Différents algorithmes d’apprentissage

Table des matières

[Markov Chain Monte Carlo 2](#_Toc488398822)

[Hidden Markov Model 2](#_Toc488398823)

[Boltzmann Machines 2](#_Toc488398824)

[Restricted Boltzmann Machines 4](#_Toc488398825)

[Deep Boltzmann Machines 5](#_Toc488398826)

[Deep Belief Networks 6](#_Toc488398827)

[Auto-encoders 8](#_Toc488398828)

[Sparse auto-encoders 9](#_Toc488398829)

[Denoising auto-encoders (DAE) 9](#_Toc488398830)

[Convolutional auto-encoder (CAE) 9](#_Toc488398831)

[Curriculum learning idea 10](#_Toc488398832)

[Semi Supervised Embedding 10](#_Toc488398833)

[Natural Gradient 11](#_Toc488398834)

[Recurrent Neural Networks 11](#_Toc488398835)

[Echo state Networks 11](#_Toc488398836)

[Stochastique Maximum Likelihood (SML) 12](#_Toc488398837)

[Contrastive divergence (CD) 13](#_Toc488398838)

[Independent Subspace Analysis 13](#_Toc488398839)

[Manifold Tangent Classifier 14](#_Toc488398840)

# Markov Chain Monte Carlo

Les méthodes de Monte-Carlo par chaînes de Markov, ou méthodes MCMC pour Markov Chain Monte Carlo, sont une classe de méthodes d’échantillonnage à partir de distributions de probabilité. Ces méthodes de Monte Carlo se basent sur le parcours de chaînes de Markov qui ont pour lois stationnaires les distributions à échantillonner.

On se place dans un espace vectoriel de dimension finie . On veut générer aléatoirement des vecteurs suivant une distribution de probabilité .

Les méthodes de Monte-Carlo par chaînes de Markov consistent à générer un vecteur uniquement à partie de la donnée du vecteur c’est donc un processus sans mémoire, ce qui caractérise les chaines de Markov. Il faut donc trouver un générateur aléatoire avec une distribution de probabilité qui permette de générer à partir de . On remplace ainsi le problème de génération avec une distribution par problèmes de génération avec des distributions , que l’on espère plus simples.

# Hidden Markov Model

Principe du modèle de Markov caché :

* Un ensemble d’états possibles :
* Processus passe d’un état à l’autre, générant ainsi une séquence
* Principe des chaines de Markov : probabilité d’occurrence d’un état dépend uniquement de l’état précédent :
* Un modèle de Markov est défini par les probabilités de transitions, et les probabilités initiales
* Les états ne sont pas visibles, mais chaque état génère un état observable parmi un certain nombre :
* Pour définir un modèle de Markov, on doit définir :
  + Matrice des probabilités de transition ,
  + Matrice des probabilités des états observables ,

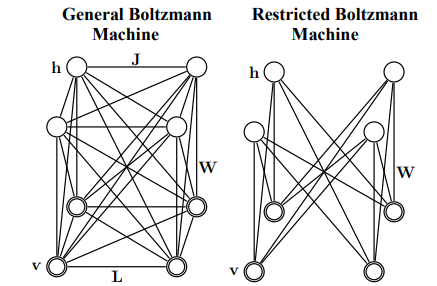
Le modèle est donc donné par .

Plus d’informations :

* <http://www.lifl.fr/~grisoni/IMA5/IHM/cours3_classification.pdf>
* <http://laurent.jeanpierre1.free.fr/recherche/markov.html>
* <http://users.polytech.unice.fr/~leroux/parolehmm.pdf>

# Boltzmann Machines

(Voir l’article <http://proceedings.mlr.press/v5/salakhutdinov09a/salakhutdinov09a.pdf>)



La machine de Boltzmann est un réseau de neurones binaires stochastiques couplés symétriquement. Il est composé de neurones visibles , et de neurones cachés . La fonction énergie est définie comme suit :

