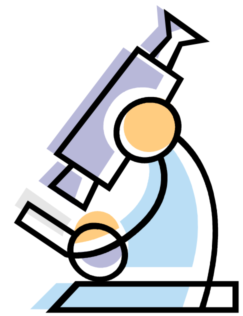
Evaluation Metrics & Methodology

Why evaluation?

- When a learning system is deployed in the real world, we need to be able to quantify the performance of the classifier
  - How accurate will the classifier be
  - When it is wrong, why is it wrong?
- This is very important as it is useful to decide which classifier to use in which situations

Evaluating ML Algorithms



- Empirical Studies
  - Correctness on **novel** examples (inductive learning)
  - Time spent learning
  - Time needed to apply result learned
  - Speedup after learning (explanation-based learning)
  - Space required
- <u>Basic idea</u>: repeatedly use <u>train/test</u> sets to estimate future accuracy

Proper Experimental Methodology Can Have a Huge Impact!

A 2002 paper in *Nature* (a major, major journal) needed to be corrected due to "training on the testing set"

Most important "thou shall not"

Original report : 95% accuracy (5% error rate)

Corrected report (which still is buggy):
73% accuracy (27% error rate)

Error rate increased over 400%!!!

Training and Test sets

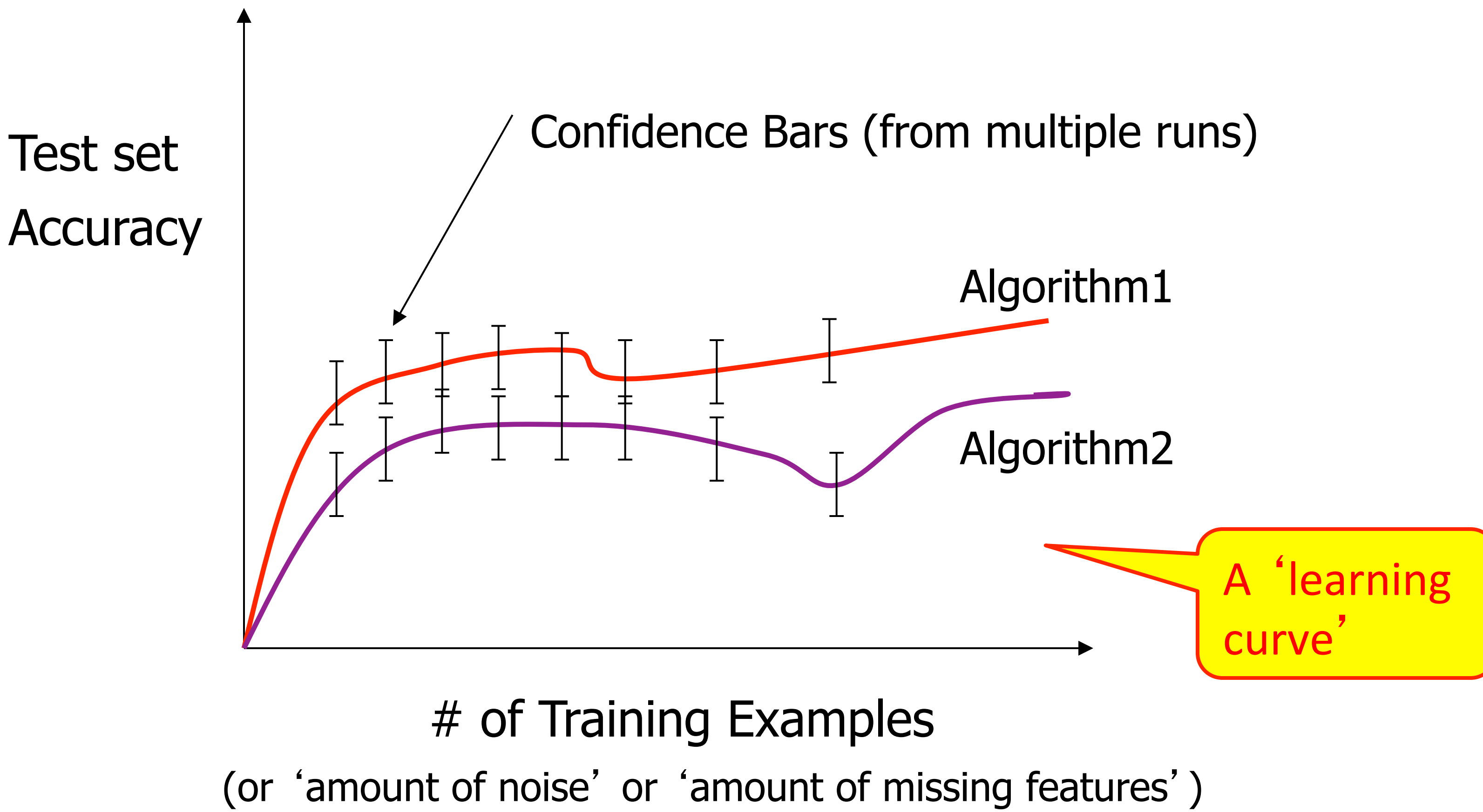
Split the available data into a training set and a test set

