Non Uniform Learnability

prof. dr Arno Siebes

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

Relaxing

We have seen that PAC learning is possible exactly when the VC dimension is finite

- other hypotheses classes can not be learned with the guarantees that PAC learning offers

But, what if we are willing to relax the guarantees that PAC offers?

- can we then learn a wider class of hypotheses?

We end by looking at two possibilities

- today: forgetting about uniformity
- next time: no longer insisting on strong classifiers

The remarkable result in both cases is the approximation of the looser framework by/to PAC learning

- non-uniform is approximated by PAC learning
- weak learners can approximate strong learners.

PAC Learning isn't a bad idea

The Only Other Rational Possibility

The two alternatives to PAC learning we discuss are not all there is. There is one more constraint that we could relax:

- the requirement that the learning works whatever the distribution $\mathcal{D}$ is

That is, we could pursue a theory that works for specific distributions

- that theory, however, already exists

It is known as the field of

Statistics

While there are many interesting problems in the intersection of computer science and statistics

- that area is too large and diverse to fit the scope of this course

SRM

PAC Learnability

Before we relax our requirements, it is probably good to recall the (general) definition of PAC learnability:

A hypothesis class $\mathcal{H}$ is agnostic PAC learnable with respect to a set Z and a loss function $l : Z \times \mathcal{H} \to \mathbb{R}_+$ if there exists a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm A with the following property:

- for every $\epsilon, \delta \in (0,1)$
- for every distribution $\mathcal{D}$ over Z
- when running A on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. samples generated by $\mathcal{D}$
- A returns a hypothesis $h \in \mathcal{H}$ such that with probability at least $1 - \delta$
$$L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon$$

The Sample Complexity

In this definition, the sample complexity $m_{\mathcal{H}}(\epsilon, \delta)$

- depends only on ϵ and δ
- it *does not* depend on a particular $h \in \mathcal{H}$
- the bound is *uniform* for all hypotheses

This appears like a reasonable requirement to relax

- as one can imagine that more complex hypothesis require more data than simpler ones even if they are in the same hypothesis class.

In fact, we have already seen examples of this

- for $C_n \subset M_n$ and $m_{C_n} < m_{M_n}$

So if we happen to be learning a function from C_n, but considered M_n as our hypothesis class

- one could say that we are using too many examples.

In non-uniform learning this constraint is relaxed, the size of the sample is allowed to depend on h.

A Direct Consequence

When PAC learning, we want to find a good hypothesis, one that is with high probability approximately correct

- one with $L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon$

Clearly, when learning non-uniformly we no longer can require this to hold. After all, if each $h \in \mathcal{H}$ has its own (minimal) sample size

- computing $\min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h')$ might require an infinitely large sample!
- think, e.g., of the set of all possible polynomials
  - if there is no bound on the degree, there can be no bound on how much data we need to estimate the best fitting polynomial
  - after all, we have already seen that the higher the degree, the more data we need

Clearly we still want a quality guarantee. What we can do is

- is to require that the learning is as good as possible given a certain sample (size)

Competitive

What does it mean that the learning is as good as possible?

- it means the hypothesis we learn is with high probability close to the best one
- i.e., the hypothesis we find is *competitive* with the rest

Two hypotheses are equally good if we expect a similar loss for both of them. Formalizing this we say that hypothesis h_1 is (ϵ, δ) competitive with hypothesis h_2 if with probability at least $(1 - \delta)$

$$L_{\mathcal{D}}(h_1) \leq L_{\mathcal{D}}(h_2) + \epsilon$$

A good learner should find a hypothesis that is competitive with all other hypotheses in $\mathcal{H}$

Note that this is very much true in the (uniform) PAC learning setting, i.e., PAC learning will be a special case of non-uniform learning.

Non-Uniformly Learnable

Based on this idea, we formalize non-uniformly learnable as follows:

A hypothesis class $\mathcal{H}$ is non-uniformly learnable if there exists a learning algorithm A and a function $m_{\mathcal{H}}^{NUL} : (0,1)^2 \times \mathcal{H} \to \mathbb{N}$ such that

- for every $\epsilon, \delta \in (0,1)$
- for every $h \in \mathcal{H}$
- when running A on $m \geq m_{\mathcal{H}}^{NUL}(\epsilon, \delta, h)$ i.i.d. samples
- then for every distribution $\mathcal{D}$ over Z
- it holds that for with probability at least $1 - \delta$ over the choice of $D \sim \mathcal{D}^m$
$$L_{\mathcal{D}}(A(D)) \leq L_{\mathcal{D}}(h) + \epsilon$$

Given a data set, A will, with high probability, deliver a competitive hypothesis; that is, competitive with those hypotheses whose sample complexity is less than $|D|$.

