

Régression logistique

REGRESSION LOGISTIQUE BINAIRE

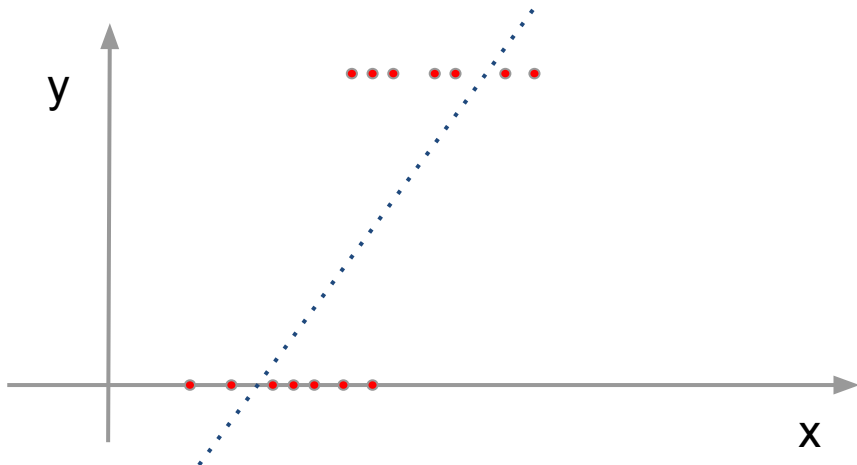
Dans la régression linéaire, on prédit une variable Y continue.

La “régression” **logistique** est en fait une technique de **classification**.

On considère ici une classification binaire :
 $Y \in \{0, 1\}$.

IDEE

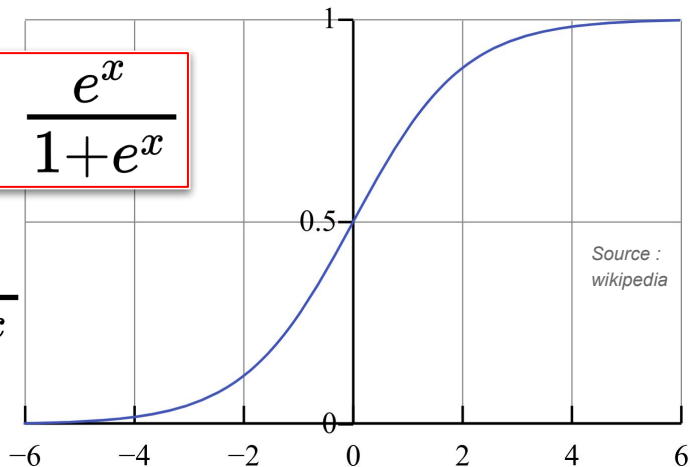
Puisque l'on a une variable à expliquer qui ne prend que les valeurs 0 et 1, un modèle linéaire ne semble pas adapté.



LA FONCTION LOGISTIQUE STANDARD

$$f(x) = \frac{e^x}{1+e^x}$$

$$= \frac{1}{1+e^{-x}}$$



La régression logistique revient à appliquer cette fonction à une régression linéaire simple; "poussant" les valeurs vers 0 et 1.

PRINCIPE

Comme d'habitude, X est l'input, Y l'output.

La régression logistique binaire modélise:

$$P(Y = 1|X) = \frac{e^{b_0 + b_1 x_1 + \dots + b_p x_p}}{\underbrace{1 + e^{b_0 + b_1 x_1 + \dots + b_p x_p}}_{\text{avec } \vec{X} = (x_1, x_2, x_3, \dots) \text{ toujours compris entre 0 et 1}}}$$

PRINCIPE

Comme d'habitude, X est l'input, Y l'output.

La régression logistique binaire modélise:

$$P(Y = 1|X) = \frac{e^{b_0 + b_1 x_1 + \dots + b_p x_p}}{\underbrace{1 + e^{b_0 + b_1 x_1 + \dots + b_p x_p}}_{\text{avec } \vec{X} = (x_1, x_2, x_3, \dots) \text{ toujours compris entre 0 et 1}}}$$

On réfléchit en termes de **rapport de probabilités** ou encore de **odd ratios** :

$$\ln \frac{P(Y = 1|X)}{P(Y = 0|X)} = b_0 + b_1 x_1 + \dots + b_p x_p$$

TROUVER LA SOLUTION

Il s'agit cette fois de maximiser une fonction que l'on appelle la **vraisemblance** du modèle. On parle de l'estimation du **maximum de vraisemblance**.