Ou sont les paramètres du modèle : représentent respectivement visible à caché, visible à visible et caché à caché. La probabilité qu’assigne le modèle à un vecteur visible est :

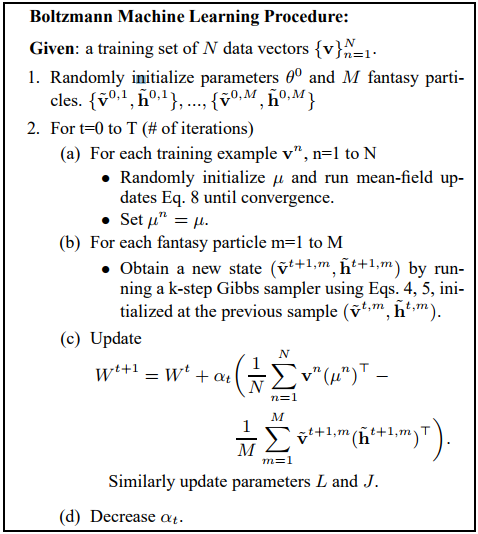
Ou dénote la probabilité non normalisée et est la fonction de partition. Les distributions conditionnelles sur les neurones cachés et visibles sont données par :

Ou est la fonction logistique

La mise à jour des paramètres qui sont nécessaires pour calculer le gradient descendant du logarithme du maximum de vraisemblance peuvent être obtenu à partir de l’équation de .

Avec le taux d’apprentissage….

Nous avons aussi



## Restricted Boltzmann Machines

En apprentissage automatique, la machine de Boltzmann restreinte est un type de réseau de neurones artificiels pour l’apprentissage non supervisé. Elle est couramment utilisée pour avoir une estimation de la distribution probabiliste d’un jeu de données.

Dans sa forme la plus simple, une machine de Boltzmann est composée d’une couche de neurones, qui reçoit l’entrée, ainsi que d’une couche de neurones cachée. Si on suppose que les neurones d’une même couche sont indépendants entre eux, on appelle cette configuration une machine de Boltzmann restreinte (RBM).

On définit une énergie d’activation pour une Machine de Boltzmann Restreinte de la manière suivante :

Avec :

* la matrice de poids entre le neurone et le neurone .
* est l’état, , du neurone .
* et sont respectivement les biais des neurones et .

La probabilité conjointe d’avoir une configuration est alors donnée par

Avec :

* la fonction d’énergie ci-dessus
* une fonction de normalisation, qui fait en sorte que la somme de toutes les probabilités fasse 1

La machine de Boltzmann s’entraîne à l’aide d’un apprentissage non supervisé. On cherche à minimise le plus possible la log-vraisemblance. La dérivée du log-vraisemblance donne l’expression suivante :

Avec :

* les variables du système (les poids ou le biais).
* l’espérance mathématiques pour les variables aléatoires et .
* une valeur du jeu de données.
* l’énergie définie ci-dessus.

On remarque la présence de deux termes dans cette expression, appelés phase positive et phase négative. La phase positive se calcule aisément pour le biais et pour la matrice des poids.

On obtient alors :

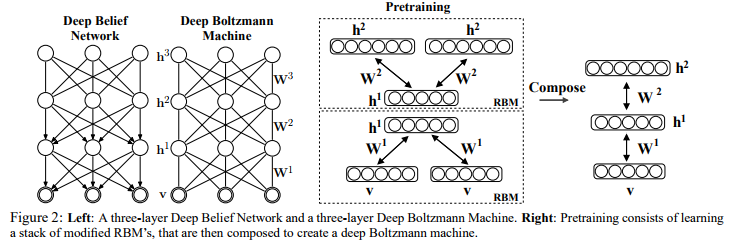
Avec l’état de la couche cachée sachant donnée par la formule

La partie la plus compliquée est de calculer ce que l’on appelle la phase négative. On ne peut pas la calculer directement car on ne connait pas la fonction de normalisation du système. Pour pouvoir effectuer une descente de gradient, on va alors calculer ce que l’on appelle la reconstruction de l’entrée . En effet, les propriétés de symétrie du système permettent de calculer l’entrée estimée par le modèle, il suffit d’appliquer la formule :

Avec le biais de la couche de neurones .

