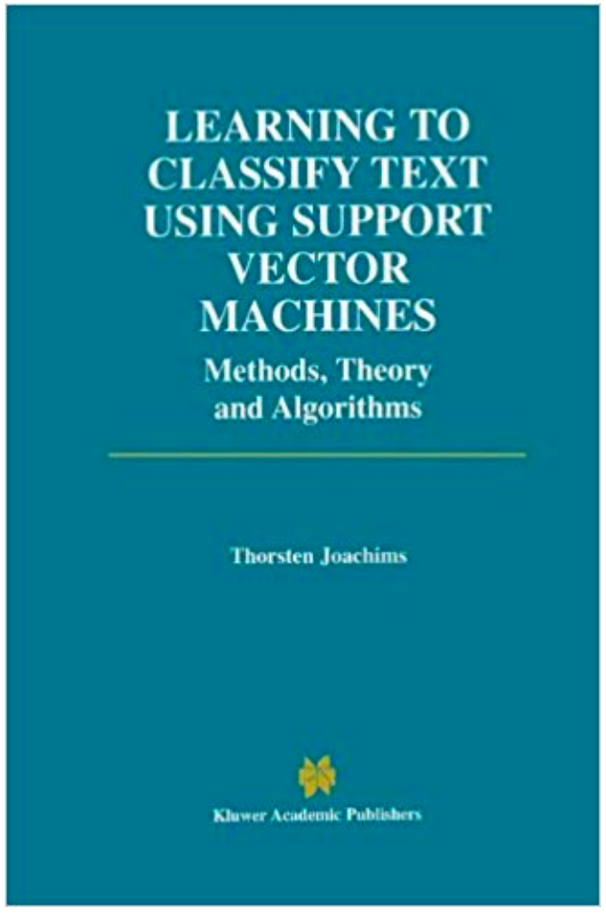
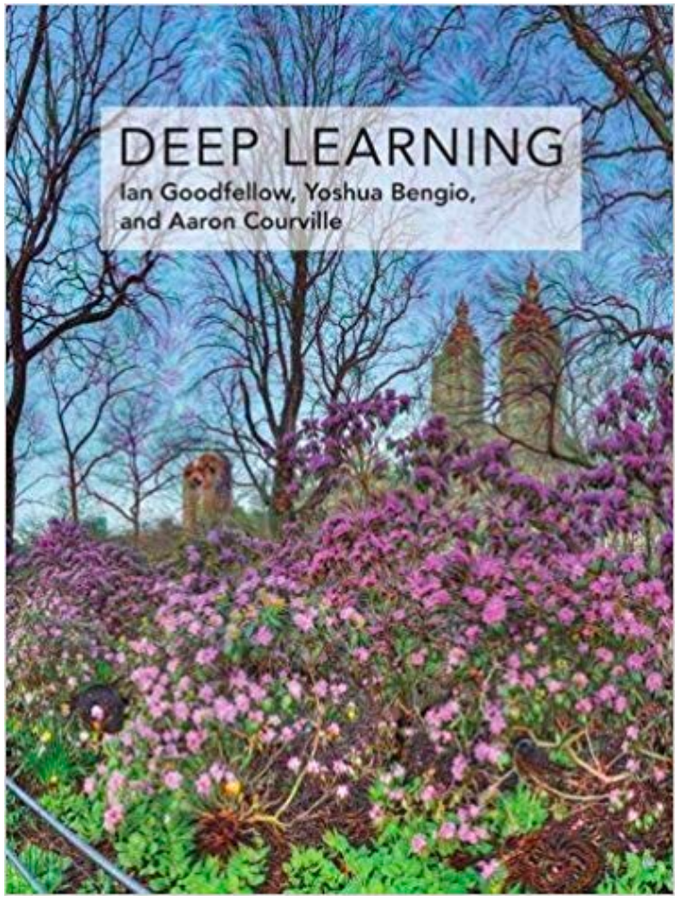
MÉTHODES DE CLASSIFICATION DE TEXTE





Emanuela Boros
boros@teklia.com

Christopher Kermorvant
kermorvant@teklia.com

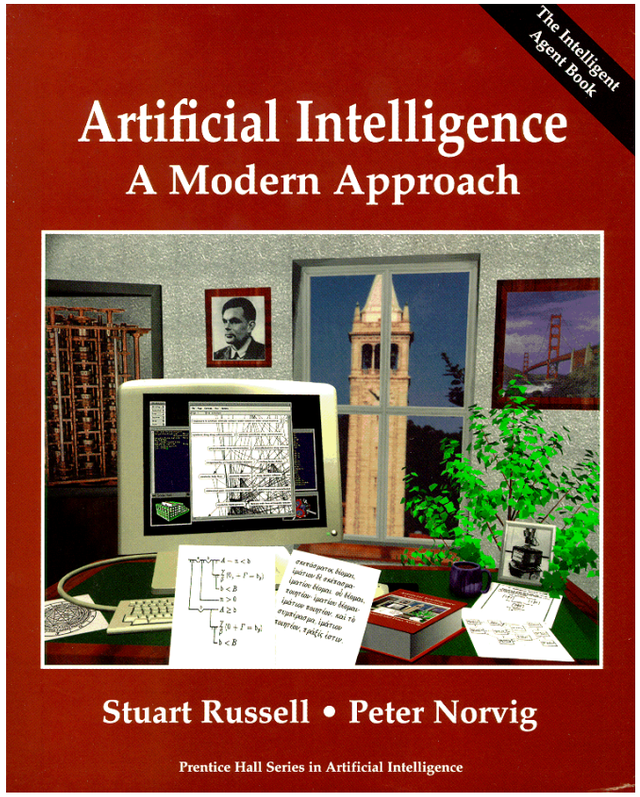
Spam filtering

- Première application industrielle à grande échelle du *machine learning*
- Problème de classification
- Succès du modèle Naïve Bayes

CentraleSupélec

Peter Norvig

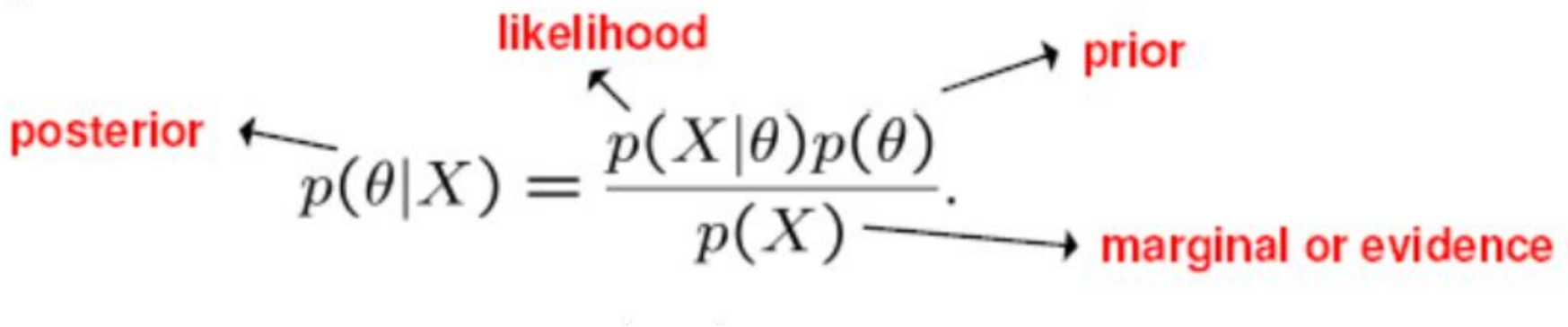
Directeur de la recherche chez Google



”At one point, The Naive Bayes was the most widely used learner there”

Approche Bayesienne

Basée sur le calcul des probabilités :



