5
Basics of generalization theory

Recall from Chapter 1 the language of Empirical Risk Minimization. A datapoint x (for classification this is actually a (vector, label) pair) is drawn from from a distribution $\mathcal{D}$ and S denotes the set of training samples. The loss of a hypothesis h on datapoint x is $\ell(x, h)$. Since hypothesis in deep learning is given by a parameter vector w we may also represent this as $\ell(x, w)$. Throughout the chapter we will assume loss takes values in $[0, 1]$. In generalization theory we are interested in mathematically understanding the relationship between the test loss and the training loss (respectively):

$$L_{\mathcal{D}}(h) = \mathop{\mathbb{E}}_{x \in \mathcal{D}} [\ell(x, h)] \quad \text{and} \quad \widehat{L}_S(h) = \mathop{\mathbb{E}}_{x \in S} [\ell(x, h)] \,. \tag{5.1}$$

(Here $\widehat{\cdot}$ refers to "empirical." The training is considered a success if $L_S(h)$ is small and the *generalization error* $\Delta_S(h) = L_{\mathcal{D}}(h) - \widehat{L}_S(h)$ is small too.

Generalization theory gives way to derive estimate of the number of training samples sufficient to guarantee low generalization error. The classic ideas described in this chapter give very loose (i.e., trivial) estimates for deep learning. We survey attempts to provide tighter estimates. Note that an experimenter does not need generalization theory, since they can estimate generalization error by keeping an held-out dataset. Thus the purpose of generalization theory is to understand how choice of architecture, algorithm, and training dataset promotes good generalization.

Generalization theory takes inspiration from an old philosophical principle called *Occam's razor*[1]: given a choice between a simpler theory of science and a more convoluted theory, both of which explain some empirical observations, we should trust the simpler one. For instance, Copernicus's heliocentric theory of the solar system gained favor in science because it explained known facts more simply than the ancient Aristotelian theory. While this makes intuitive sense, Occam's razor is a bit vague and hand-wavy. What makes a theory "simpler" or "better"?

[1] While it is attributed to 14th century friar William Ockham, similar principles had been articulated much earlier. Aristotle's version: *"the more limited, if adequate, is always preferable")*

5.1 Occam's razor formalized for ML

The following is the mapping from the above intuitive notions to notions in ML. (For simplicity we focus only on supervised learning here, and consider other settings in later chapters.)

Observations/evidence	$\leftrightarrow$	Training dataset S
theory	$\leftrightarrow$	hypothesis h
All possible theories	$\leftrightarrow$	hypothesis class $\mathcal{H}$
Finding theory to fit observations	$\leftrightarrow$	Minimize training loss to find $h \in \mathcal{H}$
Theory is good (good predictions in new settings)	$\leftrightarrow$	h has low test loss
Simpler theory	$\leftrightarrow$	h has shorter description

To give an example, $\mathcal{H}$ could be all possible deep nets of a certain architecture and size. The notion of "shorter description"will be formalized in a variety of ways using a *complexity measure* for the class $\mathcal{H}$, denoted $\mathcal{C}(\mathcal{H})$, and use it to upper bound the generalization error.

Let S be a sample of m datapoints. Empirical Risk Minimization (ERM) paradigm (see Chapter 1) involves finding $\widehat{h} = \arg\min \widehat{L_S}(h)$. Of course, in deep learning we may not find the absolute optimum h but in practice the training loss becomes very small and near-zero. Intuitively, if generalization error is large then the hypothesis's performance on training sample S does not accurately reflect the performance on the full distribution of examples, and we say it *overfitted* to the sample S.

The typical upper bound on generalization error [2] shows that with probability at least $1 - \delta$ over the choice of training data, the following holds:

$$\Delta_S(h) \leq \sqrt{\frac{\mathcal{C}(\mathcal{H}) + O(\log(1/\delta))}{m}} \tag{5.2}$$

[2] This should be interpreted as rough format of a typical generalization bound; not an actual bound! In general this chapter focuses on clear exposition of the basic ideas, while being a bit sloppy with constants.

Thus to drive the generalization error down it suffices to make m significantly larger than the "Complexity Measure". Hence classes with lower complexity require fewer training samples, in line with Occam's intuition.

But this is a good place to already note the main challenge posed by (5.2): the generalization error is nontrivial only if $\mathcal{C}(\mathcal{H}) < m$. But today's deep nets are often overparametrized, so the number of trainable parameters may exceed m. Thus the complexity measure of the deep net is smaller than the number of parameters.

