What is Machine Learning?

Marco Scutari
scutari@idsia.ch

Dalle Molle Institute for
Artificial Intelligence (IDSIA)

March 8, 2021

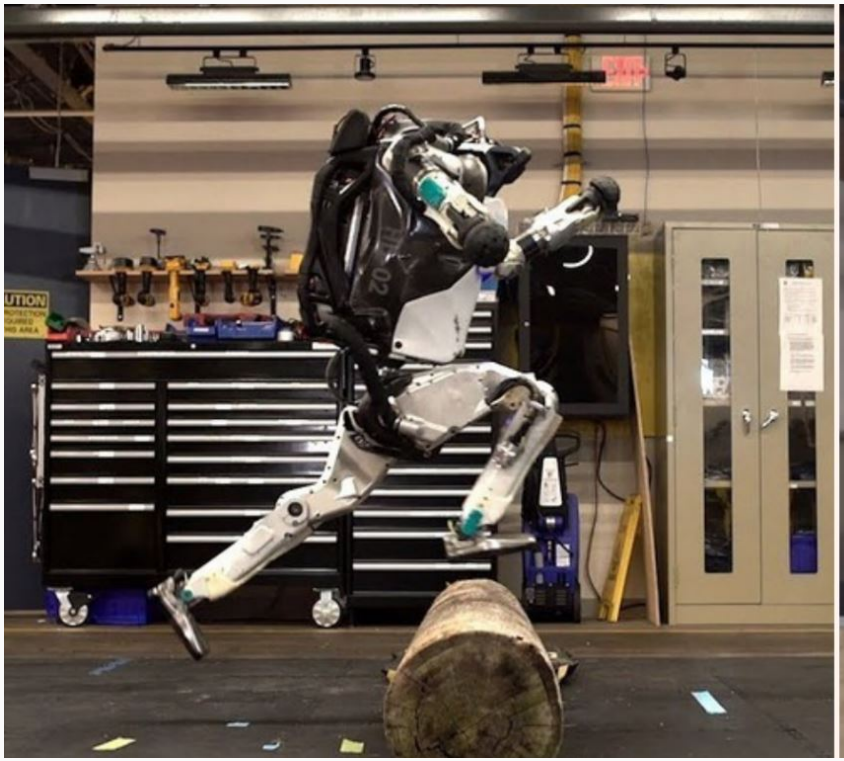
The Grand Vision

Machine learning studies the algorithms and statistical tools that allow computer systems to perform specific, well-defined tasks without explicit instructions. It is a sub-field of artificial intelligence.

Broadly speaking, in order to do this:

1. We need a working model of the world that describes the task and its context in a way a computer can understand.
2. We need a goal: how do we measure the performance of the model? Because that is what we optimise for; usually it is the ability to predict new events.
3. We encode our knowledge of the world drawing information from training data, experts or both; this is called learning.
4. The computer system uses the model as a proxy of reality and, as new inputs come in, to perform inference and decide if/how to perform the assigned task.