Intuition: "quels sont les paramètres qui ont vraisemblablement généré ces valeurs que j'observe?".

TROUVER LA SOLUTION

Il s'agit cette fois de maximiser une fonction que l'on appelle la **vraisemblance** du modèle. On parle de l'estimation du **maximum de vraisemblance**.

Intuition: “quels sont les paramètres qui ont vraisemblablement généré ces valeurs que j'observe?”.

$$\max_{\beta} \prod_{i=1}^n P(\hat{y}_i = y_i | X, \beta)$$

$$\text{avec } \vec{\beta} = (b_0, b_1, b_2, b_3, \dots)$$

EXEMPLE

.....	$X_1 =$	$(0.5, 3, 8)$	$y_1 = 1$
Données :	$X_2 =$	$(0.7, 4, 9)$	$y_2 = 1$
	$X_3 =$	$(2, 2, 1)$	$y_3 = 0$

EXEMPLE

$$\begin{array}{lll} \dots\dots\dots X_1 = & (0.5, 3, 8) & \dots\dots\dots y_1 = 1 \\ \text{Données :} & X_2 = & (0.7, 4, 9) \quad y_2 = 1 \\ & X_3 = & (2, 2, 1) \quad y_3 = 0 \end{array}$$

On cherche
le vecteur β
qui maximise
ce produit :

$$\begin{aligned} & P(y_1 = 1 \mid X_1 = (0.5, 3, 8)) \\ & \times P(y_2 = 1 \mid X_2 = (0.7, 4, 9)) \\ & \times P(y_3 = 0 \mid X_3 = (2, 2, 1)) \end{aligned}$$

$$\max_{\beta} \prod_{i=1}^n P(\hat{y}_i = y_i \mid X, \beta)$$

EXEMPLE

$$\begin{array}{lll} \text{Données :} & X_1 = (0.5, 3, 8) & y_1 = 1 \\ & X_2 = (0.7, 4, 9) & y_2 = 1 \\ & X_3 = (2, 2, 1) & y_3 = 0 \end{array}$$

On cherche le vecteur β qui maximise ce produit :

$$\begin{aligned} & P(y_1 = 1 \mid X_1 = (0.5, 3, 8)) \\ & \times P(y_2 = 1 \mid X_2 = (0.7, 4, 9)) \\ & \times P(y_3 = 0 \mid X_3 = (2, 2, 1)) \end{aligned}$$

Il suffit alors de remplacer les probabilités par leur expression logit pour avoir la fonction finale

OPTIMISATION

Trouver les poids de la régression logistique revient à **maximiser une fonction concave**, ce qui revient à minimiser son opposée, qui est **convexe**.

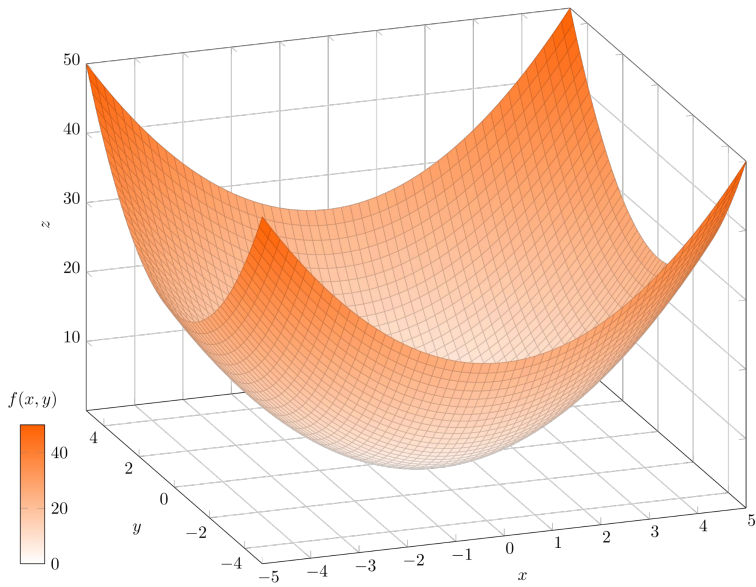
Comme plus tôt, on calcule sa dérivée et on l'annule. Cette fois, il n'existe pas de forme fermée pour ce calcul.