5.1.1 Motivation for generalization theory

Generalization bounds seek to estimate the generalization error using properties of the trained model h and the training dataset S. Students can wonder if the bound is any use if the experimenter has already decided on the architecture, training algorithm etc. Indeed, if so,, the experimenter can proceed with training and use a held out dataset to estimate the generalization error.

The hope in developing generalization theory is that it provides insights into how to design architectures and algorithms in the first place so that they result in "low complexity" in the trained net, causing it to generalize well. Clearly, more principled understanding along such lines would be nice.

5.1.2 Warmup: Classical polynomial interpolation

Suppose we are given n points $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$ chosen according to the following probabilistic process: $x^{(i)}$ is chosen from uniform distribution on $[0, 1]$ and $y^{(i)} = p(x^{(i)}) + \eta$ where $p()$ is an unknown degree d polynomial and η is a sample from the noise distribution $\mathcal{N}(0, \sigma^2)$. Since $\mathbb{E}[\eta] = 0$ and $\mathbb{E}[\eta^2] = \sigma^2$ the obvious way to find p is to minimize the least square fit to find the polynomial's coefficients $\theta_0, \theta_1, \ldots, \theta_d$ that minimizes the squared error of the fit to the data:

$$\ell(\vec{\theta}) = \sum_{i=1}^{n} (y^{(i)} - \sum_{j=0} \theta_j (x^{(i)}))^2.$$

This is implicitly doing linear regression using a new data representation, whereby the point $x \in \Re$ is represented using the vector $(1, x, x^2, \ldots, x^d)$.

But what if we don't know the correct degree d and try to fit a degree N polynomial where $N \gg d$? This is the so-called *overparametrized* setting. Under what conditions would minimizing the above loss give us roughly the same polynomial as $p()$? A practical idea —noting the fact that the above loss is $n\sigma^2$ even for the ground truth polynomial $p()$—is to add for some large-ish $\lambda > 0$ the regularizer $\lambda \|\theta\|_2^2$ to the above loss. This signals to gradient descent that it is unimportant to reduce the least squares loss all the way to zero, and instead it should find solutions θ's of low norm. More generally one could use other measures of "complexity" than square of the Euclidean norm.

This example is intuitive and can be analysed more rigorously but requires the theory of orthonormal polynomials with respect to natural distributions on $[0, 1]$.

5.2 *Union Bound technique*

Many analyses in generalization theory use the *union bound* in elementary probability: every set of events $A_1, A_2, \ldots$ satisfy $\Pr[\cup_i A_i] \leq \sum_i \Pr[A_i]$.

The first example illustrates this in an almost trivial setting. But we shall soon see the same idea at the heart of other generalization bounds[3], albeit often hidden inside the proof. The bound shows that if a hypothesis class contains at most N distinct hypotheses, then $\log N$ (i.e., close to the number of bits needed to represent the index of the hypotheses in this class) is the effective complexity measure in (5.2).

[3] The union bound is also referred to as **uniform convergence framework** in many books. Often the hypothesis class is infinite but the proof discretizes it, as in Theorem 5.2.7.

Theorem 5.2.1 (Finite Hypothesis Class). *If the loss function takes values in $[0,1]$ and hypothesis class $\mathcal{H}$ contains N distinct hypotheses then with probability at least $1-\delta$, every $h \in \mathcal{H}$ satisfies*

$$\Delta_S(h) \leq 2\sqrt{(\log N + \log(1/\delta))/m}.$$

Proof. For any *fixed* hypothesis h imagine drawing a training sample of size m. Then $\widehat{L_S}(h)$ is an average of i.i.d. variables and its expectation is $L_{\mathcal{D}}(h)$. Let $\Delta_S(h)$ denote $L_{\mathcal{D}}(h) - \widehat{L_S}(h)$. Note that $E_S[\Delta_S(h)] = 0$. Thus, Hoeffding's inequality implies that for any fixed h, $P(\Delta_S(h) > \epsilon) \leq \exp(-2m\epsilon^2)$. Then, by the union bound, $\Pr_S(\exists h : \Delta_S(h) > \epsilon) \leq N \cdot \exp(-2m\epsilon^2)$. Setting this to δ and solving for ϵ yields the result. □

At first sight the union bound may appear useless for deep nets. If the net has k real-valued parameters, the set of possible hypotheses —even having fixed the architecture—is $\mathbb{R}^k$, an uncountable set! [4] Generalization bounds often find a way around this hurdle via some form of *discretization* argument. We give the simplest example first, and then a more nontrivial example in Section 5.2.1.

