Intro to Learning Theory

1 Machine Learning and Learning Theory

Coming soon..

2 Formal Framework

2.1 Basic notions

In our formal model for machine learning, the instances to be classified are members of a set $\mathcal{X}$, the *domain set* or *feature space*. Instances are to be classified into a *label set* $\mathcal{Y}$. For now (and most of the class), we assume that the label set is binary, that is $\mathcal{Y} = \{0, 1\}$. For example, an instance $x \in \mathcal{X}$ could be an email and its label indicates whether the email is spam ($y = 1$) or not spam ($y = 0$). We often assume that the instances are represented as real-valued vectors, that is $\mathcal{X} \subseteq \mathbb{R}^d$ for some dimension d.

A *predictor* or *classifier* is a function $h : \mathcal{X} \to \mathcal{Y}$. A *learner* is a function that takes some training data and maps it to a predictor. We let the *training data* be denoted by a sequence $S = ((X_1, Y_1), \ldots, (X_n, Y_n))$. Then, formally, a learner $\mathcal{A}$ is a function

$$\begin{aligned} \mathcal{A} &: \bigcup_{i=1}^{\infty} (\mathcal{X} \times \mathcal{Y})^i \to \mathcal{Y}^{\mathcal{X}} \\ \mathcal{A} &: S \mapsto h, \end{aligned}$$

where $\mathcal{Y}^{\mathcal{X}}$ denotes the set of all functions from set $\mathcal{X}$ to set $\mathcal{Y}$. For convenience, when the learner is clear from context, we use the notation h_n to denote the output of the learner on data of size n, that is $h_n = \mathcal{A}(S)$ for $|S| = n$.

The goal of learning is produce a predictor h that correctly classifies not only the training data, but also future instances that it has not seen yet. We thus need a mathematical description of how the environment produces instances. In particular, we would like to model that the environment (or nature) remains somehow stable, that the process that generated the training data is the same that will generate future data.

We model the data generation as a probability distribution P over $\mathcal{X} \times \mathcal{Y} = \mathcal{X} \times \{0, 1\}$. We further assume that the instances (X_i, Y_i) are *i.i.d.* (independently and identically distributed) according to P.

In summary, a learner A uses the training data S to learn a predictor function, which can then be used to make predictions on new inputs. The learner A encapsulates the learning algorithm, while the predictor h is the model obtained after training.

The performance of a classifier h on an instance (X, Y) is measured by a *loss function.* A loss function is a function

$$\ell \ : \ (\mathcal{Y}^{\mathcal{X}} \times \mathcal{X} \times \mathcal{Y}) \ \to \ \mathbb{R}.$$

The value $\ell(h, X, Y) \in \mathbb{R}$ indicates "how badly h predicts on example (X, Y)". We will, for now, work with the *binary loss* (or *0/1-loss*), defined as

$$\ell(h, X, Y) = \mathbf{1}[h(X) \neq Y]\,,$$

where $\mathbf{1}[p]$ denotes the *indicator function* of predicate p, that is $\mathbf{1}[p] = 1$ of p is true and $\mathbf{1}[p] = 0$ if p is false. The binary loss is 1 if the prediction of h on example (X, Y) is wrong. If the prediction is correct, no loss is suffered and the binary loss assigns value 0.

We can now formally phrase the goal of learning, as aiming for a classifier that has low loss on expectation over the data generating distribution. That is, we would like to output a classifier that has low *expected loss*, or *risk*, defined as

$$L(h) \ = \ \mathbb{E}_{(X,Y)\sim P}[\ell(h, X, Y)] \ = \ \mathbb{E}_{(X,Y)\sim P}[\mathbf{1}[h(X) \neq Y]].$$

