Model evaluation for classification

Richard Xu

August 15, 2022

1 Introduction

Here, I will discuss some common concepts and techniques for classification model evaluation, including bootstrapping sampling, confusion matrices, receiver operating characteristic (ROC) curves.

The purpose of this note is to provide intuitive explanations for all students, with or without a mathematics background, so they may not be as rigorous.

I also removed the integral and matrix-vector representations for the entire note. I used summation and scalar for this note. Therefore, there are **no** linear algebra nor calculus requirements to understand this note.

2 Model Evaluation

2.1 bootstrap samples

Bootstrap sampling is an alternative to cross-validation.

Suppose you have N data objects. The data-set is sampled N times **with replacement**, resulting in a "bootstrap sample" of size also to be N. This is used for **training set**. Note that the same data may appear multiple times in the bootstrap sample. Let me illustrate this using an example:

$$\begin{aligned} \text{original dataset} &= \{1,2,3,4,5,6,7,8,9,10\} \\ &\begin{cases} \text{bootstrap training set} & = \{1,2,2,4,4,5,7,7,9,10\} \\ \text{bootstrap testing set} & = \{3,6,8\} \end{cases} \end{aligned} \tag{1}$$

Data objects that do not make it into the training set eventually form **test set**. Repeat training-testing split by bootstrap sampling $|B|$ times, then you have $B = \{b_1, \dots b_{|B|}\}$ different bootstrap samples. Therefore B may look something like:

$$B = \begin{cases} b_1 & = \begin{cases} \text{bootstrap training set} & = \{1,2,2,4,4,5,7,7,9,10\} \\ \text{bootstrap testing set} & = \{3,6,8\} \end{cases} \\ \vdots \\ b_{|B|} & = \begin{cases} \text{bootstrap training set} & = \{2,2,3,4,4,5,8,8,9,10\} \\ \text{bootstrap testing set} & = \{1,6,7\} \end{cases} \end{cases} \tag{2}$$

The question then is for each $b_i \in B$, after you sample the N-element data uniformly N times with replacement. Uniformly meaning each data has the same probability of being selected. Then, what percentage of N original data made into the bootstrap set (some of the values are repeated) during a particular training-test split?

2.1.1 proportion of distinct samples in training and testing

It turns out that on average 63.2% of samples appear in the bootstrap (training) sample have distinct values. In another word, on average, 63.2% of the original samples will appear in bootstrap (let me repeat, some values are repeated). Then, on average, the remaining 36.8% will form the test set.

How do we obtain the value 63.2%? Instead of compute probabilities of:

$$\Pr(\text{sample selected 1 times}) + \Pr(\text{sample selected 2 times}) + \cdots + \Pr(\text{sample selected } N \text{ times}) \tag{3}$$

after N samples with replacement, it is much easier to compute the probability of never choosing that particular sample after N uniform sampling. Since the probability that a single draw of not selecting a sample is $1 - \frac{1}{N}$, then:

$$\begin{aligned}\Pr(\text{sample never selected after } N \text{ sampling}) &= \left(1 - \frac{1}{N}\right)^N \\ \Longrightarrow \Pr(\text{sample selected at least once}) &= 1 - \left(1 - \frac{1}{N}\right)^N\end{aligned} \tag{4}$$

Here we apply some mathematical properties of $\exp(\cdot)$ function:

$$\lim_{N\to\infty} \left(1 + \frac{x}{N}\right)^N = \exp^x \tag{5}$$

You can try to use a large enough $N = 20$, and after chosen some arbitrary x, for example $x = 2$, you can see if the value of $\left(1 + \frac{2}{20}\right)^{20}$ gets very close to $\exp^2$, so when N is large, we get:

$$\Longrightarrow \left(1 - \frac{1}{N}\right)^N = \exp^{-1} \approx 0.368 \tag{6}$$

2.2 bootstrap sampling more explanation

2.2.1 estimation error (training error)

One may compute the estimation error (training error) as:

$$\text{err}_{\text{training}} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(y_i, \hat{y}(x_i)) \tag{7}$$

The reason this is training error is because the error here is evaluated using test data that also trains the model. Therefore, it should be less than the true test error when the training and test data do not overlap. This is called downward bias.