[4] Another intriguing possibility, which hasn't yet been explored as much, is to compute a finite upper bound on the types of solutions gradient descent could find. For instance we saw in Chapter 4 that gradient descent finds stationary point (i.e., where $\nabla = 0$) of the training loss. One could conceivably bound the number of such stationary points. Such investigation hasn't yet worked out because current nets are so overparametrized (i.e., number of parameters far exceeding the number of training data points) that the set of such solution points in the landscape is also in general a continuous set (i.e., uncountable).

Example 5.2.2 (Quantization). *Consider a trained network f with k real-valued parameters. Suppose we can replace all parameter values by 8-bit integers to get a new network g, such that g still performs almost as well as f on the training set.* [5] *The class of all possible networks with k parameters with each taking 8-bit values is still vast but finite, namely at most 2^{8k}. Then generalization can be proved in principle using a simple union bound as in Theorem 5.2.1.*

[5] Quantization is practically important when one tries to fit large models inside limited GPU memory. Quantization methods have helped shrink the cost of serving AI models.

5.2.1 *Vector quantization via ϵ-cover argument*

Instead of naively discretizing each coordinate, this method approximates the continuous set of all possible deep nets via a finite number

of "fairly distinct" deep nets. It involves a famous mathematical notion: *ϵ-cover*. (A related notion is ϵ-net, but it is not used here.)

Suppose we assume that the ℓ_2-norm of the parameter vectors is at most 1, meaning the set of all deep nets has been identified with Ball$(0,1)$. (Here Ball(w,r) refers to set of all points in $\mathbb{R}^k$ within distance r of w.)

Now we need loss to be locally Lipschitz in the parameters: there are constants ρ, γ such that for every datapoint x, if the parameter vectors w_1, w_2 satisfy $\|w_1 - w_2\|_2 \leq \rho$ then $\|\ell(x, w_1) - \ell(x, w_2)\| \leq \gamma \|w_1 - w_2\|_2$. [6] It makes intuitive sense such a ρ must exist for every $\gamma > 0$ since as we let $\rho \to 0$ the two nets become equal. But to obtain practical bounds, one would need ρ, γ to not be too small.

[6] Actually the argument only requires Lipschitz-ness of the average loss on the training set, not loss on single data points.

Problem 5.2.3. *Compute Lipschitz constant of the ℓ_2 regression loss: the loss on datapoint (x, y) is $(w \cdot x - y)^2$.*

Problem 5.2.4. *Compute Lipschitz constant of ℓ_2 loss for a two layer deep net with ReLU gates (zero bias) on the middle layer. Assume the two parameter vectors are infinitesimally close.*

Definition 5.2.5 (ρ-cover)**.** *A set of points $w_1, w_2, \ldots \in$ Ball$(0,1)$ is a ρ-*cover *if for every $w \in$ Ball$(0,1)$ there is some w_i such that $w \in$ Ball(w_i, ρ).*

Lemma 5.2.6 (Existence of ρ-cover)**.** *There exists a ρ-cover of size at most $((2+\rho)/2\rho)^k$.*

Proof. The proof is simple but clever. Let us pick w_1 arbitrarily in Ball$(0,1)$. For $i = 2, 3, \ldots$ do the following: arbitrarily pick any point in Ball$(0,1)$ outside $\cup_{j \leq i}$Ball(w_j, ρ) and designate it as w_{i+1}.

A priori it is unclear if this process will ever terminate. We now show it does after at most $(2/\rho)^k$ steps. To see this, it suffices to note that Ball$(w_i, \rho/2) \cap$ Ball$(w_j, \rho/2) = \emptyset$ for all $i < j$. (Because if not, then $w_j \in$ Ball(w_i, ρ), which means that w_j could not have been picked during the above process.) Thus we conclude that the process must have stopped after at most

$$\text{volume}(\text{Ball}(0, 1+\rho/2))/\text{volume}(\text{Ball}(0, \rho/2))$$

iterations[7], which is at most $((2+\rho)/2\rho)^k$ since ball volume in $\mathbb{R}^k$ scales as the kth power of the radius.