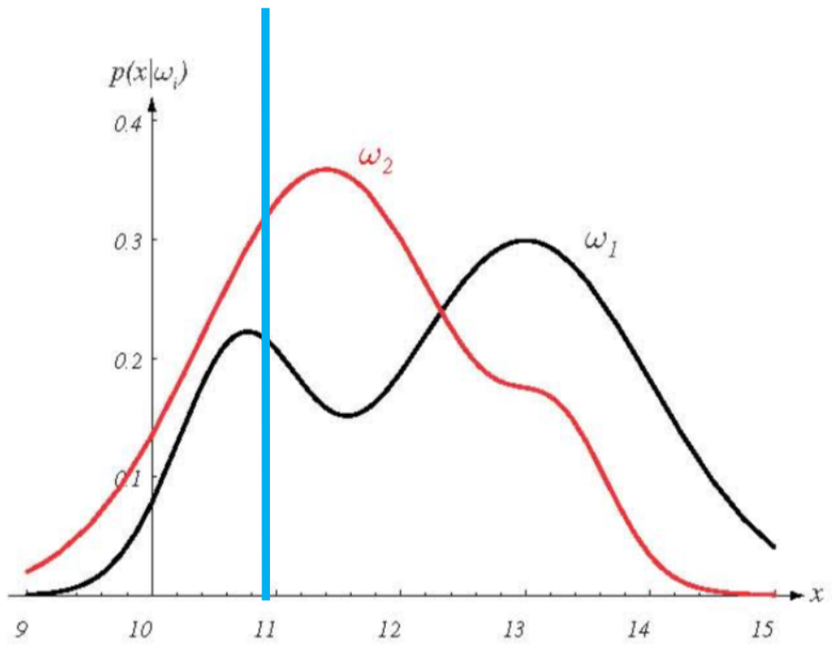
Probabilités d'une hypothèse : mise à jour d'une probabilité *a priori* à partir des données observées

Approche Bayesienne

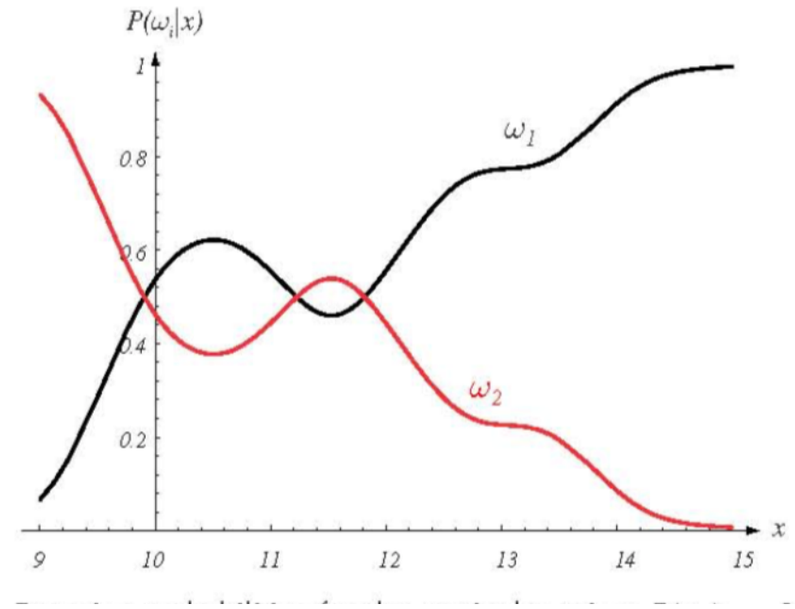
Posterior :

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)} = \frac{\texttt{likelihood x prior}}{\texttt{evidence}}$$

$$P(\theta = \theta_1) = 2/3 \qquad P(\theta = \theta_2) = 1/3$$



Likelihood

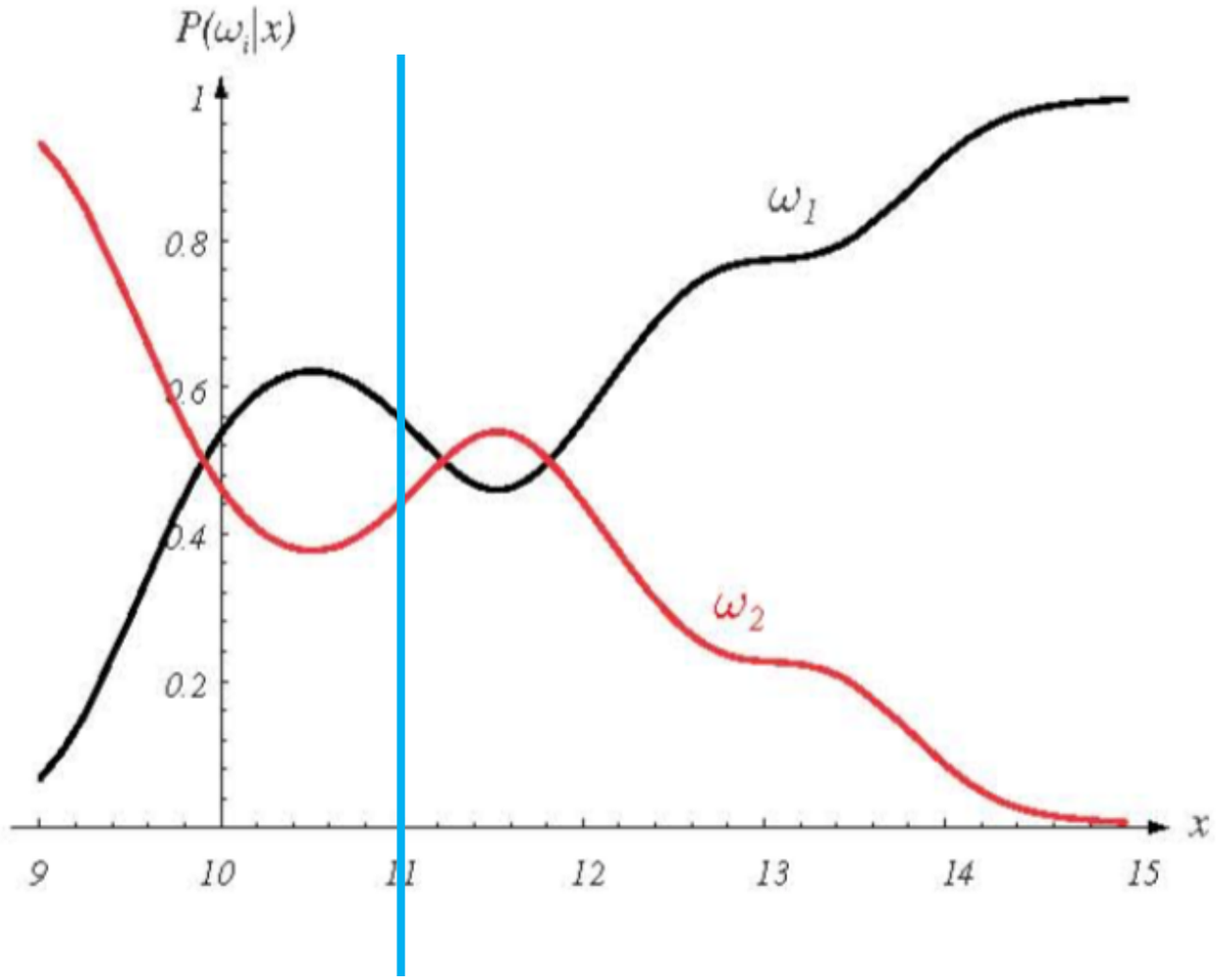


Posterior

Approche Bayesienne

Règle de décision Bayesienne :

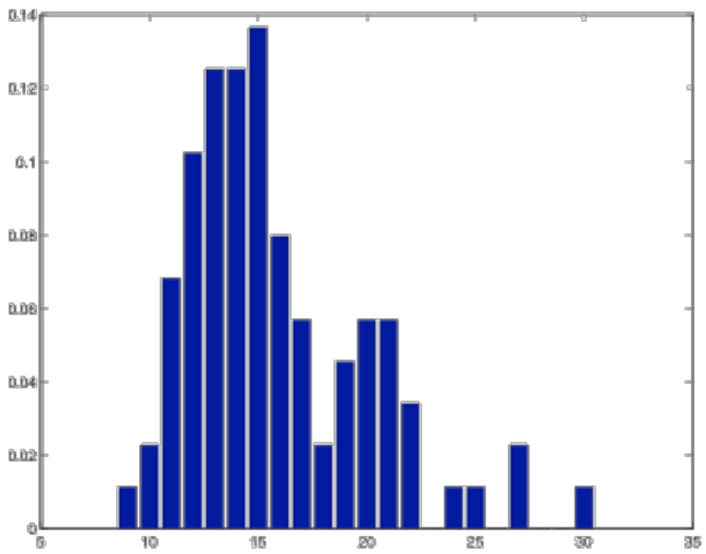
Si $P(\omega_1 \mid X) > P(\omega_2 \mid X)$ décider ω_1 sinon ω_2



