Méthodes d’apprentissage

IFT603 - 712

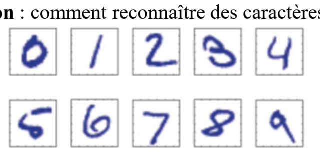
Concepts fondamentaux

Par
Pierre-Marc Jodoin
/
Hugo Larochelle

1

Apprentissage Automatique

Question : comment reconnaître des caractères manuscrits?



Réponse : Énumérer des règles?

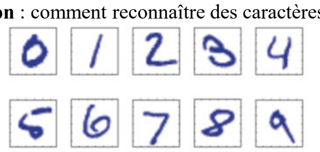
- Une série de pixels alignés => ‘1’
- Une série de pixels en rond => ‘0’
- Etc.

2

2

Apprentissage Automatique

Question : comment reconnaître des caractères manuscrits?



Réponse : ~~Énumérer des règles?~~ NON!

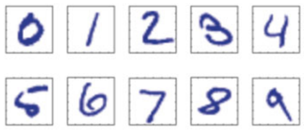
- Généralise mal à tous les cas 

3

3

Apprentissage Automatique

Question : comment reconnaître des caractères manuscrits?



Réponse : ~~Énumérer des règles?~~ NON!

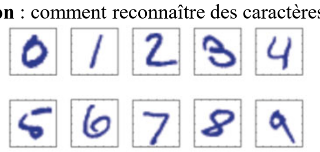
- Généralise mal à tous les cas 
- Souvent fastidieux 

Chien Vs Oiseaux

4

Apprentissage Automatique

Question : comment reconnaître des caractères manuscrits?



Réponse : Laisser l'ordinateur « **apprendre** » les règles

- Algorithmes d'apprentissage (*machine learning*)

5

Deux grandes approches

Apprentissage supervisé

Apprentissage non-supervisé.

6

6

Apprentissage supervisé

On fournit à l'algorithme des **données d'entraînement**

'0' '0' '0' '1' '0' '1' '1'

…et l'algorithme retourne une fonction capable de **généraliser** à de nouvelles données

? ? ? ? ? ? ?

7

7

Apprentissage supervisé

On fournit à l'algorithme des **données d'entraînement**

'0' '0' '0' '1' '0' '1' '1'

On note **l'ensemble d'entraînement**

$$D = \{(\vec{x}_1, t_1), (\vec{x}_2, t_2), \ldots, (\vec{x}_N, t_N)\}$$

où $\vec{x}_i \in \Re^d$ est une **entrée** (donnée brute) et t_i est la **cible**

8

8

Objectif des algorithmes d'apprentissage

Partant d'un **ensemble d'entraînement:** $D = \{(\vec{x}_1, t_1), (\vec{x}_2, t_2), \ldots, (\vec{x}_N, t_N)\}$

$\vec{x}_i \in \Re^d$ donnée

t_i cible associée à $\vec{x}_i$

le but est **d'apprendre** une function qui sache prédire t_i partant de $\vec{x}_i$

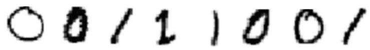
$$y_{\vec{w}}(\vec{x}_i) \to t_i$$

où $\vec{w}$ sont les **paramètres** du modèle

9

Apprentissage supervisé

Une fois le modèle $y_{\vec{w}}(\vec{x})$ entraîné, on utilise un **ensemble de test** D_{test} pour mesurer la performance du modèle en **généralisation.**



'0' '0' '1' '1' '1' '0' '0' '1'

10

10

Deux grandes approches

Apprentissage supervisé

Apprentissage non-supervisé.

