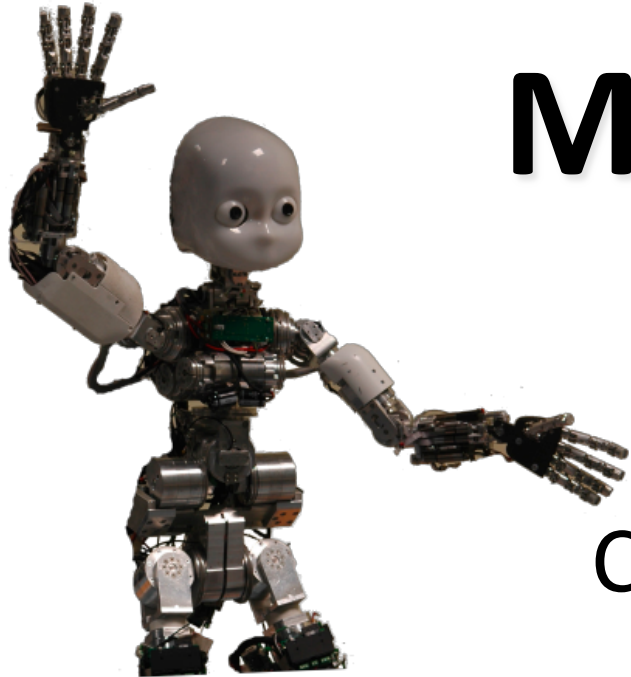


Machine Learning: Methodology



Chapter 18.1-18.3

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.

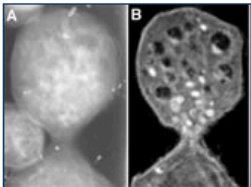
Supported By: [National Science Foundation logo] In Collaboration With: [Rexa.info logo]

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

Center for Machine Learning and Intelligent Systems

[About](#) [Citation Policy](#) [Donate a Data Set](#)
[Contact](#)

 Search

 Repository Web

[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/>

Data Set Characteristics:	Multivariate	<u>Zoo</u> 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

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}

Zoo data

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: *development* 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 as 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)
```