Characterizing Non-Uniform Learnability

There is a surprising link between uniform and non-uniform learning:

A hypothesis class $\mathcal{H}$ of binary classifiers is non-uniformly learnable iff it is the countable union of agnostic PAC learnable hypothesis classes.

The proof of this theorem relies on another theorem:

Let $\mathcal{H}$ be a hypothesis class that can be written as a countable union $\mathcal{H} = \cup_{n \in \mathbb{N}} \mathcal{H}_n$, where for all n, $VC(\mathcal{H}_n) < \infty$, then $\mathcal{H}$ is non-uniformly learnable.

Note that the second theorem is the equivalent of the *if* part of the first. The proof of the second theorem will be discussed (a bit) later.

Proving Only If

Let $\mathcal{H}$ be non-uniformly learnable. That means that we have a function $m_{\mathcal{H}}^{NUL} : (0,1)^2 \times \mathcal{H} \to \mathbb{N}$ to compute sample sizes.

- for a given ϵ_0, δ_0 define for every $n \in \mathbb{N}$

$$\mathcal{H}_n = \{h \in \mathcal{H} \mid m_{\mathcal{H}}^{NUL}(\epsilon_0, \delta_0, h) \leq n\}$$

- clearly, for every ϵ_0 and δ_0 we have that

$$\mathcal{H} = \cup_{n \in \mathbb{N}} \mathcal{H}_n$$

- Moreover, for every $h \in \mathcal{H}_n$ we know that with probability of at least $1 - \delta_0$ over $D \sim \mathcal{D}^n$ we have $L_{\mathcal{D}}(A(D)) \leq L_{\mathcal{D}}(h) + \epsilon_0$.
- since this holds *uniformly* for all $h \in \mathcal{H}_n$
- we have that H_n is agnostic PAC learnable

Note that we carve up $\mathcal{H}$ differently for every (ϵ, δ) pair, but that is fine. Any choice writes $\mathcal{H}$ as the countable union of agnostic PAC learnable classes - $\mathcal{H}$ does not become magically agnostic PAC learnable

Approach to Prove If

The proof of the opposite direction

- the countable union gives you non-uniform learnability

requires more work.

The main idea is, of course, to compute an error bound

- how much bigger than L_D can $L_{\mathcal{D}}$ be
  - knowing that $\mathcal{H}$ is the countable union...

This bound suggests a new learning rule

- from expected risk minimization to structural risk minimization

A learning rule

- that can do non-uniform learning

Background Knowledge

The new framework for learning we are building up rests on two assumptions:

- that $\mathcal{H} = \cup_{n \in \mathbb{N}} \mathcal{H}_n$
- and a weight function $w : \mathbb{N} \to [0, 1]$

Both can be seen as a form of background knowledge

- the choice of $\mathcal{H}$ itself is already background knowledge, putting structure to it even more so
- all the more since w allows us to specify where in $\mathcal{H}$ we expect it to be likely to find the model ($w(n)$ high, chance of $\mathcal{H}_n$ high)

We will see that the better your background knowledge is, the fewer data points you need.

Uniform Convergence

To build up this new framework, the (equivalent) formulation of PAC learnability that is most convenient is that of uniform convergence. To simplify your life, we repeat the definition:

A hypothesis class $\mathcal{H}$ has the *uniform convergence property* wrt domain Z and loss function l if

- there exists a function $m_{\mathcal{H}}^{UC} : (0,1)^2 \to \mathbb{N}$
- such that for all $(\epsilon, \delta) \in (0,1)^2$
- and for any probability distribution $\mathcal{D}$ on Z

If D is an i.i.d. sample according to $\mathcal{D}$ over Z of size $m \geq m_{\mathcal{H}}^{UC}(\epsilon, \delta)$. Then D is ϵ-representative with probability of at least $1 - \delta$.

