Các phương pháp đánh giá một hệ thống phân lớp

1. Giới thiệu

Khi xây dựng một mô hình Machine Learning, chúng ta cần một phép đánh giá để xem mô hình sử dụng có hiệu quả không và để so sánh các mô hình với nhau. Trong bài viết này, tôi sẽ giới thiệu các phương pháp đánh giá các mô hình classification.

Hiệu năng của một mô hình thường được đánh giá dựa trên tập dữ liệu kiểm thử (test data). Ta cần so sánh kết quả dự đoán với kết quả thật để xem tỉ lệ chính xác.

Ví dụ với bài toán có 3 lớp dữ liệu được gán nhãn là `0, 1, 2`.

`(Trong bài toán thực tế, các class có thể có nhãn bất kỳ, không nhất thiết là số, và không nhất thiết bắt đầu từ 0).`

Chúng ta hãy tạm giả sử các class được đánh số từ `0` đến `C-1` trong trường hợp có `C` lớp dữ liệu.

> *Có 10 điểm dữ liệu trong tập kiểm thử với các nhãn thực sự được mô tả bởi `y_true = [0, 0, 0, 0, 1, 1, 1, 2, 2, 2]`. Giả sử bộ phân lớp chúng ta đang cần đánh giá dự đoán nhãn cho các điểm này là `y_pred = [0, 1, 0, 2, 1, 1, 0, 2, 1, 2]`.*

Tuỳ vào những bài toán khác nhau mà chúng ta sử dụng các phương pháp khác nhau. Các phương pháp thường được sử dụng là: accuracy score, confusion matrix, ROC curve, Area Under the Curve, Precision and Recall, F1 score, Top R error, etc.

2. Accuracy

Cách đơn giản và hay được sử dụng nhất là ***accuracy*** (độ chính xác). Cách đánh giá này đơn giản tính tỉ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm thử.

Trong ví dụ trên, ta có thể đếm được có 6 điểm dữ liệu được dự đoán đúng trên tổng số 10 điểm. Vậy ta kết luận độ chính xác của mô hình là 0.6 (hay 60%). Để ý rằng đây là bài toán với chỉ 3 class, nên độ chính xác nhỏ nhất đã là khoảng 1/3, khi tất cả các điểm được dự đoán là chỉ thuộc vào một class nào đó.

> Tương tự như làm bài trắc nghiệm, nếu bạn không làm bài mà chỉ chọn 1 kết quả (toàn bộ là A chẳng hạn) thì bạn sẽ đạt 2.5 điểm (25%). Đó là độ chính xác nhỏ nhất.

3. Confusion matrix

Cách tính sử dụng accuracy như ở trên chỉ cho chúng ta biết được bao nhiêu phần trăm lượng dữ liệu được phân loại đúng mà không chỉ ra được cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất, và dữ liệu thuộc lớp nào thường bị phân loại nhầm vào lớp khác. Để có thể đánh giá được các giá trị này, chúng ta sử dụng một ma trận được gọi là *confusion matrix*.

Về cơ bản, confusion matrix thể hiện có bao nhiêu điểm dữ liệu *thực sự* thuộc vào một class, và được *dự đoán* là rơi vào một class. Để hiểu rõ hơn, hãy xem bảng dưới đây:

```
 Total: 10 | Predicted | Predicted | Predicted |
           |    as: 0  |    as: 1  |    as: 2  |
-----------|-----------|-----------|-----------|---
 True: 0   |     2     |     1     |     1     | 4
-----------|-----------|-----------|-----------|---
 True: 1   |     1     |     2     |     0     | 3
-----------|-----------|-----------|-----------|---
 True: 2   |     0     |     1     |     2     | 3
-----------|-----------|-----------|-----------|---
```