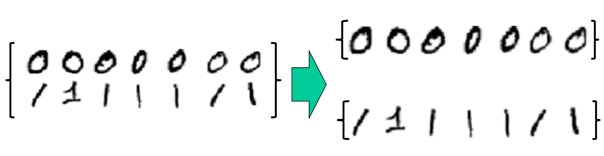
11

11

Apprentissage non supervisé

L'apprentissage non-supervisé est lorsqu'une cible n'est pas explicitement donnée

- Partitionnement de données / *clustering*



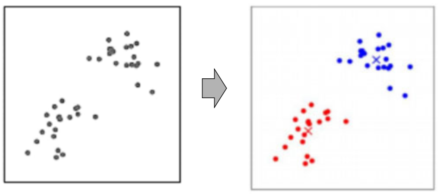
12

12

Apprentissage non supervisé

L'apprentissage non-supervisé est lorsqu'une cible n'est pas explicitement donnée

- Partitionnement de données / *clustering*



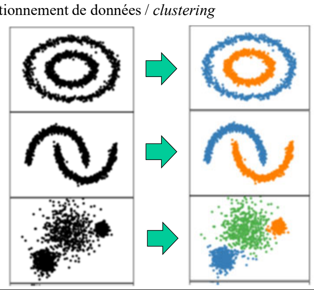
13

13

Apprentissage non supervisé

L'apprentissage non-supervisé est lorsqu'une cible n'est pas explicitement donnée

- Partitionnement de données / *clustering*



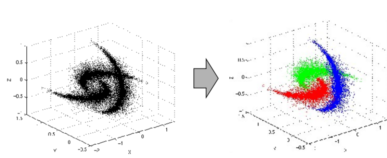
14

14

Apprentissage non supervisé

L'apprentissage non-supervisé est lorsqu'une cible n'est pas explicitement donnée

- Partitionnement de données / *clustering*



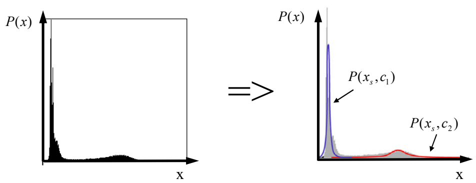
15

15

Apprentissage non supervisé

L'apprentissage non-supervisé est lorsqu'une cible n'est pas explicitement donnée

- A souvent pour but d'apprendre une loi de probabilité $p(x)$ dont les données sont issues



16

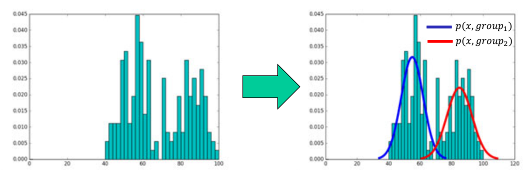
16

Apprentissage non supervisé

L'apprentissage non-supervisé est lorsqu'une cible n'est pas explicitement donnée

- A souvent pour but d'apprendre une loi de probabilité $p(x)$ dont les données sont issues

Exemple : trouver 2 groupes d'étudiants suite à un examen



17

Apprentissage non supervisé

L'apprentissage non-supervisé est lorsqu'une cible n'est pas explicitement donnée

- A souvent pour but d'apprendre une loi de probabilité $p(x)$ dont les données sont issues

Autres applications

- Compression de fichiers
- Visualisation de données
- Segmentation d'images
- etc.

18

Supervisé vs non supervisé

Principal sujet du cours

Apprentissage supervisé : il y a une cible

$$D = \{(\vec{x}_1, t_1), (\vec{x}_2, t_2), \ldots, (\vec{x}_N, t_N)\}$$

Apprentissage non-supervisé : la cible n'est pas fournie

$$D = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N\}$$

19

19

Apprentissage supervisé

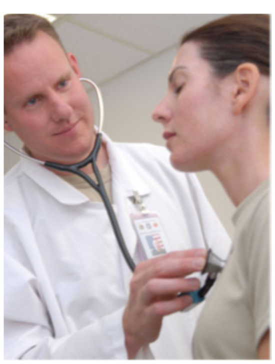
Deux grandes familles d'applications

- **Classification :** la cible est un indice de classe $t \in \{1, \ldots, K\}$
  - Exemple : reconnaissance de caractères
    - ✓ $\vec{x}$: vecteur des intensités de tous les pixels de l'image
    - ✓ t : identité du caractère
- **Régression :** la cible est un nombre réel $t \in \mathbb{R}$
  - Exemple : prédiction de la valeur d'une action à la bourse
    - ✓ $\vec{x}$: vecteur contenant l'information sur l'activité économique de la journée
    - ✓ t : valeur d'une action à la bourse le lendemain