Train the classifier on the training set and evaluate on the test set

Classifier Accuracy

- The accuracy of a classifier on a given test set is the percentage of test set examples that are correctly classified by the classifier
  - Accuracy = (# correct classifications)/ (Total # of examples)
  - Error rate is the opposite of accuracy
  - Error rate = 1 - Accuracy

Some Typical ML Experiments – Empirical Learning



Some Typical ML Experiments – “Lesion” Studies

	Testset Performance
Full System	80%
Without Module A	75%
Without Module B	62%
⋮	⋮

Learning from Examples: Standard Methodology for Evaluation

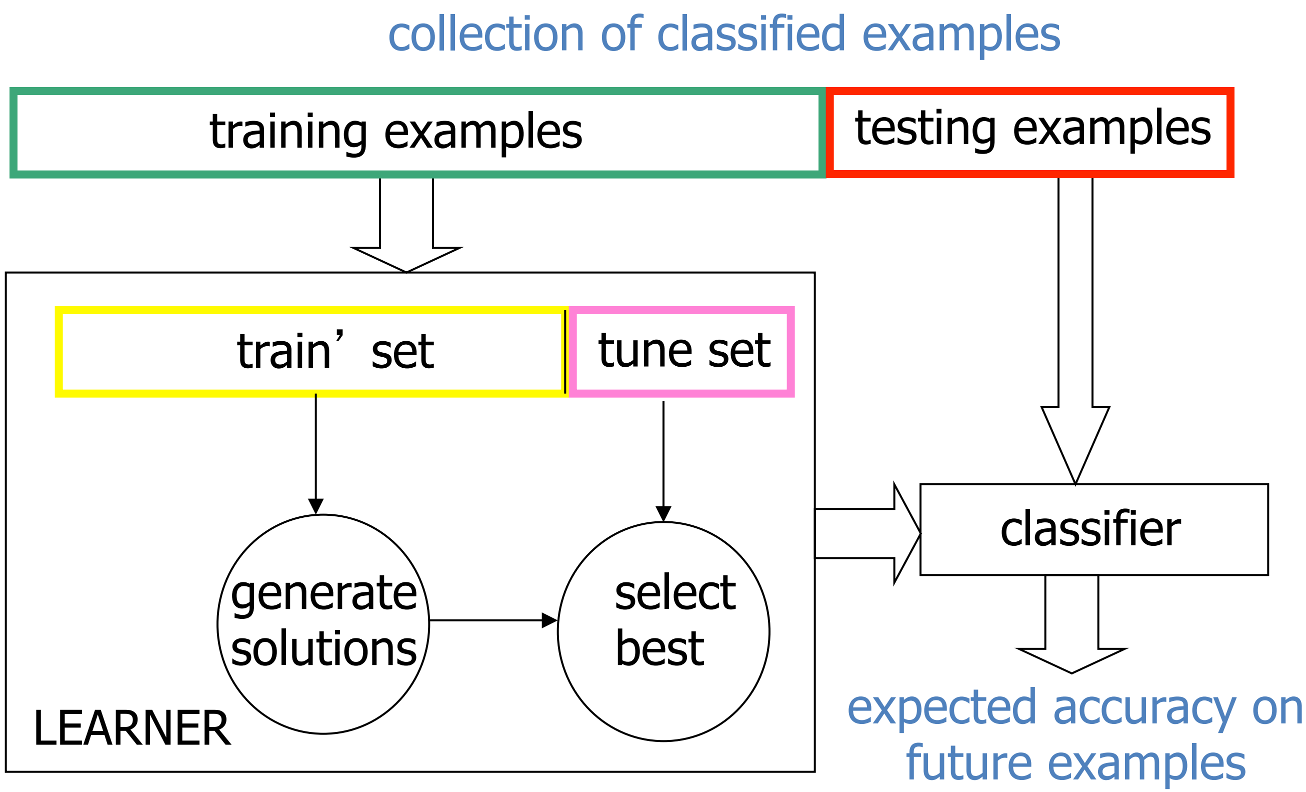
1) Start with a dataset of labeled examples
2) Randomly partition into *N* groups
3a) *N* times, combine *N* -1 groups into a train set
3b) Provide <u>train set</u> to learning system
3c) Measure accuracy on "left out" group (the <u>test set</u>)