Where ϵ-representative means

$$\forall h \in \mathcal{H} : |L_D(h) - L_{\mathcal{D}}(h)| \leq \epsilon$$

The ϵ_n Function

We assume that $\mathcal{H} = \cup_{n \in \mathbb{N}} \mathcal{H}_n$

- and that each $\mathcal{H}_n$ has the uniform convergence property

Now define the function $\epsilon_n : \mathbb{N} \times (0,1) \rightarrow (0,1)$ by

$$\epsilon_n(m, \delta) = \min\{\epsilon \in (0,1) \mid m^{UC}_{\mathcal{H}_n}(\epsilon, \delta) \leq m\}$$

That is, given a fixed sample size, we are interested in the smallest possible gap between empirical and true risk. To see this, substitute $\epsilon_n(m, \delta)$ in the definition of uniform convergence, then we get:

For every m and δ with probability of at least $1 - \delta$ over the choice of $D \sim \mathcal{D}^m$ we have

$$\forall h \in \mathcal{H}_n : |L_{\mathcal{D}}(h) - L_D(h)| \leq \epsilon_n(m, \delta)$$

This is the bound we want to extend to all of $\mathcal{H}$

The Weight Function

For that we use the weight function $w : \mathbb{N} \to [0,1]$. Not any such function will do, it should be a convergent sequence, more precisely we require that

$$\sum_{i=1}^{\infty} w(n) \leq 1$$

In a finite case, this is easy to achieve

- if you have no idea which $\mathcal{H}_n$ is best you can simply choose a uniform distribution

In the countable infinite case you *can not* do that

- the sum would diverge

And even if you have a justified believe that the lower n is, the likelier that $\mathcal{H}_n$ contains the right hypothesis, it is not easy to choose between

$$w(n) = \frac{6}{\pi n^2} \text{ and } w(n) = 2^{-n}$$

well see a rational approach after the break.

Bounding Non-Uniform Loss

Let $w : \mathbb{N} \to [0,1]$ be a function such that $\sum_{i=1}^{\infty} w(n) \leq 1$. Let $\mathcal{H}$ be a hypothesis class that can be written as $\cup_{n \in \mathbb{N}} \mathcal{H}_n$ where each each $\mathcal{H}_n$ has the uniform convergence property. Let $\epsilon_n(m, \delta)$ be as defined before, i.e., $\min\{\epsilon \in (0,1) \mid m_{\mathcal{H}_n}^{UC}(\epsilon, \delta) \leq m\}$. Then

- for every $\delta \in (0,1)$ and every distribution $\mathcal{D}$
- with probability of at least $1 - \delta$ over the choice of $D \sim \mathcal{D}^m$

$$\forall n \in \mathbb{N}\ \forall h \in \mathcal{H}_n : |L_{\mathcal{D}}(h) - L_D(h)| \leq \epsilon_n(m, w(n)\delta)$$

Therefore, every $\delta \in (0,1)$ and every distribution $\mathcal{D}$ with probability of at least $1 - \delta$

$$\forall h \in \mathcal{H} : L_{\mathcal{D}}(h) \leq L_D(h) + \min_{\substack{n \in \mathbb{N} \\ h \in \mathcal{H}_n}} \epsilon_n(m, w(n)\delta)$$

The bound we were looking for

Proof

Define for $n \in \mathbb{N}$, $\delta_n = w(n)\delta$. Then we know that if we fix n

- we have with probability at least $1 - \delta_n$ over the choice of $D \sim \mathcal{D}^m$

$$\forall h \in \mathcal{H}_n : |L_{\mathcal{D}}(h) - L_D(h)| \leq \epsilon_n(m, \delta_n)$$

Applying the union bound over $n = 1, 2, \ldots$ then gives us that

- with probability at least

$$1 - \sum_{n \in \mathbb{N}} \delta_n = 1 - \delta \sum_{n \in \mathbb{N}} w(n) \geq 1 - \delta$$

- that

$$\forall n \in \mathbb{N}\; \forall h \in \mathcal{H}_n : |L_{\mathcal{D}}(h) - L_D(h)| \leq \epsilon_n(m, \delta_n)$$

and the proof is done.