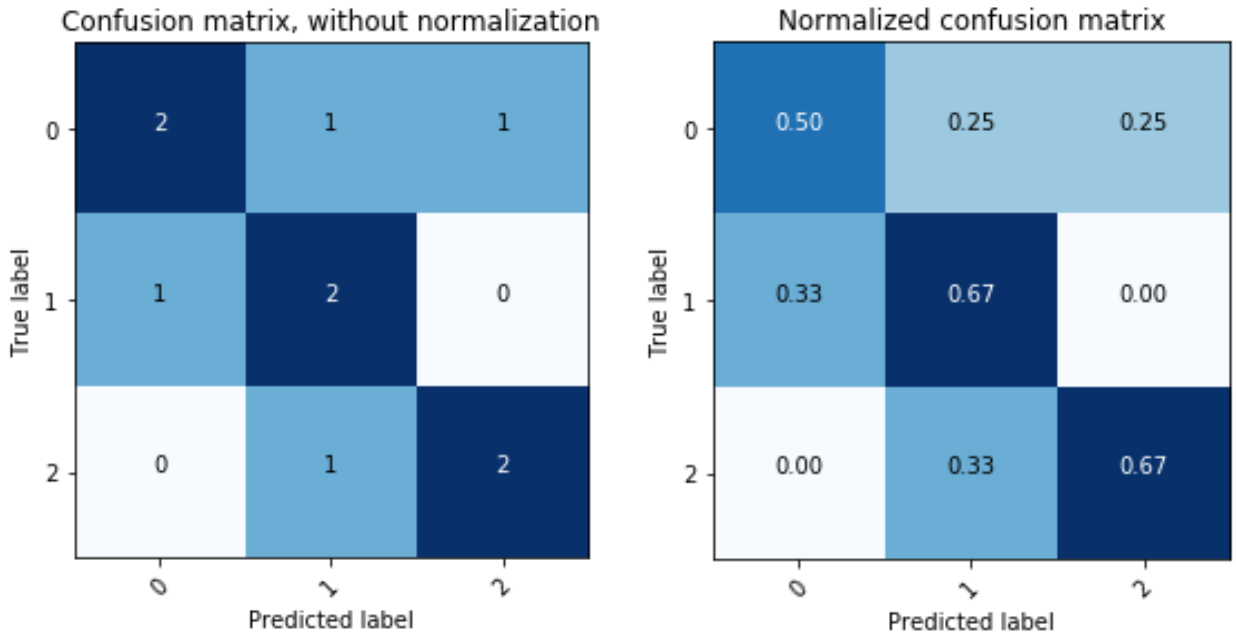
Có tổng cộng 10 điểm dữ liệu. Chúng ta xét ma trận tạo bởi các giá trị tại vùng 3x3 trung tâm của bảng.

Ma trận thu được được gọi là ***confusion matrix***. Nó là một ma trận vuông với kích thước mỗi chiều bằng số lượng lớp dữ liệu. Giá trị tại hàng thứ `i`, cột thứ `j` là số lượng điểm **lẽ ra thuộc vào class `i` nhưng lại được dự đoán là thuộc vào class `j`**. Như vậy, nhìn vào hàng thứ nhất (`0`), ta có thể thấy được rằng trong số bốn điểm thực sự thuộc lớp `0`, chỉ có hai điểm được phân loại đúng, hai điểm còn lại bị phân loại nhầm vào lớp `1` và lớp `2`.

Chú ý: *Có một số tài liệu định nghĩa ngược lại, tức giá trị tại* ***cột*** *thứ* i, ***hàng*** *thứ* j *là số lượng điểm lẽ ra thuộc vào class* i *nhưng lại được dự đoán là thuộc vào class* j*. Khi đó ta sẽ được confusion matrix là ma trận chuyển vị của confusion matrix như cách tôi đang làm. Tôi chọn cách này vì đây chính là cách thư viện sklearn sử dụng.*

Chúng ta có thể suy ra ngay rằng tổng các phần tử trong toàn ma trận này chính là số điểm trong tập kiểm thử. Các phần tử trên đường chéo của ma trận là số điểm được phân loại đúng của mỗi lớp dữ liệu. Từ đây có thể suy ra ***accuracy*** chính bằng tổng các phần tử trên đường chéo chia cho tổng các phần tử của toàn ma trận.

