Nati Srebro at UT Austin 2011-06-01 Lecture 3 Draft (v4)

Patrick W. Gallagher

December 17, 2012

Contents

1 Vapnik-Chervonenkis (VC) dimension

1.1 Reminder about cardinality-based deviation bound

First of all, recall that we are trying to find 'good' classifiers to separate some type of inputs, in general. For this, we can use training data and generate a separator that works well on this sample, which we hope works well for other instances too. Here, empirical errors represents the error of our separator on training data and expected error is the real error term that we want to minimize but unkown. By utilization of training data we can provide bound on the difference of our expected and empirical errors.

Recall that the bound on the difference in the expected (R) and empirical ($\hat{R}$) errors: with probability at least $1-\delta$, for all $h \in \mathcal{H}$,

$$\left| R(h) - \hat{R}(h) \right| \leq \sqrt{\frac{\log |\mathcal{H}| + \log \frac{2}{\delta}}{2m}}.$$

Recall that we considered cardinality of a hypothesis class as difficult to measure and use since most classes actually are not finite in cardinality because they are described by a set of continuous parameters. Yet amount such classes of infinite cardinality there are classes that are more complex than other classes. We are looking for a more consistent way of measuring complexity of a class of hypothesis. We have also demonstrated that the number of parameters is not a correct measure, since it fails to represent a true complexity of a class in the example with the sine curve, which had only two parameters and yet was able to achieve any labeling of distinct points on the real line.

We want to keep in mind two notions: "memorization" versus "learning". In particular, we will ask below what is the largest size of sample for which a given classifier can just perfectly "memorize" every possible labeling of some sample of that number of points. We will not know whether a classifier is actually "learning" anything until we know that the classifier is definitely not just "memorizing". It is also worthwhile to consider the possibility of "accidentally" getting a sample that makes a classifier look much better than it "actually" is.

1.2 Example of "actual number of behaviors" on a given sample data set

In particular, if we have a few (say eight) training examples with some labeling (Figure 1.1) then there is an infinite number of linear classifiers but there is only a fairly small number of behaviors that affine classifiers can have on this data.

1.3 The growth function $\Pi(\mathcal{H}, \{x_1, \ldots, x_m\})$ of a given hypothesis class and a specific data sample

We now define the number of different behaviors that a hypothesis class has on a specific set of points and give it a name, i.e., the **growth function**. We define the growth function $\Pi(\mathcal{H}, S)$ for a hypothesis class $\mathcal{H}$ and a set of points $S = \{x_1, \ldots, x_m\}$. We look at all the possible labelings we can have $y_1, \ldots, y_m$ such that there exists some classifier in the class that actually gives these labelings.

The growth function of a given hypothesis class and a specific sample is

$$\Pi(\mathcal{H}, \{x_1, \ldots, x_m\}) \stackrel{\Delta}{=} \left|\{(y_1, \ldots, y_m) \text{ such that } \exists h \in \mathcal{H} \text{ for which } y_i = h(x_i) \text{ for all points } i = 1, \ldots, m\}\right|.$$

Note that here we are not concerned about the "true" labeling of data S, if any. S here is simply a set of points in our domain, and for each predictor in the hypothesis class we write down the labelings that this class gives us and then we count how many different possible labelings of the sample points we generate with this class.

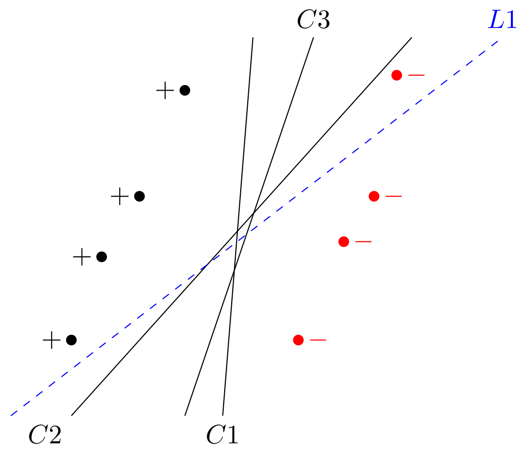


Figure 1.1: Linear classifiers with same (C1,C2,C3) and different (L1) prediction on training sample.

For instance, consider three specific points $(1,0)$, $(2,0)$ and $(3,0)$ on $\mathbb{R}^2$. We can give a linear classifier for labelings $\{(+,+,+),(+,+,-),(+,-,-),(-,-,-),(-,-,+),(-,+,+)\}$ where we cannot for $\{(-,+,-),(+,-,+)\}$. Hence, growth function for this instance is 6.

1.4 The growth function $\Pi(\mathcal{H},m)$ of a given hypothesis class and a particular size of sample

And then we will define the **growth function**[1] for a particular size of sample m as a supremum of the growth function of a given hypothesis class over all sets of m points.

The growth function of a given hypothesis class and a specific size of sample is

$$\Pi(\mathcal{H},m) \triangleq \sup_{x_1,\dots,x_m} \Pi(\mathcal{H},\{x_1,\dots,x_m\})$$

Here, instead of providing labelings to specific data sample, we are looking for the maximum number of growth function for any m data points on given space.

1.5 Behavior of the growth function $\Pi(\mathcal{H},m)$

1.5.1 Simple limits: $|\mathcal{H}|$ and 2^m

Let us see how this growth function behaves? In case of a a finite class the growth function is always less than or equal to the cardinality of the class. If we have only a thousand predictors in our class then we are definitely not going to get any more than a thousand different behaviors because every predictor only gives us one behavior. We might see several predictors with the same behavior, but we definitely cannot have more behaviors than the cardinality.

We know that, for all $\mathcal{H}$ and m

[1] For more information check *http://youtu.be/SEYAnnLazMU?t=27m46s*

$$\Pi(\mathcal{H}, m) \leq |\mathcal{H}|.$$

Also we can claim that the number of behaviors with m points is at most 2^m, since on m points, one can only have 2^m possible labelings. For instance, any three points on $\mathbb{R}^2$ can have at most 2^3 different labelings.

1.5.2 A bound involving the growth function instead of the cardinality

Let us consider the union bound again and how it was derived. We will explain that it is enough to talk about the growth function of the class. For the union bound it is not necessary to consider predictors that have the same labelings on all sample sets of size m. From the point of view of the empirical error such two predictors (hypothesis) are exactly the same. So they do not need to be counted twice in the union bound.

