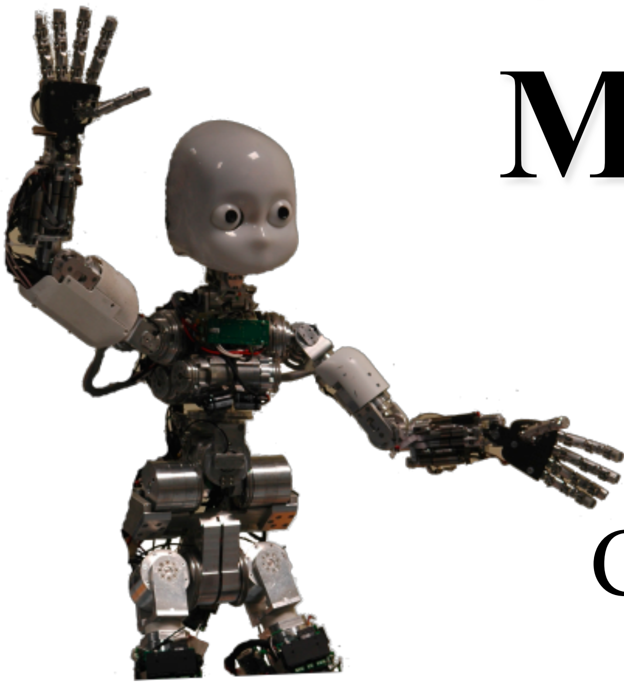
Machine Learning: Methodology



Chapter 18.1-18.3

Some material adopted from notes by Chuck Dyer

UCI Machine Learning Repos ×

archive.ics.uci.edu/ml/

http://archive.ics.uci.edu/ml

About Citation Policy Donate a Data Set Contact

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

Google™ Custom Search | Search

View ALL Data Sets

Welcome to the UC Irvine Machine Learning Repository!

We currently maintain 233 data sets as a service to the machine learning community. You may **view all data sets** through our searchable interface. Our old web site is still available, for those who prefer the old format. For a general overview of the Repository, please visit our About page. For information about citing data sets in publications, please read our citation policy. If you wish to donate a data set, please consult our donation policy. For any other questions, feel free to contact the Repository librarians. We have also set up a mirror site for the Repository.

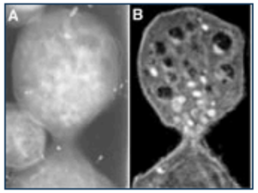
Supported By: In Collaboration With: Rexa.info • Research • People • Connections

233 data sets

Latest News:

2010-03-01:	Note from donor regarding Netflix data
2009-10-16:	Two new data sets have been added.
2009-09-14:	Several data sets have been added.
2008-07-23:	Repository mirror has been set up.
2008-03-24:	New data sets have been added!
2007-06-25:	Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope
2007-04-13:	Research papers that cite the repository have been associated to specific data sets.

Featured Data Set: Yeast



Task: Classification
Data Type: Multivariate
 # Attributes: 8
 # Instances: 1484

Predicting the Cellular Localization Sites of Proteins

Newest Data Sets:

2012-10-21:	QtyT40I10D100K
2012-10-19:	Legal Case Reports
2012-09-29:	seeds
2012-08-30:	Individual household electric power consumption
2012-08-15:	Northix
2012-08-06:	PAMAP2 Physical Activity Monitoring
2012-08-04:	Restaurant & consumer data
2012-08-03:	CNAE-9

Most Popular Data Sets (hits since 2007):

386214:	Iris
272233:	Adult
237503:	Wine
195947:	Breast Cancer Wisconsin (Diagnostic)
182423:	Car Evaluation
151635:	Abalone
135419:	Poker Hand
113024:	Forest Fires

UCI Machine Learning Repository: Zoo Data Set

http://archive.ics.uci.edu/ml/datasets/Zoo

About Citation Policy Donate a Data Set Contact

Search

Repository Web

Google

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

View ALL Data Sets

Zoo Data Set

Download: Data Folder, Data Set Description

Abstract: Artificial, 7 classes of animals

http://archive.ics.uci.edu/ml/datasets/Zoo

Data Set Characteristics:	Multivariate	**Number of Instances:**	101	**Area:**	Life
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	17	**Date Donated**	1990-05-15
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	18038

