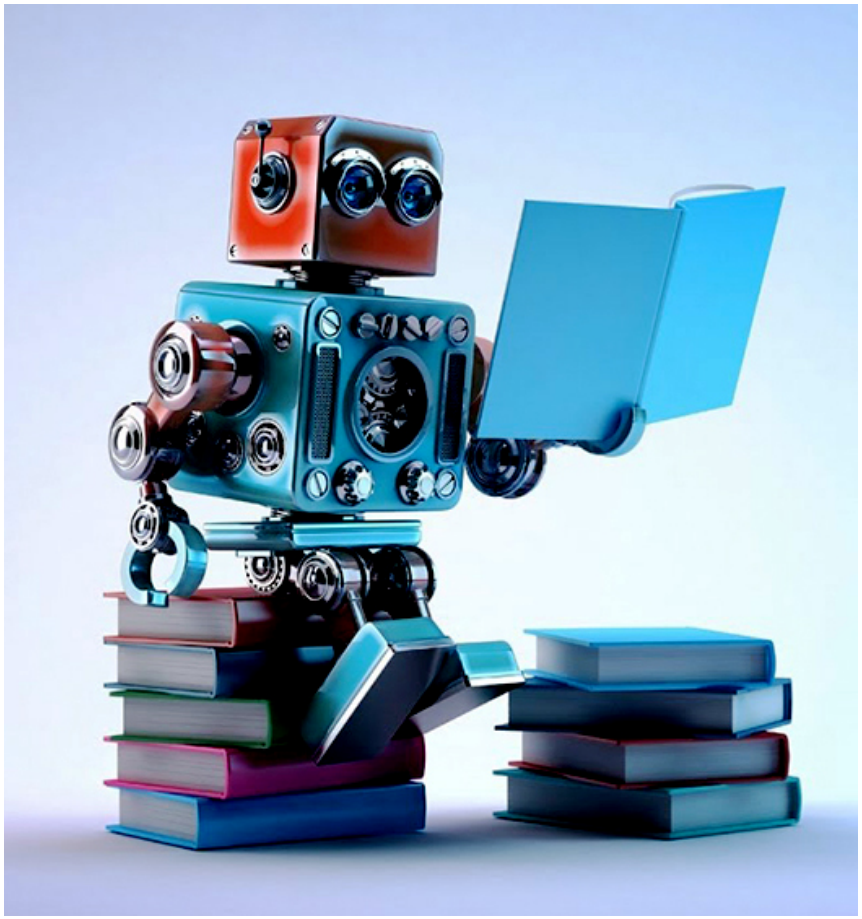


Machine Learning: Methodology

Chapter 18.1-18.3

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

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.

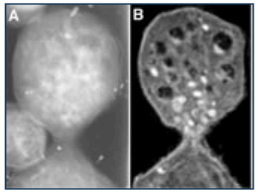
Supported By: National Science Foundation In Collaboration With: Rexa.info

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

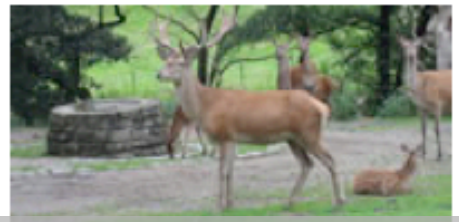
UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

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

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 (aka ground truth data)
2. Randomly divide collection into two disjoint sets: ***training*** and ***test*** *(e.g., via a 90-10% split)*
3. Apply learning algorithm to **training** set giving hypothesis H
4. Measure performance of H on the held-out **test** set

