Classifier Evaluation

UNIVERSITY OF LIVERPOOL

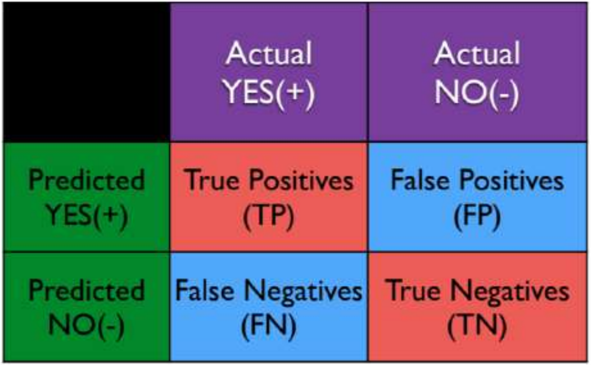
We have a classifier/model/system...

- How good is it?
- Levels of goodness
  - Absolute goodness: When we run our trained model on the "wild" it does what we expect it to do
    - no way of knowing *before* we deploy the model and when we do know, too late!
  - Relative goodness: We have a small representative sample of test data
    - We compare the output produced by our classifier on this test dataset (gold standard) and measure how well it resembles the labels in the dataset.

Gold Standard

- A dataset that we use for evaluation purpose. Also known as test data.
- Each test instance in the test data have their correct labels annotated.
- Numerous measures exist (as we shortly see) to **compare** the predicted labels by the trained classifier and actual (target) labels in the test dataset.
- Never train on test data!!!

Confusion Matrix



	Actual YES(+)	Actual NO(-)
Predicted YES(+)	True Positives (TP)	False Positives (FP)
Predicted NO(-)	False Negatives (FN)	True Negatives (TN)

Definitions

- True Positive
  - We predicted as positive and it is indeed positive
- True Negative
  - We predicted as negative and it is indeed negative
- False Positive
  - We predicted as positive but it turns out to be negative
- False Negative
  - We predicted as negative but it turns out to be positive

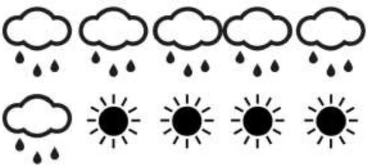
Detecting Cancer

- Let us assume we trained a classifier to detect cancer based on some features
- Predicting YES means we predict that the patient has cancer
- We predicted the patient as having cancer but further tests revealed that the patient does not have cancer.
    - False Positive
- We predicted the patient as not having cancer (so no further tests were done) but the patient died with cancer!
    - False Negative
- The moral of the story: FP and FN have very different importance in real-world data mining tasks.

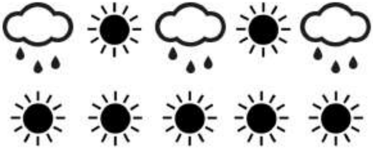
Evaluation Measures

- (TP + TN) / (TP + TN + FP +FN)
- Precision (P): Precision tells us about all the correct predictions out of total positive predictions
    - P = TP / (TP + FP)
- Recall (R): Recall tells us about how many instances were correctly classified out of all positive instances
    - R = TP / (TP + FN)
- F-score = (2 x Precision x Recall) / (Precision + Recall)

Quiz



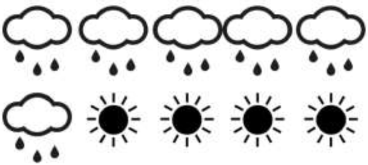
Actual Weather for 10 days



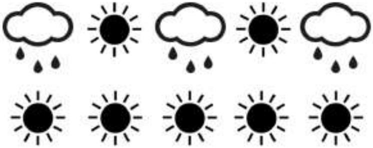
Model's prediction for 10 days

Can you create a confusion matrix and compute Precision, Recall and Accuracy?

Quiz



Actual Weather for 10 days

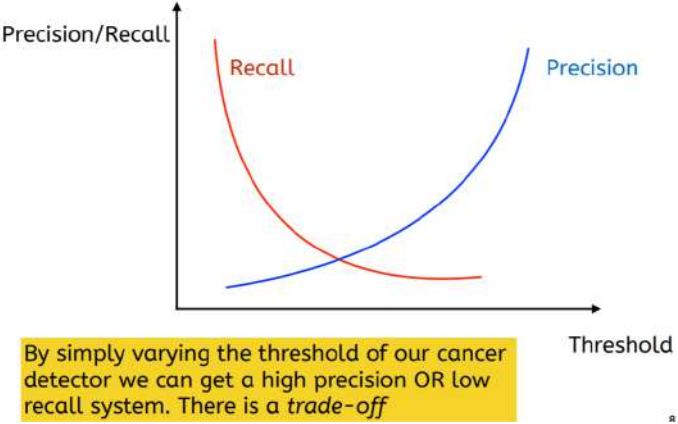


Model's prediction for 10 days

	Actual (sunny)	Actual (rainy)
Predicted (sunny)	4 (TP)	3 (FP)
Predicted (rainy)	0 (FN)	3 (TN)

