(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
02 janvier 2026 (02.01.2026)

WIPO | PCT

(10) Numéro de publication internationale
WO 2026/002906 A1

(51) Classification internationale des brevets :
G06N 3/04 (2023.01) *H03M 13/00* (2006.01)

(21) Numéro de la demande internationale :
PCT/EP2025/067607

(22) Date de dépôt international :
23 juin 2025 (23.06.2025)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
FR2406836 25 juin 2024 (25.06.2024) FR

(71) Déposant : **ORANGE** [FR/FR] ; 111, quai du Président Roosevelt, 92130 Issy-les-Moulineaux (FR).

(72) Inventeurs : **DUFRÊNE, Louis Adrien** ; ORANGE - Intellectual Property and Licensing, Orange Gardens - 44 Avenue de la République - CS 50010, 92326 Chatillon (FR). **LARUE, Guillaume** ; ORANGE - Intellectual Property and Licensing, Orange Gardens - 44 Avenue de la République - CS 50010, 92326 Chatillon (FR). **LAMPIN, Quentin** ; ORANGE - Intellectual Property and Licensing, Orange Gardens - 44 Avenue de la République - CS 50010, 92326 Chatillon (FR).

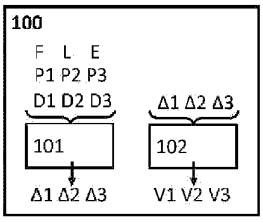
(81) États désignés *(sauf indication contraire, pour tout titre de protection nationale disponible)* : AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UY, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) États désignés *(sauf indication contraire, pour tout titre de protection régionale disponible)* : ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), eurasien (AM, AZ, BY, KG, KZ, RU, TJ, TM), européen (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV,

(54) **Title:** METHOD FOR OPTIMIZING A LINEAR CODE

(54) **Titre :** PROCEDE D'OPTIMISATION D'UN CODE LINEAIRE

[Fig. 1]



(57) **Abstract:** The invention relates to a method for optimizing a function (F), the function (F) being a linear code with which a control matrix is associated, the function (F) comprising parameters that are the elements of the control matrix, the parameters taking their values from a set (E) of discrete values, the method using another function (L), referred to as a loss function, the method, which is implemented by a data processing device (100), comprising: executing a gradient descent step on the loss function (L) in order to obtain a parameter value (V1, V2, V3), the obtained values optimizing the function (F).

(57) **Abrégé :** L'invention se rapporte à un procédé d'optimisation d'une fonction (F), la fonction (F) étant un code linéaire auquel une matrice de contrôle est associée, la fonction (F) comprenant des paramètres qui sont les éléments de ladite matrice de contrôle, lesdits paramètres prenant leurs valeurs parmi un ensemble (E) de valeurs discrètes, le procédé utilisant une autre fonction (L), dite fonction de perte, le procédé, mis en œuvre par un dispositif (100) de traitement de données comprenant : exécuter une étape de descente de gradient de la fonction de perte (L) permettant d'obtenir une valeur (V1, V2, V3) de paramètre, les valeurs obtenues optimisant la fonction (F).

WO 2026/002906 A1

[Suite sur la page suivante]

MC, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Publiée:

— *avec rapport de recherche internationale (Art. 21(3))*

PROCEDE D'OPTIMISATION D'UN CODE LINEAIRE

[0001] Le domaine technique est celui de l'optimisation des fonctions et en particulier de la recherche de codes performants, codes utilisés en télécommunications.

[0002] Plus précisément, l'invention se rapporte à un procédé d'optimisation d'une fonction. La fonction en question est définie par des paramètres qui prennent leurs valeurs parmi un ensemble de valeurs discrètes. L'optimisation de la fonction est réalisée en obtenant des valeurs de paramètres, valeurs qui optimisent la fonction.

[0003] En particulier, la fonction en question peut être un code linéaire. Un code linéaire peut être exprimé comme une matrice dont la taille dépend du schéma de codage appliqué. Le codage d'un mot d'une longueur donnée est alors réalisé en effectuant un produit matriciel du mot en question par la matrice correspondant au code. Quand le mot à coder est composé de bits, la matrice formant le code est également composée de bits et le produit matriciel correspondant à l'encodage s'effectue selon les opérations d'addition et de multiplication des bits 0 et 1. Ces principes peuvent être généralisés à des mots à coder composés de valeurs discrètes différentes des bits 0 et 1, et la matrice formant un code est alors composée de ces mêmes valeurs discrètes. Ces valeurs discrètes peuvent être par exemple des symboles correspondant à des modulations de phase utilisées pour représenter des signaux dans les communications sans fil. L'ensemble des valeurs en question peut typiquement être formé de 4, 8 ou 16 symboles distincts et une matrice formant un code linéaire pour encoder directement des mots formés de ces symboles sera également formé de ces 4, 8 ou 16 symboles distincts.

[0004] La taille typique d'une matrice formant un code linéaire sera par exemple de 16 ou 24 par 32. Le code linéaire implémenté par cette matrice est donc une fonction ayant 24 * 32 = 768 paramètres. Dans d'autres exemples classiques, la taille d'une matrice formant un code linéaire est de 200 par 256, soit une quantité de 51 200 paramètres. Ces paramètres prennent des valeurs binaires (0 ou 1) quand le code linéaire encode des mots binaires ou bien prennent des valeurs dans un ensemble de 4, 8 ou 16 symboles distincts quand le code linéaire encode des mots de ces ensembles. Dans tous les cas, les éléments de la matrice formant un code linéaire sont les paramètres d'une fonction (le code linéaire en question) et ces paramètres prennent des valeurs parmi un ensemble de valeurs discrètes, qui seront le plus souvent des valeurs binaires mais peuvent appartenir à un ensemble de valeurs discrètes de plus grande taille.

[0005] La matrice d'un code linéaire est une matrice rectangulaire. De cette manière quand un mot est encodé par un produit matriciel, le mot résultat est plus long que le mot

initial. Par exemple, quand le code linéaire est une matrice de taille 200 par 256, il va encoder des mots de 200 bits et le mot résultat, obtenu par le produit matriciel, est un mot de 256 bits. Ce mot résultat est souvent dénommé code. On peut noter que, pour un code linéaire donné, au sens de fonction, seul un sous-ensemble de codes, au sens des mots résultats possibles, peut être obtenu et non pas tous les mots de 256 bits possibles.

[0006] Les bits supplémentaires présents dans le mot résultat contiennent des informations redondantes qui permettent de retrouver le mot initial quand il est transmis et que des erreurs de transmission viennent changer les valeurs prises par des éléments du mot transmis. Ces informations redondantes sont des relations entre différents bits qui permettent de détecter des erreurs. Par exemple, un code très simple est le suivant : il va consister à ajouter à un mot binaire de longueur 2 un seul bit qui encode la parité du mot initial. Le mot 00 sera alors encodé en 000 ; le mot 11 sera encodé en 110 car 11 a la même parité que 00 ; et les mots 01 et 10 seront encodés respectivement en 011 et 101. Quand un mot est transmis, par exemple 000 et qu'il y a une erreur de transmission, ce qui donne par exemple à la réception 010, la détection d'une erreur de transmission est immédiate. En rajoutant davantage d'informations redondantes, une correction plus probable que d'autres peut être obtenue, ce que ne permet pas ce code très simple puisque, une fois qu'une erreur a été détectée, il n'est pas possible de dire laquelle est la plus probable.

[0007] Les relations entre informations redondantes peuvent être définies dans une autre matrice, dite matrice de contrôle (traduction de l'anglais *parity check matrix*). Un code linéaire peut être défini de manière équivalente par la matrice qui encode un mot par produit matriciel ou bien par la matrice de contrôle. Quand la matrice du code linéaire est de taille K par N, N étant la longueur du mot résultat obtenu après encodage, la matrice de contrôle est de taille N – K par N, c'est-à-dire la taille d'une matrice rectangulaire décrivant les liens entre informations redondantes. Dans le cas 200 par 256, la matrice de contrôle a donc une taille de 56 * 256 = 14 336. Même si la matrice de contrôle est de plus petite taille que la matrice complète de définition du code linéaire, l'ordre de grandeur du nombre de paramètres qui doivent être fixés pour définir le code est le même, à savoir plusieurs milliers, voire dizaines de milliers.

[0008] Ces différents concepts sur les codes linéaires sont bien connus de l'homme du métier et ne seront pas décrits plus avant.

État de la technique

[0009] De tels codes linéaires, permettant la détection et la correction d'erreurs, sont très utilisés pour encoder et décoder les informations transmises dans des canaux non fiables. On parle alors de codage de canal. Ces codes sont très fréquemment utilisés en

télécommunications. Pour définir un code linéaire tel qu'utilisé dans des systèmes de communications numériques sans fil (4G ou 5G par exemple), il faut donc définir une matrice qui comprend plusieurs milliers voire dizaines de milliers de paramètres. Ces paramètres sont les différents éléments de la matrice et prennent leurs valeurs parmi un ensemble de valeurs discrètes, en général des bits 0 ou 1.

[0010] Il existe plusieurs techniques traditionnelles pour construire des codes linéaires. Celles-ci ont souvent le défaut de s'intéresser aux codes en tant qu'objets mathématiques et de ne pas prendre en compte des aspects pratiques importants pour l'utilisation des codes en télécommunications. Un aspect pratique important est la simplicité du décodeur associé au code en question. Un décodeur pratique à utiliser, par exemple, utilise la propagation des convictions ou des informations (traduction de l'anglais *belief propagation*). On parle alors de décodeur par propagation d'informations (traduction de l'anglais *belief propagation decoder*). Un autre aspect pratique qui peut être important est la composition du code : un code comprenant davantage de zéros est vu comme plus avantageux que s'il comprend davantage de uns.

[0011] De plus, on a vu que trouver un nouveau code implique de trouver plusieurs milliers, voire dizaines de milliers de paramètres différents. Au vu de la taille du problème à résoudre, il est naturel de chercher à utiliser des techniques d'apprentissage automatique pour chercher de nouveaux codes linéaires.

[0012] Une technique bien connue en apprentissage automatique pour définir des fonctions avec un très grand nombre de paramètres est la technique de la descente de gradient (traduction de l'anglais *gradient descent*). Cette technique est en particulier utilisée pour apprendre les poids d'un réseau de neurones artificiel, représentant les paramètres de la fonction. Dans cette situation, un réseau de neurones implémente une fonction (par exemple une fonction de reconnaissance de forme) et cette fonction comprend un très grand nombre de paramètres (les poids des plusieurs milliers de neurones présents dans le réseau de neurones). La descente de gradient permet d'apprendre de façon automatique les valeurs de ces paramètres. Autrement dit, alors que la technique de la descente de gradient est initialement utilisée pour rechercher des points qui sont des minimums locaux ou globaux d'une fonction, elle peut être aussi utilisée pour optimiser des paramètres d'une fonction comme on le verra plus loin.

[0013] La technique de descente de gradient fonctionne de la façon suivante. Tout d'abord, une fonction, dite fonction de perte (traduction de l'anglais *loss function*) est définie. Cette fonction devra être minimisée pour optimiser la fonction pour laquelle on cherche les valeurs de paramètres. L'objectif de l'apprentissage est alors de trouver des valeurs de paramètres qui minimisent cette fonction de perte. La fonction de perte est composée de tout un ensemble de fonctions correspondant à l'effet de chaque paramètre sur la fonction de perte globale. Pour appliquer la technique de la descente

de gradient, il faut être capable de calculer des dérivées partielles pour chacune des fonctions correspondant à chaque paramètre. Des valeurs sont données pour tous les paramètres, par exemple des valeurs aléatoires, et la fonction de perte est définie ainsi que l'ensemble des dérivées partielles pour chaque paramètre.