To make this formal requires some work, as far as the generalization error is concerned one has to consider the behavior of two different predictors on other points (not in the sample set) It turns out that one can prove that one can replace in the bound $\sqrt{\frac{\log|\mathcal{H}|+\log\frac{2}{\delta}}{2m}}$ the cardinality with a log of the growth function resulting in $\sqrt{\frac{\log\Pi(\mathcal{H},2m)+\log\frac{4}{\delta}}{m}}$. In order to also address the behavior outside of the training sample one does have to look at the growth function at $2m$ points rather than m points. One way to think about it is considering a situation where we have training points and test points and we look at the growth function (i.e., the number of behaviors) not only on the training points but on the training points and the test points. Then via an argument called symmetrization one gets $\sqrt{\frac{\log\Pi(\mathcal{H},2m)+\log\frac{4}{\delta}}{m}}$ some increase in factors of 2. The bound that now involves the growth function is the same as we had before but instead of counting predictors we only count the distinct behaviors of the predictors.

Deviation bound involving the log growth function, taking into account the performance both on the m training points and the m test points:
With probability at least $1-\delta$, for all hypotheses $h \in \mathcal{H}$

$$\left|R(h) - \hat{R}(h)\right| \leq \sqrt{\frac{\log\Pi(\mathcal{H}, 2m) + \log\frac{4}{\delta}}{m}}.$$

1.6 Graphing the log growth function $\log\Pi(\mathcal{H}, m)$

So, now, we are facing the task of having to calculate this growth function. How does this growth function behave?

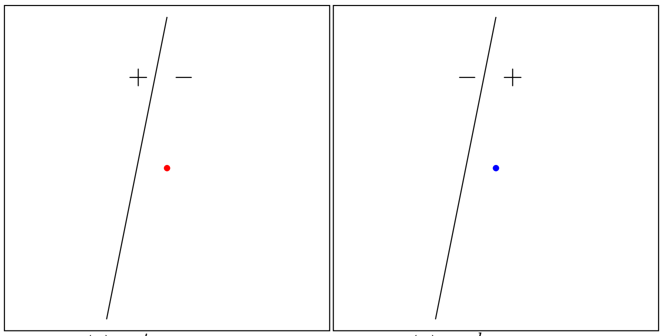
1.6.1 The number of behaviors of the class of affine separators in $\mathbb{R}^2$ for different sample sizes

Let us consider a particular example of linear separators in $\mathbb{R}^2$. Then, if we have only a single point, we can either label it positive or negative, hence we can label it in all 2^1 different ways (Figure 1.2). For each such labeling we can find a linear predictor that is consistent with it.

If I have 2 points we can have 2^2 different labelings and we can produce classifiers that will label these points in all 2^2 ways (Figure 1.3).

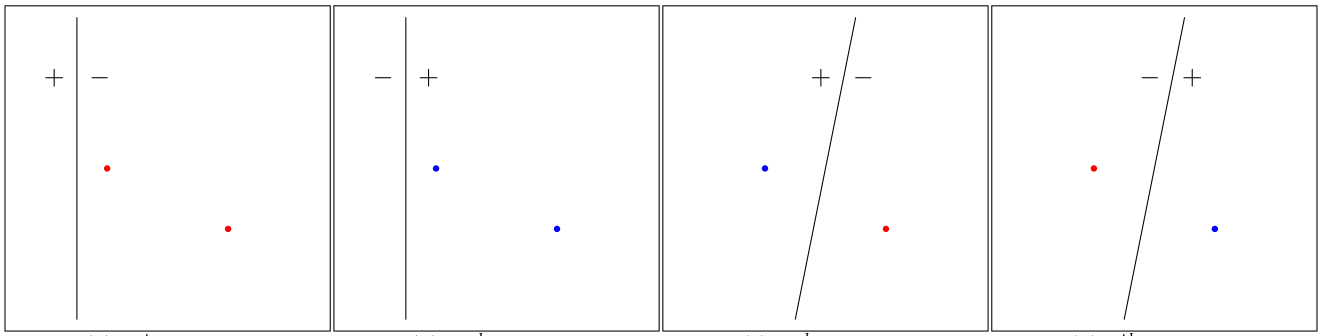
Similarly if we have 3 points we can find a linear predictor that will classify the set of three points consistently with any of the 2^3 possible labelings (Figure 1.4).

When we have 4 points, things are a little different. We can get all 4 plus 0 minus splits (2 cases). Moreover we can also label all 3 plus 1 minus splits (8 cases). However, there are some 2 plus 2 minus labeling which we cannot cover. An example is shown in Figure 1.5. Reverse of this instance is also uncoverable. So, the total number of behaviors we can get on a set of 4 points is 14. which is less than the total possible number of labelings of 4 points 2^4 which is 16. We are interested now in the largest number of points for which one can get all possible behaviors,



