Classification

prof. dr Arno Siebes

Algorithmic Data Analysis Group
Department of Information and Computing Sciences
Universiteit Utrecht

Setting the Stage

Learning Revisited

In the introduction we already discussed that the ultimate goal is to

learn $\mathcal{D}$ from D

Moreover, we noted that for this course we are mostly interested in learning

a marginal distribution of $\mathcal{D}$ from D

More in particular, let

$$D = X \times Y \sim \mathcal{D} = \mathcal{D}|_X \times \mathcal{D}|_{Y|X} = \mathcal{X} \times \mathcal{Y}$$

Then the marginal distribution we are mostly interested in is:

$$\mathbb{P}(Y = y \mid X = x)$$

where $\mathcal{Y} = \mathcal{D}|_{Y|X}$ (and thus Y) is a finite set

Classification

The rewrite of $\mathcal{D}$ to $\mathcal{X} \times \mathcal{Y}$ was on purpose

- $\mathcal{X}$ are variables that are easy to observe or known beforehand
- $\mathcal{Y}$ are variables that are hard(er) to observe or only known later

In such a case, one would like to

- predict $\mathcal{Y}$ from $\mathcal{X}$
- that is, given an $X \sim \mathcal{X}$ with an observed value of x
  1. give the marginal distribution $\mathbb{P}(Y = y \mid X = x)$
  2. or give the most likely y given that $X = x$
  3. or any other prediction of the Y value given that $X = x$

Given that (Y) is finite, this type of prediction is commonly known as classification. Bayesians prefer (1), while most others prefer (2).

While I'm a Bayesian as far as my statistical beliefs are concerned

- it is the only coherent, thus rational, approach to statistical inference

we will focus, almost always, on (2)

Classification, continued

Answering the question which y is the most probable is easy if you know the marginal distribution $\mathbb{P}(Y = y \mid X = x)$

- simply select that y that has maximal probability
- this is the Bayes optimal solution

If that is the only thing we want,

- learning the complete (marginal) distribution seems overkill

After all,

- the exact probability values are unimportant
- only the ranking the highest one right matters

For that reason, classification is often studied as the problem of

- learning a (computable) function $h : \mathcal{X} \to \mathcal{Y}$
- such that $h(x) = \operatorname{argmax}_y \mathbb{P}(Y = y \mid X = x)$
- from $D = X \times Y$

Why? Simpler Means Simpler, probably

Learning a computable function $h : \mathcal{X} \to \mathcal{Y}$ should be simpler than learning the marginal probability distribution. For,

- $\mathbb{P}(Y = y \mid X = x)$ contains more information than
- $\operatorname{argmax}_y \mathbb{P}(Y = y \mid X = x)$
- in the sense that you can use the former to compute the latter, but not vice versa

That is, an algorithm that computes the marginal distribution is easily extended in an algorithm that computes the function h.

- Hence, it is reasonable that expect that computing the classification function has lower complexity than computing the marginal distribution
  - in terms of the amount of data needed
  - in terms of computational resources

Note that a reasonable expectation is not necessarily always true

Loss Functions

Say our algorithm learns function h from D, the obvious question is:

- how good is h?

Intuitively this is easy

- the assumption is that there is a *true* function $f : \mathcal{X} \to \mathcal{Y}$
- so we simply compare h with f
- the more often they agree, the better h is.

This comparison is known as a loss function

So, intuitively, the loss is

$$L_f^i(h) = |\{x \mid h(x) \neq f(x)\}|$$

or, if you want, the average of this (over all possible x values). The intuition is good, mathematically, it stinks however.

Cleaning Up Mathematically

The reason this intuitive definition fails is

- the domain we are dealing with may very well be infinite
  - i.e., we need measure theory to make clear what the size of the set is
- the intuitive definition counts
  - a failure for an x that appears once every eon
  - as bad as one that occurs every second

  clearly, that doesn't make sense

Fortunately, both problems disappear if we turn to probabilities:

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

That is, the loss of using h rather than f is

the probability that we make a mistake

Cleaning Up, Realistically

While this is a nice loss function, probably the best one possible, there is a small problem

- it depends on both $\mathcal{D}$ and f, and we know neither!
- in fact, that is what we want to learn
  - as holy grail and as simpler goal respectively

All we have is D, our (finite!) sample

Hence, we have to make do with the training error, aka empirical error, aka empirical risk:

$$L_D(h) = \frac{|\{(x_i, y_i) \in D \mid h(x_i) \neq y_i\}|}{|D|}$$

Finding a function that minimizes this is loss is known as Empirical Risk Minimization (ERM).