Cách biểu diễn trên đây của confusion matrix còn được gọi là ***unnormalized confusion matrix***, tức ma *confusion matrix* chưa chuẩn hoá. Để có cái nhìn rõ hơn, ta có thể dùng ***normalized confuion matrix***, tức *confusion matrix* đã chuẩn hoá. Để có ***normalized confusion matrix***, ta lấy mỗi hàng của ***unnormalized confusion matrix*** chia cho tổng các phần tử trên hàng đó. Như vậy, tổng các phần tử trên một hàng của *normalized confusion matrix* luôn bằng 1.



Hình 1: Minh hoạ *unnormalized confusion matrix* và *normalized confusion matrix* .

Với các bài toán với nhiều lớp dữ liệu, cách biểu diễn bằng màu này rất hữu ích. Các ô màu đậm thể hiện các giá trị cao. Một mô hình tốt sẽ cho một ***confusion matrix*** có các phần tử trên đường chéo chính có giá trị lớn, các phần tử còn lại

có giá trị nhỏ. Nói cách khác, khi biểu diễn bằng màu sắc, đường chéo có màu càng đậm so với phần còn lại sẽ càng tốt. Từ hai hình trên ta thấy rằng confusion matrix đã chuẩn hoá mang nhiều thông tin hơn. Sự khác nhau được thấy ở ô trên cùng bên trái. Lớp dữ liệu `0` được phân loại không thực sự tốt nhưng trong ***unnormalized confusion matrix***, nó vẫn có màu đậm như hai ô còn lại trên đường chéo chính.

4. True/False Positive/Negative

4.1. True/False Positive/Negative

Cách đánh giá này thường được áp dụng cho các bài toán phân lớp có hai lớp dữ liệu. Cụ thể hơn, trong hai lớp dữ liệu này có một lớp *nghiêm trọng* hơn lớp kia và cần được dự đoán chính xác.

> *Ví dụ, trong bài toán xác định có bệnh ung thư hay không thì việc không bị sót (miss) quan trọng hơn là việc chẩn đoán nhầm âm tính thành dương tính.*
>
> *Trong bài toán xác định có mìn dưới lòng đất hay không thì việc bỏ sót nghiêm trọng hơn việc báo động nhầm rất nhiều.*

> *Hay trong bài toán lọc email rác thì việc cho nhầm email quan trọng vào thùng rác nghiêm trọng hơn việc xác định một email rác là email thường.*

Trong những bài toán này, người ta thường định nghĩa lớp dữ liệu *quan trọng* hơn cần được xác định đúng là lớp ***Positive*** (P-dương tính), lớp còn lại được gọi là ***Negative*** (N-âm tính). Ta định nghĩa ***True Positive*** *(TP),* ***False Positive*** *(FP),* ***True Negative*** *(TN),* ***False Negative*** *(FN)* dựa trên ***confusion matrix*** chưa chuẩn hoá như sau:

	Predicted as Positive	Predicted as Negative
Actual: Positive	True Positive (TP)	False Negative (FN)
Actual: Negative	False Positive (FP)	True Negative (TN)

```
------------------|--------------------|--------------------|
```

Người ta thường quan tâm đến TPR, FNR, FPR, TNR (R - Rate) dựa trên *normalized* ***confusion matrix*** như sau:

```
                  |     Predicted      |     Predicted      |
                  |    as Positive     |    as Negative     |
------------------|--------------------|--------------------|
 Actual: Positive | TPR = TP/(TP + FN) | FNR = FN/(TP + FN) |
------------------|--------------------|--------------------|
 Actual: Negative | FPR = FP/(FP + TN) | TNR = TN/(FP + TN) |
------------------|--------------------|--------------------|
```

False Positive Rate còn được gọi là ***False Alarm Rate*** (tỉ lệ báo động nhầm), ***False Negative Rate*** còn được gọi là ***Miss Detection Rate*** (tỉ lệ bỏ sót). Trong bài toán dò mìn, *thà báo nhầm còn hơn bỏ sót*, tức là ta có thể chấp nhận ***False Alarm Rate*** cao để đạt được ***Miss Detection Rate*** thấp.

