13-5-2022

"Understanding Machine Learning: From Theory to Algorithms"

Chapters 1 (A gentle start) and 2 (A formal learning model)

ALEJANDRO CARMELO PARRA GARCIA

Contents

Chapter 1: A gentle start

The basic knowledge that we need to get is a description of the framework and the different parts that are involved in machine learning.

The first part is the input that a model requires, or learner's input. That is, what information does the learner have access to.

That is the **training data**, denoted as *S*, in a sequence of pairs *X x Y*. X and Y refer to Domain and Label set respectively. **Domain set *X***, is a set of objects, for each object there can be multiple attributes, those attributes are descriptions of certain properties of the objects that the model uses to train (For example: sepal length, sepal width, petal length, petal width as descriptions of different plants). And the **Label Set *Y*,** is a set of all the labels for now let's say a binary label, {0, 1} (Each label can refer to a different type of plant, Iris Versicolor, Iris Setosa), later we will expand this domain to multiple ones.

The second part is the output of the model, or the learners output. That is what information does the model expels. For that the model needs a **prediction rule**, *h: X → Y*, this prediction rule or predictor is used to predict the label for a new domain point. Continuing with the previous example, it is the rule used to predict the type of flower base on the attributes of a new domain point (sepal length, sepal width, petal length, petal width).

In order to obtain our Training set, we need to first get *X*, for that we need to make a random sample from a distribution *D*, we don't know what type of distribution *D* is. Once we have X, we need the Label Set, for that we use the real correct function *f: X → Y*, and use it to calculate every *Y* base on *X*:

$$y_i = f(x_i)$$

Now we have the complete training data *X x Y*. That we can use to train our model, the objective of the model is to create a prediction rule *h* that is as close as possible to the real rule *f.* We can measure how close the predictor is to the real rule by measuring the error of the classifier.

We can denote this error or Loss as *L*, and L is:

$$L_{D,f}(h) \overset{def}{=} \mathbb{P}_{X\sim D}[h(x) \neq f(x)] \overset{def}{=} D(\{x: h(x) \neq f(x)\})$$

The **Error or Loss** is the probability of randomly choosing an example x for which h(x) ≠ f(x), that is the prediction is different from the real label.

What is Empirical Risk Minimization?

Since the model or learner does not know the underlying distribution *D* or the correct target function *f*, the true error is impossible to calculate. So, we can use the **Empirical Error**, since we want to minimize the error, our model is trying to create a predictor h, that minimize L_s (Loss for the training data *S*) this is call **Empirical Risk Minimization** (ERM).

$$L_s(h) = \frac{|\{i \in [m]: h(x_i) \neq y_i\}|}{m}, [m] = \{1, \dots, m\}$$

We need to be careful with **Overfitting**, that is when the predictor fits the training data *S* so well that it fails to capture the true reality of the underlying distribution *D*. The ERM method will be able to get a really good predictor for the training data, but not necessarily for the distribution D. And since we want to comprehend the true reality of *D*, we can fail in the process if we stick to this method.

That is why it is better to use **Empirical Risk Minimization with Inductive Bias**.

The first step is to create a hypothesis class *H*, form by each $h \in H$. This set of prediction rules is set in advance. Now the objective is to find the best predictor in *H* using ERM, with the lowest error on the sample data *S*.

$$ERM_H(S) \in \underset{h \in H}{\operatorname{argmin}} L_S(h)$$

The restriction of choosing only from H, is called inductive bias, these restrictions are made in advance using some domain knowledge about the field.

But know we encounter the question of how to choose this hypothesis class *H*. One way it is to set an upper limit on the size of *H*, that will make it a finite class of predictors.

