Lecture 1: PROPOSITIONAL LOGIC [*]

10-607 COMPUTATIONAL FOUNDATIONS FOR MACHINE LEARNING

1 Arguments

Arguments are everywhere: from such-and-such assumptions follow these-and-those conclusions. Arguments vary in their level of formality, but they all follow similar rules:

- They start from assumptions;
- They proceed in a sequence of steps;
- For each step, they introduce some new conclusions that are claimed to follow from earlier facts (that is, from assumptions and previous conclusions);
- Each step is at least potentially checkable.

The more formal an argument is, the more rigorously it follows these rules: an informal argument might skip steps (in the hopes that the reader or listener will fill them in), or might include steps that are heuristic i.e. steps that don't work 100% of the time, and therefore can't be perfectly checked. A formal argument will try to be more detailed and rigorous.

Even quite formal arguments, though, tend to take shortcuts. It's moderately rare to see an argument that is so completely and rigorously written out that every step is perfectly specified and airtight, since such an argument would be so long and detailed that nobody would bother to follow the whole thing.

In fact, the only place where fully-detailed arguments are commonly encountered is inside a computer-assisted proof system, such as L∃∀N, Coq, or Keymaera. These systems try to make proofs both human-readable and perfectly formal by hiding details that an automated proof search can fill in, and expanding hidden details on demand.

Even when we hide them, though, the details are what makes or breaks an argument. Just like a bug in a program, a single missing detail in an argument can render the whole thing useless. It's even worse when the missing detail is hidden inside a step we skipped over.

For this reason, every time we take a shortcut in an argument, we want to make sure we understand how the longer, fully-rigorous version of the argument would work; else a bug could be hidden in the part we skipped. When in doubt, write it out: if you're not completely sure that a step works, then even if it does, many of your readers or listeners may have trouble following it.

[*]Compiled on Thursday 14th September, 2023 at 18:57

A final thing to remember about all arguments: their purpose is to convince the listener. That means that the style of an argument is important, not just its content. For example, it's a good idea to

- Keep an *audience model*, a mental picture of what the readers or listeners already know vs. what we need to convince them of;
- Organize arguments into digestible chunks that can be understood independently;
- Be clear: state what is an assumption, what each new conclusion depends on, and what reasoning method is being used at each step;
- Annotate the structure, so that the reader or listener can understand at every point what conclusions we're currently trying to argue for.

2 Formal logic

We're going to try to understand arguments by developing a completely precise system for describing them: the system of *formal logic*. An argument in formal logic is called a *proof*.

There's not just one system of formal logic. Instead, there are many related systems, each with a different focus and purpose. Examples include:

- Classical propositional logic
- Constructive propositional logic
- Classical first-order logic
- Linear logic
- Temporal logic
- ... and many more.

Different systems might be able to represent different concepts, or might be able to represent the same concepts more or less efficiently. Different systems might even contradict each other: the same conclusion might be provable in one system but not another.

An example of this kind of difference is classical vs. constructive logic: the inference system for constructive logic allows only constructive proofs, and so there are some conclusions that we can prove in classical logic that turn out not to hold in constructive logic.

But, these systems all share some features. The biggest is that they all have two complementary parts: first, a language for stating our assumptions and conclusions, and second, a set of rules for manipulating these statements and using them to construct proofs.

Perhaps the simplest and best known of these systems is classical propositional logic; it's also at the heart of many other, more complicated reasoning systems. So, we'll start there.

3 Propositional logic expressions

In propositional logic, variables $a, b, \ldots$ (called *propositions*) represent truth values T or F. These variables can be combined using *connectives* such as $\wedge, \vee, \neg, \rightarrow$, meaning AND, OR, NOT, IMPLIES. The expressions of propositional logic are exactly the ones we can construct recursively by starting from propositions and applying connectives: for example, $a \wedge \neg b$.

The precedence is that $\neg$ binds tightest, followed by $\vee, \wedge$ (equal precedence), followed by $\rightarrow$. In case of ambiguity, or if we want a different order, we can use parentheses for grouping: $[(\{\})]$.

There are alternate notations for many of these concepts: you may see $\bar{a}$ or $\tilde{}a$ instead of $\neg a$; you may see $a\&b$ or ab instead of $a \wedge b$; you may see $a \mid b$ or $a + b$ instead of $a \vee b$; you may see $a \supset b$ or $a \Rightarrow b$ instead of $a \rightarrow b$; and you may see $\top, \bot$ instead of T, F. Falsehood F is sometimes also called *absurdity* or *contradiction*.

For readability, we'll often use longer variable names: e.g., done or happy(John). The latter name looks like a function call, and in fact we'll give it that semantics later; but for now it is just a string of symbols that represents a propositional variable.

4 Propositional logic semantics

We can interpret each expression of propositional logic as a true-or-false statement that we might have evidence for. For example, if we assert the expression a, we are claiming to have have evidence for the proposition a.

