MACHINE LEARNING

(Course Code: B18CS4101)

Lecture Notes

Training, Test and Validation sets

Dr. G.N.V.G. Sirisha
Dr. Ch. Someswararao
Sri. R. Shiva Shankar
Sri. V.V. Durga Kiran

Department of Computer Science and Engineering
Sagi RamaKrishnam Raju Engineeirng College
Bhimavaram, Andhrapradesh-534202

[Slide: 2]

Train, Validation and Test Datasets

To reiterate the findings from researching, this section provides unambiguous definitions of the three terms.

Training Dataset:

- The sample of data used to fit the model

  The actual dataset that we use to train the model (weights and biases in the case of a Neural Network). The model sees and learns from this data.

Validation Dataset:

- The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

The validation set is used to evaluate a given model, but this is for frequent evaluation. We have to use this data to fine-tune the model hyperparameters. Hence the model occasionally sees this data, but never does it "Learn" from this. We use the validation set results, and update higher level hyper parameters. So the validation set affects a model, but only indirectly.

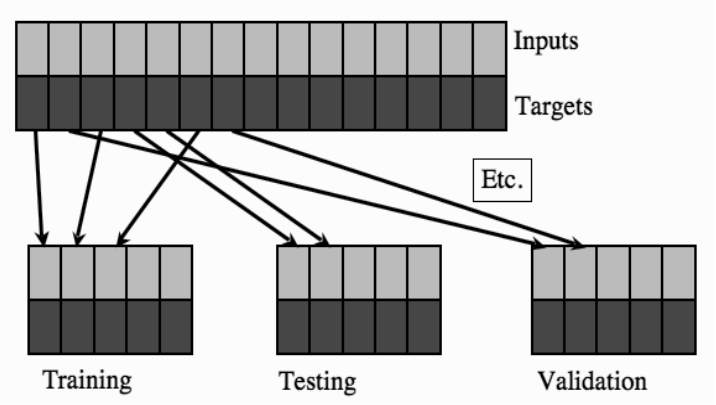
Test Dataset:

- The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

The Test dataset provides the gold standard used to evaluate the model. It is only used once a model is completely trained (using the train and validation sets). The test set is generally what is used to evaluate competing models (For example on many Kaggle competitions, the validation set is released initially along with the training set and the actual test set is only released when the competition is about to close, and it is the result of the model on the Test set that decides the winner). Many times the validation set is used as the test set, but it is not good practice. The test set is generally well curated. It contains carefully sampled data that spans the various classes that the model would face, when used in the real world.

[Slide: 3]

- The training set to actually train the algorithm
- The validation set to keep track of how well it is doing as it learns
- It is usually used for parameter selection and to avoid Overfitting.
- It is used for tuning the parameters of a model
- The test set to produce the final results



How to split your dataset into Train, Validation and Test sets.

It depends on 2 things. First, the total number of samples in your data and second, on the actual model you are training.

Some models need substantial data to train upon, so in this case you would optimize for the larger training sets. Models with very few hyperparameters will be easy to validate and tune, so you can probably reduce the size of your validation set, but if your model has many hyperparameters, you would want to have a large validation set.

Note on Cross Validation: Many a times, people first split their dataset into 2 — Train and Test. After this, they keep aside the Test set, and randomly choose X% of their Train dataset to be the actual **Train** set and the remaining (100-X)% to be the **Validation** set, where X is a fixed number(say 80%), the model is then iteratively trained and validated on these different sets. There are multiple ways to do this, and is commonly known as Cross Validation. Basically you use your training set to generate multiple splits of the Train and Validation sets. Cross validation avoids over fitting and is getting more and more popular, with K-fold Cross Validation being the most popular method of cross validation.

[Slide: 4 & 5]

Confusion Matrix

When we get the data, after data cleaning, pre-processing we measure the effectiveness of our model. Better the effectiveness, better the performance and that's exactly what we want. And it is where the Confusion matrix comes into the limelight.

- In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix.
- A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.
- It allows the visualization of the performance of an algorithm.
- It allows easy identification of confusion between classes e.g. one class is commonly mislabeled as the other.
- Most performance measures are computed from the confusion matrix.