Called *N*-fold cross validation (typically *N* =10)

Using Tuning Sets

- Often, an ML system has to choose when to stop learning, select among alternative answers, etc.
- One wants the model that produces the highest accuracy on **future** examples ("overfitting avoidance")
- It is a "**cheat**" to look at the **test** set while still learning
- Better method
  - Set aside part of the training set
  - Measure performance on this "tuning" data to estimate future performance for a given set of parameters
  - Use best parameter settings, train with **all** training data (except **test** set) to estimate future performance on **new** examples

Experimental Methodology: A Pictorial Overview



Statistical techniques such as 10-fold cross validation and *t*-tests are used to get meaningful results

Parameter Setting

Notice that each train/test fold may get <u>different</u> parameter settings!

That's fine (and proper)

I.e. , a "parameterless"* algorithm internally sets parameters for **each data set** it gets

* Usually, though, some parameters have to be externally fixed (e.g. knowledge of the data, range of parameter settings to try, etc)

Using Multiple Tuning Sets

Using a **single** tuning set can be unreliable predictor, plus some data "wasted."
Hence, often the following is done:

1) For each possible set of parameters
   a) Divide <u>training</u> data into **train'** and **tune** sets, using ***N*-fold cross validation**
   b) Score this set of parameter values: average **tune** set accuracy over the *N* folds
2) Use **best** set of parameter settings and **<u>all</u> (train' + tune)** examples
3) Apply resulting model to **test** set

Example

Expected	Predicted
Y	Y
N	Y
Y	Y
Y	Y
N	N
N	N
Y	Y
Y	N
N	N
Y	Y
N	Y
Y	Y
Y	N
N	N
Y	Y
N	N
Y	Y
Y	N
N	Y
N	N

False Positives & False Negatives

- Sometimes accuracy is not sufficient
- If 98% of examples are negative (for a disease), the classifying everyone as negative can get an accuracy of 98%
- When is the model wrong?
  - False positives and false negatives
- Often there is a cost associated with false positives and false negatives
  - Diagnosis of diseases
  - Sometimes better safe than sorry

Confusion Matrix

- Is a device used to illustrate how a model is performing in terms of false positives and false negatives
- It gives us more information than a single accuracy figure
- It allows us to think about the cost of mistakes
- It can be extended to any number of classes

Confusion Matrix

Model Result

A Non-Lapsed	B Lapsed		
True Positive (TP)	False Negative (FN)	A Non-Lapsed	**Expected Result**
False Positive (FP)	True Negative (TN)	B Lapsed	

