What is machine learning?

Machine learning is a large variety of statistical techniques that are used to solve problems that are hard to formalise.

→ If we could formalise these problems then we could use standard software development.

Examples
- What are my favorite books?
  One could develop complete theory about books and persons but it would take centuries to complete.
- Is this blob of pixels a human face?
  Again, we could develop models of plausible human faces and their potential images.
- How does robot react to a control signal?
  This could be modelled exactly, but sometimes we do not know all dimensions and parameters.

Where is machine learning used?

- Astronomy and particle physics → Fiduciality
- Amazon & Netflix → Recommendations
- Robotics & industrial applications → Prediction, approximation, control
- Human-computer interface → Speech & character recognition
- Bioinformatics & biochemistry → Prediction, similarity search, summarisation
- Brick-and-mortar shops → Sales campaigns, shop optimisation

Main inference procedure

1. Gather data
2. Choose method
3. Tune parameters
4. Validate model
5. Start using

- Quality of gathered data determines everything → Garbage in, garbage out
- The amount and the nature of data determine available methods → Less samples, simpler methods
- Model training and validation must be done on separate data.
- To validate, we need a measure of goodness. Different goodness measures lead to different types of machine learning algorithms.
Features are more important than method

- Given only a single feature, we cannot determine anything about the label (X or Y).
- Given only a single feature, we cannot determine anything about the label (X or Y).
- Given both features, it is quite obvious to see the pattern.
- When features contain strong signal any decent machine learning method should.
- If signal is weak then some methods fail less fatally than others.

Rule-based prediction

<table>
<thead>
<tr>
<th>Feature</th>
<th>Target value</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>Y = 1</td>
</tr>
<tr>
<td>X2</td>
<td>Y = 1</td>
</tr>
<tr>
<td>X3</td>
<td>Y = 0</td>
</tr>
</tbody>
</table>

In robotics, we can use physical models. However, there are some missing sub-models. For example, it is difficult to
quantify exactly how much vertical impact is generated by moving a certain amount of a plane or what happens if we
move a ladder. Thus, we pose two possibilities:

1) Deep physical model and learn the entire response of plane to control signals (difficult, requires five places)
2) Find out unknown parts of a model and try to infer the underlying model with ML
- Requires less data and provides a more adequate model (search space is much smaller)

Why rule-based prediction does not work?

Rule-based prediction is good for finding simple relations. Complex relations require low-support rules.
There are hard to distinguish from random coincidences

Support of a rule

Confidence of a rule

$$\text{conf} = \frac{\# [X_1 = a_1 \land \ldots \land X_k = a_k \land Y = b]}{\# [X_1 = a_1 \land \ldots \land X_k = a_k]}$$

100%
Decision Trees

Decision tree is a systematic way to split the feature space into blocks and assign them majority label for each of those blocks.

In each node we split the feature space (read example) according to some feature.

Example

One possible decision tree that corresponds to the split in the figure is given here.

Note that there are other more complex splits that return the same answer.

Which Attribute is a Best Classifier?

ID3 algorithm uses entropy based criterion for splitting.

Entropy is a quantitative measure of uncertainty.

Low entropy = predictable outcome, high entropy = unpredictable outcome.

To compute entropy, we must first compute frequencies

\[ p_i[Y=a] = \frac{\# \text{example with } Y=a}{\# \text{all examples}} \]

Then we can compute entropy of target variable Y

\[ H(Y) = \sum_{a \in \text{values(Y)}} p_i[Y=a] \cdot \log_2 \frac{1}{p_i[Y=a]} \]

After that we can start to estimate how does the split change the uncertainty of Y.

How to find decision trees?

Almost all decision tree learning algorithms are based on recursive splitting of examples (training data).

A learning algorithm must decide

1) Which feature (attribute) is used for split?
2) What is the split criterion?
3) When to stop splitting?
4) What is the final decision on leaf nodes?

Most algorithms also use postprocessing

- Too many rules are pruned.
- A rule with low support is indistinguishable from coincidence.

How to Measure Information Gain?

Assume that we split the initial example set S into several disjoint sets \( S_1, \ldots, S_k \). Let \( H_S(Y) \) denote entropy of Y in the set S and \( H_{S_i}(Y) \) denote the entropy of Y in the set \( S_i \). Then the corresponding information gain is

\[ \text{Gain} = H_S(Y) - \frac{1}{k} \sum_{i=1}^{k} \frac{1}{|S_i|} \cdot H_{S_i}(Y) \]

ID3 algorithm tries all attributes \( X_i \) for splitting

- To split according to \( X_i \); the set of samples \( S \) is split according to the value of \( X_i \). For instance, if \( X_i \) has values \{a, b, c\} then these sets are formed

\[ S_a = \{x : X_i = a\}, S_b = \{x : X_i = b\}, S_c = \{x : X_i = c\} \]

and then choose the attribute \( X_i \) with highest information gain.
**Planning Strategies**

Naive version of ID3 algorithm tends to overfit, i.e., there exist decision trees that perform better on finite dataset.

- Two basic ways to avoid overfitting:
  1. Stop when splitting does not make sense (use statistical tests to compare it with random data).
  2. Grow the tree but later prune it back.

Standard way to do pruning is to split data into:
- Training data (used to find original tree).
- Validation data (used to evaluate utility of pruning option).

**Rule prioritisation**

Cutting out some sub-trees might delete also some goal rules. Hence, there is an interesting alternative:

- Convert the leafs into rules $(x \in a) \Rightarrow y = b$.
- Drop assumptions that decrease accuracy on validation set.
- Get many new simplified rules.
- Send all rules by estimated accuracy (confidence).

When needed apply a rule that has the highest accuracy among applicable rules.

As you can see, decision trees are nothing more than a more systematic way of finding rules:
- They cover the entire feature space.
- They provide rules with orthogonal assumptions.

**Why does decision tree learning work?**

Assume that we have a large and separable sample set:
- All samples are independently drawn from a distribution.
- Future samples also come from the same distribution.

Then for each restriction $x_1 \in A_1, x_2 \in A_2$, the frequency of target values $P_N(y = a | x_1, x_2) = \epsilon$. In other words, it's rather close to a fixed probability (hitting frequency of samples' approach infinity).

Thus, if splits will be generated if $P_N(y = a | x_1) \neq 1$.
- Correct majority vote is assigned to leaf nodes.

The claim holds even if some percentage of target values are randomly flipped. However, the required sample size is then much larger.

**How to handle continuous variables?**

Since the sample set is finite, we can split continuous variable $x$ always according to observed values, e.g., whether $x$ is smaller than 2.5, 3.1, 4.0.

As a result, we get many alternative splittings.
- Use standard information gain to choose the optimal.
- Try three and four-way splits.

The technique is the same, however, you should try to avoid do pre-specified spliting, since these require homogenous example sizes.
Further hacks and tricks

Some variables shatter the example into tiny sets
→ They get unjustified advantage
→ Use limited number of splits or penalize according to the number of splits (C4.5)

Sometimes some attributes are missing
→ Use imputation or other more complex technique (C4.5)

Some attributes are less costly to observe
→ Add costs into the selection criteria (C4.5)

Sometimes decision borders are not rectangular
→ Use linear thresholding functions that use several variables, e.g.
\[ \frac{x_1}{x_2} = c \] (Aka mixture of experts)

Implementations in R

* Rpart package for classification trees
  and some more exotic stuff

* Tree package for classification and regression trees

* Cubist package for regression trees