2.2.2 first attempt of bootstrap sample error

Imagine we have a collection of B training-test split using bootstrap samples, we then define **bootstrap sample error** as:

$$\text{err}_{\text{boot}} = \frac{1}{|B|}\sum_{b\in B} \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(y_i, \hat{y}_b(x_i)) \tag{8}$$

where $\hat{y}_b(x_i)$ is predicted value for data x_i using the model trained by b^{th} bootstrap samples. It causes over-fitting as training and testing data also overlaps. (Note that in here, we do not explicitly form test samples set from original samples that were not part of a training set). So it also has downward bias.

You can not combine this with the training error, as they both receive downward bias, so you get a very (downward) biased estimation of test error.

2.2.3 second attempt of bootstrap sample error

so instead of summing over $b \in B$, we now introduce B^{-1} and perform the following:

$$\text{err}_{\text{boot}-i} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|B^{-i}|}\sum_{b\in B^{-1}}\mathcal{L}(y_i, \hat{y}_b(x_i)) \tag{9}$$

B^{-i} is set of bootstrap samples that do not contain data i, and $|B^{-i}|$ is the number of such samples. For example, let $N = 4$ and let:

$$\begin{aligned} b_1 &= \{1,3,5,6\} \\ b_2 &= \{2,3,5,6\} \\ b_3 &= \{3,3,5,6\} \end{aligned} \tag{10}$$

when we compute the error for the first data, i.e., $i = 1$, then:

$$\begin{aligned} B^{-1} &= \big\{\{2,3,5,6\},\{3,3,5,6\}\big\} \\ |B^{-1}| &= 2 \end{aligned} \tag{11}$$

This approach introduces an (upward) bias, as each b contains on average only $0.623 \times N$ distinct values. You can use an extreme case to appreciate this phenomenon: (by the way, in many theoretical machine learning applications, considering extreme cases is a good way to understand the model), where a single sample is repeated N times, so that the bootstrap sample has a just a single distinct value. Then its distribution will be very different from the test data, so the error should be much higher than the underlying test error. It is not difficult to see that the lesser the proportion of distinct samples in training, the lesser the weights should be used in computing the final error estimate.

2.2.4 Final error estimate is

$$0.368 \times \text{err}_{\text{training}} + 0.632 \times \text{err}_{\text{boot}-i} \tag{12}$$

the LHS receives downward bias and RHS receives upward bias and (hopefully) they cancel each other out.

3 difference between estimated $\hat{y}$ and true label y

3.1 multi-class classification

We will formally talk about cross entropy loss and softmax function when we discuss neural networks. But let me give you a preview in here. Imagine we have a 3-class classification problem, then the labels for each $y_i \in \{1, \ldots, 3\}$. However, since class label usually are nominal (by the way, I hope you still remember what is the difference between Nominal, Ordinal, Interval and Ratio-Scaled from earlier lectures), you cannot rank them.

You can not just expect to learn a function $\hat{y}_i$ to make all class-2-labeled data to be close to 2, such as $1.89, 2.02, \ldots$, and make all class-3-labeled data to be close to 3, such as $2.87, 3.10, \ldots$ etc. It does not make sense for nominal data!

However, one way to circumvent this problem is to re-present the class label as **one-hot** vector:

$$y = \begin{cases} \text{class} & 1 : \begin{bmatrix}1 & 0 & 0\end{bmatrix} \\ \text{class} & 2 : \begin{bmatrix}0 & 1 & 0\end{bmatrix} \\ \text{class} & 3 : \begin{bmatrix}0 & 0 & 1\end{bmatrix} \end{cases} \tag{13}$$

and you are hoping to learn a function $\hat{y}(x^*)$, such that it will make a test data of each of the classes to look something like:

$$\hat{y} = \begin{cases} \text{class} \quad 1: \begin{bmatrix} 0.92 & 0.08 & 0 \end{bmatrix} \\ \text{class} \quad 2: \begin{bmatrix} 0.05 & 0.9 & 0.05 \end{bmatrix} \\ \text{class} \quad 3: \begin{bmatrix} 0.12 & 0.03 & 0.85 \end{bmatrix} \end{cases} \tag{14}$$

This representation makes sense for nominal data. You should also appreciate that both y and $\hat{y}$ represent the probability associated with each class. You can also see that $[1 \quad 0 \quad 0]$ is also a probability vector, which means that all probabilities are associated with class 1 .

If you're asking me why the ground truth is (mostly) one-hot, i.e. 100% probability associated with a particular class. Because that's how humans label data. But you can have ground truth label to be a non-one-hot vector as well.

