Rapport SY19 TP04 - Support Vector Machine

Thomas ABASSI & Joan MOESCH

January 12, 2012

Le but de ce TP est la prise en main de la méthode de classification supervisée Support Vector Machine (SVM). Afin d'appréhender cette méthode, nous évaluerons l'influence de ses paramètres à travers des outils déjà implémentés au sein du package 'e1071'.
Nous verrons d'abord l'influence du paramètre C de pénalisation. Nous évaluerons ensuite l'importance du type de noyaux pour notre jeu de données ainsi que l'influence des hyper-paramètres liés aux noyaux.

Chaque paramètre sera évalué par rapport à l'erreur de généralisation, c'est à dire celle de mauvaise classification avec les données tests comme critère de mesure de performance. Cette erreur correspond a la somme des erreurs empiriques de type α et β (fausse alarme ou absence de diagnostic). Nous récupérons ces valeurs de la matrice de confusion prédiction / classe réelle :

$$\begin{pmatrix} predit/réel & benign & malignant \\ benign & M_{11} & M_{12} \\ malignant & M_{21} & M_{22} \end{pmatrix}$$

Nous avons considéré pour le calcul de cette erreur des couts de type (0,1), ainsi les décision seront pondérées des couts $C_{11} = C_{22} = 0$ et $C_{12} = C_{21} = 1$.
L'erreur de classification sera donc simplement donnée par : $E = M_{12} + M_{21}$.

Chaque individu des données d'apprentissage et des données test est de dimension 9 (9 variables descriptives), nous n'afficherons donc pas les frontières de décisions directement mais l'évolution de l'erreur de classification en fonction des différents paramètres étudiés.

1 La fonction SVM tune

Avant toute chose, nous utilisons la fonction SVM par défaut pour créer un modèle de décision par rapport aux données d'apprentissage. Nous appliquons ce modèle à la 2ème partie des données, les données tests, afin de comparer les résultats avec les classes réelles. Ceci est réalisé avec le code suivant :

$model < -svm(databctrain, classesbctrain)$
$pred < -predict(model, databctest)$
$table(pred, t(classesbctest))$

Ceci nous donne les valeurs suivantes quant a la classification :

$$\begin{pmatrix} predit/réel & benign & malignant \\ benign & 224 & 1 \\ malignant & 5 & 69 \end{pmatrix}$$

Ceci nous donne une erreur de $E = 6$ éléments mal classés, soit un taux de $\frac{6}{299} = 0.02$.

Il existe une fonction SVM appelé tune permettant de donner les paramètre de la méthode SVM connaissant les classes des données tests réelles. Une fois ces paramètres calculés, nous déterminons à nouveau le modèle avec nos données d'apprentissage puis testons ce modèle avec

les données tests.

En comparant les nouveaux résultats avec les vraies classes, nous retrouvons une classification identique. Les paramètre optimaux sont donc ceux par défauts dans notre cas. Afin de mieux comprendre comment ces paramètres interfèrent avec le modèle SVM, nous les étudions un à un dans les parties suivantes.

2 Influence du paramètre C

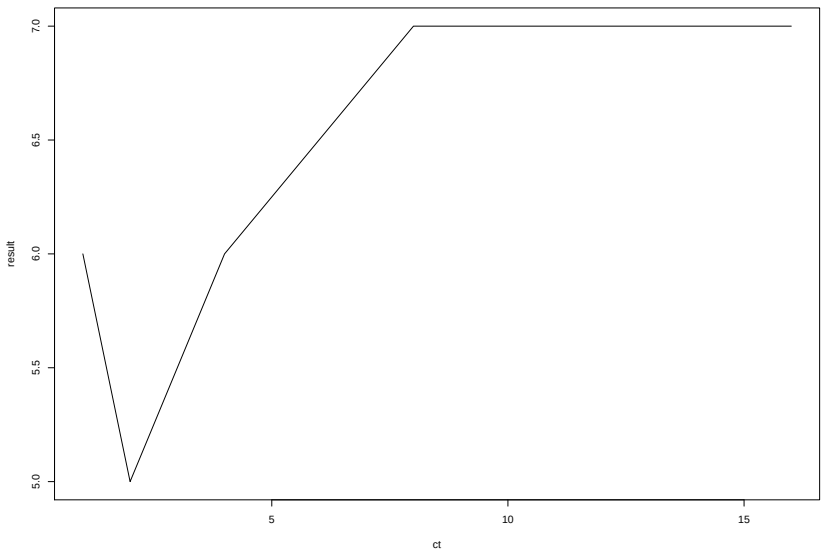
Nous implémentons une fonction R faisant varier le paramètre de pénalisation et calculons à chaque fois l'erreur obtenue :