OPTIMISATION

Fonction convexe : un **unique** minimum.

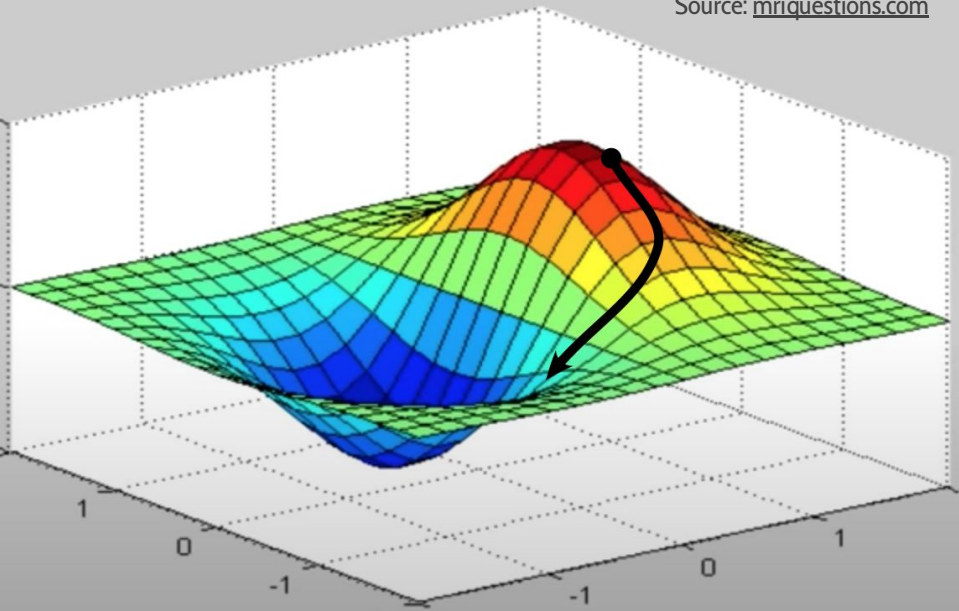
OPTIMISATION

Fonction convexe : un **unique** minimum.



DESCENTE DE GRADIENT

Source: mriquestions.com



DESCENTE DE GRADIENT

(au tableau)

Liens pour réviser:

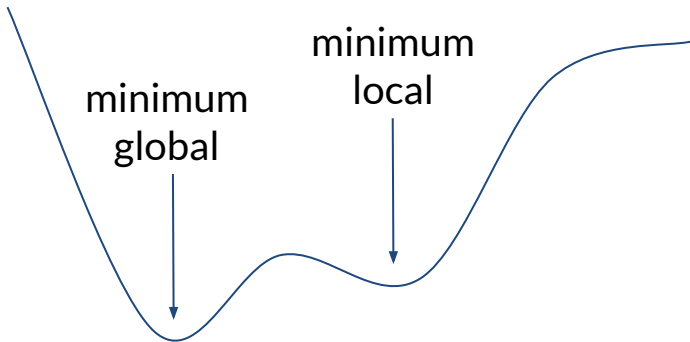
[Gradient](#),

[Algorithme du gradient](#)

(aka "[Gradient Descent](#)")

DESCENTE DE GRADIENT

Remarque: si la fonction n'est pas convexe, elle peut avoir plusieurs minimums locaux. On risque de rester bloqué dans un minimum **local**.



INTERPRETATION

La régression logistique permet d'**interpréter** l'effet de chaque variable sur ce qu'on essaie d'expliquer.