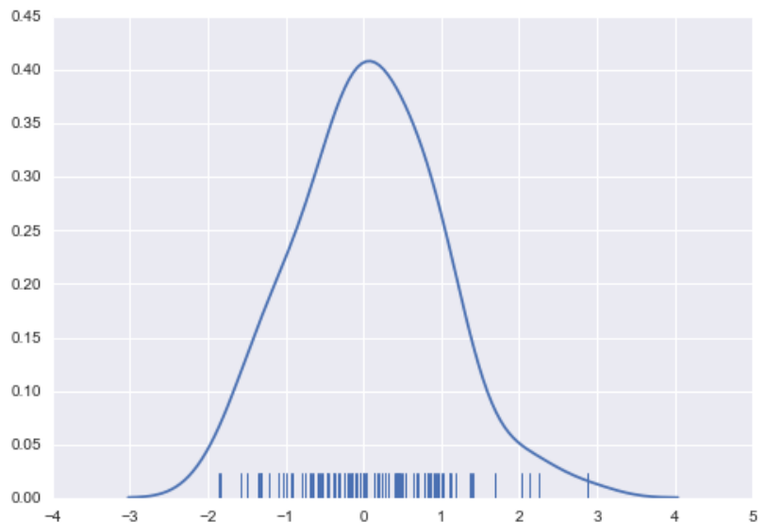
MODÈLES PROBABILISTES

Modélisation des évènements, mesures, attributs, classes avec des densités de probabilités

Evénement, mesure, attribut, classe = variable aléatoire



v.a. discrète : couleur, présence d'un mot, classe

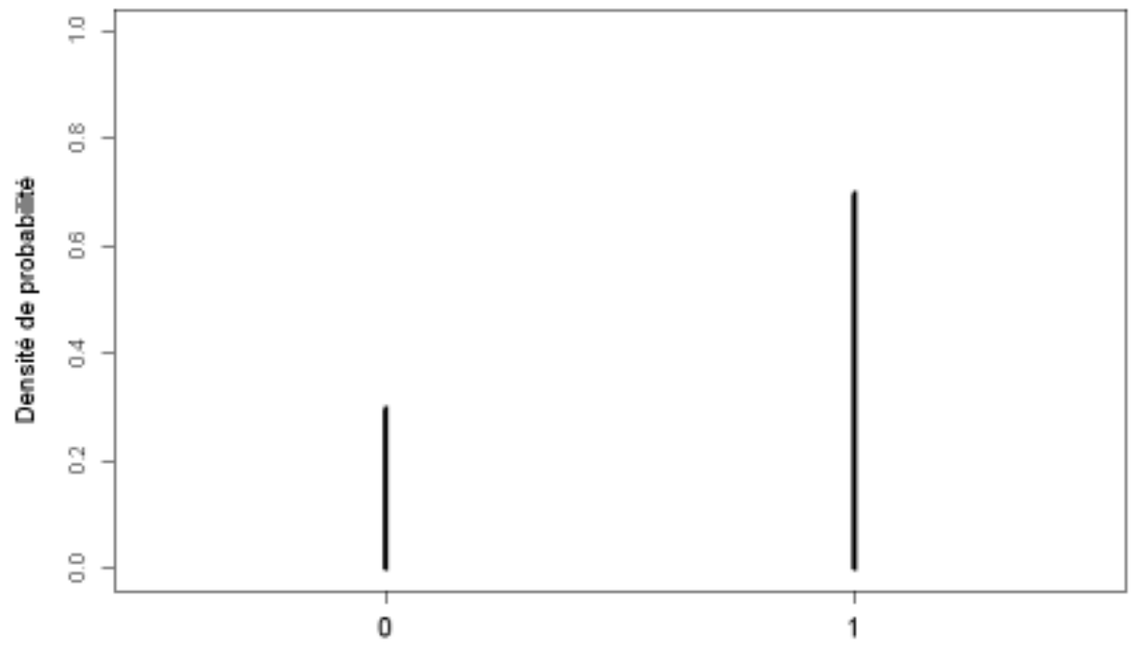


v.a. continue : densité de pixel, taille

Distribution de Bernouilli

Distribution des événements discrets à valeur dans (0,1)

$$\mathbb{P}(X = x) = \begin{cases} p & \text{si } x = 1, \\ 1 - p & \text{si } x = 0, \\ 0 & \text{sinon.} \end{cases}$$

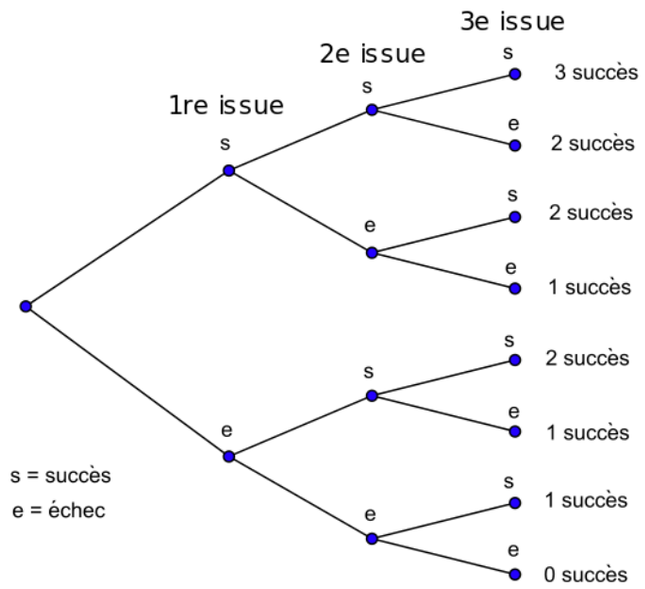


Distribution Binomiale

- Distribution du nombre de succès obtenus lors de la répétition indépendante de plusieurs expériences binomiales identiques
- Distribution d'une somme de v.a de Bernouilli

$$\mathbb{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$



Distribution Multinomiale

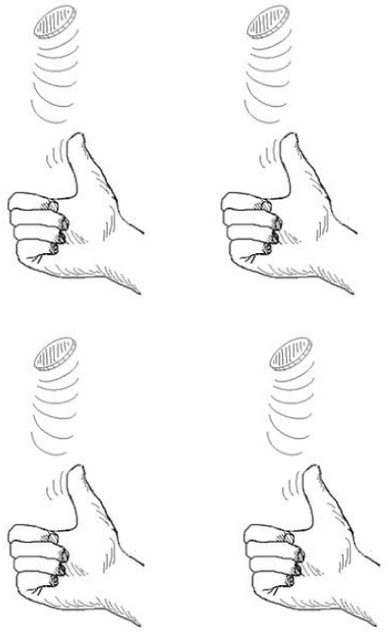
Généralisation de la loi Binomiale pour n répétitions d'une expérience à m résultats possibles

$$\mathbb{P}(N_1 = n_1, \dots N_m = n_m) = \frac{n!}{n_1! \dots n_m!} p_1^{n_1} \dots p_m^{n_m}$$

$$\sum_{i=1}^{m} N_i = n \quad \sum_{i=1}^{m} p_i = 1$$

Distributions discrètes : résumé

Bernouilli



Binomiale

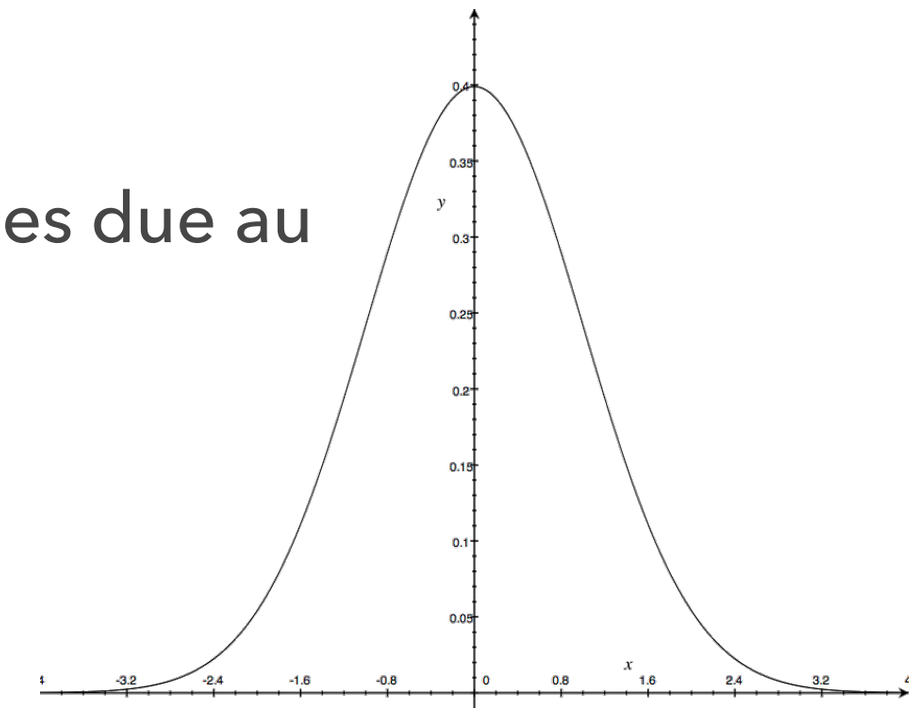
Multinomiale

Distributions continues : Gaussienne

- Modélise les phénomènes naturels issus de plusieurs événements aléatoires.