[7] The reason for $1+\rho/2$ in the numerator is that if a w_i lies at the surface of Ball$(0,1)$ then the ball of radius $\rho/2$ around it lies in the ball of radius $1+\rho/2$ around the origin

Finally, the sequence of w_i's at the end must be a ρ-cover because the process stops only when no point can be found outside $\cup_j$Ball(w_j, ρ). □

Theorem 5.2.7 (Generalization bound for normed spaces assuming Lipschitz losses)**.** *If (i) hypotheses are unit vectors in $\mathbb{R}^k$ and (ii) every two hypotheses h_1, h_2 with $\|h_1 - h_2\|_2 \leq \rho$ differ in terms of loss on every*

datapoint by at most γ then

$$\Delta_S(h) \leq \gamma + 2\sqrt{k \log(2/\rho)/m}.$$

Proof. Apply the union bound on the ρ-cover. Every other net can have loss at most γ higher than nets in the ρ-cover. □

This bound is often vacuous for deep neural networks, where the number of parameters k can be much larger than the number of training samples m. Thus parameter-count alone fails to capture the 'effective size' of models found by modern training methods. This suggests our notion of model complexity/simplicity needs to be more sophisticated, incorporating not just the size of the hypothesis space, but also the properties of the data itself. This motivates the topic of the next section.

5.3 *Data dependent complexity measures*

Thus far we considered complexity measures for hypothesis classes as a way to quantify their "complicatedness." : the size of the hypothesis class (assuming it is finite) and the size of a γ-cover in it. In pratice, the bounds on sample complexity derived using these methods are still often loose.

One thing to note is that these simple bounds hold for every data distribution $\mathcal{D}$. In practice, it seems clear that deep nets —or any learning method—works by being able to exploit properties of the input distribution (e.g., convolutional structure exploits the fact that all subpatches of images can be processed very similarly). Thus one should try to prove some measure of complicatedness that depends on the data distribution.

5.3.1 *Rademacher Complexity*

Rademacher complexity is a complexity measure that depends on data distribution. As usual our description assumes loss function takes values in $[0, 1]$.

The definition concerns the following thought experiment. Recall that the distribution $\mathcal{D}$ is on labeled datapoints (x, y). For simplicity we denote the labeled datapoint as z.

Now *Rademacher Complexity* [8] of hypothesis class $\mathcal{H}$ on a distribution $\mathcal{D}$ is defined as follows where $l(z, h)$ is loss of hypothesis h on labeled datapoint z.

$$\mathcal{R}_{m,D}(\mathcal{H}) = \mathop{\mathbb{E}}_{S_1,S_2}\left[\frac{1}{2m}\sup_{h\in\mathcal{H}}\left|\sum_{z\in S_1} l(z,h) - \sum_{z\in S_2} l(z,h)\right|\right], \qquad (5.3)$$

[8] Standard accounts of this often confuse students, or at least falsely impress them with a complicated proof of Thm 5.3.1 that hides the simple idea below. Our definition is simplified a bit: in the standard definition, one picks a sign ± 1 (or *Rademacher* random variables) for each of the $2m$ datapionts and looks at loss weighted by this sign. The value yielded by our definition is within $\pm O(1/\sqrt{m})$ of the one in the standard definition.

where the expectation is over S_1, S_2 are two iid sample sets (i.e., multisets) of size m each from the data distribution $\mathcal{D}$. Note that this definition involves the thought experiment of picking S_1, S_2 and picking a classifier whose training error on these is as different as possible. The following theorem relates this to generalization error of the trained hypothesis.

Theorem 5.3.1. *If h is the hypothesis trained via ERM using a training set S_2 of size m, then the probability (over S_2) is $> 1 - \delta$, that*

$$\Delta_{S_2}(h) \leq 2\mathcal{R}_{m,D}(\mathcal{H}) + O((\log(1/\delta))/\sqrt{m}).$$

Proof. The generalization error $\Delta_{S_2}(h) = L_{\mathcal{D}}(h) - \widehat{L_{S_2}}(h)$, and ERM guarantees an h that maximizes this. Imagine we pick another m iid samples from distribution $\mathcal{D}$ to get another (multi)set S_1. Then with probability at least $1 - \delta$ the loss on these samples closely approximates $L_{\mathcal{D}}(h)$:

$$\Delta_{S_2}(h) \leq \widehat{L_{S_1}}(h) - \widehat{L_{S_2}}(h) + O((\log(1/\delta))/\sqrt{m}).$$