[0014] La définition de la fonction de perte permet d'intégrer des critères importants dans l'apprentissage. Par exemple, dans le cas où l'on souhaite apprendre un code linéaire, la fonction de perte peut intégrer un décodeur préféré, comme un décodeur par propagation d'informations. De cette manière, le code est appris en prenant bien en compte un décodeur cible qui est adapté.

[0015] De façon classique, la technique de descente de gradient cherche à trouver un point pour lequel la fonction de perte est minimale. Pour cela, un point aléatoire est donné, la fonction de perte calculée, et le prochain point est obtenu en allant en sens inverse de la dérivée de la fonction, en faisant un saut proportionnel à la grandeur de la dérivée au point. De façon imagée, on descend la pente, en allant plus vite lorsque la pente est plus grande, jusqu'à arriver à un minimum, qui pourra être le minimum global de la fonction ou plus souvent un minimum local.

[0016] Ici, on souhaite apprendre des valeurs de paramètres pour minimiser une fonction dans sa globalité, et non pas chercher un point précis où la fonction est minimale. Pour cela, l'apprentissage va procéder par lots (traduction de l'anglais *batches*). Un lot (en anglais *a batch*) de points possibles pour la fonction de perte est défini et les valeurs et dérivées partielles en ces points de la fonction de perte sont obtenues. Les valeurs des paramètres sont alors modifiées dans le sens inverse de la dérivée partielle pour ce paramètre et d'une grandeur proportionnelle à celle de la dérivée partielle. Autrement dit, plus la pente de la fonction de perte est grande pour les dérivées partielles d'un paramètre donné pour un lot de points de calcul de la fonction, plus ce paramètre va être modifié, dans le sens inverse de la pente. L'évolution de la valeur du paramètre à la suite de l'évaluation d'un lot peut par exemple être obtenue en moyennant les différentes dérivées partielles pour tous les points du lot. L'idée est d'arriver ainsi à minimiser la fonction de perte sur l'ensemble des points où elle est définie en agissant sur les paramètres. Un critère d'arrêt est donné, par exemple quand l'évolution de la fonction de perte passe sous un seuil donné ou même quand un certain nombre de lots ont été fournis à l'apprentissage.

[0017] Ce procédé est très intensif en calcul. Pour s'effectuer de façon efficace, l'algorithme de descente de gradient va s'exécuter sur des processeurs dédiés qui, initialement, servaient dans les ordinateurs aux traitements graphiques. Il s'agit des unités GPU (acronyme de l'anglais *Graphics processing unit*) et non pas des unités CPU (acronyme de l'anglais *Central processing unit*) mieux connues traditionnellement en informatique. L'utilisation de GPU pour réaliser les calculs de descente de gradient

impose une contrainte supplémentaire. En effet, lors de l'utilisation de GPU, le format des calculs doit être défini à l'avance afin que l'unité de calcul puisse mettre en place des chaînes optimisées d'une taille donnée. Concrètement, cela implique que, dans l'exécution d'un algorithme de descente de gradient, la taille des lots doit être fixée à l'avance et ne doit pas être changée durant l'exécution de l'algorithme si l'on veut utiliser une unité GPU à la place d'une unité CPU. L'utilisation d'un CPU à la place de GPU apporterait davantage de souplesse mais au prix d'un facteur dix voire cent en termes de performances de calcul.

[0018] Cette technique de descente du gradient est bien établie et largement utilisée pour l'apprentissage des poids de réseaux de neurones. Elle permet donc d'apprendre automatiquement des valeurs de plusieurs dizaines de milliers de paramètres pour une fonction donnée, dans un temps de calcul raisonnable grâce à l'utilisation d'unités GPU.

[0019] Cependant, la technique de descente du gradient ne peut pas s'appliquer pour apprendre les valeurs des paramètres d'un code linéaire. En effet, cette technique ne s'applique que pour une fonction différentiable et pour des paramètres qui prennent des valeurs réelles. Or, les paramètres d'un code linéaire prennent des valeurs binaires ou plus généralement discrètes. Cette technique issue des mathématiques des fonctions réelles ne peut donc pas s'appliquer ici, alors qu'elle peut effectivement s'appliquer pour apprendre les poids d'un réseau de neurones, qui sont des paramètres réels.

[0020] De plus, une difficulté supplémentaire apparaît si l'on cherche à apprendre les paramètres discrets d'une fonction qui est un code linéaire.

[0021] En effet, la technique des codes linéaires est bien établie et les codes connus actuellement sont très performants. Cela veut dire que l'apprentissage de codes linéaires va s'appliquer à des ensembles de données pour lesquels de très bons résultats sont déjà disponibles. Autrement dit, les événements pour lesquels des améliorations sont à chercher dans la fonction de codage par apprentissage automatique arrivent très peu souvent. Ces événements sont les cas dans lesquels un mot est encodé, transmis avec une ou plusieurs erreurs, et le décodage n'arrive pas à rétablir le mot original car le nombre d'erreurs de transmissions a rendu le code inopérant. Il s'agit donc ici d'erreurs non corrigées, que le code cherche à rendre le plus rare possible. Les lots d'apprentissage éventuels doivent donc être de très grande taille pour espérer disposer d'événements représentatifs d'erreurs non corrigées qui pourront guider l'apprentissage.

[0022] Or, le nombre de paramètres typiques qui sont cherchés par apprentissage est de plusieurs milliers. Comme il s'agit de paramètres binaires, l'ensemble des combinaisons possibles est de 2 à la puissance du nombre de paramètres. L'apprentissage, par nécessité, prend donc place dans un espace beaucoup plus

petit que celui de toutes les combinaisons possibles. Et de plus, les événements représentatifs qui peuvent guider l'apprentissage (les erreurs non corrigées) sont rares et deviennent de plus en plus rares au fur et à mesure que l'apprentissage progresse et que le code appris s'améliore et devient plus performant. Quand de tels événements représentatifs surviennent, ils exerceront une influence qui peut être disproportionnée sur les résultats courants de l'apprentissage, puisqu'ils sont très rares. Il est donc difficile d'arriver à une convergence de l'apprentissage, d'autant plus avec des paramètres recherchés qui sont des valeurs discrètes.

[0023] L'invention vient améliorer la situation.

Exposé de l'invention

[0024] Selon un premier aspect fonctionnel, l'invention a trait à un procédé d'optimisation d'une fonction, dite fonction à optimiser, la fonction à optimiser étant un code linéaire auquel une matrice de contrôle est associée, la fonction à optimiser comprenant des paramètres qui sont les éléments de ladite matrice de contrôle, lesdits paramètres prenant leurs valeurs parmi un ensemble de valeurs discrètes, le procédé utilisant une autre fonction, dite fonction de perte, le procédé, mis en œuvre par un dispositif de traitement de données, comprenant :

– exécuter une étape de descente de gradient de la fonction de perte permettant d'obtenir une valeur de paramètre, les valeurs obtenues optimisant la fonction à optimiser.

[0025] Grâce à l'invention, la méthode de la descente de gradient, qui a fait ses preuves pour apprendre des paramètres réels d'une fonction en grande quantité, par exemple pour apprendre des poids de réseau de neurones, est modifiée de manière à pouvoir l'utiliser dans un contexte où les paramètres à apprendre prennent des valeurs discrètes.
Pour cela, une fonction de perte, qui est notamment différentiable par rapport aux paramètres considérés, est définie comme dans la technique de descente de gradient classique. La fonction de perte est notamment telle que l'optimisation de la fonction que l'on cherche à optimiser est équivalente à la minimisation de la fonction de perte. Les différentes dérivées partielles de la fonction de perte en fonction des paramètres de la fonction à optimiser peuvent être obtenues pour des points de l'ensemble pour laquelle la fonction de perte est définie. Tous ces éléments correspondent à une technique de descente de gradient classique mais le procédé selon l'invention en diffère en ce que les évolutions de valeurs de paramètres obtenues à partir des dérivées partielles ainsi que l'application effective de ces évolutions pour obtenir des valeurs sont contraintes afin que les valeurs de paramètres ne quittent pas l'ensemble de valeurs discrètes qu'elles peuvent prendre.

[0026] En effet, dans l'invention, l'exécution d'une étape de descente de gradient de la fonction de perte permet d'obtenir une valeur de paramètre, sachant que les valeurs de paramètres appartiennent à un ensemble de valeurs discrètes. La valeur de paramètre fournie par cette étape de descente de gradient de la fonction de perte est donc contrainte, après l'exécution d'une étape de descente de gradient, à appartenir à l'ensemble de valeurs discrètes. Exécuter une étape de descente de gradient correspond, par exemple, à exécuter une étape élémentaire d'un processus de descente de gradient. Exécuter une étape de descente de gradient correspond, dans des modes de réalisation particuliers, à exécuter une seule étape de descente de gradient ou d'un processus de descente de gradient. Dans d'autres modes de réalisation particuliers, exécuter une étape de descente de gradient correspond à exécuter une partie des étapes d'un processus de descente de gradient, ladite partie étant distincte de la totalité des étapes du processus de descente de gradient.

[0027] Dans une technique de descente de gradient classique, l'évolution de la valeur se fait dans le sens inverse de la pente pour ce paramètre, et de manière proportionnelle à la grandeur de la pente. Une telle évolution, proportionnelle à une grandeur continue, donnerait, appliquée à une valeur de paramètre donnée, de manière quasiment certaine, une valeur qui sortirait de l'ensemble de valeurs discrètes. La contrainte apportée par l'invention va quant à elle garantir que l'utilisation de la technique de descente de gradient, efficace pour apprendre des valeurs de paramètres en grande quantité, ne va pas faire sortir les valeurs de l'ensemble de valeurs discrètes auquel elles sont limitées. Une telle contrainte de maintien dans l'ensemble de valeurs discrètes est apportée à une ou des valeurs issues de l'exécution d'une étape de descente de gradient et est appliquée après l'exécution d'une étape ou d'une partie de la totalité d'étapes du processus de descente de gradient et non pas après l'exécution de la totalité des étapes de descente de gradient.

[0028] Dans un mode de réalisation, le procédé comprend : obtenir une valeur de paramètre à partir de l'exécution d'une étape de descente de gradient de la fonction de perte, la valeur de paramètre obtenue appartenant à l'ensemble de valeurs discrètes.

[0029] Selon un mode de réalisation, l'étape de descente de gradient comprend obtenir une évolution d'une valeur d'au moins un paramètre en fonction d'une dérivée partielle en au moins un point de la fonction de perte.

[0030] La descente de gradient comprend l'obtention d'évolution de valeurs des paramètres à optimiser selon les dérivées partielles de la fonction de perte. L'exécution classique d'une descente de gradient emmènerait à suivre proportionnellement les dérivées partielles de la fonction de perte. Une telle évolution des valeurs de paramètres proportionnellement à leurs dérivées partielles impliquerait une sortie du paramètre de l'ensemble de valeurs discrètes. Mais, dans l'invention, des évolutions des valeurs de

paramètres dans le procédé sont obtenues, puis appliquées, de manière que les valeurs appartiennent bien à l'ensemble de valeurs discrètes après que les évolutions obtenues leur sont appliquées. On verra par la suite que, dans des exemples de réalisation, les évolutions obtenues sont telles que les valeurs ne quittent pas l'ensemble de valeurs discrètes et, dans d'autres exemples de réalisation, les évolutions de valeurs obtenues ne garantissent pas que les paramètres ne quittent pas l'ensemble de valeurs discrètes, mais ces évolutions de valeurs sont appliquées d'une manière qui garantit que les paramètres ne quittent pas l'ensemble de valeurs discrètes.