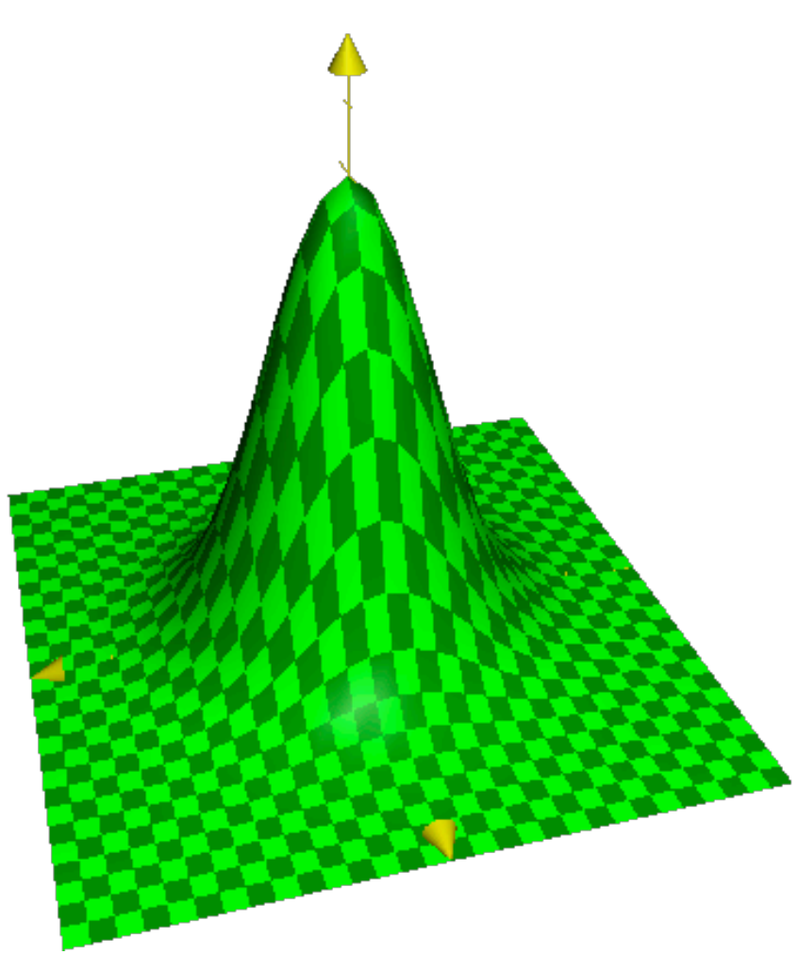
$$p(x|\mu,\sigma^2) = N(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp - \frac{(x-\mu)^2}{2\sigma^2}$$

- bonne représentation des données due au théorème centrale limite
- distribution la plus utilisée



Gaussienne en dimension 2 :

$$p(x|\mu, \Sigma^2) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} exp\Big(- \frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\Big)$$



Estimation : maximum de vraisemblance

Distribution gaussienne : $\frac{1}{\sqrt{2\pi\sigma^2}} exp - \frac{(x-\mu)^2}{2\sigma^2}$

Données : $(x_1, \cdots, x_n)$

Estimation par maximum de vraisemblance :

$$\hat{\mu} = \frac{1}{N} \sum_i x_i$$

$$\hat{\sigma}^2 = \frac{1}{N} \sum_i (x_i - \mu)^2$$

Probabilité jointe

Probabilité jointe : occurrence simultanée de deux événements P(X,Y)

Pour un problème de classification :

X est une v.a. représentant un attribut observable

Y est une v.a. représentant la classe

Probabilité jointe : P(X,Y)

Probabilité jointe : occurrence simultanée d'un mot dans un document et de la classe de ce document

	mot "adresse" présent	mot "adresse" absent
changement d'adresse	0,60	0,02
réclamation	0,03	0,35

P("adresse" présent, changement d'adresse) = 0.6

Probabilité marginale : P(X) et P(Y)

$$\sum_Y P(X,Y) = P(X)$$

$$\int_{y=-\infty}^{\infty} P(x,y)dy = P(x)$$

	mot “adresse” présent	mot “adresse” absent
changement d’adresse	0,60	0,02
réclamation	0,03	0,35

P(“adresse” présent) = 0.63

Probabilité conditionnelle

RÈGLE DE BAYES

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Thomas Bayes (1702-1761)

- P(Y|X) : probabilité de la classe en ayant observé la valeur de l'attribut X (*posterior*)
- P(Y) : probabilité de la classe avant toute observation (*prior*)

Probabilité conditionnelle

$$P(X,Y) = P(X|Y)P(Y) = P(Y|X)P(X)$$

$$P(Y|X) = \frac{P(X|Y)P(Y)}{\sum_Y P(X,Y)}$$

$$P(Y|X) = \frac{P(X,Y)}{\sum_Y P(X,Y)}$$

	mot “adresse” présent	mot “adresse” absent
changement d’adresse	0,60	0,02
réclamation	0,03	0,35

P(changement d’adresse| “adresse” présent) = 0.6/0.63

Naive Bayes

Transformation de la règle de Bayes en classifieur

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

- P(Y|X) : probabilité que X soit de la classe Y (posterior)
- P(X|Y) : probabilité du mot X dans la classe Y
- P(Y) : probabilité a priori de la classe
- P(X) : normalisation des probabilités des mots

La classe prédite est $\text{argmax}_i\ P(Y_i|X)$

Naive Bayes

Lorsque X est multidimensionnel (toujours !), NB devient :

$$P(Y|X) = \frac{P(X_1,\cdots,X_n|Y)P(Y)}{P(X)}$$

Il faut estimer $P(X_1,...,X_n|Y)$: jamais assez de données.

Hypothèse d'indépendance :

$$\begin{aligned} P(Y|X) &= \frac{P(X_1,\cdots,X_n|Y)P(Y)}{P(X)} \\ &= \frac{\prod_i P(X_i|X_1,\cdots,X_{i-1},Y)P(Y)}{P(X)} \\ &= \frac{\prod_i P(X_i|Y)P(Y)}{P(X)} \end{aligned}$$

Règle de la chaine

Hypothèse d'indépendance

Naive Bayes

Pour calculer les probabilités à posteriori P(Y|X), il faut :

- Calculer les probabilités à priori P(Y)
- Calculer les P(X|Y) :

	Mot « adresse » présent
Changement d'adresse	#nb présent / # changt adresse
réclamation	#nb présent / # réclamation

Note : le calcul de P(X) n'est pas nécessaire

Naive Bayes

Limites du Naive Bayes :

- Hypothèse d'indépendance : en classification de texte, cela signifie que les mots d'un document sont indépendants les uns des autres !

