INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN

Departement Architecture des Systèmes d'Informations

Projet Traitement des Signaux Aléatoires
Support Vector Regression

Étudiant :
Ambre Baffert

Enseignant :
Romain Hérault

12 juin 2020

Table des matières

Chapitre 1

Etude bibliographique sur le modèle

1.1 Introduction au modèle

Support Vector Regression (SVR) est un cas particulier du Support Vector Machine (SVM). En effet le SVM permet de résoudre les problèmes de classification en prédisant des étiquettes de classes alors que le SVR permet de prédire une valeur continue.

1.1.1 Principe du SVR

Le principe est le suivant, fixer un seuil d'erreur contenant nos données d'apprentissage (délimité par deux frontières de décision). Au sein de ce seuil, on va chercher la fonction qui passe par le plus de points d'apprentissage, cette fonction est nommée l'hyperplane.

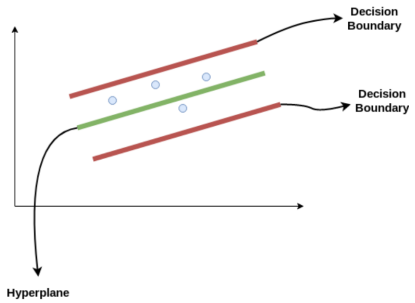


FIGURE 1.1 – Schéma principe SVR

Sur la ci-dessus, on représente une régression linéaire, néanmoins la régression peut etre polynomiale..., elle dépend du paramètre kernel utilisé.

1.2 Support Vector Regression

Maintenant que nous avons vu rapidement le principe de la méthode, voyons ce que cela veut dire mathématiquement. Dans le cas linéaire on suppose que l'hyperplane a pour équation $y = wx + b$ Les équations des frontières de décision sont alors $wx + b = +\epsilon$ et $wx + b = -\epsilon$. On va donc chercher l'hyperplane qui satisfait cette condition : $-\epsilon < y - wx - b < +\epsilon$

De manière plus générale, on va chercher une fonction $f : \chi \to \mathbb{R}$ tel que $\forall i \mid f(x_i) - y_i \mid < \epsilon$

1.2.1 Les paramètres du modèle

Les modèles sont configurés grâce à différents paramètres :

1.2.1.1 Epsilon

ϵ défini la tolérance du modèle. C'est ce paramètre qui va permettre d'ajuster le modèle aux données pour éviter les problèmes de sousapprentissage (pour un ϵ trop élevé) ou le risque de suraprentissage (pour un ϵ trop faible). Graphiquement il s'agit de la largeur du tube (espace entre les deux lignes rouges) sur la figure présentée au début.

1.2.1.2 C

Le paramètre C [1] permet d'arbitrer entre une classification correcte et la minimalisation de la complexité de la fonction de prédiction établie. Un C élevé donnera une marge plus faible si la fonction de prédiction est meilleure pour classer correctement tous les points. Visuellement un C élevé aura tendance à créer des classes autour de chaque points, alors qu'un C faible aura tendance à créer des classes de points très vastes. Pour le SVR cela veut dire selon la valeur du C, la fonction sera plus ou moins complexe.

C a été exprimé analytiquement par (Cherkassy and Ma, 2004) [2] : $C = \max(|\bar{z} + 3\upsilon_z|, |\bar{z} - 3\upsilon_z|)$ avec

— $\bar{z}$: la moyenne des réponses de l'ensemble d'apprentissage

— υ_z : l'écart type des réponses de l'ensemble d'apprentissage

1.2.1.3 gamma

le paramètre γ défini le poids d'une seule donnée d'entrainement. Il est utilisé pour un kernel polynomiale, sigmoid et rbf. Visuellement, un γ élevé va réduire la zone de classification des poids là où il y en a le plus, permettant ainsi de ne pas prendre en compte les points trop loin de l'hyperplane.

Voici une figure [3] pour bien voir les rôles de γ et C.