$paramC <- function()$
$result = rep(0, 5);$
$ct = c(1, 2, 4, 8, 16);$

for(i in 1:5)
$model <- svm(databctrain, classesbctrain, gamma = a.best.gamma, cost = ct[i]);$
$pred <- predict(model, databctest);$
$t <- table(pred, t(classesbctest));$
$result[i] = t[1, 2] + t[2, 1];$

$plot(ct, result, type = "line");$

Pour un noyau donné (Gaussien par défaut), voici l'évolution de l'erreur de classification en fonction du paramètre $Cost$ de SVM :



Le graphique indique donc un choix du paramètre $Cost = 2$. Ceci confirme la valeur indiquée par la fonction de réglage Tune, donnant comme paramètre C à posteriori 2 également.

3 Influence du type de noyau

Le noyau est l'outil nécessaire à la projection des données dans un espace de grande dimension pour séparer de manière certaine ces données (il faut être en dimension n pour séparer $n+1$ données). Il existe différents noyaux possibles. La fonction SVM implémentée dans le package 'e1071' permet l'utilisation de différents noyaux dans la liste suivante : linear, polynomial, radial et sigmoid.

Nous écrivons une méthode similaire à la précédente en modifiant désormais le type de noyau du modèle et non plus le paramètre C. Les autres paramètres utilisés sont ceux données par la fonction Tune de SVM.

$paramN <- function()$
    $result = rep(0, 4)$;
    $noyau = c("linear", "polynomial", "radial", "sigmoid")$;

    for(i in 1:8)
        $model <- svm(databctrain, classesbctrain, kernel = noyau[i]))$;
        $pred <- predict(model, databctest)$;
        $t <- table(pred, t(classesbctest))$;
        $result[i] = t[1, 2] + t[2, 1]$;

Nous obtenons les erreurs de classifications suivantes :
    linear : 4
    polynomial : 5
    radial : 5
    sigmoid : 7

Le meilleur noyau pour notre jeu de données est donc le noyau de type linéaire, c'est à dire : u'* v. Le taux d'erreur est ici de $\frac{4}{299} = 0.133$.

4 Influence de quelques hyperparamètres du noyau utilisé

4.1 Largeur de bande d'un noyau Gaussien

C'est l'influence du paramètre γ pour un noyau de type radial : $e^{-\gamma|u-v|^2}$ qui est désormais à l'étude. Le terme γ est le paramètre désignant la largeur de bande d'un noyau de type Gaussien. Nous écrivons la fonction calculant l'erreur de classification en fonction de ce paramètre dans une plage de valeur de $\gamma = 0.1$ à $\gamma = 16$. Ceci nous donne :

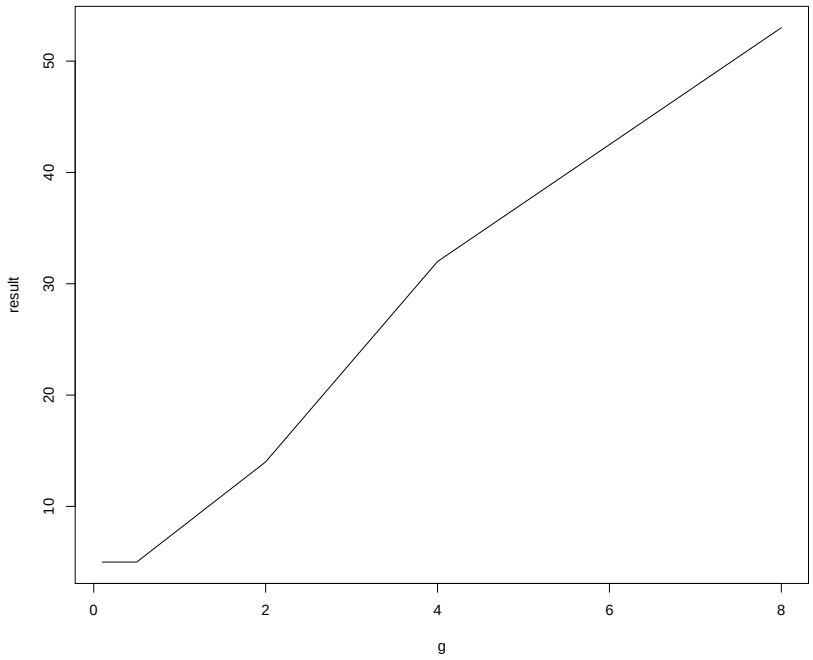
$paramG <- function()$
    $result = rep(0, 8)$;
    $g = c(0.1, 0.2, 0.5, 1, 2, 4, 8, 16)$;

    for(i in 1:8)
        $model <- svm(databctrain, classesbctrain, Gamma = g[i], Cost = a.best.Cost)$;
        $pred <- predict(model, databctest)$;
        $t <- table(pred, t(classesbctest))$;

$result[i] = t[1,2] + t[2,1];$

$plot(d, result, type = "line");$

A noter que le type de noyau n'est pas défini car le type de noyau est Gaussien (radial). Le graphique obtenu est le suivant :



Le graphique nous indique donc une erreur de classification minimale pour des largeurs de bande $\gamma = [0.1, 0.2, 0.5]$ d'un noyau gaussien avec notre jeu de données. On remarque que la encore, notre étude converge avec les valeurs données par la fonction tune pour la paramètrisation du SVM.