Let say h_s is the result of applying ERM_H to a sample data *S*, being *H* a finite class

$$h_S \in \underset{h \in H}{\operatorname{argmin}} L_S(h)$$

The realizability assumption states that there exist $h^* \in H$ such that the true loss of this predictor over the original distribution *D* label with the correct function *f* is 0.

$$L_{(D,f)}(h^*) = 0$$

This implies that with a probability 1 over random samples *S*, if *S* is sampled according to *D* and label with the correct function f, then:

$$L_S(h^*) = 0$$

It is important to state that the training data *S* (of size m) is sample accordingly to the underlying distribution *D*, and label according to the function *f*. This is called $S \sim D^m$. This is important as the model or learner has only access to *S*, and it is trying to infer knowledge of *D*, so if *S* is non representative of *D* the learner will fail in its task. The larger the size of the training data *S* the closer is going to reflect the distribution *D*.

But *S* is chosen randomly from *D*, so there is a non-zero probability that *S* is not representative of the distribution *D*, in the example of the flowers there is a chance that all flowers picked for a training set *S* are all Iris Versicolor and none are Iris Setosa. The probability of getting a nonrepresentative *S* is called δ, and the **confidence parameter** is $(1- \delta)$. Another important vale to take into consideration is the **accuracy parameter** ε.

So, if the loss function over the distribution D of the h_S is greater than the accuracy parameter ε then there is failure. But if it is smaller, it is considered a correct predictor:

$$\mathrm{L}_{(D,f)}(h_S) \leq \varepsilon$$

If we set H_B the set of bad hypotheses:

$$H_B = \{h \in H: \mathrm{L}_{(D,f)}(h_S) > \varepsilon\}$$

And the misleading samples as M:

$$M = \{S|_x: \exists h \in H_B, \mathrm{L}_s(h) = 0\}$$

This are the samples that are correct on the training data *S* but fail on *D*. Our objective is to limit the probability of $\mathrm{L}_{(D,f)}(h_S) > \varepsilon$. This will only happen in the misleading samples *M*.

$$\{S|_x: \mathrm{L}_{(D,f)}(h_S) > \varepsilon\} \leq M$$

By applying some more rules and the union bound we get to the next equation (See appendix for the details):

$$D^m\{S|_x: \mathrm{L}_{(D,f)}(h_S) > \varepsilon\} \leq |H_B|e^{-\varepsilon m} \leq |H|e^{-\varepsilon m}$$

This is a visual interpretation:

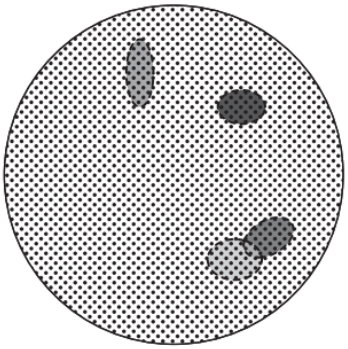

Figure 2.1. Each point in the large circle represents a possible m-tuple of instances. Each colored oval represents the set of "misleading" m-tuple of instances for some "bad" predictor $h \in \mathcal{H}_B$. The ERM can potentially overfit whenever it gets a misleading training set S. That is, for some $h \in \mathcal{H}_B$ we have $L_S(h) = 0$. Equation (2.9) guarantees that for each individual bad hypothesis, $h \in \mathcal{H}_B$, at most $(1-\epsilon)^m$-fraction of the training sets would be misleading. In particular, the larger m is, the smaller each of these colored ovals becomes. The union bound formalizes the fact that the area representing the training sets that are misleading with respect to some $h \in \mathcal{H}_B$ (that is, the training sets in M) is at most the sum of the areas of the colored ovals. Therefore, it is bounded by $|\mathcal{H}_B|$ times the maximum size of a colored oval. Any sample S outside the colored ovals cannot cause the ERM rule to overfit.

With a sufficiently large m, ERM_H rule over a finite H will be **Probably** (with confidence 1−δ) **Approximately** (up to an error of ε) **Correct** (PAC)

Chapter 2: A formal learning model

This chapter is dedicated to the **PAC learning model** introduced before.

What make a hypothesis class *H* **PAC learnability**?