Zoo data

animal name: string
hair: Boolean
feathers: Boolean
eggs: Boolean
milk: Boolean
airborne: Boolean
aquatic: Boolean
predator: Boolean
toothed: Boolean
backbone: Boolean
breathes: Boolean
venomous: Boolean
fins: Boolean
legs: {0,2,4,5,6,8}
tail: Boolean
domestic: Boolean
catsize: Boolean
type: {mammal, fish, bird, shellfish, insect, reptile, amphibian}

101 examples

aardvark,1,0,0,1,0,0,1,1,1,1,0,0,4,0,0,1,mammal
antelope,1,0,0,1,0,0,0,1,1,1,0,0,4,1,0,1,mammal
bass,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,fish
bear,1,0,0,1,0,0,1,1,1,1,0,0,4,0,0,1,mammal
boar,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1,mammal
buffalo,1,0,0,1,0,0,0,1,1,1,0,0,4,1,0,1,mammal
calf,1,0,0,1,0,0,0,1,1,1,0,0,4,1,1,1,mammal
carp,0,0,1,0,0,1,0,1,1,0,0,1,0,1,1,0,fish
catfish,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,fish
cavy,1,0,0,1,0,0,0,1,1,1,0,0,4,0,1,0,mammal
cheetah,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1,mammal
chicken,0,1,1,0,1,0,0,0,1,1,0,0,2,1,1,0,bird
chub,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,fish
clam,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,shellfish
crab,0,0,1,0,0,1,1,0,0,0,0,0,4,0,0,0,shellfish
…

Zoo example

```
aima-python> python
>>> from learning import *
>>> zoo
<DataSet(zoo): 101 examples, 18 attributes>
>>> dt = DecisionTreeLearner()
>>> dt.train(zoo)
>>> dt.predict(['shark',0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0])
'fish'
>>> dt.predict(['shark',0,0,0,0,0,1,1,1,1,0,0,1,0,1,0,0])
'mammal'
```

Evaluation methodology (1)

Standard methodology:

1. Collect large set of examples with correct classifications
2. Randomly divide collection into two disjoint sets: *training* and *test*
3. Apply learning algorithm to training set giving hypothesis H
4. Measure performance of H w.r.t. test set

Evaluation methodology (2)

- Important: keep the training and test sets disjoint!
- Study efficiency & robustness of algorithm: repeat steps 2-4 for different training sets & training set sizes
- On modifying algorithm, restart with step 1 to avoid evolving algorithm to work well on just this collection

Evaluation methodology (3)

Common variation on methodology:

1. Collect large set of examples with correct classifications
2. Randomly divide collection into two disjoint sets: *developmen*t and *test,* and further divide development into *devtrain* and *devtest*
3. Apply learning algorithm to *devtrain* set giving hypothesis H
4. Measure performance of H w.r.t. *devtest* set
5. Modify approach, repeat 3-4 ad needed
6. Final test on *test* data

Zoo evaluation

train_and_test(learner, data, start, end) uses data[start:end] for test and the rest for train

```
>>> dtl = DecisionTreeLearner
>>> train_and_test(dtl(), zoo, 0, 10)
1.0
>>> train_and_test(dtl(), zoo, 90, 100)
0.80000000000000004
>>> train_and_test(dtl(), zoo, 90, 101)
0.81818181818181823
>>> train_and_test(dtl(), zoo, 80, 90)
0.90000000000000002
```

K-fold Cross Validation

- Problem: getting “ground truth” data can be expensive
- Problem: ideally need different test data each time we test
- Problem: experimentation is needed to find right “feature space” and parameters for ML algorithm
- Goal: minimize needed training+test data needed
- Idea: split training data into K subsets, use K-1 for *training*, and one for *development testing*
- Common K values are 5 and 10

Zoo evaluation

cross_validation(learner, data, K, N) does N iterations, each time randomly selecting 1/K data points for test, rest for train