20

20

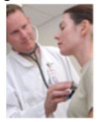
Exemple simple de classification binaire



From Wikimedia Commons
the free media repository

21

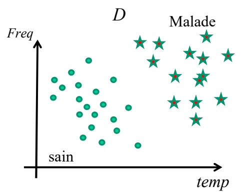
Exemple simple de classification binaire



D

	(temp, freq)	diagnostique
Patient 1	(37.5, 72)	Sain
Patient 2	(39.1, 103)	Malade
Patient 3	(38.3, 100)	Malade
	(…)	…
Patient N	(36.7, 88)	Sain

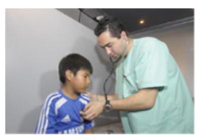
$\bar{x}$ t

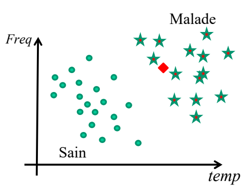


22

Exemple simple de classification binaire

Un nouveau patient vient à l'hôpital
Comment peut-on en déduire son état?


From Wikimedia Commons
the free media repository

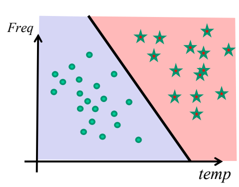


23

Solution


From Wikimedia Commons
the free media repository

Diviser l'espace des caractéristiques en deux régions : sain et malade

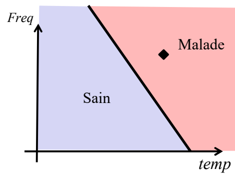


24

Solution

From Wikimedia Commons
the free media repository

Diviser l'espace des caractéristiques en deux régions : sain et malade

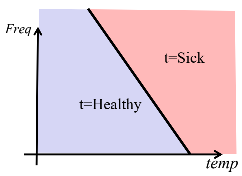


25

Plus formellement

From Wikimedia Commons
the free media repository

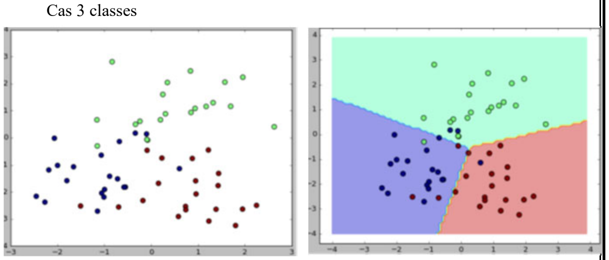
$$y_{\vec{w}}(\vec{x}) = \begin{cases} Sain \text{ si } \vec{x} \text{ est dans la région bleue} \\ Malade \quad \text{sinon} \end{cases}$$



26

Exemple de classification

Cas 3 classes



Soit des données issues de 3 classes ●, ●, ●
dans un espace à 2 dimensions

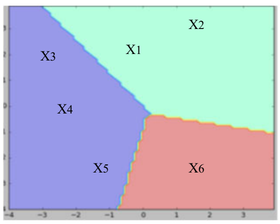
Une fois l'entraînement terminé

y(●) = class 1
y(●) = class 2
y(●) = class 3

27

Exemple de classification

Une fois l'entraînement terminé, on a une function qui converti un point 2D en une etiquette de classe class label.



y(X1) => class
y(X2) => class
y(X3) => class
y(X4) => class
y(X5) => class
y(X6) => class