## Deep Boltzmann Machines

En général, il ne sera pas intéressant de faire apprendre une compliquée et totalement connectée machine de Boltzmann. Cependant, il sera intéressant d’utiliser une machine de Boltzmann profonde avec plusieurs couches comme nous pouvons le voir sur cette figure :



Dans laquelle, chaque couche capture de complexes corrélations entre l’activité des couches cachées.

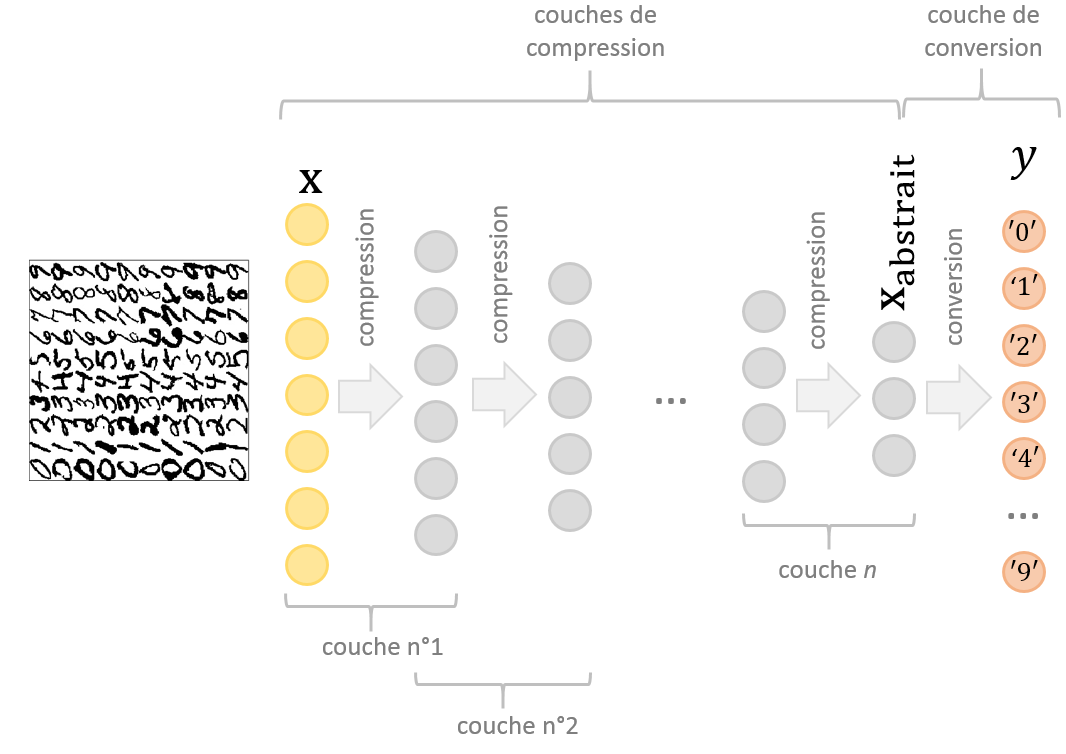
Ce type de machine est intéressante à plusieurs titres, tout d’abord comme les RBN, RBM sont capables d’apprendre des représentations internes complexes, ce qui semble utile pour la reconnaissance vocale et celle des objets. Ensuite, un haut niveau de représentation peut être construit avec des données non labellisées et un nombre peu important de données pour adapter un modèle à des tâches bien précises. Enfin, comme pour les RBN, la procédure d’inférence approximative en complément à un initial bottom-up ainsi qu’avec un dop-down feedback permettent de créer un modèle plus robuste au données ambiguës.

Plus d’informations :

* <http://proceedings.mlr.press/v5/salakhutdinov09a/salakhutdinov09a.pdf>

# Deep Belief Networks

Les Deep Belief Networks (DBN) sont des modèles graphiques qui apprennent à extraire une représentation profonde et hiérarchique des données. L’architecture de ce réseau de neurones contient deux parties comme on peut le voir sur la figure ci-dessous.



