Projet de fin d'étude :
Etude des EdP géométriques sur graphes

Théotim Barbier

4 mars 2025

Résumé

Ce rapport présente mon avancement dans mon projet de fin d'étude sur les EdP géométriques sur graphes. Ce travail s'appuie sur la thèse de Matthieu Toutain de l'Université de Caen Normandie. Les graphes sont des structures pouvant représenter énormément de type données (signaux, images, bases de données ...). Il est alors intéressant de pouvoir utiliser certains outils mathématiques sur ces structures afin de traiter les données. Un outils important est bien sûr les Equations aux Dérivées Partielles (EDP), qui reviennent dans un bon nombres de problématiques, notemment dans le traitement d'image. Ainsi, la thèse de M. Toutain rervient sur les différentes proposition de définitions des Equations aux différences Finies (EdP) pour adapter les EDP aux graphes. Dans un premier temps, Toutain revient sur les définition de graphes et de l'utilisation de la structure pour modéliser les données. Ensuite, il introduits les outils de bases des EdP. Il revient après sur l'étude théorique de différents problèmes d'EdP. Enfin, il propose l'implémentation des algorithmes de résolutions de ces problèmes. Le code et les applications sont disponibles ici.

Dans ce rapport, nous allons revenir sur ce que sont les graphes et introduire les EdP sur graphe. Ensuite, nous reviendrons sur l'utilisation EdP pour résoudre l'équation Eikonale. Enfin, nous nous intéresserons au cas de la classification semi-supervisé pour la classification d'images.

Table des matières

1 Introduction

La plupart des problèmes actuels reposent sur l'utilisation de données réelles. Ces données sont la plupart du temps discrètes et réparties dans un domaine organisés (image ou grille) ou non (base de données). Il est alors intéressant de proposer une modélisation de ces données peu importe le domaine. C'est ainsi que l'utilisation de graphe pour représenter les données est venu. En effet, les graphes vont pouvoir modéliser l'interaction entre les données (distance entre 2 pixels d'une image ou similarité des attributs d'une donnée) par des arrètes pondérées entre ces données.

Connaissant cette modélisation possible pour la plupart des données, il est alors intéressant de pouvoir simuler certains outils mathématiques théoriques sur ces structures. Pour cela, on peut utiliser les équations aux différences partielles (EdP) sur graphes qui modélise de manière discrétisé les équation aux dérivées partielles (EDP) du domaine continu. Ces EdP permettent notamment résoudre différents problème théorique comme des problèmes variationnels ou des propagations de fronts. Par exemple, ces outils permettent de d'ébruiter des images ou bien encore de faire de la classification ou du clustering.

Dans ce rapport, nous reprenons les résultats de la thèse de Matthieu Toutain qui décrit les EdP géométriques sur graphes de manière théorique et appliqué. Nous allons nous intéresser aux définitions des EdP, aux résultats théoriques de certaines EdP notamment l'équation Eikonale. Enfin, nous appliquerons ces résultats pour de la classification semi-supervisée sur des bases de données d'images (MNIST,OPTDIGITS).

2 Equation aux différences Partielles sur graphes

Dans cette section, nous rappelons les notions de graphes et leur utilisation pour modéliser les données. Ensuite nous introduirons les outils de bases des EdP sur graphes. Enfin, nous regarderons un exemple simple pour le débruitage d'une image.

2.1 Rappels des graphes

Graphes

Dans ce rapport, nous allons parler de graphe pondéré non-orienté $G = (V, E, w)$ où $V \subset \Omega$ est l'ensemble des sommets (Ω est le domaine discret d'une structure de données) , $E \subset V \times V$ est l'ensemble des arrêtes et $w : E \to [0; 1]$ la fonction de poids. La fonction de poids w va représenter la similarité entre 2 sommets du graphe G.

Nous introduisons également le voisinage d'un sommet $u \in V$ qui représentent tous les sommets reliés à u :

$$N(u) = \{v \in V : (u, v) \in E\}$$

Cette notion de voisinage, nous servira pour introduire les différences finies sur les domaines irréguliers que sont les graphes. On note également le degré d'un sommet $\delta(u) = |N(u)|$.

Construction des graphes à partir de données

Nous allons maintenant nous intéresser à l'utilisation des graphes comme modélisation des données. Pour cela nous allons considéré, une fonction d'attribut $\mathcal{F} : \Omega \to \mathbb{R}^c$ qui associe à chaque données un vecteur d'attribut. Par exemple, $\mathcal{F}$ peut représenter les coordonnées d'un pixel dans une image ou encore les attributs ou propriétés connus d'une données.

Pour construire notre graphe, nous allons nous appuyer sur la notion de distance entre les données. Pour cela, en considérant que notre ensemble de données Ω est inclut dans $\mathbb{R}^m$, nous pouvons utiliser les normes usuels pour calculer les distance :

$$d(u, v) = \|\mathcal{F}(u) - \mathcal{F}(v)\|$$

Si cela n'est pas le cas, nous pouvons également utiliser n'importe quelle fonction vérifiant les propriétés des distances.

Grâce à ces distance, on peut alors calculer la similarité s entre les points. La principale propriété d'une fonction de similarité est que lorsque la distance entre 2 points augmente, la similarité diminue. Par exemple, voici 3 exemples de fonction de similarité pouvant être utilisé :

$$\begin{aligned} s_1(u, v) &= 1 \\ s_2(u, v) &= \frac{1}{d(u, v)} \\ s_3(u, v) &= \exp(\frac{-(d(u, v))^2}{\sigma^2}) \end{aligned}$$

La fonction s_3 est fréquemment utilisé car elle prend en compte la variable σ qui représente la variance dans l'ensemble de données.

Finalement, on peut alors construire nos graphes de modélisation des données.de 2 manières différentes.

Pour les domaines non organisés, on peut choisir de construire un graphes complet où chaque données interagit avec toutes les autres. On peut également limiter les interactions en imposant un seuil pour le nombre de voisin ou bien une distance maximale à respecter.

Pour les domaines organisées comme des grilles ou images, on peut utiliser l'organisation déjà présente au sein du domaine pour construire le graphe. Pour un pixel donné sur une image, on peut considérer les pixels adjacents ou à une distance maximale de ce pixel. Pour un domaine comme un maillage 3D, on peut directement utiliser le maillage comme un graphe ou bien utiliser le graphe dual qui associe les sommets aux faces du maillage et les arrêtes aux faces adjacente sur le maillage.

2.2 Outils de bases pour les EdP sur graphes

Dans cette partie, nous allons maintenant nous intéresser aux EdP sur graphes. Pour cela nous allons introduites les notions de calcul discret sur graphe.

