EXERCICE DE PROGRAMMATION 3 :

Support Vector Machine

GAO BIN

30/12/2016

Introduction

Dans ce TP, on va utiliser le SVM(Support Vector Machine) pour construire un classicateur de spam.

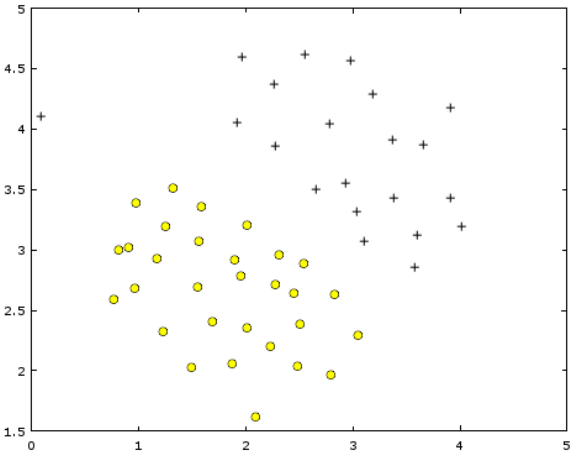
1. Support Vector Machine

1.1 Example Dataset 1

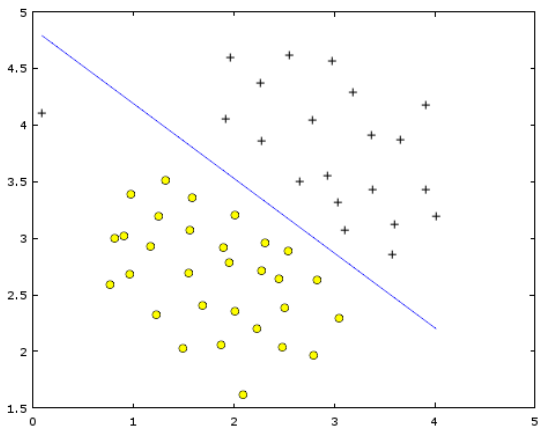
Nous allons commencer par un ensemble de données 2D exemple qui peut être séparé par un limite linéaire.

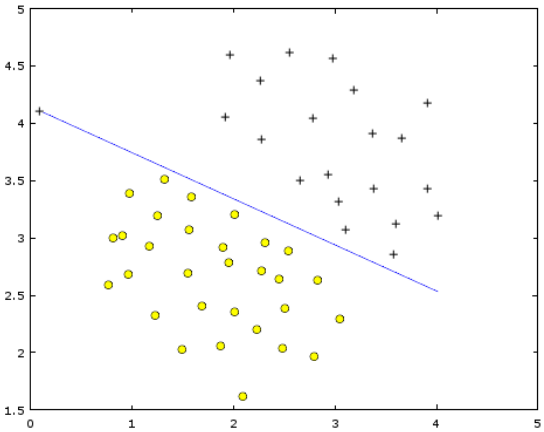
On va essayer d'utiliser différentes valeurs de la paramètre C avec les SVM. De façon informelle, le paramètre C est une valeur positive qui contrôle la pénalité pour les exemples d'entraînement mal classifiés. C joue un rôle similaire à 1/λ (λ est le paramètre de régularisation).

On peut obetir la figure de la Dataset1 :



La figure de «SVM Decision boundary with C=1»(gauche) et «SVM Decision boundary with C=100»(droit) :





Les deux figure ci-dessus avons classifié les datas corrects, mais quand C = 100, on trouve que le SVM maintenant classifié chaque exemple correctement, mais il a une limite de décision qui ne semblent correspondre naturellement aux données (figure 3).

1.2 SVM with Gaussian Kernels

Dans cette partie de l'exercice, on va utiliser les SVM pour effectuer une classification non linéaire.

1.2.1 Gaussian Kernel

Pour trouver des limites de décision non linéaires avec la SVM, nous devons d'abord un noyau gaussien.