(a) 1st separator (b) 2nd separator

Figure 1.2: Labelings for one point in $\mathbb{R}^2$



(a) 1st separator (b) 2nd separator (c) 3rd separator (d) 4th separator

Figure 1.3: Labelings for two points in $\mathbb{R}^2$

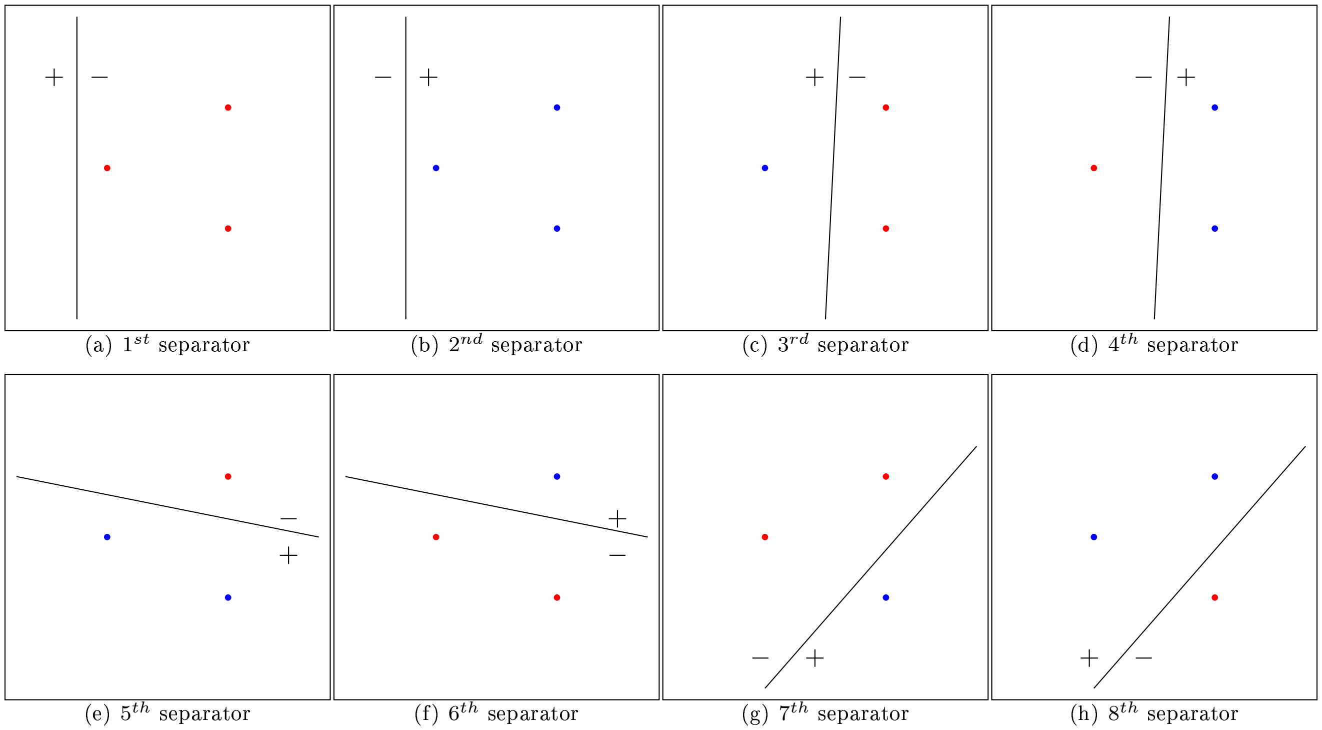


Figure 1.4: Labelings for three points in $\mathbb{R}^2$

that is, $D_{\mathcal{H}} \triangleq \sup_m \{m \mid \Pi(\mathcal{H}, m) = 2^m\}$ represents, for a given hypothesis class $\mathcal{H}$, the largest number of points m for which there is some hypothesis $h \in \mathcal{H}$ for which some configuration of those m points can be labeled in all possible ways by h. For $m \leq D_{\mathcal{H}}$ the growth function grows exponentially with m, while for $m > D_{\mathcal{H}}$ is grows roughly polynomially, in particular, roughly, as m^D.

As we just established for affine separators in $\mathbb{R}^2$, $D = 3$. By hand, we found that affine separators in the case of $m = 4$ achieved at most $2^4 - 2 = 14$ different labelings. This is much less than $m^D = 4^3 = 64$.

1.7 The Vapnik-Chervonenkis (VC) dimension and shattering

D is known as the Vapnik-Chervonenkis(VC)-dimension. The VC-dimension of a hypothesis class the maximal number of points for which you can get all possible behaviors.

$$\text{VC-dimension}(\mathcal{H}) \triangleq \text{maximal number of points } m \text{ such that } \Pi(\mathcal{H}, m) = 2^m.$$

In terms of the growth function, we can just write the VC-dimension as the maximal m such that $\Pi(\mathcal{H}, m) = 2^m$. Being able to achieve any labeling of a given set of points is also known as shattering the points. We say that if we have m points and we can get all possible behaviors on our hypothesis class for these points then these points are shattered. Hence we can write the definition

$$\text{VC-dimension}(\mathcal{H}) \triangleq \text{maximal number of points } m \text{ such that there exist points } x_1, \dots, x_m \text{ that can be shattered.}$$

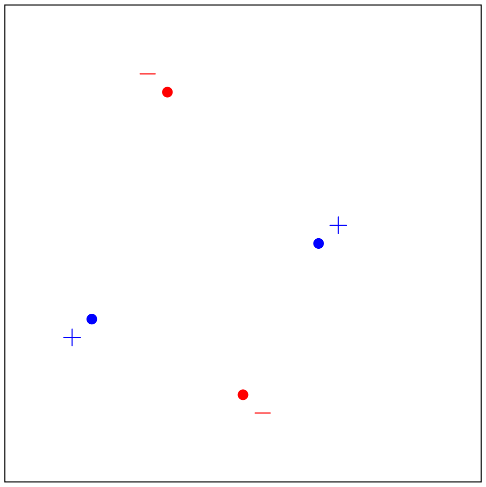


Figure 1.5: An instance of uncoverable labeling for four points in $\mathbb{R}^2$.

1.8 The shatter lemma

The following lemma connects m, VC-dimension and the growth function.

Shatter lemma (also Sauer's lemma, VC lemma) [1]:

$$\Pi(\mathcal{H}, m) \leq \sum_{i=0}^{\text{VC-Dim}(\mathcal{H})} \binom{m}{i} \leq \left(\frac{e \cdot m}{D}\right)^D \overset{(D\geq 3)}{\leq} m^D$$

where D is VC-dimension$(\mathcal{H})$.

1.9 The shatter lemma yields another deviation bound

So, now if we go back to the deviation bound and directly apply the shatter lemma we can replace log growth function with $D\log(2m)$.

Deviation bound after applying the shatter lemma, and also taking into account the performance both on the m training points and on a further m test points:
With probability at least $1-\delta$, for all hypotheses $h \in \mathcal{H}$

$$\left|R(h) - \hat{R}(h)\right| \leq \sqrt{\frac{D\log(2m) + \log\frac{4}{\delta}}{m}}. \tag{1.1}$$

1.10 Comparing this growth function/shatter-lemma-based bound to the argument-from-finite-precision-arithmetic-based bound

For 64-bit floating point number we had a different bound on error term, $\sqrt{\frac{45\cdot\#D+\log\frac{2}{\delta}}{2m}}$ where $\#D$ indicates the number of parameters in the model that are represented in floating point arithmetic. As seen, $\log(m)$ is not available in this bound. It is also possible to remove $\log(m)$ from the deviation bound (1.1) but it is difficult.

1.11 Examples: Does VC-dimension capture the effective number of parameters?

1.11.1 VC-dimension for affine separators in $\mathbb{R}^2$

Let's see if VC-dimension capture the number of effective parameters in the class. We already discussed shattering when $\mathcal{X}$ is $\mathbb{R}^2$ and $\mathcal{H}$ are linear predictors (where sign of $w^T x + b$ gives classification of point x). What was the VC-dimension of this separator on $\mathbb{R}^2$? It is possible to shatter one point (Figure 1.2), two points (Figure 1.3) and three points (Figure 1.4). However, you can prove that shattering four points on $\mathbb{R}^2$ is not possible. We saw only an instance of four points where it is not possible (Figure 1.5). So our VC-dimension of linear predictors on $\mathbb{R}^2$ is 3, maximum number of points which can be shattered. Notice that number of parameters to define our predictor is also 3 (2 parameters for w and 1 parameter for b), so the VC-dimension and the number of parameters to define our predictor are same.