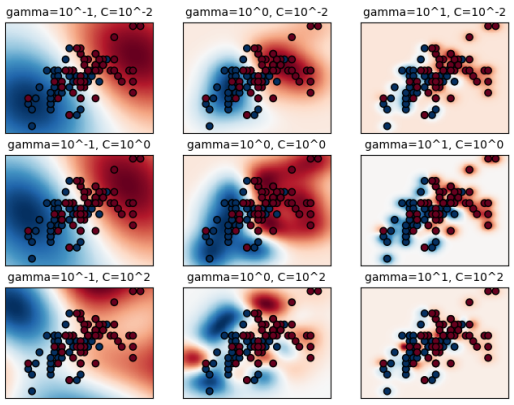


FIGURE 1.2 – Comparer C et gamma

1.2.1.4 kernel

Ce paramètre est utilisé pour faire regréssion autre que linéaire. Il peut être polynomial, sigmoid, ou rbf.

1. https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html
2. Nassim Laouti. Diagnostic de défauts par les Machines à Vecteurs Supports : application à différents systèmes mutivariables nonlinéaires. Autre. Université Claude Bernard - Lyon I, 2012. Français. ffNNT : 2012LYO10161ff. fftel-00985437
3. https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

Chapitre 2

Etude bibliographique sur l'application

La métérologie est la science qui étudie les phénomènes atmosphériques. Elle vise à prédire des données métérologiques, selon des mesures de pression, température, vent, précipitations etc...

2.1 Historique

Au XVIIè siècle, les premiers instruments de mesure apparaissent avec l'invention du baromètre et du thermomètre. Commencent alors les registres de données métérologiques réalisés par la Société Royale de Médecine entre 1778 et 1793 [1]. L'intérêt pour la métérologie, et sa démocratisation s'est fait par le biais des marins qui souhaitent connaitre les conditions en mer. C'est pourquoi Urbain Le Verrier convainc Napoléon III d'ouvrir un centre de métérologie qui permettrait d'avertir les marins sur les conditions métérologiques en mer, et notamment d'éviter les tempêtes. Un réseau de station se créer alors, et les télégrammes d'avertissement sont publiés dans les ports. Les moyens de mesure continuent d'évoluer. C'est en 1878, que le service métérologie n'est plus dépendant de l'Observatoire et devient le Bureau Central Métérologique, il est rattaché au Ministère de l'Instruction Publique. Les premiers bulletins de prévisions métérologiques sont diffusés pour la première fois en 1922 depuis l'antenne de la Tour Eiffel.

2.2 Système français de métérologie actuel

De nos jours, les appareils de mesures sont bien plus sophistiqués, il s'agit par exemple de satellites métérologiques, qui sont capables de donner des informations sur les nuages [2] (contenu en eau liquide, classification...),sur l'atmosphère(température, humidité...), sur la surface terrestre (force et direction du vent, incendies, neige...).

En plus de ces satellites, la France est organisée de telle sorte que ses stations météo et leur portée de mesure recouvrent l'intégralité de la surface du territoire. Ce réseau de station est appelé "Aramis".

2.3 Modélisation

Les modélisations actuelles sont basées sur des équations [3], notamment :
— Équation du mouvement

1. http ://www.meteofrance.fr/nous-connaitre/decouvrir-la-meteorologie/
2. http ://www.meteofrance.fr/prevoir-le-temps/observer-le-temps/moyens/
3. https ://www.meteocontact.fr/pour-aller-plus-loin/les-modeles-meteo

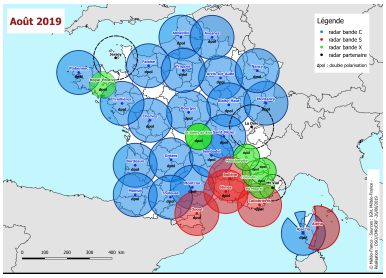


FIGURE 2.1 – http ://www.meteofrance.fr/prevoir-le-temps/observer-le-temps/moyens/

— Équation de conservation de la masse totale
— Équation des gazs parfait
— Équation de la thermodynamique

En Europe, le modèle utilisé est CEPMMT (Centre européen pour les prévisions météorologiques à moyen terme). Il permet de prédire la météo jusqu'à 10 jours. En France, un modèle plus précis est aussi utilisé mais son spectre de prédiction est moins grand, il ne peut prédire jusqu'à 36h, il s'agit du modèle AROME.

