Week 8: Learning Theory

Instructor: Sergey Levine

1 Learning Theory Recap

Last week, we saw a generalized view of machine learning, where the learning algorithm is given in generic form as the following optimization:

$$\hat{f} \leftarrow \arg\min_f \frac{1}{N}\sum_{i=1}^{N} L(y^i, f(\mathbf{x}^i)).$$

Here, $L(y, f(\mathbf{x}))$ is a loss function, which can implement a wide range of different objectives. Although we minimize the loss on the training set $\mathcal{D}$, we really care about the loss on all the *other* data not part of the training set. This is the generalization error

$$\mathcal{E}_{\text{gen},\mathcal{D}} = E[L(y, \hat{f}(\mathbf{x}))|\mathcal{D}].$$

The dataset $\mathcal{D}$ fully determines the hypothesis $\hat{f}$ that we get once we choose the hypothesis space, so if we want to analyze how good a particular hypothesis space, we might want to consider generalization independently of the particular choice of $\mathcal{D}$, by averaging over all possible training sets of a given size:

$$\mathcal{E}_{\text{gen},N} = \int p(\mathcal{D}|N)E[L(y, \hat{f}(\mathbf{x}))|\mathcal{D}]d\mathcal{D} = E[L(y, \hat{f}(\mathbf{x}))|N].$$

We can also consider the generalization error for a specific single test point $\mathbf{x}$:

$$\mathcal{E}_{\text{gen},N}(\mathbf{x}) = \int p(\mathcal{D}|N)E[L(y, \hat{f}(\mathbf{x}))|\mathcal{D}]d\mathcal{D} = E[L(y, \hat{f}(\mathbf{x}))|N],$$

where the expectation $E[L(y, \hat{f}(\mathbf{x}))|\mathcal{D}]$ is taken with respect to the label y for the given $\mathbf{x}$. We might also consider the prediction we get for a new datapoint $\mathbf{x}$ averaged over all possible training sets:

$$\bar{f}(\mathbf{x}) = \int p(\mathcal{D}|N)\hat{f}(\mathbf{x})d\mathcal{D} = E[\hat{f}(\mathbf{x})|N].$$

In the previous lecture, we saw that we can express the loss for a new datapoint in terms of three terms

$$\mathcal{E}_{\text{gen},N}(\mathbf{x}) = E[L(g(\mathbf{x}), y)] + L(g(\mathbf{x}), \bar{f}(\mathbf{x})) + E[L(\bar{f}(\mathbf{x}), \hat{f}(\mathbf{x}))],$$

where $g(\mathbf{x})$ is the true underlying signal that is a deterministic function of $\mathbf{x}$, and y is the actual label (which might have added noise). The first term is an irreducible error due to the fundamentally noisy nature of the labels, the second term is the bias, which is the loss between the true underlying signal and the mean prediction, and the third term is the variance, which has nothing to do with the labels at all, and vanishes as $N \to \infty$ and all the datasets of size N start to look the same.

Finally, we covered how we could estimate bias and variance in practice by using bootstrap replicas to approximate the expectation over all possible training sets: the idea is simply to take a big training set $\mathcal{D}$ and sample new datasets from it with replacement. This is not a learning algorithm (though bootstrap can be used as part of other algorithms, such as bagging), but rather a way to analyze how good a particular hypothesis class is and directly measure its bias and variance with some real-world sample data.

2 No Free Lunch & Inductive Bias

Now that we understand how to evaluate an algorithm with respect to bias and variance, let's discuss another very important piece of learning theory called the no free lunch theorem.

So far in this course, we've covered a range of different machine learning methods. Some of these methods are necessary to handle different types of problems (e.g. regression vs classification), but others are just different ways of doing the same thing (SVM vs perceptron vs naïve Bayes). Why do we need so many different machine learning algorithms? Why can't we just pick the best one out there, and just use it all the time? There is actually a theoretic result in learning theory that gives us part of the answer, and it's called the no free lunch theorem. Understanding this theorem requires understanding inductive bias. Before I introduce the mathematical formalism, let me start off with a little joke.