Similarly, if we assert $a \wedge b$, we are claiming to have evidence for both a and b. If we assert $a \vee b$, it means that we have evidence for either a or b (or both). And, $\neg a$ means that we have evidence that a cannot be true.

The connective that causes the most confusion is $\rightarrow$. If I state $a \rightarrow b$, I am making a promise: if you give me evidence for a, I will provide evidence for b. The confusing part is what happens when a is false. In this case, nobody can provide me evidence for a. So, $a \rightarrow b$ is an empty promise: I'll never have to provide evidence for b. Therefore, in classical propositional logic, whenever $\neg a$ holds, $a \rightarrow b$ is vacuously true.

Because of this semantics, in classical logic, $a \rightarrow b$ is equivalent to $\neg a \vee b$. That is, out of all possible settings for a and b, the only one that makes $a \rightarrow b$ false is when a is true and b is false:

a	b	$a \rightarrow b$
F	F	T
F	T	T
T	F	F
T	T	T

Knowledge check

1. **Select one:** Using the variables plays_all_day(Jack), works_all_day(Jack), tired(Jack), and hungry(Jack), how would you express the proposition ”If Jack works all day or plays all day, he will be hungry and tired”?
   A. (plays_all_day(Jack) $\wedge$ works_all_day(Jack)) $\vee$ (tired(Jack) $\vee$ hungry(Jack))
   B. (plays_all_day(Jack) $\vee$ works_all_day(Jack)) $\rightarrow$ (tired(Jack) $\vee$ hungry(Jack))
   C. (plays_all_day(Jack) $\vee$ works_all_day(Jack)) $\wedge$ (tired(Jack) $\vee$ hungry(Jack))
   D. (plays_all_day(Jack) $\vee$ works_all_day(Jack)) $\rightarrow$ (tired(Jack) $\wedge$ hungry(Jack))
   - **Answer**: D. The corresponding proposition for option A is ”Jack plays all day and Jack works all day or Jack is tired or Jack is hungry”. The corresponding proposition for option B is ”If Jack plays all day or Jack works all day, then he will be tired or hungry”. The corresponding proposition for option C is ”Jack plays all day or Jack works all day and Jack is tired or Jack is hungry”.

2. **Select one:** Which of the following is the correct truth table for $\neg a \rightarrow \neg b$?

A.

a	b	$\neg a \rightarrow \neg b$
F	F	T
F	T	T
T	F	T
T	T	T

B.

a	b	$\neg a \rightarrow \neg b$
F	F	T
F	T	F
T	F	T
T	T	T

C.

a	b	$\neg a \rightarrow \neg b$
F	F	F
F	T	T
T	F	T
T	T	T

D.

a	b	$\neg a \rightarrow \neg b$
F	F	F
F	T	F
T	F	T
T	T	T

   - **Answer**: C. First observe that when a is T, $\neg a$ is F so the result $\neg a \rightarrow \neg b$ is vacuously true. Thus, we need to focus on the two cases when a is F. In this case $\neg a$ is T so we need $\neg b$ to be T, which occurs when b if F. If a is F and b is T, then the result $\neg a \rightarrow \neg b$ does not hold. The only option that aligns with this is C. Another way to arrive at this solution is observe that $\neg a \rightarrow \neg b$ is equivalent to $a \vee \neg b$; this also corresponds to the truth table in option C.

5 Inference rules

To work with statements and evidence, we use inference rules. The most famous rule is probably *modus ponens*: from premises ϕ and $\phi \to \psi$, conclude ψ. For example, from dog(Spot) and dog(Spot) $\to$ fuzzy(Spot), we would conclude fuzzy(Spot). We can translate modus ponens to natural language as: if I have evidence for ϕ, and if I have a way to translate evidence for ϕ into evidence for ψ, then I can get evidence for ψ.

All inference rules have this same general structure: given some premises, we can draw some conclusions. A premise is a statement that we already have evidence for; perhaps it was an assumption, or perhaps we constructed evidence for it in a previous step. A conclusion is a new statement that we are constructing evidence for.

Each inference rule gives us a template to follow. That is, the premises and the conclusions must fit given forms. For modus ponens, our premises have to look like ϕ and $\phi \to \psi$.

The symbols ϕ and ψ in the template can represent single propositions or more complicated expressions containing connectives. We took ϕ = dog(Spot) and ψ = fuzzy(Spot) above, but if we had a longer implication like

$$\text{dog(Spot)} \wedge \text{happy(Spot)} \to \text{wags(Spot)}$$

we'd want to take ϕ to be the compound expression dog(Spot) $\wedge$ happy(Spot).

Modus ponens is sometimes also called $\to$ elimination, since the premises contain an implication but the conclusion does not.

