# Introduction

**La machine et l’être humain :**

La vie quotidienne d’une personne exige une immense quantité de connaissances sur le monde. Une grande partie de ces connaissances sont **subjectives** et **intuitives**, et son donc difficiles à articuler de manière formelle. Le défi de l’intelligence artificielle est de :

* Savoir comment intégrer ces connaissances informelles dans un ordinateur.
* Résoudre les tâches faciles à accomplir pour les gens, mais difficiles à décrire formellement.
* **Approche fondée sur la connaissance (knowledge based approch) :**

Un ordinateur peut raisonner automatiquement au sujet des instructions rédigées dans ces langues formelle en utilisant des règles d’inférence logiques « approche fondée sur la connaissance ». L’un des projets les plus plus célèbres est le projet « Cyc », un moteur d’inférence, est une base de données d’instructions dans un langage appelé Cycl (les déclarations sont saisies par un personnel de superviseurs humains).

Faiblesses :

* Les humains ont du mal à concevoir des règles formelles pour décrire le monde avec précision.
* Les systèmes reposent sur des connaissances codées en dur.
* **Approche apprentissage automatique.**

L’IA doit acquérir ses propres connaissances en extrayant des modèles à partir de données brutes. Algorithmes :

* + Regression logistique (recommander ou non …).
  + Bayésien naïf (séparer des courriels légitimes des spams).

**Dépendance** à l’égard de la **représentation des données** qui leur sont **transmises.**

L’apprentissage des représentations : Mettre en évidence un ensemble caractéristiques pour une tache donnée. Algorithme : Auto-encodeur

La difficulté réside dans le fait que bon nombre de **facteurs de variation influent** sur les **données** (Eg. L’éclairage (jour/nuit), l’angle de vue dans une image).

* Ce qui oblige à désenchevêtrer les facteurs de variation et à rejeter les moins intéressent.
* L’apprentissage profond :

Couche visuelle

Une serie de couches cachées extrait des caractéristiques de plus en plus abstraites de l’entrée. Le modèle détermine lui-même quels concepts sont utiles pour expliquer les relations dans les données observées

Défis remportés :

**Deep Blue d’IBM** a battu le champion du **monde Garry Kasparov** en **1997**. Le jeu d’échec représente un monde très simple (64 emplacements et 32 pièces qui se déplacent de manière strictement délimitée). La difficulté n’était pas de décrire l’ensemble des pièces du jeu et le des coups autorisés)

Les échecs peuvent être décrits par une très courte liste de règles purement formelles.

Nomination de l’apprentissage profond :

1. 1940 – 1960 : cybernétique, fondée sur les **modèles linéaires simples** – f(w, x) = \* + \* = y)

* 1943 : Neurone formel de McCulloch-Pitts, modèle primitif de la fonction cérébrale. Il reconnait 2 labels différents en testant si l’entrée est positive ou négative. Les poids sont définis par des humains.
* 1958 : Perceptron de Rosenblatt, premier modèle linéaire (1 seul neurone) capable d’apprendre les poids à partir d’exemples, à l’aide de l’algorithme Descente de gradient stochastique.
* 1960 : Windrow et Hoff, élément linéaire adaptatif (ADALINE – Modèle linéaire), renvoie la valeur réelle (prévision) de f(w, x). L’algorithme d’optimisation est un cas particulier de l’algorithme Descente de gradient stochastique.
* Les limites des modèles linéaires
  + Apprendre la fonction Xor.
    - Première chute.
    - Désaffection vers l’inspiration biologique (**neuroscience**).

1. 1980 – 1990 : connexionnisme (traitement parallèle/représentation distribué), fondée sur la rétropropagation.

Le concept :

Une entrée est représentée par plusieurs variables, chaque variable doit être impliquée dans la représentation de nombreuses entrées possibles.

1. 2006 : Apprentissage profond, fondée sur les Réseaux de Neurones Artificiels.

2006 : Hinton et al.

2007 : Bengio et al.

Inspiration biologique :

L’apprentissage profond puise ses inspirations dans la neuroscience, l’algèbre linéaire, la probabilité, la théorie de l’information et l’optimisation numérique.

**Neuroscience Computationnelle** : Vise à comprendre comment le cerveau fonctionne, au niveau algorithmique, en construisant des modèles plus précis du fonctionnement réel du cerveau. Ce domaine est distinct de l’apprentissage profond, qui vise à construire des systèmes informatiques profonds capables de résoudre avec succès les tâches exigeant de l’intelligence.