```
0.80000000000000000004
```

```
>>> train_and_test(dtl(), zoo, 90, 101)
```

```
0.81818181818181823
```

```
>>> train_and_test(dtl(), zoo, 80, 90)
```

```
0.90000000000000000002
```

K-fold Cross Validation

- Problem: getting *ground truth* data expensive
- Problem: Need different test data each time we test
- Problem: experiments needed to find right *feature space* & parameters for ML algorithm
- Goal: minimize 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

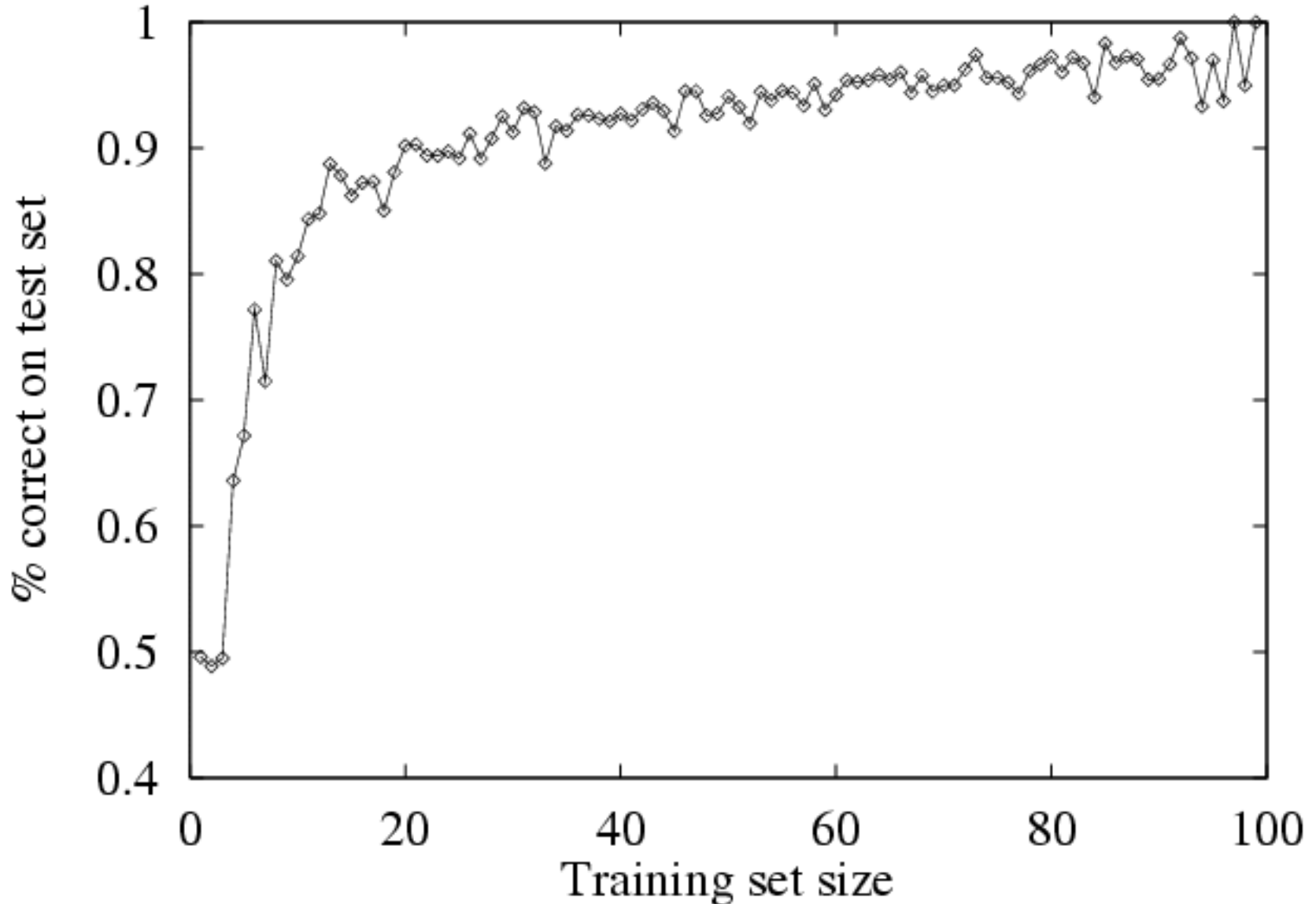
```
>>> cross_validation(dtl(), zoo, 10, 20)
0.955000000000000007
```

`leave1out(learner, data)` does $\text{len}(\text{data})$ trials, each using one element for test, rest for train

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

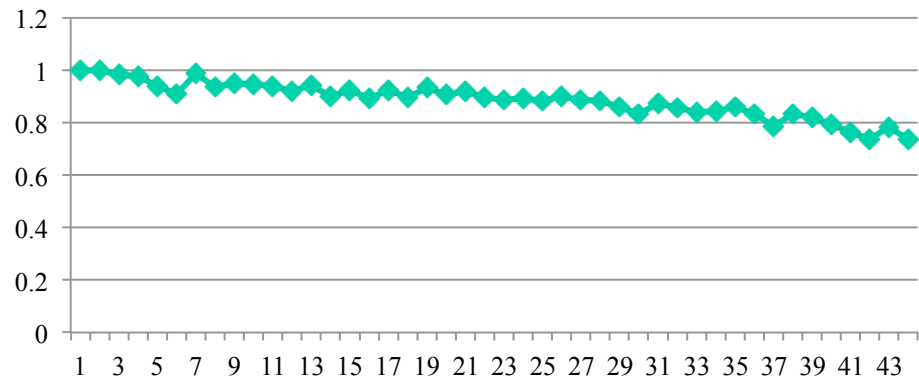
Learning curve

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



Zoo

