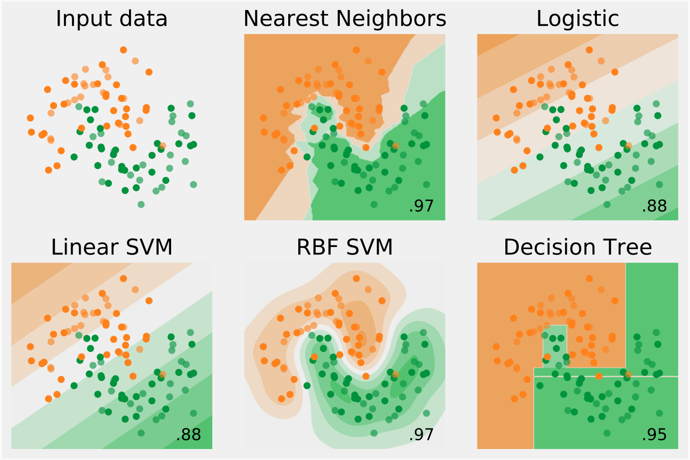
04 - Evaluating Classifier Performance

François Pitié

Ussher Assistant Professor in Media Signal Processing
Department of Electronic & Electrical Engineering, Trinity College Dublin

We have seen a number of classifiers (Logistic Regression, SVM, kernel classifiers, Decision Trees, k-NN) but we still haven't talked about their performance.

Recall some of results of classifiers:



How do we measure the performance of a classifier?

How do we compare them?

We need metrics that everybody can agree on.

Metrics for Binary Classifiers

True Positives, False Positives, True Negatives, False Negatives

If you have a binary problem with classes 0 (e.g. negative/false/fail) and 1 (e.g. positive/true/success), you have 4 possible outcomes:

True Positive : you predict $\hat{y} = 1$ and indeed $y = 1$.

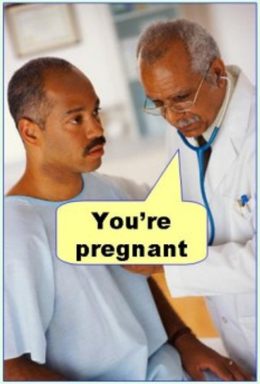
True Negative : you predict $\hat{y} = 0$ and indeed $y = 0$.

False Negative : you predict $\hat{y} = 0$ but in fact $y = 1$.

False Positive : you predict $\hat{y} = 1$ but in fact $y = 0$.

In statistics, False Positives are often called type-I errors and False Negatives type-II errors.

Type I error
(false positive)



Type II error
(false negative)

Confusion Matrix

A confusion matrix is a table that reports the number of false positives, false negatives, true positives, and true negatives for each class.

	actual: 0	actual: 1
predicted: 0	#TN	#FN
predicted: 1	#FP	#TP

For instance, on the first database slide 3

NN:

	actual: 0	actual: 1
predicted: 0	TN=166	FN=21
predicted: 1	FP=25	TP=188

Logistic Regression:

	actual: 0	actual: 1
predicted: 0	TN=152	FN=35
predicted: 1	FP=42	TP=171

Linear SVM:

	actual: 0	actual: 1
predicted: 0	TN=148	FN=39
predicted: 1	FP=41	TP=172

RBF SVM:

	actual: 0	actual: 1
predicted: 0	TN=162	FN=25
predicted: 1	FP=17	TP=196

Decision Tree:

	actual: 0	actual: 1
predicted: 0	TN=170	FN=17
predicted: 1	FP=29	TP=184

From TP, TN, FP, FN, we can derive a number of popular metrics.

Recall (also called sensitivity or true positive rate) is the probability that a positive example is indeed predicted as positive. In other words it is the proportion of positives that are correctly labelled as positives.

$$\text{recall} = \frac{\text{TP}}{P} = \frac{\text{TP}}{\text{TP} + \text{FN}} = p(\hat{y} = 1 | y = 1)$$

Precision is the probability that a positive prediction is indeed positive:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = p(y = 1 | \hat{y} = 1)$$

False Positive Rate is the proportion of negatives that are incorrectly labelled as positive:

$$\text{FP rate} = \frac{\text{FP}}{N} = \frac{\text{FP}}{\text{FP} + \text{TN}} = p(\hat{y} = 1 | y = 0)$$

Accuracy is the probability that a prediction is correct:

$$\begin{aligned}\text{Accuracy} &= \frac{\mathrm{TP}+\mathrm{TN}}{P+N} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}} \\ &= p(\hat{y}=1, y=1) + p(\hat{y}=0, y=0)\end{aligned}$$

F1 score is the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{\text{recall} \cdot \text{precision}}{\text{precision} + \text{recall}} = \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$$

Other derived metrics exist
(see [https://en.wikipedia.org/wiki/Precision_and_recall])

But remember that since there are two types of errors (false positives and false negatives), you will always need at least two metrics to capture the performance of a classifier.