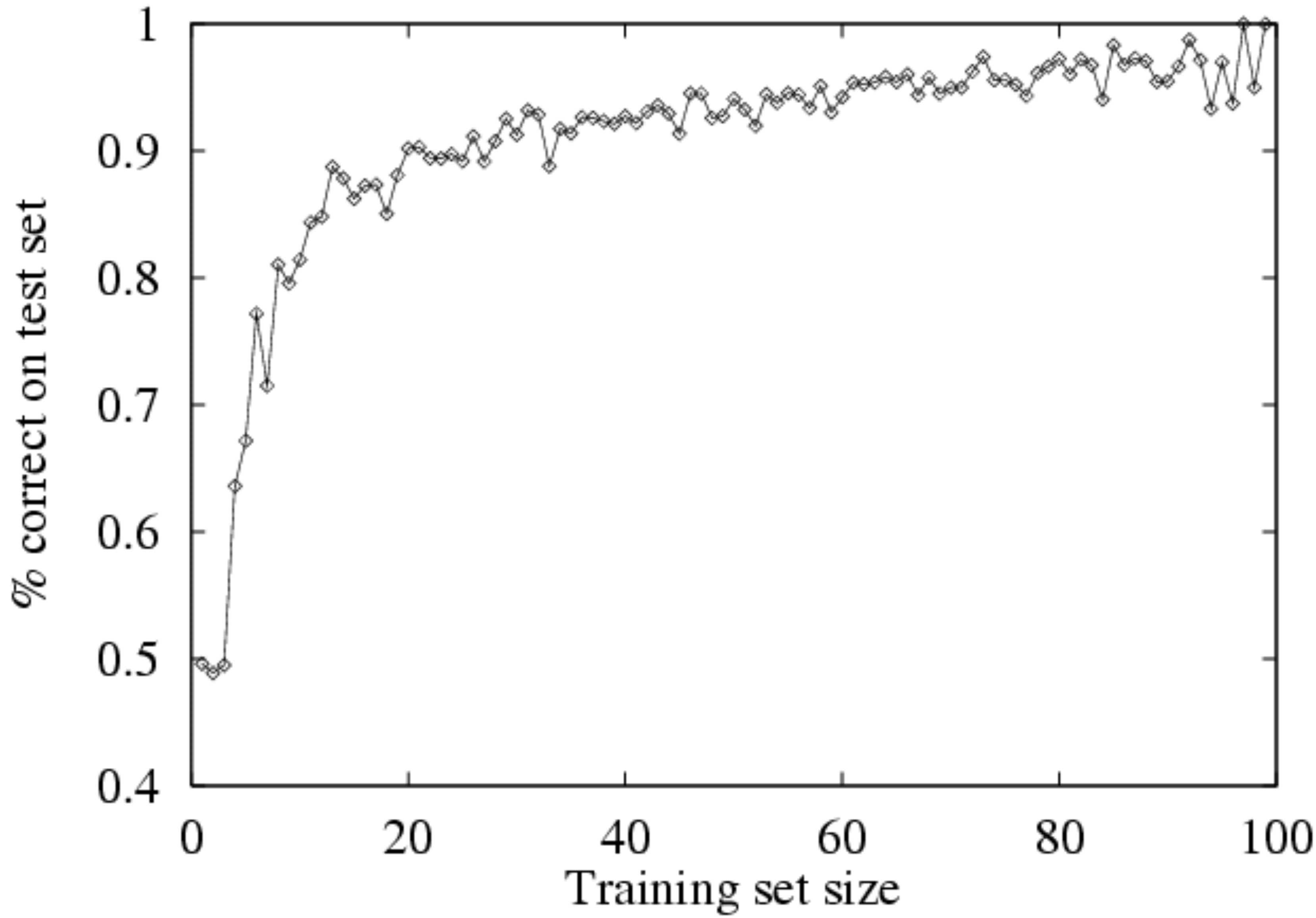
```
>>> cross_validation(dtl(), zoo, 10, 20)
0.95500000000000007
```

leave1out(learner, data) does len(data) trials, each using one element for test, rest for train

```
>>> leave1out(dtl(), zoo)
0.97029702970297027
```

Learning curve

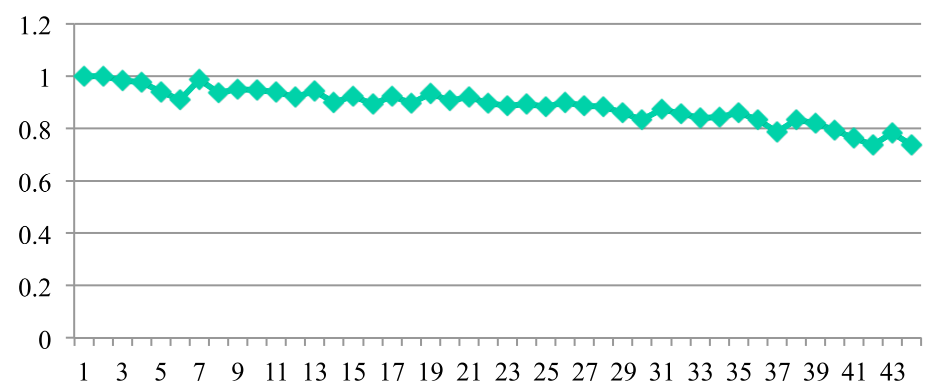
Learning curve = % correct on test set as a function of training set size