```
>>> learningcurve(DecisionTreeLearner(), zoo)
[(2, 1.0), (4, 1.0), (6, 0.98333333333333333339), (8,
0.97499999999999999998), (10, 0.94000000000000000006), (12,
0.90833333333333333321), (14, 0.98571428571428577), (16,
0.9375), (18, 0.94999999999999999996), (20,
0.94499999999999999995), ... (86, 0.78255813953488373), (88,
0.7363636363636363644), (90, 0.7077777777777777795)]
```



UCI



Machine Learning Repository

Center for Machine Learning and Intelligent Systems

[About](#) [Citation Policy](#) [Donate a Data Set](#) [Contact](#)

 Repository Web 

[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
```

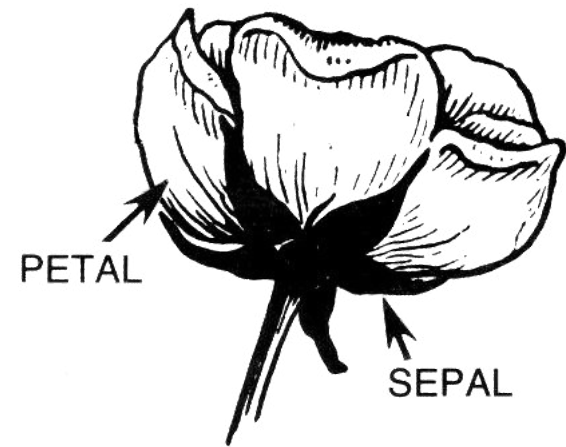
```
5.1,3.5,1.4,0.2,setosa
```

```
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	
		c	$\sim c$
actual	c	True positive	False negative
	$\sim c$	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		
		Cat	Dog	rabbit
actual	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

$$\text{Accuracy} = (TP + TN)/All$$

- **Error rate**: $1 - accuracy$, or

$$\text{Error rate} = (FP + FN)/All$$

- **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**

On Sensitivity and Specificity

- sensitivity measures avoiding of false negatives
- specificity measures avoiding false positives