1.11.2 VC-dimension for "positive segments"

Consider another case, when $\mathcal{X}$ is $\mathbb{R}$ and hypothesis clas $\mathcal{H}$ is line segments which defines positive points. Any point lies on this segments is considered as positive and any point on its complement is negative. We can shatter one point in $\mathbb{R}$ as shown in following figure:

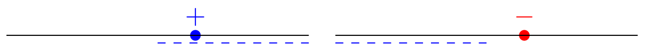


Figure 1.6: Shattering one point in $\mathbb{R}$.

We can also shatter two points as follows:

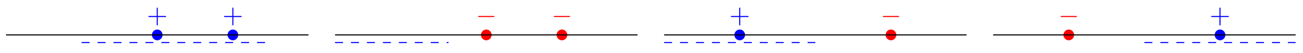


Figure 1.7: Shattering two points in $\mathbb{R}$.

However we cannot shatter 3 points for the labeling $(+, -, +)$ and $(-, +, -)$ for given order.

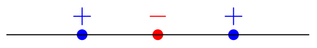


Figure 1.8: The case when three points in $\mathbb{R}$ cannot be separated by using positive line segment.

So, we cannot shatter three points. So the VC-Dimension in here is 2. The number of parameters that is needed to specify in this class is also 2, the start point of the positive segment and the end point of the positive segment. Hence, the VC-dimension matches the number of parameters that we had to specify to identify a particular hypothesis in the hypothesis class.

1.11.3 VC-dimension of “positive or negative segments”

Let’s also add negative segments to our previous predictor class on $\mathbb{R}$. In this case, we can shatter one point and two points as provided in Figure 1.6 and Figure 1.7, respectively. Moreover, we can also separate any labelings of three points. Consider following figure, where dashed lines represent positive segments and solid lines represent negative segments:

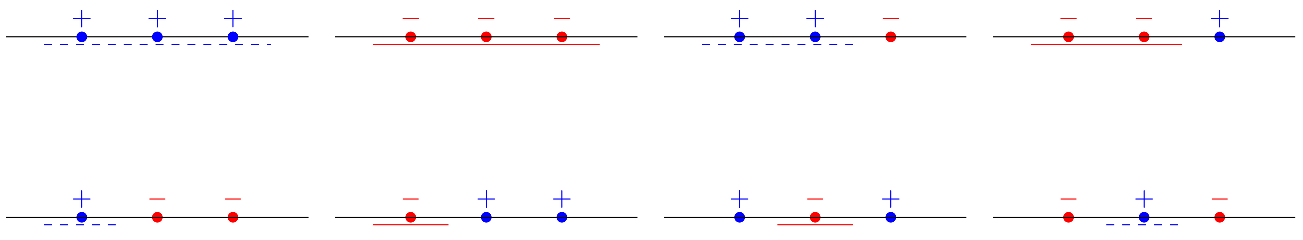


Figure 1.9: Shattering three points in $\mathbb{R}$ by using positive and negative line segments.

And this time number of points that we cannot shatter is 4, as we cannot separate the following labeling:

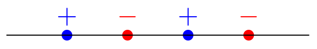
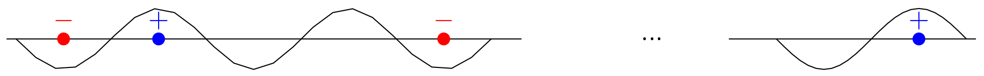


Figure 1.10: The case when four points in $\mathbb{R}$ cannot be separated by using positive and line segments.

So our VC-dimension for this example is 3. Recall that we need 2 parameters to define our segment; start and end points. However, we need one extra parameter to define its type (positive or negative) this time, so the number of effective parameters is also 3, so again VC-dimension matches it.

1.11.4 VC-dimension of “sign of sine function”

Another interesting case is shattering points with sine function. The explicit proof is not provided here, but you can shatter any number of points on $\mathbb{R}$ with a sine function as your hypothesis class.



We can shatter any number of points hence we can get any possible labellings. So the VC-dimension of that class would be infinite although it is not useful for learning.

1.11.5 VC-dimension of axis-aligned rectangles

Suppose our hypothesis class consists of aligned rectangles in $\mathbb{R}^2$. Let’s say anything inside in the rectangle is positive and anything left is negative. We can shatter 1 point in $\mathbb{R}^2$ as in Figure 1.11. Similarly we can separate any labelings of two points (Figure 1.12).

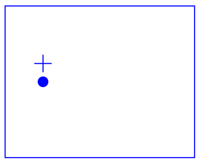


Figure 1.11: Shattering 1 point in $\mathbb{R}^2$ with aligned positive rectangles.

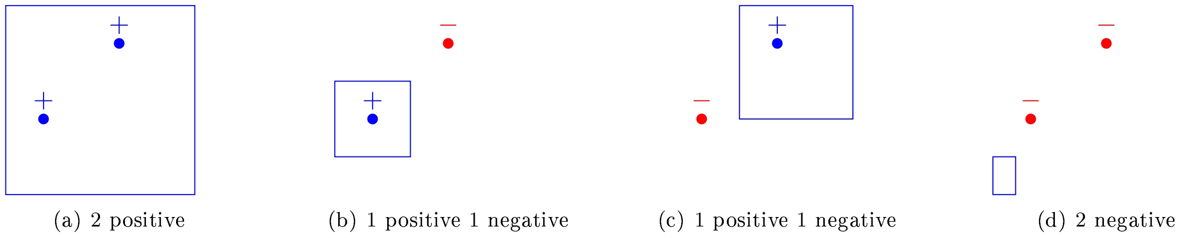


Figure 1.12: Shattering 2 points in $\mathbb{R}^2$ with aligned positive rectangles.

For 3 points, we can apply similar separations. There is a specific setting that cannot be separated and thus shattered. Consider the following:

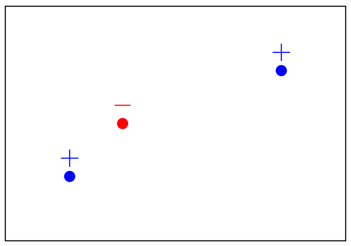


Figure 1.13: Three points in $\mathbb{R}^2$ that cannot be separated by aligned positive rectangles.

However for 3 points given as in the Figure 1.14 we can separate all 2^3 different labelings.

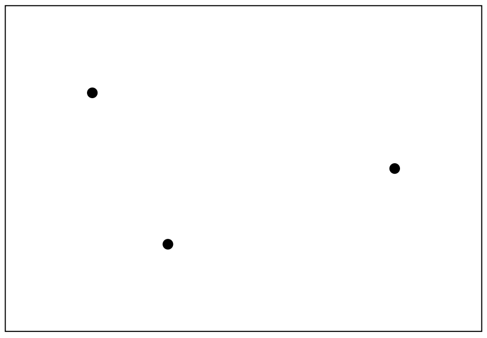


Figure 1.14: Three points in $\mathbb{R}^2$ that can be shattered by aligned positive rectangles.