3.1.1 measure the difference between y and $\hat{y}$

after you obtained $\hat{y}(x^*)$ from your testing data x^*, how can you compare it with ground truth y? For classification **cross entropy** is typically used:

$$\mathcal{C} = -\sum_{i=1}^{N}\sum_{k=1}^{K} \underbrace{y_{i,k}}_{p(x)} \log(\underbrace{\hat{y}_{i,k}}_{q(x)}) \tag{15}$$

In general, what is cross entropy? We'll discuss more when we discuss **Neural Networks**, and you'll see the formal meaning of entropy when we discuss decision trees in the next class.

For now, think it as a way to measure the difference between two probability densities $p(x)$ and $q(x)$.

3.1.2 relationship of Cross Entropy and KL-divergence

Remember that we mentioned Kullback-Leibler divergence several times, both in the first class and when we discussed t-SNE in class, because it is also a measure of the difference between two distributions. It turns out that the two are closely related:

$$\begin{aligned} H(p,q) &= \mathrm{E}_p[-\log q] \\ &= -\sum_x p(x)\ \log q(x) \\ &= \sum_x [-p(x)\ \log q(x) + p(x)\log(p(x)) - p(x)\log(p(x))] \\ &= \sum_x -p(x)\ \log p(x) + \sum_x p(x)\ \log \frac{p(x)}{q(x)} \\ &= H(p) + \mathrm{KL}(p\|q), \end{aligned} \tag{16}$$

After you studied the next class, you will know the Entropy H for a one-hot vector is zero! Therefore, $H(p,q) = \mathrm{KL}(p\|q)$ when p is one-hot!

4 Confusion matrix

When you are building a multi-class classifier, some of the classes are recognized more accurately than the other. For example, imagine you are building a classier trying to classify images of the categories:

- cat, tiger, dog, wolf, elephant

You can appreciate that misclassifying an image of an elephant as a cat is less common than misclassifying an image of a tiger as a cat or an image of a wolf as a dog. So how do you quantify the results?

Therefore, the confusion matrix becomes a useful tool for analyzing the ability of the classifier to recognize different classes of objects. Given a class $m (m \geq 2)$, a confusion matrix is a table of at least size $m \times m$.

4.1 Example of a Confusion matrix

imagine you have 10 objects per class for ground truth labels:

Table 1: Example of a confusion matrix

	predicted as 1	predicted as 2	predicted as 3	predicted as 4	**sum**
ground truth class 1	7	2	0	1	10
ground truth class 2	1	8	1	0	10
ground truth class 3	0	0	10	0	10
ground truth class 4	0	1	0	9	10
sum	8	11	11	10	

the (i, j) entry of Confusion Matrix indicates number of objects of class i that predicated by the classifier for class j. It is not difficult to tell that the best confusion matrix one is able to get is:

Table 2: Example of a best confusion matrix

	predicted as 1	predicted as 2	predicted as 3	predicted as 4	**sum**
ground truth class 1	10	0	0	0	10
ground truth class 2	0	10	0	0	10
ground truth class 3	0	0	10	0	10
ground truth class 4	0	0	0	10	10
sum	10	10	10	10	

In reality, it is difficult to achieve that, then if a classifier have good accuracy, most objects would be represented along diagonal of confusion matrix, and the rest of the entries being zero or close to zero (comparatively).

4.1.1 looking at binary classification

Then, the confusion matrix will be 2×2 and we can compute the these terms. Also, instead of calling it class 1, class 2, let's just call them P (positive class) and N (negative class):

Table 3: 2-class Confusion Matrix

	predicted: P	predicted: N	**sum**
ground truth: P	TP	FN	P
ground truth: N	FP	TN	N
sum	P'	N'	$P+N$

From this table, we can compute these rates. Remember in earlier lectures, we have already discussed about Precision and recall. (I know they go by some many other names, depends on the disciplinary)

1. Accuracy, recognition rate:

$$\frac{(TP+TN)}{(P+N)} \tag{17}$$

2. Error rate (misclassification rate):

$$\frac{(FP+FN)}{(P+N)} = 1 - \text{Accuracy} \tag{18}$$

3. Precision

$$\frac{TP}{(TP+FP)} = \frac{TP}{P'} \tag{19}$$

4. Specificity (true negative rate)

$$\frac{TN}{(TN+FP)} = \frac{TN}{N} \tag{20}$$

5. False negative

$$\frac{FN}{(TP+FN)} = \frac{FN}{P} = (1 - \text{Recall}) \tag{21}$$

sometimes, we call it [type II error]

The two metrics I would like you to pay a closer attention are:

1. Recall, Sensitivity, **True Positive Rate (TPR)**

$$\frac{TP}{(TP+FN)} = \frac{TP}{P} \tag{22}$$

2. **False positives rate**

$$\frac{FP}{(FP+TN)} = \frac{FP}{N} = (1 - \text{Specificity}) \tag{23}$$

sometimes, we call it [type I error]. We can also calling it false alarm rate.