- Problème de l'estimation des événements rares, par exemple les mots apparaissant zéro ou une fois (voir Zipf law)

- Modèle génératif : l'objectif est de maximiser la vraisemblance des données, pas de minimiser l'erreur de classification

Régression logistique

Objectif : maximiser la classification à partir des probabilités conditionnelles (pas jointes comme Naive Bayes)

$$\log \frac{P(Y=1|X)}{P(Y=0|X} = \overrightarrow{w}^T \phi(x)$$

$\phi(x)$: représentation vectorielle des mots

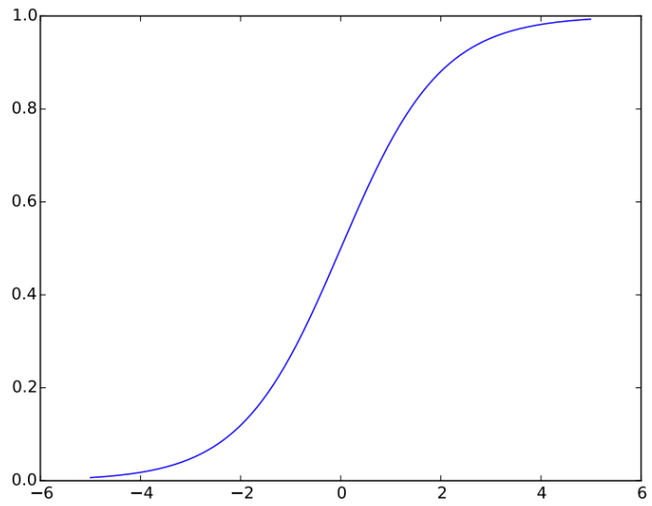
$\overrightarrow{w}$: vecteurs de poids appris

Régression logistique

$$P(Y = 1|X) = \frac{1}{1 + exp(-\overrightarrow{w}^T \phi(x))}$$

$$\frac{1}{1 + e^{-t}}$$

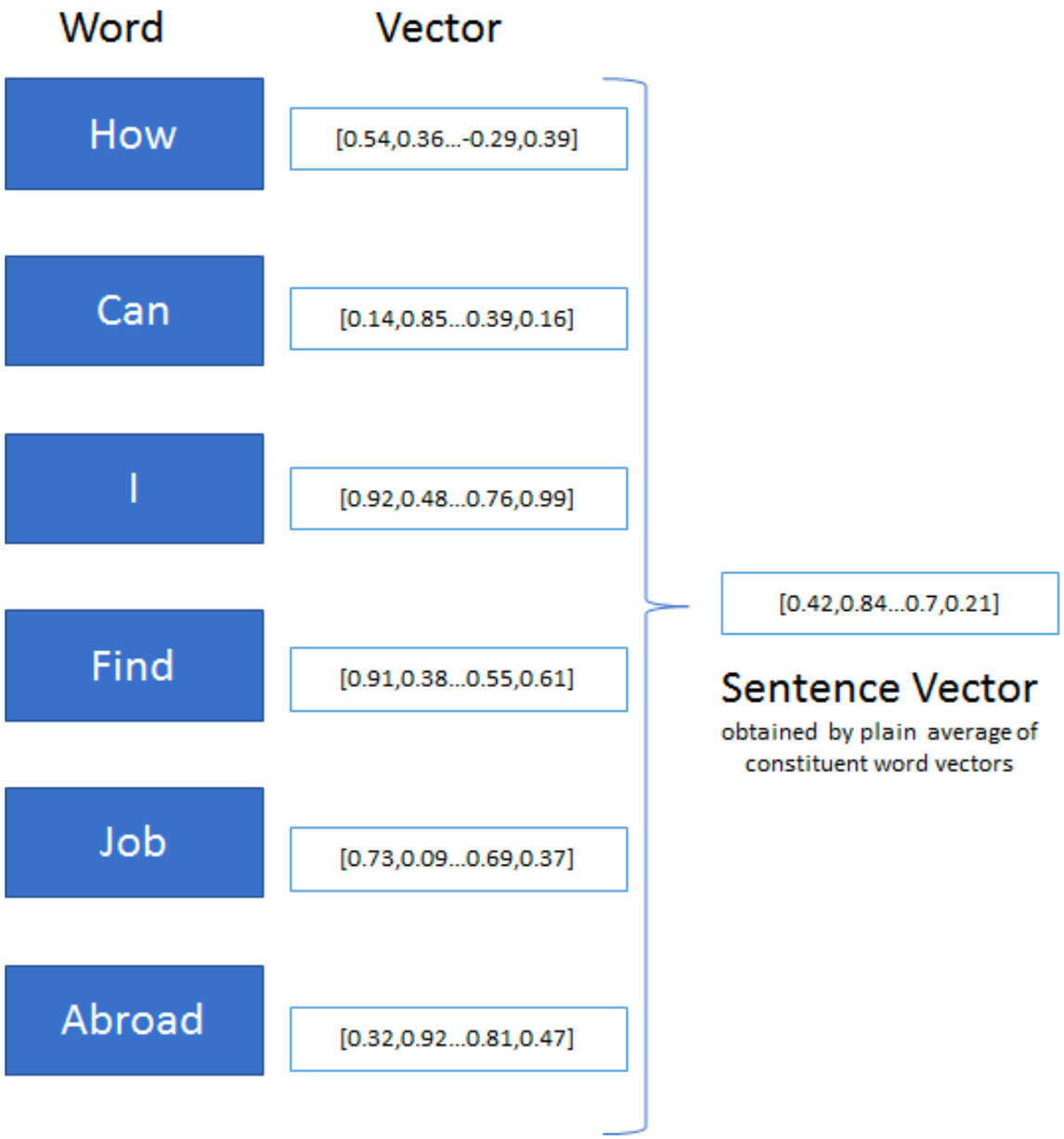
Fonction logistique



- Combinaison d'une régression et d'une fonction logistique
- Optimisation par maximum de vraisemblance
- Pas de solution analytique : optimisation par descente de gradient

Embedding de documents

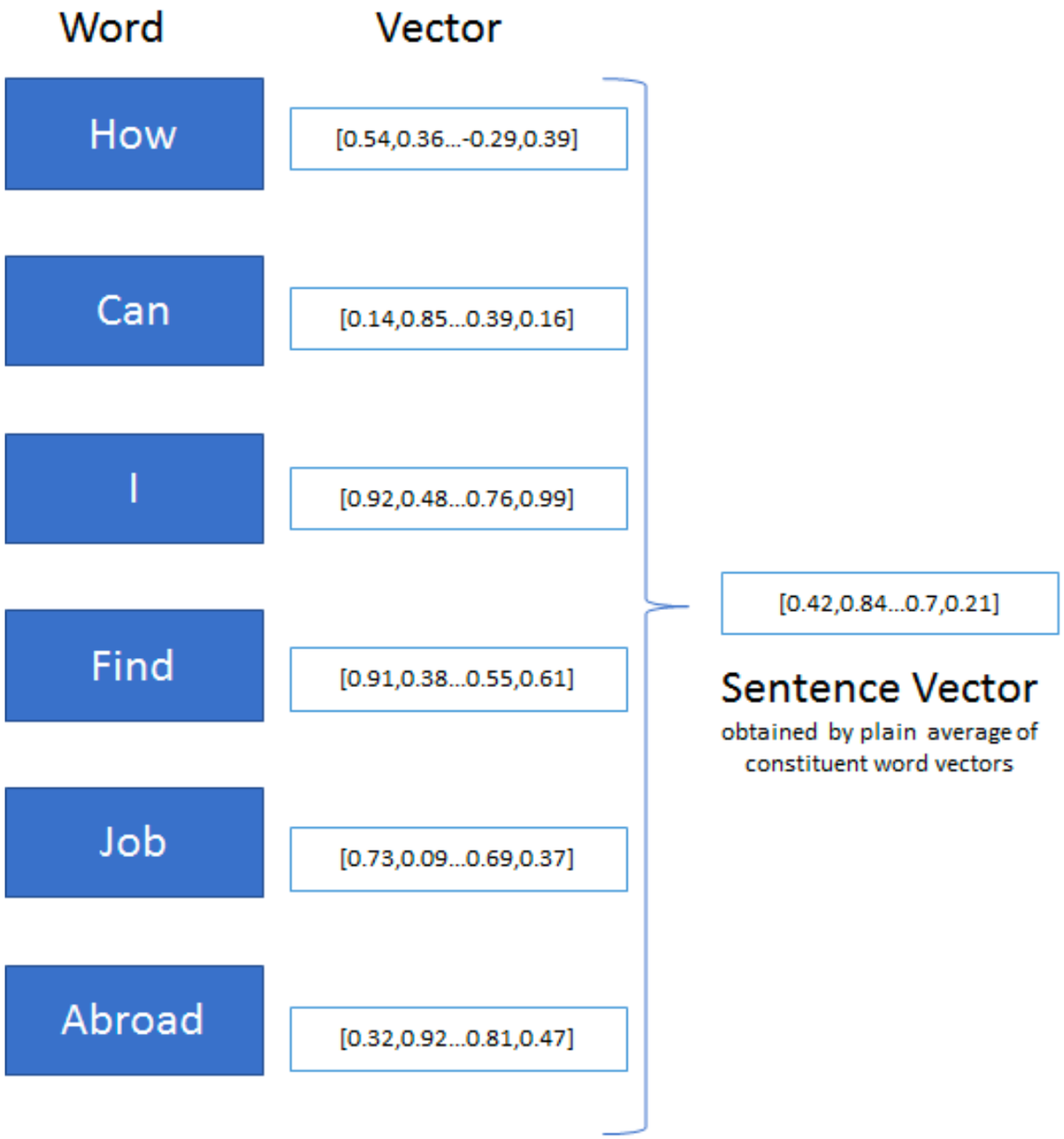
Embedding de phrase



Simple moyenne des vecteurs de mots la phrase ou du document

Embedding de documents

Embedding de phrase



Simple moyenne des vecteurs de mots la phrase ou du document

Embedding de documents

Distributed Representations of Sentences and Documents

Quoc Le QVL@GOOGLE.COM
Tomas Mikolov TMIKOLOV@GOOGLE.COM
Google Inc, 1600 Amphitheatre Parkway, Mountain View, CA 94043