Zoo

>>> learningcurve(DecisionTreeLearner(), zoo)

[(2, 1.0), (4, 1.0), (6, 0.98333333333333339), (8, 0.97499999999999998), (10, 0.94000000000000006), (12, 0.90833333333333321), (14, 0.98571428571428577), (16, 0.9375), (18, 0.94999999999999996), (20, 0.94499999999999995), … (86, 0.78255813953488373), (88, 0.73636363636363644), (90, 0.70777777777777795)]



UCI Machine Learning Repos

archive.ics.uci.edu/ml/datasets/Iris

About Citation Policy Donate a Data Set Contact

Search

Repository Web

Google™

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

View ALL Data Sets

Iris Data Set

Download: Data Folder, Data Set Description

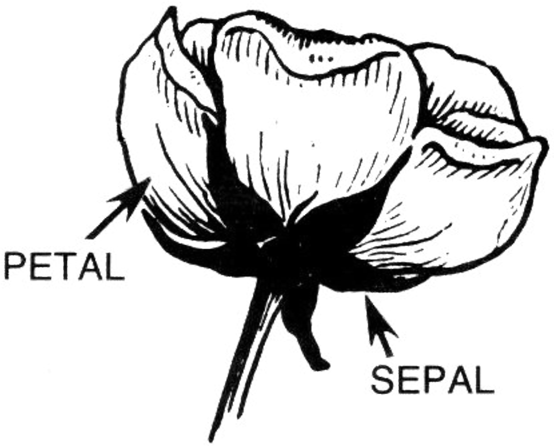
Abstract: Famous database; from Fisher, 1936

http://archive.ics.uci.edu/ml/datasets/Iris

Data Set Characteristics:	Multivariate	**Number of Instances:**	150	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	4	**Date Donated**	1988-07-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	386237

Source:

Iris Data



- Three classes: Iris Setosa, Iris Versicolour, Iris Virginica
- Four features: sepal length and width, petal length and width
- 150 data elements (50 of each)

aima-python> more data/iris.csv
5.1,3.5,1.4,0.2,setosa
4.9,3.0,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
4.6,3.1,1.5,0.2,setosa
5.0,3.6,1.4,0.2,setosa

http://code.google.com/p/aima-data/source/browse/trunk/iris.csv

Comparing ML Approaches

- The effectiveness of ML algorithms varies depending on the problem, data and features used
- You may have intuitions, but run experiments
- Average accuracy (% correct) is a standard metric

```
>>> compare([DecisionTreeLearner, NaiveBayesLearner,
NearestNeighborLearner], datasets=[iris, zoo], k=10, trials=5)
```

	iris	zoo
DecisionTree	0.86	0.94
NaiveBayes	0.92	0.92
NearestNeighbor	0.85	0.96

Confusion Matrix (1)

- A confusion matrix can be a better way to show results
- For binary classifiers it's simple and is related to type I and type II errors (i.e., false positives and false negatives)
- There may be different costs for each kind of error
- So we need to understand their frequencies

predicted

actual

a/c	C	~C
C	True positive	False negative
~C	False positive	True negative

Confusion Matrix (2)

- For multi-way classifiers, a confusion matrix is even more useful
- It lets you focus in on where the errors are

predicted

actual

	Cat	Dog	rabbit
Cat	5	3	0
Dog	2	3	1
Rabbit	0	2	11

Accuracy, Error Rate, Sensitivity, Specificity

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

- **Classifier Accuracy,** or recognition rate: percentage of test set tuples that are correctly classified

  Accuracy = (TP + TN)/All

- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/All

- **Class Imbalance Problem**:
  - One class may be *rare*, e.g. fraud, HIV-positive, ebola
  - Significant *majority of the negative class* and minority of the positive class
  - **Sensitivity**: True Positive recognition rate
    - **Sensitivity = TP/P**
  - **Specificity**: True Negative recognition rate
    - **Specificity = TN/N**

Precision and Recall

Information retrieval uses precision and recall to characterize retrieval effectiveness