What is the VC-dimension in this case? There are some settings where we cannot shatter 3 points, although there are some other where we can. Remember the VC-dimension definition: we need only some set of m points that can be shattered. So Figure 1.14 shows that VC-dimension is at least 3.

However we cannot shatter any 4 points in $\mathbb{R}^2$ with our hypothesis set. Consider the following figures:

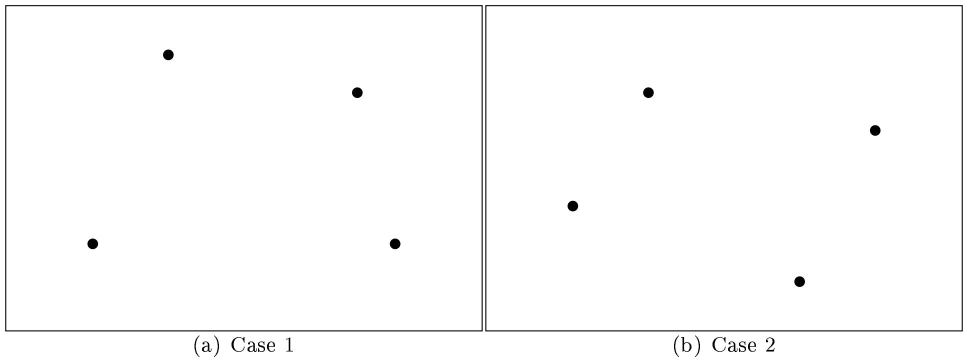


Figure 1.15: Two cases of four points in $\mathbb{R}^2$.

In Figure 1.15 Case 1, we cannot separate points if left-top and right-bottom points are positive. However, in Case 2, we can shatter these points. So we can shatter some 4 points settings, which is enough to say that VC-dimension is larger or equal than 4. We cannot shatter 5 points with aligned positive rectangles, because there will be always one point that stays inside the convex hull of remaining points. So simply setting corner points as positive and leaving one negative inside cannot be separated.

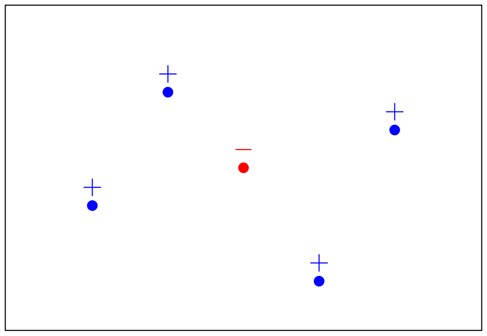


Figure 1.16: 5 points in $\mathbb{R}^2$ cannot be shattered.

So VC-dimension is 4. Note that, I also need 4 parameters to define a rectangle: left-bottom corner (2 parameters) and right-top corner (2 parameters) or left-bottom corner (2 parameters) and lengths (2 parameters).

1.11.6 VC-dimension of affine separators in $\mathbb{R}^d$

Up to here, we saw that the VC-dimension does capture the number of parameters to define our hypothesis. Some important classes for us is in a slightly higher dimension. So, in general you can show that for a d-dimensional class the VC-dimension is exactly $d + 1$. As you see, in any case it is just exactly the number of parameters.

2 Return to the case of prior distribution over hs

We discussed a refinement of the cardinality bound which was based not about having some our prior knowledge encoded with some single fixed hypothesis class. But, actually we have our prior knowledge encoded with some prior or description language.

Deviation bound in the case of a more specific prior than just specifying an hypothesis class or a hierarchy of hypothesis classes for any prior p or equaivalently for any description language:
With probability at least $1-\delta$, for all hypotheses h (no longer restricted to an hypothesis class in this, the specific prior case),

$$R(h) \leq \hat{R}(h) + \sqrt{\frac{-\log p(h) + \log \frac{2}{\delta}}{2m}} = \hat{R}(h) + \sqrt{\frac{|h|_p + \log \frac{2}{\delta}}{2m}},$$

where we require that $\sum_h p(h) \leq 1$. We could alternately have written $|h|_p$ as $\mathbf{DL}_p(h)$.

2.1 Minimum description length (MDL) learning rule

The minimum description length learning rule is a learning rule that chooses the hypothesis that just minimizes this bound ($\hat{R}(h) + \sqrt{\frac{\mathbf{DL}_p(h) + \log \frac{2}{\delta}}{2m}}$). With this rule, we just bound minimization like in Structural Risk Minimization (SRM). So, you can think MDL as a more refined version of SRM in a sense. Then, MDL learning rule is:

The minimum description length learning rule:

$$\mathbf{MDL}_p(\mathcal{S}) = \hat{h}_p = \underset{h}{\operatorname{argmin}}\ \hat{R}(h) + \sqrt{\frac{-\log p(h) + \log \frac{2}{\delta}}{2m}},$$

or equivalently

$$\mathbf{MDL}_p(\mathcal{S}) = \hat{h}_p = \underset{h}{\operatorname{argmin}}\ \hat{R}(h) + \sqrt{\frac{\mathbf{DL}_p(h) + \log \frac{2}{\delta}}{2m}},$$

where p specifies the description language.

2.2 Comparison of the MDL result to previous more limited cases with finite hypothesis class or with a complexity hierarchy of hypothesis classes

For example; having a fixed hypothesis class just corresponds having p uniform on $\mathcal{H}$. It can generalize also having a sequence of hypothesis classes for which you just have break up the probability and have it uniform on each of a bunch of hypothesis classes which are members of the sequence or hierarchy of hypothesis classes that you are considering. Minimizing the combination of the empirical risk and description length (or our prior belief) can be considered also as minimizing the empirical risk subject to some constraints on the prior belief. Let's call the hypothesis class $\mathcal{H}_k$ which is all the hs that have description length as $\mathbf{DL}_p(h)$ which are at most k. In other words, all the hs such that $p(h) \geq \frac{1}{2^k}$.

$$\begin{aligned}\mathcal{H}_k &= \{h \mid \mathbf{DL}_p(h) \le k\} \text{ (hypotheses with "sufficiently short" description length)}\\ &= \left\{h \mid p(h) \ge \frac{1}{2^k}\right\} \text{ (hypotheses with "sufficiently high" probability)}\end{aligned}$$

Question: What is the cardinality of this hypothesis class or can we say anything about the cardinality of the restricted hypothesis class $\mathcal{H}_k$?
Answer: The cardinality ($|\mathcal{H}_k|$) is at most 2^k. If we are looking at all the things that we can describe using k bits, this is basically what $\mathbf{DL}_p(h) \le k$ says using the description language. There can be at most k things rather, it can be at most 2^k things that we can describe using k bits in my description language.