Definition: *"A hypothesis class H is PAC learnable if there exists a function* $m_H: (0,1)^2 \to \mathbb{N}$ *and a learning algorithm with the following property: For every* $(\mathcal{E}, \delta) \in (0,1)$*, for every distribution D over X, and for every labeling function* $F: x \to \{0,1\}$*, If the realizable assumption holds with respect to H,D, f, then when running the algorithm on* $m \geq m_H(\mathcal{E}, \delta)$ *independently and identically distributed examples generated by D and label by f, the algorithm returns a hypothesis h such that, with probability of at least 1 – δ (over the choice of the examples),* $L_{(D,f)}(h) \leq \mathcal{E}$*."*

PAC learnability have two parameters:

1) The accuracy *Ɛ*, measure the difference between the output of the model and the true correct solution.
2) The confidence *δ*, this parameter indicates the likelihood of the model achieving the accuracy measure.

These parameters are needed because the model works on the training set *S*, and as we have seen before there is a chance that this set *S* is nonrepresentative of the underlying distribution *D*, and even if it is, since *S* is a finite set it cannot reflect all the details of the distribution *D*.

Previously we have mentioned a function $m_H: (0,1)^2 \to \mathbb{N}$, but what does this function mean. It determines the **sample complexity** of learning *H*. This function depends on the accuracy *Ɛ* and confidence *δ* as well as in the properties of the class H. It determines the number of examples needed to get a PAC solution.

Many different functions m_H work on the definition of PAC learnability if H is PAC learnable. So, to disambiguate this, the sample complexity refers to the "minimal function".

Every finite Class H is PAC learnable with sample complexity:

$$m_H(\mathcal{E}, \delta) \leq \left\lceil \frac{\log(|H|/\delta)}{\mathcal{E}} \right\rceil$$

Now it's time to generalize the mode, for that, two routes will be taken, first by removing the realizability assumption and second by expanding the scope of the learning models.

1) Releasing the Realizability Assumption – Agnostic PAC Learning

So far, we have created the training set by using a distribution D, from which we extract a set of features for some datapoints we have denoted this as *X*, and then used a labeling function *f(x)* to create the data *Y*.

So now we do not use this labeling function f, instead we obtain *X x Y* from D, as now D is a joint distribution of data points and labels. This distribution is composed of two parts, a

marginal distribution and a conditional probability. The marginal distribution is D_x over unlabeled domain points, and the conditional probability are the labels for each point, $D\big((x,y)\big|x\big)$. Using the examples of the flowers, the marginal distribution determines the probability of encountering a flower whose sepal length, sepal width, petal length, petal widths fall in some domain of those variables, and the conditional probability is the probability that that flower with those attributes x, is an Iris Versicolor or Iris Setosa.

But now we have a problem, we have eliminated the use of the function *f* to label the data, so now the true error or risk cannot work. If we recall from the previous chapter, the definition of this error was: $L_{D,f}(h) \stackrel{def}{=} P_{X\sim D}[h(x) \neq f(x)] \stackrel{def}{=} D(\{x: h(x) \neq f(x)\})$, and since it uses the function f, we cannot longer use it, or we need to define it again without using this function.

$$L_D(h) \stackrel{def}{=} \mathop{\mathbb{P}}_{(x,y)\sim D}[h(x) \neq y] \stackrel{def}{=} D(\{(x,y): h(x) \neq y\})$$

Now both *X* and *Y* come from the underlying distribution *D*, and there is no correct labeling function *f*. But it is important to say that the model or learner does not have access to the underlying distribution *D* but to the training set S. So, we need to look at the empirical risk:

$$L_s(h) = \frac{|\{i \in [m]: h(x_i) \neq y_i\}|}{m}, [m] = \{1, \dots, m\}$$

Here the formula for the empirical risk is the same as before, since it didn't depend on the function *f*, so we do not need to change this formula.

As before the objective is to find a predictor h, that minimizes the true risk $L_D(h)$.