La fonction Gaussian Kernel :

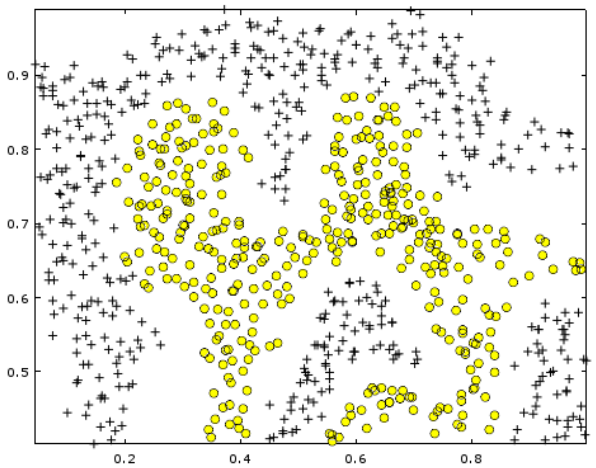
$$K_{gaussian}(x^{(i)}, x^{(j)}) = \exp\left(-\frac{\|x^{(i)} - x^{(j)}\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{k=1}^{n}(x_k^{(i)} - x_k^{(j)})^2}{2\sigma^2}\right)$$

On complète donc le code de la fonction avec Octave:

```
19  x12 = x1 - x2;
20  temp = -x12' * x12 / (2 * sigma * sigma);
21  sim = exp(temp);
```

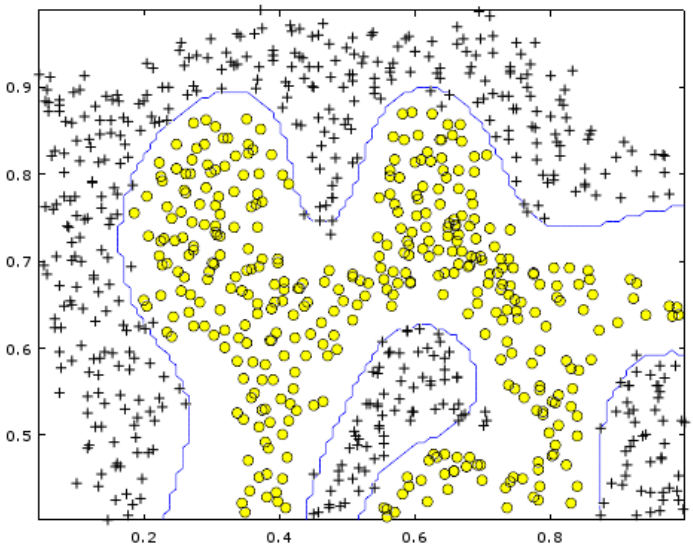
1.2.2/3 Example Dataset2 et Dataset3

On soumet la fonction, on peut obtenir suivant :



Sur cette figure, on peut observer qu'il n'y a pas de limite de décision linéaire qui sépare les exemples positives et négatives de cet ensemble de données. Cependant, par en utilisant le noyau gaussien avec le SVM, vous serez en mesure d'apprendre un non-linéaire décision qui peut raisonnablement bien fonctionner pour l'ensemble de données.

On fait appel au programme， donc on obtient alors la figure :



Dans cette partie, on complète donc le code de la fonction avec Octave:

```
C_array = [0.01;0.03;0.1;0.3;1;3;10;30];
sigma_array = [0.01;0.03;0.1;0.3;1;3;10;30];
error_array = zeros(8,8);
error_min = 10000;

for i = 1:8,
    for j = 1:8,
        model= svmTrain(X, y, C_array(i), @(x1, x2) gaussianKernel(x1, x2, sigma_array(j)));
        predictions = svmPredict(model, Xval);
        error_array(i,j) =  mean(double(predictions ~= yval));
        if(error_array(i,j) < error_min)
            error_min = error_array(i,j);
            C = C_array(i);
            sigma = sigma_array(j);
        end
    end
end
fprintf('The training C    The training sigma    error\n');
for i = 1:8,
    for j = 1:8,
       fprintf('%f   %f   %f\n',C_array(i),sigma_array(j),error_array(i,j));
    end
end
fprintf('%f and %f perform best, the error is %f',C,sigma,error_min);

% ==========================================================================

end
```