Accuracy Measures

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

$$Misclassification\,Rate = \frac{FP + FN}{TP + FP + TN + FN}$$

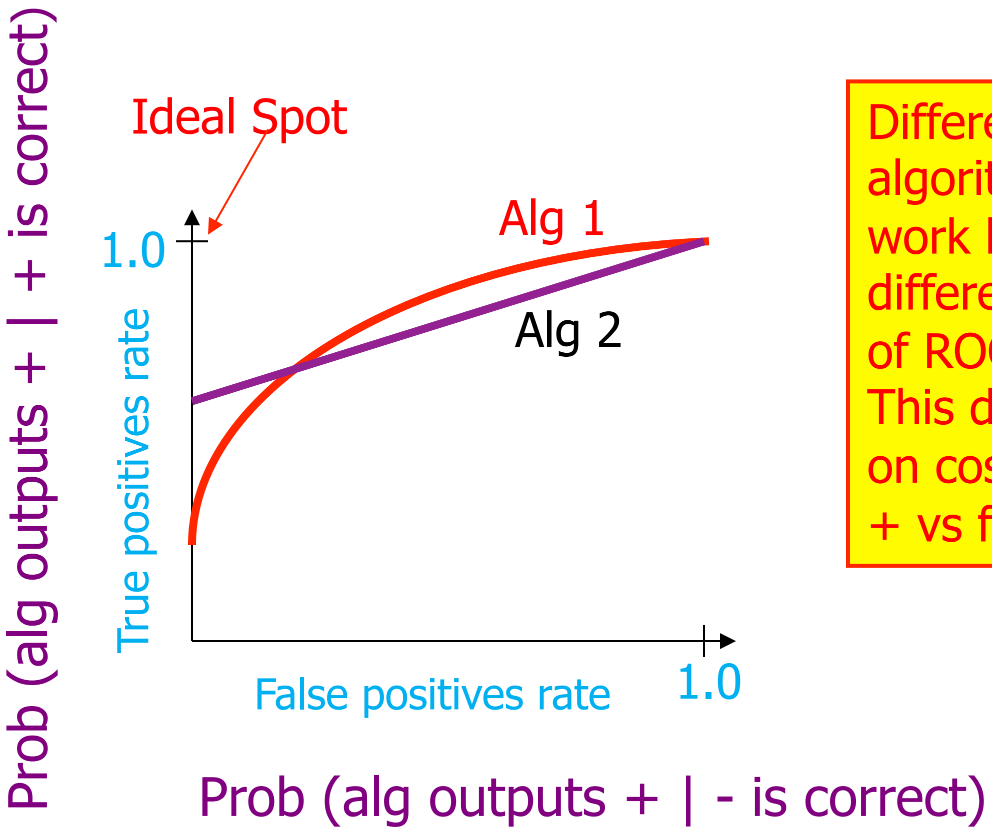
$$True\,Positive\,Rate\,(sensitivity) = \frac{TP}{TP + FN}$$

$$True\,Negative\,Rate\,(specificity) = \frac{TN}{TN + FP}$$

ROC Curves

- ROC: *Receiver Operating Characteristics*
- Started for radar research during WWII
- Judging algorithms on accuracy alone may not be good enough when <u>getting a positive wrong costs</u> more than <u>getting a negative wrong</u> (or vice versa)
  - Eg, medical tests for serious diseases
  - Eg, a movie-recommender (ala' NetFlix) system

ROC Curves Graphically



Different algorithms can work better in different parts of ROC space. This depends on cost of false + vs false -