La principale raison de la diminution du rôle des neurosciences dans la recherche sur l’apprentissage profond aujourdhui est le manque d’information sur le cerveau.

# Algèbre linéaire

### Scalaires vecteurs matrices et tenseurs

|  |  |
| --- | --- |
| Scalaire | Un nombre unique |
| Vecteur | V est écrit sous la forme de colonne.  Si chaque élément est dans R, et que le vecteur n éléments, alors le vecteur se trouve dans un ensemble formé en prenant le produit cartésien de R n fois avec lui-même.  On peut considérer les vecteurs comme des points d’identification dans l’espace, chaque élément donnant la coordonnée le long d’un axe différent.  Produit scalaire est le produit entre deux vecteurs : s y = si = 0, alors perpendiculaire (produit vectoriel => colinéaire) |
| Matrice | Est un tableau bidimensionnel.  Un vecteur est considéré comme une matrice à une seule colonne, sa transposition est une matrice à une seule ligne.  Un scalaire est une matrice avec une seule entrée  Propriétés :  Additionner :  Matrice et scalaire : = + b  Matrice et vecteur :  C = + , le vecteur v est ajouté à chaque **ligne** de la matrice.  C = A + v, est une copie implicite de v à plusieurs endroits « Diffusion »  Produit terme à terme :A B  Distributivité : A(B+C) = AB + AC  Associativité : (BC) = (AB)C  Transposition d’un produit matricielle :  **Non commutative** : AB != BA, **n’est pas vérifié contrairement à la multiplication scalaire** |
| Tensor | Est un tableau multidimensionnel |

Système d’équation :

Sachant : multiplication à gauche puis à droite,

A.x = b

Savoir si l’équation à une solution, revient à tester si b est dans le sous-espace vectoriel engendré par les colonnes de A.

Pour que Ax = b ait une solution pour toutes les valeurs de b , il faut que l’espace colonne de A soit , si un point dans est exclu de cette espace, ce point est une valeur potentielle de b qui n’a pas de solution.

La contrainte sur l’espace colonne de A vis-à-vis de implique immédiatement que A doit avoir au moins m colonnes, **n >= m (condition nécessaire),** sinon la dimension de l’espace colonne serait inférieure à m.

Exemple :

A une matrice 3X2 et b en 3D, x est donc en 2D.

Modifier les valeurs de x nous permet au mieux de tracer un plan 2D dans . L’équation a une solution si et seulement si b se trouve dans ce plan.

Condition suffisante et nécessaire :

Les colonnes ne doivent ni etre redondantes ni avoir de dépendance linéaire, sinon elle n’ajoute aucun point à l’espace vectoriel engendré par l’ensemble. Elles doivent être linéairement indépendante pour englober la totalité de .

**Matrice singulière : matrice carrée et que les colonnes soient linéairement indépendantes.**

**On préfère avoir plus de colonne (match avec b) que de ligne !**

La norme :

Pour mesurer la taille d’un vecteur, on utiliser la fonction F norme avec p , qui associe un vecteur à un scalaire non négatif. Intuitivement, la norme d’un vecteur x mesure la distance entre l’origine et le point x.

La fonction norme F, vérifie :

F(x) = 0 🡺 x = 0

F(x + y) <= f(x) + f(y), inégalité triangulaire

F(a.x= |a|.F(x), a

: la norme euclidienne, distance euclidienne de l’origine au point identifié par x, désignée par ||x|| =

En ML, on travaille plus avec la norme L2 au carré qu’avec une norme **L2** en elle-même. Car la **dérivée** des éléments de x de la norme L2 au carrée ne dépend que de l’élément correspondant de x, tandis que toutes les dérivées de la norme L2 dépendent du vecteur entier. La norme L2 au carré peut être **indésirable** parce qu’elle **augmente très faiblement près de zéro**. Dans ce cas, se tourner vers une norme L1 pour faire la **différence entre les éléments nuls et non nuls.** Chaque fois qu’un élément de c s’éloigne de 0 avec un écart , L1 augmente de

La norme L0, mesure la taille d’un vecteur selon le nombre d’élément non nul (terminologie fausse, utiliser la norme L1)

La norme uniforme L.

La norme de Frobenius : norme d’une matrice

Vecteurs orthonormés : vecteur orthogonaux et unitaire (norme =1)

Les matrices diagonales et les matrices orthogonales sont faciles à manipuler.