Exemple: l'étudiant va-t-il valider Fouille de Données?

$$\ln \frac{P(Valider=1 \mid travail, cadeaux, vidéos)}{P(Valider=0 \mid travail, cadeaux, vidéos)} = b_0 + b_1 travail + b_2 cadeaux + b_3 vidéos$$

INTERPRETATION

$$\ln \frac{P(Valider=1 \mid travail, cadeaux, vidéos)}{P(Valider=0 \mid travail, cadeaux, vidéos)} = b_0 + b_1 travail + b_2 cadeaux + b_3 vidéos$$

Supposons que la régression logistique donne:

$$\hat{b}_0 = 0, \hat{b}_1 = 0.5, \hat{b}_2 = 1, \hat{b}_3 = -0.5$$

- $\exp(0) = 1$: de base, $P(\text{valider}) = P(\text{pas valider})$.
- $\exp(0.5) \approx 1.65$: chaque jour de travail en plus **multiplie par 1.65** l'odd ratio de valider.
- $\exp(1) \approx 2.7$: chaque cadeau multiplie par 2.7 ...
- $\exp(-0.5) \approx 0.6$: chaque vidéo regardée ...

RECAPITULONS

La régression logistique est le pendant de la régression linéaire pour la classification binaire.

On exprime les probabilités d'observer 1 ou 0 conditionnellement aux valeurs de X avec la fonction logit et des poids b_0, b_1, b_2, \dots

Les poids optimaux sont ceux qui rendent le plus *vraisemblable* d'observer ce qu'on observe (méthode: descente de gradient, solution exacte)

Les poids obtenus s'interprètent comme des multiplicateurs d' "odd ratio" et permettent de prédire la valeur de y pour un nouveau X .

EN PYTHON

```
from sklearn.linear_model import  
    LogisticRegression  
model = LogisticRegression()  
model.fit(Xtrain, Ytrain)  
predictions = model.predict(Xtest)
```

CONCLUSION SUR LA REGRESSION LOGISTIQUE

Avantages

- Une vision probabiliste de la classification
- Interprétation de chaque variable de manière probabiliste
- Simple et rapide à estimer

Inconvénients

- Mêmes que la régression linéaire
- En particulier, mauvais résultats avec beaucoup de variables (over-fitting)

TRANSFORMER DU TEXTE ET DES IMAGES

EX 1: RECONNAISSANCE DE TEXTE

(Exemple déjà vu en cours)

X									Y	
	voici	un	1 ^{er}	texte	2 nd	ce	doc	a		spam
x₁	1	1	1	1	0	0	0	0	y₁	0
x₂	1	1	0	1	1	0	0	0	y₂	0
x₃	0	1	0	1	0	1	1	1	y₃	1

$$(x_1, y_1) = ([1, 1, 1, 1, 0, 0, 0, 0], 0)$$

$$(x_2, y_2) = ([1, 1, 0, 1, 1, 0, 0, 0], 0)$$

$$(x_3, y_3) = ([0, 1, 0, 1, 0, 1, 1, 1], 1)$$

EX 2: RECONNAISSANCE DE CHIFFRES

Données: chiffres manuscrits, de 0 à 4

- 901 exemples
≈ 180 exemples par classe
- Chaque image: 8x8 pixels,
niveaux de gris. Un chiffre =
vecteur de 64 valeurs $\in [0,1]$



Rappel : Objectifs de l'apprentissage

1. Savoir classifier automatiquement les images;
2. Établir des règles de séparation;
3. Appliquer la règle à un nouveau chiffre entrant
(exemple d'application: code postal)

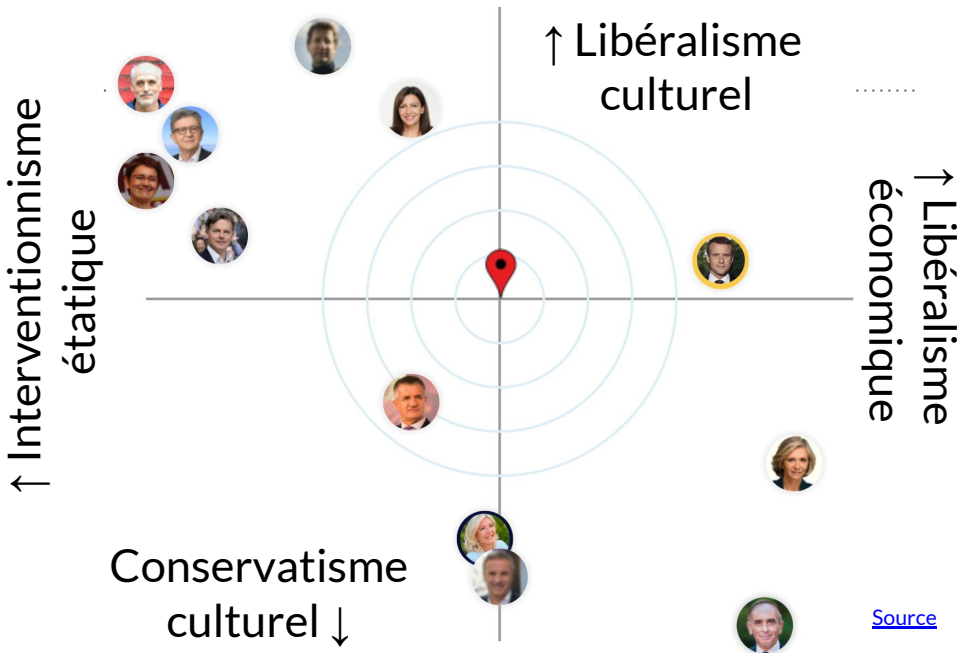
POURQUOI TRANSFORMER LES DONNÉES ?

