Machine Learning: Methodology
Chapter 19
ML is an experimental science

• Most ML work has an engineering or experimental flavor
  – it’s being used as a tool to solve a problem
• Methodology is important
• As are approaches for evaluating results
• It’s common to try multiple ML methods, features, and parameters for a problem to find what works best
• See Google’s Rules of Machine Learning for more information
Approaches

• Different classes of ML algorithms have different kinds of evaluation techniques

• Supervised ML
  – We can use our data with the right answers

• Unsupervised ML
  – Some general metrics exist (e.g., for clusters)
  – May need human assessments

• Reinforcement learning
  – Problem usually determines good/bad outcomes (e.g., points won in a game)
<table>
<thead>
<tr>
<th>animal name</th>
<th>hair</th>
<th>feathers</th>
<th>eggs</th>
<th>milk</th>
<th>airborne</th>
<th>aquatic</th>
<th>predator</th>
<th>toothed</th>
<th>backbone</th>
<th>breathes</th>
<th>venomous</th>
<th>fins</th>
<th>legs</th>
<th>tail</th>
<th>domestic</th>
<th>catsize</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>aardvark</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0,1,1,1</td>
<td>1,1</td>
<td>0,0,4,0,0,1</td>
<td>mammal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mammal</td>
</tr>
<tr>
<td>antelope</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0,0,1,1</td>
<td>1,1</td>
<td>0,0,4,1,0,1</td>
<td>mammal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mammal</td>
</tr>
<tr>
<td>bass</td>
<td>0,0</td>
<td>1,0,0,1</td>
<td>1,1,1,0,0,1,0,1,0,0</td>
<td>fish</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>fish</td>
</tr>
<tr>
<td>bear</td>
<td>1,0</td>
<td>0,1,0</td>
<td>1,1,1,0,0,0,1,0,1,0,0,1</td>
<td>mammal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mammal</td>
</tr>
<tr>
<td>boar</td>
<td>1,0</td>
<td>0,1,0</td>
<td>1,1,1,0,0,0,4,1,0,1</td>
<td>mammal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mammal</td>
</tr>
<tr>
<td>buffalo</td>
<td>1,0</td>
<td>0,1,0</td>
<td>0,0,1,1,1,0,0,4,1,0,1</td>
<td>mammal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mammal</td>
</tr>
<tr>
<td>calf</td>
<td>1,0</td>
<td>0,1,0</td>
<td>0,0,1,1,1,0,0,4,1,1,1</td>
<td>mammal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mammal</td>
</tr>
<tr>
<td>carp</td>
<td>0,0</td>
<td>1,0,0,1</td>
<td>0,1,1,0,0,1,0,1,1,0</td>
<td>fish</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>fish</td>
</tr>
<tr>
<td>catfish</td>
<td>0,0</td>
<td>1,0,0,1</td>
<td>1,1,1,0,0,0,1,0,1,0,0,1</td>
<td>fish</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>fish</td>
</tr>
<tr>
<td>cavy</td>
<td>1,0</td>
<td>0,1,0</td>
<td>0,0,1,1,1,0,0,4,0,1,0</td>
<td>mammal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mammal</td>
</tr>
<tr>
<td>cheetah</td>
<td>1,0</td>
<td>0,1,0</td>
<td>0,1,1,1,0,0,4,1,0,1</td>
<td>mammal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mammal</td>
</tr>
<tr>
<td>chicken</td>
<td>0,1</td>
<td>1,0,0,1</td>
<td>1,0,0,0,1,1,0,0,2,1,1,0</td>
<td>bird</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bird</td>
</tr>
<tr>
<td>chub</td>
<td>0,0</td>
<td>1,0,0,1</td>
<td>1,1,1,0,0,1,0,1,0,0,1</td>
<td>fish</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>fish</td>
</tr>
<tr>
<td>clam</td>
<td>0,0</td>
<td>1,0,0,1</td>
<td>1,0,0,0,0,0,0,0,0,0,0</td>
<td>shellfish</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>shellfish</td>
</tr>
<tr>
<td>crab</td>
<td>0,0</td>
<td>1,0,0,1</td>
<td>1,1,0,0,0,0,0,4,0,0,0</td>
<td>shellfish</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>shellfish</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Zoo example

```python
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
Common variation on methodology:

1. Collect set of examples with correct classifications
2. Randomly divide it into two disjoint sets: *development* & *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)

1. Collect set of examples with correct classifications
2. Randomly divide it into two disjoint sets: development & 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

- 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

• 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?
Zoo evaluation

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

```python
>>> 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 expensive
- **Problem:** need 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
N-fold Cross Validation

• 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

• Very common approach to evaluating model accuracy during development

• Best practice: hold out a final test data set
Leave one out validation

• 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)

    0.97029702970297027

• 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.
- Represent faster learning with less data.