The **Bayes optimal predictor**, given a distribution D over a training set S, the best label function from X to Y, given that Y is a binomial set {0,1} is:

$$f_D(x) = \begin{cases} 1 \; if \; P[y = 1|x] \geq 1/2 \\ 0 \; otherwise \end{cases}$$

The predictor f_D is optimal, since no other predictor g:X→{0,1}, has a lower error, $L_D(f_D) \leq L_D(g)$. The predictor f_D predicts 1 if the probability of the label being 1 given the domain point x is grater that one half and predicts 0 in any other cases.

The problem with this optimal predictor is that it works on the underlying distribution *D*, but the model or learner only have access to the training set *S*, which mean that we cannot use it.

Now we can introduce the **agnostic PAC learnability**.

Definition: *"A hypothesis class H is agnostic PAC learnable if there exists a function* $m_H: (0,1)^2 \to \mathbb{N}$ *and a learning algorithm with the following property: For every* $(\mathfrak{E}, \delta) \in (0,1)$*, for every distribution D over X x Y, when running the algorithm on* $m \geq m_H(\mathfrak{E}, \delta)$ *independently and identically distributed examples generated by D, the algorithm returns a hypothesis h such that, with probability of at least 1 − δ (over the choice of the m training examples),* $L_D(h) \leq \min_{h' \in H} L_D(h') + \mathfrak{E}$*."*

We can easily see that this is a generalize case of the normal PAC learnability, since if the realizability assumption holds then there exist an $h^* \in H$ for which $L_D(h^*) = 0$, this means that the $\min_{h' \in H} L_D(h') = 0$ and in turns change the equation from the definition above into

$L_D(h) \leq 0 + \mathcal{E}$, which is the same as the definition of the normal PAC learnability. Thus, we can see that this is a more general model.

2) The Scope of Learning Problems Modeled

So far, we have been talking about binomial cases, the flower was Iris Versicolor or Iris Setosa. But we can expand the model to multiple other types, for example Multiple class classification or Regression. The **multiple class classification** case is really simple as we don't need to modify much from the previous cases, here the main difference is in the label set *Y*, when previously it was a set of only two elements, here it is a large finite set. For the **regression** case things are a little more different, that is because in this case we want to find the relationship between *X* and *Y* (Example: Salary (*Y*) based on *X*: years of education, years of experience, Gender, Ethnicity). In this case we need to change the Loss function needs to be different, so we use the *expected square difference*:

$$L_D(h) \overset{def}{=} \underset{(x,y)\sim D}{\mathbb{E}}(h(x) - y)^2$$

Now we can introduce the concept of **Generalized Loss Functions**, there are other task beyond supervised learning, for example unsupervised learning, in this case there are no Y data or labels, and in turn we cannot use the previously used loss functions. That is why we need to change a couple of things. Now we will talk about domain *Z*, in our previous cases $Z = X\ x\ Y$, but now we can used it for other cases, Z follows the underlying distribution D. We also need a function $l: H\ x\ Z \to \mathbb{R}_+$ from the domain Z and the set of predictors H to the positive real numbers. Now we can construct the Risk function:

$$L_D(h) \overset{def}{=} \underset{Z\sim D}{\mathbb{E}}[l(h, z)]$$

We also need to rewrite the empirical risk functions with the new constrains. So, over a sample $S = (z_1, \dots, z_m) \in Z^m$ we get the next formula:

$$L_S(h) \overset{def}{=} \frac{1}{m}\sum_{i=1}^{m} l(h, z_i)$$

Now we need to define the function l, for the two cases we have been talking about, the binary or multiclass classification and the regression.