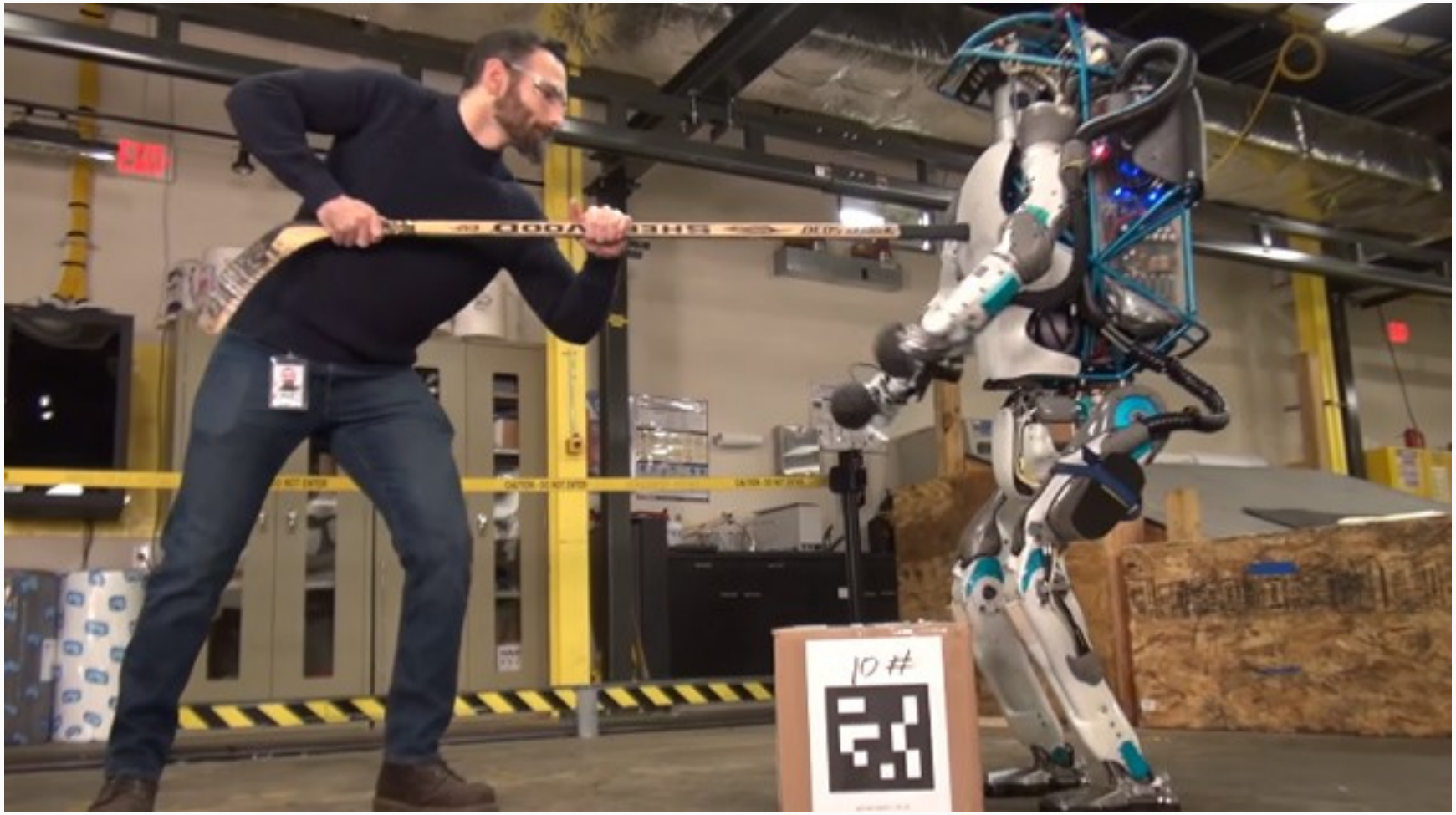
Robotics



Boston Dynamics robots can walk, run, move around (or jump over!) obstacles and carry objects...

... even with pesky humans interfering...

Robotics



... in violent ways!

Playing Games

DeepMind AlphaGo beating the best human Go player!

Diagnosing Diseases

The Unfortunate Reality

https://xkcd.com/1838

However, building machine learning applications is far from trivial and it is a craft as much as it is a science.

- It requires large amounts of data, which has to be collected keeping the goal of the model in mind.
- It is difficult to decide how to structure the model from a mathematical and probabilistic point of view
- It is difficult to evaluate and troubleshoot models of any real-world complexity.

Identify the Variables to Include in the Model

The first step in building a machine learning model is to choose which variables to include. Which aspects of/entities in the world do we need the model to represent for the computer to carry out the assigned task? This is known as feature selection.

- Each aspect of the world or entity is modelled with one random variable.
- We should use a small enough number of variables because if we have too many:
  - it is difficult it is to construct the model;
  - it is difficult to interpret and troubleshoot it;
  - the model requires too much computing power to learn and to run.
- We must choose which are the relevant events that make up the sample space of each variable, again taking care of not having too many.

Identify the Variables to Include in the Model

For instance, to play a game of Go we need to model each piece using its position on the board (which provides a regular grid to use for coordinates) and which player it belongs to.

In robotics applications:

- the position of the robot, and the positions of its hands relative to its body?
- the size and position of all the objects in the room?
- the size and position of the box to pick up?

All of these can be guessed from the information the robot gets from the sensors it is equipped with (camera, infrared, radar, etc.). Then there is the question of how to represent them (coordinates on a grid, and sizes in increments of 5cm? real numbers?).

In clinical applications we need additional machine learning to even figure out which variables we should include in our model...

Generative versus Discriminative Models

The second step is choosing which class of machine learning models to select from.

- Generative models: we have a set of variables $X_1, \ldots, X_N$ describing various components of a complex phenomenon, and we are interested in modelling that phenomenon in a mechanistic way. Hence, we want to show how the various parts interact with each other, and in order to do that we choose to model their joint probability $\mathrm{P}(X_1, \ldots, X_N)$.
- Discriminative models: we have one particular variable (say, X_1) that is closely tied with our model task, and a number of other variables ($X_2, \ldots, X_N$) which we believe can be used to explain it. We do not care about how the X_i are related to each other, so we just model $\mathrm{P}(X_1 \mid X_2, \ldots, X_N)$.

Generative versus Discriminative Models

Generative Models

Pros

More flexible in terms of what questions they can answer.

Cons

More complicated to formulate.
Require more data to learn.

Discriminative Models

Pros

Require less data to learn.
Better at prediction than generative models.

Cons

Exclusively focused on predicting the variable of interest.

Model Relationships Between Variables

How do we decide whether there is a relationships between variables? If we had perfect knowledge we could completely describe the world. But we never have perfect knowledge:

- in a game of Go, some moves are more likely than others but we do not know for sure what more our opponent will pick;
- in robotics we are limited by what the sensors can tell us;
- in clinical applications we are limited by what we can learn from patients.

Hence we use the language of probability, and we say that two variables are associated if the occurrence of an event in one variable affects the probability of an event occurring in another variable. In other words, they are associated if they are not independent, possibly given other variables.

An example: the Car Start Problem

"*In the morning, my car will not start. The start engine turns, but nothing happens. The battery is OK. The problem may be due to dirty spark plugs or the fuel may be stolen. I look at the fuel meter. It shows* $1/2$*, hence I expect the spark plugs to be dirty.*"

We need to formalise this kind of reasoning into a model that a computer can understand.

- What made we think of fuel and spark plugs?
- Why did we look at the fuel meter?
- Why had fuel meter reading an impact on our belief that spark plugs are dirty?

Car Start: Variables and Events

From the problem description, our task is to explain why the car will not start using a model comprising these four variables:

- `Fuel`;
- `Spark Plugs`;
- `Start`;
- `Fuel Meter`.