Fonction sur graphe

Avant toute choses, nous allons nous intéresser aux fonctions définies sur les graphes. Soit un graphes $G = (V, E, w)$. Il peut y en avoir de 2 types : les fonctions $f : V \to \mathbb{R}$ sur l'ensemble des sommets et $F : E \to \mathbb{R}$ sur l'ensemble des arrêtes. On considère ces fonctions dans leur espace de Hilbert $f \in \mathcal{H}(V)$ et $F \in \mathcal{H}(E)$ munis des produits scalaires :

$$< f, g > = \sum_{u \in V} f(u)g(u)$$
$$< F, G > = \sum_{(u,v) \in E} F(u, v)G(u, v)$$

ce qui permet également de définir les normes $\mathcal{L}_p$.

Opérateur des différence pondérée

Nous allons maintenant introduire les opérateurs de différences pondérée qui sont dérivés des opérateurs de différence finis. Ces opérateurs représentent les dérivées discrète le long des arrête du graphes, on peut alors introduire $d_w : \mathcal{H}(V) \to \mathcal{H}(E)$ l'opérateur de différence pondéré telle que :

$$\partial_v f(u) = d_w(f)(u, v) = \sqrt{w_{uv}}(f(v) - f(u))$$

Gradient sur graphe

On peut alors introduire l'opérateur gradient qui est simplement le vecteur des différences pondérées de $f \in \mathcal{H}(V)$:

$$\nabla_w(f)(u) = (d_w(f)(u, v))_{v \in V}$$

et l'on peut alors calculer sa norme $\mathcal{L}_p$ qui donne :

$$\|\nabla_w(f)(u)\|_p = (\sum_{v \in N(u)} \sqrt{w_{uv}}^p (f(v) - f(u))^p)^{\frac{1}{p}}$$
$$\|\nabla_w(f)(u)\|_\infty = \max_{v \in N(u)} \sqrt{w_{uv}}|f(v) - f(u)|$$

2.3 Opérateurs directionnels

Les opérateurs directionnels sont une généralisation des opérateurs différentiels classiques sur graphe et permettent de mieux capturer certaines dynamiques locales. C'est opérateurs ont été introduits afin de reproduire les phénomènes connus de morphologie mathématiques de dilatation et d'érosion obtenues par les EDP (respectivement pour la dilatation et l'érosion) :

$$\frac{\partial f(x,t)}{\partial t} = \|\nabla f(x,t)\|_p$$
$$\frac{\partial f(x,t)}{\partial t} = -\|\nabla f(x,t)\|_p$$

Pour cela, on introduit les fonction $(x)^+ = \max(0,x)$ et $(x)^- = -\min(0,x)$.

Opérateurs de Différences Directionnelles

On définit l'opérateur externe d_w^+ de différences directionnelles d'une fonction $f \in \mathcal{H}(V)$ comme :

$$d_w^+(f)(u,v) = \sqrt{w_{uv}}(f(v) - f(u))^+$$

et l'opérateur interne comme :

$$d_w^-(f)(u,v) = \sqrt{w_{uv}}(f(v) - f(u))^-$$

Opérateurs Gradients Directionnels

On définit les gradients externes et internes respectivement comme les vecteurs :

$$\nabla_w^+(f)(u) = (d_w^+(f)(u,v))_{v\in V}$$
$$\nabla_w^-(f)(u) = (d_w^-(f)(u,v))_{v\in V}$$

On peut alors introduire leur normes $\mathcal{L}_p$ et $\mathcal{L}_\infty$ définit en $u \in V$:

$$\|\nabla_w^\pm(f)(u)\|_p = (\sum_{v\sim u} \sqrt{w_{uv}}^p |(f(v) - f(u))^\pm|^p)^{\frac{1}{p}}$$
$$\|\nabla_w^\pm(f)(u)\|_\infty = \max_{u\sim v}(\sqrt{w_{uv}}^p |(f(v) - f(u))^\pm|)$$

Lien avec les processus morphologique

A partir de ces définitions, on peut réécrire les processus d'érosion et de dilatation des EDP classique pour nos graphes. Cela donne pour la **dilatation** :

$$\frac{\partial f(u)}{\partial t} = \|\nabla_w^+(f)(u)\|_p \tag{1}$$

et l'**érosion** :

$$\frac{\partial f(u)}{\partial t} = -\|\nabla_w^-(f)(u)\|_p \tag{2}$$

On peut également voir ces 2 EdP comme les mécanismes d'érosions et de dilatation d'ensemble du graphes avec leur frontière comme dans la figure 1.

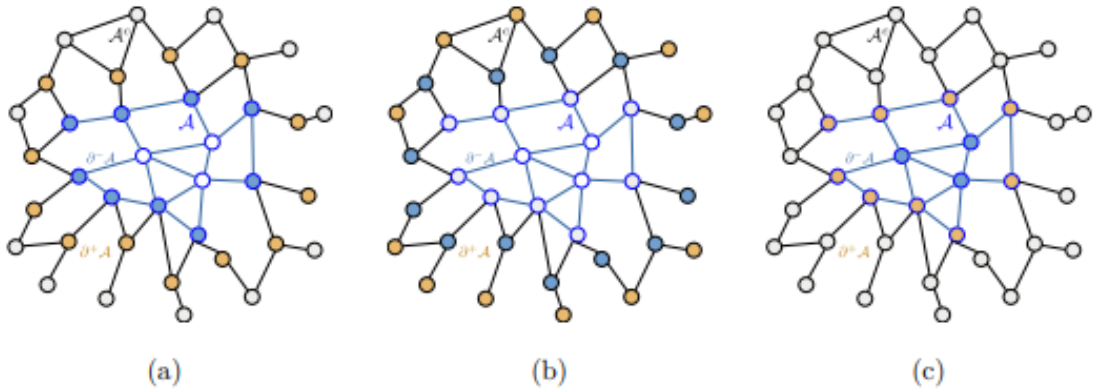


FIGURE 1 – Illustration de la dilatation et de l'érosion d'un ensemble de sommet. (a) Ensemble de sommet V et sous ensemble A ; (b) Dilatation de l'ensemble A ; (c) Érosion de l'ensemble A. Cette image est tiré de [3].

Remarque Dans notre cas de propagation de front, on s'intéressera principalement au gradient interne (Voir section **3**). On peut simplifier l'écriture de sa norme avec :

$$\begin{aligned}\|\nabla_w^-(f)(u)\|_p &= (\sum_{v\sim u} \sqrt{w_{uv}}^p |(f(v)-f(u))^-|^p)^{\frac{1}{p}} \\ &= (\sum_{v\sim u} \sqrt{w_{uv}}^p | - \min(0, f(v)-f(u))|^p)^{\frac{1}{p}} \\ &= (\sum_{v\sim u} \sqrt{w_{uv}}^p (\max(0, f(u)-f(v)))^p)^{\frac{1}{p}}\end{aligned}$$

2.4 Exemple débruitage d'image

Le débruitage d'image est une tâche essentielle en traitement d'image, visant à restaurer une image corrompue par du bruit. Parmi les approches efficaces, les opérateurs moyenneurs non locaux, dérivés des p-Laplaciens sur graphes, offrent une solution robuste en exploitant la structure globale des données.