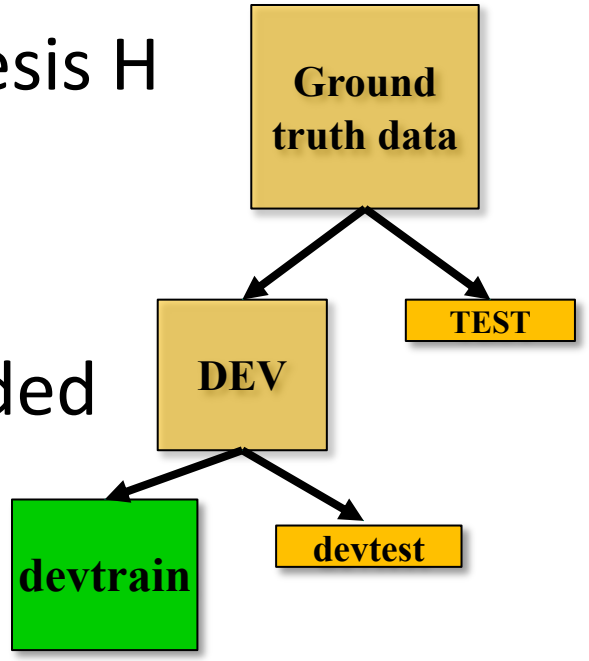
Evaluation methodology (2)

- 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 set of examples with correct classifications
2. Randomly divide it into two disjoint sets: ***developmen*t** & ***test;*** further divide development into ***devtrain*** & ***devtest***
3. Apply ML to *devtrain,* giving hypothesis H
4. Measure performance of H w.r.t. *devtest* data
5. Modify approach, repeat 3-4 as needed
6. Final test on *test* data

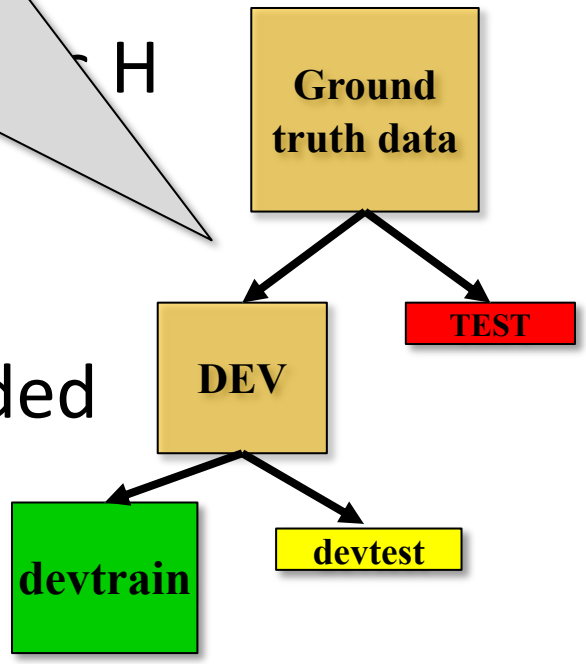


Evaluation methodology (4)

Co

1. classifications
2. sets:
   development
3. H
4. *devtest* data
5. Modify approach, repeat 3-4 as needed
6. Final test on *test* data

- Only **devtest** data used for evaluation during system **development**
- When all development has ended, **test** data used for **final evaluation**
- Ensures final system not influenced by test data
- If more development needed, get new dataset!



Zoo evaluation

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

```
>>> dtl = DecisionTreeLearner
>>> train_and_test(dtl(), zoo, 0, 10)
1.0
>>> train_and_test(dtl(), zoo, 90, 100)
0.8000000000000004
>>> train_and_test(dtl(), zoo, 90, 101)
0.8181818181818183
>>> train_and_test(dtl(), zoo, 80, 90)
0.9000000000000002
```

Zoo evaluation

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

- We hold out 10 data items for test; train on the other 91; show the accuracy on the test data
- Doing this four times for different test subsets shows accuracy from 80% to 100%
- What's the true accuracy of our approach?