- TSA security scenario:
 - metal scanners set for low specificity (e.g.trigger on keys) to reduce risk of missing dangerous objects
 - result is high sensitivity overall

Precision and Recall

Information retrieval uses [precision and recall](#) to characterize retrieval effectiveness

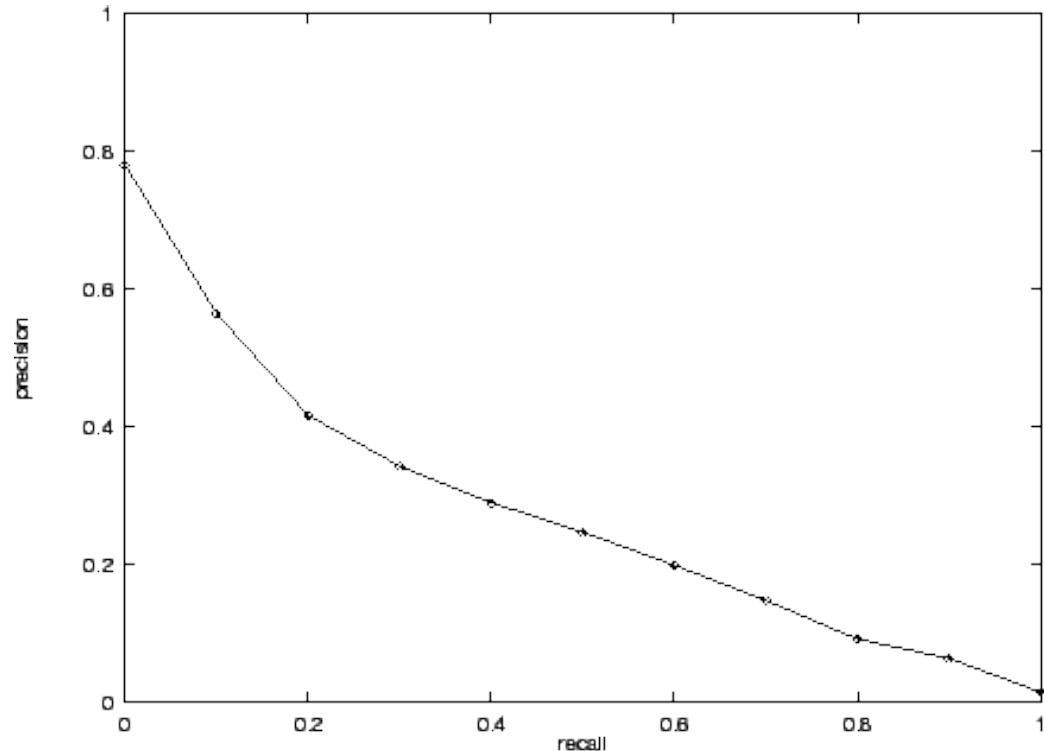
- **Precision:** exactness – what % of tuples that the classifier labeled as positive are actually positive
- **Recall:** completeness – what % of positive tuples did the classifier label as positive?

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

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