But how do we model the sample space in terms of events?

	Realistic	Pragmatic
`Fuel`	`0%–100%`	`Yes, No`
`Spark Plugs`	`Work, Fault`	`Work, Fault`
`Start`	`Yes, No`	`Yes, No`
`Fuel Meter`	`0%–100%`	`Empty, Half, Full`

Car Start: Which Variables Are Associated?

In a generative model, $\mathrm{P}(\texttt{Fuel}, \texttt{Spark Plugs}, \texttt{Fuel Meter}, \texttt{Start})$ is our probabilistic model in its most general form. It leads to a table with $2 \times 2 \times 2 \times 3 = 24$ probabilities; it would be difficult for us to choose accurate values for each of them.

We could simply use the chain rule to write the model as

$$\begin{aligned}
&\mathrm{P}(\texttt{Start}, \texttt{Fuel Meter}, \texttt{Fuel}, \texttt{Spark Plugs}) \\
&\quad = \mathrm{P}(\texttt{Start} \mid \texttt{Fuel Meter}, \texttt{Fuel}, \texttt{Spark Plugs}) \\
&\qquad \mathrm{P}(\texttt{Fuel Meter} \mid \texttt{Fuel}, \texttt{Spark Plugs}) \\
&\qquad \mathrm{P}(\texttt{Fuel} \mid \texttt{Spark Plugs})\, \mathrm{P}(\texttt{Spark Plugs})
\end{aligned}$$

but that does not change the complexity model, even if it breaks it apart in smaller pieces.

To actually make it simpler we should ask: do we know which variables are associated with each other, and which are not?

Car Start: Expert Knowledge of Associations

In our (expert?) knowledge:

- `Start` is associated with `Fuel`: from the definition of independence

$$\mathrm{P}(\texttt{Start} = \texttt{Yes} \mid \texttt{Fuel} = \texttt{No}) = 0 \neq \mathrm{P}(\texttt{Start} = \texttt{Yes}) > 0$$

  so the two are not independent.

- Similarly, `Start` is associated with `Spark Plugs`.

- `Fuel Meter` is associated with `Fuel`, because the former is a transformation of the latter. It is also easy to see that

$$\mathrm{P}(\texttt{Fuel Meter} = \texttt{Full} \mid \texttt{Fuel} = \texttt{No}) = 0 \neq \mathrm{P}(\texttt{Fuel Meter} = \texttt{Full}) > 0.$$

Car Start: Expert Knowledge of Associations

- `Start` is not associated with `Fuel Meter` given `Fuel` for the same reason: if we know the amount of `Fuel`, whether `Fuel Meter` is `Empty`, `Half` or `Full` should not alter the probability that `Start` is `Yes` or `No`.
- `Fuel` is not associated with `Spark Plugs` because knowing whether `Fuel` is `Yes` or `No` should not alter the probability that `Spark Plugs` are `Work` or `Fault`.

We can take these considerations and use them to make the model simpler by removing the variables we do not need from the conditional probabilities.

Car Start: a Simpler Model

Hence we are left with:

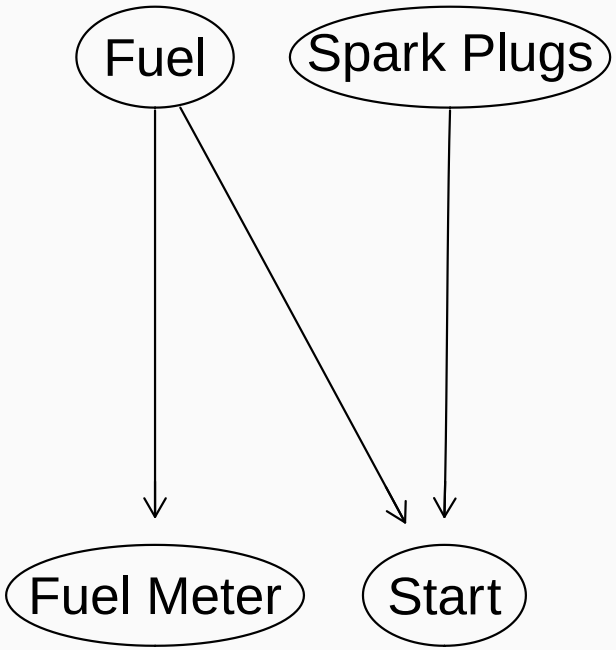
$$\begin{aligned}&\mathrm{P}(\texttt{Start}, \texttt{Fuel Meter}, \texttt{Fuel}, \texttt{Spark Plugs}) \\ &\quad = \mathrm{P}(\texttt{Start} \mid \texttt{Fuel}, \texttt{Spark Plugs})\, \mathrm{P}(\texttt{Fuel Meter} \mid \texttt{Fuel}) \\ &\qquad\qquad \mathrm{P}(\texttt{Fuel})\, \mathrm{P}(\texttt{Spark Plugs})\end{aligned}$$

If we represent this model as a graph, with

- variables as nodes and
- associations as arcs

we get a qualitative view of what our model looks like.

This representation is the key idea of Bayesian networks and of many other classes of machine learning modules.