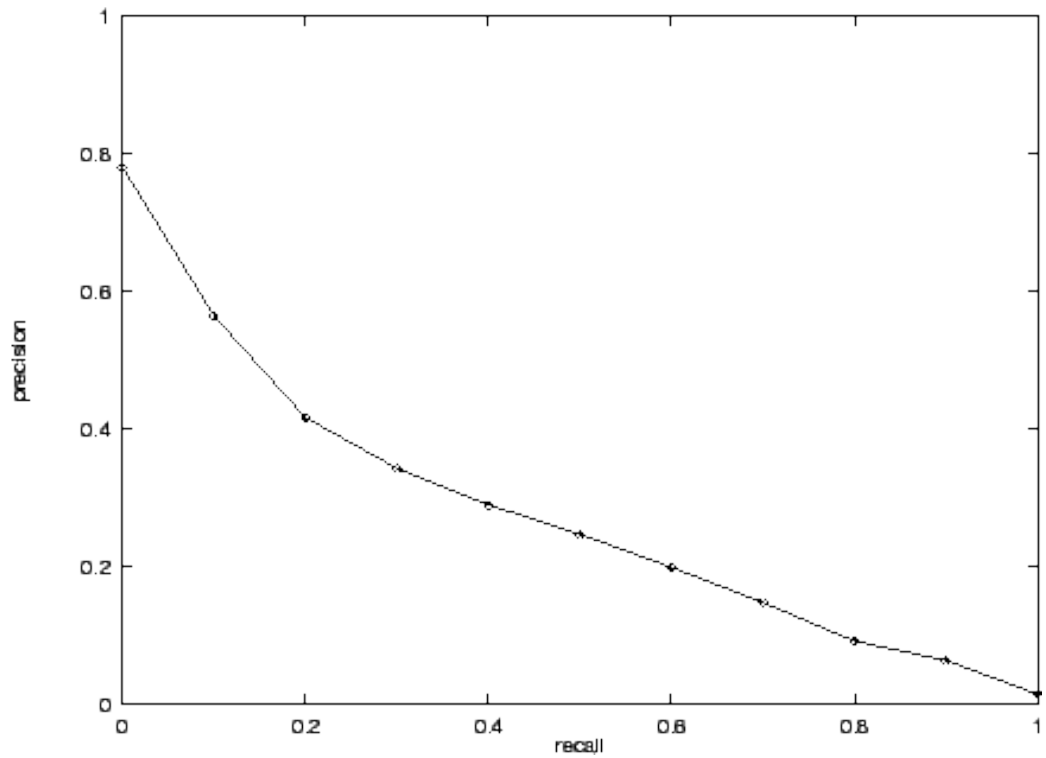
- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive
- **Recall:** completeness – what % of positive tuples did the classifier label as positive?

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

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

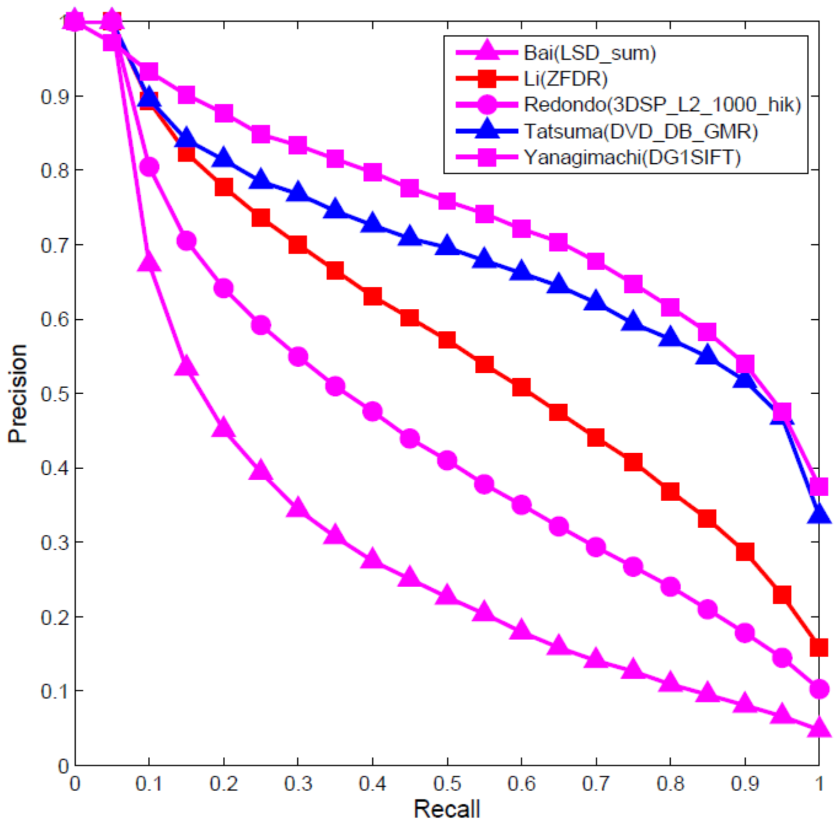
Precision and Recall

- In general, increasing one causes the other to decrease
- Studying the precision recall curve is informative