K-fold Cross Validation

- **Problem:** getting *ground truth* data expensive
- **Problem: n**eed different test data for each test
- **Problem:** experiments needed to find right *feature space* & parameters for ML algorithms
- **Goal:** minimize training+test data needed
- **Idea:** split training data into K subsets; use K-1 for *training* and one for *development testing*
- Repeat K times and average performance
- Common K values are 5 and 10

Zoo evaluation

- AIMA code has a cross_validation function that runs K-fold cross validation
- cross_validation(learner, data, K, N) does N iterations, each time randomly selecting 1/K data points for test, leaving rest for train

```
>>> cross_validation(dtl(), zoo, 10, 20)
0.9550000000000007
```

- This is a very common approach to evaluating the accuracy of a model during development
- Best practice is still to hold out a final test data set

Leave one out

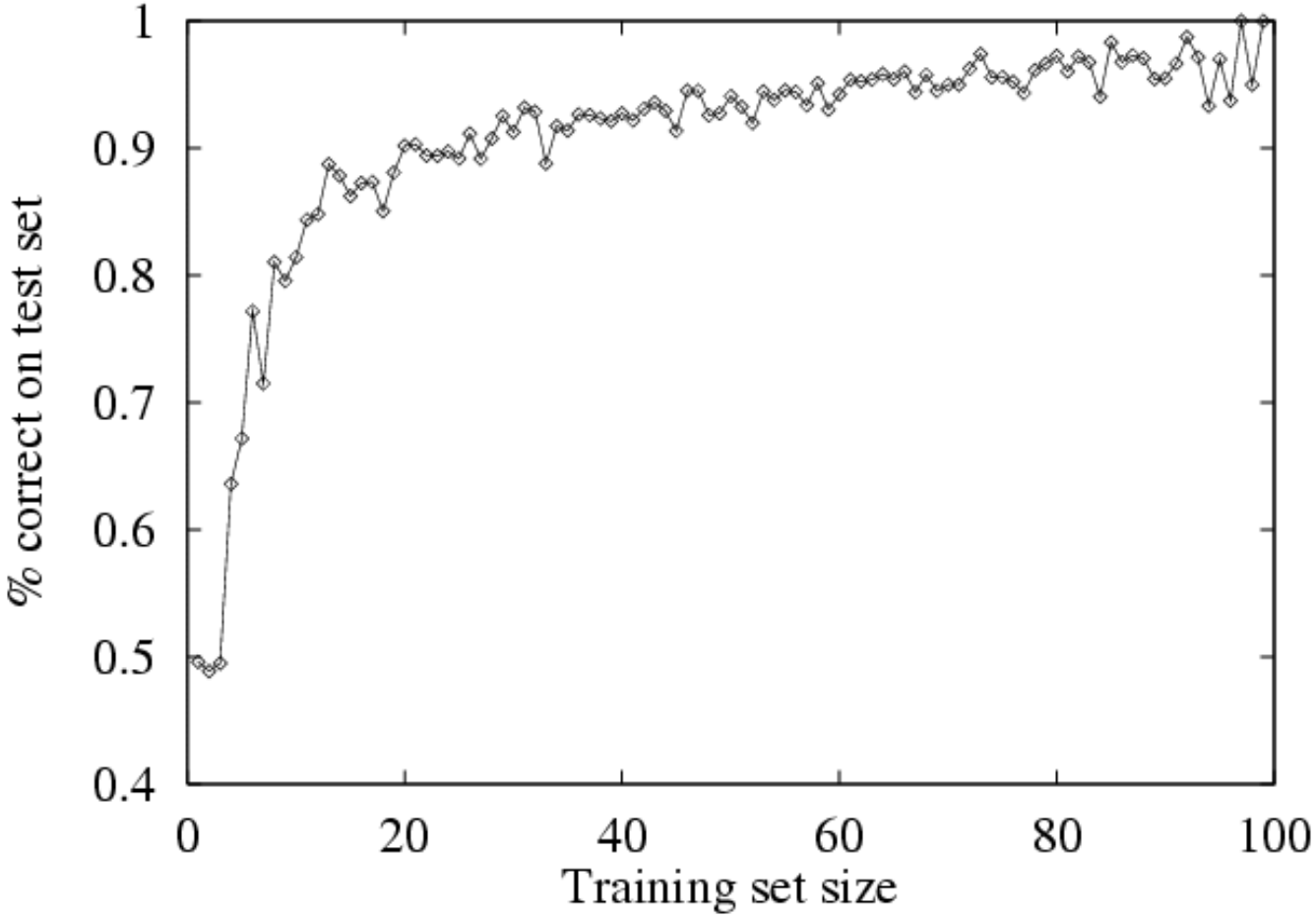
- AIMA code also has a leave1out function that runs a different set of experiments to estimate accuracy of the model
- leave1out(learner, data) does len(data) trials, each using one element for test, rest for train

```
>>> leave1out(dtl(), zoo)
0.9702970297029702
```