Car Start: Probabilistic versus Causal Construction

In probability associations are symmetric; the derivation of Bayes' theorem makes it really clear that

$$\mathrm{P}(X_1 \mid X_2)\,\mathrm{P}(X_2) = \mathrm{P}(X_1, X_2) = \mathrm{P}(X_2 \mid X_1)\,\mathrm{P}(X_1).$$

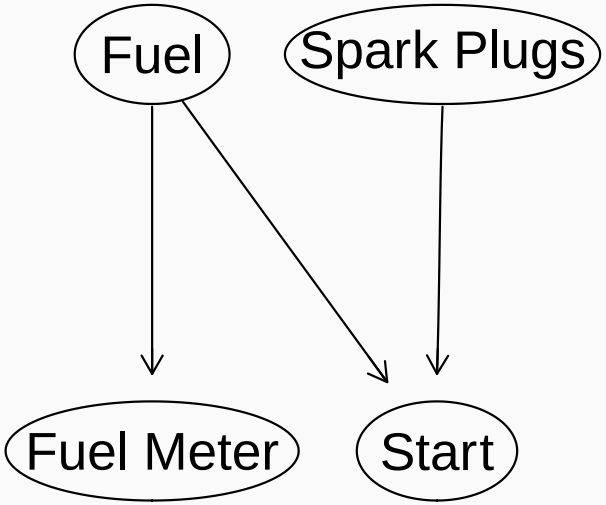
In order to write the conditional probabilities, we used common sense to choose the conditioning variables such that they affect the conditioned variables.

But what does that mean from a modelling point of view? It means we are giving arcs a causal interpretation and we choose arc directions to go from cause (nodes) to effect (nodes).

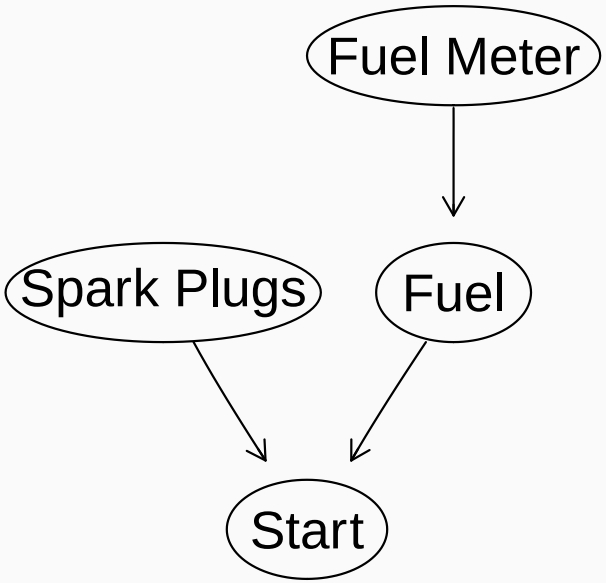
How do we do that?

Car Start: Playing with Arc Directions

$$\mathrm{P}(\texttt{Start}, \texttt{Fuel Meter}, \texttt{Fuel}, \texttt{Spark Plugs})$$
$$= \mathrm{P}(\texttt{Start} \mid \texttt{Fuel}, \texttt{Spark Plugs})$$
$$\mathrm{P}(\texttt{Fuel Meter} \mid \texttt{Fuel})\,\mathrm{P}(\texttt{Fuel})$$
$$\mathrm{P}(\texttt{Spark Plugs})$$



$$\mathrm{P}(\texttt{Start}, \texttt{Fuel Meter}, \texttt{Fuel}, \texttt{Spark Plugs})$$
$$= \mathrm{P}(\texttt{Start} \mid \texttt{Fuel}, \texttt{Spark Plugs})$$
$$\mathrm{P}(\texttt{Fuel Meter})\,\mathrm{P}(\texttt{Fuel} \mid \texttt{Fuel Meter})$$
$$\mathrm{P}(\texttt{Spark Plugs})$$



Car Start: Playing with Arc Directions

The criterion to identify causes and effect is intervention. Consider:

- If we fill the tank with fuel, the fuel meter goes up.
- If we tamper with the fuel meter to make is say `Full`, the fuel tank does not magically refill itself.

Hence, `Fuel` is the cause and `Fuel Meter` is the effect and the most intuitive arc direction is $\texttt{Fuel} \rightarrow \texttt{Fuel Meter}$.

What the probability $\mathrm{P}(\texttt{Fuel Meter} \mid \texttt{Fuel})$ tells us is just that if the fuel meter says `Full` there probably is fuel in the tank, whereas if the fuel meter says `Empty` there may be no fuel in the tank (assuming the fuel meter works reliably).

What if we do not have expert knowledge of associations, so we are stuck at

$$\mathrm{P}(\texttt{Start}, \texttt{Fuel Meter}, \texttt{Fuel}, \texttt{Spark Plugs}) = ?$$

It is possible to learn associations from data to select a good model. The key idea is that:

1. we collect data (that is, sets of values for all the variables in the model);
2. we take different models and we compute the probability they give to the data;
3. we choose the model that gives the highest probability to the data, taking the complexity of the model into consideration.

Car Start: the Conditional Probabilities

Spark Plugs	
Work	Fault
?	?

Fuel	
Yes	No
?	?

Fuel Meter	Fuel = Yes	Fuel = No
Empty	?	?
Half	?	?
Full	?	?

Start

	Spark Plugs = Work	
	Fuel = Yes	Fuel = No
Yes	?	?
No	?	?

	Spark Plugs = Fault	
	Fuel = Yes	Fuel = No
Yes	?	?
No	?	?

After we decide that this model is good to go, we need to fill in the values of all the conditional probabilities that are implied by the model.