Tr(A) = sommes des éléments diagonaux

Déterminant :

La valeur absolue du déterminant peut être considérée comme une norme, sa multiplication par la matrice contracte ou augmente l’espace.

Si déterminant = 0, l’espace se contracte complétement le long d’au moins une dimension, ce qui lui fait perdre tout son volume.

Si le déterminant est égal à 1, alors la transformation préserve le volume.

Décomposition en éléments propres : décomposer une matrice en un ensemble de vecteurs propres et de valeurs propres. Cette décomposition permet d’étirer l’espace dans les directions désirées.

Un vecteur propre d’une matrice carrée A est un vecteur v non nul, tel que la multiplication par A ne modifie que l’échelle de v. A.v = l.v, l la valeur propre

APC une fonction encodage décodage :

Pour rendre l’encodage facile, l’APC contraint les colonnes à être orthogonales et unitaire pour n’avoir qu’une solution unique.

Minimise la distance entre le point d’entrée x et sa reconstruction

# Probabilité et théorie de l’information

L’informatique est un environnement relativement **déterministe**. Néanmoins, ML on utilise la théorie de la l’information et les probabilités pour étudier des phénomènes aléatoires (non déterministe) et faire face à **l’incertitude.**

En ML, la théorie de l’information s’illustre en deux manières :

1. Les lois de probabilité informent sur la manière dont le système d’IA devrait raisonner, ce qui permet de concevoir des algorithmes pour calculer ou approximer diverses expressions dérivées.
2. Utiliser la probabilité et la statistique pour analyser le comportement théorique des systèmes IA.

* **La probabilité permet de raisonner en présence d’incertitude.**
* **La théorie de l’information permet de quantifier ces incertitudes par le biais de lois de probabilité.**

**A partir de 1980, on quantifie l’incertitude à l’aide de notion de probabilité.**

**Il existe 3 sources d’incertitude :**

1. **La stochasticité inhérente ou système modélisé : Exemple mécanique quantique**
2. **Observabilité incomplète : Même les systèmes déterministes peuvent paraître stochastique lorsque nous ne pouvons pas observer toutes les variables qui déterminent le comportement du système.**

**Exemple : choix déterministe mais résultat incertain.**

1. **Modélisation incomplète : Lorsqu’un modèle doit rejeter une information qui a été observée.**

En préfère les règles simples et incertaines plutôt que les règles complexes et certaines. Car, elles sont difficiles à développer, à entretenir, à communiquer et reste sujettes à l’echèc.

## Loi de probabilité fréquentiste vs loi de probabilité bayésienne

Les lois de probabilités très utile et applicable pour des événements **répétables** « Probabilité fréquentiste » Dans le cas contraire, on utilise la probabilité pour représenter un degré de croyance, lié aux niveaux qualitatifs de certitude « probabilité bayésienne » (1 : certitude absolue).

La probabilité est considérée comme l’extension de la logique pour faire face à l’incertitude.

|  |  |
| --- | --- |
| Logique | Ensemble de règle formelle pour déterminer quelles propositions sont vraies ou fausses étant donné l’hypothèse selon laquelle un autre ensemble de proposition soit vrai ou faux. |
| Probabilité | Fournit des règles formelles pour déterminer la probabilité qu’une proposition soit vraie compte tenu de la vraisemblance d’autres propositions. |

**Variable aléatoire :** Une application définie sur l’ensemble des résultats possibles d’une expérience aléatoire (Prend différentes valeurs de manières aléatoires/description des états possibles). Elle doit être couplée à une loi de probabilité qui spécifie la probabilité de chacun de ces états. Les variables aléatoires peuvent être discrètes ou continues (à dentsité).

**Lois de probabilité :** est une description de la vraisemblance selon laquelle une VA, ou un ensemble de VA, prenne chacun de ses états (valeurs possibles).