Chú ý::

- Việc biết một cột của ***confusion matrix*** này sẽ suy ra được cột còn lại vì tổng các hàng luôn bằng 1 và chỉ có hai lớp dữ liệu.
- **Với các bài toán có nhiều lớp dữ liệu**, ta có thể xây dựng bảng True/False Positive/Negative cho **mỗi lớp** nếu coi lớp đó là lớp *Positive*, các lớp còn lại gộp chung thành lớp *Negative*, giống như cách làm trong one-vs-rest. Bạn có thể xem thêm ví dụ tại đây.

4.2. Receiver Operating Characteristic curve

Trong một số bài toán, việc tăng hay giảm FNR, FPR có thể được thực hiện bằng việc thay đổi một *ngưỡng* (threshold) nào đó. Lấy ví dụ khi ta sử dụng thuật toán Logistic Regression, đầu ra của mô hình có thể là các *lớp cứng* `0` hay `1`, hoặc cũng có thể là các giá trị thể hiện xác suất để dữ liệu đầu vào thuộc vào lớp `1`. Khi sử dụng thư viện sklearn Logistic Regression, ta có thể lấy được các giá trị xác xuất này bằng phương thức `predict_proba()`. Mặc định, ngưỡng được sử dụng là 0.5, tức là một điểm dữ liệu `x` sẽ được dự đoán rơi vào lớp `1` nếu giá trị `predict_proba(x)` lớn hơn 0.5 và ngược lại.

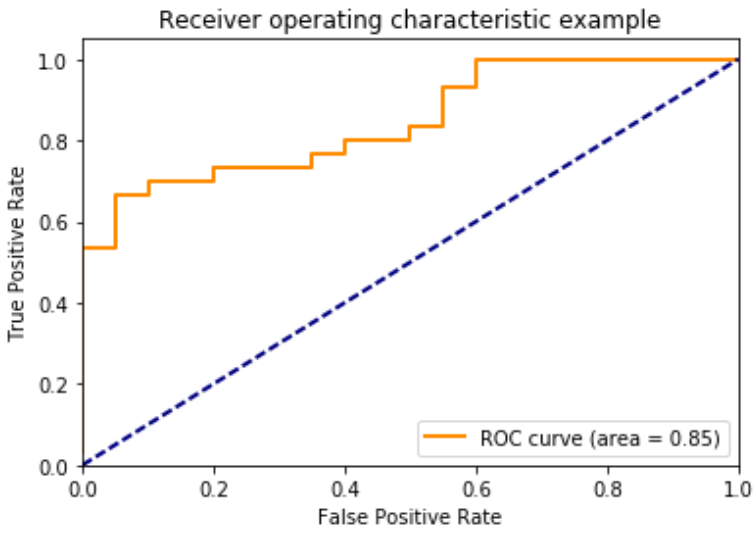
Nếu bây giờ ta coi lớp `1` là lớp *Positive*, lớp `0` là lớp *Negative*, câu hỏi đặt ra là làm thế nào để tăng mức độ ***báo nhầm*** (FPR) để giảm mức độ ***bỏ sót*** (FNR)?

Chú ý rằng tăng FNR đồng nghĩa với việc giảm TPR vì tổng của chúng luôn bằng 1.

Một kỹ thuật đơn giản là ta thay giá trị threshold từ 0.5 xuống một số nhỏ hơn như là 0.3, thì mọi điểm được dự đoán lớn hơn 0.3 sẽ được dự đoán là thuộc lớp Positive. Nói cách khác, tỉ lệ các điểm được phân loại là Positive sẽ tăng lên, kéo theo cả False Positive Rate và True Positive Rate cùng tăng lên (cột thứ nhất trong ma trận tăng lên). Từ đây suy ra cả FNR và TNR đều giảm.

Ngược lại, nếu ta muốn *bỏ sót còn hơn báo nhầm*, như bài toán xác định email rác chẳng hạn, ta cần tăng threshold lên một số lớn hơn 0.5. Khi đó, hầu hết các điểm dữ liệu sẽ được dự đoán thuộc lớp `0` (tức *Negative)*, và cả TNF và FNR đều tăng lên, tức TPR và FPR giảm xuống.