Il faut **transformer l'entrée** (texte, image, vidéo, etc) pour que la machine puisse la comprendre.

On parle de "**features**" = nouvelles données obtenues en transformant des données brutes.

Manière de **résumer l'information** contenue dans des données complexes.

FEATURE VECTORS: Quoi? Comment?



ENCODER DU TEXTE

EXEMPLE 1 : ANTI-SPAM TEXTOS

Données: messages SMS en anglais

- **747 spams**. Exemple: "WINNER!! As a valued network customer you have been selected to receive a prize reward! [...]"
- **4827 non-spams**, aka "**ham**". Exemple: "Nah I don't think he goes to usf, he lives [...]"
- **9663 mots distincts** sur l'ensemble

Objectifs :

1. Comprendre ce qui différencie ces 2 types.
2. Établir une règle de séparation.
3. Appliquer la règle à de nouveaux messages.

INTRODUCTION

Dans notre exemple:.....

- chaque SMS = un **document**
- chaque document = suite de mots (string) ;
- chaque mot = un identifiant unique
- l'ordinateur ne comprend pas les mots...
mais il "comprend les chiffres".

Il faut donc transformer les document en chiffres.

"Le cours de Fouille de Données est top"
devient: [13, 651, 12, 7421, 12, 1201, 99, 432]

INTRODUCTION

Dans notre exemple:.....

- chaque SMS = un **document**
- chaque document = suite de mots (string) ;
- chaque mot = un identifiant unique
- l'ordinateur ne comprend pas les mots...
mais il "comprend les chiffres".

Il faut donc transformer les document en chiffres.

"**Le cours de Fouille de Données est top**"
devient: [**13**, **651**, **12**, **7421**, **12**, **1201**, **99**, **432**]

Mais ça ne suffit pas!

LE MODELE "TF"

"Le cours de Fouille de Données est top"
= [13, 651, 12, 7421, 12, 1201, 99, 432]

Typiquement on veut un vecteur (**de taille fixe!**) de chiffres qui sont des **grandeurs**, et pas une liste de taille variable avec des indices de mots.

On transforme la liste en vecteur de taille N, si il y a N mots dans le dictionnaire:

= [0...0, 2, 1, 0...0, 1, 0...0, 1, 0...0]

LE MODELE "TF"

"Le cours de Fouille de Données est top"
= [13, 651, 12, 7421, 12, 1201, 99, 432]

Typiquement on veut un vecteur (**de taille fixe!**) de chiffres qui sont des **grandeurs**, et pas une liste de taille variable avec des indices de mots.

On transforme la liste en vecteur de taille N, si il y a N mots dans le dictionnaire:

= [0...0, 2, 1, 0...0, 1, 0...0, 1, 0...0]

On peut **normaliser** ce vecteur en s'assurant que la somme des coordonnées soit égale à 1. (*pourquoi?*)

= [0...0, 1/4, 1/8, 0...0, 1/8, 0...0, 1/8, 0...0]

TERM FREQUENCIES (TF)

Document → fréquence de ses mots:

$$TF(t, d) = \frac{|t \in d|}{|d|} = \frac{\text{nombre de fois où le terme apparaît}}{\text{nombre de termes}}$$