### Espérance, variance et covariance

|  |  |  |
| --- | --- | --- |
| Variable discrète | Variable continue | |
| Fonction de masse (FM) | Densité de probabilité (DP) | |
| Une loi de probabilité sur des variables discrètes peut-être décrite à l’aide d’une fonction de masse P(x = x).  Elle peut également agir sur plusieurs variables en même temps **« probabilité jointe – P(x = x, y = y)»**  **P doit vérifier :**   * Le domaine de P doit être l’ensemble des états possibles de x. | Une loi de probabilité sur des variables continues peut-être décrite à l’aide d’une densité de probabilité (DP).  Théorie de la mesure  Une DP ne donne pas la probabilité d’un état spécifique, elle donne la probabilité d’une région infinitésimale de volume  **P doit vérifier :**   * Le domaine de P doit être l’ensemble des états possibles de x. | |
| La loi de probabilité marginale : On connait la loi de P(x, y) et on souhaite savoir la loi de probabilité d’un sous ensemble de celle-ci. | | |
|  | |  |
| Loi de probabilité conditionnelle | | |
| Expérience  Une fonction f(x) par rapport à une loi de probabilité P(x) est la moyenne que f prend lorsque x est tirée de P.  L’espérance est linéaire. | | |
|  | |  |
| Variance  Donne une mesure de dispersion sur les valeurs d’une fonction d’une variable aléatoire x lorsque nous échantillons différentes valeurs de x de sa loi de probabilité. | | |
| Covariance  Donne une idée de la relation linéaire entre deux valeurs, ainsi que l’échelle de ces variables.   * Covariance nulle => variables indépendantes (pas de dépendance lin éaire). * Covariance positif => les valeurs changent bcp et sont toues deux très éloignées de leurs moyennes respectives en même temps. * Covariance négatif => une augmente et l’autre décroit (vice versa).   La covariance et l’indépendance sont des concepts liés mais distincts. Car l’indépendance est une exigence plus forte que la covariance nulle.  Il se peut que deux Vas soient dépendantes mais avec une covariance nulle. | | |

### Loi de probabilités usuelles.

1. Loi normale :

Un bon choix par défaut pour deux raisons :

1. Le théorème centrale limite montre que la somme d’un grand nombre de Vas indépendantes suit approximativement une loi normale. En pratique, ce signifie que de nombreux systèmes peuvent être modélisés avec succès comme des bruits normalement distribués.
2. De toutes les lois de probabilités possibles ayant la même variance, la loi normale code la quantité maximale d’incertitude sur les nombres réels. Elle insère donc le moins de connaissances préalables dans un modèles.

La loi normale se généralise sur Rn. Cette généralisation est connue sous le nom de la loi normale multivariée.

1. La loi exponentielle et de Laplace

Pratique quand on a besoin d’une loi de probabilité ayant un point net à x = 0.

Propriétés utiles de fonctions usuelles.

Fonction sigmoïde : couramment utilisée pour représenter le paramètre d’une loi de Bernoulli. Elle sature en , c.a.d. elle devient très plate et insensible aux petits changements de valeurs dans son entrée.

La fonction Softplus/logit : couramment utilisée pour représenter les paramètres mu et sigma de loi normale. Car, elle a pour ensemble image ]0,

La fonction softmax/exponentielle normalisée : utilisée pour prédire les probabilités associées à une distribution multinoulli.

### La théorie de l’information :

Est la branche des mathématique appliquées qui concerne la mesure de la quantité de l’information présente dans un signal. Initialement inventée pour étudier l’envoi de message à partir d’alphabets discrets sur un canal matériel ou énergétique. Dans ce contexte la TI explique comment concevoir des codes optimaux et calculer la longueur attendue des messages échantillonnés à partir de lois de probabilités spécifiques en utilisant divers schémas de codage. En ML, on peut l’appliquer à :

* Des variables continues lorsque certaines des interprétations en termes de longueur des messages ne s’appliquent pas.
* Caractériser des lois de probabilités.
* Quantifier la similarité entre les lois de probabilités.

L’idée fondamentale : apprendre qu’un événement improbable (on le représente par un contenu d’information plus élevé) s’est réalisé est plus instructif que d’apprendre qu’un événement probable s’est réalisé (on le représente par un faible contenu d’information, événement garantie ne devrait avoir aucun contenu d’information).

## Calcul numérique

Résoudre des problèmes mathématiques par des méthodes qui mettent à jour les estimations de la solution au moyen itératif, plutôt que de dériver analytiquement une formule pour fournir une expression symbolique de la solution correcte.

Les opérations comprennent :

* L’optimisation trouver la valeur d’un argument qui minimise/maximise une fonction.
* La résolution de systèmes d’équation linéaires.

Problème : un nombre réel ne peut pas être représenté avec précision, il est limité par la quantité fini de mémoire (nombre fini de configuration de bits). Pour représenter des nombres réels, une erreur d’approximation (erreur d’arrondi) est faite lors de la représentation.