Modèle de Régularisation par p-Laplacien Soit $G = (V, E, w)$ un graphe pondéré et $f^0 \in H(V)$ une image bruitée définie sur G. Nous cherchons une approximation f minimisant l'énergie suivante :

$$E_{w,p}(f, f^0, \lambda) = R_{w,p}(f) + \frac{\lambda}{2}\|f - f^0\|^2, \tag{3}$$

où $R_{w,p}(f)$ représente la régularité de f et est donnée par la p-énergie de Dirichlet :

$$R_{w,p}(f) = \frac{1}{2p}\sum_{u\in V} \|\nabla_w(f)(u)\|_p^p. \tag{4}$$

Opérateur moyenneur non local Lorsque $p = 2$, la minimisation de l'énergie conduit au schéma de diffusion suivant :

$$f^{(n+1)}(u) = \frac{\lambda f^0(u) + \sum_{v\sim u} w_{uv} f(v)}{\lambda + \sum_{v\sim u} w_{uv}}. \tag{5}$$

Ce schéma correspond à une diffusion non locale. Ce schéma permet d'introduire l'opérateur moyenneur non local (car ne dépend pas de la valeur en u), définit comme la solution du schéma quand $\lambda = 0$:

$$NLMean(f)(u) = \frac{\sum_{v\sim u} w_{uv} f(v)}{\sum_{v\sim u} w_{uv}}$$

Autres opérateurs non locaux Comme pour le schéma de diffusion pour le débruitage, on peut formuler les opérateurs non locaux des processus morphologique d'érosion et de dilatation vu précédemment. Ces opérateurs sont définis pour $p = \infty$.

Pour l'érosion, l'équation (2) donne un schéma classique de la forme :

$$f^{(n+1)}(u) = f^{(n)}(u) - \|\nabla_w^-(f^{(n)})(u)\|_\infty =: NLE_\infty(f^{(n)})(u)$$

De même, l'équation (1) donne :

$$f^{(n+1)}(u) = f^{(n)}(u) + \|\nabla_w^+(f^{(n)})(u)\|_\infty =: NLD_\infty(f^{(n)})(u)$$

3 Equation Eikonale et classification

L'idée de cette section est de présenter un algorithme de classification des données sur graphes en utilisant les EdP.

3.1 Problème de classification par propagation de front

L'idée de cette classification est que l'on possède un ensemble de points dont on connaît les labels. Le but va alors être de propager ces label dans le domaine en partant de ces marqueurs initiaux.

Pour cela on va utiliser un front γ représentant la courbe de niveau d'une fonction ϕ en 0, i.e. :

$$\gamma(t) = \{x/\phi(x,t) = 0\}$$

. Cette fonction ϕ va évoluer au fil du temps grâce à l'équation suivante :

$$\frac{\partial \phi}{\partial t} = \mathcal{F}|\nabla \phi|$$

où $\mathcal{F}$ est une fonction de vitesse régulant la propagation du front et $\phi(x,0) = \phi_0(x)$ le front initial. Ainsi, en propageant sur le domaine une courbe de front initialement situé sur les marqueurs initiaux, on peut labelliser chacun des points du domaines en fonction d'où vient le front de propagation.

Transposition sur graphe

Nous allons maintenant regarder l'adaptation de la propagation de front dans le cas des graphes. Soit $G = (V, E, w)$ un graphe pondéré. On peut alors décrire le front initiale partant de $\Omega_0 \subset V$ (ensemble des marqueurs initiaux dans notre cas) comme $\phi_0(v) = \chi_{\Omega_0}(v) - \chi_{\Omega_0^c}(v)$ où χ est une fonction indicatrice et Ω_0^c est le complémentaire de Ω_0. La fonction ϕ_0 renvoie donc 1 pour les élément marquées initialement et -1 pour le reste. L'article [1] propose alors de définir la propagation du front comme :

$$\begin{cases} \frac{\partial \phi}{\partial t}(u,t) = \mathcal{F}(u)\|\nabla_w(\phi)(u,t)\|_p \\ \phi_0(u) = \phi(u,0) \end{cases}$$

où $\|\nabla_w(\phi)(u,t)\|_p$ est exprimé au sens des opérateur vus précédemment.

On peut alors utiliser les gradients directionnels pour réécrire l'équation comme :

$$\begin{cases} \frac{\partial \phi}{\partial t}(u,t) = \mathcal{F}^+(u)\|\nabla_w^+(\phi)(u,t)\|_p + \mathcal{F}^-(u)\|\nabla_w^-(\phi)(u,t)\|_p \\ \phi_0(u) = \phi(u,0) \end{cases}$$

Lien avec l'équation eikonale sur graphe

Dans la thèse [3], l'équation eikonale sur graphe est donnée comme une équation stationnaire du type :

$$\|\nabla_w^-(f)(u)\|_p = P(u)$$

où P est un potentiel et f représente le temps d'arrivé au sommet u d'un front se propageant sur le graphe.

Pour simplifier notre problème, nous considérons que le front ne peut que s'étendre, et que la vitesse de propagation $\mathcal{F}$ est toujours négative. Dans notre équation, si l'on considère la vitesse de propagation $\mathcal{F}$ de signe constant, on peut réécrire notre équation comme :

$$\mathcal{F}\|\nabla_w^-(\mathcal{T})(u)\|_p = 1$$

où $\mathcal{T} : \Omega \to \mathbb{R}^+$ associe chaque point au temps d'arrivé du front en ce point, i.e $\phi(u,t) = t - \mathcal{T}(u)$ et donc $\phi(u,t) = 0$ lorsque $\mathcal{T}(u) = t$.

En introduisant P un potentiel correspondant à l'inverse de la vitesse, on peut réécrire notre équation de propagation de front comme une équation eikonale :

$$\begin{cases} \|\nabla_w^-(f)(u)\|_p = P(u) & \forall u \in V \\ f(u) = 0 & \forall u \in V_0 \end{cases}$$

où $V_0 \subset V$ correspond aux marqueurs initiaux.