An engineer, a physicist, and a mathematician are on a train in Scotland, and they see a black sheep. The engineer says "look: all sheep in Scotland are black." The physicist says "no, some sheep in Scotland are black." The mathematician is irritated with them and replies: "you have it all wrong: there is at least one sheep in Scotland, and it is black on at least one side."

Which one of them is right? Why? If you are a space alien that comes to Earth and has never seen a sheep, you might well decide the engineer is right... or the mathematician. Either seem like reasonable answers when you don't have the right *inductive bias*: that is, when you do not have anything to go on to extrapolate a reasonable hypothesis about the world. Hume described this idea more generally, when wrote: "even after the observation of the frequent or constant conjunction of objects, we have no reason to draw any inference concerning any object beyond those of which we have had experience."

That is, just because we saw something happen many times before, doesn't mean it will happen again. What does this have to do with machine learning and inductive bias? Well, the only reason machine learning works at all is because we apply some inductive bias to conclude that, once we've seen a particular pattern multiple times, we'll see that pattern again in unseen test data. For example, if we are performing regression, and we see training points arranged in a line, we believe that it is reasonable to extrapolate that they originate from a linear function. This is called induction, and it is precisely what Hume is challenging. Any induction implies an inductive bias: a tendency to believe that one kind of extrapolation is more correct than another. There is not a fundamental mathematical or statistical reason for this to be true, though of course certain inductive biases tend to be more true in practice because of how the physical world works.

Note that this problem with induction does not necessarily depend on the size of the dataset: we can construct pathologic functions (for example ones that are not continuous, such as the function that is 1 for all rational numbers and 0 for all irrational ones) that can confound us even with very large training sets. We might just assume, perhaps reasonably, that real-world problems are unlikely to look like this.

This idea can be formalized by the pessimistically named "No Free Lunch Theorem," which we will not prove, but we will describe the statement of the theorem and its implications for machine learning. In order to state the theorem, we need to introduce a little bit of notation:

$$\begin{aligned}
\mathcal{D} &\text{ – training set}\\
g(\mathbf{x}) &\text{ – the true function that gives rise to the data}\\
\hat{f}(\mathbf{x}) &\text{ – hypothesis predicted by our learning algorithm}\\
A(\mathcal{D}) &\text{ – learning algorithm (takes in } \mathcal{D} \text{ and produces } \hat{f}(\mathbf{x})\\
\mathcal{E}_{\text{gen},\mathcal{D}}(\hat{f}) &\text{ – error on all data not in the set } \mathcal{D}\text{, depends on hypothesis } \hat{f}
\end{aligned}$$

Unlike in our earlier discussion, we no longer need to assume that $A(\mathcal{D})$ actually minimizes some loss function (in fact, we don't even need to assume that A is deterministic). For the purpose of this discussion, "learning algorithm" is just something that outputs a hypothesis $\hat{f}(\mathbf{x})$, possibly by looking at the training set, but maybe even not (so simply guessing the answer at random counts as a learning algorithm). We do assume that the training set $\mathcal{D}$ is consistent with the true function $g(\mathbf{x})$ (meaning that $\mathcal{D}$ is not just garbage). We can also incorporate some noise or irreducible error, but that's not necessary right now.

Now, in the absence of any information besides $\mathcal{D}$, a reasonable assumption to make is that all true functions $g(\mathbf{x})$ that are consistent with $\mathcal{D}$ are equally likely. If this assumption holds, the no free lunch theorem tells us that for any two learning algorithms $A(\mathcal{D})$ and $A'(\mathcal{D})$, we have

$$E_{p(g)}[\mathcal{E}_{\text{gen},\mathcal{D}}(A(\mathcal{D}))] = E_{p(g)}[\mathcal{E}_{\text{gen},\mathcal{D}}(A'(\mathcal{D}))].$$

That is to say, no learning algorithm is better than any other given a training set and *no other knowledge about the problem.* In our expanded definition of "learning algorithm," this also includes random guessing, always outputting 0, etc. The formal proof of this is a bit involved, but the intuition follows directly from Hume's logic: simply because we saw a trend in our training set $\mathcal{D}$ doesn't mean that this trend will actually hold, unless our knowledge about the structure of the world suggests that it should (and in the general case, we have no such knowledge).