Abstract

Many machine learning algorithms require the input to be represented as a fixed-length feature vector. When it comes to texts, one of the most common fixed-length features is bag-of-words. Despite their popularity, bag-of-words features have two major weaknesses: they lose the ordering of the words and they also ignore semantics of the words. For example, "powerful," "strong" and "Paris" are equally distant. In this paper, we propose *Paragraph Vector*, an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents. Our algorithm represents each document by a dense vector which is trained to predict words in the document. Its construction gives our algorithm the potential to overcome the weaknesses of bag-of-words models. Empirical results show that Paragraph Vectors outperform bag-of-words models as well as other techniques for text representations. Finally, we achieve new state-of-the-art results on several text classification and sentiment analysis tasks.

1. Introduction

Text classification and clustering play an important role in many applications, e.g, document retrieval, web search, spam filtering. At the heart of these applications is machine learning algorithms such as logistic regression or K-means. These algorithms typically require the text input to be represented as a fixed-length vector. Perhaps the most common fixed-length vector representation for texts is the bag-of-words or bag-of-n-grams (Harris, 1954) due to its simplicity, efficiency and often surprising accuracy.

However, the bag-of-words (BOW) has many disadvantages. The word order is lost, and thus different sentences can have exactly the same representation, as long as the same words are used. Even though bag-of-n-grams considers the word order in short context, it suffers from data sparsity and high dimensionality. Bag-of-words and bag-of-n-grams have very little sense about the semantics of the words or more formally the distances between the words. This means that words "powerful," "strong" and "Paris" are equally distant despite the fact that semantically, "powerful" should be closer to "strong" than "Paris."

In this paper, we propose *Paragraph Vector*, an unsupervised framework that learns continuous distributed vector representations for pieces of texts. The texts can be of variable-length, ranging from sentences to documents. The name Paragraph Vector is to emphasize the fact that the method can be applied to variable-length pieces of texts, anything from a phrase or sentence to a large document.

In our model, the vector representation is trained to be useful for predicting words in a paragraph. More precisely, we concatenate the paragraph vector with several word vectors from a paragraph and predict the following word in the given context. Both word vectors and paragraph vectors are trained by the stochastic gradient descent and backpropagation (Rumelhart et al., 1986). While paragraph vectors are unique among paragraphs, the word vectors are shared. At prediction time, the paragraph vectors are inferred by fixing the word vectors and training the new paragraph vector until convergence.

Our technique is inspired by the recent work in learning vector representations of words using neural networks (Bengio et al., 2006; Collobert & Weston, 2008; Mnih & Hinton, 2008; Turian et al., 2010; Mikolov et al., 2013a;c). In their formulation, each word is represented by a vector which is concatenated or averaged with other word vectors in a context, and the resulting vector is used to predict other words in the context. For example, the neural network language model proposed in (Bengio et al., 2006) uses the concatenation of several previous word vectors to form the input of a neural network, and tries to predict the next word. The outcome is that after the model is trained, the word vectors are mapped into a vector space such that

Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

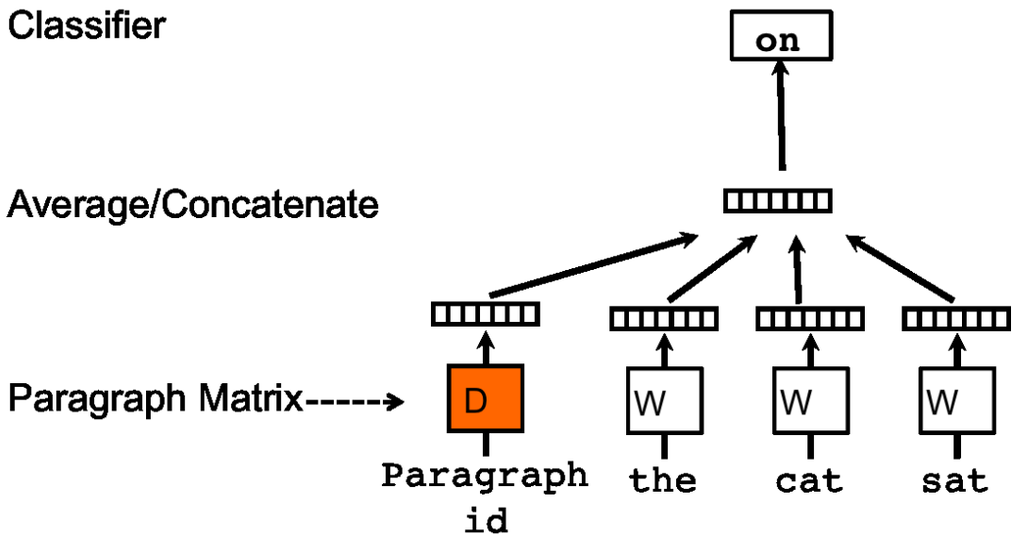
Distributed Representations of Sentences and Documents

Quoc Le & Tomas Mikolov

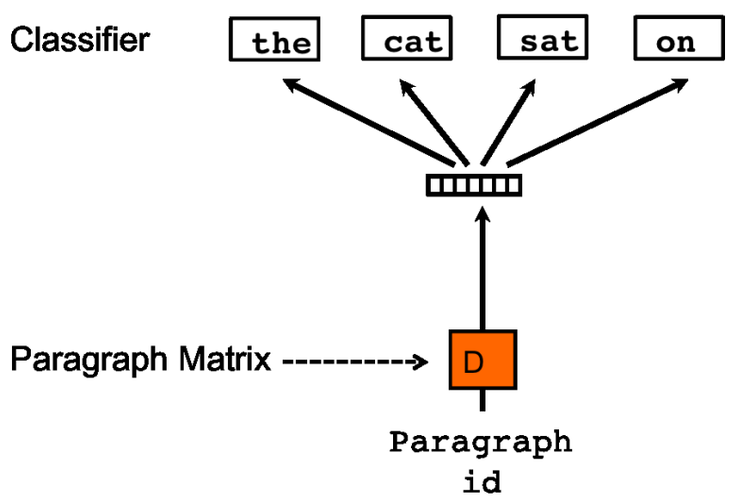
ICML 2014

Embedding de documents

doc2vec



Distributed Memory version of Paragraph Vector (PV-DM)



Distributed Bag of Words version of Paragraph Vector (PV-DBOW)

- Ajout d'une représentation du paragraphe (topic)
- Nécessité d'apprendre cette représentation sur les nouveaux documents (*test time*)

Classification avec embedding + CNN

Convolutional Neural Networks for Sentence Classification

Yoon Kim
New York University
yhk255@nyu.edu

Abstract

We report on a series of experiments with convolutional neural networks (CNN) trained on top of pre-trained word vectors for sentence-level classification tasks. We show that a simple CNN with little hyperparameter tuning and static vectors achieves excellent results on multiple benchmarks. Learning task-specific vectors through fine-tuning offers further gains in performance. We additionally propose a simple modification to the architecture to allow for the use of both task-specific and static vectors. The CNN models discussed herein improve upon the state of the art on 4 out of 7 tasks, which include sentiment analysis and question classification.

1 Introduction

Deep learning models have achieved remarkable results in computer vision (Krizhevsky et al., 2012) and speech recognition (Graves et al., 2013) in recent years. Within natural language processing, much of the work with deep learning methods has involved learning word vector representations through neural language models (Bengio et al., 2003; Yih et al., 2011; Mikolov et al., 2013) and performing composition over the learned word vectors for classification (Collobert et al., 2011). Word vectors, wherein words are projected from a sparse, 1-of-V encoding (here V is the vocabulary size) onto a lower dimensional vector space via a hidden layer, are essentially feature extractors that encode semantic features of words in their dimensions. In such dense representations, semantically close words are likewise close—in euclidean or cosine distance—in the lower dimensional vector space.