3.2 Existence et unicité de la solution

Nous pouvons montrer l'existence et l'unicité de la solution en reprenant la preuve faite dans l'article [1]. Tout d'abords, reprenons notre équation comme :

$$\begin{cases} \|\nabla_w^-(f)(u)\|_p = P(u) & \forall u \in A \\ f(u) = 0 & \forall u \in \partial A \end{cases} \tag{6}$$

On, pose alors la fonction S comme :

$$\begin{cases} S(u, f, [f(v)]_{v\sim u}) = \|\nabla_w^-(f)(u)\|_p - P(u) & \forall u \in A \\ S(u, f, [f(v)]_{v\sim u}) = 0 & \forall u \in \partial A \end{cases} \tag{7}$$

Notre système s'écrit alors comme l'équation :

$$S(u, f, [f(v)]_{v\sim u}) = 0$$

Propriétés utiles

Cette fonction possède différentes **propriétés** :

Théorème 1.
1. $\frac{\partial S}{\partial f(v)} \leq 0$, $\forall u \neq v$
2. $S(f + M) = S(f)$, $\forall M \in \mathbb{R}$, $\forall u$
3. $S(\lambda f) = \lambda S(f) + (\lambda - 1)P(u)$, $\forall \lambda \geq 1$, $\forall u$
4. $\lim_{f(u)\to+\infty} S(u, f, f(v)) = -P(u)$
5. $\lim_{f(u)\to-\infty} S(u, f, f(v)) = +\infty$

Démonstration. 1) Si $u \in \partial A$ la propriété est évidente. Si $u \in A$, on a :

$$\frac{\partial S}{\partial f(v')} = \frac{\partial \|\nabla_w^-(f)(u)\|_p}{\partial f(v')}$$

Supposons que $p \neq \infty$. Pour rappel, on a

$$\|\nabla_w^-(f)(u)\|_p = (\sum_{v\sim u} \sqrt{w_{uv}}^p \max(0, f(u) - f(v))^p)^{\frac{1}{p}}$$

Ainsi, en dérivant par rapport à $f(v')$, on obtient :

$$\frac{\partial \|\nabla_w^-(f)(u)\|_p}{\partial f(v')} = \underbrace{\frac{1}{p}(\sum_{v\sim u} \sqrt{w_{uv}}^p \max(0, f(u) - f(v))^p)^{\frac{1-p}{p}}}_{\geq 0} \frac{\partial \sum_{v\sim u} \sqrt{w_{uv}}^p \max(0, f(u) - f(v))^p}{\partial f(v')}$$

et

$$\frac{\partial \sum_{v\sim u} \sqrt{w_{uv}}^p \max(0, f(u) - f(v))^p}{\partial f(v')} = 0$$

si v n'est pas voisin de u. Sinon, on a :

$$\underbrace{\sqrt{w_{uv}}^p}_{\geq 0} \frac{\partial \max(0, f(u) - f(v'))^p}{\partial f(v')}$$

Si $f(u) - f(v) \leq 0$, le max vaut 0 et la dérivée et nulle. Sinon, elle vaut :

$$\frac{\partial (f(u) - f(v'))^p}{\partial f(v')} = -p(f(u) - f(v))^p \leq 0$$

2) Pour tout $u \in A$, on a $S(f+M) = \|\nabla_w^-(f+M)(u)\|_p - P(u)$, or les M se compensent dans le gradient directionnel :

$$\|\nabla_w^-(f+M)(u)\|_p = (\sum_{v\sim u} \sqrt{w_{uv}}^p \max(0, f(u)+M-f(v)-M)^p)^{\frac{1}{p}}$$

$$= \|\nabla_w^-(f)(u)\|_p$$

3) On a :

$$\begin{aligned} S(\lambda f) &= \|\nabla_w^-(\lambda f)(u)\|_p - P(u) \\ &= \lambda \|\nabla_w^- f(u)\|_p - P(u) \\ &= \lambda S(f) + (\lambda - 1)P(u) \end{aligned}$$

4) On a trivialement que $\lim_{f(u)\to+\infty} \max(0, f(u)-f(v)) = 0$ et donc :

$$\lim_{f(u)\to+\infty} \|\nabla_w^- f(u)\|_p = 0$$

$$\iff \lim_{f(u)\to+\infty} S(u, f, [f(v)]_{v\sim u}) = -P(u)$$

5) De manière équivalente au point 4)

□

Grâce à ces propriétés, on peut prouver l'unicité et l'existence de la solution de (6).

Unicité

Théorème 2. *Si une solution de* (7) *existe, alors elle est unique.*

Démonstration. Procédons par contradiction, supposons qu'on possède 2 solutions distinctes f et g telles qu'il existe un nœud u pour lequel leur valeur est différente, i.e :

$$\max_{u\in A}(f(u)-g(u)) > 0$$

On peut alors trouver $\lambda \geq 1$ tel que :

$$M_H := \max_{u\in A}(f(u)-\lambda g(u)) \geq 0$$

Posons u_0 le nœud en lequel le maximum est atteint, i.e $M_H = f(u_0) - \lambda g(u_0)$. Soit

$$h(u) := M_H + \lambda g(u) = f(u_0) + \lambda(g(u) - g(u_0))$$

Ce qui montre que $h(u_0) = f(u_0)$. De plus, on a :

$$h(u) = M_H + \lambda g(u) \geq f(u) - \lambda g(u) + \lambda g(u) \geq f(u)$$

La propriété 1) de S implique la croissance de S par rapport à f. Ainsi, on a :

$$S(u_0, f) \geq S(u_0, h) = S(u_0, M_H + \lambda g)$$

$$= S(u_0, \lambda g)$$
$$= \lambda S(u, g) + (\lambda - 1)P(u_0)$$

Comme g et f sont solutions, on a $S(u_0, f) = 0$ et $S(u, g) = 0$. Or on a que $P(u) \geq 0$ pour tout u (car P est l'inverse de la vitesse F positive). Ce qui donne une contradiction. Alors, on a $f(u) = g(u)$ pour tout $u \in A$. □

Existence

Théorème 3. *Une solution locale en $u \in V$ de (7) existe.*

Démonstration. Soit $u \in A$. D'après la propriété 1), S est continu par rapport à f. De plus, nous savons d'après les propriétés 4) et 5) en utilisant le théorème des valeurs intermédiaires qu'il existe $f(u)$ tel que, $S(u, f(u)) = 0$. □

Ce théorème nous donne l'existence d'une solution locale mais pas directement d'une solution globale. Pour notre problème de propagation de front, la solution locale est suffisante pour assurer une convergence vers une solution globale. En effet, nous verrons que dans notre problème nous utilisons seulement le gradient externe, ce qui a pour conséquence que la propagation de front se fera uniquement vers l'extérieur de l'ensemble. Ainsi, en utilisant itérativement la solution locale sur l'ensemble des nœuds adjacents au front, on finit forcément par faire propager le front sur tous l'ensemble.