Chapitre 3

Présentation du jeu de données choisi ou construit

3.1 Les données mesurées

Le jeu de donnée présente des données météo récupérées par une station, située dans le quartier Saint-Cyprien de Toulouse [1]. C'est le 27ème jeu de données d'une série de donnée météo récupérées à travers la métropole de Toulouse. Les données sont composées des 15 attributs suivants :

- id : identifiant de la station météo
- humidité : Valeur d'humidité en (%)
- direction du vecteur de vent max : direction de la rafale de vent max. Indique l'angle par rapport au Nord, gradué de 0 à 15, avec $0 = 0°$ et $15 = 337,5°$
- pluie intensite max : intensité maximale de pluie sur une minute
- pression : pression atmosphérique
- direction du vecteur vent moyen : Indique l'angle entre la direction moyenne dont vient le vent et le nord.
- type de station : ici nous avons un seul type de station, il s'agit de Integrated Sensor Suite (ISS) regroupant un capteur de : Température, Humidité, Anémomètre, Pluviomètre, Capteur de Pression.
- pluie : quantité de précipitation (en mm)
- direction du vecteur de rafale de vent max : direction du vecteur de vent max dans les 15 dernières minutes (en °)
- force moyenne du vecteur vent : (en km/h)
- force rafale max : rafale maximale dans les 15 dernières minutes
- temperature : température moyenne des 15 dernières minutes (en °)
- heure de paris : heure de paris à laquelle la mesure est prise (a/m/j h :m :s)
- heure utc : heure au format utc, au moment où la mesure est prise.

Les mesures sont prises toutes les 15 minutes à partir du 2019-06-26 00 :00 :00 jusqu'au 2020-06-03 13 :15 :00. Le jeu présente 32 947 mesures.

3.2 Description du jeu de donnée

Par analyse du jeu de donnée, on se rend compte qu'il ne manque aucune donnée pour aucun des attributs. Étudions maintenant les données mesurées de quelques attributs en détail :

1. `https://www.data.gouv.fr/fr/datasets/station-meteo-saint-cyprien`

```
count    32947.0
mean         0.0
std          0.0
min          0.0
25%          0.0
50%          0.0
75%          0.0
max          0.0
Name: pluie, dtype: float64
```

Pour la pluie, on remarque notamment que toutes les valeurs sont à 0, donc cet attribut n'est pas important pour étudier l'évolution de la température, il n'apportera pas d'informations.

```
count    32947.000000
mean        15.942156
std          7.605225
min        -50.000000
25%         10.500000
50%         15.000000
75%         20.900000
max         40.600000
Name: temperature, dtype: float64
```

Après étude de la variable Température, on se rend compte que certaines valeurs sont aberrantes, par exemple la valeur minimale mesurée est -50° ce qui est une température abbérante pour la ville de Toulouse.

```
count    32947.000000
mean        56.451634
std         14.184005
min          0.000000
25%         48.000000
50%         60.000000
75%         67.000000
max         82.000000
Name: humidite, dtype: float64
```

La mesure de l'humidité est une variable qui semble correcte, ses valeurs sont cohérentes. Cette variable pourra être traitée sans problème.

Chapitre 4

Explication de l'implémentation

Pour l'implémentation, j'ai choisi le langage python et l'utilisation de la librairie sklearn, celle-ci proposant l'algorithme de SVR, les paramètres étant à ajuster selon nos données. J'utilise aussi les libraires matplotlib et numpy, celles-ci étant souvent utilisées, je ne les présenterai pas ici.