- K-fold cross validation can be too pessimistic, since it only trains with 80% or 90% of the data
- The leave one out evaluation is an alternative

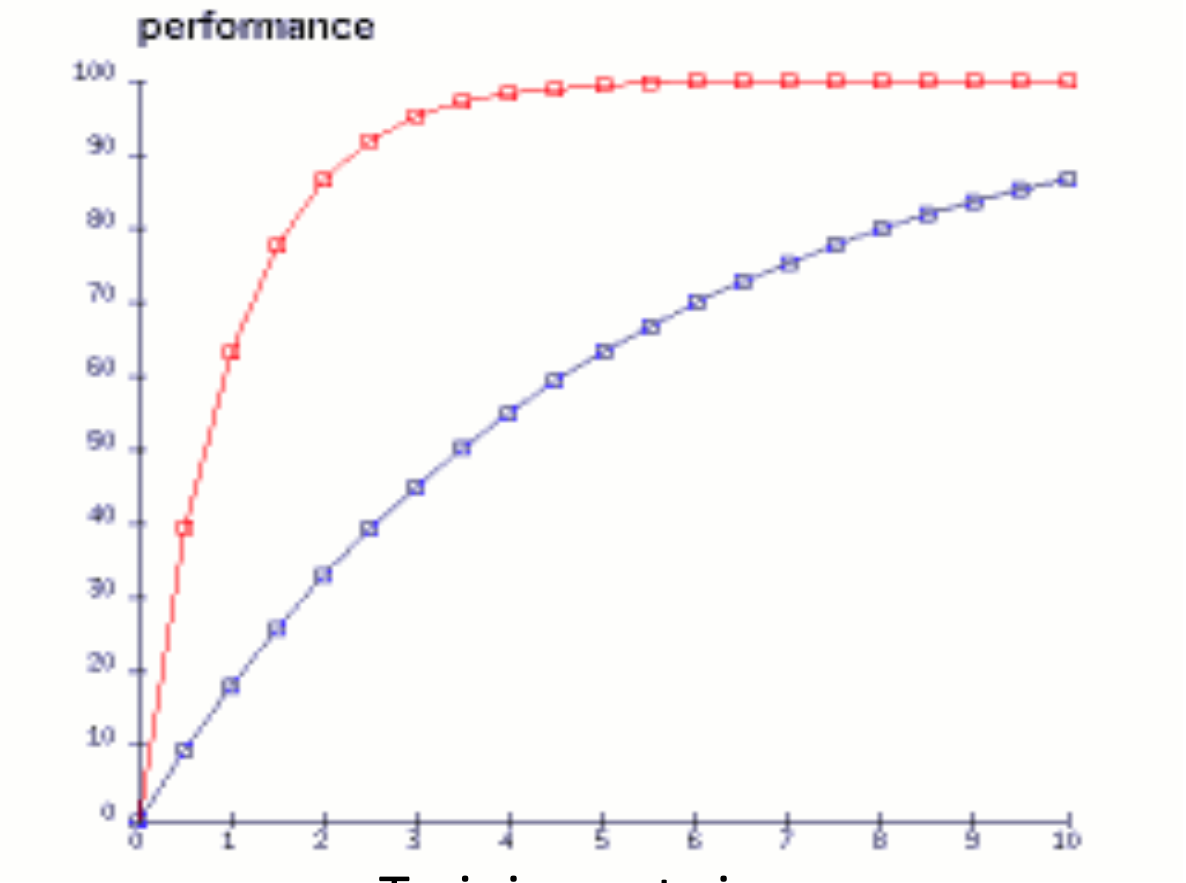
Learning curve (1)

A learning curve shows accuracy on test set as a function of training set size or (for neural networks) running time