3.3 Solution locale

Afin d'utiliser notre équation eikonale pour de la propagation de front, il est nécessaire de pouvoir calculer une solution locale de l'équation. En effet, au nœud $u \in V$ on doit pouvoir calculer la solution locale dépendant seulement du voisinage de u afin de mettre à jour notre fonction f au temps t.

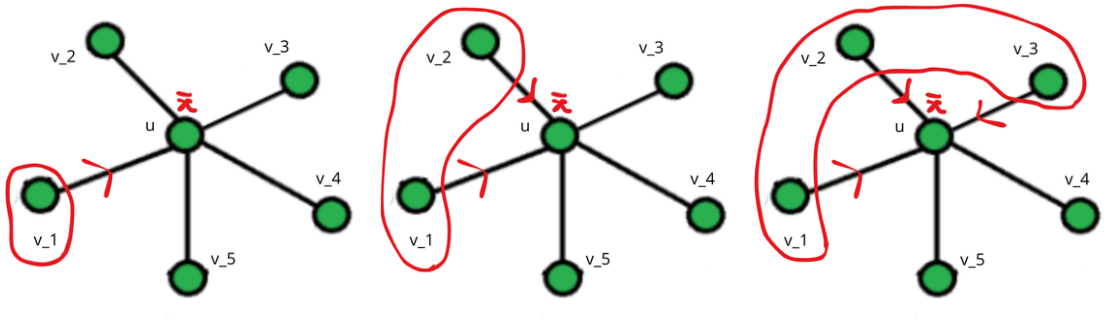
En utilisant les définitions du gradient directionnel, on peut réécrire l'équation (6) :

$$\|\nabla_w^-(f)(u)\|_p = P(u)$$
$$\iff (\sum_{v \sim u} \sqrt{w_{uv}}^p (\max(0, f(u) - f(v)))^p)^{\frac{1}{p}} = P(u)$$

On s'intéresse alors à la solution locale en $u \in V$ pour différentes valeurs de p. On notera $\bar{x}$ la solution locale trouver pour $f(u)$. Pour mettre à jour la valeur de $f(u)$, il faut s'intéresser à la valeur de $f(u) - f(v)$ pour chaque v voisin de u. Pour cela, on trie les v voisins de u par valeurs croissantes de $f(v)$ en $(v_i)_{1 \leq i \leq m}$, i.e :

$$i < j \implies f(v_i) \leq f(v_j)$$

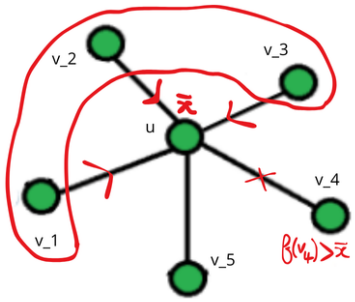
Le choix qui est fait dans l'article original [1] est alors de mettre à jour $\bar{x}$ successivement avec un sous ensemble de n sommets $(v_i)_{1 \leq i \leq n}$ jusqu'à ce que $\bar{x} - f(v_{n+1}) \leq 0 \iff \bar{x} \leq f(v_{n+1})$. La figure 2 explique le processus pour mettre à jour la solution locale.



(a) Étape 1 : On commence par mettre à jour $\bar{x}$ avec uniquement v_1.

(b) Étape 2 : Comme $f(v_2) < \bar{x}$, on met à jour $\bar{x}$ avec $\{v_1, v_2\}$.

(c) Étape 3 : Comme $f(v_3) < \bar{x}$, on met à jour $\bar{x}$ avec $\{v_1, v_2, v_3\}$.



(d) Étape 4 : $f(v_4) \geq \bar{x}$ donc on s'arrête et $f(u) = \bar{x}$.

FIGURE 2 – Exemple d'étapes successives du calcul de la solution locale en u. Les v_i sont classés par ordre croissant de valeur de f.

Pour p = 1

Pour mettre à jour la solution $\bar{x}$ en u on utilise :

$$\bar{x} = \frac{\sum_{i=1}^{n} \sqrt{w_{uv_i}} f(v_i) + P(u)}{\sum_{i=1}^{n} \sqrt{w_{uv_i}}} \tag{8}$$

où le n fait référence au n vu précédemment pour lequel $\bar{x} \leq f(v_{n+1})$. En effet, pour $p = 1$, on a :

$$\|\nabla_w^-(f)(u)\| = \sum_{v \sim u} \sqrt{w_{uv}}(\max(0, f(u) - f(v)))$$

Et en injectant la solution $\bar{x}$ on peux se débarrasser du max :

$$\sum_{i=1}^{n} \sqrt{w_{uv_i}}\left(\frac{\sum_{i=1}^{n} \sqrt{w_{uv_i}} f(v_i) + P(u)}{\sum_{i=1}^{n} \sqrt{w_{uv_i}}} - f(v_i)\right)$$

$$= P(u)$$

Pour p = 2

La solution locale pour $p = 2$ est donnée par :

$$\bar{x} = \frac{\sum_{i=1}^{n} w_{uv_i}^2 f(v_i) + \sqrt{\sum_{i=1}^{n} (w_{uv_i}^2 P^2(u) - \sum_{j>i} w_{uv_i}^2 w_{uv_j}^2 (f(v_i) - f(v_j))^2)}}{\sum_{i=1}^{n} w_{uv_i}^2} \tag{9}$$

Pour p = ∞

La solution locale est donnée par :

$$\bar{x} = \min_{i=1}^{n} (f(v_i) + \frac{P(u)}{w_{uv_i}}) \tag{10}$$

3.4 Algorithme de classification

L'article [1] propose alors l'algorithme suivant de classification utilisant l'équation précédente. Cet algorithme que nous allons décrire ici est également repris dans [3].

L'idée de l'algorithme est d'utiliser la propagation de front de l'équation eikonale (6) pour faire de la classification. Considérons un graphe $G = (V, E, w)$ où l'on cherche à classifier les différents nœuds du graphe (pouvant par exemple représenter des images). L'idée va être de labelliser les nœuds du graphe en fonction des nœuds voisins déjà labellisés. Pour cela, on commence avec un ensemble de nœuds, que l'on nommera des "germes", dont on connaît la classe. On peut alors associé un label à chaque classe connu. Ces germes vont représenter le front d'origine de l'équation eikonale, i.e l'ensemble V_0. On va donc initialiser le front f à 0 pour les nœuds de V_0, et à $+\infty$ pour les autres nœuds (c'est à dire que le front arrive dans un temps infini). L'idée va alors être de mettre à jour les sommets voisins du fronts que l'on va appelé la bande mince. Pour cela, on calcul une première fois la solution locale pour tous les sommets de la bande mince tout en labellisant les sommets avec le label le plus proche. Ensuite, on itère en prenant le nœud de la bande mince dont la valeur de f est la plus faible (temps d'arrivée le plus court) que l'on va pouvoir ajouter au front. On va donc ajouter les nœuds voisins de ce sommet à la bande mince et calculer leur solution locale. On leur attribut également un label en fonction de la valeur de f de leur voisin pondéré par le poids. On répète cela jusqu'à ce que tous les sommets soient dans la bande mince.