4.1 Pré-traitement des données

Premièrement, après avoir fait l'étude des variables (min,max,moyenne...), j'ai retiré les variables considérées comme inutiles du jeux de données, celles-ci sont par exemple représentées en deux formats (heure, direction vecteur vent max), ou encore comme montré dans description des données, certaines sont vides . Il s'agit des suivantes :

— data
— id
— type de station
— heure de paris
— direction du vecteur de vent max

Enfin, mes données n'étaient pas triées dans l'ordre chronologique, la date n'était pas dans un format adéquat. Il y avait aussi beaucoup de données, les intervalles de mesures étant de 15 mins, il fallait réduire mon nombre de données et aussi éliminer les valeurs abérantes de certaines variables (notamment la température), j'ai fait le choix de regrouper les données par jour, avec comme fonction d'aggrégation la mediane pour ne pas être sensible aux valeurs aberrantes. Il fallait donc prétraiter ces données.

```
    def Pretraitement(data,dateDebut,dateFin) :
2   data['heure_utc'] = pd.to_datetime(data['heure_utc']) #on met les donnees au
    format de date
    data.sort_values(by=['heure_utc'], inplace=True, ascending=True) #on trie
    nos donnees selon les dates, du plus vieux au plus recent
4   data = data.set_index('heure_utc')
    data = data.resample('D').median() #groupement par jour en faisant la
    mediane des valeurs
6   data['temperatureDecallee'] = data.temperature.shift(-1) #decalage des
    donnees de 1 de telle sorte que pour les donnees du jour J on ait une colonne
    temperatureDecallee qui correspond a la temperature du jour J+1
    data.dropna(inplace=True) #on retire le dernier jour de mesure car on ne
    connait pas la temperature du lendemain
8   data = data.loc[dateDebut:dateFin]
    y = pd.DataFrame(data['temperatureDecallee'])
10  df = data.drop(['temperatureDecallee'],axis = 1)
    x = df
12  return x,y
```

En sortie de cette fonction j'obtiens des données triées, avec une nouvelle variable de température du lendemain, et surtout une quantité de données réduite à 343 mesures au lieu des 32 947 initiales. Cela permet alors d'utiliser la fonction SVR de sklearn qui supporte des jeux de données allant jusqu'à 10 000 mesures.

4.1.1 Choix des variables

A ce point du code, les variables encore dans le jeux de données sont au nombre de 8, notamment avec beaucoup de variables dérivées du vent (force maximal, force moyenne etc). Après séparation en données d'entrainement et données de test, j'ai choisi de classer les variables par rapport à leur importance dans le modèle, je choisis ici un modèle très simple sans paramètres.

```
estimator = SVR(kernel="linear")
selector = RFE(estimator)
selector = selector.fit(xtrain, ytrain)
rank = selector.ranking_ #on a alors les features les plus importants pour le modele
indice = np.ravel(np.argwhere(rank == 1 ))
```

J'obtiens donc une sélection de 4 variables qui sont : l'humidité , la pression, la direction du vecteur de rafale de vent max, la température.

4.2 Modèle selon plusieurs variables du jour J

Pour ce modèle je vais tenter de prédire la valeur de la température du jour J+1 en fonction des variables humidité, pression, direction du vecteur de vent max, et température du jour J. Tout d'abord comme il s'agit de variables ayant des échelles totalement différentes, il faut les normaliser.

```
sc_X = StandardScaler()
x = sc_X.fit_transform(x)
```

Ensuite il s'agit simplement d'appliquer le modèle avec les paramètres obtenus par le GridSearch. On note que gamma n'est pas utilisé pour le kernel Linéaire.

```
regressorGridSearch=SVR(C= 9.0, epsilon = 0.1, gamma = 1e-09,kernel='linear') #mise en place du modele
regressorGridSearch.fit(xtrain,ytrain) #adaptation du modele sur nos donnees d'entrainement
pred=regressorGridSearch.predict(xtest) #temperature predite sur les donnees de test
```

Nous disposons de plusieurs indicateurs pour évaluer la qualité du modèle.

```
print('nombre de donnee d''entrainement : ',xtrain.shape)
print('R2 = ',regressorGridSearch.score(xtest,ytest))
print('RMSE = ', np.sqrt(mean_squared_error(ytest, pred)),'C')
```

Le premier indicateur est de calculer le coefficient de régression R^2. Le meilleur coefficient possible est 1, synonyme d'un modèle parfait, au contraire si le modèle ne tient pas compte des données en entrée on aura un coefficient de 0. Le deuxième indicateur est le Root Mean Square Error, qui nous donne l'erreur moyenne en degrée de notre modèle. On cherche donc à le minimiser.