Learning curve

- When evaluating ML algorithms, steeper learning curves are better
- They represents faster learning with less data



Here the system with the red curve is better since it requires less data to achieve given accuracy

UCI Machine Learning Repos

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

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

Iris Data Set

Download: Data Folder, Data Set Description

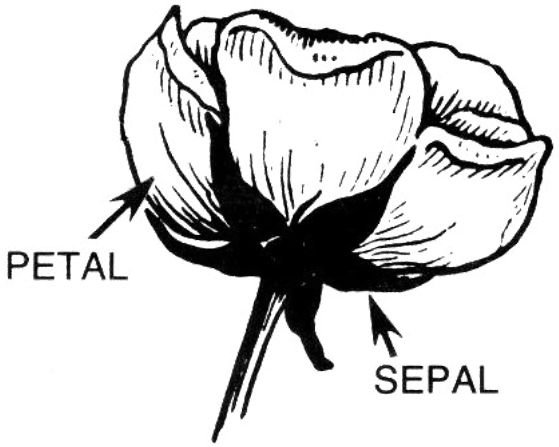
Abstract: Famous database; from Fisher, 1936

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	C	~C
C	True positive	False positive
~C	False negative	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

actual

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

- This result suggests we find it easy to confuse dogs and cats

Accuracy, Error Rate, Sensitivity, Specificity

P/A	C	¬C	
C	**TP**	**FP**	**P'**
¬C	**FN**	**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 in negative class &* rest in positive class
- **Sensitivity**: True Positive recognition rate
  - **Sensitivity = TP/P**
- **Specificity**: True Negative recognition rate
  - **Specificity = TN/N**

On Sensitivity and Specificity

- High sensitivity: few false negatives
- High specificity: few false positives
- TSA security scenario:

  metal scanners set for high sensitivity and low specificity (e.g., trigger on keys) to reduce risk of missing dangerous objects
- COVID-19 testing scenario?

COVID-19 Sensitivity & Specificity

- COVID-19: test sensitivity and specificity both 0.99 (i.e., 99% accuracy)
- Assume 1% of population infected(pos)
- Test 10,000 people (100 pos, 9900 neg)
  - 99 + 99 will show positive (half right, half wrong)
- Dr. Birx: “I want to be very clear to the American people, none of our tests are 100% sensitive and specific. What do I mean by that? None of our tests that we use in medicine and diagnose 100% of the people who are positive, and correctly diagnose 100% of the people who are negative"