Cet algorithme est donné dans l'algorithme 1 avec les notations de [3].

3.5 Propagation de front multi-classe

Dans la thèse [3], Toutain propose un autre algorithme de classification supervisé pour propager différents fronts en même temps. Cette propagation de front est cette fois ci basé sur la courbure du front à la place de la vitesse de propagation. Cette équation de courbure sur graphe est donnée dans l'article de Chakik et al. [2].

L'idée pour gérer différents front est d'utiliser plusieurs fonctions $f_l : V \to [-1; 1]$ associé à chaque front $l \in \{1, .., m\}$. Chaque fonction f va donner l'appartenance d'un nœud u à un front. En effet, f_l vaut 1 si u appartient au front l, -1 si il appartient à un autre front et 0 sinon. On fait alors varier chaque front avec un schéma itératif ressemblant à 6 en utilisant la courbure.

Algorithm 1 Propagation de front sur graphe

Données : S^0 : L'ensemble des marqueurs initiaux ;
A : L'ensemble des sommets actifs ;
NB : File de priorité des sommets de la bande mince, ordonné selon f ;
FA : L'ensemble des sommets lointains ;
lab : La fonction de marqueur.

début
for all $u \in V$ **do**
    if $u \in S^0$ **then**
        $lab(u) \leftarrow$ Marqueur initial de u
        $f(u) \leftarrow 0$
    else
        $f(u) \leftarrow +\infty$
    end if
    $s(u) \leftarrow +\infty$
end for
$A \leftarrow S^0$
$NB \leftarrow \{u \mid \exists v \in A \text{ et } v \in N(u)\}$
$FA \leftarrow V \setminus (A \cup NB)$
for all $u \in NB$ **do**
    $f(u) \leftarrow$ solution locale
end for
while $FA \neq \emptyset$ **do**
    $u \leftarrow$ premier élément de NB
    Retirer u de NB et ajouter u à A
    for all $v \in N(u) \setminus A$ **do**
        Calculer la solution locale $t \leftarrow f(v)$
        if $t < f(v)$ **then**
            $f(v) \leftarrow t$
            if $v \in FA$ **then**
                Retirer v de FA et ajouter v à NB
            else
                Mettre à jour la priorité de v dans NB
            end if
        end if
        if $\frac{f(u)}{w_{uv}} < s(v)$ **then**
            $s(v) \leftarrow \frac{f(u)}{w_{uv}}$
            $lab(v) \leftarrow lab(u)$
        end if
    end for
end while

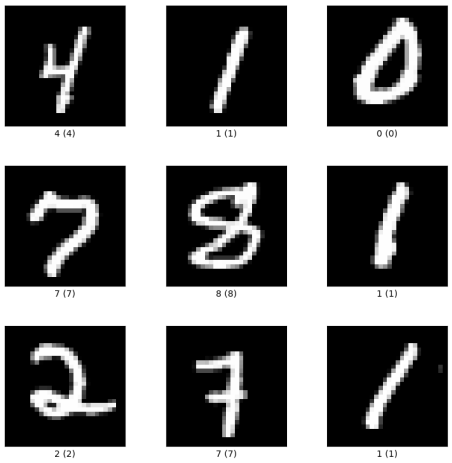


FIGURE 3 – Exemple d'image de la base de données MNIST.

Cette modélisation des fronts et de l'algorithme permet de gérer à la fois la situation où seulement un sous ensemble des nœuds est étiqueté mais également le cas où l'ensemble des nœuds sont étiquetés. Le deuxième cas corresponds alors a un raffinage de l'étiquetage des nœuds.

Combinaison des algorithmes

Une idée de la thèse est de combiner les deux algorithmes précédents. En effet, possédant un sous ensemble de nœuds labellisés, nous pouvons utiliser le première algorithme pour calculer un étiquetage global. Ensuite, nous pouvons utiliser le second algorithme pour raffiner les résultats du premier. L'auteur de la thèse montre que cette utilisation des 2 algorithmes permet de trouver de meilleurs résultats. Cependant, nous n'explorerons pas davantage cette algorithme ici.

4 Expériences et résultats

Dans cette section, nous allons tester l'algorithme de classification par propagation de front donné dans la section précédente. L'ensemble des algorithmes est implémenté en Python et est disponible ici.

Nous allons nous intéresser au problème classique de classification d'image de les base de données MNIST (figure 3) et OPTDIGITS (figure 4). Les images MNIST, et OPTDIGITS sont des imagettes représentant des chiffres écrits. Le but est donc de retrouver automatiquement quels sont les chiffres écrits sur les images.

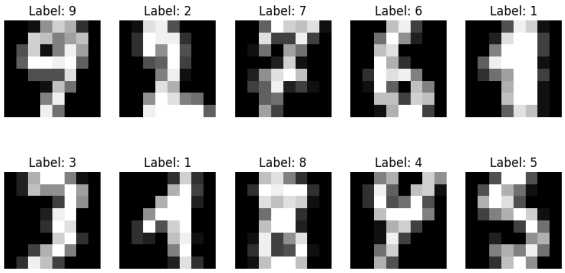


FIGURE 4 – Exemple d'image de la base de données OPTDIGITS.

4.1 Construction du graphe

Pour utiliser notre algorithme, il faut tout d'abord construire un graphe représentant nos données. Pour cela, comme vu précédemment il nous faut choisir une distance et une fonction de similarité pour construire le graphe et les poids.

MNIST Comme la base de données MNIST n'est pas normalisé, il nous faudrait une distance robuste aux transformations affines. Dans la thèse, Toutain propose d'utiliser la "two-sided tangent distance". Dans notre cas, cette distance est un peu trop coûteuse en temps pour pouvoir l'utiliser sur ma machine ou colab. Nous utiliserons donc les distances euclidiennes et cosinus.

OPTDIGITS Les images de cette base de données sont normalisés, nous pouvons donc utiliser la distance euclidienne pour construire le graphe.

Dans les deux cas, nous utilisons un graphe des 10 plus proches voisins. De plus, les poids du graphes sont calculés par la fonction de similarité $s_2(u, v) = \frac{1}{d(u,v)}$. L'utilisation de cette similarité est justifié pour le coup en calcul qui est beaucoup moins grand que pour la similarité s_3 dont il faut calculer la valeur de σ.

Remarque L'algorithme de classification ne dépend que du graphe données en entrées et non des données réelles. Il est donc assez générale et ne requiert pas de calcul sur les données autres que celles requises pour la création du graphe. Ceci est un grand avantage, notamment pour des images de grandes tailles où les calculs peuvent être très grands.

