An Introduction to Convolutional Neural Network Benchmarks and Metrics

Landon Buell

27 January 2020

1 Introduction

Tom Mitchell, former Chair of Machine Learning Department at Carnegie Melon University, describes *machine learning* as [2]:

> A computer program is said to learn from experience E with respect to some task T, and some performance measure P, if it's performance on T, as measured by P, improves with experience, E.

As the complexity of machine learning algorithms increase, it's important that we can still measure the performance, P, as described by Mitchell. In terms of programs structures like Convolutional Neural Networks (CNNs), there exists various sets of benchmarks that we can use to evaluate the performance of that network. Each metric has a particular use and validity and each one may only apply to particular type of algorithm [5]. In this introduction, I will break these benchmarks into two very board categories: *Regression Metrics* and *Classification Metrics.*

For this work I will also use a notation convention as used by Gareth James in his book "An Introduction to Statistical Learning" [4]. A value or function that is approximated or estimated with set of data X, is denoted with a *hat.* Thus an approximated function, f becomes $\hat{f}$ which produces an estimated output $\hat{Y}$, which can be compared to known outputs, Y. Additionally, X and Y can be treated as discrete vector-like objects, which can be indexed through with a subscript. Thus sets X and Y, which n elements are equivalently:

$$X = [x_1, x_2, x_3, ..., x_i, ..., x_{n-1}, x_n] \tag{1}$$

and

$$Y = [y_1, y_2, y_3, ..., y_i, ..., y_{n-1}, y_n] \tag{2}$$

2 Regression Metrics

Generally, a regression algorithm is a type of Machine Learning Algorithm that seeks to build a function f, using a set of data X such that the result of the function is continuous, real number value [6]. In function notation:

$$f : X \to \mathbb{R} \tag{3}$$

A regression algorithm generally takes some function and applies a *best fit* line or curve through the data, X as a method of approximation [4]. The output of that function is $Y \in \mathbb{R}$.

The estimated function $\hat{f}$ operates as:

$$\hat{f}(X) = \hat{Y} \tag{4}$$

or for each individual element of X:

$$\hat{f}(x_i) = \hat{y}_i \tag{5}$$

For each of the benchmarks described below, we will assume a sufficiently large data set has been used to produce some model $\hat{f}(X) = \hat{Y}$. In each case, the function $\hat{f}$ will be some sort of N-th degree polynomial produced via linear regression. Thus, each $\hat{f}$ has the structure:

$$\hat{f}(x_n) = \hat{y} = \sum_{i=0}^{N} \beta_i x_n^i \tag{6}$$

Where β_i is a scalar coefficient, and x_i^n is the n-th predicting feature raised to the i-th power.

2.1 Mean Squared Error

Mean squared error (MSE) is a very basic way of quantifying how well the predicted response matches a known set of responses. To compute this value, we take the known output for a sample, y_i and find it's difference from the predicted output $\hat{f}(x_i)$, and then the result is squared. This quantity is repeated for each index, i, and divided by the total number of elements in the set, n.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{f}(x_i) \right)^2 \tag{7}$$

So if the predicted response from $\hat{f}$ differs very slightly from each index in the expected response, y_i, then the whole MSE takes on a very small values. This indicates a regression that fits the data well. Conversely, if the predicted response differs greatly from each index, the the whole MSE error takes on a very large value, and indicates that the data is not fit very well. In general, the goal of regression algorithms is to minimize the MSE value [6].

2.2 Root Mean Squared Error

Root mean squared error (RMSE) is another basic way of quantifying the difference between known and predicted sets of outputs. It is computed in the exact same way as means squared error (7) , but the square root of the whole function is taken. Thus RMSE is defined as:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2} \tag{8}$$

This benchmark may seem redundant, but it allows for the*units* of the metric to be consistent with that of the features and responses of the model [6]. For example, if we were using the arm length of a human to predict their height, both in centimeters, then each x_i and y_i would have units of cm. The MSE (7) would have units of cm^2 and the RMSE (8) would have units of cm.

2.3 Residual Sum of Squares

Residual sum of squares (RSS) is a benchmark used for computing a very similar measurement to MSE (7). It is computed by summing up the difference between the i-th observed response and the i-th predicted response This difference is called the *residual*, notated as e [4], and is computed:

$$e_i = \left(y_i - \hat{y}_i\right) \tag{9}$$

The RSS value is then found by squaring each difference, and then summing them:

$$RSS = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \tag{10}$$

Notice how it is remarkably similar to MSE this metric is. The only difference is the MSE is an average value, thus scaled by $1/n$. RSS, is not scaled, this for each additional, sampled e_i^2, the value of RSS will always increase or remain the same, where MSE has the possibility to reduce.

