SY09 Printemps 2019
TP 10 — Régression logistique

1 Implémentation

1.1 Régression logistique binaire

On souhaite implémenter le modèle logistique *binaire*. On complétera tout d'abord deux fonctions, l'une permettant de faire l'apprentissage du modèle (on utilisera l'algorithme de Newton-Raphson présenté en cours), l'autre permettant de faire le classement d'un ensemble de données.

Apprentissage. La fonction `log.app`, permettant d'apprendre les paramètres du modèles, prendra comme arguments d'entrée le tableau de données `Xapp`, le vecteur `zapp` des étiquettes associées, ainsi qu'une variable binaire `intr` indiquant s'il faut ou non ajouter une ordonnée à l'origine (*intercept*) à la matrice d'exemples et un scalaire `epsi` correspondant au seuil ε en-deçà duquel on considère que l'algorithme d'apprentissage a convergé.

Elle devra retourner la matrice `beta` correspondant à l'estimateur du maximum de vraisemblance $\widehat{\beta}$ des paramètres, de dimensions $p \times 1$ (ou $(p+1) \times 1$ si une ordonnée à l'origine a été ajoutée), le nombre `niter` d'itérations effectuées par l'algorithme de Newton-Raphson, et la valeur `logL` de la vraisemblance à l'optimum.

On pourra utiliser comme matrice de paramètres initiale $\beta^{(0)} = (0, \ldots, 0)^T$. On testera la convergence en comparant la norme de la différence entre deux estimations successives $\beta^{(q)}$ et $\beta^{(q+1)}$ au seuil ε (on pourra choisir $\varepsilon = 1e-5$).

Classement. La fonction `log.val`, permettant de classer un ensemble d'individus, prendra comme arguments d'entrée le tableau de données `Xtst` à classer et la matrice `beta` des paramètres déterminés par la fonction `log.app`. Cette fonction fera un test sur les dimensions de la matrice `beta`, pour déterminer si une ordonnée à l'origine doit être ou non ajoutée à la matrice d'exemples `Xtst` à évaluer.

Elle devra retourner une structure contenant la matrice `prob` des probabilités a posteriori estimées et le vecteur des classements associés.

Probabilités *a posteriori*. Ces fonctions pourront s'appuyer sur une fonction `post.prob` calculant les probabilités *a posteriori* à partir d'une matrice de paramètres `beta` et d'un tableau de données `X`.

1.2 Régression logistique quadratique.

Il est possible de généraliser le modèle de régression logistique de manière très simple. La stratégie consiste à transformer les données dans un espace plus complexe, dans lequel les classes peuvent être séparées par un hyperplan. La régression logistique est alors effectuée dans cet espace.

Le modèle ainsi défini est donc plus flexible (il permet de construire une frontière de décision plus complexe) ; mais il peut également s'avérer moins robuste que le modèle classique déterminé dans l'espace des caractéristiques initiales, puisqu'il nécessite d'estimer davantage de paramètres.

Par exemple, dans le cas où les individus sont décrits par les variables naturelles X^1, X^2 et X^3, la régression logistique quadratique consiste à apprendre un modèle de régression logistique classique dans l'espace $\mathcal{X}^2 = \{X^1, X^2, X^3, X^1X^2, X^1X^3, X^2X^3, (X^1)^2, (X^2)^2, (X^3)^2\}$, plutôt que dans l'espace $\mathcal{X} = \{X^1, X^2, X^3\}$ constitué des trois variables naturelles.

On calculera les produits des réalisations décrivant les individus d'apprentissage — on utilisera pour cela la fonction `poly` :

```
> poly(as.matrix(donnees.num, degree=2, raw=T)
```

On apprendra ensuite le modèle logistique sur les $p(p+3)/2$ variables ainsi obtenues.

1.3 Vérification des fonctions

La Figure 1 montre les frontières de décision obtenues lorsque la totalité des données `Synth1-40` (voir TP précédents) sont utilisées pour l'apprentissage du modèle.

On pourra utiliser les fonctions `prob.log` et `prob.log2` fournies pour visualiser les courbes de niveau des probabilités *a posteriori* ou les frontières de décision obtenues par régression logistique classique et quadratique. On veillera à ce que l'ordre des termes quadratiques soit bien le même dans le code développé que dans la fonction `prob.log2`.

2 Application

2.1 Test sur données simulées

On souhaite comparer les performances de la régression logistique (linéaire et quadratique) sur les jeux de données simulées `Synth1-1000`, `Synth2-1000` et `Synth3-1000` disponibles sur le site de l'UV. Pour ce faire, on utilisera le même protocole expérimental que précédemment (séparation des données en ensembles d'apprentissage et de test, apprentissage du modèle, et évaluation des performances) que l'on répétera $N = 20$ fois pour chaque jeu de données.

