Melissa LaHoud

Grand Canyon University

DSC-540: Machine Learning

Dr. Aiman Darwiche

3/31/2021

Confusion Matrix

Confusion matrix has a role of being a measure used while solving classification problems. Pattern recognition or object detection is a little different than numeric. According to Koech, K. (2020), “But calculating of confusion matrix for object detection and instance segmentation tasks is less intuitive. First, it is necessary to understand another supporting metric: Intersection over Union (IoU). A key role in calculating metrics for object detection and instance segmentation tasks is played by Intersection over Union (IoU).” IoU is calculated as the area of overlap/intersection between gt and pd divided by the area of the union between the two, that is,

$$\text{IoU} = \frac{\text{area}(gt \cap pd)}{\text{area}(gt \cup pd)}$$

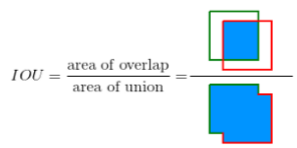


Fig 1 (Source: Author)

“A confusion matrix is made up of 4 components, namely, True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). To define all the components, we need to define some threshold (say α) based on IoU.” (Koech, K., 2020)

- True Positive (TP) — This is an instance in which the classifier predicted positive when the truth is indeed positive, that is, a detection for which IoU ≥ α.
- False Positive (FP) — This is a wrong positive detection, that is, a detection for which IoU < α.
- False Negative (FN) — This is an actual instance that is not detected by the classifier.
- True Negative (TN) — This metric implies a negative detection given that the actual instance is also negative. In object detection, this metric does

not apply because there exist many possible predictions that should not be detected in an image. Thus, TN includes all possible wrong detection that were not detected.

These will also help you find Precision, sensitivity, specificity, error rate, and accuracy. Below are the equations to each listed along with a confusion matrix itself to see where each equation is pulling from.

		Predicted Category		
		0	**1**	**Total**
Actual category	**0**	*TN*	*FP*	**TAN**
	1	*FN*	*TP*	**TAP**
	Total	**TPN**	**TPP**	**GT**

$$Precision = \frac{TP}{TPP}$$

$$Sensitivity = \frac{Number\ of\ true\ positives}{Total\ actually\ positive} = \frac{TP}{TAP} = \frac{TP}{TP+FN}$$

$$Specificity = \frac{Number\ of\ true\ negatives}{Total\ actually\ negative} = \frac{TN}{TAN} = \frac{TN}{FP+TN}$$

$$Error\ Rate = 1 - Accuracy = \frac{FN+FP}{TN+FN+FP+TP} = \frac{FN+FP}{GT}$$

$$Accuracy = \frac{TN+TP}{TN+FN+FP+TP} = \frac{TN+TP}{GT}$$

Let’s look at an image for an example made by Koech, K (2020) for image recoginition:

Below are the parameters:

Parameters:

- **ground** — is $n \times m \times 2$ array where n is number of the ground truth instances for the given image, m is the number of (x,y) pairs sampled on the circumference of the mask.
- **pred** is $p \times q \times 2$ array where p is the number of detections, and q is the number of (x,y) points sampled for the prediction mask
- **iou_value** is the IoU threshold

With lots of code in python, he concluded:

For Fig 5 and IoU threshold, $\alpha = 0.5$, `evaluation(ground,pred,iou_value)` →

```
TP: 9   FP: 5   FN: 0   GT: 10
Precall: 0.643   Recall: 1.0   F1 score: 0.783
```

Another example I did in a previous class which is using the Loans dataset where I used a confusion matrix to come up with each of model evaluation measures.

```
In [9]:  y = training[['Approval']]

         x = pd.concat((training[['Debt-to-Income Ratio']], training[['FICO Score']], training[['Request Amount']]), axis=1)

         x_names = ["Debt to Income Ratio", "FICO","Request Amt"]

         y_names = ["T","F"]

         C50_01 = DecisionTreeClassifier(criterion = "entropy", max_leaf_nodes=5).fit(x,y)

         training['C1'] = C50_01.predict(x)

         C51 = pd.crosstab(training['Approval'], training['C1'])
         C51
```