28

Exemple de base de données de classification

Inria person dataset

Positif

Négatif

29

Exemple de base de données de classification

Inria person dataset

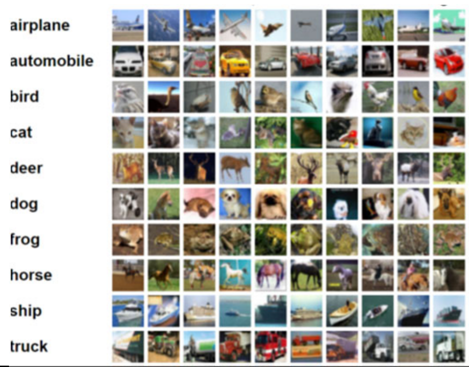
- 2 classes
- 20,252 images,
  => 14,596 entraînement
  => 5,656 test
- Chaque image est en RGB
  => 64x128x3

On peut simplement **vectoriser ces images** et les représenter par des vecteurs de 64x128x3 = **9,984 dimensions.**

30

Exemple de base de données de classification

Exemples, Cifar10



31

Exemple de base de données de classification

Exemples, Cifar10

- 10 classes
- 60,000 images,
  => 50,000 entraînement
  => 10,000 test
- Chaque image est RGB
  => 32x32x3

On peut simplement **vectoriser ces images** et les représenter par des vecteurs de 32x32x3 = **3072 dimensions.**

32

Exemple de base de données de classification

Exemples, mnist

33

Exemple de base de données de classification

Exemples, mnist

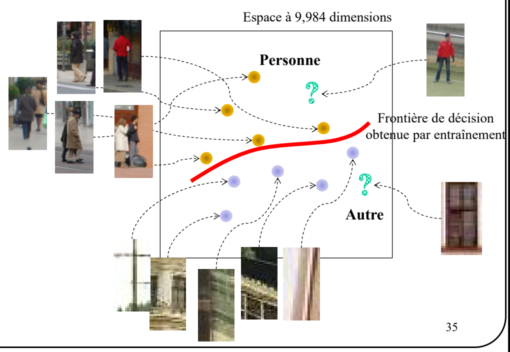
- 10 classes
- 70,000 images
  => 60,000 entraînement
  => 10,000 test
- Les images sont en niveaux de gris
  => 28x28

On peut simplement **vectoriser ces images** et les représenter par des vecteurs de 28x28 = **784 dimensions.**

34

Apprentissage supervisé

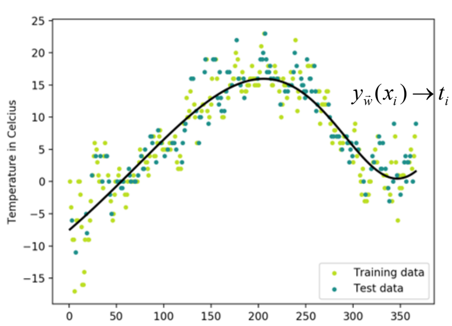
Inria person dataset



35

35

Exemple de régression



36

36

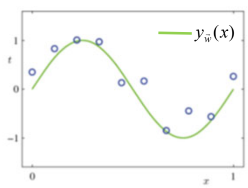
Exemple formel:
régression 1D

37

37

Régression 1D

Exemple simple: régression 1D



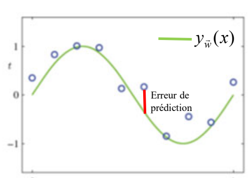
- **Données**
  - ✓ entrée : scalaire x
  - ✓ cible : scalaire t
- **Ensemble d'entraînement** D contient:
  - ✓ $X = (x_1, \dots, x_N)^T$
  - ✓ $T = (t_1, \dots, t_N)^T$
- **Objectif :**
  - ✓ Faire une prédiction $\hat{t}$ pour chaque nouvelle entrée $\hat{x}$