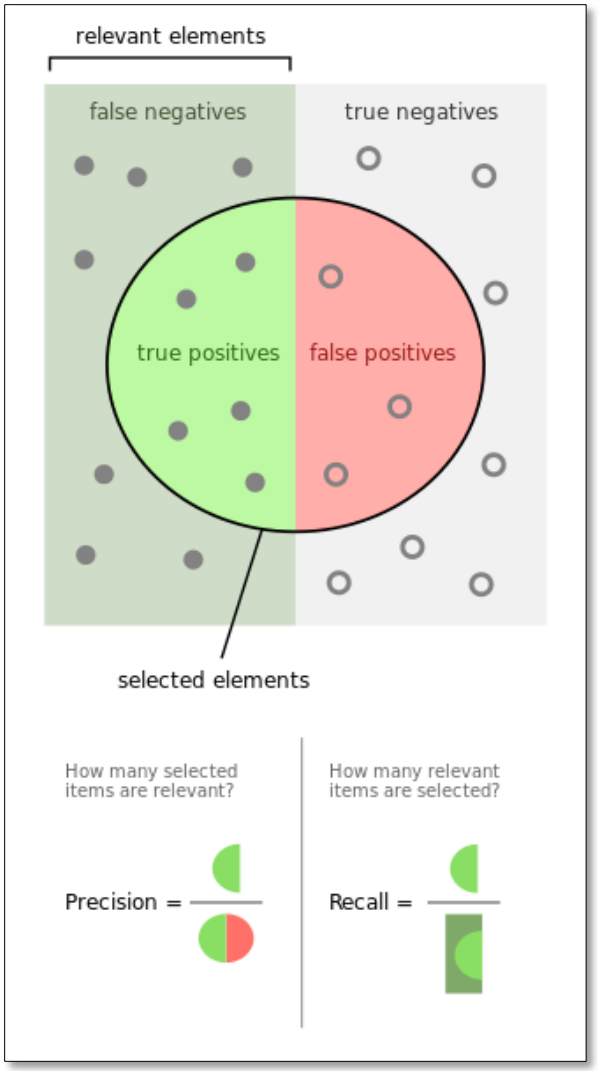
Precision and Recall

Information retrieval uses similar measures, precision & recall, to characterize retrieval effectiveness

- **Precision**: % of tuples classifier labels as positive that are actually positive
- **Recall:** % of positive tuples classifier labels as positive