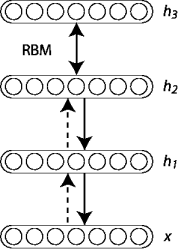
La première partie du DBN est constituée d’un ensemble de couches de compression qui convertissent les données entrées en une représentation abstraite . La seconde partie convertie cette représentation en labels de classification .

La première partie a pour objectif d’apprendre la distribution des données présentées en entrée sans tenir compte des labels . Elle est constituée d’une succession de couches dont chacune contiendra une représentation plus abstraite ou compressé que la précédente. Pour ancrer l’intuition considérons un réseau de neurones chargé de classifier des images. La première couche stockera les niveaux de gris de l’image (l’équivalent de la rétine), la seconde contiendra, par exemple, un encodage des lignes ou des zones de contraste de l’image, la troisième détectera l’existence de certaines formes géométriques simples comme des cercles, la quatrième identifiera certains agencements particuliers de ces figures comme celles qui représentent un ‘8’ formé de deux cercles juxtaposés et ainsi de suite. Ainsi on automatise en quelque sorte le processus de feature engineering manuel du machine learning. Une fois entraînée, cette première section du réseau de neurones contiendra une représentation hiérarchique des données en entrée, la dernière couche encodant la représentation abstraite la plus abstraite et aussi, c’est l’idée, la plus utile. Cette première phase peut être conçue comme une initialisation efficace du DBN, on l’appelle le pré-entrainement.

Le rôle de la seconde partie du DBN est de convertir la représentation abstraite et obscure en labels utilisables par exemple dans le cadre d’un apprentissage supervisé. L’entrainement du DBN est considéré comme achevé lorsque la performance du DBN évaluée sur un ensemble de validation distinct de l’ensemble d’entrainement ne progresse plus significativement. Cette seconde étape est appelée le fine-tuning, elle est généralement beaucoup plus lente que l’initialisation.

Les Deep Belief Networks modélisent une loi de probabilité à plusieurs variables (joint distribution) entre les données observées et les couches cachées de comme suit :

Ou , est une probabilité conditionnelle pour les neurones visibles conditionnés aux neurones cachés du RBM au niveau k, et est le visible-caché distribution jointe au plus haut niveau du RBM. Ceci est illustré à la figure suivante :



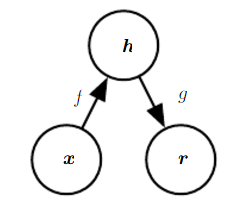
Plus d’informations ici :

* <https://www.technologies-ebusiness.com/enjeux-et-tendances/le-deep-learning-pas-a-pas>
* <http://deeplearning.net/tutorial/DBN.html>

# Auto-encoders

Un auto-encoders est un réseau de neurone qui est entrainé pour tenter de copier ses données d’entrée à celles de sortie. Intérieurement, il est composé d’une couché cachée qui décrit un code utilisé pour représenter les données d’entrée. Le réseau doit être vu comme composé de deux parties : une fonction d’encodage et une fonction de décodage qui produit une reconstruction .

Sur ce graphique, nous pouvons voir cette architecture,



Si un auto-encoder réussit simplement à apprendre comme résultat de partout, ce n’est pas très utile. Au contraire, les auto-encoders sont construits pour ne pas être capable d’apprendre à copier parfaitement. Du fait de l’apprentissage forcé pour prioriser certains aspects des données d’entrée copiés, cela apprend souvent des propriétés intéressantes sur les données.

## Sparse auto-encoders

Un Sparce auto-encoder est simplement un auto-encoder auquel on ajoute un critère d’entraînement de sparsity penalty sur la couche de code en plus de la fonction d’erreur de reconstruction

Ou est le résultat du décodeur et typiquement nous avons , résultat de la fonction d’encodage.

Ce type d’auto-encoder est utilisé pour des tâches comme la classification.