Pour chaque jeu de données, calculer le taux d'erreur (de test) moyen à partir des $N = 20$ estimations disponibles. On pourra s'appuyer sur l'affichage des frontières de décision pour analyser les résultats. Comment peut-on interpréter ces derniers ? Les comparer aux résultats obtenus par les trois modèles d'analyse discriminante précédemment testés.

2.2 Test sur données réelles

On appliquera la régression logistique aux données `spambase`, et on comparera les résultats à ceux obtenus avec les modèles d'analyse discriminante utilisés au TP précédent.

Semble-t-il ici possible d'apprendre un modèle de régression logistique quadratique pour ces données ? Pour quelle raison ?

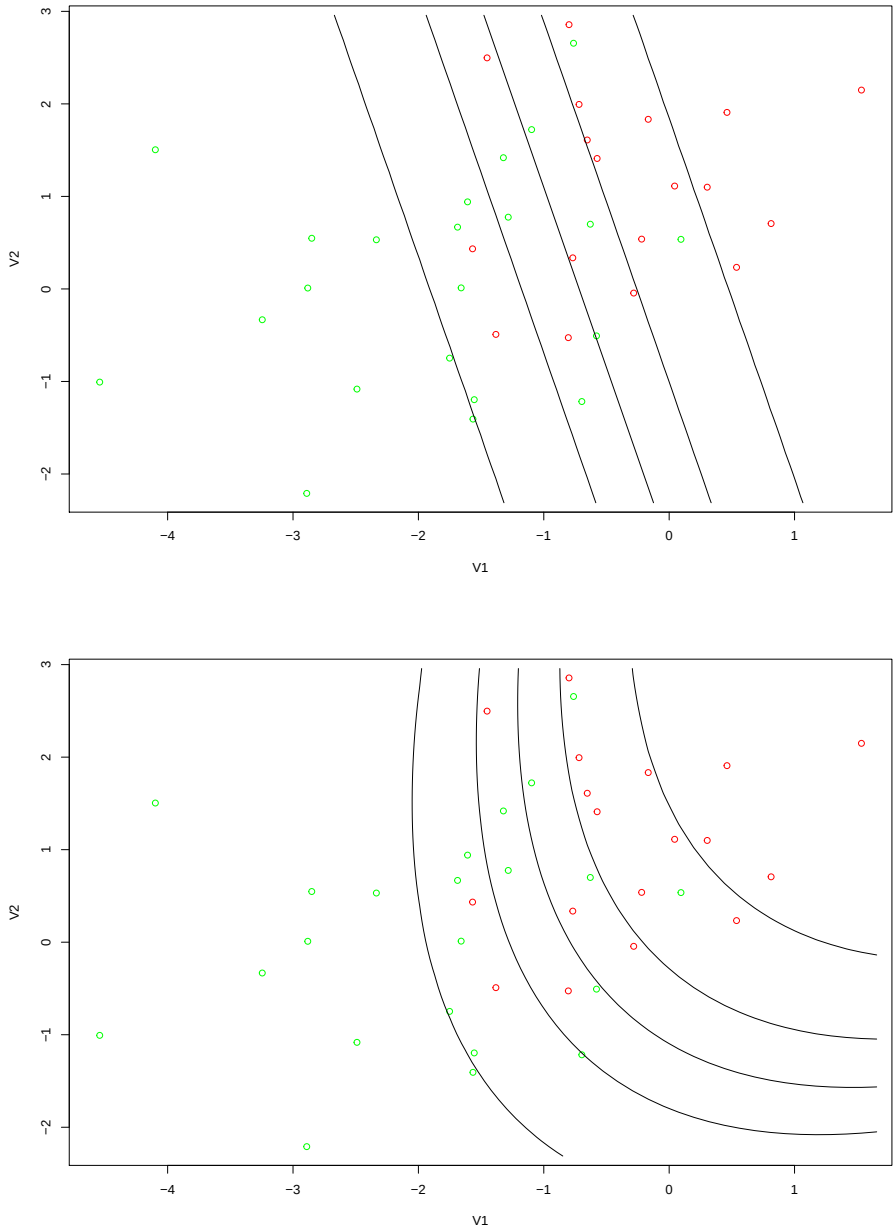

FIGURE 1 – Courbes de niveau des probabilités a posteriori $\widehat{\Pr}(\omega_1|\boldsymbol{x})$ estimées sur les données `Synth1-40` par régression logistique (haut) et régression logistique quadratique (bas).