Since our loss function assumes only values in $\{0, 1\}$, the above expectation is equal to the probability of generating an example X on which h makes a wrong prediction. That is, we have

$$L(h) \ = \ \mathbb{E}_{(X,Y)\sim P}\mathbb{E}_{(X,Y)\sim P}[\mathbf{1}[h(X) \neq Y]] \ = \ \mathbb{P}_{(X,Y)\sim P}[h(X) \neq Y].$$

Note however, that the learner does not get to see the data generating distribution. It can thus not merely output a classifier of lowest expected loss. The learner needs to make its decisions based on the data S. Given a classifier h and data S, the learner can evaluate the *empirical risk* of h on S

$$L_n(h) \ = \ \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[h(X_i) \neq Y_i]\,.$$

2.2 On the relation of empirical and true risk

A natural strategy for the learner would be, to simply output a function that has small empirical risk. In favor of this approach, we now show that the empirical risk is an unbiased estimator of the true risk.

Claim 1. *For all functions $h : \mathcal{X} \to \{0, 1\}$ and for all sample sizes n we have*

$$\mathbb{E}_S L_n(h) = L(h)$$

Proof.

$$\begin{aligned}
\mathbb{E}_S L_n(h) &= \mathbb{E}_S \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[h(X_i) \neq Y_i] \\
&= \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_S \mathbf{1}[h(X_i) \neq Y_i] \\
&= \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{(X,Y)} \mathbf{1}[h(X) \neq Y] \\
&= \frac{1}{n}\sum_{i=1}^{n} L(h) \\
&= L(h)
\end{aligned}$$

where the second equality holds by linearity of expectation, and the third inequality holds since that expectation depends only on one (the i-th) example in S. □

Thus, for any fixed function, the empirical risk gives us an unbiased estimate of the quantity that we are after, the true risk. Note that this holds even for small sample sizes. Moreover, by the *law of large numbers*, the above claim implies that, with large sample sizes, the empirical risk of a classifier converges to its true risk (in probability). As we see more and more data, the empirical risk of a function becomes a better and better estimate of its true risk.

This may lead us to believe that the simple learning strategy of just finding some function with low empirical risk should succeed at achieving low true risk as we see more and more data. However, the following phenomenon shows that this strategy can in fact go wrong arbitrarily badly.

Claim 2. *There exists a distribution P and a learner, such that for all n we have*

$$L_n(h_n) = 0 \text{ and } L(h_n) = 1$$

Proof. As the data generating distribution, consider the uniform distribution over $\mathbb{R}\times\{1\}$. That is, in any sample S, generated by this P, the examples are labeled with 1, that is $S = ((X_1, 1)), \ldots, (X_n, 1)))$. We construct a "stubborn" learner $\mathcal{A}$. The stubborn learner outputs a function that agrees with the sample's labels on points that were in the sample S, but keeps believing that the label is 0 everywhere else. Formally:

$$h_n(X) = \mathcal{A}(S)(X) = \begin{cases} 1 \text{ if } (X,1) \in S \\ 0 \text{ otherwise} \end{cases}$$

Now we clearly have $L_n(h_n) = 0$ for all n. However, since S is finite, the set of instances X on which h_n predicts 1 has measure 0. Thus, with probability 1, h_n outputs the incorrect label 0. Thus $L(h) = 1$. □

The difference between the situations in the above two claims is that, in the second case, the function h_n *depends on the data.* While, *for every fixed function* h (fixed before the data is seen), the empirical risk estimates converge to the true risk of this function, this convergence is not uniform over all functions. Claim 2 shows that, at any given sample size, *there exist* functions, for which true and empirical risk are arbitrarily far apart.

Now, in machine learning, we do want the function that the learner outputs to be able to depend on the data. Furthermore, the learner only ever gets to see a finite amount of data. We have seen that, for any finite sample size, that is, on any finite amount of data, the empirical risk can be a very bad indicator of the true risk of a function.

Basic questions of learning theory thus are: How can we control the (true) risk of a function learned based on a finite amount of data? Can we identify situations where we can relate the true and empirical risk?