Convolutional neural networks (CNN) utilize layers with convolving filters that are applied to local features (LeCun et al., 1998). Originally invented for computer vision, CNN models have subsequently been shown to be effective for NLP and have achieved excellent results in semantic parsing (Yih et al., 2014), search query retrieval (Shen et al., 2014), sentence modeling (Kalchbrenner et al., 2014), and other traditional NLP tasks (Collobert et al., 2011).

In the present work, we train a simple CNN with one layer of convolution on top of word vectors obtained from an unsupervised neural language model. These vectors were trained by Mikolov et al. (2013) on 100 billion words of Google News, and are publicly available.[1] We initially keep the word vectors static and learn only the other parameters of the model. Despite little tuning of hyperparameters, this simple model achieves excellent results on multiple benchmarks, suggesting that the pre-trained vectors are 'universal' feature extractors that can be utilized for various classification tasks. Learning task-specific vectors through fine-tuning results in further improvements. We finally describe a simple modification to the architecture to allow for the use of both pre-trained and task-specific vectors by having multiple channels.

Our work is philosophically similar to Razavian et al. (2014) which showed that for image classification, feature extractors obtained from a pre-trained deep learning model perform well on a variety of tasks—including tasks that are very different from the original task for which the feature extractors were trained.

2 Model

The model architecture, shown in figure 1, is a slight variant of the CNN architecture of Collobert et al. (2011). Let $\mathbf{x}_i \in \mathbb{R}^k$ be the k-dimensional word vector corresponding to the i-th word in the sentence. A sentence of length n (padded where

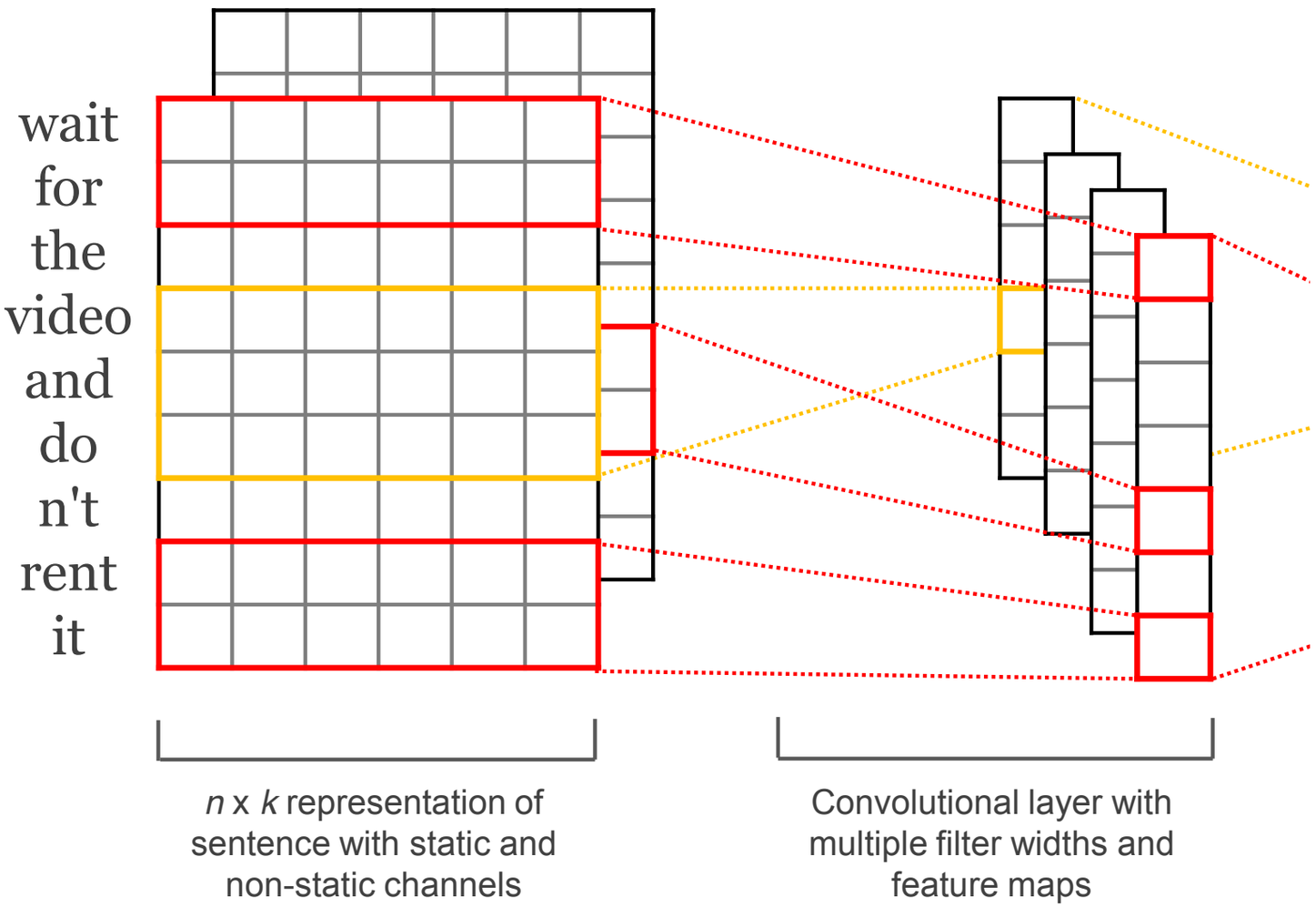
[1] https://code.google.com/p/word2vec/

Convolutional Neural Networks for Sentence Classification

Yoon Kim

EMNLP 2014

Classification avec embedding + CNN



Prise en compte de contextes de taille variable par des filtres de convolution de différentes tailles

Réseaux à convolution

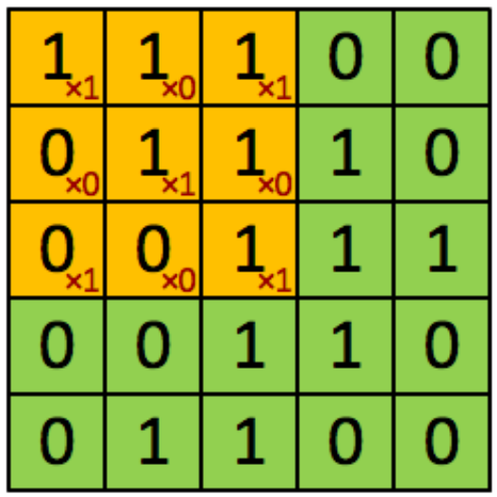
- Les réseaux à convolutions sont adaptés aux données disposées sur une grille 1D (séquence) ou 2D (image).
- La convolution (2D) calcule la valeur d'une *feature* par le produit d'un filtre de convolution (*kernel*) et du signal d'entrée (*input*)

RÉSEAUX À CONVOLUTION

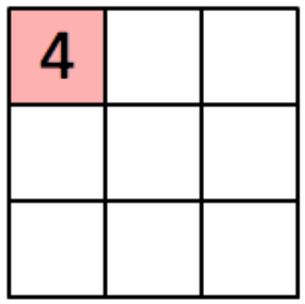
- Calcul d'une convolution : produit terme à terme du filtre de convolution par les valeurs de l'input (image ou sortie d'une convolution précédente)
- Pour garder la taille d'entrée identique, on peut ajouter une bordure (*padding*)