Exemple:

- Doc 1: "voici un 1er texte"
- Doc 2: "voici un 2nd texte"
- Doc 3: "ce doc a ce texte"

Encodage:

1: "voici": 1/4, "un": 1/4, "1er": 1/4, "texte": 1/4
2: "voici": 1/4, "un": 1/4, "2nd": 1/4, "texte": 1/4
3: "ce": 2/5, "doc": 1/5, "a": 1/5, "texte": 1/5

UN PROBLEME DU MODELE "TF"

	voici	un	1er	texte	2nd	ce	doc	a
doc1	1/4	1/4	1/4	1/4	0	0	0	0
doc2	1/4	1/4	0	1/4	1/4	0	0	0
doc3	0	0	0	1/5	0	2/5	1/5	1/5
#docs	2	2	1	3	1	1	1	1

On cherche à **différencier** les documents.

Les mots apparaissant dans beaucoup de documents sont moins **discriminants**.

Exemple: le mot "texte" n'a aucun intérêt discriminant. **Il faut donner moins d'importance aux mots apparaissant dans beaucoup de docs.**

INVERSE DOC. FREQUENCIES(IDF)

On a N documents au total. Pour chaque mot, on compte le **nombre D_{mot} de docs où ce mot apparaît**. Alors: $IDF_{\text{tmp}}(\text{mot}) = N / D_{\text{mot}}$.

$$IDF_{\text{tmp}}(t) = \frac{N}{|d : t \in d|} = \frac{\text{nombre de documents}}{\text{nombre de documents où le terme apparaît}}$$

	voici	un	1er	texte	2nd	ce	doc	a
doc1	1/4	1/4	1/4	1/4	0	0	0	0
doc2	1/4	1/4	0	1/4	1/4	0	0	0
doc3	0	0	0	1/5	0	2/5	1/5	1/5
IDF_{tmp}	1.5	1.5	3	1	3	3	3	3

→ Mais ce IDF_{tmp} est “trop fort” [au tableau]

INVERSE DOC. FREQUENCIES(IDF)

IDF est le **logarithme** de IDF_{tmp} : on garde le même principe, mais il **n'annule pas aussi brutalement l'effet des mots importants.**

$$IDF(t) = \ln \left(\frac{N}{|d : t \in d|} \right)$$

	voici	un	1er	texte	2nd	ce	doc	a
doc1	1/4	1/4	1/4	1/4	0	0	0	0
doc2	1/4	1/4	0	1/4	1/4	0	0	0
doc3	0	0	0	1/5	0	2/5	1/5	1/5
IDF	0.4	0.4	1.1	0	1.1	1.1	1.1	1.1

MODELE FINAL : TF-IDF

Tf-Idf : fréquence du terme dans le document pondéré par l'importance inverse du terme dans l'ensemble des documents.

$$\text{TfIdf}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

	voici	un	1er	texte	2nd	ce	doc	a
doc1	0.1	0.1	0.28	0	0	0	0	0
doc2	0.1	0.1	0	0	0.28	0	0	0
doc3	0	0	0	0	0	0.44	0.22	0.22

[illegible]

CONCLUSION SUR LE TF-IDF

Il nous dit pour chaque document quels sont les mots qui apparaissent particulièrement plus souvent dans ce document que dans les autres.

Résultats intéressants en soi (e.g. wordcloud), mais but principal: modèles prédictifs.

Le TF-IDF “brut” ne corrige pas les fautes de frappe; et se rend pas compte que “vends” et “vendre” ont un sens très lié.

En fonction de la qualité des données en entrée, vous allez peut-être devoir faire quelques travaux manuels.

NETTOYAGE DE TEXTE

Certains facteurs peuvent modifier crucialement le modèle:

- Casse
- Fautes d'orthographe ou de frappe
- Mots de la même famille
- Ponctuation/Smileys
- Chiffres
- URLs
- Synonymes
- Caractères spéciaux
- Acronymes, mots "texto"

Là commence le travail du data miner: que modifier? comment? Pas de solution unique (mais certaines meilleures que d'autres!).

LA CASSE

"Yes", "YES", "yes"

Ces trois mots sont les mêmes, mais les spams ont tendance à utiliser plus de majuscules (qui se suivent).