Precision and Recall

If one system's curve is always above the other, it's better



F measure

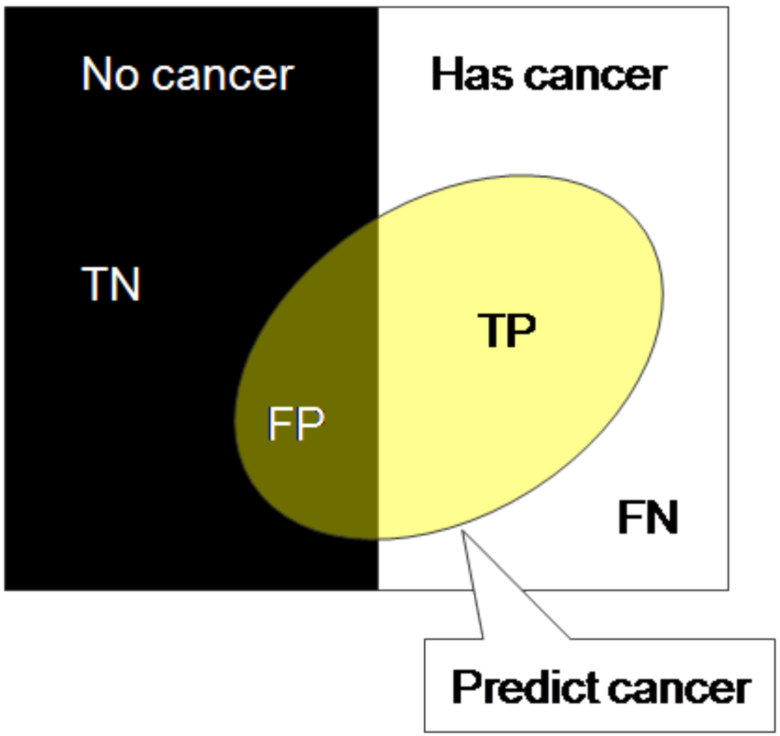
The F1 measure combines both into a useful single metric

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

Actual\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity*
cancer = no	**140**	**9560**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)

ROC Curve (1)