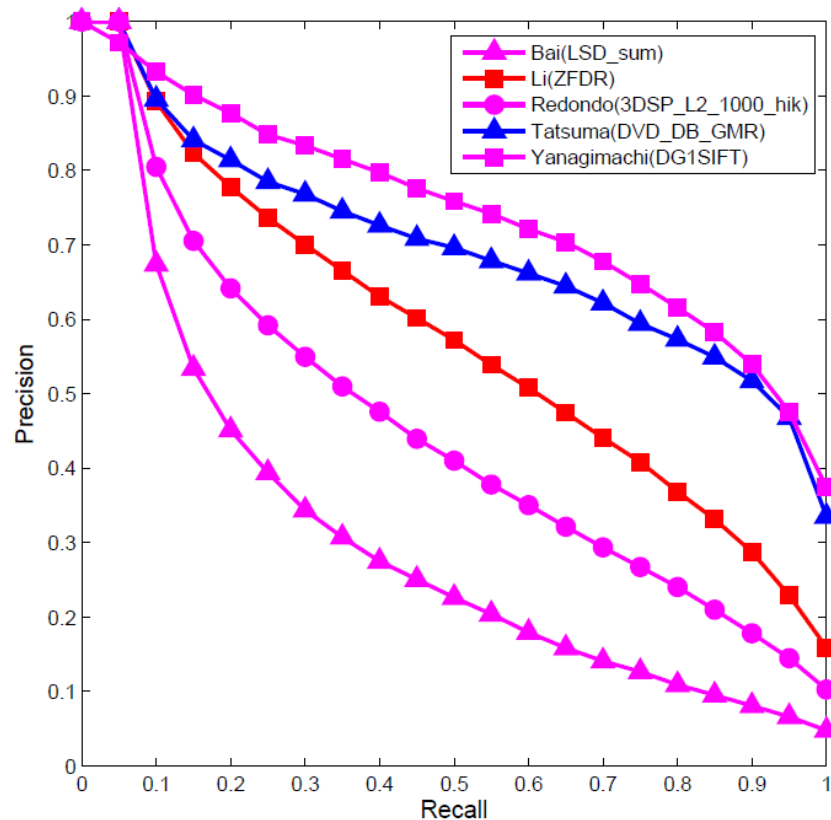
Precision and Recall

- 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

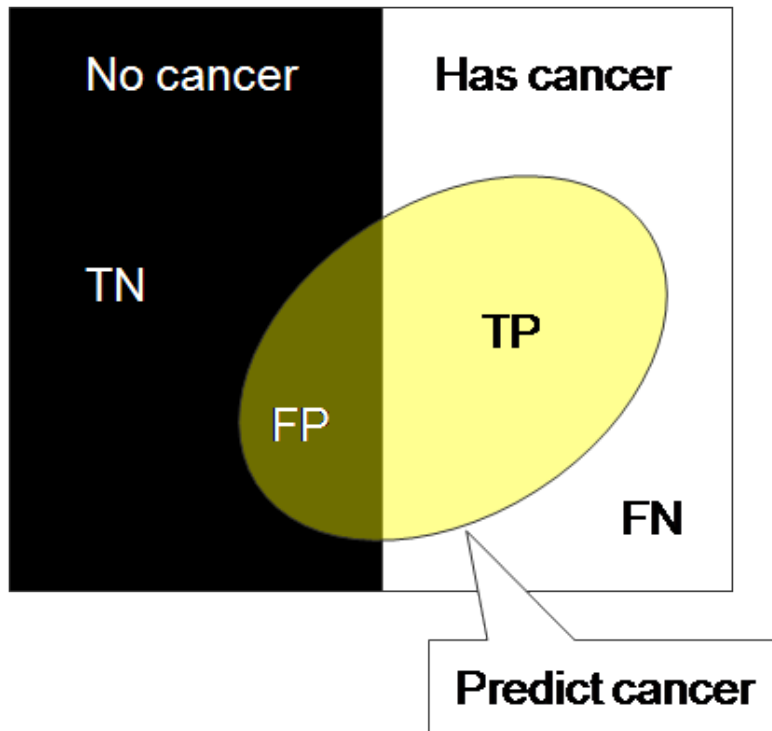
The [F1 measure](#) combines both into a useful single metric

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

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

ROC Curve (1)

Binary Classification Problem



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

Fail to detect

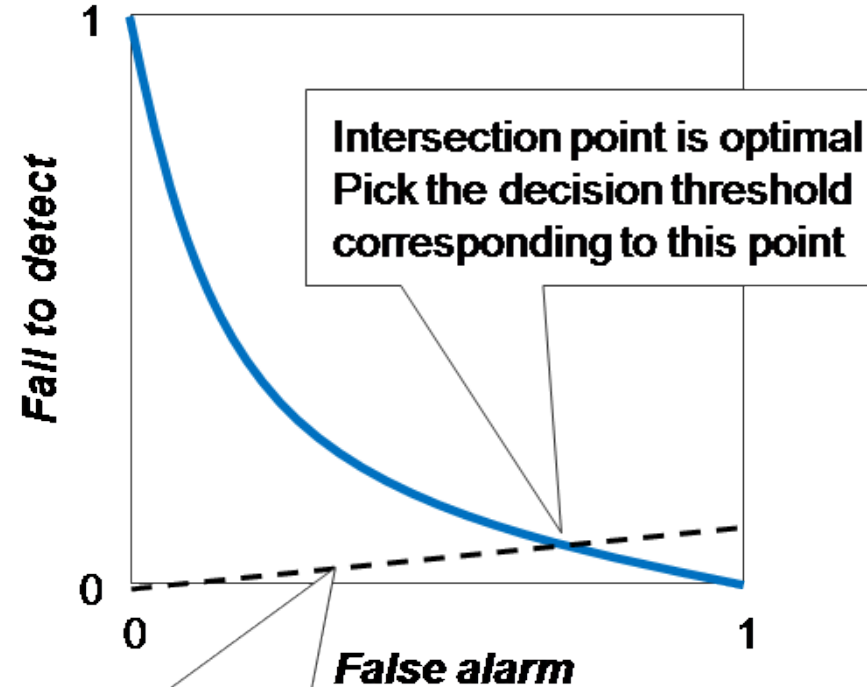
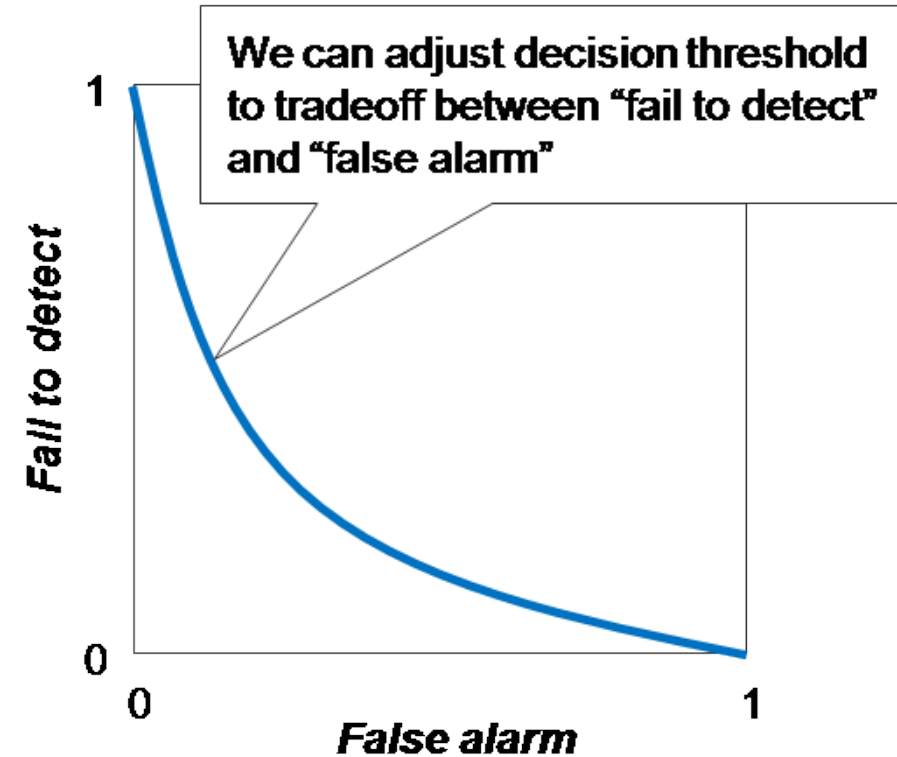
$$P(\text{Miss} | \text{Cancer}) = \text{FN} / (\text{TP} + \text{FN})$$

False alarm