For the binary or multiclass classification, let's call it l_{0-1}, this function is dependent on the predictor function *h* and on the domain *Z*, it this case this domain is $X\ x\ Y$:

$$l_{0-1}\big(h, (x, y)\big) \overset{def}{=} \begin{cases} 0\ if\ h(x) = y \\ 1\ if\ h(x) \neq y \end{cases}$$

For the regression case let's call this function l_{sq}(as square loss), this case is similar as before in the sense that it is dependent on the predictor function h, and on the domain $X\ x\ Y$:

$$l_{sq}\big(h, (x, y)\big) \overset{def}{=} (h(x) - y)^2$$

Since we have added new elements to the formulas and made a generalization, we need to define the **Agnostic PAC Learnability for General Loss Functions.**

Definition: *"A hypothesis class H is agnostic PAC learnable with respect to a set Z and a loss function* $l: H\ x\ Z \to \mathbb{R}_+$*, if there exist a function* $m_H: (0,1)^2 \to \mathbb{N}$ *and a learning algorithm with*

the following property: For every $(\mathfrak{E}, \delta) \in (0,1)$*, and for every distribution D over Z, when running the learning algorithm on* $m \geq m_H(\mathfrak{E}, \delta)$ *independently and identically distributed examples generated by D, the algorithm returns* $h \in H$ *such that, with probability of at least 1 − δ (over the choice of the m training examples),* $L_D(h) \leq \min_{h' \in H} L_D(h') + \mathfrak{E}$*, where* $L_D(h) = \mathbb{E}_{Z \sim D}[l(h, z)]$*"*

We can see that this definition is really similar to the previous one, the main difference is the addition of the function l, and the domain Z.

Exercises

From "Chapter 1: A gentle start" ex 2.2.

Let H be a class of binary classifiers over a domain X. Let D be an unknown distribution over X, and let f be the target hypothesis in H. Fix some h ∈ H. Show that the expected value of $L_S(h)$ over the choice of $S|_x$ equals $L_{(D,f)}(h)$, namely

$$\mathop{\mathbb{E}}_{S|_x \sim D^m}[L_S(h)] = L_{(D,f)}(h)$$

We know:

$$L_s(h) = \frac{|\{i \in [m]: h(x_i) \neq y_i\}|}{m}, [m] = \{1, \dots, m\}$$

$$L_{D,f}(h) = P_{X \sim D}[h(x) \neq f(x)]$$

Now, based on these formulas we get:

$$\mathop{\mathbb{E}}_{S|_x \sim D^m}[L_S(h)] = \mathop{\mathbb{E}}_{S|_x \sim D^m}\left[\sum_{i=1}^{m} \frac{\begin{cases}1, if\ h(x_i) \neq f(x_i)\\ 0, otherwise\end{cases}}{m}\right] =$$

$$= \frac{1}{m} \mathop{\mathbb{E}}_{S|_x \sim D^m}\left[\sum_{i=1}^{m} \begin{cases}1, if\ h(x_i) \neq f(x_i)\\ 0, otherwise\end{cases}\right] = \frac{1}{m}\sum_{i=1}^{m} \mathop{\mathbb{E}}_{x_i \sim D}\left[\begin{cases}1, if\ h(x_i) \neq f(x_i)\\ 0, otherwise\end{cases}\right] =$$

$$= \frac{1}{m}\sum_{i=1}^{m} \mathop{\mathbb{P}}_{x \sim D}[h(x_i) \neq f(x_i)] = \frac{1}{m}\sum_{i=1}^{m} L_{(D,f)}(h) = \frac{1}{m} m * L_{(D,f)}(h)$$

$$\mathop{\mathbb{E}}_{S|_x \sim D^m}[L_S(h)] = \frac{1}{m} m * L_{(D,f)}(h) = L_{(D,f)}(h)$$

$$\mathop{\mathbb{E}}_{\boldsymbol{S|_x \sim D^m}}[\boldsymbol{L_S(h)}] = \boldsymbol{L_{(D,f)}(h)}$$

From "Chapter 2: A formal learning model" ex 3.6.