System with the red curve is better since it requires less data to achieve a given accuracy.
Iris Data Set

*Download:* Data Folder, Data Set Description

*Abstract:* Famous database; from Fisher, 1936

---

**Data Set Characteristics:**
- Multivariate
- Number of Instances: 150

**Attribute Characteristics:**
- Real
- Number of Attributes: 4

**Associated Tasks:**
- Classification
- Missing Values?
- No

**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

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

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

<table>
<thead>
<tr>
<th></th>
<th>iris</th>
<th>zoo</th>
</tr>
</thead>
<tbody>
<tr>
<td>DecisionTree</td>
<td>0.86</td>
<td>0.94</td>
</tr>
<tr>
<td>NaiveBayes</td>
<td>0.92</td>
<td>0.92</td>
</tr>
<tr>
<td>NearestNeighbor</td>
<td>0.85</td>
<td><strong>0.96</strong></td>
</tr>
</tbody>
</table>
```
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

<table>
<thead>
<tr>
<th>predicted</th>
<th>actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>True positive</td>
</tr>
<tr>
<td>~C</td>
<td>False negative</td>
</tr>
</tbody>
</table>
• For multi-way classifiers, a confusion matrix is even more useful
• It lets you focus in on where the errors are

<table>
<thead>
<tr>
<th></th>
<th>Cat</th>
<th>Dog</th>
<th>rabbit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>5</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Dog</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Rabbit</td>
<td>0</td>
<td>2</td>
<td>11</td>
</tr>
</tbody>
</table>

• This result suggests we find it easy to confuse dogs and cats
Accuracy, Error Rate, Sensitivity, Specificity

<table>
<thead>
<tr>
<th>P/A</th>
<th>C</th>
<th>¬C</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>¬C</td>
<td>FN</td>
<td>TN</td>
</tr>
<tr>
<td>All</td>
<td>P</td>
<td>N</td>
</tr>
</tbody>
</table>

- **Classifier Accuracy**, or recognition rate: percentage of test set tuples are correctly classified
  \[ \text{Accuracy} = \frac{(TP + TN)}{\text{All}} \]

- **Error rate**: 1 – accuracy, or
  \[ \text{Error rate} = \frac{(FP + FN)}{\text{All}} \]

**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
  \[ \text{Sensitivity} = \frac{TP}{P} \]

- **Specificity**: True Negative recognition rate
  \[ \text{Specificity} = \frac{TN}{N} \]
On Sensitivity and Specificity

• **High sensitivity**: few false negatives
  
  sensitivity=$1 \Rightarrow TP=P \Rightarrow$ you correctly identify all positives, but may include many negatives

• **High specificity**: few false positives
  
  specificity=$1 \Rightarrow TN=N \Rightarrow$ you correctly identify all negatives but may include many positives

• TSA security scenario:

  Scanners set for high sensitivity & low specificity (e.g., trigger on keys) reducing risk of missing dangerous objects

• Web search scenario:

  Set specificity high so first page has nearly all relevant documents
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"
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.

\[
\text{precision} = \frac{TP}{TP + FP}
\]

\[
\text{recall} = \frac{TP}{TP + FN}
\]

![Diagram showing precision and recall concepts](image-url)
Precision and Recall

• In general, increasing one causes the other to decrease
• Studying precision-recall curve is informative
If one system’s curve is always above the other, it’s better.
F1 measure

• We often want just one measure to compare two systems to decide which is best overall

• **F1 measure** combines both into a useful single metric

• It’s the **harmonic mean** of precision & recall

\[
H = \frac{2x_1 x_2}{x_1 + x_2} \quad F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]
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 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 is a lot more to the topic