L’erreur d’arrondi est problématique lorsque qu’elle se propage au cours de nombreuses opérations et peut faire échouer dans la pratique des algorithmes qui fonctionnent en théorie s’ils ne sont pas conçus pour minimiser l’accumulation d’erreurs d’arrondi.

Forme d’erreur :

|  |  |
| --- | --- |
| Underflow | Overflow |
| Des chiffres proches de 0 sont arrondis à 0.  Les fonctions se comportent différents lorsque leur argument est égal à 0 ou proche de 0. | Des chiffres de grande magnitude sont approximés comme |
| * Environnement logiciel, soulève des exceptions : * Division par zéros. * Logarithmique de 0. * Not a Number (NaN). * La fonction softmax doit être stabilisée contre l’underflow et l’overflow.   Les difficultés peuvent être résolues en évaluant softmax(z), z = x -).  Une manipulation algébrique simple montre que la softmax n’st pas modifiée analytiquement en ajoutant ou en soustrayant un scalaire du vecteur d’entrée.  Soustraire ) implique que l’argument le plus grand de la fonction expo devient 0. Sinon, exclu l’underflow car le dénominateur vaudra 1. | |

### Validation croisée

Diviser l’ensemble de données en un ensemble d’apprentissage/test fixe est problématique lorsque la taille de **l’ensemble de test petit.** Un petit ensemble de test implique une **incertitude statistique autour de l’erreur moyenne estimée**, il devient alors difficile de prétendre que l’algorithme A fonctionné mieux que l’algorithme B sur la tâche C. L’idée est de répéter l’ensemble d’apprentissage et de tester le calcul sur différents sous-ensemble choisis au hasard ⇒ k-fold, ou l’erreur de test est estimée en calculant l’erreur de test moyenne sur l’ensemble des k essais.

Problème quand on calcul à l’aide d’un nombe fini d’échantillon un estimateur du véritable paramètre, le résultat est incertain ! Il n’existe pas d’estimateur non biaisés de la variance de ces estimateurs d’erreur moyennes, des approximations sont utilisées.

Estimateur biais/Variance

Estimateur ponctuelle (statistique) / teta chapeau / approximation (terme non approprié) != paramètres teta réel.

On s'intéresse à son bais et sa variance

1. Le biais:

Il fournit une valeur proche du paramètre, **qui dépend des données, les données sont tirées aléatoirement IID, donc teta une VA**).

Biais(teta chapeau) = E(teta chapeau) - teta réel, défini sur les données et la vraie valeur du paramètre définissant la distribution génératrice des données.

Un estimateur est dit sans biais si Biais (teta chapeau) = 0

Un estimateur est dit asymptotiquement sans biais si limite Biais(teta chapeau) = 0

1. Variance : Voir comment il varie en fonction de l'échantillonnage pour quantifier le degré de variation prévu pour l’estimateur.

Le biais et la variance mesurent deux erreurs différentes d’un estimateur.

**Le bais mesure l’écart prévu par rapport à la valeur réelle de la fonction ou du paramètre.**

**La variance, fournit une mesure de l’écart par rapport à la valeur en espérance possiblement fournie par un quelconque rééchantillonnage des données.**

Compromis ?

Dans un cadre applicatif, la validation croisée est très efficace. La relation entre biais et variance est liée avec le sur/sous apprentissage.

Généralisation : capacité d’avoir de bonne performance sur des entrées non observées précédemment.

Optimisation « Réduire l’erreur d’entraînement et de test »

Tâche d’apprentissage automatique :

1. Classification
2. Classification avec entrées manquantes (fréquent en médecine)
3. Régression (prévision)
4. Transcription : l’algorithme d’apprentissage observe une représentation relativement non structurée d’un certain type de données et de la transcrire sous forme de texte discret. Exemple :

Entrée : image de texte – Sortie : séquence de caractère.

Entrée : onde audio – Sortie : séquence de caractère/code d’identification de mot.

1. Traduction automatique : convertir une séquence en une autre séquence.
2. Sortie structurée (tache de transcription, de traduction):

Exemple :

Conversion d’une phrase en un arbre qui décrit la structure grammaticale)

Segmentation d’image fondée sur une classification ou le seuillage des pixels en fonction de leur intensité

1. Détection d’anomalie :

Exemple : Fraude à la carte de crédit (la distribution de probabilité du voleur et differente que celle de la carte de crédit).