## Denoising auto-encoders (DAE)

Au lieu d’ajouter un terme de pénalité comme précédemment, nous pouvons obtenir un auto-encoder qui apprend des choses intéressantes en changeant le terme d’erreur de reconstruction dans la fonction de coût.

C’est pour cela qu’un Denoising Auto-encoder ou DAE minimise la fonction suivante

Ou est une copie de qui a été corrompu par du bruit. Cet auto-encoder doit donc annuler ce bruit plutôt que de simplement copier les données d’entrée.

L’entrainement pour enlever le bruit force f et g d’apprendre implicitement la structure de

## Convolutional auto-encoder (CAE)

Une autre stratégie pour régulariser un auto-encoder est d’utiliser une pénalité comme dans un sparse auto-encoder,

Mais avec une forme différente pour

Cela force le modèle d’apprendre une fonction qui ne change pas ou presque lorsque x change légèrement, cela permet à l’auto-encoder d’apprendre des exemples qui permettent de capturer l’information sur la distribution des données.

Plus d’informations sur les auto-encoders :

* <http://www.deeplearningbook.org/contents/autoencoders.html>

# Curriculum learning idea

Les motivations de cette méthode d’apprentissage viennent de l’observation des Humains et des animaux qui apprennent graduellement, c’est-à-dire qu’ils commencent à apprendre des choses simples et augmentent la difficulté de l’apprentissage petit à petit.

En s’inspirant de ces observations, la méthode Curriculum Learning Idea va s’appliquer à l’apprentissage d’algorithme d’apprentissage, en commençant par des éléments simples et graduellement d’augmenter le niveau de difficulté.

Plus d’informations :

* <https://ronan.collobert.com/pub/matos/2009_curriculum_icml.pdf>
* <https://gist.github.com/shagunsodhani/7e4e1c9817c46e3cb1932f62aac8806b>

# Semi Supervised Embedding

Cette méthode s’inspire de l’analyse en composantes indépendantes qui est notoirement connue en tant que méthode de séparation aveugle de source mais est aujourd’hui appliquée à divers problèmes.

Il existe de nombreux modèles dont le plus simple, le modèle linéaire instantané non bruité.

Quand l’information mutuelle est choisie comme fonction de contraste particulière, l’analyse en composantes indépendantes du vecteur aléatoire revient à identifier le modèle génératif linéaire instantané non bruité suivant :

Ou les composantes du vecteur sont mutuellement indépendantes et la matrice est de taille .

Néanmoins, les conditions d’identifiabilité suivantes doivent être vérifiées pour que l’on soit assuré de pouvoir retrouver le modèle :

* Au plus une des sources (composante de ) peut suivre une distribution normale (gaussienne)
* Le rang de la matrice doit être égal au nombre de sources (à retrouver).

Plus d’informations :

* <https://ronan.collobert.com/pub/matos/2012_deeplearning_springer.pdf>
* <http://deeplearning.net/wp-content/uploads/2013/03/pseudo_label_final.pdf>
* <https://fr.slideshare.net/alembert2000/130201-deep-learning-130131220>

# Natural Gradient

La méthode du Natural Gradient s’inspire de la divergence de Kullback-Leibler, qui est une mesure de dissimilarité entre deux distributions de probabilités et .

Pour deux distributions de probabilités discrètes et la divergence de Kullback-Leibler de par rapport à est définie par

Plus d’informations :

* <https://arxiv.org/pdf/1301.3584v7.pdf>

# Recurrent Neural Networks

Un réseau de neurones récurrents est un réseau de neurones artificiels présentant des connexions récurrentes. Un réseau de neurones récurrents est constitué d’unités (neurones) interconnectés interagissant non-linéairement et pour lequel il existe au moins un cycle dans la structure. Les unités sont reliées par des arcs (synapses) qui possèdent un poids. La sortie d’un neurone est une combinaison non linéaire de ses entrées.