Như vậy, ứng với mỗi giá trị của threshold, ta sẽ thu được một cặp (FPR, TPR). Biểu diễn các điểm (FPR, TPR) trên đồ thị khi thay đổi threshold từ 0 tới 1 ta sẽ thu được một đường được gọi là *Receiver Operating Characteristic curve* hay ROC curve. (*Chú ý rằng khoảng giá trị của threshold không nhất thiết từ 0 tới 1 trong các bài toán tổng quát. Khoảng giá trị này cần được đảm bảo có trường hợp TPR/FPR nhận giá trị lớn nhất hay nhỏ nhất mà nó có thể đạt được*).



Hình 2: Ví dụ về Receiver Operating Characteristic curve và Area Under the Curve.

4.3. Area Under the Curve

Dựa trên ROC curve, ta có thể chỉ ra rằng một mô hình có hiệu quả hay không. Một mô hình hiệu quả khi có FPR thấp và TPR cao, tức tồn tại một điểm trên

ROC curve gần với điểm có toạ độ (0, 1) trên đồ thị (góc trên bên trái). Curve càng gần thì mô hình càng hiệu quả.

Có một thông số nữa dùng để đánh giá mà tôi đã sử dụng ở trên được gọi là *Area Under the Curve* hay *AUC*. Đại lượng này chính là diện tích nằm dưới ROC curve màu cam. Giá trị này là một số dương nhỏ hơn hoặc bằng 1. Giá trị này càng lớn thì mô hình càng tốt.

Chú ý: Cross validation cũng có thể được thực hiện bằng cách xác định ROC curve và AUC lên [validation set].

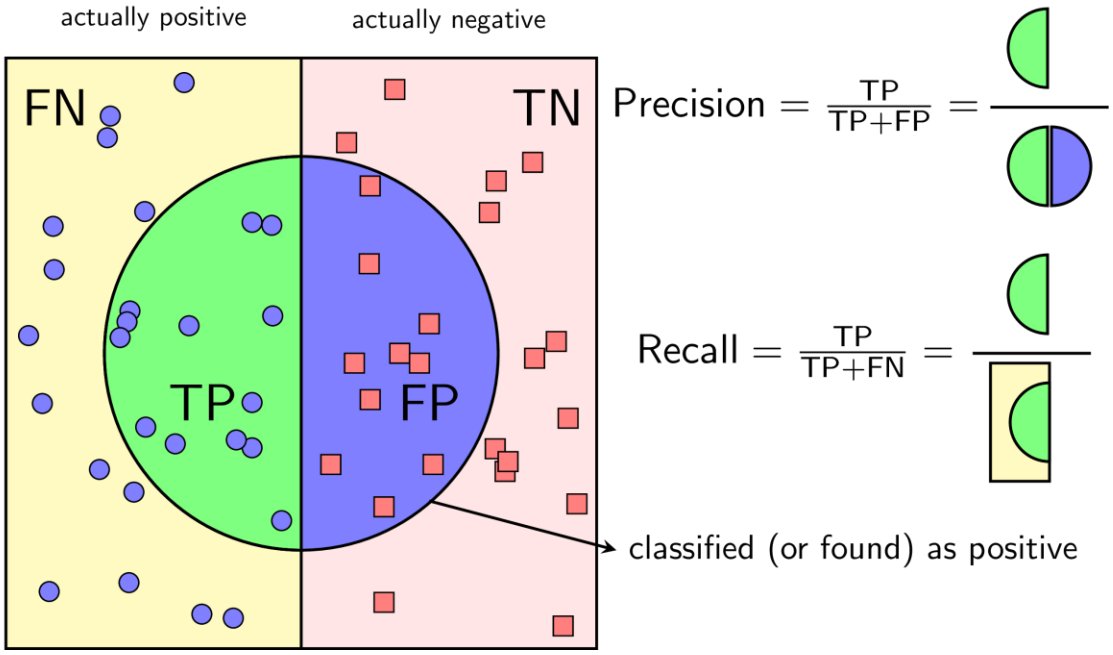
5. Precision và Recall

5.1 Định nghĩa

Với bài toán phân loại mà tập dữ liệu của các lớp là chênh lệch nhau rất nhiều, có một phép đó hiệu quả thường được sử dụng là Precision-Recall.