4.2 Paramètres des expériences

Les expériences menés pour ce rapport ont les caractéristiques suivantes. Elles utilisent la solution locale pour p=1. La fonction de potentiel est fixé égale à 1 pour tout $u \in V$. Pour valider les résultats, nous regardons principalement la métrique d'accuracy, qui représente le taux de données bien classés sur le nombre de données total. Cette mesure est pertinente car nos données sont toujours équilibré, i.e il n'y a pas de label prépondérant. Pour compléter ces mesures, nous utiliserons les autres métriques habituelles de validation en classification ainsi que la matrice de confusion.

4.3 Résultats

Dans cette section, nous allons présenter les résultats des différentes expériences menés. Dans un premier temps nous regarderons les données MNIST avec une étude sur le nombre de données et sur la distance a utiliser pour ces données. Dans un second temps, nous analyserons avec les données OPTDIGITS l'influence du nombre de germes et de l'initialisation de l'algorithme.

MNIST

Tout d'abord nous limitons le nombre de données à 5000 images pour réduire le temps en calcul. En construisant le graphe en utilisant différentes distances puis en appliquant l'algorithme sur chacun des graphes avec 1% de germes tirer aléatoirement, on trouve un taux de classification (accuracy) d'environ 69% dans chacun des cas. Ce taux de classification est assez mauvais par rapport aux résultats obtenus dans la thèse. Cela est probablement du au manque de données utilisé et aux distances.

Pour avoir de meilleurs résultats, nous nous intéressons maintenant à l'utilisation de 30000 images de la base de données. Nous allons alors comparer l'utilisation de 2 distances différentes lors de la construction du graphe : la distance euclidienne et la distance cosinus. Les résultats sont disponibles dans le tableau 1. On remarque directement que l'utilisation de la distance euclidienne

Distance	Euclidienne	Cosinus
Taux de classification (%)	81,6	87,5

TABLE 1 – Résultats des expériences sur la base de données MNIST pour 30000 images et différentes distances dans la construction du graphe.

sur les données MNIST donne de moins bon résultats que la distance cosinus. On en conclut donc l'importance de bien choisir la distance utiliser en fonction du problème. Cependant, il faut également nuancer ce résultat. En effet, en regardant les matrices de confusions des 2 expériences dans les figures 5 et 6 on remarque qu'avec la distance cosinus, le modèle a un moins bon taux de classification des images représentant le nombre 4 que le modèle avec la distance euclidienne. Cela peut être un problème en général si l'on s'intéresse a une classe plutôt qu'une autre.

Remarque : L'expérience précédente n'a été réalisé qu'une seule fois. Il faudrait pour avoir des résultats plus crédibles en réalisé plusieurs avec des initialisations aléatoires. Cependant, pour cette expériences, le temps de calcul de l'algorithme dépasser l'heure et demi, je n'avais donc pas vraiment le temps d'amélioré les résultats. Cette durée de l'algorithme n'est pas seulement du au fait que l'on a un grand nombre de données, mais aussi au fait que l'algorithme implémenter n'est pas optimisé. En effet, d'une, Python n'est pas forcément le langage le plus efficace pour la résolution de système numérique. Ensuite, la modélisation du graphe que j'ai utilisé est une modélisation donnée par la librairie NetworkX qui est simple d'utilisation mais pas adapté au problème présent. Il aurait fallu utiliser la modélisation proposé dans la thèse [3] qui utilise un rangement efficace en mémoire du graphe. Cependant, l'utilisation de python ne permet pas à ma connaissance de faire ceci. De plus, le

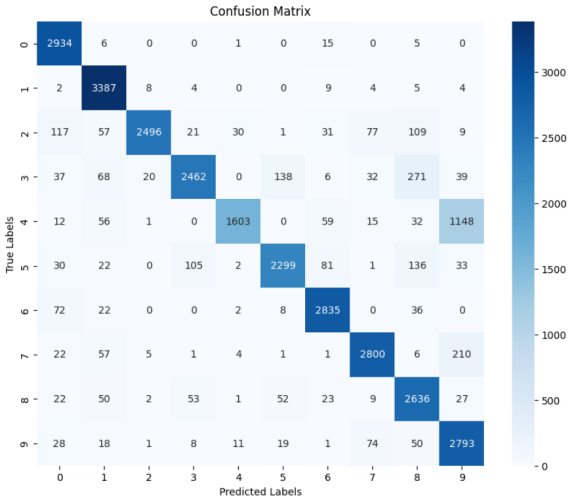


FIGURE 5 – Matrice de confusion de l'expérience sur 30000 images MNIST avec la distance cosinus pour la construction du graphe.

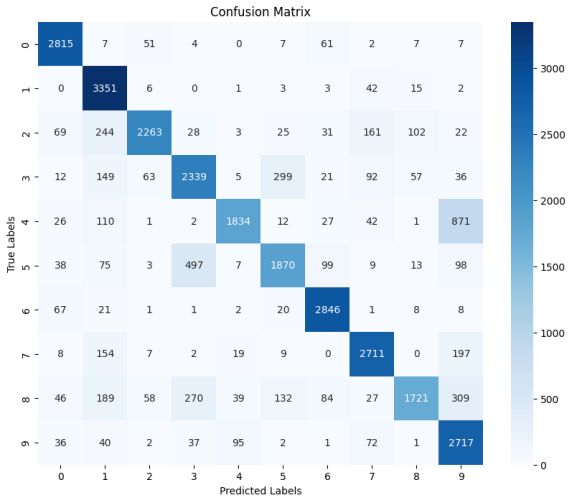


FIGURE 6 – Matrice de confusion de l'expérience sur 30000 images MNIST avec la distance euclidienne pour la construction du graphe.

choix du potentiel utilisé et de la fonction de similarité pourrait également aider à une convergence plus rapide.

OPTDIGITS

Dans cette partie, nous allons regarder la base de données OPTDIGITS composé de 5620 éléments. Les graphes de données ont été construit avec la distance Euclidienne et la similarité s_3.

Pour commencer, nous allons comparer l'algorithme pour un différents nombre de "germes" d'initialisation. Nous nous proposons de regarder les résultats pour 1%, 5% et 10% des données marqués initialement. Pour chacun des paramètres précédent, l'expérience a été répété 50 fois avec un tirage aléatoire des germes initiales afin d'augmenter la pertinence des résultats. Les résultats peuvent être retrouver dans le tableau 2.