4.2 Receiver Operation Characteristic (ROC) curve

$$\begin{cases} \textbf{True Positive Rate (TPR)} & = \frac{TP}{(TP+FN)} = \frac{TP}{P} \\ \textbf{False positives rate (FPR)} & = \frac{FP}{(FP+TN)} = \frac{FP}{N} \quad \text{false-alarm rate} \end{cases} \tag{24}$$

Why did I ask you to pay a closer attention to both **True Positive Rate (TPR)** and **False positives rate**? Because they can be further used to compute the ROC curve.

ROC curve is measure of **separability** of class distributions, as one reduce the threshold for positive classification, one may expect both **TPR** and **FPR** to increase

imagine your algorithm $\hat{y}_\theta(x)$ has the ability to "project" data in the line segment $[a, \dots b]$. therefore, you can set threshold t be:

$$\text{prediction} = \begin{cases} \text{positive} & \text{if } \hat{y}_\theta(x) \geq t \\ \text{negative} & \text{otherwise} \end{cases} \tag{25}$$

4.2.1 completely separable

firstly, we show an example when the classes are completely separable. In here many different values of t makes it possible to separate the classes. So what do I meant when the classes are completely separable? When I talk about this in terms of **TPR** and **FPR**, it is not difficult to see that there exist a threshold t, making:

$$\begin{cases} \textbf{True Positive Rate (TPR)} & = \frac{TP}{(TP+FN)} = 1 \\ \textbf{False positives rate (FPR)} & = \frac{FP}{(FP+TN)} = 0 \end{cases} \tag{26}$$

we can see that choosing one of the t_1, t_2 and t_3, we will have TPR and FPR of the above.

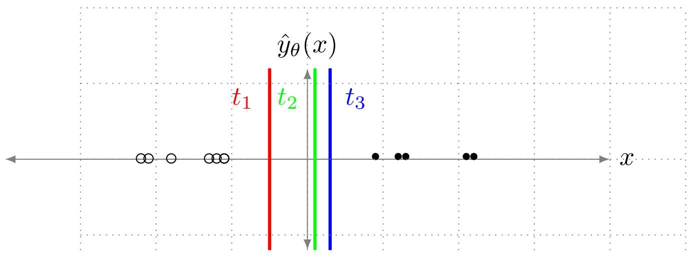

Figure 1: many t values is possible to completely separate the classes

Now, let's place the following threshold t more strategically, and to see what their corresponding **TPR** and **FPR** are. Even better, let's plot the corresponding (**TPR**, **FPR**) in a 2-D figure. Note that this figure is called Receiver Operation Characteristic (ROC) curve

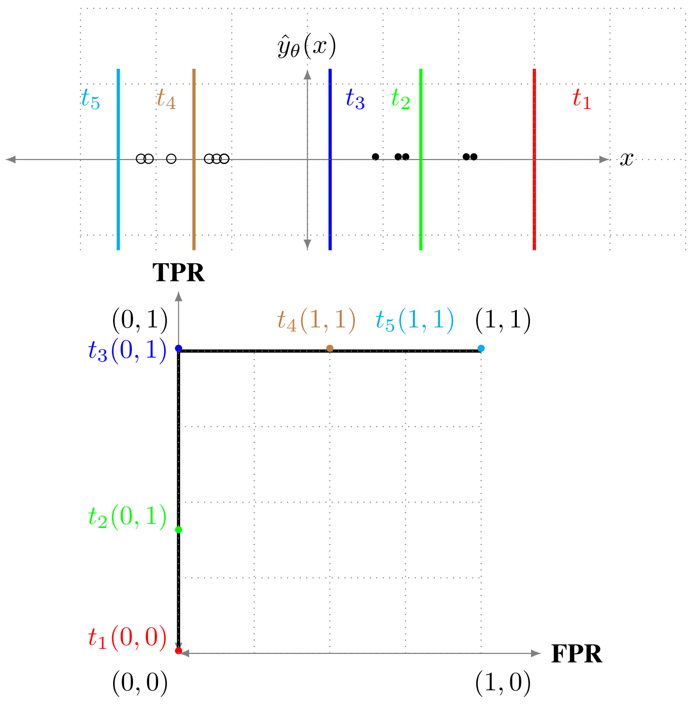

Figure 2: ROC curve for completely separable classes

For classes to be **separable**, one would expect **TPR** to increase to full (i.e., 1), before **FPR** start to increase.