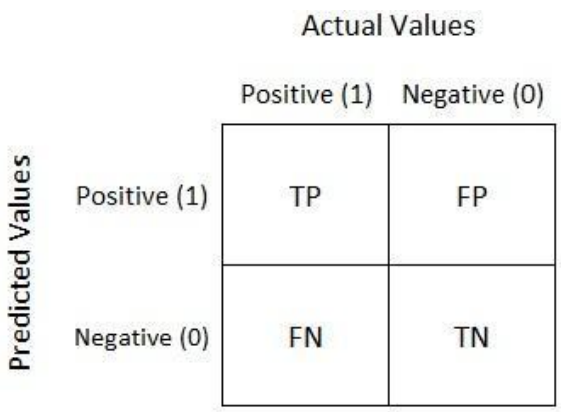
This topic aims at

1. What the confusion matrix is and why you need to use it.
2. How to calculate a confusion matrix for a 2-class classification problem from scratch.
3. How to create a confusion matrix in Python.

It is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.

- A confusion matrix is a summary of prediction results on a classification problem.
- The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.
- The confusion matrix shows the ways in which your classification model is confused when it makes predictions.
- It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

It is extremely useful for measuring Recall, Precision, Specificity, Accuracy and most importantly AUC-ROC Curve.



	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

Here,

Class 1: Positive

Class 2: Negative

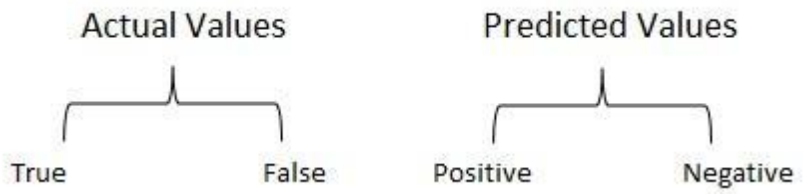
Definition of the Terms

- Positive (P): Observation is positive (for example: is an apple).
- Negative (N): Observation is not positive (for example: is not an apple).
- True Positive (TP): Observation is positive, and is predicted to be positive.
- False Negative (FN): Observation is positive, but is predicted negative.
- True Negative (TN): Observation is negative, and is predicted to be negative.
- False Positive (FP): Observation is negative, but is predicted positive.

Let's understand TP, FP, FN, TN in terms of pregnancy analogy.

True Positive:
You predicted that a woman is pregnant and she actually is.
True Negative:
You predicted that a man is not pregnant and he actually is not.
False Positive: (Type 1 Error)
You predicted that a man is pregnant but he actually is not.
False Negative: (Type 2 Error)
You predicted that a woman is not pregnant but she actually is.

Just Remember, We describe predicted values as Positive and Negative and actual values as True and False.



[Slide: 9 & 10]

Accuracy Metrics

Accuracy

- Classification Rate or Accuracy is given by the relation

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

- The problem with accuracy is that it doesn't tell us everything about the results, since it turns four numbers into just one.
- There are two complementary pairs of measurements that can help us to interpret the performance of a classifier, namely sensitivity and specificity, and precision and recall.

Sensitivity & Specificity

- Sensitivity (also known as the true positive rate or recall) is the ratio of the number of correct positive examples to the number classified as positive, while specificity is the same ratio for negative examples.

Sensitivity=TP/TP+FN
Specificity=TN/TN+FP

- **Sensitivity** measures how apt the model is to detecting events in the positive class.
- **Specificity** measures how exact the assignment to the positive class.

Recall

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

- Recall can be defined as the ratio of the total number of correctly classified positive examples divide to the total number of positive examples.
- **Recall** measures how good the model is in detecting positive events.
- High Recall indicates the class is correctly recognized (a small number of FN).
-

Out of all the positive classes, how much we predicted correctly. It should be high as possible.

[Slide: 11]

Precision

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

Out of all the positive classes we have predicted correctly, how many are actually positive.

- Divide the total number of correctly classified positive examples by the total number of predicted positive examples.
- High Precision indicates an example labeled as positive is indeed positive (a small number of FP).

- **Precision** measures how good the model is at assigning positive events to the positive class.
- Recall and precision are often reported pairwise because these metrics report the relevance of the model from two perspectives, also called type I error as measured by recall and type II error as measured by precision

High recall, low precision

- This means that most of the positive examples are correctly recognized (low FN) but there are a lot of false positives.

Low recall, high precision

- This shows that we miss a lot of positive examples (high FN) but those we predict as positive are indeed positive (low FP)

[Slide: 12]

F Measure

$$F\text{ - measure} = \frac{2*Recall*Precision}{Recall + Precision}$$

- We calculate an F-measure which uses Harmonic Mean in place of Arithmetic Mean as it punishes the extreme values more.
- The F-Measure will always be nearer to the smaller value of Precision or Recall.

It is difficult to compare two models with low precision and high recall or vice versa. So to make them comparable, we use F-Score. F-score helps to measure Recall and Precision at the same time. It uses Harmonic Mean in place of Arithmetic Mean by punishing the extreme values more.

I hope I've given you some basic understanding on what exactly is confusing matrix.