Binary Classification Problem



	Has cancer	No cancer
Predict cancer	TP	FP
Predict No cancer	FN	TN

Fail to detect

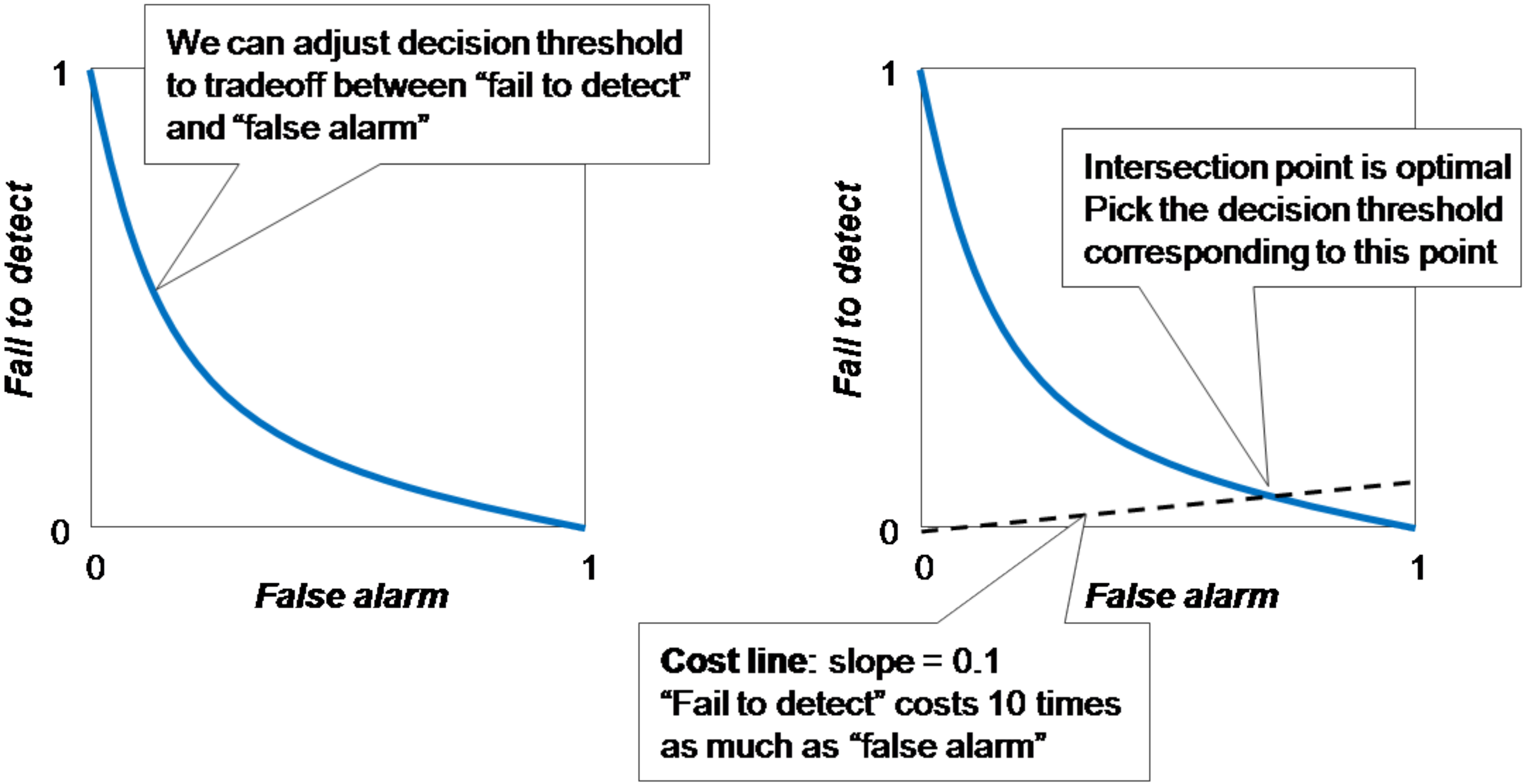
P (Miss | Cancer) = FN / (TP + FN)

False alarm

P (Alarm | NoCancer) = FP / (FP + TN)

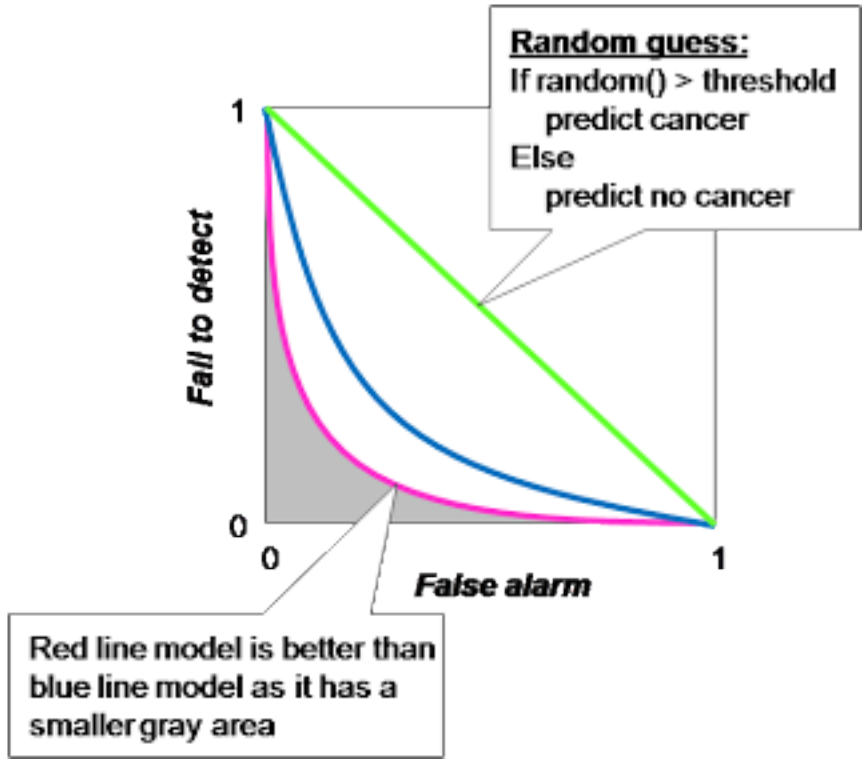
ROC = Receiver operating characteristic

ROC Curve (2)