Structural Risk Minimization

The error you estimate for a $h \in \mathcal{H}$ depends on the $\mathcal{H}_n$, h is a member of. If it is a member of multiple, one should, of course, go for the smallest n:

$$n(h) = \min\{n \mid h \in \mathcal{H}_n\}$$

Then we have

$$L_{\mathcal{D}}(h) \leq L_D(h) + \epsilon_{n(h)}(m, w(n(h))\delta)$$

The Structural Risk Minimization Learning Rule is to output

$$h \in \underset{h \in \mathcal{H}}{\operatorname{argmin}} \left[L_D(h) + \epsilon_{n(h)}(m, w(n(h))\delta) \right]$$

So, not just minimal empirical risk, but a balance between

- the empirical risk $L_D(h)$
- and the "class-risk" $\epsilon_{n(h)}(m, w(n(h))\delta)$

SRM Learning Works

Let $\mathcal{H}$ be a hypothesis class that can be written as $\cup_{n\in\mathbb{N}}\mathcal{H}_n$ where each each $\mathcal{H}_n$ has the uniform convergence property with sample complexity $m^{UC}_{\mathcal{H}_n}$. Let $w(n) = \frac{6}{n^2\pi^2}$. Then

- $\mathcal{H}$ is non-uniformly learnable using the SRM rule with sample complexity

$$m^{NUL}_{\mathcal{H}}(\epsilon, \delta, h) \leq m^{UC}_{\mathcal{H}_{n(h)}}\left(\epsilon/2, \frac{6\delta}{(\pi n(h))^2}\right)$$

Note that

- this theorem does hold far more general than for this specific weight function only
- the choice for the weight function directly influences the complexity and that makes it hard to write a general form

Moreover note

- this result also finishes the proof that non-uniformly learnable equates with a countable union of of classes with a finite VC dimension.

Proof

First of all, note that $\sum_{n\in\mathbb{N}} w(n) = 1$. Next, let A be the SRM learning algorithm with respect to $w(n)$ And for all $h \in \mathcal{H}$, ϵ, and δ, let $m \geq m^{UC}_{\mathcal{H}_{n(h)}}(\epsilon, w(n(h))\delta)$.

- then, with probability at least $1 - \delta$ for the choice of $D \sim \mathcal{D}^m$
- for all $h' \in \mathcal{H}$

$$L_{\mathcal{D}}(h') \leq L_D(h') + \epsilon_{n(h')}(m, w(n(h'))\delta)$$

This holds in particular for hypothesis $A(D)$. By the definition of SRM we get:

$$\begin{aligned} L_{\mathcal{D}}(A(D)) &\leq \min_{h'} \left[L_D(h') + \epsilon_{n(h')}(m, w(n(h'))\delta) \right] \\ &\leq L_D(h) + \epsilon_{n(h)}(m, w(n(h))\delta) \end{aligned}$$

Proof continued

So, we have that $L_{\mathcal{D}}(A(D)) \leq L_D(h) + \epsilon_{n(h)}(m, w(n(h))\delta)$.

- by definition we have that $m \geq m^{UC}_{\mathcal{H}_{n(h)}}(\epsilon/2, w(n(h))\delta)$ implies that $\epsilon_{n(h)}(m, w(n(h))\delta) \leq \epsilon/2$.
- moreover, because the $\mathcal{H}_n$ have the universal convergence property , we now with probability at least $1 - \delta$:

$$L_D(h) \leq L_{\mathcal{D}}(h) + \epsilon/2$$

That is:

$$\begin{aligned} L_{\mathcal{D}}(A(D)) &\leq L_D(h) + \epsilon_{n(h)}(m, w(n(h))\delta) \\ &\leq L_{\mathcal{D}}(h) + \epsilon/2 + \epsilon/2 \\ &\leq L_{\mathcal{D}}(h) + \epsilon \end{aligned}$$

For the sample complexity, note that

$$m^{UC}_{\mathcal{H}_{n(h)}}(\epsilon/2, w(n(h))\delta) = m^{UC}_{\mathcal{H}_{n(h)}}\left(\epsilon/2, \frac{6\delta}{(\pi n(h))^2}\right)$$

Learning Uniformly

An often used approach to learn non-uniformly is to posit a tower of hypothesis classes

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \mathcal{H}_3 \subset \cdots$$

The we start by selecting the best model from $\mathcal{H}_1$, then from $\mathcal{H}_1 \cup \mathcal{H}_2 = \mathcal{H}_2$ and so on

- note that at each step the choice for the function $w(n)$ is extremely simple

while keeping an eye on

$$L_D(h) + \epsilon_{n(h)}(m, w(n(h))\delta)$$

And choose the model where the risk is minimal

You can do that, for example.