38

38

Régression 1D

Exemple simple: régression 1D

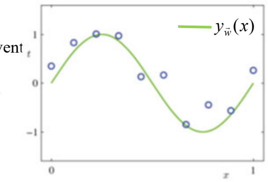


- **Données**
  - ✓ entrée : scalaire x
  - ✓ cible : scalaire t
- **Ensemble d'entraînement** D contient:
  - ✓ $X = (x_1, \dots, x_N)^T$
  - ✓ $T = (t_1, \dots, t_N)^T$
- **Objectif :**
  - ✓ Faire une prédiction $\hat{t}$ pour chaque nouvelle entrée $\hat{x}$

39

39

Régression polynomiale



- On va supposer que nos données suivent une **forme polynomiale**

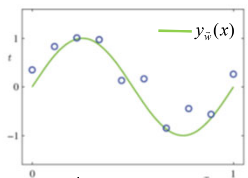
$$y_{\vec{w}}(x) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$
$$= \sum_{i=0}^{M} w_i x^i$$

- $y_{\vec{w}}(x)$ est notre **modèle**
  - ✓ Représente nos hypothèses sur le problème à résoudre
  - ✓ Un modèle a toujours des paramètres qu'on doit trouver (ici $\vec{w}$)

40

40

Inconnues



$$y_{\vec{w}}(x) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$
$$= \sum_{i=0}^{M} w_i x^i$$

Deux inconnues

$$\vec{w} \in R^M$$

Paramètres

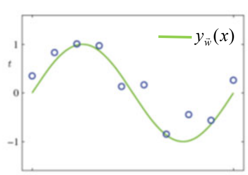
$$M \in \mathrm{N}^{\geq 0}$$

Hyper-paramètre

41

41

Entraînement



$$y_{\vec{w}}(x) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$
$$= \sum_{i=0}^{M} w_i x^i$$

Deux inconnues

$$\vec{w} \in R^M$$

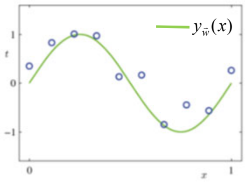
$$M \in \mathrm{N}^{\geq 0}$$

Entraînement = trouver w (et parfois M) à partir de X et T

42

42

Régression polynomiale



➢ Une fois entraîné, un modèle prédit la cible d'une nouvelle entrée *x* à l'aide d'un bout de code comme celui-ci:

```
def predict(x,w):
    x_poly = x ** np.arange(len(w))
    return np.dot(x_poly,w)
```

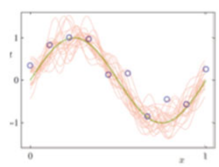
➢ $y_{\vec{w}}(x)$ est notre **modèle**

- ✓ Représente nos hypothèses sur le problème à résoudre
- ✓ Un modèle a toujours des paramètres qu'on doit trouver (ici $\vec{w}$)

43

43

Régression polynomiale



➢ Connaissant ***M***, comment trouver le bon $\vec{w}$?

Le « meilleur » $\vec{w}$ est celui qui minimise la somme de notre perte / erreur / coût sur les données d'entraînement

$$E_D(\vec{w}) = \sum_{n=1}^{N} \left(y_{\vec{w}}(x_n) - t_n \right)^2$$

➢ La solution à ce problème sera vue au **chapitre 3.**

$$\vec{w} = \arg\min_{\vec{w}} E_D(\vec{w})$$

44

44

Sur- et sous-apprentissage

➢ Comment trouver le bon ***M***?

Le problème avec les hyper-paramètres est qu'ils ne **peuvent pas être estimés** à l'aide des **algorithmes d'optimisation classiques** (descente de gradient, méthode de Newton, etc.) comme pour les paramètres $\vec{w}$.

Par conséquent, on fixe souvent « ***à la main*** » **les hyper-paramètres**.

Mais attention, leur valeur influence grandement le **résultat final**.