[0031] Selon un mode de réalisation, qui peut être réalisé alternativement ou cumulativement avec le mode précédent, une évolution obtenue d'une valeur d'au moins un paramètre considéré comprend une incrémentation par un ou une décrémentation par un en fonction inverse du signe de la dérivée partielle en au moins un point de la fonction de perte obtenue pour le paramètre considéré ainsi qu'une évolution nulle d'une valeur d'un paramètre considéré si ce paramètre a déjà pour valeur le minimum ou le maximum de l'ensemble de valeurs discrètes.

[0032] Grâce à ce mode de réalisation, l'évolution des valeurs de paramètres se fait bien en restant dans l'ensemble de valeurs discrètes prédéterminées. Si la dérivée partielle de la fonction de perte pour un paramètre est positive, la valeur de ce paramètre sera diminuée de un (et inversement augmentée de un si la dérivée partielle est négative) mais si le minimum ou le maximum des valeurs possibles est atteint, il n'y aura pas d'évolution au-delà même si le signe de la dérivée partielle le demande. Ainsi, l'ensemble des valeurs de paramètres n'est pas quitté. L'évolution des valeurs se fait en sens inverse des signes des dérivées partielles car l'optimisation qui est cherchée pour les paramètres de la fonction à optimiser correspond à la minimisation de la fonction de perte. En revanche, contrairement à la technique de descente de gradient classique, les évolutions de valeurs ne sont pas proportionnelles à l'amplitude des dérivées partielles mais ne dépendent que de leurs signes et procèdent par sauts discrets de un (ou sont nulles) pour assurer que la valeur du paramètre concerné demeure parmi l'ensemble de valeurs discrètes.

[0033] Selon un mode de réalisation, qui peut être réalisé alternativement ou cumulativement avec les modes précédents, une évolution obtenue d'une valeur d'au moins un paramètre considéré comprend une variation par un nombre entier relatif, le signe de la variation étant en fonction inverse du signe de la dérivée partielle en au moins un point de la fonction de perte obtenue pour le paramètre considéré, et la valeur absolue de la variation étant en fonction de la grandeur de la dérivée partielle, ladite valeur absolue étant réduite si nécessaire pour contraindre le paramètre à ne pas dépasser le minimum ou le maximum de l'ensemble de valeurs discrètes.

[0034] Grâce à ce mode de réalisation, le procédé peut être adapté au cas où l'ensemble de valeurs discrètes que peuvent prendre les paramètres est plus grand que l'ensemble binaire formé de 0 et de 1. Dans ce cas, il peut être intéressant de prendre en compte des grandeurs élevées de dérivées partielles, ce qui n'est pas possible quand les évolutions se font par des pas de « un » uniquement. Le signe de la dérivée partielle va donner le sens de l'évolution, comme précédemment, mais le pas d'évolution pourra être plus grand que « un » en présence de dérivées partielles élevées. Cependant, pour contraindre les valeurs de paramètres obtenues à rester dans l'ensemble de valeurs discrètes, les évolutions doivent être contraintes. Les évolutions sont des entiers relatifs, et leur valeur absolue est contrainte de manière à ne pas dépasser le minimum ou le maximum si besoin est.

[0035] Selon un mode de réalisation, qui peut être réalisé alternativement ou cumulativement avec les modes précédents, le procédé est itératif et comprend, pour une itération, le choix d'un ensemble de points de l'ensemble de définition de la fonction de perte, ledit ensemble de points étant dit lot, et l'étape d'obtention d'une évolution de la valeur d'un paramètre considéré comprend le calcul d'une moyenne de la fonction de perte pour les points du lot et l'évolution de la valeur du paramètre considéré est fonction inverse de la dérivée partielle de la moyenne de la fonction de perte pour ce paramètre.

[0036] Grâce à ce mode de réalisation, il est possible de prendre en compte un grand nombre de points de la fonction de perte pour réaliser l'apprentissage des paramètres de la fonction à optimiser. Pour cela, des lots de points sont créés, en général par des tirages aléatoires. L'évolution d'une valeur de paramètre sera alors fonction d'une dérivée partielle pour la fonction de perte moyennée sur les points du lot. De cette manière, il est possible de prendre en compte un plus grand nombre de points durant l'apprentissage qui cherche à minimiser la fonction de perte plutôt que de travailler point par point de l'ensemble de définition.

[0037] En général, les calculs nécessaires au procédé seront réalisés par une unité GPU et non par une unité CPU pour avoir une plus grande efficacité de calcul. En conséquence, la taille du lot de points sera la même entre les différentes itérations du procédé afin de pouvoir bénéficier des optimisations du GPU.

[0038] Selon un mode de réalisation, qui peut être réalisé alternativement ou cumulativement avec les modes précédents, le procédé est itératif et comprend, pour une itération, le choix d'un ensemble de points de l'ensemble de définition de la fonction de perte, ledit ensemble de points étant dit lot, et l'étape d'obtention d'une évolution de la valeur d'un paramètre considéré comprend le calcul de dérivées partielles de la fonction de perte pour le paramètre considéré en les points du lot, suivi par la réduction de ces dérivées partielles à des incrémentations ou

à des décrémentations en fonction inverse du signe des dérivées partielles, et la combinaison de ces incrémentations et décrémentations en une seule incrémentation ou décrémentation, l'évolution de la valeur étant obtenue en combinant ces incrémentations et décrémentations.

[0039] Grâce à ce mode de réalisation, une alternative de calcul est proposée pour éviter de calculer la moyenne de la fonction de perte sur l'ensemble des points d'un lot. Plutôt que de s'intéresser aux dérivées partielles de la fonction de perte moyennée sur les points du lot, les différentes dérivées partielles sont obtenues en chaque point du lot et quantifiées en des incrémentations ou des décrémentations de façon inverse à leur signe. Ces différentes incrémentations et décrémentations sont ensuite réduites à une seule incrémentation ou décrémentation qui pourra être appliquée pour obtenir la valeur du paramètre considéré. Dans ce mode de réalisation, chaque point du lot a la même influence potentielle sur l'évolution possible de la valeur d'un paramètre alors que, dans le mode précédent, l'évolution dépend d'une moyenne des dérivées partielles, et les point pour lesquelles la dérivée partielle est plus forte auront donc une influence plus grande. L'alternative de ce mode de réalisation peut être plus adaptée pour égaliser les influences apportées par les différents points du lot.

[0040] Selon un mode de réalisation, qui peut être réalisé alternativement ou cumulativement avec les modes précédents, la dérivée partielle en fonction de laquelle une évolution de valeur de paramètre est obtenue est réduite à zéro si sa valeur absolue est inférieure à un seuil donné.

[0041] Grâce à ce mode de réalisation, l'apprentissage est amélioré. En effet, des valeurs faibles de dérivées partielles peuvent avoir un effet disproportionné dans le procédé selon l'invention puisqu'une très faible valeur de dérivée partielle pourrait impliquer une évolution d'une valeur discrète donc grande par rapport à des évolutions continues minimes. Il est donc préférable en-dessous d'un certain seuil d'annuler les dérivées partielles plutôt que de les voir risquer déclencher des évolutions de valeurs discrètes.

[0042] Selon un mode de réalisation, qui peut être réalisé alternativement ou cumulativement avec les modes précédents, l'étape d'obtention d'une valeur d'un paramètre comprend une étape de combinaison des évolutions obtenues avec les évolutions obtenues lors des étapes précédentes et une étape de mise à jour de la valeur d'un paramètre en fonction des évolutions combinées.

[0043] Grâce à ce mode de réalisation, les performances de l'algorithme d'apprentissage sont améliorées. Nous avons vu qu'une difficulté des algorithmes d'apprentissage est de fonctionner en présence du fléau de la dimension (traduction de l'anglais *curse of dimensionality*), c'est-à-dire que les éléments intéressants pour réaliser un apprentissage sont très rares parmi l'ensemble de points possibles. Pour un apprentissage de code linéaire, cela peut vouloir dire que les points qui correspondent

à des erreurs non corrigées, qui sont ceux intéressants pour apprendre un meilleur codage, apparaîtront très rarement dans un lot donné de points de la fonction de perte utilisé dans une itération du procédé. Pour compenser cette rareté, et étant donné que la taille d'un lot de points est fixée une fois pour toutes à cause de l'utilisation d'une unité GPU pour réaliser les calculs, il est utile de combiner et d'agréger les résultats des dérivées partielles obtenues sur plusieurs lots avant d'effectuer une mise en jour en fonction de la combinaison de ces résultats. De cette manière, un plus grand nombre de points de l'espace de définition est pris en compte avant de réaliser une évolution de la valeur du paramètre pour une taille de lots donnée.

[0044] Selon un mode de réalisation, qui peut être réalisé alternativement ou cumulativement avec les modes précédents, l'étape d'obtention d'une valeur d'un paramètre selon l'évolution obtenue comprend la mise à jour d'un compteur associé au paramètre en fonction des dérivées partielles de la fonction de perte par rapport aux paramètres respectifs et la valeur obtenue ne change que lorsque le compteur associé au paramètre a dépassé un seuil donné.

[0045] Selon un mode de réalisation qui peut être réalisé cumulativement avec le mode précédent, les compteurs associés aux paramètres additionnent entre elles les valeurs inverses des dérivées partielles de la fonction de perte par rapport aux paramètres respectifs obtenus lors des itérations précédentes.

[0046] Selon un mode de réalisation qui peut être réalisé alternativement avec le mode précédent, les compteurs associés aux paramètres additionnent entre eux les signes inverses des dérivées partielles de la fonction de perte par rapport aux paramètres respectifs obtenus lors des itérations précédentes.

[0047] Grâce à ces modes de réalisation, les performances de l'algorithme d'apprentissage sont encore améliorées. Comme vu précédemment, il est utile de réaliser une combinaison des résultats de plusieurs lots avant de faire évoluer une valeur de paramètre. L'utilisation d'un compteur associé au paramètre permet de réaliser cette combinaison de façon simple et efficace. Ces compteurs doivent prendre en compte les dérivées partielles de façon inverse aux signes des dérivées partielles pour bien réaliser une descente de gradient qui cherche à minimiser la fonction de perte.

[0048] Deux modes différents d'utilisation de compteurs sont possibles selon l'invention. Dans un premier mode, les valeurs des dérivées partielles pour un paramètre donné sont additionnées entre elles et peuvent être moyennées pour leur donner une grandeur maximale. Lorsque le compteur dépasse un certain seuil, la valeur du paramètre changera tout en restant contrainte par l'ensemble de valeurs discrètes que peut prendre le paramètre. Dans un deuxième mode, seuls les signes des dérivées partielles sont additionnés dans le compteur. Les compteurs peuvent alors augmenter ou diminuer de un selon le signe de la dérivée partielle concernée. Là encore, lorsque le compteur

dépasse un certain seuil, la valeur du paramètre changera tout en restant contrainte par l'ensemble de valeurs discrètes que peut prendre le paramètre. L'intérêt du compteur prenant en compte les valeurs moyennées de dérivées partielles est de prendre en compte la grandeur de celles-ci alors que le deuxième mode présente un avantage de simplicité de calcul. Les évolutions, dans les deux cas, doivent se faire dans le sens inverse du signe des dérivées partielles.

[0049] Dans nos expériences, l'effet de la rareté des points intéressants pour l'apprentissage, rareté qui s'accroît au fur et à mesure que l'apprentissage rend une solution meilleure, est que leur effet aura tendance à être disproportionné en l'absence d'un mécanisme compensateur qui est obtenu par une combinaison des résultats sur plusieurs lots. Dès qu'un point fournit une information utile à l'apprentissage, celle-ci va être utilisée même si c'est de façon exagérée. Ceci peut déclencher des oscillations des valeurs de paramètres autour de valeurs cibles dès que des informations pertinentes peuvent être utilisées pour l'apprentissage. Cet effet est amplifié par le fait que les évolutions des valeurs sont discrètes. En effet, dans la technique traditionnelle de descente du gradient, les évolutions des valeurs sont proportionnelles à la grandeur des dérivées partielles. Ceci permet de se rapprocher plus ou moins vite des minimums recherchés par la méthode et d'éviter de les dépasser et d'osciller autour dès que des points informatifs sont utilisés pour l'apprentissage.