Overfitting

Let $D \subseteq \mathcal{D}$ (i.e., the values we find in our sample) be such that:

- $\forall (x, y) \in D : y = 1$ while
- $\forall (x, y) \in \mathcal{D} \setminus D : y = 0$

A function that minimizes the empirical risk is given by

$$h(x) = 1$$

Depending on the respective sizes of $\mathcal{D} \setminus D$ and D, the true loss can be arbitrarily big

- we will miss-classify every new example!

This is what is known as overfitting.

- the example may look a bit contrived, but the problem is real
- an aspect of the problem of induction

The solution we will mostly take is: restricting the hypothesis class

The Online Game

(we are now following some slides from Shai Shalev-Shwartz)

for $t = 1, 2, \ldots$

- our learner is presented with example $x_t \in \mathcal{X}$
- the learner predicts $\hat{y}_t$
- he is shown the true y_t
- if $\hat{y}_t \neq y_t$ the cost is 1

The goal is to make as few mistakes as possible.

Learning Thresholds

The goal is to learn a simple threshold, i.e., our set of hypotheses is given by

$$\mathcal{H} = \{h_\theta \mid \theta \in \mathbb{R}\}, \text{ where } h_\theta(x) = \text{sign}(x - \theta)$$

The three rational learners are

$$\hat{\theta}_t^r = \min\{x_{t'} \mid t' \leq t \wedge f(t') = 1\}$$
$$\hat{\theta}_t^l = \max\{x_{t'} \mid t' \leq t \wedge f(t') = -1\}$$
$$\hat{\theta}_t^m = \frac{\hat{\theta}_t^r + \hat{\theta}_t^l}{2}$$

but you can pick any learner you want.

Your adversarial teacher knows θ and he knows your current estimate $\hat{\theta}_t$, so he will choose

$$x_{t+1} = \frac{\theta + \hat{\theta}_t}{2}$$

and your learner will be wrong *every time*

Too Rich

It might have shocked you that we cannot even learn such a simple example

- the reason is that it isn't simple at all
- the hypothesis class is far too rich (expressive)

If we restrict ourselves to integer thresholds

- for the moment both for the hypotheses and the true classification function
- we'll look at the more general problem later

it is suddenly an easy to learn task.

And this is a major component of this course:

- studying hypothesis classes we can actually learn

starting with finite hypothesis classes.

Finite Hypothesis Classes

Finite isn't that Trivial?

Examples of finite hypothesis classes are

- threshold function with 256 bits precision reals
  - who would need or even want more?
- conjunctions
  - a class we will meet quite often during the course
- all Python programs of at most 10^{32} characters
  - automatic programming aka inductive programming
  - given a (large) set of input/output pairs
  - you don't program, you learn!

Whether or not these are trivial learning tasks, I'll leave to you

- but, if you think automatic programming is trivial, I am interested in your system

It isn't just about theory, but also very much about practice.

The Set-Up

We have

- a finite set $\mathcal{H}$ of hypotheses
- and a (finite) sample $D \sim \mathcal{D}$
- and there exists a function $f : \mathcal{X} \to \mathcal{Y}$ that does the labelling

Note that since Y is completely determined by X, we will often view $\mathcal{D}$ as the distribution for $\mathcal{X}$ rather than for $\mathcal{X} \times \mathcal{Y}$.

The $\text{ERM}_{\mathcal{H}}$ learning rule tells us that we should pick a hypothesis h_D such that

$$h_D \in \operatorname*{argmin}_{h \in \mathcal{H}} L_D(h)$$

That is we should pick a hypothesis that has minimal empirical risk

The Realizability Assumption

For the moment we are going to assume that the true hypothesis is in $\mathcal{H}$; we will relax this later. More precisely, we are assuming that

$$\textit{there exists a } h^* \in \mathcal{H} \textit{ such that } L_{\mathcal{D},f}(h^*) = 0$$

Note that this means that with probability 1

- $L_D(h^*) = 0$

(there are bad samples, but the vast majority is good).

This implies that,

- for (almost any) sample D the $\text{ERM}_{\mathcal{H}}$ learning rule will give us a hypothesis h_D for which

$$L_D(h_D) = 0$$

The Consistent Learner

A simple way to implement the $\text{ERM}_{\mathcal{H}}$ learning rule is by the following algorithm; in which V_t denotes the hypotheses that are still viable at step t

- the first t $d \in D$ you have seen are consistent with all hypotheses in V_t.
- all $h \in V_t$ classify $x_1, \ldots, x_{t-1}$ correctly, all hypotheses in $\mathcal{H} \setminus V_t$ make at least 1 classification mistake

V is used because of version spaces

1. $V_1 = \mathcal{H}$
2. For t = 1, 2. ...
   - 2.1 take x_t from D
   - 2.2 choose a hypothesis $h_t \in V_t$
   - 2.3 predict $\hat{y}_t = h_t(x_t)$
   - 2.4 get y_t from D (i.e., $(x_t, y_t) \in D$)
   - 2.5 $V_{t+1} = \{h \in V_t \mid h(x_t) = y_t\}$

Analysing the Consistent Learner

Firstly note that

- that because of the realizability assumption we will get at least one hypothesis that makes zero errors

But, how many mistakes will we take?