P = TP/(TP+FP) = 4/7 = 57%
R = TP/(TP+FN) = 4/0 = 100%
A = (TP+TN)/(TP+FP+FN+TN)
= 70%
F = (2*0.57*1)/(0.57+1) = 72 %

Precision-Recall Trade-off



By simply varying the threshold of our cancer detector we can get a high precision OR low recall system. There is a *trade-off*

Harmonic Mean

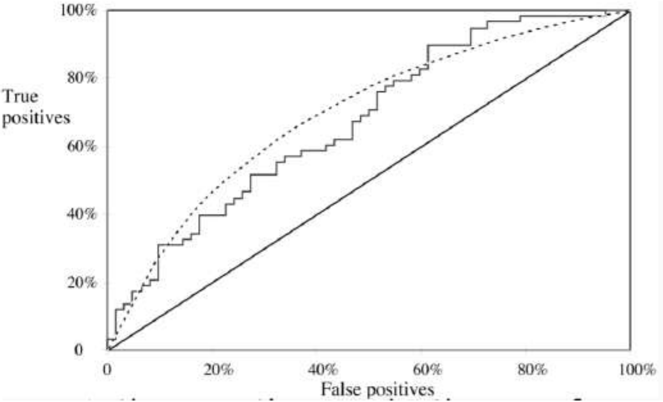
- Harmonic mean is a good way to compute an average when there is an inverse relation between two variables.
- F-score (F) is the harmonic mean between Precision (P) and Recall (R).

$$F = \frac{1}{\frac{\frac{1}{P}+\frac{1}{R}}{2}} = \frac{2PR}{P+R}$$

ROC Curves

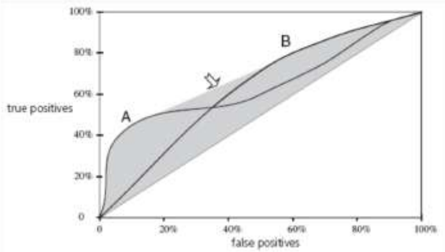
- Receiver Operating Characteristic
  - Originally a concept borrowed from signal processing
  - Tradeoff between hit rate and false alarm rate when trying to find real data in noisy channel.
- Plot true positive vertically, and false positives horizontally.
- The place to be is on the top left (high TP and low FP)
- Generate a series of models operating at various threshold values, measure TP and FP for each model to plot ROC curve
- We can generate a smooth ROC curve by the of cross-validation (discussed later) by generating a curve for each fold, and then averaging them.

ROC


100%
80%
60%
40%
20%
0
True positives
0
20%
40%
60%
80%
100%
False positives

Comparing Classifiers using ROC

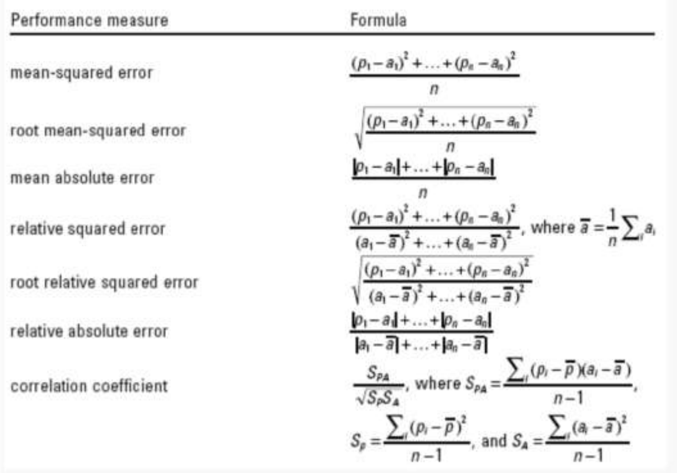
- We can also plot two curves on the same chart, each generated from different classifiers. This lets us see at which point it is better to use one classifier rather than the other.
- By using both A and B classifiers with appropriate weightings, it is possible to get at points in between the two peaks