1. Estimation de densité
2. Synthèse et échantillonnage
3. Imputation de valeurs manquantes
4. Débruitage

Défis en optimisation des réseaux de neurones :

1. Mauvais conditionnement
2. Minima locaux :

Optimisation de fonction convexe :

Trouver un minimum local, sachant que tout minimum local est garanti d’être un minimum global.

Optimisation de fonction non convexe (comme les réseaux de neurones) :

Il est possible d’obtenir plusieurs minima locaux. Plus le réseau est profond plus il a un grand nombre de minima locaux, mais les minima locaux sont équivalents les uns aux autres en termes de valeur de la fonction de coût (par rapport au minimum global). Par conséquent, ces minima locaux ne constituent pas une forme problématique de non-convexité.

« Dans des espaces à grande dimension, il est difficile de s’assurer que les minima locaux sont problématiques – domaine de recherche ouvert »

1. Les points-selles, région plates et plateaux :
2. Falaise :

Les réseaux de neurones possédant de nombreuses couches présentent souvent des régions extrêmement raides (falaise) ; une mise à jour fait varier les paramètres très violemment (sauter de la falaise). Ce phénomène est courant avec les réseaux de neurones récurrents. On peut l’éviter avec une heuristique de seuillage du gradient.

1. Dépendance à long terme :

Avec les RNNs la fonction objective contient souvent des non-linéarités nettes dans l’espace des paramètres résultant de la multiplication de plusieurs paramètres. Cette non-linéarité donne lieu à des dérivées très élevées en certains endroits. Lorsque les paramètres se rapprochent d’une telle région « falaise » une mise à jour peut catapulter les paramètres très loin.

1. Gradient inexacte
2. Défaut de correspondance entre structure locale et globale
3. Limite théorique de l’optimisation

Réseaux de neurones récurrents

Les réseaux de neurones récurrents sont une famille de réseaux neuronaux spécialisée dans le traitement de données séquentielles. Tout comme un réseau convolutionnel est un réseau de neurones spécialisé dans le traitement d’une grille de valeurs X telle qu’une image.

« Apprentissage d’un seul modèle partagé » Un réseau de neurone récurrent partage les memes poids à travers plusieurs pas de temps ; En effet, on applique une fonction de transition avec les mêmes paramètres à chaque pas de temps sur différentes séquences, de longueur variable et non vus pendant la phase d’apprentissage. On n’apprend pas une fonction pour chaque pas de temps.

Déplier un calcul récursif ou récurrent dans un graphe de calcul qui a une structure répétitive, correspond typiquement à une chaine d’événements. Le dépliement de ce graphe entraîne le partage des paramètres à travers une structure de réseau profonde.

L’état caché est une sorte de résumé :

* Des aspects de la séquence passée des entrées jusqu’à t.
* A perte car il correspond une séquence de longueur arbitraire à un vecteur de longueur fixe ).
* Sélectif car il conserve certains aspects de la séquence passée avec plus/moins de précision pour certaines caractéristiques.

Avec les idées : dépliement de graphe et de partage de paramètres on peut concevoir une grande variété de réseaux de neurones récurrents :

* Structure causale (l’état à l’instant t ne capture que les informations du passé)
* One to one
* many to one
* many to many …
* RNR bidirectionnels

Un réseau de neurones récurrents est universel dans le sens ou toute fonction calculable par une machine de Turing peut être calculée par un tel réseau récurrent d’une taille finie. …

Le temps d’exécution est , et ne peut pas être réduit par parallélisation car le graphe de propagation avant est intrinsèquement séquentiel.

Le cout en espace mémoire est , car les états calculés dans la phase avant doivent être stockés jusqu’à ce qu’ils soient réutilisés pendant la passe arrière.

L’algorithme de rétropropagation appliqué au graphe déplié est la rétropropagation temporelle BPTT.

Défi des dépendances à long terme :

Le gradient propagé sur de nombreuses étapes ont tendance à disparaître ou à exploser du au multiplication de nombreuses jacobiennes. Pour l’éviter il faudrait que les poids restent dans une région de l’espace des paramètres ou les gradients ne disparaissent/n’explosent pas.

Le gradient d’une interaction à long terme à une amplitude exponentiellement plus petite que le gradient d’une interaction à court terme.

Réseaux à états échoïques :

Les poids sont difficiles à apprendre dans un RNN, une approche proposée par Jaeger et Mass, est de fixer les poids récurrents de telle sorte qu les unités cachées récurrentes capturent de manière satisfaisante l’historique des entrées passées, et apprennent seulement les poids de sortie.