[0050] Pour éviter cet écueil, le procédé selon l'invention introduit des ensembles de compteurs associés aux paramètres. Une valeur de paramètre n'évoluera (dans un sens ou dans l'autre) par exemple qu'après que son compteur associé ait enregistré un nombre d'évolutions à effectuer suivant les signes de dérivées partielles qui soit au-dessus d'un seuil donné, ou qu'après que l'addition des dérivées partielles aient atteint un certain seuil, selon le mode de réalisation. Cette évolution du procédé selon l'invention permet d'éviter le phénomène observé dans certaines versions d'oscillations des valeurs et d'impact trop fort sur l'apprentissage de points isolés. Des évolutions ne sont mises en œuvre que si elles s'appliquent dans plusieurs lots ce qui va emmener les compteurs associés à dépasser leur seuil.

[0051] De plus, cette combinaison des résultats de plusieurs lots avant de déclencher une évolution de la valeur d'un paramètre permet de réaliser des calculs avec un nombre important de données d'apprentissage tout en maîtrisant la complexité des calculs. Prenons comme exemple que le lot de points de calculs ait une taille de 32 points. Cette taille est fixée, comme on l'a vu, car les calculs sont réalisés sur des unités GPU. Si on prend un seuil pour déclencher une évolution de la valeur d'un paramètre à 20, cela veut dire qu'une évolution de la valeur de paramètre ne peut se faire qu'après avoir pris en compte 640 points (32 fois 20) ce qui correspond à un lot très grand, probablement non traitable en temps raisonnable.

[0052] Selon un mode de réalisation, qui peut être réalisé alternativement ou cumulativement avec les modes précédents, lorsqu'une valeur d'un paramètre change après que le compteur a dépassé un seuil donné, les compteurs associés à des paramètres qui sont inférieurs à un autre seuil, inférieur ou égal au seuil donné, sont remis à zéro.

[0053] Selon un mode de réalisation qui peut être réalisé cumulativement avec le mode précédent, ledit autre seuil est égal au seuil donné.

[0054] Grâce à ce mode de réalisation, l'utilisation de compteurs pour diriger les évolutions de valeurs de paramètres est affinée. Dans nos expériences, il est préférable, lorsqu'au moins une évolution de valeur de paramètre est décidée, de remettre à zéro les compteurs associés aux paramètres puisque les points de calcul ultérieurs de la fonction de perte vont être obtenus avec des valeurs de paramètres qui auront changé. Il peut être décidé de remettre à zéro tous les compteurs inférieurs au seuil donné, ou bien seulement ceux qui sont inférieurs à un autre seuil, plus petit que le seuil donné, et donc de conserver les compteurs qui sont situés entre les deux seuils, qui ont déjà accumulé un certain nombre d'indices quant à la direction dans laquelle ils doivent évoluer.

[0055] Selon un mode de réalisation, qui peut être réalisé alternativement ou cumulativement avec les modes précédents, lorsqu'une valeur d'un paramètre change après que le compteur a dépassé un seuil donné, les valeurs des paramètres dont les compteurs associés sont proches du seuil donné à une distance donnée sont également changées.

[0056] Grâce à ce mode de réalisation, l'utilisation des compteurs est encore affinée. On a vu qu'on change la valeur d'un paramètre quand son compteur associé atteint un certain seuil et que les autres compteurs sont alors remis à zéro. Il est possible également dans ce mode de réalisation de faire évoluer les valeurs des paramètres dont les compteurs associés n'ont pas atteint le seuil mais en sont proches, par exemple à une distance de 1 à 5, pour ne pas remettre à zéro une information accumulée pendant l'apprentissage. L'ensemble des compteurs peuvent alors être remis à zéro après que ces changements des valeurs des paramètres ont été appliqués.

[0057] Selon un mode de réalisation, qui peut être réalisé alternativement ou cumulativement avec les modes précédents, le seuil donné est fixé à une valeur entre 10 et 30.

[0058] Dans nos expériences, le seuil pour un compteur au-delà duquel un paramètre va évoluer ne doit pas être choisi trop faible. Par ailleurs, des compteurs trop élevés auront tendance à retarder des évolutions assurées des valeurs de paramètres. La plage de valeurs entre 10 et 30 pour ce seuil apporte de bons résultats avec une préférence pour un seuil fixé à 20.

[0059] Selon un mode de réalisation, qui peut être réalisé alternativement ou cumulativement avec les modes précédents, le seuil donné augmente en fonction directe du nombre de valeurs de paramètres à apprendre.

[0060] Toujours dans nos expériences, plus le code que l'on cherche à apprendre est grand, plus il permet de corriger des événements rares. Cela veut dire que les événements utiles à l'apprentissage, les erreurs non corrigées, sont très peu fréquents. Le procédé d'apprentissage aura donc besoin de seuils élevés pour juger de la pertinence ou non de ces événements rares.

[0061] Selon un mode de réalisation, qui peut être réalisé cumulativement ou alternativement avec les modes précédents, la combinaison des évolutions obtenues comprend l'estimation d'un intervalle de confiance pour les valeurs des dérivées partielles associée aux paramètres respectifs en utilisant une loi binomiale, et la mise à jour de la valeur du paramètre se fait quand cette estimation dépasse un certain seuil.

[0062] Grâce à ce mode de réalisation, une autre manière d'agréger des résultats de calculs sur plusieurs lots est proposée. Ici, plutôt que d'utiliser des compteurs, les valeurs des dérivées partielles prises dans les différents points sont utilisées pour effectuer une estimation d'un intervalle de confiance de la valeur de cette dérivée partielle sur tous les points de l'ensemble de définition. Une méthode telle que la méthode Agresti-Coull peut être utilisée pour cela. Une fois que l'intervalle de confiance indique que la valeur de la dérivée partielle pour un paramètre donné a très probablement dépassé un seuil donné (positif ou négatif), le paramètre en question peut être augmenté ou diminué selon le signe du seuil.

[0063] Selon un mode de réalisation, qui peut être réalisé alternativement ou cumulativement avec les modes précédents, le procédé comprend une étape d'initialisation aléatoire des valeurs des paramètres de la fonction à optimiser.

[0064] Grâce à ce mode de réalisation, un point de départ est donné aux valeurs de paramètres de la fonction à optimiser. Dans nos expériences, un choix aléatoire est performant et n'empêche pas la réalisation efficace du procédé selon l'invention.

[0065] Selon un mode de réalisation, qui peut être réalisé alternativement ou cumulativement avec les modes précédents, le procédé comprend une étape d'initialisation aléatoire des valeurs des paramètres de la fonction à optimiser selon une probabilité différenciée pour les différentes valeurs de l'ensemble de valeurs discrètes que peuvent prendre les paramètres.

[0066] Dans nos expériences, il peut être intéressant de différencier les probabilités d'obtention d'une valeur discrète donnée pour un paramètre selon les valeurs appartenant à l'ensemble de valeurs discrètes possibles. Par exemple, dans un code linéaire pour le cas binaire, un tirage aléatoire des paramètres du code avec une probabilité de 70% pour la valeur 0 et de 30% pour la valeur 1 donne de bons

résultats car elle initie le procédé avec une desité de paramètres à 1 proche d'un code performant.

[0067] Selon un mode de réalisation, qui peut être réalisé alternativement ou cumulativement avec les modes précédents, la fonction à optimiser est un code linéaire et les paramètres de la fonction à optimiser sont les éléments de la matrice de contrôle associée au dit code linéaire.

[0068] Grâce à ce mode de réalisation, l'invention est appliquée au cas particulier des codes linéaires et permet d'apprendre un code linéaire de façon efficace. Les paramètres dont les valeurs sont apprises sont ceux de la matrice de contrôle afin de réduire un peu la taille du problème d'apprentissage par rapport à un apprentissage direct de la matrice du code linéaire. Les paramètres de la fonction à optimiser sont alors les éléments de la matrice de parité du code et seront alors, en général, des bits. Les paramètres prennent donc bien leurs valeurs dans un ensemble de valeurs discrètes, à savoir ici l'ensemble formé des valeurs 0 et 1. Dans d'autres exemples, le code linéaire s'applique à des mots formés de symboles non pas binaires mais appartenant à des ensembles de valeurs plus grands. Il peut s'agir par exemple de codes encodant directement des modulations de phase utilisées dans les communications sans fil. Selon les modes de communication sans fil déployés, les modulations de phase utilisées peuvent être par exemple au nombre de 2, 4, 8 ou 16. Par exemple, dans les cas 4, 8 ou 16, les codes utilisés seront des matrices dont les éléments appartiendront à des corps finis de taille 4, 8 ou 16 (respectivement Z/4Z, Z/8Z et Z/16Z) pour lesquels un minimum et un maximum sont définis (respectivement, 0 et 3, 0 et 7, 0 et 15) et les changements de valeurs de paramètre dans ces corps finis se feront par des additions ou des soustractions d'entiers qui seront limitées le cas échéant pour ne pas dépasser le minimum ou le maximum.

[0069] Selon un mode de réalisation, qui peut être réalisé alternativement ou cumulativement avec les modes précédents, la fonction à optimiser est un code linéaire et la fonction de perte est mesurée selon l'entropie croisée entre un mot résultat de codage par la fonction à optimiser et le décodage de ce mot résultat par un décodeur utilisant la matrice de contrôle associée au code linéaire de la fonction à optimiser et un point d'un lot est fourni à la fonction de perte en appliquant des erreurs aléatoires au mot formé uniquement des symboles minimaux parmi l'ensemble de valeurs discrètes.

[0070] Grâce à ce mode de réalisation, une fonction de perte adaptée à l'apprentissage d'un code linéaire est définie. La fonction de perte est définie comme l'entropie croisée mesurée entre un mot connu, résultat d'un codage, et son décodage après que des erreurs aléatoires aient été appliquées au mot connu. Si le codage est performant, le décodage permettra de retrouver le mot connu même en présence d'erreurs, et l'entropie croisée sera nulle. Optimiser le codage revient donc bien à minimiser cette

fonction de perte. Le mot connu doit être un résultat possible de codage ; pour tous les codes linéaires possibles, le mot formé uniquement des symboles minimaux, par exemple formé uniquement de 0 pour un code binaire, est un tel mot.

[0071] Selon un premier aspect matériel, l'invention a trait à un dispositif de traitement de données mettant en œuvre un procédé d'optimisation d'une fonction, dite fonction à optimiser, la fonction à optimiser étant un code linéaire auquel une matrice de contrôle est associée, la fonction à optimiser comprenant des paramètres qui sont les éléments de ladite matrice de contrôle, lesdits paramètres prenant leurs valeurs parmi un ensemble de valeurs discrètes, le procédé utilisant une autre fonction, dite fonction de perte, ledit dispositif comprenant le module suivant :

- Un module d'exécution d'une étape de descente de gradient de la fonction de perte permettant d'obtenir une valeur de paramètre, les valeurs obtenues optimisant la fonction à optimiser.

[0072] Dans un mode de réalisation du dispositif de traitement de données, le module d'exécution d'une étape de descente de gradient comprend un élément permettant d'obtenir une évolution d'une valeur d'au moins un paramètre en fonction d'une dérivée partielle en au moins un point de la fonction de perte.