Evaluation measures for regression

- What if we are predicting real numbers instead of class labels?
    - This setting is called regression
- It is too harsh to consider non-exact matches as incorrect predictions
- For an instance x, our trained model predicts the output to be 1.001, but the actual value is 1.
    - Is this a correct prediction or an incorrect prediction
    - We are off by just 0.001

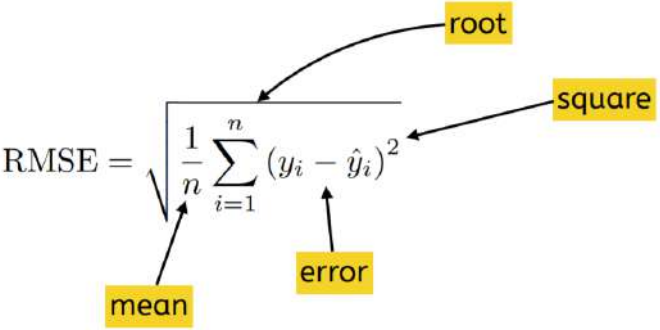
Evaluation Measures for Regression



Performance measure	Formula
mean-squared error	$\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{n}$
root mean-squared error	$\sqrt{\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{n}}$
mean absolute error	$\frac{\lvert p_1-a_1\rvert+\ldots+\lvert p_n-a_n\rvert}{n}$
relative squared error	$\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{(a_1-\bar{a})^2+\ldots+(a_n-\bar{a})^2}$, where $\bar{a}=\frac{1}{n}\sum_i a_i$
root relative squared error	$\sqrt{\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{(a_1-\bar{a})^2+\ldots+(a_n-\bar{a})^2}}$
relative absolute error	$\frac{\lvert p_1-a_1\rvert+\ldots+\lvert p_n-a_n\rvert}{\lvert a_1-\bar{a}\rvert+\ldots+\lvert a_n-\bar{a}\rvert}$
correlation coefficient	$\frac{S_{PA}}{\sqrt{S_P S_A}}$, where $S_{PA}=\frac{\sum_i(p_i-\bar{p})(a_i-\bar{a})}{n-1}$, $S_p=\frac{\sum_i(p_i-\bar{p})^2}{n-1}$, and $S_A=\frac{\sum_i(a_i-\bar{a})^2}{n-1}$

Root Mean Square Error

- A popular measure for evaluating numerical (real/ordinal) regression models is the *root mean square error* (RMSE) defined as follows:



$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

root

square

mean

error

Robust Evaluation

Case Study

- We have a single train dataset. We do not have a separate test dataset. We want to evaluate a classification algorithm as **reliably** as possible using this train dataset. How should we go about this?

A Bad Idea

- Train using ALL the train data
- Evaluate on ALL the train data
- Likely to overfit!
  - We might get very impressive train performance but we have no idea how this classification model is going to perform on UNSEEN test data.
  - If the classifier simply REMEMBERS every instance in the train dataset, it will get 100% accuracy under this evaluation scheme
- Bad idea!

Cross-Validation

- We already discussed validation datasets under *how to set the hyper-parameters of a classifier* (see k-NN lecture notes)
- set aside a portion of **train** data for validation purposes

Cross-Validation (contd.)

- For example, use 1/5-th of train data for validation. This is called a **fold** of the dataset.
  - Train using the remaining 4/5-th and evaluate on the held-out 1/5th.
  - Repeat this process 5 times, each time selecting a different fold.
  - Evaluate the performance (using any evaluation measure we discussed so far) on the held-out data (1/5-th not used in training)
  - Take the average of the five numbers
  - This is called **5-fold cross-validation**

Cross-Validation (contd.)

- In N-fold cross-validation, we split the dataset into N equal partitions (folds), repeat the process N times, and take the average of the N evaluation measure values.

Issues

- Cross-validation is a trade-off between speed and the train dataset size
- Increase the number of folds:
    - More data to train from
    - Better estimates (averaging over many folds)
    - Slow (many re-trains)

Issues

- If we are performing cross-validation to set hyper-parameters, then it could be the case that a different value of the hyper-parameter is producing the best results in different folds! No clear winner to select
  - Select the best fold (max instead of avg)

Leave-One-Out Cross-Validation

- Leave only one of the train instances in each fold of validation
- train dataset size = (N-1) instances
- validation dataset size = 1 instance
- Repeat the process N (=dataset size) times
- Pros
    - We have lots of data to train from
- Cons
    - Very slow in practice because we have to re-train our classifier N-1 number of times