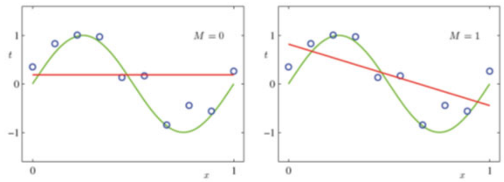
45

45

Sur- et sous-apprentissage

➢ Comment trouver le bon ***M***?

Un petit *M* donne un modèle trop simple causant du **sous-apprentissage**



$E_D(\vec{w}) \Rightarrow$ élevée

$E_{D_{test}}(\vec{w}) \Rightarrow$ élevée

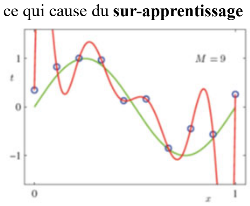
46

46

Sur- et sous-apprentissage

➢ Comment trouver le bon ***M***?

Un grand *M* donne un modèle qui « apprend par cœur » les données d'apprentissage ce qui cause du **sur-apprentissage**



$E_D(\vec{w}) \Rightarrow$ TRÈS faible

$E_{D_{test}}(\vec{w}) \Rightarrow$ élevée

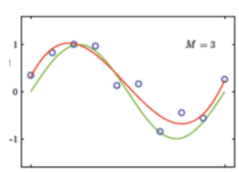
47

47

Sur- et sous-apprentissage

➢ Comment trouver le bon ***M***?

Idéalement, il faudrait une valeur intermédiaire de sorte que **l'erreur d'entraînement et de test soient faibles**.



$E_D(\vec{w}) \Rightarrow$ faible

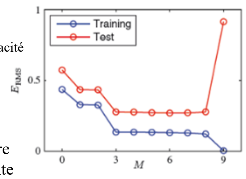
$E_{D_{test}}(\vec{w}) \Rightarrow$ faible

48

48

Sur- et sous-apprentissage

Capacité d'un modèle
- ✓ aptitude d'un modèle à apprendre «par coeur»
- ✓ exemple : plus M est grand, plus le modèle a de capacité



Plus la capacité est grande, plus la différence entre l'erreur d'entraînement et l'erreur de test augmente
- ✓ en régression, l'erreur sur tout un ensemble est souvent mesurée par la racine de la moyenne des erreurs au carré (*root-mean-square error*)

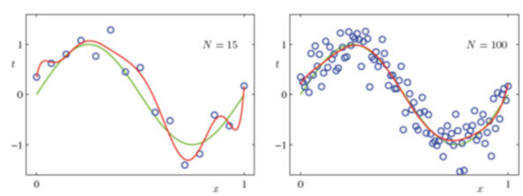
$$E_{RMS} = \sqrt{\frac{E(\vec{w})}{N}}$$

49

49

Généralisation

Plus la quantité de données d'entraînement augmente, plus le modèle entraîné va bien généraliser



50

50

Régularisation

Valeurs apprise des paramètres $\vec{w}$ pour différents $\boldsymbol{M}$ sans régularisation

	$M=0$	$M=1$	$M=3$	$M=9$
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18
w_8				-557682.99
w_9				125201.43

51

51

Régularisation

Lorsqu'on souhaite éviter qu'on modèle sur-apprenne

1. On sélectionne un petit « M »
2. On réduit la capacité du modèle par **régularisation**

Exemple : on pénalise la somme du carré des paramètres

Constante qui contrôle la capacité

$$E_D(\vec{w}) = \sum_{n=1}^{N} \left(t_n - y_{\vec{w}}(\vec{x})\right)^2 + \lambda \|\vec{w}\|^2$$

$$\|\vec{w}\|^2 = \vec{w}^{\mathrm{T}}\vec{w} = w_0^2 + w_1^2 + \ldots + w_M^2$$

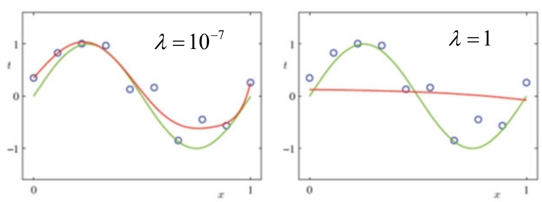
Modèle de Ridge!