Trước hết xét bài toán phân loại nhị phân. Ta cũng coi một trong hai lớp là *positive*, lớp còn lại là *negative*.

Xét Hình 3 dưới đây:



Nguồn: https://machinelearningcoban.com/2017/08/31/evaluation/

Hình 3: Cách tính Precision và Recall.

Với một cách xác định một lớp là *positive*, **Precision** được định nghĩa là tỉ lệ số điểm **true positive** trong số những điểm **được phân loại là *positive*** (TP + FP).

Recall được định nghĩa là tỉ lệ số điểm **true positive** trong số những điểm **thực sự là *positive*** (TP + FN).

Một cách toán học, Precison và Recall là hai phân số có tử số bằng nhau nhưng mẫu số khác nhau:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$$

Bạn đọc có thể nhận thấy rằng **TPR** và **Recall** là hai đại lượng bằng nhau. Ngoài ra, cả **Precision** và **Recall** đều là các số không âm nhỏ hơn hoặc bằng một.

Precision cao đồng nghĩa với việc độ chính xác của các điểm tìm được là cao. **Recall** cao đồng nghĩa với việc **TPR** cao, tức tỉ lệ bỏ sót các điểm thực sự ***positive*** là thấp.

Khi Precision = 1, mọi điểm tìm được đều thực sự là ***positive***, tức không có điểm ***negative*** nào lẫn vào kết quả. Tuy nhiên, Precision = 1 không đảm bảo mô hình là tốt, vì câu hỏi đặt ra là liệu mô hình đã tìm được tất cả các điểm ***positive*** hay chưa. Nếu một mô hình chỉ tìm được đúng một điểm ***positive*** mà nó chắc chắn nhất thì ta không thể gọi nó là một mô hình tốt.

Khi Recall = 1, mọi điểm ***positive*** đều được tìm thấy. Tuy nhiên, đại lượng này lại không đo liệu có bao nhiêu điểm ***negative*** bị lẫn trong đó. Nếu mô hình phân loại mọi điểm là ***positive*** thì chắc chắn Recall = 1, tuy nhiên dễ nhận ra đây là một mô hình cực tồi.

Một mô hình phân lớp tốt là mô hình có cả **Precision** và **Recall** đều cao, tức càng gần một càng tốt. Có hai cách đo chất lượng của bộ phân lớp dựa vào Precision và Reall: Precision-Recall curve và F-score.

5.2. Precision-Recall curve và Average precision

Tương tự như ROC curve, chúng ta cũng có thể đánh giá mô hình dựa trên việc thay đổi một ngưỡng và quan sát giá trị của Precision và Recall. Khái niệm Area

Under the Curve (AUC) cũng được định nghĩa tương tự. Với Precision-Recall Curve, AUC còn có một tên khác là **Average precision (AP).**

Giả sử có N ngưỡng để tính precision và recall, với mỗi ngưỡng cho một cặp giá trị precision, recall là P_n, R_n, $n=1,2,\ldots,N$. Precision-Recall curve được vẽ bằng cách vẽ từng điểm có toạ độ (R_n, P_n) trên trục toạ độ và nối chúng với nhau. AP được xác định bằng:

$$AP = \sum_n (R_n - R_{n-1}) P_n$$

ở đó $(R_n - R_{n-1})P_n$ chính là diện tích hình chữ nhật có chiều rộng $(R_n - R_{n-1})$ và chiều cao P_n, đây cũng gần với cách tính tích phân dựa trên cách tính diện tích của từng hình chữ nhật nhỏ. (Nếu bạn đọc còn nhớ khái niệm *diện tích hình thang cong* thì sẽ tưởng tượng ra.)

Xem thêm Precision-Recall–scikit-learn.