The only thing is when we specify the problem in terms of minimizing the empirical risk subject to a constraint on the description length of h or alternately on the probability $p(h)$, we need to know what k is. Here in minimum description-length learning rule that is, when we were minimizing the error bound made up of an empirical risk term and an error term involving the description length. The approach says that minimize the empirical risk subject to some constraint on either the description length or the prior probability of the hypothesis being considered. We rely strongly on our bound because of this reason. This means how to balance the empirical error and your prior belief or description length. The problem is that this is the tightest guarantee that you want. In other words, minimum description length relative learning guarantee is almost the best you can do.

2.3 A relative learning guarantee for the MDL learning rule

We can calculate that with probability at least ($1-\delta$ over the sample), the expected risk of the hypothesis ($\hat{h}_p$) returned by the MDL learning rule will definitely satisfy

$$R\left(\hat{h}_p\right) \le \hat{R}\left(\hat{h}_p\right) + \sqrt{\frac{-\log p\left(\hat{h}_p\right) + \log\frac{2}{\delta}}{2m}}$$

for all hypotheses that satisfy this bound. But now, note that this hypothesis is chosen in order to minimize this bound. It represents that it is less than or equal to plugging in any other hypothesis into this bound. For example, let's plug in any hypothesis, $\tilde{h}$, into this bound. So, this is true for all $\tilde{h}$. Since relative learning guarantee is related with the definition of $\hat{h}_p$, it can be calculated with the probability at least **for all** ($1-\delta$ over the sample) .
We noted that $\hat{R}\left(\tilde{h}\right)$ is close to $R\left(\tilde{h}\right)$ and the difference between them is bounded by $\sqrt{\frac{-\log p\left(\tilde{h}\right)+\log\frac{2}{\delta}}{2m}}$ (the derivation of the bound is in appendix).

If we run the MDL learning rule with prior probability of p alternately, our description language specified by p. Then, the hypothesis we get is competitive with all other hypotheses. When we substitute $\tilde{h}$ with $h^\star$ as an inf over h, then this bound we have just been considering holds for all hypotheses (generically denoted as $\tilde{h}$). In particular, it holds for the case where our generic hypothesis $\tilde{h}$ takes on the specific value $h^\star = \underset{h}{\operatorname{arginf}}\ R(h)$, the expected-risk-minimizing hypothesis.

Note that, as long as we have a description language, we can learn anything. So we can be competitive with anything. But the question is how competitive we are. It depends on its description length that depends on the hypothesis being considered.

2.4 The main message: balance empirical error and complexity

It is not clear the trade-off between the hypothesis empirical risk and hypothesis description length or prior probability. The main message we should get from this is balancing out some notion of complexity encoded here through this p with the empirical error as the other term being balanced. Now, we are using three notions here completely interchangeably which are **prior belief, complexity, and description length**. These are all same. What is complex is something that requires lots of bits or lots of words. What simple is something that you know and you can express in a few bits. That has high value of p. Then, we want to balance the empirical error and the complexity. We can balance it by using the MDL learning rule as:

In typical learning settings, we need to balance the empirical risk of a hypothesis with the prior belief (or complexity, or description length) of the hypothesis. We can indicate that we seek such a trade-off by writing these two desired terms in bi-criterion minimization form:

$$\text{minimize } \hat{R}(h), \mathbf{DL}_p(h)$$

In terms of specific possible approaches to balancing these terms, we have talked about the MDL approach

$$\mathbf{MDL}_p(\mathcal{S}) = \hat{h}_p = \underset{h}{\operatorname{argmin}}\ \hat{R}(h) + \sqrt{\frac{\mathbf{DL}_p(h) + \log\frac{2}{\delta}}{2m}},$$

the constrained minimization approach

$$\underset{\mathbf{DL}_p(h)\leq k}{\operatorname{argmin}}\ \hat{R}(h),$$

and now we consider the parametrized trade-off form

$$\underset{h}{\operatorname{argmin}}\ \hat{R}(h) + \lambda \cdot \mathbf{DL}_p(h).$$

Here, in the MDL bound minimization, the parametrized trade-off form $\underset{h}{\operatorname{argmin}}\ R(h) + \lambda \cdot \mathbf{DL}_p(h)$

We want a balance between $\hat{R}(h)$ and $\mathbf{DL}_p(h)$. What we want to do is minimizing two objectives : empirical error and complexity (the description length, the prior belief). We don't know exactly, how we balance them. We probably set that by using cross validation to determine the value of the trade-off parameter λ.

2.5 Brief reminder about the connection between prior belief and description length

The connection between description length and prior belief is just consider $\mathbf{DL}_p(h)$. It is by definition $-\log p(h)$ which is one way to view all those equations. Another way is, if we have some description language.

Question: What is a description language? **Answer:** It is something that tells us how to interpret a bit stream as representing a hypothesis.

Our objective is designing a learning algorithm. So, how do we choose a prior belief p. We can think it as a description language as a prior belief. Currently the size of data sets that we are talking about are pretty huge. If we are limited in dataset size, we might want to improve on p to be more specific to our domain. So, we think of a language that is very specific to our domain. But, the main difficulty here is optimizing $\mathbf{DL}_p(h)$ which is impossible. We can not optimize this over computer programs. It is just intractable. To remind that the optimization problems that we are interested are minimizing the MDL bound, solving a complexity constrained empirical risk minimization problem, or solving a parametrized trade-off problem. All of these problems are differently solved bicriterion optimization problems.

The whole goal of learning is finding description languages and finding notions of complexity. There are hypotheses that have low complexity, good performance and they are also easy to optimize.

3 Convexity

3.1 Different representations of the bicriterion problem and some things you can't learn (i.e. nonconvex things)

Let's consider the minimization the empirical error problem subject to some constraints as:

$$\underset{h\in\mathcal{H}}{\text{minimize}}\ \ \hat{R}(h),$$

where $\mathcal{H}$ for all of the hypotheses that have level k. and description-length is up to some k.. Let $\mathcal{H}$ is any hypothesis class makes a convex optimization problem which is the most simple case where $\mathcal{H}$ is convex in other words we have linear classifiers. So, we will take $\mathcal{X}$ to be in $\mathbb{R}^d$ and $\mathcal{H}$ and all the linear functions are in the form $w^T x + b$.

$$\mathcal{X} = \mathbb{R}^d$$
$$\mathcal{H} = \left\{x \mapsto w^T x + b \mid w \in \mathbb{R}^d,\ b \in \mathbb{R}\right\}$$

At this problem, optimization over w and b represents the empirical loss as one over m sum over loss of $w^T x_i + b$ relative to y_i.