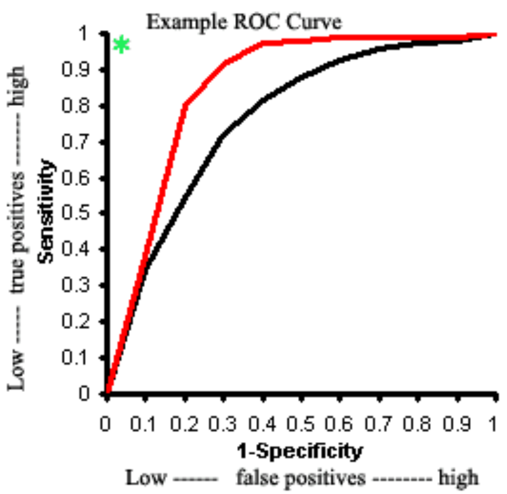
Algo for Creating ROC Curves

Step 1: Sort predictions on test set

Step 2: Locate a *threshold* between examples with opposite categories

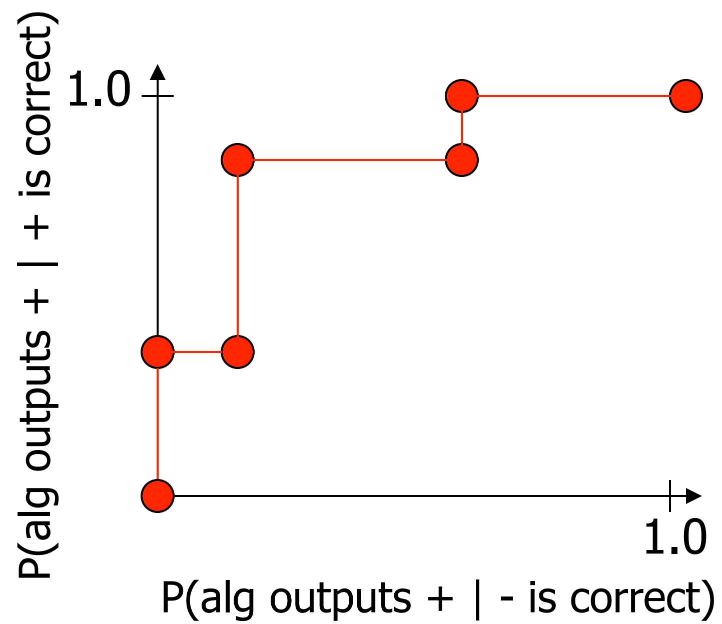
Step 3: Compute TPR & FPR for each threshold of Step 2

Step 4: Connect the dots



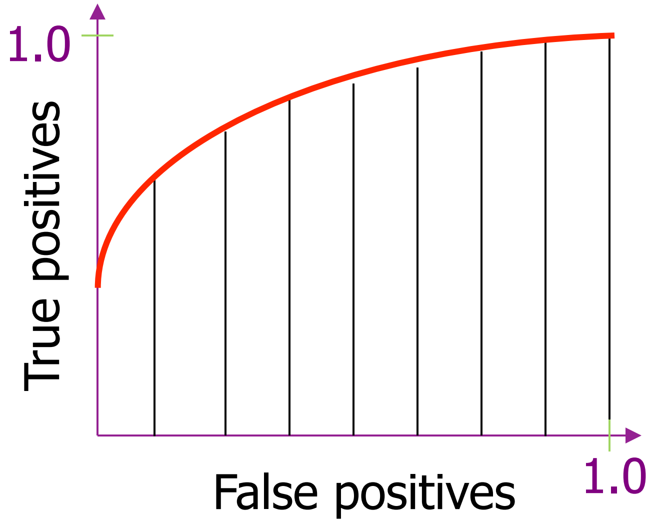
Plotting ROC Curves - Example

ML Algo Output (Sorted)			Correct Category
Ex 9	.99		+
Ex 7	.98	TPR=(2/5), FPR=(0/5)	+
Ex 1	.72	TPR=(2/5), FPR=(1/5)	-
Ex 2	.70		+
Ex 6	.65	TPR=(4/5), FPR=(1/5)	+
Ex 10	.51		-
Ex 3	.39	TPR=(4/5), FPR=(3/5)	-
Ex 5	.24	TPR=(5/5), FPR=(3/5)	+
Ex 4	.11		-
Ex 8	.01	TPR=(5/5), FPR=(5/5)	-



Area Under ROC Curve

A common metric for experiments is to numerically integrate the ROC Curve



Asymmetric Error Costs

- Assume that cost(FP) ≠ cost(FN)
- You would like to pick a threshold that mimimizes

  E(total cost) =

  cost(FP) x prob(FP) x (# of neg ex’s) +

  cost(FN) x prob(FN) x (# of pos ex’s)
- You could also have (maybe negative) costs for TP and TN (assumed zero in above)

Precision vs. Recall

(think about search engines)

- **Precision** = (# of relevant items retrieved) / (total # of items retrieved)

  = TP / (TP + FP)

  ≅ P(is pos | called pos)

- **Recall** = (# of relevant items retrieved) / (# of relevant items that exist)

  = TP/(TP+FN) = TPR

  ≅ P(called pos | is pos)

- Notice that n(0,0) is not used in either formula
  Therefore you get no credit for filtering out irrelevant items