Let *H* be a hypothesis class of binary classifiers. Show that if *H* is agnostic PAC learnable, then *H* is PAC learnable as well. Furthermore, if *A* is a successful agnostic PAC learner for *H*, then *A* is also a successful PAC learner for *H*.

If H is agnostic PAC learnable then there exists a function $m_H: (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: For every $(\varepsilon, \delta) \in (0,1)$, for every distribution D over X x Y, when running the algorithm on $m \geq m_H(\varepsilon, \delta)$, $S \overset{iid}{\sim} D$, the algorithm returns a hypothesis h such that:

$$\mathbb{P}\left(L_D(h) \leq \min_{h' \in H} L_D(h') + \varepsilon\right) \leq 1 - \delta$$

Now we want to see if H is PAC learnable, for that if we look at the definition of PAC learnability the Realizability Assumption, needs to hold. This assumption states there exist $h^* \in H$ such that $L_{(D,f)}(h^*) = 0$. Now if we look at the formula for the agnostic PAC learnability, we can see that if

$$h^* \in H: L_{(D,f)}(h^*) = 0$$

then

$$\min_{h' \in H} L_D(h') = 0$$

This will make the initial equation like this:

$$\mathbb{P}\left(L_D(h) \leq \min_{h' \in H} L_D(h') + \varepsilon\right) \leq 1 - \delta <==>$$

$$<==> \mathbb{P}\left(L_{(D,f)}(h) \leq 0 + \varepsilon\right) \leq 1 - \delta <==>$$

$$<==> \mathbb{P}\left(L_{(D,f)}(h) \leq \varepsilon\right) \leq 1 - \delta$$

We can see that H is PAC learnable, as well as agnostic PAC learnable.

If A is a successful agnostic PAC learner for H, then there exists an algorithm A which produces a hypothesis in H with small generalization error. As before we can see that if the Realizability Assumption holds then it can be a successful PAC learner for H.

Appendix

$$\{S|_x: \mathrm{L}_{(D,f)}(h_S) > \varepsilon\} \leq M$$

M is:

$$M = \bigcup_{h \in H_B} \{S|_x: \mathrm{L}_S(h) = 0\}$$

$$D^m(\{S|_x: \mathrm{L}_{(D,f)}(h_S) > \varepsilon\}) \leq D^m(M)$$

$$D^m(\{S|_x: \mathrm{L}_{(D,f)}(h_S) > \varepsilon\}) \leq D^m\left(\bigcup_{h \in H_B} \{S|_x: \mathrm{L}_S(h) = 0\}\right) \xrightarrow{Union\ Bound}$$

$$=> D^m(\{S|_x: \mathrm{L}_{(D,f)}(h_S) > \varepsilon\}) \leq \sum_{h \in H_B} D^m(S|_x: \mathrm{L}_S(h) = 0)$$

We can rewrite $D^m(S|_x: \mathrm{L}_S(h) = 0)$ as:

$$D^m(\{S|_x: \mathrm{L}_s(h) = 0\}) = D^m(\{S|_x: \forall i, h(x_i) = f(x_i)\}) = \prod_{i=1}^{m} D(\{x_i: h(x_i) = f(x_i)\})$$

For each element of *S,* we have:

$$D(\{x_i: h(x_i) = f(x_i)\}) = 1 - \mathrm{L}_{(D,f)}(h) \leq 1 - \varepsilon$$

Combining the previous two equations we get:

$$D^m(S|_x: \mathrm{L}_S(h) = 0) \leq (1 - \varepsilon)^m \leq e^{-\varepsilon m}$$

Using this equation and $D^m(\{S|_x: \mathrm{L}_{(D,f)}(h_S) > \varepsilon\}) \leq \sum_{h \in H_B} D^m(S|_x: \mathrm{L}_S(h) = 0)$ we get:

$$D^m(\{S|_x: \mathrm{L}_{(D,f)}(h_S) > \varepsilon\}) \leq |H_B| e^{-\varepsilon m} \leq |H| e^{-\varepsilon m}$$