Eric Marchessault
111 156 117

Techniques avancées en intelligence artificielle
IFT-4102

Projet partiel

Travail présenté à
Brahim Chaib-draa

Département d'informatique et de génie logiciel
Université Laval
Hiver 2018

Confusion Matrix

Une matrice de confusion ou “Confusion Matrix”, se trouve à être le résultat de nos données d'entraînement.
Pour expliquer le concept de matrice de confusion, nous allons prendre l’exemple de la page Wikipédia.

		Actual class	
		Cat	**Non-cat**
Predicted class	**Cat**	5 True Positives	2 False Positives
	Non-cat	3 False Negatives	17 True Negatives

[1]

Dans cette exemple, nous avions un algorithme qui tentait d'identifier des photo de chat d’un chien. La matrice de confusion contient 4 types de données.

- True positives
- False positives
- False negatives
- True negatives

Les données “true positives” sont les photos qui ont bien été identifié comme étant des chats.
Les données “false positives” sont les photos qui on été identifié comme étant des chat alors quel étaient des photo de chien.
Les données “false negatives” sont les photo qui était des chats, mais qui on été identifié comme étant des chiens.
Les données “true negatives” sont les photo qui n’était pas des chats et qui on été bien identifié comme photo de chien.

Le résultat de cette matrice nous donne des informations sur la précision de notre algorithme à savoir s’il est capable de bien identifier nos photos de chats

Précision

La précision correspond au nombre de données sélectionné selon celle qui sont pertinentes.
En reprenant l’exemple plus haut, nous avions un total de 8 chat et 19 chiens.
Afin de calculer la précision nous avons besoin de 2 données. Les “true positive” et les “false positive”.

$$\frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

[1] https://en.wikipedia.org/wiki/Confusion_matrix

Ce calcule nous permet de savoir ratio des photos de chat bien identifié sur le nombre de photo de chat sélectionné par notre algorithme. Dans notre cas, la précision était de 5 / 7.

Rappel

Le rappel ou “recall” correspond au nombre de données pertinent selon les données sélectionné. Dans l’exemple des photos de chat et chiens, nous avions un total de 8 chat et 19 chiens. L’algorithme a identifié correctement que 5 chat et rejeté 3. Ce qui correspond à notre “true positive” et notre “false negative”

$$\frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

De cette façon, il nous est possible de savoir le ratio de photo de chat bien identifié sur le nombre de photo de chat que nous avions réellement. Dans notre cas , le rappel était de 5 / 8

https://stackoverflow.com/questions/31324218/scikit-learn-how-to-obtain-true-positive-true-negative-false-positive-and-fal

K-nearest neighbor

La métrique de distance choisi a été la distance euclidienne. Elle a été choisi, car il est possible de combiner plusieurs valeurs et d'obtenir un distance.

pseudo-code

```
data = array of data
k_nearest_neighbor(data) {
   dist_index = []
   labels = []

   for i in range(len(data)){
         dist = euclid( data - train[ i ] )
         dist_index.append([dist, i])
   }

   dist_index = sorted(dist_index)

  for i in range(0, k) {
         labels.append(train_label[dist_index[i]])
   }

   return most_common_item from labels
}
```

Les données de la matrice de confusion sont insérés dans des listes. Leur index corresponds à au label qui leur est associé

Exemple :
conversion_labels = {'Iris-setosa': 0, 'Iris-versicolor' : 1, 'Iris-virginica' : 2}
conversion_labels = {'republican' : 0, 'democrat' : 1}

Iris dataset
avec 0.5 des data en test

```
accuracy:  [ 1.          0.94666667  0.94666667]
confusion matrix :
[[ 20.   0.   0.]
 [  0.  26.   1.]
 [  0.   3.  25.]]
Precision :  [ 1.          0.89655172  0.96153846]
Recall :  [ 1.          0.96296296  0.89285714]
```

Congressional dataset
avec 0.5 des data en test

accuracy: [0.92201835 0.92201835]
confusion matrix :
[[79.   9.]
 [8. 122.]]
Precision : [0.90804598 0.93129771]
Recall : [0.89772727 0.93846154]

monks 1

accuracy: [0.84259259 0.84259259]
confusion matrix :
[[184.   32.]
 [36. 180.]]
Precision : [0.83636364 0.8490566]
Recall : [0.85185185 0.83333333]

monks 2

accuracy: [0.69907407 0.69907407]
confusion matrix :
[[232.   58.]
 [72.   70.]]
Precision : [0.76315789 0.546875]
Recall : [0.8   0.49295775]

monks 3

accuracy: [0.89351852 0.89351852]
confusion matrix :
[[181.   23.]
 [23. 205.]]
Precision : [0.8872549   0.89912281]
Recall : [0.8872549   0.89912281]