4.2.2 not completely separable

Realistically we will have a situation where classes of data are **not** completely separable from your algorithm, i.e., we cannot find a any threshold t, so that $\mathbf{TPR} = 1$ and $\mathbf{FPR} = 0$, and example of such is show:

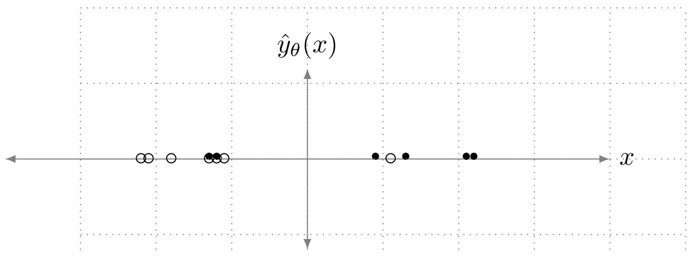

Figure 3: no t exist to completely separate the classes

Now, just like the complete separable case, let's place the following threshold t more strategically, and to see what their corresponding **TPR** and **FPR** are. We also plot the (**TPR**, **FPR**) and their corresponding Receiver Operation Characteristic (ROC) curve:

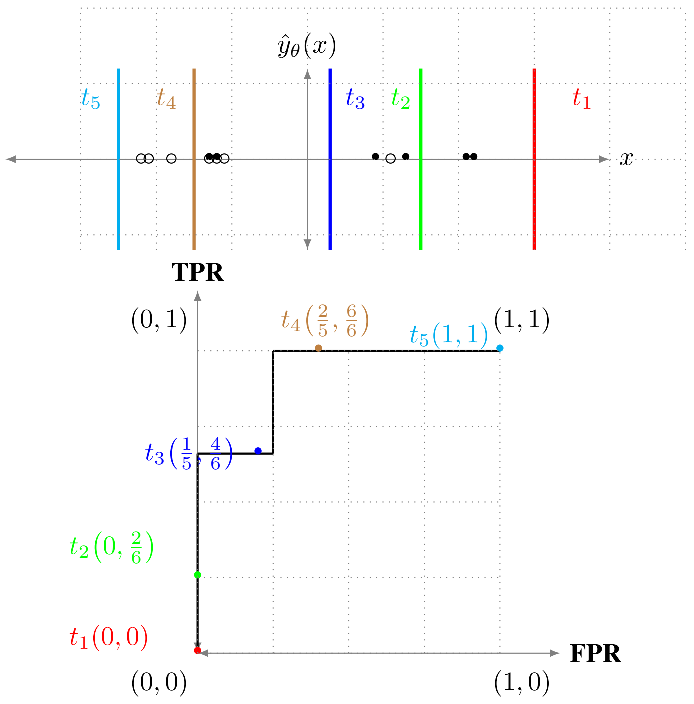

Figure 4: ROC curve for non-separable classes

4.2.3 Completely random labels

think about what may be the worst-separable classes, it's not difficult to see that a completely random labels will have ROC curve that looks like:

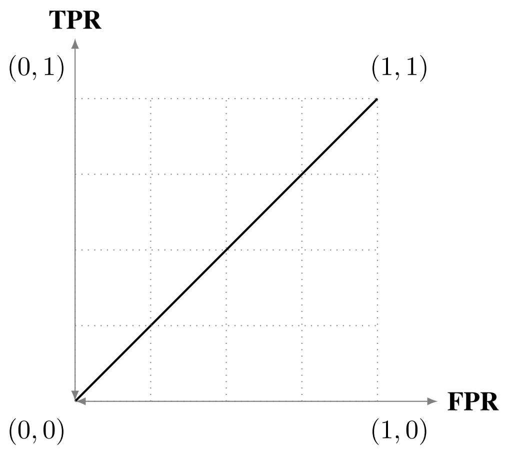

Figure 5: ROC curve for random class labels

Basically, this is where both FPR and TPR increase at the same time when the labels are completely overlapping.

So if we just want a single number to characterize the entire ROC curve, then we can just compute the area under the ROC curve. The number will be from 0.5 (completely overlapping case) to 1.0 (completely separable case)