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
- Common to try multiple ML methods, features, and parameters for a problem to find what works best
- Google’s [Rules of Machine Learning](#) has more information
Many moving parts

Solving a problem with machine learning involves many decisions

• Selecting training data and deciding how much is needed
• Preprocessing the data, creating new features from it
• Selecting a machine learning algorithm
• Choosing its parameters
• Deciding on a metric to optimize
• Running evaluation experiments
Approaches

• Different classes of ML algorithms have different kinds of evaluation techniques
  – Some are common to most, however

• **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 determines good/bad outcomes (e.g., points won in a game)
### Zoo data

<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>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>mammal</td>
<td></td>
</tr>
<tr>
<td>antelope</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>mammal</td>
<td></td>
</tr>
<tr>
<td>bass</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>fish</td>
<td></td>
</tr>
<tr>
<td>bear</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>mammal</td>
<td></td>
</tr>
<tr>
<td>boar</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>mammal</td>
<td></td>
</tr>
<tr>
<td>buffalo</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>mammal</td>
<td></td>
</tr>
<tr>
<td>calf</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>mammal</td>
<td></td>
</tr>
<tr>
<td>carp</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>fish</td>
<td></td>
</tr>
<tr>
<td>catfish</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>fish</td>
<td></td>
</tr>
<tr>
<td>cavy</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>mammal</td>
<td></td>
</tr>
<tr>
<td>cheetah</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>mammal</td>
<td></td>
</tr>
<tr>
<td>chicken</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>bird</td>
<td></td>
</tr>
<tr>
<td>chub</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>fish</td>
<td></td>
</tr>
<tr>
<td>clam</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>shellfish</td>
<td></td>
</tr>
<tr>
<td>crab</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</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

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 & test (e.g., via a 90-10% split)
3. Train a model using your algorithm on the training set giving hypothesis H
4. Measure performance of the model (and H) on the held-out test set
Accuracy: a simple metric

• What measure of performance can we use?
• It depends on the kind of task, e.g.,
  – Classification (e.g., which species of iris)
  – Information retrieval (find relevant documents)
• One of the simplest is accuracy
  – Fraction of the answers that are correct
• It doesn't weigh different kinds of errors differently (e.g., false positive vs false negatives)
Evaluation methodology (2)

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

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 trained system not influenced by test data
- If more development needed, get new dataset!
Zoo evaluation

\( \text{train\_and\_test(learner, data, start, end)} \) uses data[start:end] for test and rest for train

- 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)
0.8000000000000004
>>> train_and_test(dtl(), zoo, 90, 101)
0.81818181818181823
>>> train_and_test(dtl(), zoo, 80, 90)
0.9000000000000002
```

We might use the average accuracy of the experiments as the overall metric, in this case 0.9.
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.

```python
>>> 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 experiments to estimate model accuracy
• 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
Fast and slow learners

- Some approaches require less training data to reach a given performance level than others
- We can think of them as faster learners
- Differences can be due to data preprocessing, algorithm choice, and/or parameter settings
- Faster is generally better for many reasons (e.g., may want to apply it to many huge datasets)
- Learning curve give an intuitive way to assess
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
Neural network learning curves

For neural networks, the **x axis** is usually the **number of iterations** of the training algorithm.

System with the red curve is better since it requires fewer iterations and less time to achieve a given accuracy.
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)
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 many problems.
- For binary classifiers it’s simple and related to **type I and type II errors** (i.e., false positives and false negatives).
- We may have different costs for each error.
- So, we must understand their frequencies.

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>~C</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>True positive</td>
<td>False positive</td>
</tr>
<tr>
<td>~C</td>
<td>False negative</td>
<td>True negative</td>
</tr>
</tbody>
</table>
Confusion Matrix (2)

• 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>1</td>
<td>9</td>
</tr>
</tbody>
</table>

• This result suggests a system finds 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>P</td>
<td>N</td>
<td>All</td>
</tr>
</tbody>
</table>

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

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

- **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 = \(\frac{TP}{TP+FN}\)
  - **Specificity**: True Negative recognition rate
    - Specificity = \(\frac{TN}{TN+FP}\)
On Sensitivity and Specificity

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

• **High specificity:** few false positives
  
specificity=1 => TN=N => 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 for *high specificity* 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 where 100 pos, 9900 neg
  – 99 + 99 will test positive (half right, half wrong)
  – 01 + 9801 will test negative (virtually all correct)

• Dr. Birx, April 2020: “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 items classifier labels as positive that are actually positive
- **Recall**: % of positive items classifier labels as positive

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

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

- In general, increasing one causes other to decrease
- Get recall=1 by marking every item as positive
- Get highest precision by marking only one item positive, the one you’re most certain of
- We usually want some balance of precision and recall
- Studying the 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 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
Precision at N

- Ranking tasks return a set of results ordered from best to worst
  - E.g., documents about “barack obama”
  - Best knowledge graph type 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
Grid search

• ML algorithms tend to have many parameters
• How can we effectively find the best setting for all of them?
• A grid search takes a list of possible values for each of a set of parameters
• ...and tests each combination, to get a metric (e.g., accuracy, F1)
• See this scikit learn colab example
Model evaluation in scikit learn

scikit.metrics’s evaluation module supports most of its models in a uniform way

- It has functions that make it easy to
  - Split the data into train and test subsets
  - Do cross validation
  - Get various metrics
  - Do a grid search for a set of parameters and their possible values

- See our colab notebooks for examples
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