[0073] Dans un mode de réalisation, le dispositif du traitement de données comprend en outre un module d'obtention d'une valeur de paramètre à partir de l'exécution d'une étape de descente de gradient de la fonction de perte, la valeur de paramètre obtenue appartenant à l'ensemble de valeurs discrètes.

[0074] Selon un autre aspect matériel, l'invention a trait à un programme d'ordinateur apte à être mis en œuvre par un dispositif de traitement de données, le programme comprenant des instructions de code qui, lorsqu'il est exécuté par un processeur, réalise les étapes du procédé d'obtention défini ci-dessus.

[0075] Selon un autre aspect matériel, l'invention a trait à un support de données sur lequel est enregistré un programme d'ordinateur comprenant une séquence d'instructions pour la mise en œuvre du procédé d'obtention défini ci-dessus.

[0076] Selon un autre aspect matériel, l'invention a trait à un code linéaire obtenu par un procédé selon l'invention mis en œuvre par un dispositif de traitement de données selon l'invention.

[0077] Le support de données peut être n'importe quelle entité ou dispositif capable de stocker les programmes. Par exemple, le support peut comporter un moyen de stockage, tel qu'une ROM, par exemple un CD ROM ou une ROM de circuit microélectronique, ou encore un moyen d'enregistrement magnétique tel qu'un un disque dur. D'autre part, le support peut être un support transmissible tels qu'un signal électrique ou optique, qui peut être acheminé via un câble électrique ou optique, par radio ou par d'autres moyens. Le programme selon l'invention peut être en particulier

téléchargé sur un réseau de type Internet. Alternativement, le support d'informations peut être un circuit intégré dans lequel le programme est incorporé, le circuit étant adapté pour exécuter ou pour être utilisé dans l'exécution du procédé en question. Le programme selon l'invention peut utiliser tout type de technologie informatique en terme de langages de programmation compilés, interprétés, ou une combinaison des deux ainsi qu'en terme de systèmes d'exploitation.

Brève description des figures

[0078] L'invention sera mieux comprise à la lecture de la description qui suit, donnée à titre d'exemple, et faite en référence aux dessins annexés sur lesquels :

[0079] [Fig.1] représente un dispositif de traitement de données mettant en œuvre un procédé d'obtention des valeurs de paramètres d'une fonction parmi un ensemble de valeurs discrètes.

[0080] [Fig.2] illustre un exemple de succession d'étapes mises en œuvre dans un procédé selon l'invention.

[0081] [Fig.3] illustre un exemple de mise en œuvre du procédé permettant d'obtenir les valeurs de paramètres d'un code linéaire.

[0082] [Fig.4] illustre les performances d'un code linéaire dont les valeurs de paramètres ont été obtenues suivant un procédé selon l'invention.

Description détaillée

[0083] La [Fig.1] représente un exemple de dispositif 100 de traitement de données mettant en œuvre un procédé d'obtention des valeurs V1, V2, V3 de paramètres P1, P2, P3 d'une fonction F parmi un ensemble E de valeurs discrètes.

[0084] Le procédé comprend le choix préalable d'une autre fonction L, dite fonction de perte.

[0085] Dans cet exemple de réalisation, le dispositif 100 comprend un module 101 d'exécution d'une étape de descente de gradient de la fonction de perte L. Ce module 101 comprend, dans cet exemple de réalisation, un élément d'obtention pour les paramètres P1, P2, P3 d'évolutions $\Delta 1$, $\Delta 2$, $\Delta 3$ des valeurs V1, V2, V3 des paramètres P1, P2, P3 en fonction des dérivées partielles D1, D2, D3 de la fonction de perte L pour les paramètres P1, P2, P3 en un point de l'ensemble de définition de la fonction de perte L.

[0086] De plus, dans cet exemple de réalisation, le dispositif 100 comprend un module 102 d'obtention de nouvelles valeurs V1, V2, V3 des paramètres P1, P2, P3 selon les évolutions obtenues $\Delta 1$, $\Delta 2$, $\Delta 3$, les nouvelles valeurs obtenues étant contraintes d'appartenir à l'ensemble E de valeurs discrètes.

[0087] Le dispositif 100 de traitement de données présente l'architecture matérielle d'un ordinateur conventionnel. Il comporte notamment au moins un processeur, une

mémoire vive de type RAM et une mémoire morte telle qu'une mémoire de type Flash, ROM, (non représentés sur la figure) ainsi que des dispositifs d'entrée-sortie tels que, dans certains cas, des claviers et/ou écrans (non représentés sur la figure), et des ports réseaux permettant de communiquer avec d'autres entités et serveurs grâce à un réseau de communication tel qu'Internet non représenté sur la figure.

[0088] Les unités de calcul présentes dans le dispositif de traitement 100 et utilisées pour la réalisation du procédé selon l'invention seront, dans des exemples de modes de réalisation des unités de calcul graphiques, dites GPU (non représentées sur la figure). Dans d'autres exemples, les unités de calcul sont des unités CPU (non représentées sur la figure). L'utilisation d'une unité de calcul GPU pour la réalisation du procédé selon l'invention apporte plus de performance aux calculs permettant de réaliser le procédé selon l'invention et donc de réaliser des apprentissages plus poussés pour un temps de calcul donné, au détriment de la souplesse de définition des calculs pouvant être effectués par une unité CPU. Plusieurs modes de réalisation de l'invention sont spécialement adaptés à l'utilisation d'unités de calcul GPU.

[0089] Le dispositif 100 sera par exemple un serveur informatique qui réalise le procédé d'obtention afin de disposer d'un ensemble de valeurs V1, V2, V3 de paramètres P1, P2, P3 d'une fonction F, lesdites valeurs V1, V2, V3 appartenant à un ensemble E de valeurs discrètes. La fonction F est par exemple un code linéaire, et les paramètres P1, P2, P3 du code sont les éléments de la matrice de contrôle du code, soit des bits qui prennent les valeurs 0 ou 1. Dans certains cas, le code peut encoder plusieurs symboles avant transmission, et les valeurs V1, V2, V3 des paramètres P1, P2, P3 ne sont plus des bits mais appartiennent quand même toujours à un ensemble E de valeurs discrètes.

[0090] Dans le cas où le dispositif 100 est un serveur informatique, il va donc réaliser un procédé d'apprentissage qui permettra ensuite de disposer d'une fonction F optimisée, par exemple un nouveau code linéaire. Le dispositif 100 peut également appartenir à une architecture en nuage (traduction de l'anglais *cloud architecture*) et les calculs permettant de réaliser le procédé d'obtention sont alors effectués sur des ressources de calcul, de mémoire et réseau fournies à la demande par le système en nuage (traduction de l'anglais *cloud system*).

[0091] Dans d'autres exemples de réalisation, le dispositif 100 de traitement de données appartient à un équipement qui va mettre en œuvre la fonction F dont les valeurs V1, V2, V3 de paramètres P1, P2, P3 sont obtenues par le procédé mis en œuvre par le dispositif 100. Par exemple, le dispositif 100 peut être inclus dans un équipement de télécommunications, par exemple un routeur, ou une station de base d'un réseau mobile, ou tout autre équipement de télécommunications. Le dispositif 100 peut alors apprendre un code linéaire et le distribuer directement à l'équipement dans lequel il est inclus. Dans des modes de réalisation, le dispositif 100 peut utiliser des données réelles

du trafic transitant par l'équipement de télécommunications dans lequel il est inclus afin d'améliorer la mise en œuvre du procédé d'obtention.

[0092] Le module 101 du dispositif 100 de traitement de données va permettre d'obtenir des évolutions $\Delta 1$, $\Delta 2$, $\Delta 3$ des valeurs V1, V2, V3 des paramètres P1, P2, P3 en fonction des dérivées partielles D1, D2, D3 de la fonction de perte L pour les paramètres P1, P2, P3 en un point de l'ensemble de définition de la fonction de perte L.

[0093] Dans un exemple de réalisation du procédé, le module 101 va calculer l'ensemble des dérivées partielles de la fonction de perte L pour un ensemble, dit lot, de points de l'ensemble de définition de la fonction de perte L par rapport aux paramètres de la fonction à optimiser F. A partir de ces dérivées partielles, le module 101 pourra par exemple en calculer les valeurs moyennes sur l'ensemble des points du lot pour obtenir une seule valeur. Les valeurs moyennes obtenues ainsi peuvent être directement considérées comme les évolutions possibles $\Delta 1$, $\Delta 2$, $\Delta 3$ des valeurs V1, V2, V3 des paramètres P1, P2, P3. De telles évolutions possibles $\Delta 1$, $\Delta 2$, $\Delta 3$ obtenues comme moyennes sur les différents points du lot des dérivées partielles D1, D2, D3 ne seront évidemment pas, en général, des évolutions discrètes.

[0094] Un autre exemple de réalisation du procédé va consister, pour le module 101 du dispositif 100, à se contenter d'obtenir les signes des dérivées partielles de la fonction de perte L pour un ensemble, dit lot, de points de l'ensemble de définition de la fonction de perte L par rapport aux paramètres de la fonction à optimiser F. A partir de ces signes de dérivées partielles, le module 101 pourra par exemple obtenir des évolutions possibles $\Delta 1$, $\Delta 2$, $\Delta 3$ des valeurs V1, V2, V3 des paramètres P1, P2, P3 en additionnant le nombre de signes positifs de dérivées partielles pour un paramètre donnée et en soustrayant le nombre de signes négatifs. Cet exemple de réalisation donnera des évolutions de valeurs de paramètres qui seront discrètes mais qui pourront être plus grande que +1 ou -1.

[0095] Dans des exemples de réalisation, les dérivées partielles D1, D2, D3 dont les valeurs absolues sont inférieures à un seuil donné sont considérées comme étant nulles, afin de ne pas prendre en compte dans l'apprentissage des valeurs trop faibles, qui ne seraient pas significatives et qu'il vaut mieux ignorer. Cette annulation peut prendre place avant de réaliser le moyennage des dérivées partielles D1, D2, D3 sur l'ensemble des points du lot ou bien pour les dérivées partielles moyennées sur l'ensemble des points du lot.

[0096] D'autres exemples de réalisation sont possibles qui permettent au module 101 d'obtenir des évolutions possibles $\Delta 1$, $\Delta 2$, $\Delta 3$ des valeurs V1, V2, V3 des paramètres P1, P2, P3 de la fonction à optimiser F en fonction des dérivées partielles D1, D2, D3 de la fonction de perte L en un ou plusieurs points de l'ensemble de définition de la fonction de perte L.

[0097] Le module 102 du dispositif 100 de traitement de données, quant à lui, va permettre d'obtenir de nouvelles valeurs V1, V2, V3 des paramètres P1, P2, P3 selon les évolutions obtenues Δ1, Δ2, Δ3 par le module 101, les nouvelles valeurs obtenues étant contraintes d'appartenir à l'ensemble E de valeurs discrètes.

[0098] On a vu que le module 101 pouvait obtenir des évolutions possibles Δ1, Δ2, Δ3 des valeurs V1, V2, V3 qui ne sont pas des grandeurs discrètes ou bien, même si elles sont des grandeurs discrètes dans certains exemples de réalisation, qui peuvent être plus grandes que +1 et -1. De telles évolutions appliquées à des valeurs discrètes, qui par exemple dans le cas de l'apprentissage de paramètres binaires sont les valeurs 0 ou 1, aurait pour conséquence que les nouvelles valeurs V1, V2, V3 obtenues après application des évolutions Δ1, Δ2, Δ3 n'appartiendraient plus en général à l'ensemble E auquel les valeurs V1, V2, V3 des paramètres P1, P2, P3 doivent appartenir.

[0099] Le module 102 doit donc appliquer les évolutions obtenues Δ1, Δ2, Δ3 en assurant une contrainte qui assure que les valeurs V1, V2, V3 restent dans l'ensemble E de valeurs discrètes. Plusieurs modes de réalisation sont possibles pour réaliser cela.