Les noms propres ("France", "Sharon", "Destiny") peuvent aussi poser problème.

→ **Idées?**

FAUTES D'ORTHOGRAPHE / TYPOS

"helo", "hello", "helol"

On peut passer un correcteur orthographique.

Mais les spams ont plus de fautes.

→ **Idées?**

MOTS DE LA MEME FAMILLE

"write", "writing", "wrote", "writer", ...

Stemming :

writ(e), writ(ing), writ(er) | wrot(e)

Lemmatization:

write = writing = wrote = writer

... Si on ces mots sont connus dans notre DB. Sinon: on garde l'original.

PONCTUATION

"U.S.A.", "<3", "...", "!?", ...

On peut choisir de supprimer certains types de ponctuation (U.S.A. = USA)

On peut à l'inverse choisir de les garder.

Attention à la séparation des mots: "Hello,you"

CHIFFRES

"0800555344", "0800655877", "4", "100", ...

On peut dans un premier temps les supprimer ou les neutraliser (tous les chiffres = 0)

Mais on peut aussi détecter leur formes spéciales: n° tel, présence d'un \$ ou € à côté, ..

Normalisation de ceux qu'on garde: nb chiffres?

"www.cashbin.co.uk", "www.b4utele.com"

→ Il sera souvent difficile de détecter si une URL provient d'un spam ou non. En revanche, on peut regrouper toutes les URL comme un même "mot".

Les synonymes ("great", "amazing", "fantastic")

→ Il est dommage d'encoder comme deux mots différents des mots qui ont le même sens. Une étape avancée peut donc consister à utiliser un dictionnaire de synonymes pour regrouper des mots très similaires sous un identifiant unique.

"\$", "@", "%"

→ Certains de ces caractères spéciaux ont-ils un sens particulier? Par exemple, "\$" peut-être associé à un chiffre et nous permettre d'identifier un prix...

":-P", "lol", "LMAO", ":/"

→ Comme pour les smileys, on peut définir un dictionnaire d'expressions typiques "textos". Ce n'est pas une priorité.

Faut-il tout faire? Dans quel ordre?

Il y a potentiellement beaucoup de choses à modifier. Toutes n'ont pas la même importance et il est rare qu'on pense à tout immédiatement. On peut **procéder itérativement**:

Tant qu'on n'est pas satisfait:

- 1 Apporter une nouvelle amélioration
- 2 Encoder
- 3 Faire les analyses/ prédictions

Rapport temps/qualité du modèle: préfère-t-on l'ultra-performance en 30 heures de travail ou quelque chose de correct en 1 heure?

OPTIMISATION DES FEATURES

(aka: FEATURE ENGINEERING)

VARIABLES CATEGORIELLES

Jour de la semaine.....

“perdu, égalité, gagné”

Représenter des goûts:

Aimer Batman, Superman, Ironman, ...

VARIABLES CATEGORIELLES

Jour de la semaine.....

→ “1-Hot encoding” [au tableau]

“perdu, égalité, gagné”

Représenter des goûts:

Aimer Batman, Superman, Ironman, ...

VARIABLES CATEGORIELLES

Jour de la semaine.....

→ “1-Hot encoding” [au tableau]

“perdu, égalité, gagné”

→ perdu=0, égalité=1, gagné=2, car valeurs
ordonnées: gagné > égalité > perdu.

Représenter des goûts:

Aimer Batman, Superman, Ironman, ...

VARIABLES CATEGORIELLES

Jour de la semaine.....

→ “1-Hot encoding” [au tableau]

“perdu, égalité, gagné”

→ perdu=0, égalité=1, gagné=2, car valeurs ordonnées: gagné > égalité > perdu.

Représenter des goûts:

Aimer Batman, Superman, Ironman, ...

→ 1-Hot, mais la somme peut être $\neq 1$

VARIABLES CATEGORIELLES

Jour de la semaine.....

→ “1-Hot encoding” [au tableau]

“perdu, égalité, gagné”

→ perdu=0, égalité=1, gagné=2, car valeurs ordonnées: gagné > égalité > perdu.

Représenter des goûts:

Aimer Batman, Superman, Ironman, ...


→ 1-Hot, mais la somme peut être $\neq 1$

... etc. Vous rencontrerez tous types de données! En cas de doute, demander!