$$P(\text{Alarm} | \text{NoCancer}) = \text{FP} / (\text{FP} + \text{TN})$$

ROC = [Receiver operating characteristic](#)

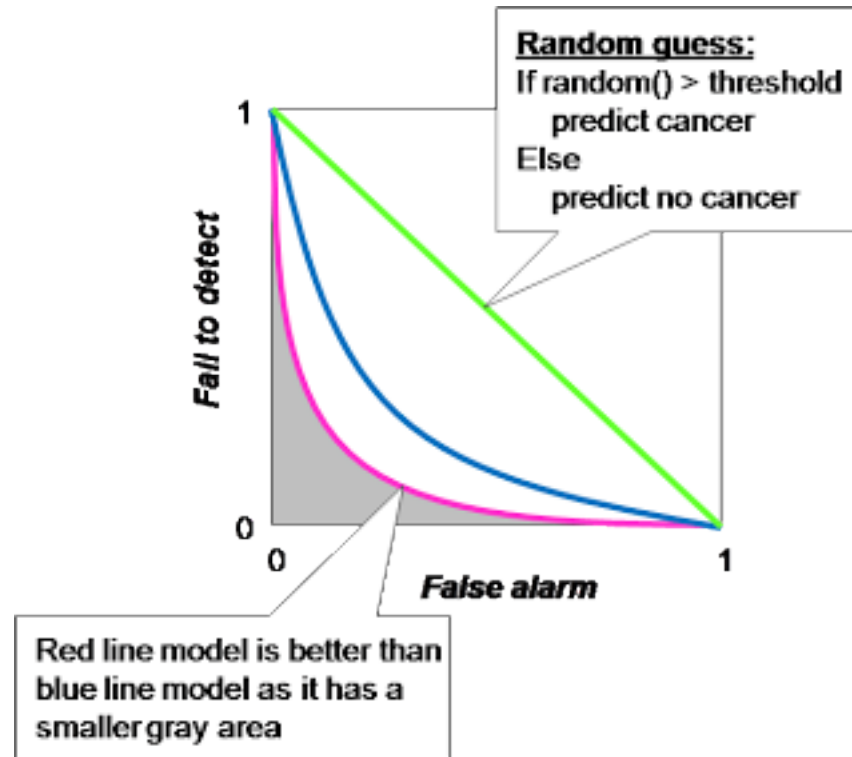
ROC Curve (2)



Cost line: slope = 0.1
"Fail to detect" costs 10 times as much as "false alarm"

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

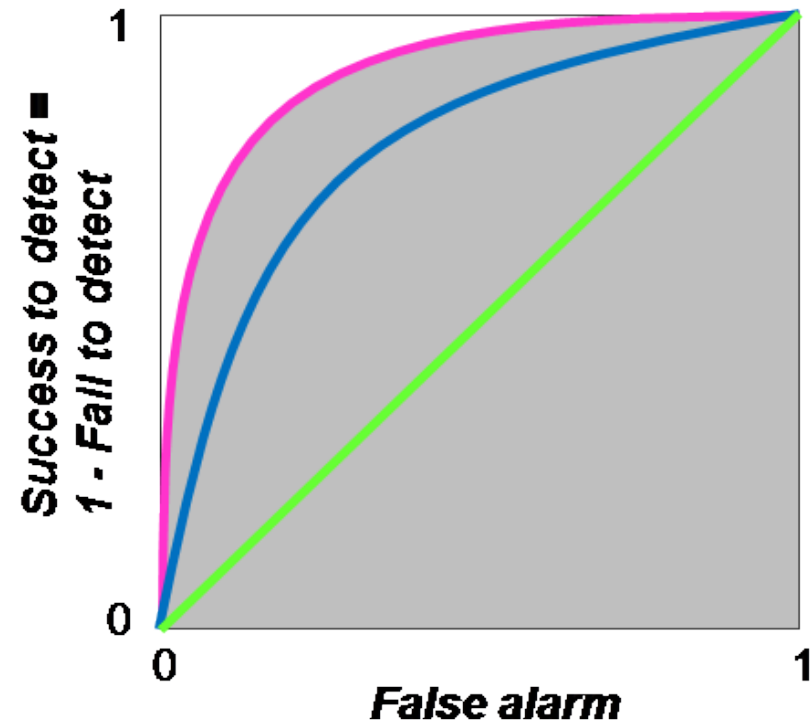
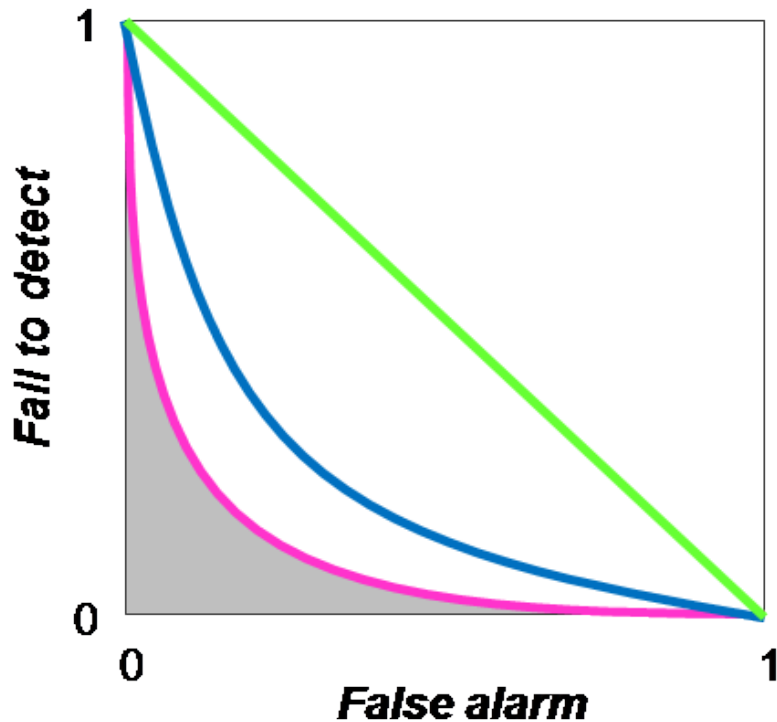
ROC Curve (3)



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

ROC Curve (4)

ROC Curve



ROC Curve transforms the y-axis from "fail to detect" to $1 - \text{"fail to detect"}$, 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