- to learn a polynomial classifier when you don't know what the optimal degree would be

Note that the degree cannot exceed $|D|$ anyway.

The Most Famous Example

The most famous use of Structural Risk Minimization is without a doubt

Support Vector Machines

In fact, exaggerating a bit, one could almost say

- that SRM was invented just to do that

It is an exaggeration as in its the simplest form (equivalent to perceptrons and hyperplane classification)

- SVM's were invented by Vapnik and Chervonenkis a decade before they invented SRM

Only later, in collaboration with many other smart people

- Vapnik developed SVM's in their full glory

A full treatment of SVM's is beyond our scope, we do look at the simplest case, however.

Linear SVM's

As usual we have a data set D over $\mathcal{X} \times \mathcal{Y}$ where

- $\mathcal{X} = \mathbb{R}^n$ for some $n \in \mathbb{N}$
- $\mathcal{Y} = \{-1, +1\}$

And as usual we want to learn a classifier from $\mathcal{X}$ to $\mathcal{Y}$. In the simplest case, we assume

- that the two classes are linearly separable

That is, there is at least one hyperplane that is a perfect classifier for the given data.

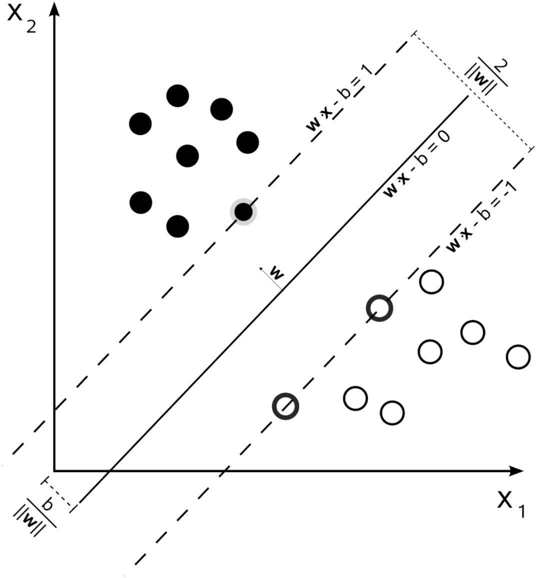
If one hyperplane separates,

- many will

So, the question is: which one should we choose?

- the answer is: the one that maximizes the margin.

The Picture



x_2
x_1
$\mathbf{w}\cdot\mathbf{x}-b=1$
$\mathbf{w}\cdot\mathbf{x}-b=0$
$\mathbf{w}\cdot\mathbf{x}-b=-1$
$\frac{2}{\|\mathbf{w}\|}$
$\frac{b}{\|\mathbf{w}\|}$
$\mathbf{w}$

(thank you wikipedia)

Why Maximizing the Margin?

So, the theory tells us that we should maximize the margin

- given by $\frac{2}{||\mathbf{w}||}$
- i.e., finding the solution with minimal $||\mathbf{w}||$

While this may make intuitive sense

- minimizing the weights is akin to making sure that as many weights as possible are set to 0
- the famous Lasso penalty term some of you know

This is not a very satisfactory explanation. It turns out there is a far nicer explanation

- Vapnik proved that in this case, maximal margin corresponds to minimal VC dimension

Hence, everything we have learned tells us: maximize that margin

Vapnik on Margins

Consider hyperplanes $h(\vec{x}) = sign(\vec{w} \cdot \vec{x} + b)$ as hypothesis class for $\mathbb{R}^n$. Let all examples x_i in a ball with radius R and assume that for all x_i it holds that

$$abs(\vec{w} \cdot \vec{x}_i + b) \geq 1$$

Then this set of hyperplanes has a VC dimension d that is bounded by

$$d \leq \min \left\{ \left\lfloor \frac{R^2}{\delta^2} \right\rfloor , n \right\}$$

in which δ is the margin.

More Structure

In the second lecture we discussed the curse of dimensionality

- while we haven't discussed it yet in our ERM/SRM framework, it clearly should play a role

We can battle it here in our search for a separating hyperplane

- by taking progressively larger sets of features for the hyperplane construction

That is, e.g.,

- you start with x_1, then x_1 and x_2 and so on
- variables/features/attributes you don't use have their weight effectively set to 0 (the Lasso again!)

