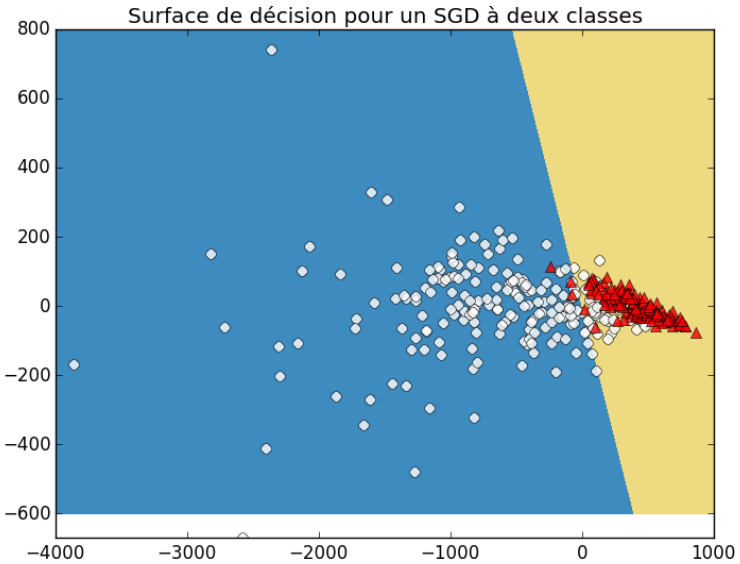

FIGURE 6 – Classification des données "Wisconsin breast cancer" à l'aide de la fonction SGDClassifier de Python

Si l'on trouve les paramètres de notre fonction de coût et qu'on les teste sur les mêmes données, nous faisons une erreur méthodologique. Nous aurions un modèle parfait pour les données que nous avons mais qui pourrait faillir dans un but prévisionnel. Pour éviter cela, nous allons utiliser une pratique courant utilisée dans le machine learning. La validation croisée ("

4. En pratique, cela revient à diagonaliser la matrice de covariance et observer les axes portant les plus grandes variances

cross-validation") est une méthode d'estimation de fiabilité d'un modèle fondé sur une technique d'échantillonnage. C'est un principe assez simple. il suffit de diviser l'échantillon en un échantillon d'apprentissage (> 60 % de l'échantillon) et un échantillon de test. Le modèle est bâti sur l'échantillon d'apprentissage et validé sur l'échantillon de test. L'erreur est estimée en calculant un score de performance du modèle sur l'échantillon de test, par exemple l'erreur quadratique moyenne.
La validation croisée nous donne un taux de nous donne un taux de 97.37% ce qui est très satisfaisant.

Conclusion

L'algorithme de gradient stochastique s'est révélé être assez bon sous certaines conditions. De plus, il est assez facile d'implémentation. Cependant, il requiert un certain nombre de valeurs extérieures à imposer comme l'hyperparamètre de régularisation λ ainsi que le nombre d'itérations.
Une prochaine étape serait de regarder l'utilisation des "noyaux" pour les SVM, c'est à dire des déformations de l'hyperplan en non-linéaire. La librairie scikit-learn permet d'utiliser différents noyaux. Par exemple, si l'on regarde le type d'iris, en fonction de la largeur et la longueur des pétales, nous obtenons, pour 3 noyaux différents, 3 types de séparateurs.

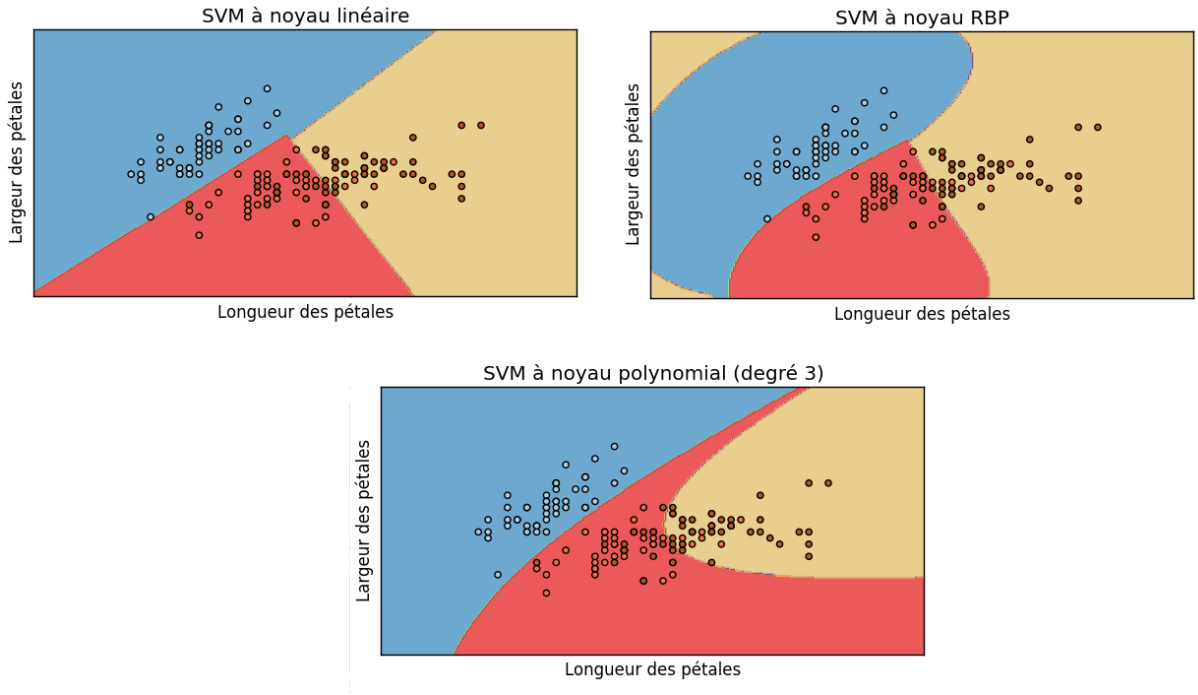

FIGURE 7 – Différents classifieurs SVM des données "Iris" à l'aide de la fonction svm.SVC

Références

[1] Léon BOTTOU, Online learning and stochastic approximations, 2012.

[2] Léon BOTTOU, Stochastic gradient descent tricks.

[3] Corinna CORTES et Vladimir VAPNIK, Support-vector networks,1995.

[4] Wikistat, Machines à vecteur supports.

[5] Wikistat, Introduction à l'Analyse en Composantes Principales (ACP).

[6] Aude RONDEPIERRE et Pierre WEISS, Cours de méthodes standards en optimisation non linéaire déterministe.