4.3 Modèle selon 1 variable prise sur plusieurs jours

Pour ce second modèle je vais tenter de prédire la valeur de la température du jour J+1 en fonction des températures des jours précédents. Pour cela je cherche à trouver le nombre de jours idéal à prendre en compte pour éviter d'ajouter trop de données qui nuirait au modèle. Je réalise une fonction qui permet de concatener dans un même vecteur les valeurs des températures de X jours précedents et la température du jour J.

```
def PriseEnCompteDeXJours(x,y,nbjours) :
    y = np.ravel(y.iloc[nbjours:,:].values) #on change nos dataFrames en tableaux de valeurs pour qu'elles soient manipulees par la fonction SVR()
    x = x.iloc[:,0:].values
    xnew = np.zeros((x.shape[0]-nbjours,x.shape[1]*(nbjours+1)))
    for i in range(nbjours,x.shape[0]) :
        xnew[i-nbjours] = np.ravel(x[i-nbjours:i+1,:])
    return xnew, y
```

Je réalise ensuite une boucle pour des jours allant de 1 à 7.

```
print('Prise en compte des temperatures')
for nbjours in range(1,8) :
    X, y = Pretraitement(data,"2019-06-26","2020-06-03")
    X = pd.DataFrame(X['temperature'])
    Xnew, ynew = PriseEnCompteDeXJours(X,y,nbjours)
    indice = round(0.8*len(Xnew))
    xtrain = Xnew[0:indice,:]
    ytrain = ynew[0:indice]
    xtest = Xnew[indice:,:]
    ytest = ynew[indice:]
    regressor=SVR(kernel='linear')
    regressor.fit(xtrain,ytrain)
    pred=regressor.predict(xtest)
    print('J-',nbjours,'len(xtrain): ',xtrain.shape[0],'RMSE = ', np.sqrt(mean_squared_error(ytest, pred)),'C')
```

Le modèle qui donne le meilleur RMSE sera celui retenu, on pourra l'améliorer en réalisant une GridSearch sur les hyper-paramètres comme réalisé avec le modèle précédent.

Chapitre 5

Choix des hyper-paramètres

La fonction SVR() proposé par Sklearn permet de moduler le modèle selon divers paramètres, nous manipulerons seulement les suivants :

— C
— Kernel
— ϵ
— γ

Afin de choisir les hyper-paramètres idéaux, pour améliorer la précision de mon modèle, j'ai choisi d'utiliser la fonction GridSearchCV de Sklearn. Cette méthode me permet de définir des intervalles de recherche pour chaque paramètres, la méthode va alors tester les combinaisons possibles, et les valider par la méthode de Cross Validation. Voici les intervalles que j'ai défini pour mes paramètres. On ne peut pas être trop précis dans les intervalles car ce calcul demande beaucoup de temps.

```
C = np.arange(1,10,0.5)
gamma = np.arange(1e-9, 1e-7,1e-8)
epsilon = np.arange(0.1,0.3,0.1)
```

5.1 Modèle selon plusieurs variables du jour J

Pour le premier modèle construit, le GridSearch nous donne les résultats suivants :

```
{'C': 1.5, 'epsilon': 0.2, 'gamma': 1e-09, 'kernel': 'linear'}
```

5.2 Modèle selon 1 variable prise sur plusieurs jours

Pour le modèle prenant en compte le nombre de jours optimal, pour prédire la température du lendemain, on utilise les paramètres suivants.