SRM then tells you which hyperplane, and thus which feature set, to choose.

The Cost of SRM

Since non-uniform learning equals learning from

- the countable union of hypothesis classes with a finite VC dimension

And the chosen model h obviously is in some $\mathcal{H}_n$

- i.e., $\exists n \in \mathbb{N} : A(D) \in \mathcal{H}_n$

One could wonder what the cost is for using all of $\mathcal{H}$

- rather than just $\mathcal{H}_n$

A rather straight forward calculation shows that

$$m_{\mathcal{H}}^{NUL}(\epsilon, \delta, h) - m_{\mathcal{H}_n}^{UC}(\epsilon/2, \delta) = O\left(\frac{\log(2n)}{\epsilon^2}\right)$$

That is, the cost increases with the log of the index

- which makes intuitive sense

MDL

The Weight Function, Again

The one conceptually weak point of SRM is that we need a weight function $w : \mathbb{N} \to [0,1]$ for which

$$\sum_{n \in \mathbb{N}} w(n) \leq 1$$

Clearly, in any practical case using a finite union of hypothesis classes is sufficient

- if only because we have a finite data set D to start with
- even Big Data is finite

In the infinite case we can start with a convergent series, such as,

$$\sum_{n \in \mathbb{N}} \frac{1}{n^2} = \frac{\pi^2}{6} \text{ or } \sum_{n \in \mathbb{N}} \frac{1}{2^n} = 1$$

and take the elements of the sequence as the weight

- and you can even swap some elements around if you think that $w(3)$ should be bigger than $w(2)$

How to Choose?

So, there are infinitely many possibilities to choose from

- all of them more or less showing the same qualitative effect that larger *n*'s get smaller weights
- but quantitatively different, i.e., the actual weights are different

And we have seen that the error term

- and thus the necessary sample size to achieve some desired maximal error

depend on the actual weights.

That is a conceptually weak point,

- we have to make a choice,
- one that has a direct effect on our results

and there doesn't seem to be a way to choose

- relying on expert knowledge seems a weak excuse here

Fortunately, there is an objective way to assign weights.

Countable Hypothesis Classes

We are going to assume that $\mathcal{H}$ is a countable set of hypotheses. The first observation is that this is not really a limiting assumption

- we are dealing with machine learning: the model should be learned by a computer
- hence, a computer should be able to represent it
- such a representation is ultimately a *finite* bit string
  - if there would be no finite representation, how could one ever say that the computer has learned the model?
- and there are only countable many such bit strings

One could equivalently argue that

- a hypothesis class should be (recursively) enumerable
- how can we evaluate a model we cannot reach?

In other words, all hypothesis classes we can consider are countable

Countable Class is Countable Union

If $\mathcal{H}$ is countable, we have the countable union

$$\mathcal{H} = \cup_{h\in\mathcal{H}}\{h\} = \cup_{n\in\mathbb{N}}\{h_n\}$$

Clearly, each set $\{h_n\}$ is finite and for finite $\mathcal{H}$ we know that they are UC with

$$m_{\mathcal{H}}^{UC}(\epsilon,\delta) = \left\lceil \frac{\log(2|\mathcal{H}|/\delta)}{2\epsilon^2} \right\rceil$$

So, we have

$$m_{\{h_n\}}^{UC}(\epsilon,\delta) = \left\lceil \frac{\log(2/\delta)}{2\epsilon^2} \right\rceil$$

This means that ϵ_n, defined by

$$\epsilon_n(m,\delta) = \min\{\epsilon \in (0,1) \mid m_{\mathcal{H}_n}^{UC}(\epsilon,\delta) \le m\}$$

becomes

$$\epsilon_n(m,\delta) = \min\{\epsilon \in (0,1) \mid \left\lceil \frac{\log(2/\delta)}{2\epsilon^2} \right\rceil \le m\} = \sqrt{\frac{\log(2/\delta)}{2m}}$$