On soumet la fonction, on peut obtenir les resultats suivant, il y a un problem s’appele «unknown hggroup property Color» :

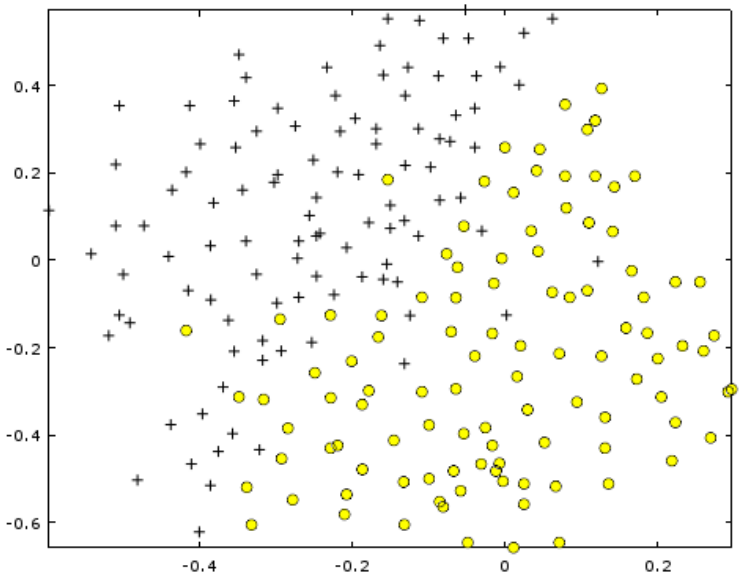
```
Training ...........................................................
...........................................................
...........................................................
...........................................................
...........................................................
...........................................................
..................................... Done!

set: unknown hggroup property Color
error: called from '__contour__' in file E:\Octave-4.0.3\share\octave\4.0.3\m\plot\draw\private\__contour__.m near line 201, column 5
>>
>>
>>
```

Pour résoudre ce problèm, on modifier le code dans visualizeBoundary.m line 21:

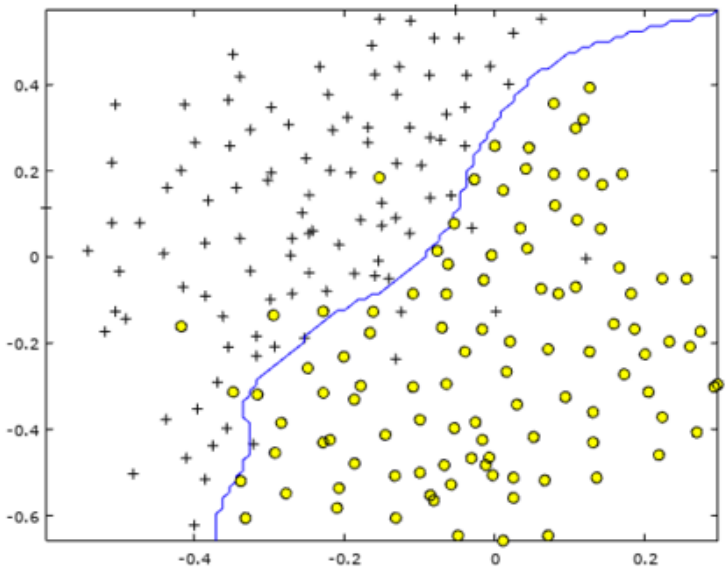
```
19  % Plot the SVM boundary
20  hold on
21  contour(X1, X2, vals, [0 0], 'LineColor', 'b');
22  hold off;
23
24  end
```