$$\underset{w\in\mathbb{R}^d, b\in\mathbb{R}}{\text{minimize}}\ \frac{1}{m}\sum_{i=1}^{m}\mathbf{loss}_{01}\left(w^T x_i + b, y_i\right)$$

or

$$\begin{gathered}\underset{w\in\mathbb{R}^d, b\in\mathbb{R}, z\in\mathbb{R}^d}{\text{minimize}}\ \frac{1}{m}\sum_{i=1}^{m}\mathbf{loss}_{01}\left(z_i, y_i\right)\\ \text{subject to } z_i = w^T x_i + b\end{gathered}$$

3.2 The consequences of 01 loss

01 loss is a binary thresholding to the loss function. This class ($w^T x + b$) returns a real number ,but we care about its sign. If y and z have same sign in other words, y times z is greater than 0, it means that we are not going to suffer loss. The loss is 0. So, we say that z ,the prediction, has the correct sign. If they don't have same signs, then the loss is 1.

$$\mathbf{loss}_{01}\left(z_i, y_i\right) = \begin{cases}0 & \text{if } y_i z_i > 0\\ 1 & \text{if } y_i z_i \leq 0\end{cases},$$

where $z_i = h_{w,b}\left(x_i\right) = w^T x_i + b$ is the output of the predictor for x_i.

Note: If $z = 0$, it incurs a loss of 1, because that prediction can never match the true label y.

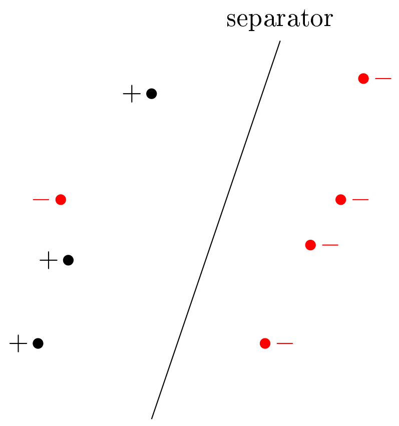


Figure 3.1: Affine separator performance in the case of 01 loss. 01 loss is not good at measuring the loss magnitude.

3.2.1 The 01 loss is not convex

Our objective is finding the linear predictor that minimizes the number of things on the wrong side in other words minimizes the number of mistakes. It turns out that this problem is a NP-hard. Even if the hypothesis class is convex , the problem becomes NP-hard. **Question:** What is not convex? Why can't we think of this as a convex optimization problem? **Answer:** The loss function which is not convex.

3.3 A convex alternative: the logistic loss

We need to do work with solvable problems. So, we want the optimization problem to be at least convex. It means we need two both the hypothesis class and also the loss to be convex. We can change our 01 loss to a convex loss. One classical example is logistic loss. So, let us call it $\mathbf{loss}_g$, such that

$$\text{Logistic loss:}$$
$$\mathbf{loss}_g\left(z,y\right) \triangleq \log\left(1+e^{-yz}\right) = \log\left(1+e^{-y\left(w^Tx+b\right)}\right)$$

where the logistic loss is a convex function as in Figure 3.3.

3.4 The optimization problem with logistic loss

Here is our first convex optimization problem:

$$\underset{w\in\mathbb{R}^d, b\in\mathbb{R}}{\text{minimize}}\ \frac{1}{m}\sum_{i=1}^{m}\mathbf{loss}_g\left(w^Tx_i+b, y_i\right).$$

Question: Does it solve the problem that we had before which do the values of w and b that we get from minimizing the logistic loss that also minimizes the 01 loss? **Answer:** Not really.
Minimizing the logistic loss gives us a convex optimization problem that roughly does what we want. But, changing from 01 loss to this logistic loss changes the solution. Minimizing the logistic loss is not same as minimizing the 01 loss. We can have cases in which we get different answers. Let's just see such an example.

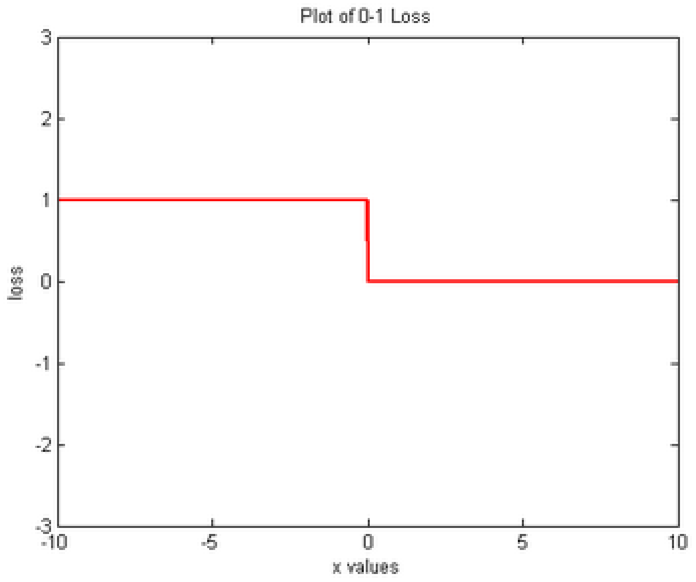


Figure 3.2: Chart of 01 loss $\mathbf{loss}_{01}(z, 1)$ as a function of the prediction z.

3.5 An example showing the different behavior of 01 loss and logistic loss

Picture of separation with 0-1 loss, separation with logistic loss

3.5.1 Considering the sample average logistic loss of a midpoint decision boundary

If my threshold boundary is at the midpoint, then that means two large groups of points that are correctly classified. So, these examples are large positive group and these are examples of the large negative group almost don't participate in the game. If we move the threshold, the contribution they make to the average logistic loss on the sample which is really not going to affect the sample average logistic loss at all. That is from $y_i z_i > 0$ to $y_i z_i \leq 0$. So, I have right now points with $y_i z_i$ on the correct side of 0 on the logistic loss plot and E over here with $y_i z_i$ quite far from the correct side of 0 on the logistic loss plot.

3.5.2 Considering the sample average logistic loss of a decision boundary shifted away from the midpoint closer to the significantly misclassified positive example

When we slightly move this decision boundary, the large and correctly classified clusters of negative points and positive points are not going to be affected much, because all of these examples are over here comfortably on the correct side of 0 in their values of $y_i z_i$. If we move our decision boundary on the correct side of 0 with their values of $y_i z_i$. now they are actually on the incorrect side of the margin that is, of the boundary threshold. So, they actually moved over here. Their z times y is actually going to become negative. So, they moved slightly over from the correct side to the incorrect side with their values of $y_i z_i$.