SRM for Singleton Classes

Substituting $\sqrt{\frac{\log(2/\delta)}{2m}}$ for ϵ_n is the SRM rule gives us:

$$\begin{aligned}
&\operatorname*{argmin}_{h_n\in\mathcal{H}}\left[L_D(h_n)+\epsilon_{n(h_n)}(m,w(n(h_n))\delta)\right]\\
&=\operatorname*{argmin}_{h_n\in\mathcal{H}}\left[L_D(h_n)+\sqrt{\frac{\log(2/w(n)\delta)}{2m}}\right]\\
&=\operatorname*{argmin}_{h_n\in\mathcal{H}}\left[L_D(h_n)+\sqrt{\frac{-\log(w(n))+\log(2/\delta)}{2m}}\right]
\end{aligned}$$

Given that for each n, $\mathcal{H}_n$ is simply a singleton class. we can view w equivalently as a function $\mathcal{H}\to[0,1]$, giving us the SRM rule

$$\operatorname*{argmin}_{h\in\mathcal{H}}\left[L_D(h)+\sqrt{\frac{-\log(w(h))+\log(2/\delta)}{2m}}\right]$$

What is the Weight of a Hypothesis?

The swap from $w(n)$ to $w(h)$ may seem not very useful, but actually it is

- we are going to attach a weight to h based on its *description*

We already noted that h has to be represented somehow

- and that that ultimately can be seen as some bit string

We are going to do this argument a bit more careful:

- each $h \in \mathcal{H}$ has to be described somehow
  - whether by natural language
  - as a mathematical formula
  - a programming language, like C or Python, or ...
  - the latter choice is safe against paradoxes
- This description is always a string over some alphabet
- Coding theory then tells us how to turn that into a word in a prefix code
- which by Kraft's inequality gives us a probability and, thus, a weight!

Coding Theory

We want to store or transmit sequences of elements of a finite set $A = \{a_1, \ldots, a_n\}$ by binary strings

- A is known as the *alphabet*, if we describe our hypotheses in natural language, A would simply be our own well-known alphabet

A *code* is a function

- $C : A \to \{0,1\}^*$
- mapping each symbol in the alphabet to its code word

Coding is easily extended to strings of symbols to sequences by concatenation:

- $C : A^* \to \{0,1\}^*$
- by $C(xy) = C(x)C(y)$

Note, we require a code ($C : A \to \{0,1\}^*$) to be invertible

- otherwise you cannot decode, i.e., recover what the original sequence was

Codes and Trees

A code C defines a binary tree in which each code word $C(a_i)$ denotes a path from the root of the tree to a leaf

- say 0 is branch to the left, 1 is branch to the right
- i.e., you label the edges with 0 and 1
- and the the symbols from your alphabet A in the node where their path ens

This tree makes it easy to decode a binary string

- at least when we know when a code word ends and the next one begins
- we could achieve this by a special symbol
  - a comma, added to our $0/1$ alphabet or a reserved word
- but we can also simply stipulate that no code word is the prefix of another code word
  - all alphabet symbols are in a leaf node

This is known as a *prefix code*

Decoding Prefix Codes

If we have a prefix code C

- decoding a string $C(x)$ with $x \in A^*$

is easy:

- start at the root
- if the first bit is 0 go to the left, otherwise go right
- continue until you hit a leaf: output the symbol in that leaf node and return to the root

Lossless coding/decoding is an important requirement

- in Algorithmic Information Theory

which is perhaps the most interesting topic, but not part of this course.

Kraft's Inequality

For prefix codes there is an important inequality for the lengths of the code words $|C(a)|$, i.e., the number of bits used:

$$\sum_{a \in A} 2^{-|C(a)|} \leq 1$$

This inequality provides a link between probability distributions and coding, both in our finite setting and more general in the countable case. For $a \in A$, its probability is given by

$$\mathbb{P}(a) = \frac{2^{-|C(a)|}}{\sum_{a \in A} 2^{-|C(a)|}}$$

This relationship also holds in the other direction:

- but first we prove Kraft

Proving Kraft

If our code does not correspond to a complete binary tree

- a tree that splits in two at every internal node
- equivalently all leaves of the tree correspond to a symbol in A.

we can always extend it so that it is complete

- adding some bogus symbols to our alphabet

Using Induction:

- Kraft holds for the two leaf tree: both probabilities are $1/2$
- let w be a path with length w splitting the node gives us two paths w_1 and w_2 such that $2^{-|w_1|} + 2^{-|w_2|} = 2^{-|w|}$

In other words, for prefix codes corresponding with complete binary trees equality holds