1	0	1
0	1	0
1	0	1

Filtre de convolution



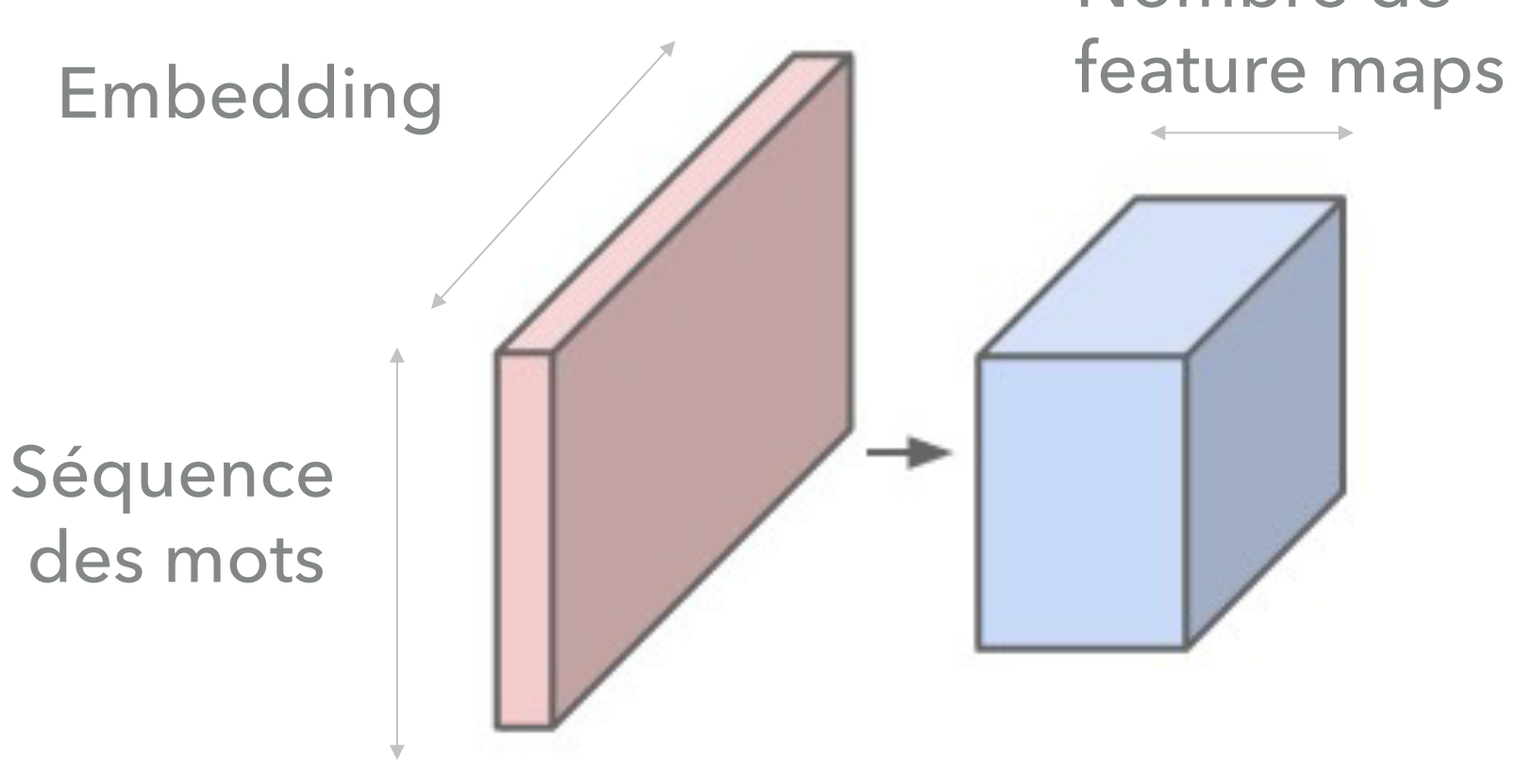
Image



Features

RÉSEAUX À CONVOLUTION

- Les features sont organisées par couche (features maps)
- Le nombre de feature maps est un paramètre du réseau



RÉSEAUX À CONVOLUTION

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	2	2	2	1	0
0	2	1	2	2	0	0
0	2	1	0	1	1	0
0	0	1	0	2	1	0
0	2	0	1	2	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	1	0	2	1	2	0
0	2	0	0	2	2	0
0	1	2	1	0	2	0
0	1	0	0	2	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	0	1	1	1	0	0
0	0	0	1	2	2	0
0	0	2	1	2	1	0
0	0	2	2	1	0	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

0	0	0
0	1	1
1	-1	1

w0[:,:,1]

1	-1	1
1	-1	0
-1	-1	0

w0[:,:,2]

1	0	-1
0	-1	1
-1	0	-1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

0	-1	1
0	0	0
1	-1	0

w1[:,:,1]

-1	0	0
0	1	0
-1	0	0

w1[:,:,2]

1	1	-1
0	0	1
1	1	0

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

-3	3	-2
-2	5	-5
1	5	4

o[:,:,1]

0	5	3
-1	8	3
2	5	1

toggle movement

RÉSEAUX À CONVOLUTION

L’entrée étant en 3 dimensions (plusieurs features maps en sortie de la couche précédente), les valeurs des convolutions sont sommées par couche d’entrée pour obtenir la valeur sur la feature map de sortie correspondante

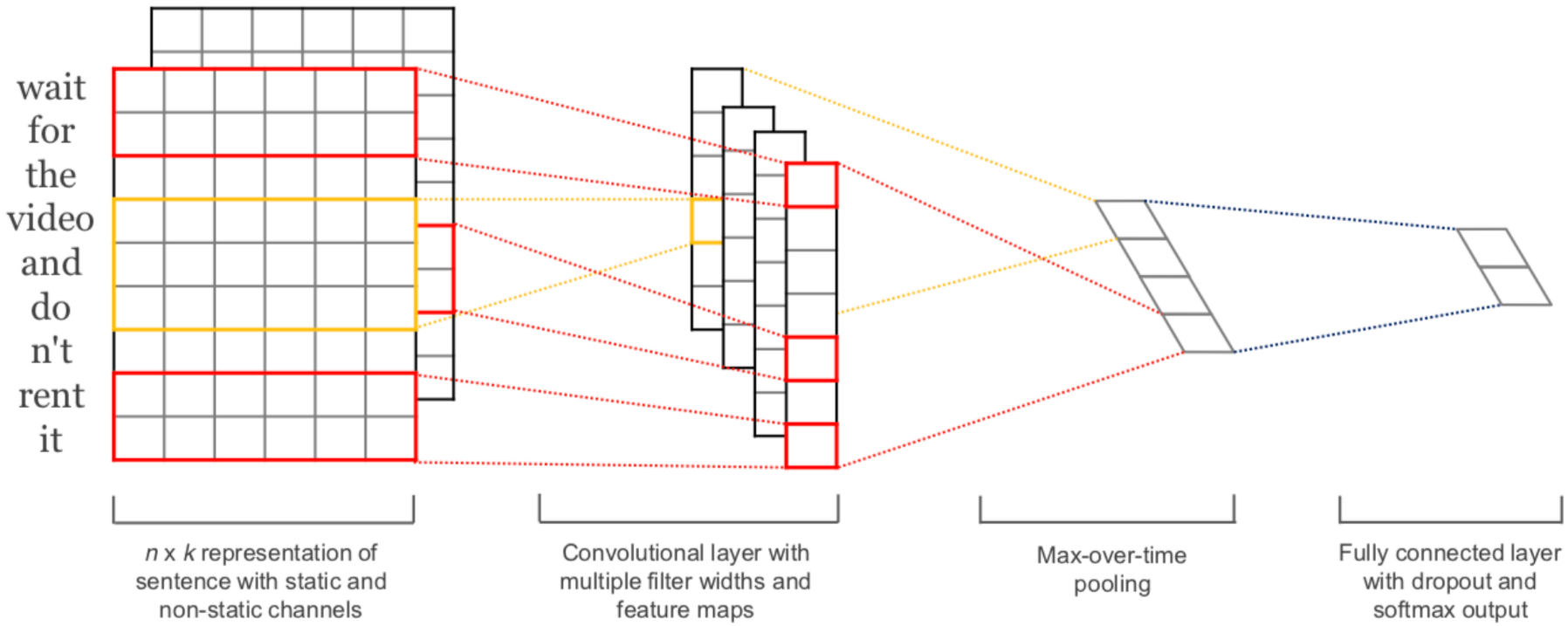
Le bias est ajouté à la somme des valeurs des convolutions des couches d’entrée

Dans le cas du texte, la convolution se déplace uniquement dans 1 dimension (convolution 1D)

Avantages des réseaux à convolutions

- *Parsimonie* : le kernel est plus petit que l'input donc moins de paramètres qu'une couche dense et moins de calculs à effectuer
- *Partage de paramètres* : les paramètres du kernel sont identiques pour toutes les positions sur l'image (moins de paramètres à stocker)
- *Invariance en translation* : due au partage de paramètres

Classification avec embedding + CNN



- Variante de Collobert et al. (2011)
- Embedding word2vec CBOW 300
- Padding des phrases à la même taille
- Multiples filtres (100), de multiple taille (3, 4, 5)
- Adaptation ou pas des filtres