[0100] Par exemple, dans certains modes de réalisation, les évolutions Δ1, Δ2, Δ3 sont des grandeurs réelles obtenues par un moyennage des dérivées partielles D1, D2, D3 sur les différents points d'un lot. Dans ces exemples, le module 102 pourra utiliser un système de seuils pour transformer les évolutions réelles Δ1, Δ2, Δ3 obtenues par le module 101 en des évolutions discrètes qui assurent que les nouvelles valeurs V1, V2, V3 appartiendront bien toujours à l'ensemble E de valeurs discrètes.

[0101] De plus, si une évolution discrète s'applique à une valeur déjà au minimum ou au maximum de l'ensemble E de valeurs discrètes, ou proche de ceux-ci, le module 102 va réduire les évolutions Δ1, Δ2, Δ3 afin de s'assurer que le minimum ou le maximum ne sont pas dépassés.

[0102] Dans les exemples où les évolutions Δ1, Δ2, Δ3 obtenues par le module 101 sont déjà des grandeurs discrètes, le module 102 peut également utiliser un système de seuils. Ces seuils peuvent d'une part servir à éviter des évolutions trop fortes, qui pousseraient les valeurs V1, V2, V3 des paramètres P1, P2, P3 en dehors de l'ensemble E de valeurs discrètes. Ces seuils peuvent aussi servir à assurer qu'une évolution ne prendra place que lorsqu'une grande quantité de points d'apprentissage pointent vers une évolution donnée afin d'éviter des oscillations en grand nombre causées par un petit nombre de données non significatives. Par exemple, dans un lot de 32 points, les évolutions Δ1, Δ2, Δ3 peuvent être obtenues en sommant les signes des dérivées partielles D1, D2, D3 et limitées à des évolutions de +1 ou -1 si les paramètres P1, P2, P3 ont des valeurs V1, V2, V3 qui sont binaires. Plutôt que de réaliser ces évolutions de +1 ou -1 dès que les évolutions Δ1, Δ2, Δ3 obtenues par le module 101 sont respectivement positives ou négatives, il est possible d'utiliser un seuil, par

exemple à +30 ou -30 pour réaliser une évolution effective de +1 ou -1 des valeurs V1, V2, V3. Dans cet exemple, une évolution ne sera effective que si 30 points au moins du lot de 32 ont une dérivée partielle D1, D2, D3 du même signe, ce qui indique que la quasi-totalité des points du lot donnent la même indication dans leur dérivée partielle.

[0103] Dans des exemples de réalisation, le procédé est itératif et le module 102 réalise la combinaison des évolutions Δ1, Δ2, Δ3 obtenues par le module 101 lors de plusieurs étapes successives avant de procéder à un changement des valeurs V1, V2, V3 des paramètres P1, P2, P3.

[0104] Cette combinaison peut comprendre, par exemple, l'utilisation de compteurs. Les évolutions Δ1, Δ2, Δ3 obtenues par le module 101 peuvent par exemple se réduire au signe des dérivées partielles D1, D2, D3 pour les paramètres P1, P2, P3, et donc à des incrémentations ou décrémentations de 1 en sens inverse du signe des dérivées partielles D1, D2, D3 moyennées sur l'ensemble des points du lot. Ces évolutions Δ1, Δ2, Δ3 sont alors combinées par le module 102 dans un compteur associé au paramètre P1, P2, P3. L'évolution des valeurs V1, V2, V3 ne seront effectives que lorsqu'un des compteurs dépasse un seuil donné, et s'appliquera aux valeurs des paramètres pour lesquels le compteur associé a atteint le seuil donné. Les autres compteurs seront alors remis à zéro dans des exemples de réalisation. Il est possible aussi de ne pas remettre tous les compteurs à zéro, mais de conserver ceux qui sont proches du seuil donné. Il est également possible de choisir de faire évoluer les valeurs des paramètres associés à des compteurs qui n'ont pas atteint le seuil donné mais qui en sont proches, puis de remettre l'ensemble des compteurs à zéro.

[0105] Dans un autre exemple de combinaison, les valeurs des évolutions Δ1, Δ2, Δ3 obtenues par le module 101 sont égales aux dérivées partielles D1, D2, D3 de la fonction de perte L, éventuellement moyennées sur l'ensemble des points d'un lot de calcul. Ces différentes évolutions Δ1, Δ2, Δ3 sont alors additionnées entre elles dans des compteurs associés aux paramètres et, de même que dans l'exemple précédent, le module 102 réalise une évolution effective des valeurs V1, V2, V3 des paramètres P1, P2, P3 uniquement quand ces compteurs dépassent un seuil donné. Contrairement à l'exemple précédent, cette combinaison prend en compte la grandeur des dérivées partielles D1, D2, D3 et pas seulement leur signe. Comme dans le mode précédent, les évolutions effectives des valeurs V1, V2, V3 doivent se faire dans le sens inverse des signes des dérivées partielles D1, D2, D3 puisque l'optimisation de la fonction F que l'on cherche à réaliser en modifiant les valeurs V1, V2, V3 des paramètres P1, P2, P3 correspond à une minimisation de la fonction de perte L.

[0106] Dans un autre exemple de combinaison, le procédé n'utilise pas des compteurs mais une estimation de l'intervalle de confiance pour les valeurs des dérivées partielles D1, D2, D3. Comme précédemment, les évolutions Δ1, Δ2, Δ3 obtenues sont directement

liées aux dérivées partielles D1, D2, D3. Le module 102 va alors réaliser, au fur et à mesure de la réception des évolutions Δ1, Δ2, Δ3 une estimation de l'intervalle de confiance dans lequel les dérivées partielles pour les paramètres P1, P2, P3 se trouve pour la totalité de l'ensemble de définition. La méthode Agresti-Coull peut être utilisée pour calculer cet intervalle de confiance. Quand une certitude suffisante est atteinte, le module 102 peut réaliser une évolution effective des valeurs V1, V2, V3 des paramètres P1, P2, P3 pour lesquels la confiance est suffisante, et remettre à zéro l'ensemble des estimations pour les mesures suivantes.

[0107] La [Fig.2], quant à elle, présente un exemple de succession d'étapes mises en œuvre dans un procédé selon l'invention.

[0108] Dans cet exemple de réalisation, le procédé est itératif. Une étape d'initialisation des valeurs V1, V2, V3 des paramètres P1, P2, P3 a eu lieu, non représentée sur la figure. Cette initialisation peut être aléatoire, ou dirigée par un expert. Quand elle est aléatoire des probabilités différentes peuvent être données aux différentes valeurs présentes dans l'ensemble E de valeurs discrètes pour initialiser chaque paramètre, ceci afin de donner un point de départ pertinent au procédé.

[0109] L'étape S1 correspond à l'exécution d'une étape de descente du gradient de la fonction de perte L, qui permet l'obtention des évolutions possibles Δ1, Δ2, Δ3 des valeurs V1, V2, V3 par le module 101 en fonction des dérivées partielles D1, D2, D3 de la fonction de perte L par rapport aux paramètres P1, P2, P3. Ces évolutions possibles peuvent être par exemple des incrémentations ou décrémentations de « un » en fonction inverse du signe des dérivées partielles D1, D2, D3 moyennées sur l'ensemble des points du lot.

[0110] L'étape S2 correspond à une combinaison sur des itérations successives des évolutions Δ1, Δ2, Δ3. Dans l'exemple présenté ici, cette combinaison est réalisée en utilisant des compteurs. Les évolutions de +1 et -1 sont ajoutées à chaque itération à des compteurs C1, C2, C3 associés aux paramètres P1, P2, P3.

[0111] Quand les compteurs C1, C2, C3 ont été mis à jour avec les évolutions Δ1, Δ2, Δ3 obtenus à l'étape S1, il est testé si un ou plusieurs compteurs Ci parmi les compteurs C1, C2, C3 a atteint un seuil donné S. Ceci est représenté par un test dans la [Fig.2].

[0112] Si ce n'est pas le cas (branche N du test), le procédé itératif reprend à l'étape S1 avec une obtention de nouvelles évolutions Δ1, Δ2, Δ3 pour un nouveau lot de points appartenant à l'ensemble de définition de la fonction de perte L.

[0113] Si un compteur Ci atteint le seuil donné S (branche Y du test), l'évolution correspondante Δi est appliquée à la valeur Vi. C'est bien sûr le cas pour tous les compteurs qui atteignent le seuil et, dans certains modes de réalisation, cela peut aussi être le cas pour des compteurs proches du seuil.

[0114] Ensuite, dans cet exemple, l'ensemble des compteurs C1, C2, C3 sont remis à zéro, ce qui est indiqué par la mention RAZ dans la [Fig.2], et le procédé se poursuit à l'étape S1 avec un nouveau lot de points appartenant à l'ensemble de définition de la fonction de perte L.

[0115] Plusieurs critères d'arrêt du procédé itératif peuvent être choisis. Le procédé peut s'arrêter quand il n'y a plus eu d'évolutions depuis un certain nombre d'itérations, ou bien au bout d'un nombre fixé d'itérations par exemple. Le procédé peut également s'arrêter après une évaluation de la fonction F à optimiser en utilisant les valeurs obtenues V1, V2, V3 des paramètres P1, P2, P3 à un moment donné, mais une telle évaluation peut prendre un temps significatif à réaliser.

[0116] La [Fig.3], quant à elle, illustre un exemple de mise en œuvre du procédé permettant d'obtenir les valeurs de paramètres d'un code linéaire.

[0117] Dans cet exemple, la fonction à optimiser F est un codage linéaire, et les paramètres P1, P2, P3 forment par exemple une matrice de contrôle de parité qui définissent le codage. Un module de décodage DEC est donné, qui permet de retrouver un mot encodé, puis transmis, même en présence d'un certain nombre d'erreurs de transmissions. Le module de décodage DEC peut être par exemple un décodeur à propagation d'information, afin que la fonction F soit optimisée en prenant en compte ce type de décodage très fréquent. Le module de décodage DEC utilise pour réaliser un décodage un état courant de la matrice de contrôle de parité, représentée par l'élément COD dans la [Fig.3]. L'état courant de la matrice de parité est donné par des valeurs V1, V2, V3 des paramètres P1, P2, P3.

[0118] Pour réaliser le procédé itératif, un lot de points de mesure doit être défini. Pour cela, le mot X formé uniquement de zéros est donné. Un tel mot est un résultat possible d'encodage pour tout code linéaire, et, dans un code linéaire, un résultat possible d'encodage représente toutes les possibilités à des transformations connues près. Le mot X formé uniquement de zéros est donc représentatif. Du bruit ERR est rajouté à ce mot, ce qui est indiqué sur la [Fig.3] par un opérateur + cerclé.

[0119] Le décodeur DEC va alors chercher à corriger l'erreur ERR. Le résultat du décodage DEC pour un point est noté $\hat{X}$. Si le décodeur DEC arrive à corriger l'erreur ERR introduite, le résultat noté $\hat{X}$ sera lui aussi un mot formé uniquement de zéros. La fonction F de codage est optimisée quand le décodeur DEC arrive à corriger le maximum d'erreurs. La fonction de perte L à minimiser est définie comme l'entropie croisée BCE (traduction de l'anglais *binary cross entropy*) entre le mot X original formé uniquement de zéros et le mot résultat noté $\hat{X}$ obtenu après décodage DEC qui a cherché à corriger l'erreur ERR introduite. La fonction de perte L est bien minimale quand le décodage DEC arrive à corriger un maximum d'erreurs ERR introduites.