- we could be unlucky in choosing $h_t \in V_t$
- it could be the only hypothesis that makes a wrong prediction
- if we are consistently unlucky

we will make

$$|\mathcal{H}| - 1$$

mistakes. Can't we do better?

The Halving Learner

We make so many mistakes because we predict on the basis of 1 hypothesis only. If we would listen to the majority, we make far fewer:

1. $V_1 = \mathcal{H}$
2. For t = 1, 2. ...
   - 2.1 take x_t from D
   - 2.2 predict majority $(\{h(x_t) \mid h \in V_t\})$
   - 2.3 get y_t from D (i.e., $(x_t, y_t) \in D$)
   - 2.4 $V_{t+1} = \{h \in V_t \mid h(x_t) = y_t\}$

Which only makes

$$\log(|\mathcal{H}|)$$

mistakes

- with every mistake you reject at least half of the hypotheses.

But, How About Complexity?

With the halving learner we make fewer mistakes

- which is good

But we may need to examine every $x \in D$

- for it may be the very last x we see that allows us to discard many members of V_t

In other words, the halving algorithm is

$$O\left(|D|\right)$$

Linear time is OK, but sublinear is better.

Sampling is one way to achieve this

Thresholds Again

To make our threshold example finite, we assume that for some (large) n

$$\theta \in \{0, \frac{1}{n}.\frac{2}{n}, \ldots, 1\}$$

Basically, we are searching for an element of that set

- and we know how to search fast

To search fast, you use a search tree

- the index in many DBMSs

The difference is that we

- build the index on the fly

We do that by maintaining an interval

- an interval containing the remaining possibilities for the threshold (that is, the halving algorithm)

Statistically halving this interval every time

- gives us a logarithmic algorithm

The Algorithm

- $l_1 := -\frac{0.5}{n}, r_1 = 1 + \frac{0.5}{n}$
- for $t = 1, 2, \ldots$
  - get $x_t \in [l_t, r_t] \cap \{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\}$
    - (i.e., pick again if you draw an non-viable threshold)
  - predict $sign((x_t - l_t) - (r_t - x_t))$
  - get y_t
    - if $y_t = 1$, $l_{t+1} := l_t, r_{t+1} := x_t - \frac{0.5}{n}$
    - if $y_t = -1$, $l_{t+1} := x_t + \frac{0.5}{n}, r_{t+1} := r_t$

Note, this algorithm is only *expected* to be efficient

- you could be getting x_t's at the edges of the interval all the time
  - hence reducing the interval width by 1 only
- while, e.g., the threshold is exactly in the middle

Sampling

If we are going to be linear in the worst case, the problem is:

how big is linear?

That is, at how big a data set should we look

- until we are reasonably sure that we have almost the correct function?

In still other words.

how big a sample should we take to be reasonably sure we are reasonably correct?

The smaller the necessary sample is

- the less bad linearity (or even polynomial) will hurt

But, we rely on a sample, so we can be mistaken

- we want a guarantee that the probability of a big mistake is small

IID

(Note, $\mathcal{X} \sim \mathcal{D}$, $\mathcal{Y}$ computed using the (unknown) function f).
Our data set D is sampled from $\mathcal{D}$. More precisely, this means that we assume that

> *all the $x_i \in D$ have been sampled independently and identically distributed according to $\mathcal{D}$*

- when we sample x_i we do not take into account what we sampled in any of the previous (or future) rounds
- we always sample from $\mathcal{D}$

If our data set D has m members we can denote the iid assumption by stating that

$$D \sim \mathcal{D}^m$$

where $\mathcal{D}^m$ is the distribution over m-tuples induced by $\mathcal{D}$.

Loss as a Random Variable

According to the $\text{ERM}_{\mathcal{H}}$ learning rule we choose h_D such that

$$h_D \in \operatorname*{argmin}_{h \in \mathcal{H}} L_D(h)$$

Hence, there is randomness caused by

- sampling D and
- choosing h_D

Hence, the loss $L_{\mathcal{D},f}(h_D)$ is a random variable. A problem we are interested in is

- the *probability* to sample a data set for which $L_{\mathcal{D},f}(h_D)$ is not too large

usually, we denote

- the probability of getting a non-representative (bad) sample by δ
- and we call $(1-\delta)$ the confidence (or confidence parameter) of our prediction

Accuracy

So, what is a bad sample?