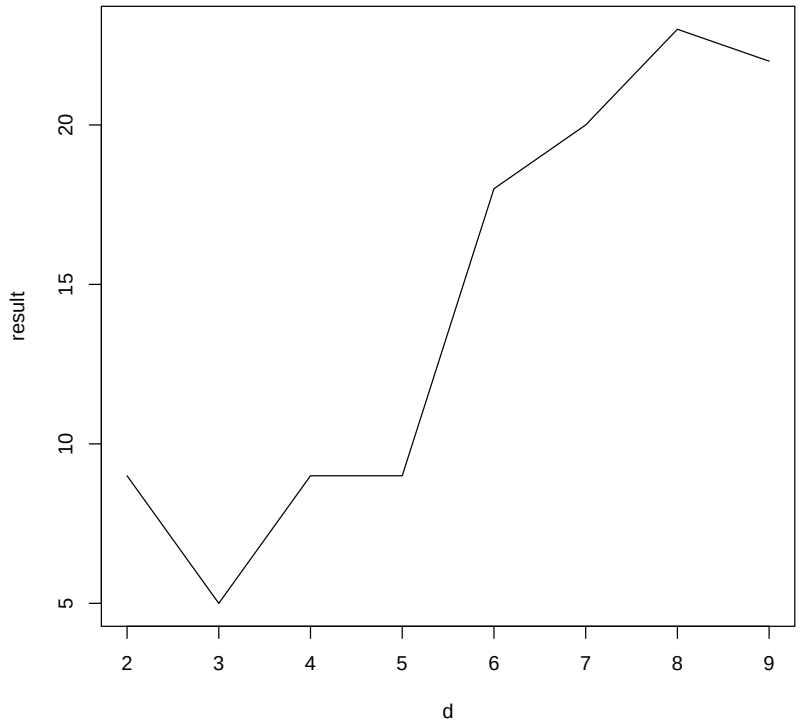
4.2 Degré d'un noyau polynomial

Nous étudions maintenant l'influence du degré d'un noyau de type polynomial. Avec les paramètres Gamma et Cost trouvés par la fonction tune et coef0 = 0, nous faisons varier le degré du noyau entre 2 et 9 par une fonction analogue à celles décrites précédemment :

$paramD < -function()$
$result = rep(0, 8);$
$d = c(2, 3, 4, 5, 6, 7, 8, 9);$

for(i in 1:5)
$model < -svm(databctrain, classesbctrain, kernel = "polynomial", degre = d[i]);$
$pred < -predict(model, databctest);$
$t < -table(pred, t(classesbctest));$
$result[i] = t[1,2] + t[2,1];$

$plot(d, result, type = "line")$;



Le graphique obtenu nous indique donc clairement que le meilleur degré du noyau polynomial dans le cadre de nos données est 3. Ceci coïncide une nouvelle fois avec les valeurs par défaut et celles données par la fonction SVM Tune .

5 Conlcusion

Nous remarquons tout d'abord que nos études ont démontrées la véracité des paramètres optimaux donnés par la fonction du package SVM tune.

Plus généralement, pour la quasi totalité des paramètres étudiés, nous avons remarqué une certaine similitude dans les variations de l'erreur de classification. En effet, la classification est souvent optimale pour des paramètres faibles mais pas minimaux. Par suite, l'augmentation de la valeur des paramètres implique une augmentation de l'erreur de généralisation. Ainsi ces différents paramètres sont des mesures de compléxité du modèle implémenté.

Ceci s'explique par le fait, que pour des paramètres inférieurs aux optimaux, le modèle n'est pas assez performant sur les données : il est en sous-apprentissage. A l'inverse, a mesure que le modèle se complexifie en fonction des paramètres (par exemple le degré d'un noyau polynomial), l'erreur va augmenter aussi. Ceci s'explique par le fait que le modèle est très appliqué à ses données d'apprentissage et donc moins général pour des données tests : il est en sur-apprentissage.
Il est donc important d'être attentif avec ses paramètres car le but de la méthode SVM est

de définir des frontières de décisions qui seront les plus efficientes possibles à la généralisation, attention aux modèle trop spécialisés aux données d'apprentissage.