[0120] Le dispositif de traitement 100 prend donc en entrée les dérivées partielles D1, D2, D3 de la fonction de perte L telle que vue précédemment. Ces dérivées partielles D1, D2, D3 de la fonction de perte L sont obtenues pour un lot de points d'une taille donnée, formés uniquement de zéros et auxquels des erreurs ERR ont été ajoutées. Les deux modules 101 et 102 procèdent alors respectivement à l'obtention des évolutions $\Delta 1$, $\Delta 2$, $\Delta 3$ puis à l'application éventuelle des évolutions $\Delta 1$, $\Delta 2$, $\Delta 3$ pour obtenir de nouvelles valeurs V1, V2, V3 des paramètres P1, P2, P3. Dans des exemples de réalisation, les résultats de plusieurs lots sont combinés entre eux, par exemple en utilisant un mécanisme de compteurs comme vu précédemment.

[0121] Si de nouvelles valeurs V1, V2, V3 sont obtenues, elles sont intégrées à la matrice de parité COD. La fonction F à optimiser est ainsi potentiellement améliorée avant une nouvelle itération du procédé sur un nouveau lot de points qui seront décodés DEC avec les nouvelles valeurs V1, V2, V3 des paramètres P1, P2, P3 de la fonction F à optimiser.

[0122] La [Fig.4], quant à elle, illustre les performances d'un code linéaire dont les valeurs de paramètres ont été obtenues suivant un procédé selon l'invention.

[0123] Dans cette figure, les performances de trois codes linéaires sont représentées par trois lignes brisées formées de points de mesure. Ces trois codes sont des codes (128, 64), c'est-à-dire qu'ils encodent des mots de 64 bits en doublant leur taille et en les portant à 128 bits. La matrice réalisant ces codes a donc une taille de 128 * 64 = 8 192 bits.

[0124] Ces trois codes linéaires sont présentés ici à la taille 128 par 64 et sont dénommés ZC, CCSDS et AR3A. Les performances du code ZC (acronyme pour l'anglais *Zero Coder*) sont représentées par des triangles noirs : c'est ce code qui a été obtenu par la mise en œuvre du procédé de l'invention. Les deux autres codes sont dénommés CCSDS et AR3A (acronymes respectifs de l'anglais *Consultative Committee for Space Data Systems* et *Accumulate Repeat-3 and Accumulate*) et représentés par des cercles noirs et blancs respectivement. Ces deux codes sont construits par des procédés théoriques et heuristiques traditionnels et sont connus pour avoir de très bonnes performances à la taille 128 par 64 bits. Les performances des codes CCSDS et AR3A telles que présentées ici sont issues de l'article suivant : *Code Design for Short Blocks: A Survey*, de Liva, Gaudio, Ninacs et Jerkovits, in Proceedings of EuCNC 2016, disponible sur https:/arxiv.org/abs/1610.00873. Les codes CCSDS et AR3A ont les meilleures performances parmi 14 codes recensés dans cet article.

[0125] Les performances des trois codes sont mesurées de la manière suivante. L'axe horizontal Eb/N0 est une mesure liée au bruit dans un canal de communication. Plus cette mesure est grande, moins la communication est perturbée par le bruit dans le canal. Très précisément, Eb/NO est une mesure de la quantité d'énergie dépensée par symbole d'information (Eb pour énergie par bit) normalisée par la quantité de bruit

subie par le signal portant l'information. Cette grandeur est très couramment utilisée dans le domaine.

[0126] L'axe vertical, quant à lui, mesure les erreurs non corrigées après décodage. L'acronyme BLER signifie *BLock Error Rate*, soit le taux de blocs (mots) en erreur après décodage.

[0127] Les graphes se lisent comme suit : plus la grandeur Eb/N0 est faible, plus du bruit vient perturber la communication et plus le taux d'erreurs non corrigées BLER sera grand. Pour un rapport à Eb/N0 à 1, le taux d'erreurs non corrigées est typiquement à 0,5, soit une erreur non corrigée sur deux, alors que pour un Eb/N0 à 4, le taux d'erreurs non corrigées tombe à un sur mille voire un sur dix mille. Les points du graphe sont mesurés pour un rapport Eb/N0 donné en soumettant un grand nombre de mots, en les codant suivant le code évalué, en les transmettant avec un canal ayant le rapport Eb/N0 donné, en les décodant avec un même décodeur à propagation de connaissances et avec un même nombre d'itérations (200). Le nombre d'erreurs non corrigées pour un code donné donne alors un point de la courbe représentative du code.

[0128] Pour mieux juger des performances des trois codes, deux courbes théoriques sont ajoutées au graphe. La courbe *Sphere Packing* indique la limite en-dessous de laquelle il n'existe aucun code pouvant atteindre ses performances. La courbe *Random Coding Union* indique une zone en-dessous de laquelle on est sûr qu'il existe au moins un code présentant ses performances.

[0129] Le code ZC a été obtenu par la mise en œuvre du procédé objet de l'invention. Pour l'obtenir, il a fallu apprendre 8192 paramètres binaires. Le procédé a été initialisé avec une matrice aléatoire où la densité de 1 à l'initialisation était de 30%. Le mode de réalisation dans cet exemple a consisté à mesurer les gradients sur des lots de taille 8, soit 8 points sur lesquels sont mesurées les dérivées partielles par rapport aux paramètres. Les dérivées partielles sont moyennées sur les huit points du lot puis seuls les signes sont conservés. Ces signes vont servir à incrémenter ou décrémenter un compteur. Le seuil de mise à jour d'un paramètre a été fixé à 20.

[0130] Grace à cette mise en œuvre du procédé, plusieurs codes peuvent être obtenus et vont être évalués jusqu'à retenir le meilleur qui est le code ZC dont les performances sont présentées sur le graphe. On voit que ses performances sont proches des performances théoriques supérieures et équivalentes aux meilleurs codes de l'état de l'art.

[0131] Signalons enfin ici que, dans le présent texte, le terme « module » peut correspondre aussi bien à un composant logiciel qu'à un composant matériel ou un ensemble de composants matériels et logiciels, un composant logiciel correspondant lui-même à un ou plusieurs programmes ou sous-programmes d'ordinateur ou de manière plus générale à tout élément d'un programme apte à mettre en œuvre une fonction ou un ensemble de fonctions telles que décrites pour les modules concernés. De la même

manière, un composant matériel correspond à tout élément d'un ensemble matériel (ou hardware) apte à mettre en œuvre une fonction ou un ensemble de fonctions pour le module concerné (circuit intégré, carte à puce, carte à mémoire, etc.).

Revendications

[Revendication 1] Procédé d'optimisation d'une fonction (F), la fonction (F) étant un code linéaire auquel une matrice de contrôle est associée, la fonction à optimiser comprenant des paramètres (P1, P2, P3) qui sont les éléments de ladite matrice de contrôle, lesdits paramètres (P1, P2, P3) prenant leurs valeurs (V1, V2, V3) parmi un ensemble (E) de valeurs discrètes, le procédé utilisant une autre fonction (L), dite fonction de perte, le procédé, mis en œuvre par un dispositif (100) de traitement de données, comprenant :
– exécuter une étape de descente de gradient de la fonction de perte (L) permettant d'obtenir une valeur (V1, V2, V3) de paramètre (P1, P2, P3), les valeurs (V1, V2, V3) obtenues optimisant la fonction (F).

[Revendication 2] Procédé d'optimisation d'une fonction (F) selon la revendication 1 caractérisé en ce que l'étape de descente de gradient comprend :
– obtenir une évolution (Δ1, Δ2, Δ3) d'une valeur (V1, V2, V3) d'au moins un paramètre (P1, P2, P3) en fonction d'une dérivée partielle (D1, D2, D3) en au moins un point de la fonction de perte (L).

[Revendication 3] Procédé d'optimisation d'une fonction (F) selon la revendication 2 caractérisé en ce qu'une évolution (Δ1, Δ2, Δ3) obtenue d'une valeur (V1, V2, V3) d'au moins un paramètre (P1, P2, P3) considéré comprend une incrémentation par un ou une décrémentation par un en fonction inverse du signe de la dérivée partielle (D1, D2, D3) en au moins un point de la fonction (L) de perte pour le paramètre (P1, P2, P3) considéré ainsi qu'une évolution nulle d'une valeur d'un paramètre considéré si ce paramètre a déjà pour valeur le minimum ou le maximum de l'ensemble (E) de valeurs discrètes.

[Revendication 4] Procédé d'optimisation d'une fonction (F) selon l'une des revendications 2 ou 3 caractérisé en ce que le procédé est itératif et comprend, pour une itération, le choix d'un ensemble de points de l'ensemble de définition de la fonction de perte (L), ledit ensemble de points étant dit lot, et l'étape d'obtention d'une évolution (Δ1, Δ2, Δ3) de la valeur (V1, V2, V3) d'un paramètre (P1, P2, P3) considéré comprend le calcul d'une moyenne de la fonction de perte (L) pour les points du lot et l'évolution (Δ1, Δ2, Δ3) de la valeur du paramètre considéré est fonction inverse de la dérivée partielle de la moyenne de la fonction de perte (L) pour ce paramètre.

[Revendication 5] Procédé d'optimisation d'une fonction (F) selon l'une des revendications 2 ou 3 caractérisé en ce que le procédé est itératif et comprend, pour une itération, le choix d'un ensemble de points de l'ensemble de définition de la fonction de perte (L), ledit ensemble de points étant dit lot, et l'étape d'obtention d'une évolution (Δ1, Δ2, Δ3) de la valeur (V1, V2, V3) d'un paramètre (P1, P2, P3) considéré comprend le calcul de dérivées partielles (D1, D2, D3) de la fonction de perte (L) pour le paramètre considéré (P1, P2, P3) en les points du lot, suivi par la réduction de ces dérivées partielles à des incrémentations ou à des décrémentations en fonction inverse du signe des dérivées partielles, l'évolution (Δ1, Δ2, Δ3) étant obtenue en combinant ces incrémentations et décrémentations.

[Revendication 6] Procédé d'optimisation d'une fonction (F) selon l'une des revendications 2 à 5 caractérisé en ce que la dérivée partielle (D1, D2, D3) en fonction de laquelle une évolution (Δ1, Δ2, Δ3) de valeur (V1, V2, V3) de paramètre (P1, P2, P3) est obtenue est réduite à zéro si sa valeur absolue est inférieure à un seuil donné.

[Revendication 7] Procédé d'optimisation d'une fonction (F) selon l'une des revendications 2 à 6 caractérisé en ce que l'étape d'obtention d'une valeur d'un paramètre comprend une étape de combinaison des évolutions (Δ1, Δ2, Δ3) obtenues avec les évolutions (Δ1, Δ2, Δ3) obtenues lors des étapes précédentes et une étape de mise à jour de la valeur (V1, V2, V3) d'un paramètre (P1, P2, P3) en fonction des évolutions (Δ1, Δ2, Δ3) combinées.

[Revendication 8] Procédé d'optimisation d'une fonction (F) selon l'une des revendications 2 à 7 caractérisé en ce que l'étape d'obtention d'une valeur (V1, V2, V3) d'un paramètre (P1, P2, P3) selon l'évolution (Δ1, Δ2, Δ3) obtenue comprend la mise à jour d'un compteur (C1, C2, C3) associé au paramètre (P1, P2, P3) en fonction des dérivées partielles de la fonction de perte (L) par rapport aux paramètres (P1, P2, P3) respectifs et caractérisé en ce que la valeur (V1, V2, V3) obtenue ne change que lorsque le compteur (C1, C2, C3) associé au paramètre (P1, P2, P3) a dépassé un seuil (S) donné.