- simply a sample that gives us a high loss

To formalise this we use the accuracy parameter ϵ:

1. a sample D is good if $L_{\mathcal{D},f}(h_D) \leq \epsilon$
2. a sample D is bad if $L_{\mathcal{D},f}(h_D) > \epsilon$

If we want to know how big a sample D should be, we are interested in

- an upperbound on the probability that a sample of size m (the size of D) is bad

That is, an upperbound on:

$$\mathcal{D}^m\left(\{D \mid L_{\mathcal{D},f}(h_D) > \epsilon\}\right)$$

Misleading Samples, Bad Hypotheses

Let $\mathcal{H}_B$ be the set of bad hypotheses:

$$\mathcal{H}_B = \{h \in \mathcal{H} \mid L_{\mathcal{D},f}(h) > \epsilon\}$$

A misleading sample teaches us a bad hypothesis:

$$M = \{D \mid \exists h \in \mathcal{H}_B : L_D(h) = 0\}$$

On sample D we discover h_D. Now note that because of the realizability assumption

$$L_D(h_D) = 0$$

So, $L_{\mathcal{D},f}(h_D) > \epsilon$ can only happen

- if there is a $h \in \mathcal{H}_B$ for which $L_D(h) = 0$

that is, if our sample is misleading. That is,

$$\{D \mid L_{\mathcal{D},f}(h_D) > \epsilon\} \subseteq M$$

a bound on the probability of getting a sample from M gives us a bound on learning a bad hypothesis!

Computing a Bound

Note that

$$M = \{D \mid \exists h \in \mathcal{H}_B : L_D(h) = 0\} = \bigcup_{h \in \mathcal{H}_B} \{D \mid L_D(h) = 0\}$$

Hence,

$$\begin{aligned}
\mathcal{D}^m\left(\{D \mid L_{\mathcal{D},f}(h_D) > \epsilon\}\right) &\leq \mathcal{D}^m(M) \\
&\leq \mathcal{D}^m\left(\bigcup_{h \in \mathcal{H}_B} \{D \mid L_D(h) = 0\}\right) \\
&\leq \sum_{h \in \mathcal{H}_B} \mathcal{D}^m\left(\{D \mid L_D(h) = 0\}\right)
\end{aligned}$$

To get a more manageable bound, we bound this sum further, by bounding each of the summands

Bounding the Sum

First, note that

$$\begin{aligned}\mathcal{D}^m\left(\{D \mid L_D(h) = 0\}\right) &= \mathcal{D}^m\left(\{D \mid \forall x_i \in D : h(x_i) = f(x_i)\}\right) \\ &= \prod_{i=1}^{m} \mathcal{D}\left(\{x_i : h(x_i) = f(x_i)\}\right)\end{aligned}$$

Now, because $h \in \mathcal{H}_B$, we have that

$$\mathcal{D}\left(\{x_i : h(x_i) = y_i\}\right) = 1 - L_{\mathcal{D},f}(h) \leq 1 - \epsilon$$

Hence we have that

$$\mathcal{D}^m\left(\{D \mid L_D(h) = 0\}\right) \leq (1-\epsilon)^m \leq e^{-\epsilon m}$$

(Recall that $1 - x \leq e^{-x}$).

Putting it all Together

Combining all our bounds we have shown that

$$\mathcal{D}^m\left(\{D \mid L_{\mathcal{D},f}(h_D) > \epsilon\}\right) \leq |\mathcal{H}_B| e^{-\epsilon m} \leq |\mathcal{H}| e^{-\epsilon m}$$

So what does that mean?

- it means that if we take a large enough sample (when m is large enough)
- the probability that we have a bad sample
  - the function we induce is rather bad (loss larger than ϵ)
- is small

That is, by choosing our sample size, we control how likely it is learn we learn a well-performing function. We'll formalize this on the next slide.

Theorem

Let $\mathcal{H}$ be a finite hypothesis space. Let $\delta \in (0, 1)$, let $\epsilon > 0$ and let $m \in \mathbb{N}$ such that

$$m \geq \frac{\log\left(|\mathcal{H}|/\delta\right)}{\epsilon}$$

Then, for any labelling function f and distribution $\mathcal{D}$ for which the realizability assumption holds, with probability of at least $1 - \delta$ over the choice of an i.i.d. sample D of size m we have that for every ERM hypothesis h_D:

$$L_{\mathcal{D},f}(h_D) \leq \epsilon$$

This is an instance Probably Approximately Correct (PAC) learning which we'll study over the next few sessions. For know note that this theorem tells us that our simple threshold learning algorithm will in general perform well on a logarithmic sized sample.