- in all other cases we get an inequality since we remove the probabilities that correspond to the bogus symbols

Codes and Probabilities

We already saw that prefix code words for an alphabet A define a probability distribution on A by

$$\mathbb{P}(a) = \frac{2^{-|C(a)|}}{\sum_{a \in A} 2^{-|C(a)|}}$$

This relation also holds in the other direction

- for every probability distribution on A
- there is a corresponding prefix code for A

To prove this we first show that if we have a set of integers

$$\{n_1, \ldots, n_k\}$$

such that

$$\sum_{i=1}^{k} 2^{-n_i} \leq 1$$

Then there is an alphabet $A = \{a_1, \ldots, a_k\}$ such that

- there is a prefix encoding C for A
- such that $C(a_i) = n_i$

Constructing the Code

Assume that the n_i are ordered by

$$n_1 \leq n_2 \leq \cdots \leq n_k$$

Take the fully balanced binary tree of depth n_k.

- take the left most path 000..00 till length n_1, choose a symbol for that node
- and cut the rest of the tree below that node

For the other n_i we de the same

- i.e., take the left-most path that does not end in a labelled leaf node and repeat.

Note that this gives us a relation between all (finite) probability distributions and codes by choosing the integers

$$n(a) = \left\lceil \log \left(\frac{1}{\mathbb{P}(a)} \right) \right\rceil$$

Shannon Fano Coding

This simple lemma gives us the promised translation

- from probability distributions on A to coding A

by choosing the integers

$$n(a) = \left\lceil \log \left(\frac{1}{\mathbb{P}(a)} \right) \right\rceil$$

This is known as a Shannon Fano coding of A. It is optimal in the following sense:

Let C be a prefix code for A, with $|C(a_i)| = n_i$ and $\mathbb{P}$ a probability distribution on A with $\mathbb{P}(a_i) = p_i$. Then

1. $\mathbb{E}_{\mathbb{P}}(l) = \sum p_i n_i \geq \sum p_i \log 1/p_i \overset{\text{def}}{=} H(\mathbb{P})$
2. $\mathbb{E}_{\mathbb{P}}(l) = H(\mathbb{P}) \Leftrightarrow \forall i : p_i = 2^{-n_i}$

this is known as Shannon's noise free coding theorem

Proof

We have:

$$\begin{aligned}
\sum p_i \log 1/p_i - \sum p_i n_i &= \sum p_i \log 1/p_i - \sum p_i \log 2^{n_i} \\
&= \sum p_i \log 1/p_i + \sum p_i \log 2^{-n_i} \\
&= \sum p_i \log \frac{2^{-n_i}}{p_i} \\
&= \log e \sum p_i \ln \frac{2^{-n_i}}{p_i} \\
&\leq \log e \left(\sum p_i \frac{2^{-n_i}}{p_i} - 1 \right) \quad \text{because } \ln x \leq x - 1 \\
&= \log e \left(\sum 2^{-n_i} - 1 \right) \leq 0
\end{aligned}$$

Note that all our results also hold for countable A.

Back to Weights

We now have that if we describe our hypotheses

- that is, we encode them with some prefix code C

then Kaft's inequality gives us weights. More specifically, if we denote $|C(h)|$ simply by $|h|$ we can use the weights

$$w(h) = \frac{1}{2^{|h|}}$$

Using this, we have

Let $\mathcal{H}$ be a countable hypothesis class and be $C : \mathcal{H} \to \{0,1\}^*$ be a prefix code for $\mathcal{H}$. Then, for every sample size m, every confidence parameter δ and every probability distribution $\mathcal{D}$, with probability at least $1 - \delta$ over the choice of $D \sim \mathcal{D}^m$ we have that

$$L_{\mathcal{D}}(h) \leq L_D(h) + \sqrt{\frac{|h| + \log(2/\delta)}{2m}}$$

Minimum Description Length

This result suggest the Minimum Description Length Rule:

$$h \in \underset{h \in \mathcal{H}}{\operatorname{argmin}} \left[L_D(h) + \sqrt{\frac{|h| + \log(2/\delta)}{2m}} \right]$$

Note that this is related to the Minimum Description Length principle

- where we choose the model that compresses the data best

But, it is *not* the same

If you wonder

- how about the choice for C?

Doesn't the language you choose matter?

- it does and it does not, but that is too much of a detour