Pourcentage de germes (%)	Taux de classification moyen (%)	Variance du taux de classification	Temps d'exécution moyen (s)
1	88.9	0.001	112.8
5	93.4	0.0002	171.6
10	95	0.00026	178.2

TABLE 2 – Tableau des performances de classification pour différents nombres de labels initiaux. L'expérience a été réalisé 50 fois pour chaque pourcentage avec tirage aléatoire de l'initialisation. La solution locale pour p=1 a été utilisé avec un potentiel fixe égal à 1.

Tout d'abord, comme prévu, le taux de classification moyen augmente avec le nombre de germes données initialement. De plus, on remarque que la variance diminue avec le nombre de germes initiale. On peut alors déduire que le résultat dépend moins de l'initialisation ce qui est une bonne chose dans notre cas. Enfin, un point un peu étonnant est que le temps d'exécution moyen augmente avec le nombre de nœuds labellisés initialement. Cela semble contradictoire car en augmentant le nombre de marqueurs initiaux, il reste moins de marqueurs à labellisé. Il peut y avoir 2 explications a ce paradoxe :

— La fonction de potentiel utilisé peut ne tient pas en compte le nombre de données et peut donc ralentir le processus de propagation.

— Comme l'initialisation est comprends plus de nœud, lorsque l'on démarre l'algorithme, les nœuds regarder vont avoir plus de chance d'avoir un voisin déjà labellisé. Ainsi, la confrontation entre deux fronts provenant de 2 voisins différents freine la convergence d'un front en ce point et ralentit donc l'algorithme.

Comme on a un algorithme de classification, on peut s'intéresser à la matrice de confusion pour valider les résultats. La matrice de confusion pour une expérience avec 1% de germes est présenté figure 7 . Grâce a cette métrique, on remarque que l'algorithme a du mal a distinguer les 8 et les 1. Cette difficulté repose encore une fois sur les problèmes de distances. En effet, dans la base de données OPTDIGITS, les 8 et les 1 sont tous les 2 centré horizontalement et prennent toute la hauteur. Comme ils recouvrent beaucoup de même pixels, la distance euclidienne peut considérer

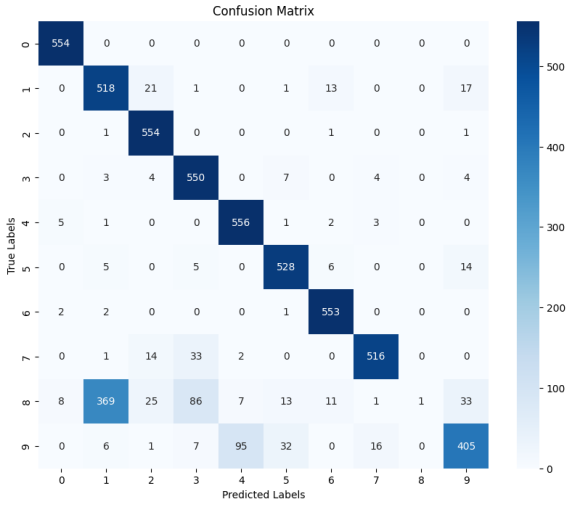


FIGURE 7 – Matrice de confusion de l'expérience sur OPTDIGITS avec la distance euclidienne pour la construction du graphe et 1% de germes. Le taux de classification pour cette expérience est de 84%.

un 8 proche d'un 1. On peut retrouver également le même problème avec les 9 et les 3 ou les 9 et les 5.

Analyse de l'initialisation

On s'intéresse dans cette partie à l'impact de l'initialisation sur l'algorithme. Ce qui nous intéresse est principalement les expériences où le l'ensemble de de nœuds initial n'est pas uniforme est comprends bien plus de nœud labellisé par un même label. Pour cela, nous reprenons notre jeu de données OPTDIGITS. Pour l'initialisation, nous gardons 1% de germes et chaque label aura 2 représentant sauf un qui aura tous les autre nœuds (37 nœuds). L'expérience a été répété 10 fois avec des initialisation aléatoires qui respectent la propriété précédente. Les résultats sont répertoriés dans le tableau 3.

Label sur-représenté	Taux de classification moyen (%)	Variance du taux de classification	Temps d'exécution moyen (s)
0	87.2	0.0016	88.6
1	73.3	0.0208	94.2
8	70.8	0.0012	90.1
aléatoire	76.2	0.011	89.5

TABLE 3 – Tableau des performances de classification pour une surepresentation d'un label

Ces résultats montrent que l'initialisation a un vrai impact sur la convergence de l'algorithme.

Ceci est particulièrement vrai sur les valeurs problématiques. En effet, comme vu précédemment, les labels 1 et 8 ne sont pas toujours bien classifiés. Lorsque l'initialisation est déséquilibré d'avantage sur ces valeurs là, on observe que la classification finale est bien plus mauvaise. Cependant, l'algorithme reste tout de même assez robuste et donne d'assez bon résultats même avec une initialisation mauvaise.

5 Conclusion

Dans ce rapport, nous avons exploré l'utilisation des équations aux différences partielles (EdP) sur graphes, en nous appuyant sur la thèse de Matthieu Toutain. Nous avons détaillé les concepts fondamentaux des graphes et leur utilisation pour la modélisation des données, avant d'étudier l'équation eikonale et ses applications en classification semi-supervisée.

Les expérimentations menées sur les bases de données MNIST et OPTDIGITS ont permis d'évaluer les performances de la méthode de propagation de front en fonction des choix de distance, de la proportion de germes et de l'initialisation des labels. Les résultats obtenus montrent l'importance de la construction du graphe ainsi que de la distance utilisée pour optimiser la classification. Toutefois, certaines limites ont été observées, notamment la dépendance des performances à l'initialisation et la sensibilité aux choix de paramètres.

Ces travaux ouvrent la voie à plusieurs perspectives. Une optimisation de l'implémentation permettrait d'accélérer les calculs, notamment via une meilleure gestion mémoire et l'utilisation de structures de données plus adaptées. De plus, l'exploration d'autres fonctions de potentiel et de similarité pourrait améliorer la précision des classifications. Enfin, l'extension de ces méthodes à des graphes plus complexes, notamment dans des contextes de données réelles à grande échelle, constituerait une évolution intéressante pour des applications concrètes en data mining ou en traitement d'image.

Références

[1] Xavier Desquesnes, Abderrahim Elmoataz, and Olivier Lézoray. Eikonal equation adaptation on weighted graphs : fast geometric diffusion process for local and non-local image and data processing. *Journal of Mathematical Imaging and Vision*, 46(2) :238–257, 2013.

[2] Abdallah El Chakik, Abderrahim Elmoataz, and Ahcene Sadi. On the mean curvature flow on graphs with applications in image and manifold processing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 697–704, 2013.

[3] Matthieu Toutain. *EdP géométriques pour le traitement et la classification de données sur graphes.* PhD thesis, Université de Caen Normandie, 2015.