2.4 Total Sum of Squares

Total sum of squares (TSS), is very similar to the variance of a data set. It is found by computing the sum of the differences squared between each index, and the average value of the data set. Unlike the variance, the TSS metric is not then scaled by the number of samples. Instead it is forces to grow with each successive index. It is defined as:

$$TSS = \sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2 \tag{11}$$

TSS can be though of as a measurement of the volatility of the output y_i, before some sort of regression is performed [4]. It is essentially a measurement in the total variance of some data set Y because the $\frac{1}{n}$ term in each variance is canceled by repeating the operation n times.

2.5 Residual Standard Error

In every regression algorithm, there is some error term, ϵ associated with each observation. Even with a perfectly accurate model, it is still impossible to perfectly predict Y given some input X [4]. The RSE estimates the standard deviation of this error term. It can be thought of as the average deviation of reach response from the regression fit. RSE is defined as:

$$RSE = \sqrt{\frac{1}{n-2}RSE} = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2} \tag{12}$$

The RSE benchmark measures a *lack* of fit for a model. The higher the value of the RSE, the less the fit truly models that data. Just like the RMSE (8), the RSE is measured in the same units as the response y. However, it is important to know that RSE leaves some ambiguity as to what a valid RSE threshold would be, especially because it doesn't have an upper bound value.

2.6 R-Squared

The R-squared (R^2, but NOT r^2) statistic is very similar to the RSE, except that is serves as more of a proportion of fit and is bounded to exist on the interval $[0, 1]$ [4]. R^2 is a dimensionless measure, and is the combination of the TSS (11) and RSS (10) benchmarks. It is defined as:

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS} \tag{13}$$

As described by Gareth James, the R^2 metric measures the propertion of variability in Y that can be explained using X [4] An R^2 value of 1 is indicative of a strong proportion of fit [6]. A value of 0 indicates that the model is not very well fitting of the data set.

3 Classification Metrics

A classification algorithm is a type of Machine Learning Algorithm that seeks to build a function f, using a set of data X, such that the result of the function is one of k classes [6]. In function notion:

$$f : X \to \{1, 2, ..., k-1, k\} \tag{14}$$

A classification algorithm takes some set of data, X, and using qualities or *features* of that data, places each element, x_i into a bin, called a *class* with other similar elements [2]. For each of the benchmarks described below, we will assume a sufficiently large data set has been used to produce some model $\hat{f}(X) \rightarrow \{1, 2, ..., k-1, k\}$.

3.1 K - Folds Cross Validation

K - Folds Cross Validation (also called K-Folds X-val) is a method that is part of a larger set of cross validation algorithms. K- Folds is a resampling procedure that allows for the evaluation of a certain model based on a limited set of data [1]. For the algorithm to work, it requires that there be a data set X, that is fully labeled, and can be broken down into k amount of subsets. The parameter k gives the algorithm the amount of sub groups that sample data is split into for the duration of it's execution. Very generally, the procedure of the algorithm is as follows:

1. Shuffle data set X as needed. Split the data set into k similarly sized subsets. $X \rightarrow \left[X_1, X_2, ...X_k\right]$.
2. For each set of data, X_n, save it as a testing subset. Use the remaining $k-1$ subsets as a the set of training data for the model.
3. Fit the model with $k-1$ subsets and evaluate. Hold the evaluation (can be percentage accuracy) and discard the particular model.
4. Repeat steps 2 and 3 for the remaining $k-1$ subsets.

After the procedure has been completed for all k subsets, the results can be amalgamated as desired. This method ensures that every sample in the entire original data set, X is used at least once as a testing element, and $k-1$ times as a training element. The exact value of k is contingent on the particular data set. No single value is useful for all types of models. Often, k should be chosen such that each subset is also statistically similar to the larger data set [2]. Additionally, the value $k = 10$ has been experimentally shown to produce generally low-biased estimates on the performance of a classifier [1].

it is important to recognize that percent accuracy alone is a very poor way of determining the validity of model [2]. K- Folds alone provides very little valuable information, and thus in often used in conjunction with other benchmark methods.

3.2 Confusion Matrix

A Confusion Matrix is a one of the most standardized tools in evaluating the performance of a classifier program. Suppose a classifier program has been constructed to place input samples into one of k classes. Thus, the confusion matrix, C, associated with that algorithm takes the form of a $k \times k$ matrix. The columns of that matrix represent predicted classes, and the rows represent actual classes [2]. A confusion matrix is generally assembled from the results of some sort of K-Folds Cross Validation Algorithm.

Each element of this matrix, $C_{i,j}$ indicated the number of samples predicted by the classifier to be in class k_i, but are actually labeled as being in class k_j. In the case where $i = j$ (the main diagonal entries), that indicates samples that were correctly predicted to be in the class that they are labeled as. This means that a confusion matrix with strong main diagonal entries indicates a classifier program that performs very well.