Apres on peut obtenir suivant :

```
The training C    The training sigma    error
0.010000    0.010000    0.565000
0.010000    0.030000    0.060000
0.010000    0.100000    0.045000
0.010000    0.300000    0.145000
0.010000    1.000000    0.180000
0.010000    3.000000    0.180000
0.010000    10.000000    0.180000
0.010000    30.000000    0.180000
0.030000    0.010000    0.565000
0.030000    0.030000    0.060000
0.030000    0.100000    0.045000
0.030000    0.300000    0.140000
0.030000    1.000000    0.180000
0.030000    3.000000    0.185000
0.030000    10.000000    0.185000
0.030000    30.000000    0.180000
0.100000    0.010000    0.565000
0.100000    0.030000    0.060000
0.100000    0.100000    0.045000
0.100000    0.300000    0.080000
0.100000    1.000000    0.170000
0.100000    3.000000    0.180000
0.100000    10.000000    0.185000
0.100000    30.000000    0.180000
0.300000    0.010000    0.565000
0.300000    0.030000    0.060000
0.300000    0.100000    0.035000
```



On fait appel au programme， donc on obtient alors la figure :



La figure ci-dessus montre la limite de décision trouvée par le SVM avec Gaussian Kernel. La limite de décision permet de séparer la plupart des facteurs positives et négatives correctement et suit bien les contours de l'ensemble de données.

2- Spam Classification

2.1 Preprocessing Emails

Une méthode souvent utiliser dans le traitement des courriels est "Normaliser" ces valeurs, de sorte que toutes les URL sont traitées de la même façon, tous les numéros sont traités de la même façon.

2.1.1 Vocabulary List

Dans cette partie, notre tâche consiste maintenant à remplir le code dans processEmail.m pour effectuer cette cartographie. Dans le code, on a un string ‘str’ qui est un seul mot a partir du courrier traité. On doit chercher le mot dans le vocabulaire liste ‘vocabList’ et trouver si le mot existe dans la liste de vocabulaire.

On complète donc le code de la fonction dans processEmail.m:

```
100     vocab_length = length(vocabList);
101     for i = 1:vocab_length,
102         if(strcmp(str,vocabList(i)) == 1)
103             word_indices = [word_indices;i];
104         end
105     end
106
```

On soumet la fonction, on peut obtenir les deux resultats suivant :

```
anyon know how much it cost to host a web portal well it depend on how mani
visitor you re expect thi can be anywher from less than number buck a month
to a coupl of dollarnumb you should checkout httpaddr or perhap amazon ecnumb
if your run someth big to unsubscrib yourself from thi mail list send an
email to emailaddr
```

```
=========================
Word Indices:
 86 916 794 1077 883 370 1699 790 1822 1831 883 431 1171 794 1002 1893 1364 592 1676 238 162 89 688 945 1663 1120 1062 1699 375
1162 479 1893 1510 799 1182 1237 810 1895 1440 1547 181 1699 1758 1896 688 1676 992 961 1477 71 530 1699 531

Program paused. Press enter to continue.
```

2.2 Extracting Features from Emails

Dans cette partie, on va aller maintenant implémenter l'extraction de fonctionnalités qui convertit chaque un vecteur dans R.

On complète donc le code de la fonction dans emailFeature.m:

```
51  k = length(word_indices);
52 for i = 1:k,
53     if(x(word_indices(i)) == 0)
54         x(word_indices(i)) = x(word_indices(i)) + 1;
55     end
56  end
57
```

On soumet la fonction, on peut obtenir suivant :

```
=========================
Length of feature vector: 1899
Number of non-zero entries: 45
Program paused. Press enter to continue.
```

2.3 Training SVM for Spam Classification

Quand la formation terminée, on peut voir le “Training Accuracy“ est 99,825% et “Test Accuracy“ est 97,7%

```
Training .....................................................................
.....................................................................
.....................................................................
.......................................... Done!

Training Accuracy: 99.825000

Evaluating the trained Linear SVM on a test set ...
Test Accuracy: 97.700000
```

2.4 Top Predictors for Spam

```
Top predictors of spam:
 our              (0.498733)
 click            (0.463759)
 remov            (0.418204)
 guarante         (0.384007)
 visit            (0.366182)
 basenumb         (0.344383)
 dollar           (0.327274)
 will             (0.271563)
 price            (0.265417)
 pleas            (0.261157)
 most             (0.253882)
 lo               (0.251805)
 nbsp             (0.249211)
 ga               (0.244540)
 hour             (0.238945)
```