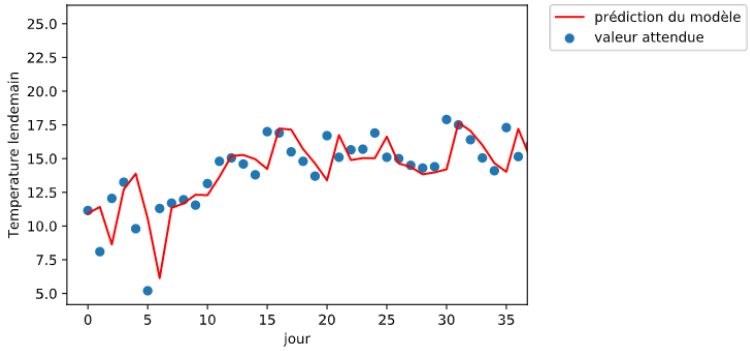
```
{'C': 1.0, 'epsilon': 0.1, 'gamma': 1e-09, 'kernel': 'linear'}
```

Chapitre 6

Résultats et interprétations

6.1 Modèle selon plusieurs variables du jour J

Nous cherchions donc la température pour le jour J+1. En paramétrant le modèle avec les paramètres trouvés par la GridSearch, on obtient la figure suivante :



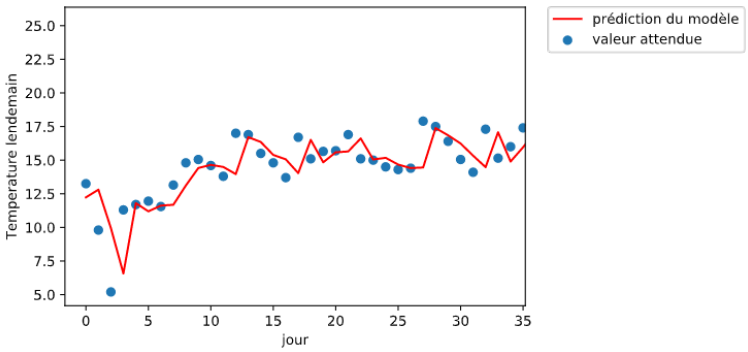
Les indicateurs pour évaluer le modèle sont les suivants :
— nombre de donnée d'entrainement = 272
— $R^2 = 0.7302388105171913$
— RMSE = 2.078830438831827
Grâce à ce modèle on peut prédire la valeur de la température du lendemain avec plus ou moins 2.07 degrée Celsius.

6.2 Modèle selon 1 variable prise sur plusieurs jours

Pour ce modèle, on cherchait à prédire la valeur de la température à J+1 en prenant en compte les températures des X jours précédents. Voici les RMSE des différents modèles selon le nombre de jour pris en compte :

```
Prise en compte des températures
J- 1 len(xtrain):  274 RMSE =  2.027082188874238 °C
J- 2 len(xtrain):  273 RMSE =  1.998807588228589 °C
J- 3 len(xtrain):  272 RMSE =  1.9912563101063014 °C
J- 4 len(xtrain):  271 RMSE =  2.127995200399387 °C
J- 5 len(xtrain):  270 RMSE =  2.134428562923103 °C
J- 6 len(xtrain):  270 RMSE =  2.141932224246946 °C
J- 7 len(xtrain):  269 RMSE =  2.1460264010832155 °C
```

On remarque alors le modèle est meilleur lorsqu'on prend en compte les trois jours précédents le jour J. C'est donc avec ce jeux de donnée reconstruit que le modèle va être établit. Voici la figure obtenue, et les valeurs des indicateurs.



— nombre de donnée d'entrainement = 272
— $R^2 = 0.7309668174031742$
— RMSE = 1.9912563101063014

Ce modèle permet donc de prédire la température du lendemain avec une erreur de plus ou moins 1.99 degrée Celsius. C'est jusque-là le meilleur modèle obtenu avec ce jeu de donnée.

6.3 Remarques

Pour améliorer le modèle j'ai tenté de combiner les deux modèles énoncés précédemment, soit ajouter les données de pression, humidité, vent, température du jour J et les données de température des jours précédents. Mais ajouter des attributs nuisait au modèle, et on obtenait finalement une prédiction moins bonne, même avec une nouvelle selection des attributs selon leur importance dans le modèle. Ces deux modèles présentent donc les meilleurs résultats que j'ai pu obtenir.