[Revendication 9] Procédé d'optimisation d'une fonction (F) selon la revendication 8 caractérisé en ce que, lorsqu'une valeur (V1, V2, V3) d'un paramètre (P1, P2, P3) change après que le compteur (C1, C2, C3) a dépassé le seuil (S) donné, tous les compteurs (C1, C2, C3) associés à des

paramètres (P1, P2, P3) qui sont inférieurs à un autre seuil, inférieur ou égal au seuil (S) donné, sont remis à zéro.

[Revendication 10] Procédé d'optimisation d'une fonction (F) selon l'une des revendications 8 ou 9 caractérisé en ce que le seuil (S) donné est fixé à une valeur entre 10 et 30.

[Revendication 11] Procédé d'optimisation d'une fonction (F) selon l'une des revendications 1 à 10 caractérisé en ce que le procédé comprend une étape d'initialisation aléatoire des valeurs (V1, V2, V3) des paramètres (P1, P2, P3) de la fonction (F) selon une probabilité différenciée pour les différentes valeurs de l'ensemble (E) de valeurs discrètes que peuvent prendre les paramètres (P1, P2, P3).

[Revendication 12] Procédé d'optimisation d'une fonction (F) selon l'une des revendications 4 ou 5 caractérisé en ce que la fonction de perte (L) est mesurée selon l'entropie croisée entre un mot (X) résultat de codage par la fonction (F) et le décodage de ce mot résultat (X) par un décodeur utilisant la matrice de contrôle associée au code linéaire de la fonction (F) et caractérisé en ce qu'un point d'un lot est fourni à la fonction de perte (L) en appliquant des erreurs aléatoires au mot formé uniquement des symboles minimaux parmi l'ensemble (E) de valeurs discrètes.

[Revendication 13] Dispositif de traitement de données (100) mettant en œuvre un procédé d'optimisation d'une fonction (F), la fonction (F) étant un code linéaire auquel une matrice de contrôle est associée, la fonction (F) comprenant des paramètres (P1, P2, P3) qui sont les éléments de ladite matrice de contrôle, lesdits paramètres (P1, P2, P3) prenant leurs valeurs (V1, V2, V3) parmi un ensemble (E) de valeurs discrètes, le procédé utilisant une autre fonction (L), dite fonction de perte, ledit dispositif (100) comprenant le module suivant :

- Un module (101) d'exécution d'une étape de descente de gradient de la fonction de perte (L) permettant d'obtenir une valeur (V1, V2, V3) de paramètre (P1, P2, P3), les valeurs (V1, V2, V3) obtenues optimisant la fonction (F).

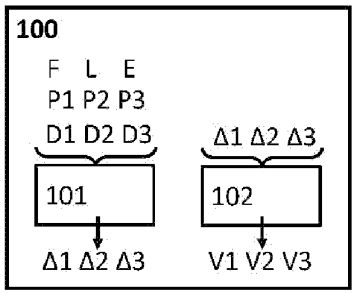
[Revendication 14] Programme d'ordinateur apte à être mis en œuvre par un dispositif (100) de traitement de données selon la revendication 13, le programme comprenant des instructions de code qui, lorsqu'il

est exécuté par un processeur, réalise les étapes du procédé d'optimisation selon l'une des revendications 1 à 12.

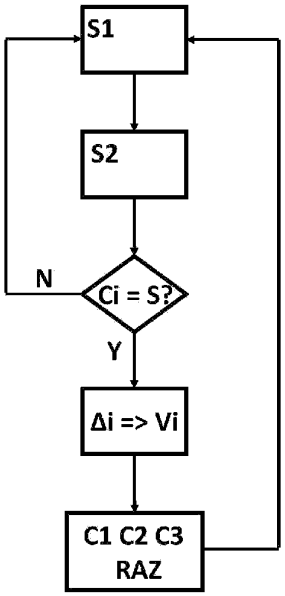
[Revendication 15] Support de données sur lequel est enregistré un programme d'ordinateur selon la revendication 14 comprenant une séquence d'instructions pour la mise en œuvre du procédé d'optimisation selon l'une des revendications 1 à 12.

[Revendication 16] Code linéaire obtenu par un procédé d'optimisation selon l'une des revendications 1 à 12 mis en œuvre par un dispositif de traitement de données (100) selon la revendication 13.

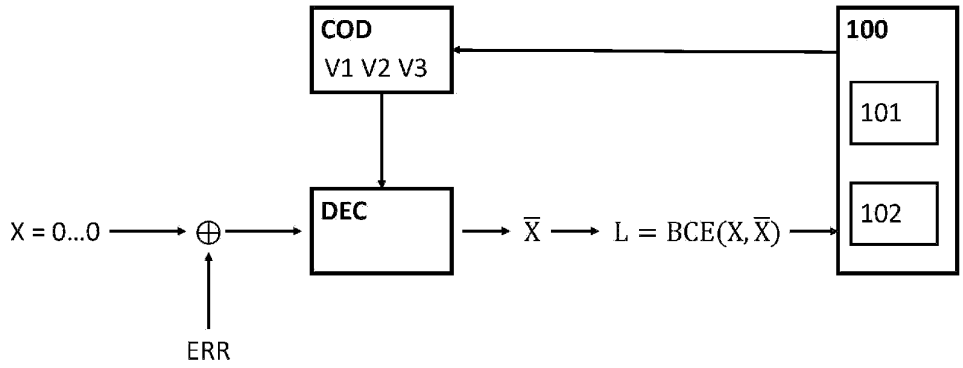
[Fig. 1]


100
F L E
P1 P2 P3
D1 D2 D3
Δ1 Δ2 Δ3
101
102
Δ1 Δ2 Δ3
V1 V2 V3

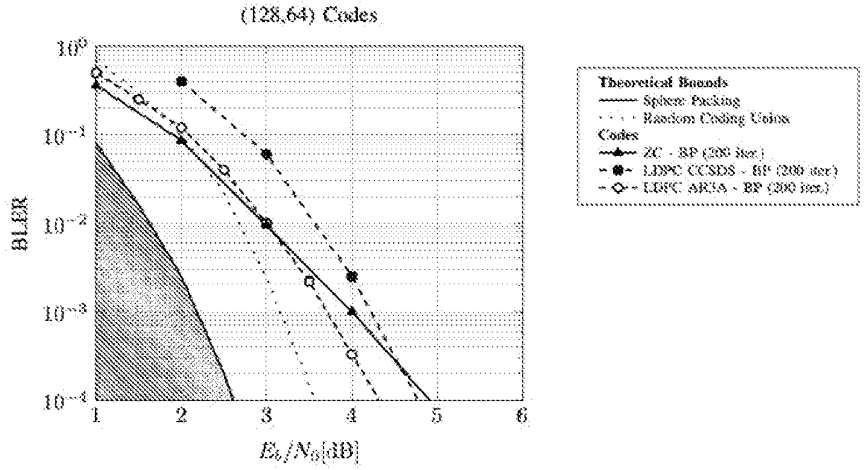
[Fig. 2]


S1
S2
N
Ci = S?
Y
Δi => Vi
C1 C2 C3
RAZ

[Fig. 3]


COD
V1 V2 V3
100
101
102
DEC
X = 0...0
$\overline{X}$
L = BCE(X, $\overline{X}$)
ERR

[Fig. 4]


(128,64) Codes
BLER
E_b/N_0[dB]
Theoretical Bounds
Sphere Packing
Random Coding Union
Codes
ZC - BP (200 iter.)
LDPC CCSDS - BP (200 iter.)
LDPC AR5A - BP (200 iter.)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/EP2025/067607

A. CLASSIFICATION OF SUBJECT MATTER

G06N 3/04(2023.01)i; ***H03M 13/00***(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06N; H03M

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WADAYAMA TADASHI ET AL. "Deep Learning-Aided Trainable Projected Gradient Decoding for LDPC Codes" *2019 IEEE INTERNATIONAL SYMPOSIUM ON INFORMATION THEORY (ISIT), IEEE,* 07 July 2019 (2019-07-07), pages 2444-2448 DOI: 10.1109/ISIT.2019.8849215 XP033619875 sections III-A, section III-D	1-16
X	XIAO XIN ET AL. "Designing Finite Alphabet Iterative Decoders of LDPC Codes Via Recurrent Quantized Neural Networks" *IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE SERVICE CENTER, PISCATAWAY, NJ. USA,* Vol. 68, No. 7, 04 April 2020 (2020-04-04), pages 3963-3974, [retrieved on 2020-07-14] DOI: 10.1109/TCOMM.2020.2985678 ISSN: 0090-6778, XP011798574 sections IV, V	1-16

☑ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
07 October 2025	**15 October 2025**
Name and mailing address of the ISA/EP **European Patent Office** **p.b. 5818, Patentlaan 2, 2280 HV Rijswijk** **Netherlands (Kingdom of the)** Telephone No. **(+31-70)340-2040** Facsimile No. **(+31-70)340-3016**	Authorized officer **Yamajako-Anzala, A** Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/EP2025/067607

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	NACHMANI ELIYA ET AL. "Deep Learning Methods for Improved Decoding of Linear Codes" *IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING, IEEE, US,* Vol. 12, No. 1, 01 February 2018 (2018-02-01), pages 119-131, [retrieved on 2018-02-19] DOI: 10.1109/JSTSP.2017.2788405 ISSN: 1932-4553, XP011678126 sections III, V	1-16

RAPPORT DE RECHERCHE INTERNATIONALE

Demande internationale n°
PCT/EP2025/067607

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
INV. G06N3/04 H03M13/00
ADD.

Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)
G06N H03M

Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche

Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si cela est réalisable, termes de recherche utilisés)
EPO-Internal

C. DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
X	WADAYAMA TADASHI ET AL: "Deep Learning-Aided Trainable Projected Gradient Decoding for LDPC Codes", 2019 IEEE INTERNATIONAL SYMPOSIUM ON INFORMATION THEORY (ISIT), IEEE, 7 juillet 2019 (2019-07-07), pages 2444-2448, XP033619875, DOI: 10.1109/ISIT.2019.8849215 sections III-A, section III-D ----- -/--	1-16

[X] Voir la suite du cadre C pour la fin de la liste des documents

[ ] Les documents de familles de brevets sont indiqués en annexe

* Catégories spéciales de documents cités:

"A" document définissant l'état général de la technique, non considéré comme particulièrement pertinent

"E" document antérieur, mais publié à la date de dépôt international ou après cette date

"L" document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée)

"O" document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens

"P" document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée

"T" document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention

"X" document particulièrement pertinent;; l'invention revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément

"Y" document particulièrement pertinent;; l'invention revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier

"&" document qui fait partie de la même famille de brevets

Date à laquelle la recherche internationale a été effectivement achevée	Date d'expédition du présent rapport de recherche internationale
7 octobre 2025	15/10/2025

Nom et adresse postale de l'administration chargée de la recherche internationale	Fonctionnaire autorisé
Office Européen des Brevets, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Yamajako-Anzala, A

3

Formulaire PCT/ISA/210 (deuxième feuille) (avril 2005)

C(suite). DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
X	XIAO XIN ET AL: "Designing Finite Alphabet Iterative Decoders of LDPC Codes Via Recurrent Quantized Neural Networks", IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE SERVICE CENTER, PISCATAWAY, NJ. USA, vol. 68, no. 7, 4 avril 2020 (2020-04-04), pages 3963-3974, XP011798574, ISSN: 0090-6778, DOI: 10.1109/TCOMM.2020.2985678 [extrait le 2020-07-14] sections IV, V ----- NACHMANI ELIYA ET AL: "Deep Learning Methods for Improved Decoding of Linear Codes", IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING, IEEE, US, vol. 12, no. 1, 1 février 2018 (2018-02-01), pages 119-131, XP011678126, ISSN: 1932-4553, DOI: 10.1109/JSTSP.2017.2788405 [extrait le 2018-02-19] sections III, V -----	1-16 1-16