52

52

Régularisation

Forte régularisation = modèle moins flexible

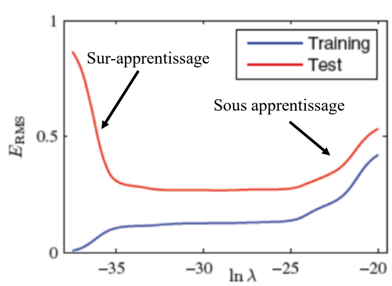


53

53

Régularisation

Forte régularisation a un influence sur l'erreur d'entraînement et de test



54

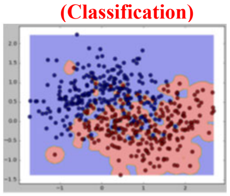
54

On peut également sur- et sous-apprendre en classification

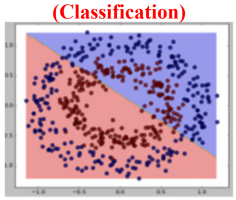
55

55

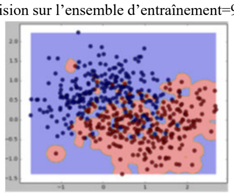
Sur-apprentissage (Classification) | **Sous-apprentissage (Classification)**



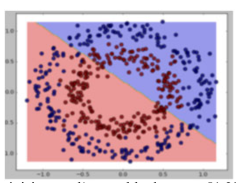
Précision sur l'ensemble d'entraînement=99.6%



Précision sur l'ensemble d'entraînement=52.2%



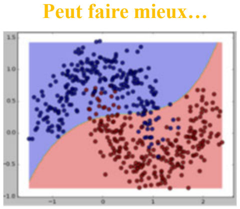
Précision sur l'ensemble de test = 78%



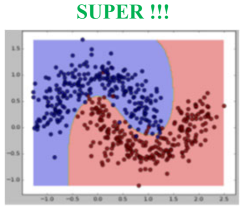
Précision sur l'ensemble de test = 51.2%

56

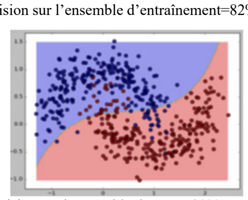
Peut faire mieux… | **SUPER !!!**



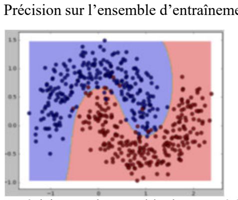
Précision sur l'ensemble d'entraînement=82%



Précision sur l'ensemble d'entraînement=97.8%



Précision sur l'ensemble de test = 80%



Précision sur l'ensemble de test = 96.2%

57

$$E_D(\vec{w}) = \sum_{n=1}^{N} \left(y_{\vec{w}}(x_n) - t_n \right)^2 + \lambda \|\vec{w}\|^2$$

$$\|\vec{w}\|^2 = \vec{w}^T \vec{w} = w_0^2 + w_1^2 + \ldots + w_M^2$$

Sélection de modèle

<u>Comment trouver les bons hyper-paramètres?</u>

M et λ

58

58

Sélection de modèle

Comment trouver le bon M et le bon λ ?

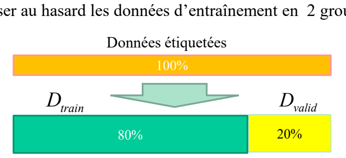
- **Très mauvaise solution** : choisir au hasard
- **Mauvaise solution** : prendre plusieurs paires (M, λ) et garder celle dont l'erreur d'entraînement est la plus faible
  - ➢ Sur-apprentissage
- **Mauvaise solution** : prendre plusieurs paires (M, λ) et garder celle dont l'erreur de test est la plus faible
  - ➢ D_{test} ne doit pas être utilisé pour entraîner le modèle
- <u>**Bonne solution**</u> : prendre plusieurs paires (M, λ) et garder celle dont <u>**l'erreur de validation**</u> est la plus faible