In the example case where the entry $C_{a,b}$ has a large value, then that is an indication that several samples are being placed into class. It also may be likely that element $C_{b,a}$ also has a similarly large entry. This means that classes a and b are commonly confused with each other.

3.3 Precision Score

The Precision score of a classification algorithm is most commonly applied to a *binary classifier*. Once a cross - validation score algorithm has been implemented to a binary classifier (With outputs True or False), It can be used to create a 2×2 confusion with the elements *True Positive*, *True Negative*, *False Positive* and *False Negative*, represented as TP,TN,FP,FN respectively [4]. The confusion matrix is then:

$$C = \begin{bmatrix} \text{True Positive} & \text{False Negative} \\ \text{False Positive} & \text{True Negative} \end{bmatrix} \tag{15}$$

Using these quantities, we can construct the *precision score* of the classifier:

$$P = \frac{TP}{TP + FP} = \frac{C_{1,1}}{C_{1,1} + C_{2,1}} \tag{16}$$

The value of P is bounded between 0 and 1. In short, the precision score is a measurement of *how many selected items are relevant* to the problem. Notice is deals only with the first column of the matrix C. The precision socre is a metric that determines the ratio between how many elements selected were correct (TP) to how many total elements were selected (TP + FP). It is the fraction of correct detection by the model [3].

In the case of a $K \neq 2$ folds classifier, the precision score definition would change based on the class of interest. Thus the numerator is then always along the main diagonal, $C_{n,n}$ and the denominator is the sum of all entries in column n. This once again is the measurement of how many of the items predicted to be in class n are in fact in class n. So the score for the n-th class would then be given by:

$$P_n = \frac{C_{n,n}}{\sum_{i=1}^{k} C_{i,n}} \tag{17}$$

3.4 Recall Score

The Recall score of of a classification algorithm is also mostly used with a binary classifier. Again, it begins by implementing some sort of cross - validation algorithm, and then constructing the results into a 2×2 confusion matrix.:

$$C = \begin{bmatrix} \text{True Positive} & \text{False Negative} \\ \text{False Positive} & \text{True Negative} \end{bmatrix} \tag{18}$$

Once again with these, we can compute the *recall score* of the classifier:

$$R = \frac{TP}{TP + FN} = \frac{C_{1,1}}{C_{1,1} + C_{1,2}} \tag{19}$$

Once again, the value of R is bounded between 0 and 1. The recall score is a a measurement of *how many relevant items are selected* in the problem. Just as precision deals only with the first column of the C matrix, the recall deals only with the first row of that same matrix. It is the ratio bwteen how many elements selected were correct (TP) to how many elements are labeled as such (TP + FN). It is the fraction of true events that were detected [3].

In the case of a $K \neq 2$ folds classifier, the recall score definition would change based on the class of interest just like the precision score did. Thus the numerator is again always along the main diagonal, $C_{n,n}$ and the denominator is the sum of all entries in row n. This is the measurement of how many of the items predicted to be in class n are relevant class n. So the score for the n-th class would then be given by:

$$P_n = \frac{C_{n,n}}{\sum_{i=1}^{k} C_{n,i}} \tag{20}$$

3.5 F1 Score

Precision and Recall are both valuable metrics of a binary classifier, but each one comes at the cost of the other. Thus a higher precision causes a lower recall and a higher recall causes a lower precision. This concept is called *precision - recall tradeoff* [2]. We can actually adjust parameters in our classifier to allow for the favoring of one measure over the other, but this presents a whole new set variables and is certainly dependent on the situation at hand. The $F1$ (may also be notated as F_1) score of a classifier allows for a sort of compromise between P and R. It is the harmonic mean between precision and recall [3]. It is defined as:

$$F1 = 2\left(\frac{P \times R}{P + R}\right) \tag{21}$$

Once again, the $F1$ score is bounded between 0 and 1. It is also the area underneath the curve created by mapping P as a function of R or vice-versa - this is called a $P - R$ curve [3]. The $F1$ score is high (closer to 1) when the algorithm has both a high precision and recall score, and a low $F1$ score when the two differ greatly.

References

[1] Brownlee, Jason. “A Gentle Introduction to k-Fold Cross-Validation.” *Machine Learning Mastery*, 8 Aug. 2019, machinelearningmastery.com/k-fold-cross-validation/.

[2] Géron Aurélien. *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O’Reilly, 2017.

[3] Goodfellow, Ian, et al.*Deep Learning*. MIT Press, 2017.

[4] James, Gareth, et al. *An Introduction to Statistical Learning with Applications in R*. Springer, 2017.

[5] Olsen, R.S., et al. 2017. ’PMLB: A Large Benchmarking Suite for Machine Learning Evaluation and Comparison’. *Biodata Mining*

[6] Petrik, Marek. “Introduction to Machine Learning.” Machine Learning. 22 Jan. 2020, Durham, New Hampshire.