The number of these probabilities gives the complexity of the model ($2 + 2 + 6 + 8 = 18 < 24$).

Car Start: the Conditional Probabilities

Again, we have two ways of doing that:

- ask someone with expert knowledge who can tell us which values to fill in;
- estimate the (conditional) probabilities from the data.

Starting from expert knowledge, we can create the probability and conditional probability tables as follows.

1. Store the events in the sample space for each variable.

```
Fuel.lvl = c("Yes", "No")
Spark.Plugs.lvl = c("Work", "Fault")
Fuel.Meter.lvl = c("Full", "Half", "Empty")
Start.lvl = c("Yes", "No")
```

Car Start: Expert Probabilities in R

2. Create the probability tables labelling rows and columns for easy reference.

```
Fuel.probs = array(c(0.98, 0.02), dim = 2, dimnames = list(Fuel = Fuel.lvl))
Spark.Plugs.probs = array(c(0.96, 0.04), dim = 2,
                          dimnames = list(Spark.Plugs = Spark.Plugs.lvl))
Fuel.Meter.probs = array(c(0.39, 0.60, 0.01, 0.01, 0.01, 0.98), dim = c(3, 2),
                         dimnames = list(Fuel.Meter = Fuel.Meter.lvl,
                                         Fuel = Fuel.lvl))
Start.probs = array(c(0.99, 0.01, 0, 1, 0.01, 0.99, 0, 1), dim = c(2, 2, 2),
                    dimnames = list(Start = Start.lvl, Fuel = Fuel.lvl,
                                    Spark.Plugs = Spark.Plugs.lvl))
```

3. Store all probability tables together in a list.

```
expert.probabilities = list(
  Fuel = Fuel.probs,
  Spark.Plugs = Spark.Plugs.probs,
  Fuel.Meter = Fuel.Meter.probs,
  Start = Start.probs
)
```

Car Start: Expert Probabilities in R

The end result is this collection of expert probabilities, organised in tables:

```
expert.probabilities$Fuel
```

```
  Fuel
   Yes   No
  0.98 0.02
```

```
expert.probabilities$Spark.Plugs
```

```
  Spark.Plugs
   Work Fault
   0.96  0.04
```

```
expert.probabilities$Fuel.Meter
```

```
             Fuel
  Fuel.Meter  Yes   No
       Full  0.39 0.01
       Half  0.60 0.01
       Empty 0.01 0.98
```

```
expert.probabilities$Start
```

```
, , Spark.Plugs = Work

      Fuel
Start  Yes No
  Yes 0.99  0
  No  0.01  1

, , Spark.Plugs = Fault

      Fuel
Start  Yes No
  Yes 0.01  0
  No  0.99  1
```

How can we get the same thing from data, without using an expert?

Car Start: Probabilities from Data in R

Suppose we collect a number of data points for the four variables in the model.

```
dim(training.set)
```

```
[1] 500   4
```

```
head(training.set)
```

```
  Fuel Fuel.Meter Spark.Plugs Start
1  Yes       Full        Work   Yes
2  Yes       Half        Work   Yes
3  Yes       Full        Work   Yes
4  Yes       Half        Work   Yes
5  Yes       Half        Work   Yes
6  Yes       Half        Work   Yes
```

We can use these data to compute the frequentist probabilities we need to fill in the model using the corresponding relative frequencies.

Car Start: Probabilities from Data in R

Again, we can compute the absolute frequencies (`*.counts`) and then the relative frequencies (`*.probs`) for each variable, making sure we get the right conditional probabilities from `prop.table()`.

```
Fuel.counts = table(training.set[, "Fuel"])
Fuel.probs = prop.table(Fuel.counts)
Spark.Plugs.counts = table(training.set[, "Spark.Plugs"])
Spark.Plugs.probs = prop.table(Spark.Plugs.counts)
Fuel.Meter.counts = table(training.set[, c("Fuel.Meter", "Fuel")])
Fuel.Meter.probs = prop.table(Fuel.Meter.counts, margin = 2)
Start.counts = table(training.set[, c("Start", "Fuel", "Spark.Plugs")])
Start.probs = prop.table(Start.counts, margin = 2:3)
```

We can then organise them in a list as we did for the expert probabilities.

```
probability.tables = list(
  Fuel = Fuel.probs,
  Spark.Plugs = Spark.Plugs.probs,
  Fuel.Meter = Fuel.Meter.probs,
  Start = Start.probs
)
```

Car Start: Probabilities from Data in R

The end result is this collection of frequentist probabilities, organised in the same way as the expert probabilities:

```
probability.tables$Fuel
```

```
   Yes    No
 0.988 0.012
```

```
probability.tables$Spark.Plugs
```

```
  Work Fault
 0.972 0.028
```

```
probability.tables$Fuel.Meter
```

```
            Fuel
Fuel.Meter     Yes      No
     Full  0.37854 0.00000
     Half  0.61538 0.00000
     Empty 0.00607 1.00000
```

```
probability.tables$Start
```

```
, , Spark.Plugs = Work

      Fuel
Start    Yes     No
  Yes 0.9833 0.0000
  No  0.0167 1.0000

, , Spark.Plugs = Fault

      Fuel
Start    Yes No
  Yes 0.0000
  No  1.0000
```

Each set of probabilities gives a different model for the car start problem.