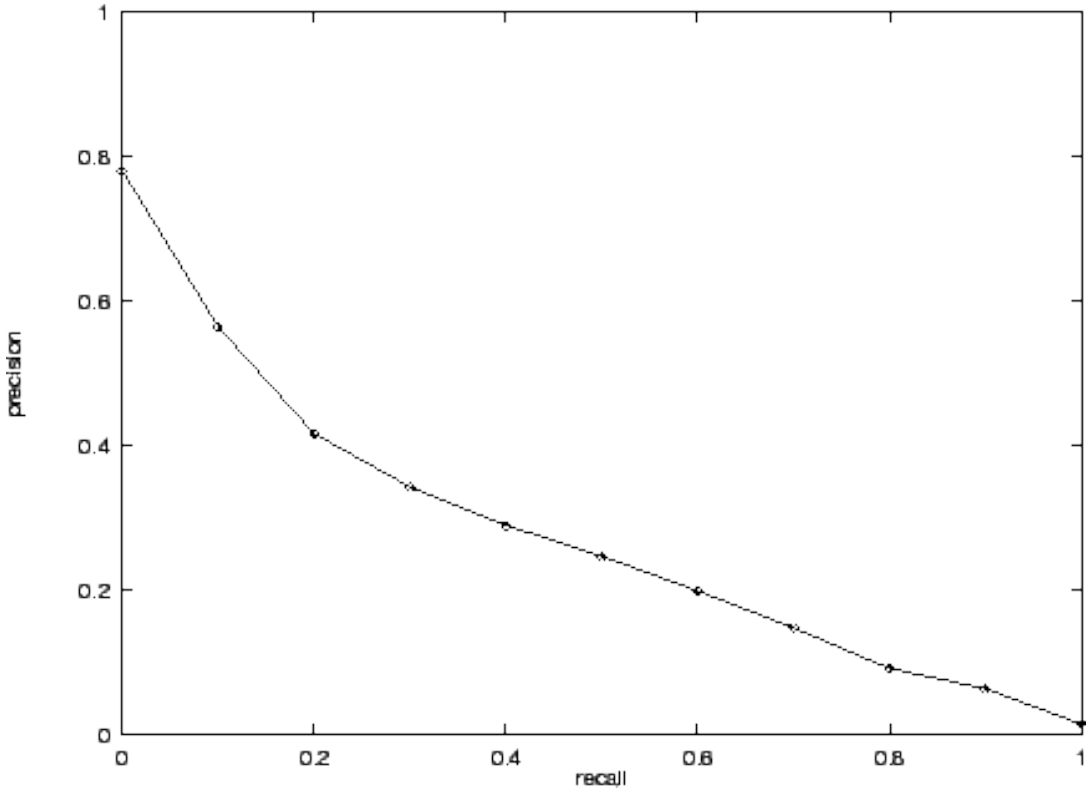
$$precision = \frac{TP}{TP + FP}$$

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



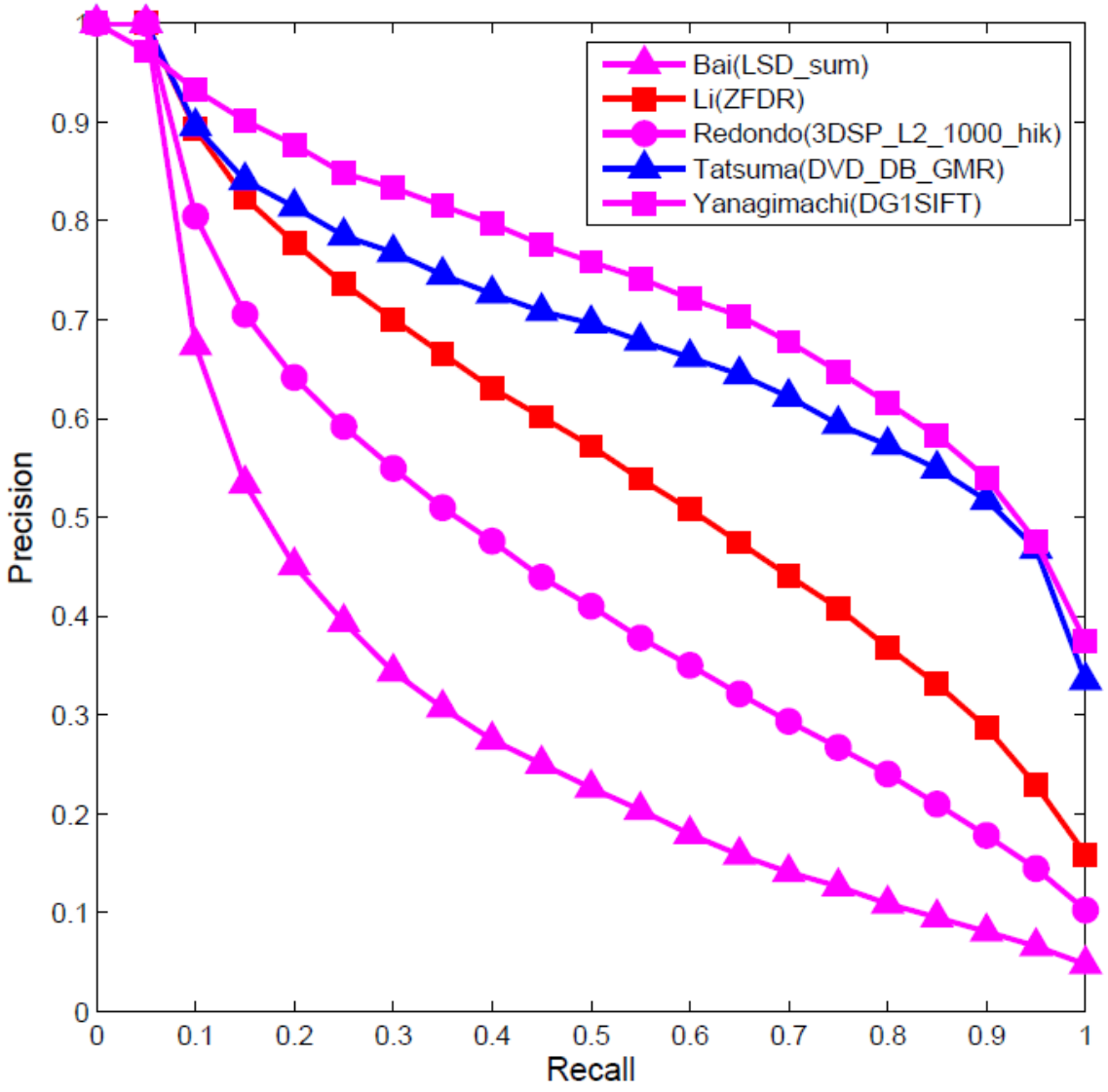
Precision and Recall

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



Precision and Recall

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



F1 measure

- We often want just one measure to comare two systems
- F1 measure combines both into a useful single metric
- It's the harmonic mean of precision & recall

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

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 K is the fraction of top K answers that are correct

Summary

- Evaluating the results of a ML system is very important!
- Part of the development process to decide
  - What parameters maximize performance?
  - Is one system better?
  - Do we need more data?
  - etc.
- Many ML algorithms have specialized evaluation techniques
- There a lot more to the topic