Performing well on a single metric is usually meaningless. A good classifier should perform well on 2 metrics.

Example

Consider a classifier with this confusion matrix:

	actual: 0	actual: 1
predicted: 0	TN=70	FN=5
predicted: 1	FP=15	TP=10

The actual number of positives (class 1) is 15 and the actual number of negatives is 85 (class 0).

The recall is TP/(TP+FN)=10/(5+10)=66.7%

The accuracy is (TP+TN)/(TP+FN+TN+FP)=(70+10)/100=80%.

If we take a classifier (A) that always returns 1, the confusion matrix for the same problem becomes:

	actual: 0	actual: 1
predicted: 0	TN=0	FN=0
predicted: 1	FP=85	TP=15

and its recall is 15/(15+0) = 100%!!

If we take instead a classifier (B) that always returns 0, we have:

	actual: 0	actual: 1
predicted: 0	TN=85	FN=15
predicted: 1	FP=0	TP=0

and its accuracy is (85+0)/(100) = 85%!!

But both classifiers (A) and (B) are non informative and really poor choices but you need two metrics to see this:

For (A), the recall is 100% but the accuracy is only 15%.

For (B), the accuracy is 85% but the recall is 0%.

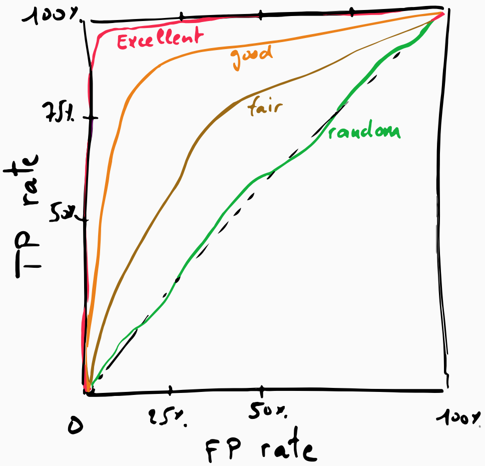
Conclusion: you need two metrics.

ROC

When you start measuring the performance of a classifier, chances are that you can tune a few parameters. For instance, if your classifier is based on a linear classifier model $y = [\mathbf{x}^\top \mathbf{w} > T]$, you can tune the threshold value T. Increasing T means that your classifier will be more conservative about classifying points as $y = 1$.

By varying the parameter T, we can produce a family of classifiers with different performances. We can report the $\mathbf{FP}$ rate = FP/(FP+TN) and $\mathbf{TP}$ rate = TP/(TP+FN) for each of the different values of T on a graph called the receiver operating characteristic curve, or R.O.C. curve.

Here is an example showing the ROC curves for 4 different classifiers.

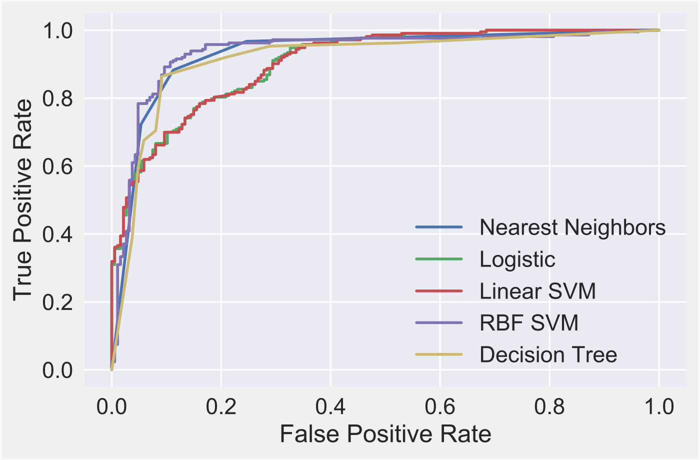


A perfect classifier will have a TP rate or 100% and a FP rate of 0%.