Spot the Difference

- Expert probabilities are "nicer" in the sense that they are usually reasonably round numbers with 2-3 decimal places.
- Expert probabilities give probabilities greater than zero to events that are really rare, because the experts know these events are not impossible. (An example: `Fuel Meter` $=$ `Full` when `Fuel` $=$ `No`; fuel meters are very reliable but they definitely can break.)
- Frequentist probabilities are limited by the data we can collect; we may not actually observe rare events and then they get a probability of exactly zero. But they are not impossible!
- For this reason, frequentist probabilities are bad at representing small (≈ 0) and large (≈ 1) probabilities because they have a granularity of $1/$`nrow(training.set)` (here $1/500$).
- And if `nrow(training.set)` is small they are bad at representing any probability.

Training and Validation Data Sets

In machine learning we assess how well a model works by looking at how accurate it is in predicting new observations. To do this we need a data set that has not been previously used to define the model (such as computing frequentist probabilities); this second data set is called the validation data set. It is "new" in the sense that the model has hot seen these data before.

In contrast, the data set we use to define the model is called the training data set, because it is used to train the model.

```
head(validation.set)
```

```
  Fuel Fuel.Meter Spark.Plugs Start
1   No      Empty        Work    No
2  Yes       Full        Work   Yes
3  Yes       Half        Work   Yes
4  Yes       Half        Work   Yes
5  Yes       Half        Work   Yes
6  Yes       Full        Work   Yes
```

Probability of the Validation Data Set

If we take the first data point from the validation set, we have complete set of values for all the variables and we can feed those values to the model to compute their joint probability.

```
new.data = validation.set[1, ]
probability.tables$Fuel[new.data$Fuel]
```

```
    No
 0.012
```

```
probability.tables$Spark.Plugs[new.data$Spark.Plugs]
```

```
  Work
 0.972
```

```
probability.tables$Fuel.Meter[new.data$Fuel.Meter, new.data$Fuel]
```

```
[1] 1
```

```
probability.tables$Start[new.data$Start, new.data$Fuel, new.data$Spark.Plugs]
```

```
[1] 1
```

So, the probability of that data point is $0.0120.97211 \approx 0.012$.

Probability of the Validation Data Set

If we iterate over all the validation set, and we multiply all the resulting probabilities together we obtain its predictive probability.

```
validation.set.probability = 1
for (i in seq(nrow(validation.set))) {

  val = validation.set[i, ]
  new.prob = probability.tables$Fuel[val$Fuel] *
             probability.tables$Spark.Plugs[val$Spark.Plugs] *
             probability.tables$Fuel.Meter[val$Fuel.Meter, val$Fuel] *
             probability.tables$Start[val$Start, val$Fuel, val$Spark.Plugs]

  validation.set.probability = validation.set.probability * new.prob

}#FOR
as.numeric(validation.set.probability)
```

```
[1] 7.1e-44
```

Why do we care about having a model with a good predictive probability?

Why the Probability of the Validation Data Set

The probability of the validation test is a measure of predictive accuracy, that is, the ability of the model to predict new events. The reason why we want a model that maximises it is as follows.

1. The four variables in the model can take 24 combinations of values.
2. Some combinations of their values will have higher probabilities (according to the model) than others.
3. In the validation sets, some combinations of values will appear more frequently than others.
4. If the machine learning model is a good working model of the world, it should assign high probability to combinations of values that appear more often.
5. Hence, we want a model that gives a high probability to the validation set as a whole.

The Log-Probability is What You Actually Want

Predictive accuracy is usually measured on a log-scale; on its natural scale it becomes too small very quickly when the validation set or the number of variables grow. It's already $\approx 10^{-43}$ for this simple model!

```
validation.log.probability = 0
for (i in seq(nrow(validation.set))) {

  val = validation.set[i, ]
  new.prob = probability.tables$Fuel[val$Fuel] *
             probability.tables$Spark.Plugs[val$Spark.Plugs] *
             probability.tables$Fuel.Meter[val$Fuel.Meter, val$Fuel] *
             probability.tables$Start[val$Start, val$Fuel, val$Spark.Plugs]

  validation.log.probability = validation.log.probability + log(new.prob)

}#FOR
as.numeric(validation.log.probability)
```

```
[1] -99.4
```

(The model with the expert probabilities gets -99.298, which is about the same.)

Car Start: Predicting Start

Sometimes we may also be interested in the accuracy of predicting just one variable given some other variables. For instance, we would like to predict `Start` from the other variables.

```
errors = 0
for (i in seq(nrow(validation.set))) {

  val = validation.set[i, ]
  base.prob = probability.tables$Fuel[val$Fuel] *
              probability.tables$Spark.Plugs[val$Spark.Plugs] *
              probability.tables$Fuel.Meter[val$Fuel.Meter, val$Fuel]
  Yes.prob = base.prob *
              probability.tables$Start["Yes", val$Fuel, val$Spark.Plugs]
  No.prob = base.prob *
              probability.tables$Start["No", val$Fuel, val$Spark.Plugs]

  if (ifelse(Yes.prob > No.prob, "Yes", "No") != val$Start)
    errors = errors + 1

}#FOR
errors / nrow(validation.set)
```

```
[1] 0.01
```

Car Start: Interrogating the Model

A more general way of using a model is to interrogate it: we have some evidence on some of the variables (that is, we assume we know their values), and we would like to know the the probability of some event.