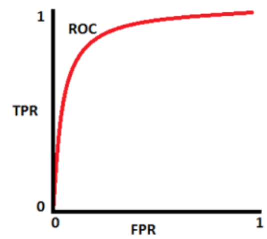
Out[9]:

C1	F	T
Approval		
F	53589	21477
T	2237	72999

Evaluation Measure	Model 1 Formula	Model 1 Value	Model 2 Formula	Model 2 Value
Accuracy	53589+72999/150302	.842	62443+60789/150302	.820
Error Rate	1-.842	.158	1-.820	.180
Sensitivity	72999/75236	.970	60789/75236	.808
Specificity/Recall	53589/75066	.714	62443/75066	.832
Precision	72999/94476	.773	60789/73412	.828

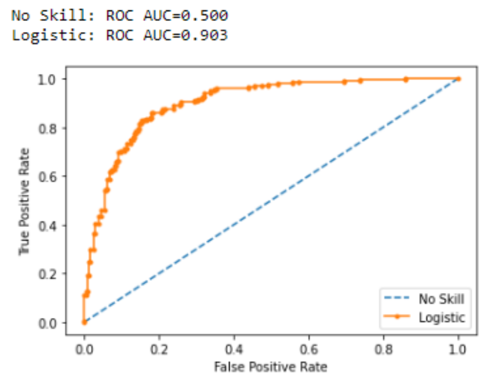
ROC Curve

The role of a ROC curve is to show the performance measurement for the classification problems at various threshold settings, ROC is a probability curve. According to Narkhede, S. (2021), “It is one of the most important evaluation metrics for checking any classification model's performance because we need to check or visualize the performance of the multi-class classification problem." We can plot the ROC curve by knowing the True Positive Rate/Recall/Sensitivity and the False Positive Rate which is the complement of the specificity. "An ROC graph, hence, shows relative trade-offs between advantages (true positives) and costs (false positives)."



There was a great example done in python that I tried:

```
[1]:
# roc curve and auc
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot

# generate 2 class dataset
X, y = make_classification(n_samples=1000, n_classes=2, random_state=1)

# split into train/test sets
trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.5, random_state=2)

# generate a no skill prediction (majority class)
ns_probs = [0 for _ in range(len(testy))]

# fit a model
model = LogisticRegression(solver='lbfgs')
model.fit(trainX, trainy)

# predict probabilities
lr_probs = model.predict_proba(testX)

# keep probabilities for the positive outcome only
lr_probs = lr_probs[:, 1]

# calculate scores
ns_auc = roc_auc_score(testy, ns_probs)
lr_auc = roc_auc_score(testy, lr_probs)

# summarize scores
print('No Skill: ROC AUC=%.3f' % (ns_auc))
print('Logistic: ROC AUC=%.3f' % (lr_auc))

# calculate roc curves
ns_fpr, ns_tpr, _ = roc_curve(testy, ns_probs)
lr_fpr, lr_tpr, _ = roc_curve(testy, lr_probs)

# plot the roc curve for the model
pyplot.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
pyplot.plot(lr_fpr, lr_tpr, marker='.', label='Logistic')

# axis labels
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')

# show the legend
pyplot.legend()

# show the plot
pyplot.show()
```



(Brownlee, J., 2021)

References

Brownlee, J. (2021, January 12). How to Use ROC Curves and Precision-Recall Curves for Classification in Python. Retrieved from https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

Koech, K. E. (2020, July 31). Confusion Matrix and Object Detection. Retrieved from https://towardsdatascience.com/confusion-matrix-and-object-detection-f0cbcb634157

Narkhede, S. (2021, January 14). Understanding AUC - ROC Curve. Retrieved from https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5