59

59

Validation croisée *(cross-validation)*

1- Diviser au hasard les données d'entraînement en 2 groupes



2- Pour M allant de M_{min} à M_{max}

Pour λ allant de λ_{min} à λ_{max}

Entraîner le modèle sur D_{train}

Calculer l'erreur sur D_{valid}

3- Garder la paire (M,λ) dont <u>**l'erreur de validation**</u> est la plus faible

60

Validation croisée K fois *(k-fold cross-validation)*

Pour M allant de $M_{\min}$ à $M_{\max}$
Pour λ allant de $\lambda_{\min}$ à $\lambda_{\max}$
Pour j allant de 0 à K

Diviser au hasard les données d'entraînement => D_{train} D_{valid}

Entraîner le modèle sur D_{train}
Calculer l'erreur sur D_{valid}

Garder la paire *(M,λ)* dont **l'erreur de validation MOYENNE** est la plus faible

61

Exemple d'une validation croisée avec K = 10

Erreur moyenne

Écart type

```
2.832 (+/-0.116) for {'regression': 'poly', 'M': 3, 'lambda': 0.01}
1.854 (+/-0.072) for {'regression': 'poly', 'M': 3, 'lambda': 0.1}
1.910 (+/-0.065) for {'regression': 'poly', 'M': 3, 'lambda': 1}
1.902 (+/-0.077) for {'regression': 'poly', 'M': 3, 'lambda': 10}
2.844 (+/-0.101) for {'regression': 'poly', 'M': 4, 'lambda': 0.01}
2.864 (+/-0.089) for {'regression': 'poly', 'M': 4, 'lambda': 0.1}
1.910 (+/-0.065) for {'regression': 'poly', 'M': 4, 'lambda': 1}
1.894 (+/-0.086) for {'regression': 'poly', 'M': 4, 'lambda': 10}
2.848 (+/-0.080) for {'regression': 'poly', 'M': 5, 'lambda': 0.01}
1.904 (+/-0.064) for {'regression': 'poly', 'M': 5, 'lambda': 0.1}
0.916 (+/-0.069) for {'regression': 'poly', 'M': 5, 'lambda': 1}
1.870 (+/-0.072) for {'regression': 'poly', 'M': 5, 'lambda': 10}
2.846 (+/-0.090) for {'regression': 'poly', 'M': 6, 'lambda': 0.01}
2.906 (+/-0.062) for {'regression': 'poly', 'M': 6, 'lambda': 0.1}
1.904 (+/-0.075) for {'regression': 'poly', 'M': 6, 'lambda': 1}
2.858 (+/-0.112) for {'regression': 'poly', 'M': 6, 'lambda': 10}
```

Meilleur!

M=5,
λ = 1

62

En résumé, un algorithme d'apprentissage

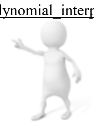
- ✓ entraîne un **modèle** à partir d'un **ensemble d'entraînement**, pouvant faire des prédictions sur de nouvelles données
- ✓ a des **hyper-paramètres** qui contrôlent la **capacité** du modèle entraîné, choisis à l'aide d'une procédure de **sélection de modèle**
- ✓ mesure sa performance de **généralisation** sur un **ensemble de test**
- ✓ Aura une meilleure performance de généralisation si la **quantité de données d'entraînement augmente**
- ✓ Peut souffrir de **sous-apprentissage** (pas assez de capacité) ou de **sur-apprentissage** (trop de capacité)

63

63

Bien que nous n’ayons pas encore vu les algorithmes permettant de faire de la régression, vous pouvez déjà en explorer les tenants et les aboutissants avec **sklearn** et la fonction **« Ridge »**.

scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html
scikit-learn.org/stable/auto_examples/linear_model/plot_polynomial_interpolation.html



64

64