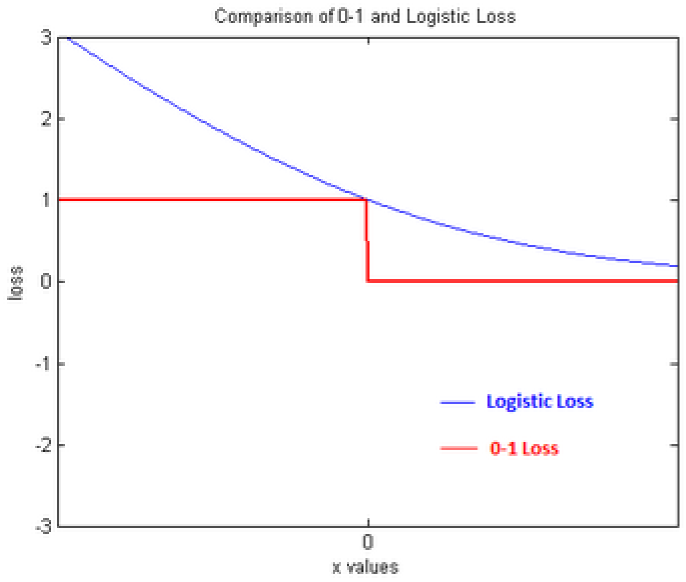


Figure 3.3: Chart of logistic loss $\mathbf{loss}_g(z, 1)$ as a function of the prediction z.

3.5.3 The gradient of the loss matters

We have to look at essentially the gradient at the plotted location of $y_i z_i$ for the incorrectly classified points, because we are going to affect them by the amount we are going to change them is going to be proportional. The point is the logistic function focus on what the local gradient is. We pay some price proportional to the gradient. The logistic function that correctly classify points almost don't affect the cost the sample average logistic loss at all.

3.5.4 Logistic loss cares about outliers

The logistic loss might not actually optimize the 01 loss, because we said that optimizing this problem with respect to 01 loss is NP-hard so we can not expect a convex optimization problem to solve it. But, the point is when we change the loss function, we did change ourfunction. The logistic loss function really encodes things differently than the 01 loss. The main thing here is logistic loss is much more sensitive to outliers. At 01 loss, once something was misclassified, we didn't really care how badly it was misclassified. However, logistic loss is a convex loss function and has a linear behavior for the strongly misclassified points' values of $y_i z_i$. So, even if something is misclassified, we still care about it and try to move it as close as possible to the decision boundary.

3.5.5 Logistic loss can minimize the 01 loss when the data are linearly separable

Example will be provided here.

3.6 Three reasons to use the logistic loss

We can think of three reasons to minimize the logistic loss function: 1) what we care about is better modeled by the logistic loss for example: we really care about outliers where the points are strongly misclassified.We are not interested with that points. They are a bit misclassified but we do not want them to be a lot misclassified. We really care about how much they are misclassified. Note that the 01 loss is not caring about outliers. Second reason is, we want to get a convex loss and thirdly, the logistic loss is L-Lipschitz with $L = 1$, and thus behaves well in terms of scale-sensitive complexity.

Three reasons to minimize the logistic loss:

1. You believe that logistic loss is actually correct, as would be the case when your classes are Gaussian.
2. The logistic loss is convex thus our loss minimization becomes a convex optimization problem.
3. The logistic loss is L-Lipschitz with $L = 1$, and thus behaves well in terms of scale-sensitive complexity (as does the hinge loss).

References

[1] N Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145 – 147, 1972.

Appendix

To get a learning bound in the setting where we specify a prior, we start as follows.

With probability at least $1 - \delta$ over the sample $\mathcal{S}$

$$R\left(\hat{h}_p\right) \leq \hat{R}\left(\hat{h}_p\right) + \sqrt{\frac{-\log p\left(\hat{h}_p\right) + \log \frac{2}{\delta}}{2m}},$$

where $\hat{h}_p$ is the MDL-learning-rule-selected hypothesis $\hat{h}_p = \underset{h}{\operatorname{arginf}} \hat{R}(h) + \sqrt{\frac{-\log p(h) + \log \frac{2}{\delta}}{2m}}$. Since $\hat{h}_p$ is the MDL-learning-rule-selected hypothesis, we also know that the error bound that MDL minimizes over, when evaluated with any other hypothesis, say $\tilde{h}$, will be greater than it is when evaluated with $\hat{h}_p$. That is,

$$\hat{R}\left(\hat{h}_p\right) + \sqrt{\frac{-\log p\left(\hat{h}_p\right) + \log \frac{2}{\delta}}{2m}} \leq \hat{R}\left(\tilde{h}\right) + \sqrt{\frac{-\log p\left(\tilde{h}\right) + \log \frac{2}{\delta}}{2m}}.$$

Thus, we can combine this with our initial result to obtain

$$R\left(\hat{h}_p\right) \leq \hat{R}\left(\tilde{h}\right) + \sqrt{\frac{-\log p\left(\tilde{h}\right) + \log \frac{2}{\delta}}{2m}}.$$

We now note that

$$\hat{R}\left(\tilde{h}\right) \leq R\left(\tilde{h}\right) + \sqrt{\frac{-\log p\left(\tilde{h}\right) + \log \frac{2}{\delta}}{2m}},$$

which we can combine with the previous result to get

$$\begin{aligned} R\left(\hat{h}_p\right) &\leq \hat{R}\left(\tilde{h}\right) + \sqrt{\frac{-\log p\left(\tilde{h}\right) + \log\frac{2}{\delta}}{2m}} \\ &\leq R\left(\tilde{h}\right) + \sqrt{\frac{-\log p\left(\tilde{h}\right) + \log\frac{2}{\delta}}{2m}} + \sqrt{\frac{-\log p\left(\tilde{h}\right) + \log\frac{2}{\delta}}{2m}}, \end{aligned}$$

or, more simply

$$R\left(\hat{h}_p\right) \leq R\left(\tilde{h}\right) + 2\sqrt{\frac{-\log p\left(\tilde{h}\right) + \log\frac{2}{\delta}}{2m}}.$$

In particular, this result holds for the hypothesis $h^\star$ that minimizes the expected risk $h^\star = \underset{h}{\operatorname{arginf}} R\left(h\right)$. Plugging in $h^\star$, we thus find

$$R\left(\hat{h}_p\right) \leq R\left(h^\star\right) + 2\sqrt{\frac{-\log p\left(h^\star\right) + \log\frac{2}{\delta}}{2m}}.$$

We could alternately have begun with $\mathbf{DL}_p\left(\cdot\right)$, in which case we would have ended up with

$$R\left(\hat{h}_p\right) \leq R\left(h^\star\right) + 2\sqrt{\frac{\mathbf{DL}_p\left(h^\star\right) + \log\frac{2}{\delta}}{2m}}$$