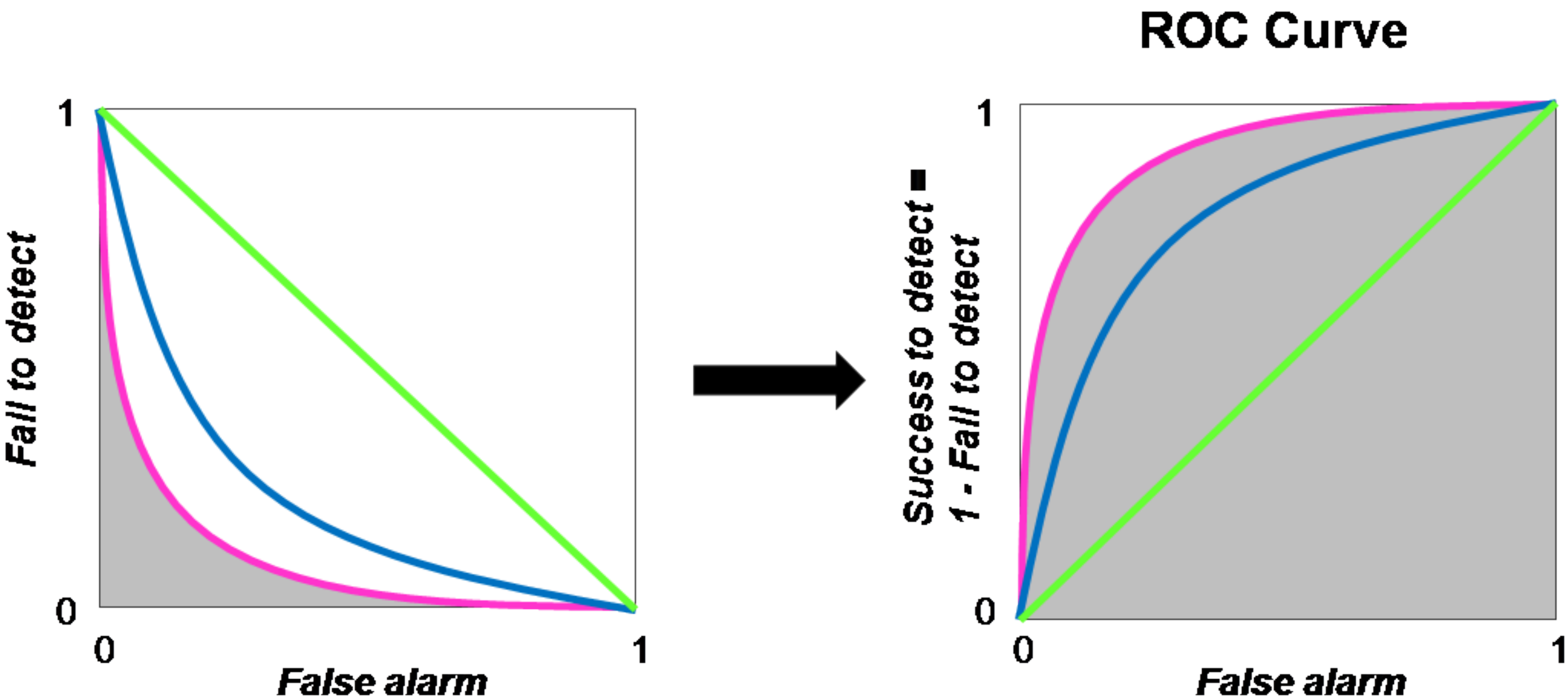
There is always a tradeoff between the false negative rate and the false positive rate

ROC Curve (3)



"Random guess" is worst prediction model and is used as a baseline. The decision threshold of a random guess is a number between 0 to 1 in order to determine between positive and negative prediction.

ROC Curve (4)



ROC Curve transforms the y-axis from "fail to detect" to 1 - "fail to detect", i.e., "success to detect"

Precision at N

- Ranking tasks return a set of results ordered from best to worst
  - E.g., documents about “barack obama”
  - Types for “Barack Obama”
- Learning to rank systems can do this using a variety of algorithms (including SVM)
- Precision at N is the fraction of top N answers that are correct