L’ESN et le LSM sont des machines à réservoir pour indiquer le fait que les unités cachées forment un réservoir de caractéristiques temporelles qui peuvent saisir différents aspects de l’historique de l’entrées. Le critère d’entraînement est rendu convexe par rapport aux poids de sortie (régression linéaire + critère d’apprentissage erreur quadratique moyenne = fonction convexe, qui peut être résolue de manière fiable).

La question importante : Comment définir les poids d’entrée et les poids récurrents pour qu’un riche historique puisse être représenté dans l’état récurrent du réseau de neurones ?

* Considérer le réseau récurrent comme un système dynamique, et de fixer les poids d’entrée et les poids récurrents de telle sorte que le système dynamique soit proche du bord de stabilité.

Le rayon spectral  < 1, si linéaire sinon > 1

Texte :

On récupère le **corpus**, on fait une **tokenisation** (découpage en mots des différents documents qui constituent votre corpus), puis on applique une **normalisation** pour enlever les mots non discriminants.

Normalisation :

* Suppression des stopword.
* Lemmatisation : on représente le lemme (forme canonique (être), infinitif, singulier), nécessite un dictionnaire.
* Racinisation/stemming : conserver que la racine des mots étudiés (supprimer le suffixe et le préfixe).

Juger/critiquer un document en le représentant par un bag-of-word. Il existe plusieurs façons de le représenté :

Représentation de comptage creuse - fréquence d’apparition : Mots indépendamment les uns des autres avec la fréquence d’apparition et sans prendre en compte le sens.

1. Non pondérées :
2. Représentation brute : considérer tous les mots du corpus avec normalisation.
3. Représenter chaque document par un vecteur de la taille du vocabulaire et on utilisera la matrice composée de l’ensemble de ces N documents qui forment le corpus comme entrée de nos algorithmes.
4. Représenter par un groupe de mots (n-grammes) :

« Je mange une pomme » 🡺 bi-grammes {(je, mange), (mange, une) et (une, pomme)}

Pourquoi ?

* Etudier le sens d’un mot il faut observer son contexte.
* Etudier une forme de dépendance plus ou moins grande entre les mots.
* On peut faire des probabilités conditionnelles.

1. Pondérées :
2. TF-IDF : est une métrique utilisée comme indicateur de similarité

Poids=fréquence du terme×indicateur similarité

Problème de ces représentations : Matrice creuse qui peuvent biaiser les algorithmes.

Représentation type word2vec dense - dans laquelle le mot possède une représentation dans un espace qui le positionne en fonction des mots adjacents : Créer des caractérisations du texte (capturer le sens des mots) dans un espace multidimensionnel (créer par des systèmes de compression) plus représentatifs du sens des mots. En pratique, il s’agit de rapprocher les mots similaires (similarité (probabiliste) synonymes ou meme champs lexical)

L’idée fondamentale reste la même : compresser de manière non supervisée la représentation d’un mot à partir d’un gros corpus de texte représentatif de votre langage, afin d’obtenir un vecteur dense qui permet de visualiser et de fournir à nos algorithmes des features plus intéressantes.

1. Word2Vect : prend en compte le “contexte” dans lequel le mot a été trouvé. On appelle cette hypothèse distributional hypothesi.

L’idée est de compresser notre corpus vers un dictionnaire de vecteurs denses de dimension bien inférieure choisie.

Word Embedding – plongement de mot : une représentation latente d’un mot

Il existe deux principales méthodes en word2vect.

* « Continuous Bag of Words » (CBOW), qui entraîne le réseau de neurones pour prédire un mot en fonction de son contexte, c’est à dire les mots avant/après dans une phrase.

Entrée : une fenêtre autour du mot (contexte)

Sortie : prédire le mot

* Skip-gram : on essaie de prédire le contexte en fonction du mot.

L’idée est d’utiliser les représentations vectorielles classiques du bag of word du mot qu’on veut représenter et des mots l’entourant. Il existe des modèles déjà entrainés en anglais « Glove », « fast text » ou on peut faire notre propre embedding pour qu’il soit spécifique un contexte donné.

Word2vect : Mot 🡺 vecteur 🡺 plonge dans un espace de dimension inférieur (20-100)

Modélisation automatique de topic :

LDA non supérvisé :

L’hypothese :

Chaque mot