Les réseaux de neurones récurrents sont adaptés pour des données d’entrée de taille variable. Ils conviennent en particulier pour l’analyse de séries temporelles. Ils sont utilisés en reconnaissance automatique de la parole ou de l’écriture manuscrite ou encore en traduction automatique.

Plus d’informations :

* <http://www.scholarpedia.org/article/Recurrent_neural_networks>

# Echo state Networks

L’Echo State Network (ESN) est un réseau de neurones récurrents avec une couche cachée peu connecté (environ 1% de connectivité). La connectivité et les poids des neurones cachés sont fixés et aléatoirement assignés. Le grand intérêt de ce réseau est que son comportement est non-linéaire, mais aussi que seuls les poids des synapses reliant les neurones de la couche cachés à ceux de sortie sont modifiés. De plus, la fonction de coût est quadratique en respect des paramètres et peut être facilement différenciée comme un système linéaire.

Plus d’informations :

* <http://www.scholarpedia.org/article/Echo_state_network>

# Stochastique Maximum Likelihood (SML)

La méthode du Stochastique Maximum Likelihood est basée sur les hypothèses suivantes :

* Nous faisons la supposition que les N échos contenus dans la densité spectrale de puissance de la série temporelle sont gaussiens. Nous avons donc,

Avec,

Ou est la puissance d’un bruit blanc Gaussien additionnel, est une variable aléatoire et , et sont les inconnus du ième écho Gaussien.

* Dénotons la série temporelle de dimension obtenu à partir de données enregistrées, satisfait

Ou est un processus Gaussien stochastique et est un bruit blanc de variance indépendant de . est donc un processus Gaussien stochastique de moyenne nulle et a pour covariance

Basée sur ces hypothèses, peut-être écrit :

Où

Ou , est la matrice identité, la pulsation de répétition du radar, correspond à la matrice conjuguée transposée et le paramètre de dimension que l’on doit estimer :

Le Maximum de Vraisemblance (Maximum Likelihood) estime de en calculant la valeur de qui maximise la valeur de la fonction de vraisemblance c’est-à-dire en sélectionnant les paramètres qui rendent le plus probable les données observées or par équivalence qui minimise la fonction du logarithme de vraisemblance négative ,

est la matrice de covariance

Quant à lui, le SML est initialisé avec des paramètres différents et il est optimisé avec une descente du gradient du second ordre

Ou et représentent respectivement le gradient et le Hessien, .

Plus d’informations :

* <http://www.cs.toronto.edu/~kswersky/wp-content/uploads/ita2010.pdf>
* <https://www.ann-geophys.net/22/3983/2004/angeo-22-3983-2004.pdf>
* <http://stats.net.au/Maths_REML_manual.pdf>

# Contrastive divergence (CD)

La méthode du Constrative divergence permet d’approximer le maximum de vraisemblance

Plus d’informations :

* <http://www.cs.toronto.edu/~fritz/absps/cdmiguel.pdf>
* <http://www.robots.ox.ac.uk/~ojw/files/NotesOnCD.pdf>
* <http://www.iro.umontreal.ca/~bengioy/papers/ftml_book.pdf>
* <http://www.gatsby.ucl.ac.uk/~turner/Notes/ContrastiveDivergence/CDv3.pdf>

# Independent Subspace Analysis

Plus d’informations :

* <http://www.nld.ds.mpg.de/downloads/paper/ICA2007_paper.pdf>
* <https://www.cs.helsinki.fi/u/ahyvarin/papers/ESANN06.pdf>

# Manifold Tangent Classifier

Cette méthode que l’on peut retrouver dans l’article cité ci-dessus est l’agrégation de différentes hypothèses de l’apprentissage des algorithmes d’apprentissage :

* L’apprentissage semi-supervisé
* Non supervisé manifold (des données réelles peuvent être regrouper dans des sous-manifold de dimensions inférieures)
* Hypothèse de classification avec un Manifold (classer des données dans des sous-manifold séparés par des régions de faible densité.

Plus d’informations :

* <https://papers.nips.cc/paper/4409-the-manifold-tangent-classifier.pdf>