While this result may at first seem unintuitive, it may be easier to understand with an example. Let's go back to the joke about Scottish sheep: if we see one sheep, and expect to then see a second, we could devise a family of learning algorithms: for the second sheep, we could either guess the sheep to always be black, always white, always the same color as the first, or always a different color than the first. If we assume that all true functions are equally likely, then the we are equally likely to see each of the following combinations of first and second sheep: (black, black), (white, white), (black, white), (white, black). Indeed, each of our learning rules in this case has the same expected error: $\frac{1}{2}$.

If we have a little bit more knowledge about the world, we might conclude that animals tend to be the same color, so the worlds where we see (black, white) or (white, black) are unlikely. In that case, guessing the same color as the first is much more likely to be right! So if we have additional knowledge, we can pick a better learning algorithm.

Question. Which of the four learning algorithms in this case would be better in the real world?

Answer. The one that always picks white, because most sheep are white. This illustrates part of the implication of the no free lunch theorem: the rest of the world and the context of the problem matters, and a seemingly very reasonable algorithm (guessing "same") is actually not very good in practice, compared to a much dumber algorithm that completely ignores the data.

We might think that the real world doesn't behave like this, and that certain structural regularities always hold. But in fact, even problems with seemingly simple structure can acquire very complex structure if we frame them the wrong way, leading to a situation where seemingly very good and general learning algorithms perform very poorly. For example, imagine we have data sampled from the function $y = x$. This is an ideal candidate for using linear regression, and we can learn the function perfectly with just a single weight on the input x (or a weight and a bias) and two datapoints. However, if we are instead given a vector $\bar{\mathbf{x}}$, where each entry in $\bar{\mathbf{x}}$ is one bit in the binary encoding of the real-valued x, now linear regression will be disastrous: we would do very poorly at predicting values of x that are much larger than ones we saw, once bits in $\bar{\mathbf{x}}$ that were always 0 during training become 1. The lesson here is that even seemingly "universal" regularities like continuity don't necessarily hold, especially if we are not careful about which representation we use.

What does all this mean for real machine learning problems? One lesson we might draw for this is that induction is hopeless and we can never learn anything, so we should just always guess that the sheep will be white. But of course this is ridiculous: machine learning does work (and so does induction), and insofar as something works, it makes sense to use it. Rather, the lesson we should take away is that there is no one perfect algorithm, no one perfect prior, etc., but that we should consider the different options at hand for tackling a new machine learning problem, experiment with different approaches and choose the best one. If we have knowledge about the problem setting (and we usually do), we should use that knowledge to select the right representation and method, and we should think about the regularities that make sense in our problem setting, and the ones that don't make sense. This can become particularly important if we want to tackle weird, esoteric applications of machine learning in domains that we are less familiar with.

There are a few examples of lessons to draw:

Cross-validation: Cross-validation is often a good way to choose a good algorithm, representation, or model, and can help us find the right inductive bias. However, we should also keep in mind that cross-validation can be viewed as just another ("meta") learning algorithm, which takes in a training set, partitions it, tries a few models, and picks the best one on the validation error. It can still be expressed as a function of the form $A(\mathcal{D}) = \hat{f}$, just a fancier one.

Representation vs. algorithm: The no free lunch theorem makes no distinction between algorithm, regularization, or model. As we saw in the linear regression case earlier, sometimes it's not the method that makes the difference, but the representation. The same applies to regularization: sometimes it's just a matter of putting in the right prior. Oftentimes, most any algorithm can be adapted successfully to most any problem with the right choice of representation and regularization. That is part of the reason for the success of neural network models in current machine learning: they can be flexibly adapted, adjusted, regularized, and tuned to various domains.

Domain knowledge: Domain knowledge can be hugely important for identifying the representation and model that fits a particular application domain, though of course we also want to be careful not to introduce *too much* bias in this way.