5.3. F1-score

F_1 score, hay F1-score, là *harmonic mean* của precision và recall (giả sử rằng hai đại lượng này khác không):

$$\frac{2}{F_1} = \frac{1}{\text{precision}} + \frac{1}{\text{recall}}$$

hay

$$F_1 = 2\frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2\frac{\text{precision.recall}}{\text{precision+recall}}$$

$F-1$-score có giá trị nằm trong nửa khoảng $(0,1]$. F_1 càng cao, bộ phân lớp càng tốt. Khi cả recall và precision đều bằng 1 (tốt nhất có thể), $F_1=1$. Khi cả recall và precision đều thấp, ví dụ bằng 0.1, $F_1=0.1$. Dưới đây là một vài ví dụ về F_1

precision	**recall**	F_1F1
1	1	1
0.1	0.1	0.1
0.5	0.5	0.5
1	0.1	0.182
0.3	0.8	0.36

Như vậy, một bộ phân lớp với precision = recall = 0.5 tốt hơn một bộ phân lớp khác với precision = 0.3, recall = 0.8 theo cách đo này.

Trường hợp tổng quát của F_1 score là F_β score:

$$F_\beta = (1+\beta^2)\frac{\text{precision.recall}}{\beta^2.\text{precision+recall}}$$

F_1 chính là một trường hợp đặc biệt của F_β khi $\beta=1$. Khi $\beta>1$, recall được coi trọng hơn precision, khi $\beta<1$, precision được coi trọng hơn. Hai đại lượng β thường được sử dụng là $\beta=2$ và $\beta=0.5$

5.4. Precision-recall cho bài toán phân lớp nhiều lớp

Cũng giống như ROC curve, precision-recall curve ban đầu được định nghĩa cho bài toán phân lớp nhị phân. Để có thể áp dụng các phép đo này cho bài toán multi-class classification, các đại lượng đầu ra (ground truth và predicted output) cần được đưa về dạng nhị phân.

Bằng trực giác, ta có thể đưa bài toán phân lớp nhiều lớp về bài toán phân lớp nhị phân bằng cách xem xét từng lớp. Với mỗi lớp, ta coi dữ liệu thuộc lớp đó có label là ***positive***, tất cả các dữ liệu còn lại có label là ***negative***. Sau đó, giá trị Precision, Recall, và PR curve được áp dụng lên từng lớp. Với mỗi lớp, ta sẽ nhận được một cặp giá trị precision và recall. Với các bài toán có ít lớp dữ liệu, ta có thể minh hoạ PR curve cho từng lớp trên cùng một đồ thị. Tuy nhiên, với các bài toán có rất nhiều lớp dữ liệu, việc này đôi khi không khả thi. Thay vào đó, hai phép đánh giá dựa trên Precision-Recall được sử dụng là *micro-average* và *macro-average*.

6. Tóm tắt

- Accuracy là tỉ lệ giữa số điểm được phân loại đúng trên tổng số điểm. Accuracy chỉ phù hợp với các bài toán mà kích thước các lớp dữ liệu là tương đối như nhau.
- Confusion matrix giúp có cái nhìn rõ hơn về việc các điểm dữ liệu được phân loại đúng/sai như thế nào.
- True Positive (TP): số lượng điểm của lớp ***positive*** được phân loại đúng là ***positive***.

- True Negative (TN): số lượng điểm của lớp ***negative*** được phân loại đúng là *negative*.
- False Positive (FP): số lượng điểm của lớp ***negative*** bị phân loại nhầm thành ***positive***.
- False Negative (FN): số lượng điểm của lớp ***positive*** bị phân loại nhầm thành ***negative***
- True positive rate (TPR), false negative rate (FNR), false positive rate (FPR), true negative rate (TNR):

```
                 |     Predicted      |     Predicted      |
                 |    as Positive     |    as Negative     |
-----------------|--------------------|--------------------|
 Actual: Positive | TPR = TP/(TP + FN) | FNR = FN/(TP + FN) |
-----------------|--------------------|--------------------|
 Actual: Negative | FPR = FP/(FP + TN) | TNR = TN/(FP + TN) |
-----------------|--------------------|--------------------|
```

- Khi kích thước các lớp dữ liệu là chênh lệch (*imbalanced data* hay *skew data*), precision và recall thường được sử dụng:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$$

- F_1 score:

$$F_1 = 2\frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2\frac{\text{precision.recall}}{\text{precision+recall}}$$