6 Proofs

If we chain several inference rules together, we get a *proof*. We will write a proof as a sequence of lines, where each line is either an assumption or an application of an inference rule. We can number or label the lines so that we can refer back to them. For example:

1. Assume dog(Spot).
2. Assume dog(Spot) $\to$ fuzzy(Spot).
3. Conclude fuzzy(Spot) by modus ponens from 1, 2.

Each line in the proof must have a justification! The only possible kinds of justification are the two above: either a line is an assumption, or it follows from some previous lines via an inference rule. Sometimes people omit justifications or skip steps when they are clear from context; but beware: "clear from context" is in the eye of the beholder.

If a proof is a bit longer than the one above, it can be worth annotating its structure, e.g., adding blank lines between sections, indenting parts or drawing boxes around them to indicate grouping, and including comments. We'll see examples of this sort of annotation below.

7 More Inference Rules

To get a complete logical system, we'll need some inference rules to work with each of our constants and connectives. There are multiple equivalent systems of inference rules for classical propositional logic, but for concreteness we'll describe a particular one here.

No matter which system we use, it's important to pick one and stick with it: we should always know which inference rules are allowed, and use only those. We can introduce subtle errors into our reasoning if we accidentally mix two different systems of rules.

We'll start with rules for introducing and eliminating $\vee$ and $\wedge$. For $\wedge$:

- $\wedge$ introduction: for any expressions ϕ and ψ, if we separately prove ϕ and ψ, that constitutes a proof of $\phi \wedge \psi$.
- $\wedge$ elimination: if we know $\phi \wedge \psi$, then we can conclude ϕ, and we can also conclude ψ.

For $\vee$:

- $\vee$ introduction: from ϕ we can conclude $\phi \vee \psi$ for any ψ. Similarly, we can also conclude $\psi \vee \phi$.
- $\vee$ elimination (also called proof by cases): if we know $\phi \vee \psi$, and if we have both $\phi \to \chi$ and $\psi \to \chi$, then we can conclude χ. Here ϕ and ψ are the cases; if we can prove a desired conclusion χ in both cases, then χ holds, even if we don't know which case we are in.

We have an introduction rule for T and an elimination rule for F. The introduction rule for T is perhaps the simplest of all of our rules:

- T introduction: from no premises, we can conclude T.

The elimination rule for F is a bit more counter-intuitive, but we'll see examples below of where we need it.

- F elimination: from the assumption F, we can conclude an arbitrary expression ϕ. This rule is sometimes called *ex falso* or *ex falso quodlibet*, meaning "from falsehood, anything" in Latin. One way to think about ex falso is this: we should never be able to prove F, so there's no danger in having F let us conclude an arbitrary formula.

You may notice that we haven't given any rules for $\neg$, and we haven't given an introduction rule for $\to$. We'll say more about how to handle $\neg$ and $\to$ below, but it requires a bit more setup and discussion first.

From the above rules we can derive a few other useful ones, like:

- Associativity: both $\vee$ and $\wedge$ are associative, meaning that it doesn't matter how we parenthesize an expression like $a \vee b \vee c \vee d$. This rule is so common that we often skip over it: we just leave out parentheses when they don't matter, as we did above.
- Distributivity: the connectives $\vee$ and $\wedge$ distribute over one another. For example, $a \wedge (b \vee c)$ is equivalent to $(a \wedge b) \vee (a \wedge c)$.
- Commutativity: both $\vee$ and $\wedge$ are symmetric in the order of their arguments. So for example $b \vee c \vee a$ is equivalent to $a \vee b \vee c$.
- Idempotence: repeated arguments of $\vee$ or $\wedge$ are superfluous. So for example, $a \vee b \vee b$ is equivalent to $a \vee b$.

These last few rules are a great example of where we might want to simplify a proof by grouping multiple steps and skipping explicit justifications: it would take forever if we listed out every application of associativity and commutativity. Nonetheless, we might ask you to be this detailed in some of our exercises, just for practice.

Knowledge check

1. Prove associativity for $\vee$. Be fully detailed and use only the introduction and elimination rules given above.
   - **Hint**: Start by assuming $(a \vee b) \vee c$ and try to show $a \vee (b \vee c)$.
   - **Answer**:
     (a) Assume $(a \vee b) \vee c$
     (b) Conclude either $(a \vee b)$ or c from (a) by $\vee$ elimination
     (c) Assume c
       i. Conclude $b \vee c$ from (c) by $\vee$ introduction
       ii. Conclude $a \vee (b \vee c)$ from i by $\vee$ introduction
     (d) Assume $(a \vee b)$
       i. Conclude either a or b from (d) by $\vee$ elimination
       ii. Assume a
         A. Conclude $a \vee (b \vee c)$ from ii by $\vee$ introduction
       iii. Assume b
         A. Conclude $b \vee c$ from iii by $\vee$ introduction
         B. Conclude $a \vee (b \vee c)$ from A by $\vee$ introduction
     (e) Conclude $[(a \vee b) \vee c] \to [a \vee (b \vee c)]$ from (c).ii and (d).iii.B