AJUSTER LES VALEURS

Dans un même dataset, les valeurs numériques peuvent avoir des intervalles de valeurs complètement différents selon les colonnes.

[Au tableau]

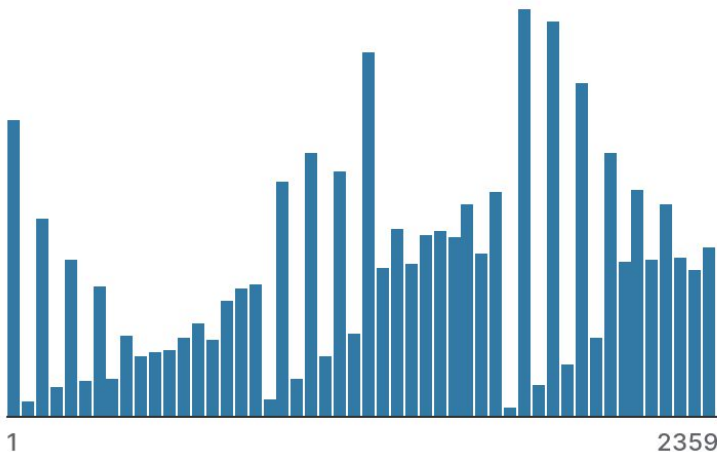
- **Visualiser!**
- $[-\infty, +\infty] \rightarrow [a, b]$?
 - Cap?
 - Log? (ex: Foot. Latence.  aux zéros!)
- **Normalisation** à l'intérieur d'une colonne ?
 $\rightarrow [0, 1]$? $[-1, 1]$? Moyenne/Mediane? Extrêmes?
- **Normalisation** des colonnes entre elles ?

Crime Data in Los Angeles (2020)

..... [Data Card](#) [Code \(1\)](#) [Discussion \(0\)](#)

TIME OCC

Source:
Kaggle



DONNEES MANQUANTES, ERRONEES

C'est la vraie vie. Votre dataset est incomplet, ou certaines valeurs sont bizarres ("N/A", "NaN", NULL). Beaucoup d'approches possibles. Par ordre d'intelligence:

- Enlever les lignes où il manque quelque chose
- Remplacer par 0, ou autre valeur bateau
- Remplacer par **une** moyenne
- ... ditto, et ajouter une colonne "présence".
- Règles ad-hoc, dépendant de la colonne, du reste des données, et de l'algorithme, donc: **en connaissance de cause.**

AJOUTER DES COLONNES / DONNEES

- Combinaisons de colonnes
Ex: Regr. lineaire, N^2 , $N \times M$, M^2 , N^2M , NM^2 , ...
- Eclatement de colonnes
Ex. one-hot; “indicator” column for N/A; ...
- Fonction connue d'une ou plusieurs colonnes
Ex. conductivité, température \rightarrow force camion
- Clamp d'une colonne : garder la colonne originale
- ... etc, il y a toujours d'autres idées !

ADAPTER LE Y

Exemple:.....

Valeur de footballeur (100K .. 100M); Durée d'un solveur sur un modèle combinatoire (10 μ s...1h).

→ On peut essayer fitter $\log(Y)$ au lieu de Y

Avantage: erreur *relative* au lieu d'**absolue**.

Autres fonctions possibles : à vous d'imaginer!



Ça peut aussi créer des problèmes



Il peut être judicieux d'appliquer des mêmes transformations à des colonnes de X .



Ne pas oublier d'inverser \hat{Y} à la fin.

IMPORTANCE DES COLONNES

Votre dataset, après encodage en features, a 1000 colonnes (ou 10K, 100K, etc).

Est-ce un problème ?

Ça dépend de l'algorithme!

- Régression linéaire, logistique: **Aïe!**
→ Lasso, Ridge, Elastic-Net ...
- Distances, clustering: **Ouille!**
- Random Forest: **Meh**

Et ça dépend des colonnes!

UN EXEMPLE

Exemple: Classification de la catégorie d'un film (Comédie, ...)

Données: Titre, Durée, Acteurs

Features: TF-IDF du titre, Durée, 1-Hot des acteurs

→ Trop (?) de colonnes par rapport à l'info d'un film donné!