For instance: say that $\texttt{Fuel Meter} = \texttt{Half}$. How does $\mathrm{P}(\texttt{Start} = \texttt{Yes})$ change after we introduce this evidence in the model?

Predicting `Start` from all the other variables is a particular case in which we have evidence on all the other variables.

Car Start: the Exhaustive (Dumb) Way

Using probability axiom #2, we can write

$$\mathrm{P}(\texttt{Start} = \texttt{Yes}) = \mathrm{P}(\texttt{Start} = \texttt{Yes}, \texttt{Fuel} = \texttt{Yes}) + \mathrm{P}(\texttt{Start} = \texttt{Yes}, \texttt{Fuel} = \texttt{No})$$

and then, recursively,

$$\mathrm{P}(\texttt{Start} = \texttt{Yes}, \texttt{Fuel} = \texttt{Yes}) = \mathrm{P}(\texttt{Start} = \texttt{Yes}, \texttt{Fuel} = \texttt{Yes}, \texttt{Spark.Plugs} = \texttt{Work}) + \mathrm{P}(\texttt{Start} = \texttt{Yes}, \texttt{Fuel} = \texttt{Yes}, \texttt{Spark.Plugs} = \texttt{Fault})$$

$$\begin{aligned}
&\mathrm{P}(\texttt{Start} = \texttt{Yes}, \texttt{Fuel} = \texttt{Yes}, \texttt{Spark.Plugs} = \texttt{Work}) \\
&= \mathrm{P}(\texttt{Start} = \texttt{Yes}, \texttt{Fuel} = \texttt{Yes}, \texttt{Spark.Plugs} = \texttt{Work}, \texttt{Fuel.Meter} = \texttt{Full}) + \\
&\quad \mathrm{P}(\texttt{Start} = \texttt{Yes}, \texttt{Fuel} = \texttt{Yes}, \texttt{Spark.Plugs} = \texttt{Work}, \texttt{Fuel.Meter} = \texttt{Half}) + \\
&\quad \mathrm{P}(\texttt{Start} = \texttt{Yes}, \texttt{Fuel} = \texttt{Yes}, \texttt{Spark.Plugs} = \texttt{Work}, \texttt{Fuel.Meter} = \texttt{Empty})
\end{aligned}$$

Car Start: the Exhaustive (Dumb) Way

In practice this means that, for small models, we can just go through all combinations of values of the other variables.

```
yes.prob = 0
for (FL in Fuel.lvl)
  for (SP in Spark.Plugs.lvl)
    for (FM in Fuel.Meter.lvl) {

      yes.prob = yes.prob +
                 expert.probabilities$Fuel[FL] *
                 expert.probabilities$Spark.Plugs[SP] *
                 expert.probabilities$Fuel.Meter[FM, FL] *
                 expert.probabilities$Start["Yes", FL, SP]

      }#FOR
as.numeric(yes.prob)
```

```
[1] 0.932
```

So $\mathrm{P}(\texttt{Start} = \texttt{Yes}) = 0.932$, and $\mathrm{P}(\texttt{Start} = \texttt{No}) = 1 - 0.932 = 0.068$.

Car Start: the Exhaustive (Dumb) Way

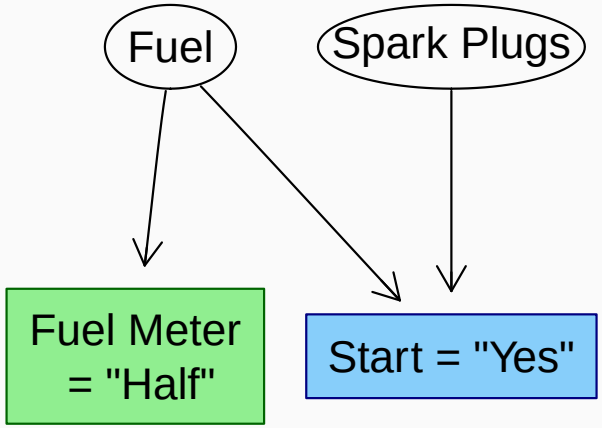
```
half.prob = yes.and.half.prob = 0
for (FL in Fuel.lvl)
  for (SP in Spark.Plugs.lvl) {

    yes.and.half.prob = yes.and.half.prob + expert.probabilities$Fuel[FL] *
                          expert.probabilities$Spark.Plugs[SP] *
                          expert.probabilities$Fuel.Meter["Half", FL] *
                          expert.probabilities$Start["Yes", FL, SP]

  }#FOR
for (FL in Fuel.lvl)
  for (SP in Spark.Plugs.lvl)
    for (ST in Start.lvl) {

      half.prob = half.prob + expert.probabilities$Fuel[FL] *
                    expert.probabilities$Spark.Plugs[SP] *
                    expert.probabilities$Fuel.Meter["Half", FL] *
                    expert.probabilities$Start[ST, FL, SP]

    }#FOR
as.numeric(yes.and.half.prob) / as.numeric(half.prob)
```

```
[1] 0.95
```

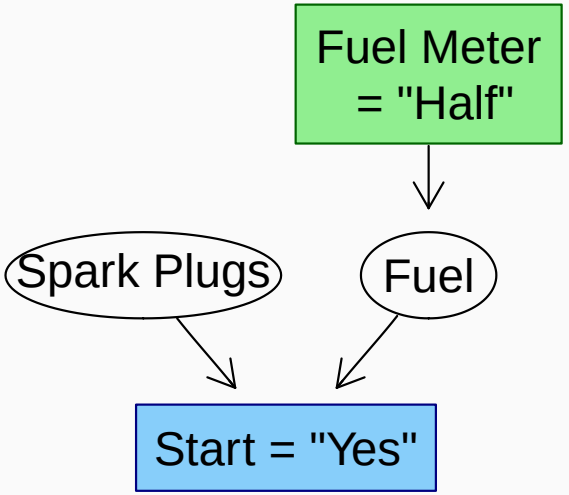
But is there a more efficient way of doing the same thing?