Bayes Naïf

```
for each different label {
        means = data.means()
        variances = data.variances()
}

for each different label {
        for each value in train {
                part_1 = 1/ (np.sqrt(2 * np.pi) * variances[label][i])
                part_2 = (np.power((exemple[i] - means[label][i]), 2) * -1)/ (2
*np.power(variances[label][i],2))
                result =  part_1 * np.exp(part_2)
                probabilities[label] *= result
        }
}
return probabilities
```

Iris dataset
avec 0.5 des data en test

```
accuracy:  [ 0.83739837  0.6097561   0.77235772]
confusion matrix :
[[ 20.  20.   0.]
 [  0.  27.   0.]
 [  0.  28.  28.]]
Precision :  [ 1.          0.36  1.  ]
Recall :  [ 0.5  1.   0.5]
```

Congressional dataset
avec 0.5 des data en test

```
accuracy:  [ 0.7124183  0.7124183]
confusion matrix :
[[  88.   88.]
 [   0.  130.]]
Precision :  [ 1.          0.59633028]
Recall :  [ 0.5  1. ]
```

Monks 1

```
accuracy:  [ 0.66666667  0.66666667]
confusion matrix :
[[ 216.  216.]
 [   0.  216.]]
Precision :  [ 1.   0.5]
```

Recall : [0.5 1.]

Monks 2

accuracy: [0.59833795 0.59833795]
confusion matrix :
[[290. 290.]
[0. 142.]]
Precision : [1. 0.3287037]
Recall : [0.5 1.]

Monks 3

accuracy: [0.67924528 0.67924528]
confusion matrix :
[[204. 204.]
[0. 228.]]
Precision : [1. 0.52777778]
Recall : [0.5 1.]

Temps d'exécution de la fonction test :

	Iris	Congress	Monks 1	Monks 2	Monks 3
Knn	0.070886850	0.68112897	0.56426191	0.771773099	0.558285951
Bayes	0.016270875	0.101378202	0.087309122	0.082180023	0.084768056

Nous pouvons remarquer qu'en général l'algorithme de Bayes Naïf est plus rapide.

Accuracy :

	Knn	Bayes
Iris	[1. 0.94666667 0.94666667]	[0.83739837 0.6097561 0.77235772]
Congress	[0.92201835 0.92201835]	[0.7124183 0.7124183]
Monks 1	[0.84259259 0.84259259]	[0.66666667 0.66666667]
Monks 2	[0.69907407 0.69907407]	[0.59833795 0.59833795]
Monks 3	[0.89351852 0.89351852]	[0.67924528 0.67924528]

Précision :

	Knn	Bayes
Iris	[1. 0.89655172 0.96153846]	[1. 0.36 1.]
Congress	[0.90804598 0.93129771]	[1. 0.59633028]
Monks 1	[0.83636364 0.8490566]	[1. 0.5]
Monks 2	[0.76315789 0.546875]	[1. 0.3287037]
Monks 3	[0.8872549 0.89912281]	[1. 0.52777778]

Recall :

	Knn	Bayes
Iris	[1. 0.96296296 0.89285714]	[0.5 1. 0.5]
Congress	[0.89772727 0.93846154]	[0.5 1.]
Monks 1	[0.85185185 0.83333333]	[0.5 1.]
Monks 2	[0.8 0.49295775]	[0.5 1.]
Monks 3	[0.8872549 0.89912281]	[0.5 1.]

Conclusion

Avec les données que nous avons recueillis nous pouvons constaté que l'algorithme de k-nearest neighbor donne une meilleur accuracy que Bayes Naïf. Cependant, le temps d'exécution de Bayes est plus petit.

La plus grande difficultés rencontré lors de ce travail a été l'implémentation de l'algorithme de Bayes. Plusieurs problème ce sont produit comme d'avoir une variance qui correspondait a des valeurs trouvé sur wikipédia. Je me servais de ces données pour voir si mon calcul de variance faisait du sens, mais je n'arrivais jamais au même résultat. J'ai donc laissé tomber après avoir atteint un résultat assez près. J'ai peur que cela est un impact sur mes resultat de Accuracy, Precision et Recall.