A random classifier will have TP rate equal to the FP rate.

If your ROC curve is below the random classifier diagonal, then you are doing something wrong.

Below are reported a few ROC curves for the problem of slide 3.



Multiclass Classifiers

Binary metrics don't adapt nicely to problems where there are more than 2 classes.

For multiclass problems with n classes, there are $n - 1$ possible ways of miss-classifying each class. Thus there are $(n - 1) \times n$ types of errors in total.

You can always present your results as a confusion matrix.

For instance:

	actual: 0	actual: 1	actual: 2
predicted: 0	102	10	5
predicted: 1	8	89	12
predicted: 2	7	11	120

You can also think of your multiclass problem as a set of binary problems (does an observation belong to class k or not), and then aggregate the binary metrics in some way.

Next are presented two ways of aggregating metrics for multiclass problems.

In micro averaging, the metric (e.g. precision, recall, F1 score) is computed from the combined true positives, true negatives, false positives, and false negatives of the K classes.

For instance the micro-averaged precision is:

$$\mathrm{microPRE} = \frac{\mathrm{microTP}}{\mathrm{microTP} + \mathrm{microFP}}$$

with $\mathrm{microTP} = \mathrm{TP}_1 + \cdots + \mathrm{TP}_K$, and $\mathrm{microFP} = \mathrm{FP}_1 + \cdots + \mathrm{FP}_K$

In macro-averaging, the performances are averaged over the classes:

$$\mathrm{macroPRE} = \frac{\mathrm{PRE}_1 + \cdots + \mathrm{PRE}_K}{K} \qquad \text{where} \quad \mathrm{PRE}_k = \frac{\mathrm{TP}_k}{\mathrm{TP}_k + \mathrm{FP}_k}$$

Example

```
y_true = [0, 1, 2, 0, 1, 2, 2]
y_pred = [0, 2, 1, 0, 0, 1, 0]
```

we have $\mathrm{TP}_0 = 2$, $\mathrm{TP}_1 = 0$, $\mathrm{TP}_2 = 0$, $\mathrm{FP}_0 = 2$, $\mathrm{FP}_1 = 2$, $\mathrm{FP}_2 = 1$

$$\mathrm{microPRE} = \frac{2+0+0}{(2+0+0)+(1+2+1)} = 0.286$$

$$\mathrm{macroPRE} = \frac{1}{3}\left(\frac{2}{2+2} + \frac{0}{0+2} + \frac{0}{0+1}\right) = 0.167$$

Training/Validation/Testing Sets

Now that we have established metrics, we still need to define the data that will be used for evaluating the metric.

You usually need:

a Training set that you use for learning the algorithm.

a Dev or Validation set, that you use to tune the parameters of the algorithm.

a Test set, that you use to fairly assess the performance of the algorithm. You should not try to optimise for the test set.

Why so many sets? Because you want to avoid over-fitting.

Say your model has many parameters. With enough training, it is likely to overfit your training set. Thus we need a testing set to check the performance on unseen data.

Now, if you tune the parameters of your algorithm to perform better on the testing set, you are likely to overfit it as well.

But we want to use the testing set as a way of accurately estimating the performance on unseen data.

Thus we use a different set, the dev/validation set, for any parameter estimation.

Important: the test and dev sets should contain examples of what you ultimately want to perform well on, rather than whatever data you happen to have for training.

How large do the dev/test sets need to be?

Training sets: as large as you can afford.

Validation/Dev sets with sizes from 1,000 to 10,000 examples are common. With 10,000 examples, you will have a good chance of detecting an improvement of 0.1%.

Test sets should be large enough to give high confidence in the overall performance of your system. One popular heuristic had been to use 30% of your data for your test set. This is makes sense when you have say 100 to 10,000 examples but not anymore as it is common you might have nowadays billion of training examples.

Take Away from a Practitioner's point of view

When approaching a new project, your first steps should be to design your Training/Validation/Test sets and decide on the metrics. Only then should you start thinking about which classifier to train.

Remember that the metrics are usually not the be-all and end-all of the evaluation. Each metric can only look at a particular aspect of your problem. You need to monitor multiple metrics.

As the project progresses, it is expected that the datasets will be updated and that new metrics will be introduced.