Car Start: the Principled (Probabilistic) Way

$$\begin{aligned}
&\mathrm{P}(\texttt{Start} = \texttt{Yes}, \texttt{Fuel Meter} = \texttt{Half},\\
&\qquad \texttt{Fuel}, \texttt{Spark Plugs}) =\\
&= \mathrm{P}(\texttt{Start} = \texttt{Yes} \mid \texttt{Fuel}, \texttt{Spark Plugs})\\
&\quad \mathrm{P}(\texttt{Fuel Meter} = \texttt{Half} \mid \texttt{Fuel})\\
&\quad \mathrm{P}(\texttt{Fuel})\,\mathrm{P}(\texttt{Spark Plugs})
\end{aligned}$$

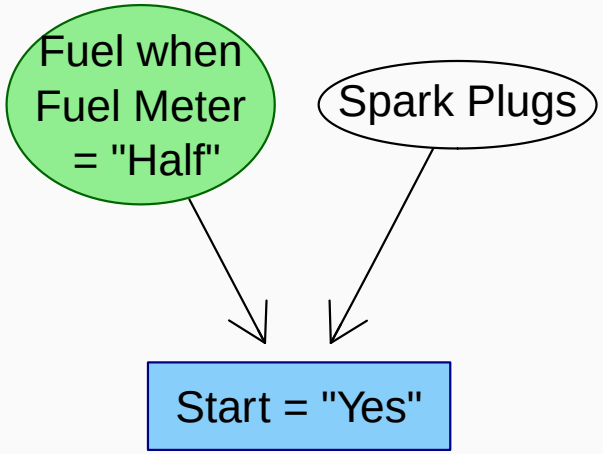


$$\begin{aligned}
&\mathrm{P}(\texttt{Start} = \texttt{Yes}, \texttt{Fuel Meter} = \texttt{Half},\\
&\qquad \texttt{Fuel}, \texttt{Spark Plugs})\\
&= \mathrm{P}(\texttt{Start} \mid \texttt{Fuel}, \texttt{Spark Plugs})\\
&\quad \mathrm{P}(\texttt{Fuel} \mid \texttt{Fuel Meter} = \texttt{Half})\\
&\quad \frac{\mathrm{P}(\texttt{Fuel.Meter} = \texttt{Half})}{\cancel{\mathrm{P}(\texttt{Fuel})}}\\
&\quad \cancel{\mathrm{P}(\texttt{Fuel})}\,\mathrm{P}(\texttt{Spark Plugs})
\end{aligned}$$



Car Start: the Principled (Probabilistic) Way

$$\begin{aligned}
&\mathrm{P}(\texttt{Start} = \texttt{Yes}, \texttt{Fuel}, \texttt{Spark Plugs} \mid \\
&\qquad \texttt{Fuel Meter} = \texttt{Half}) \\
&= \mathrm{P}(\texttt{Start} \mid \texttt{Fuel}, \texttt{Spark Plugs}) \\
&\quad \mathrm{P}(\texttt{Fuel} \mid \texttt{Fuel Meter} = \texttt{Half}) \\
&\quad \frac{\cancel{\mathrm{P}(\texttt{Fuel.Meter} = \texttt{Half})}}{\cancel{\mathrm{P}(\texttt{Fuel.Meter} = \texttt{Half})}} \\
&\quad \mathrm{P}(\texttt{Spark Plugs})
\end{aligned}$$



This leaves three variables, of which $\texttt{Start}$ is fixed to $\texttt{Yes}$: hence we have to consider $\mathrm{P}(\texttt{Start} = \texttt{Yes})$ under four scenarios:

$$\texttt{Fuel} = \texttt{Yes} \mid \texttt{Fuel Meter} = \texttt{Half}, \texttt{Spark Plugs} = \texttt{Work}$$
$$\texttt{Fuel} = \texttt{Yes} \mid \texttt{Fuel Meter} = \texttt{Half}, \texttt{Spark Plugs} = \texttt{Fault}$$
$$\texttt{Fuel} = \texttt{No} \mid \texttt{Fuel Meter} = \texttt{Half}, \texttt{Spark Plugs} = \texttt{Work}$$
$$\texttt{Fuel} = \texttt{No} \mid \texttt{Fuel Meter} = \texttt{Half}, \texttt{Spark Plugs} = \texttt{Fault}$$

and sum the corresponding $\mathrm{P}(\texttt{Start} = \texttt{Yes} \mid \text{scenario})\,\mathrm{P}(\text{scenario})$.

Summary and Remarks

- Machine learning aims to make computer systems able to learn from and carry out tasks in the real world.
- Machine learning models represent a model of the world in a form useful to a computer, and use the language of probability to represent uncertainty.
- The focus of machine learning models is prediction, and by extension probabilistic reasoning and inference, so that the computer system can use the models to decide how to react to its environment.
- Generative models are better for reasoning, discriminative models are better for prediction.

Thanks!

Any questions?