Now we notice that S_1, S_2 thus drawn are exactly like the sets drawn in the thought experiment of (5.3) [9] (5.3) and the maximizer h for this expression defined $\mathcal{R}_{m,D}$. So the right hand side is at most

$$2\mathcal{R}_{m,D}(\mathcal{H}) + O((\log(1/\delta))/\sqrt{m}).$$

□

[9] Here hypothesis h is allowed to depend on S_2 but not S_1. In the thought experiment the supremum is over h that can depend on both. This only helps the inequality, since the latter h can achieve a larger value. Note that the factor 2 is because of scaling of $2m$ in (5.3).

Problem 5.3.2. *Show that the Rademacher complexity of the set of linear classifiers (unit norm vectors $U = \{w | w \in \mathbb{R}^d, \|w\|_2 = 1\}$), on a given sample $S = \{x_1, x_2, \cdots, x_m\}$ (each $x_i \in \mathbb{R}^d$) is $\leq \max_{i \in [m]} \|x_i\|_2 / \sqrt{m}$.*

Problem 5.3.3. *Consider the kernel classifier of the form $h(x) = z^\top G^{-1} y$ we studied in Section 2.2 where G is the $n \times n$ kernel matrix, y is the labels and z is the column vector whose i-th coordinate is $K(x, x_i)$. Show that the Rademacher complexity upper is $\sqrt{2y^\top Gy \cdot \mathrm{Tr}(G)}/n$. (We will use this result in Chapter 9 to prove certain over-parameterized student nets can learn simple two-layer teacher nets.)*

5.3.2 *Alternative Interpretation: Ability to correlate with random labels*

Teachers explain Rademacher complexity more intuitively as *ability of classifiers in $\mathcal{H}$ to correlate with random labelings of the data.* This is best understood for binary classification (i.e., labels are 0/1), and the loss function is also binary (loss 0 for correct label and 1 for incorrect label). Now consider the following experiment: Pick S_1, S_2 as in the definition of Rademacher Complexity, and imagine flipping the

labels of S_1. Now average loss on S_2 is $1 - \widehat{L_{S_2}}(h)$. Thus selecting h to maximise the right hand side of (5.3) is like finding an h that has low loss on $S_1 \cup S_2$ where the labels have been flipped on S_1. In other words, h is able to achieve low loss on datasets where labels were flipped for some randomly chosen set of half of the training points.

When the loss is not binary a similar statement still holds qualitatively.

5.4 *Understanding limitations of the union-bound approach*

The phenomenon captured in the union bound approach and related approaches is also refered to as *uniform convergence*. If we have identified a finite set $\mathcal{H}$ of hypotheses and sample S of datapoints is large enough then with probability is at least $1 - \delta$ over choice of S that

$$\|L_{\mathcal{D}}(h) - \widehat{L}_S(h)\| \leq \epsilon \quad \forall h \in \mathcal{H}. \tag{5.4}$$

Here the important point to note is that a fixed sample set S can be used for good estimate of generalization error for *every* classifier h in the class[10]. Of course, using γ-cover this kind of conclusion can be shown also for classes $\mathcal{H}$ that are a continuous set, e.g. hypotheses with bounded ℓ_2 norm. Now we describe a nice and simple example by Nagarajan and Kolter [11] that pinpoints why this framework may be tricky to apply in modern settings, especially deep learning. The point is that the hypothesis class of interest is implicitly defined via the optimization algorithm (say, gradient descent), and this class may not allow a clean analysis via a union bound.

[10] Note that most of these classifiers may have terrible loss on S; the union bound only guarantees that the generalization error is small.

[11] V Nagarajan and Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. *NeurIPS*, 2019

5.4.1 *An illustrative example that mixes optimization and generalization*

Suppose the points are in $\Re^{D+K}$ and the labels are ± 1. There is a fixed vector $u \in \Re^k$ such that labeled datapoints (x, y) come from the following distribution $\mathcal{D}$: first label y is uniformly picked in $\{\pm 1\}$ and then the first K coordinates of x —which we denote x_1 for convenience—are set to the vector $y \cdot u$. The remaining D coordinates, denoted x_2 consist of a random vector $\Re^D$, whose each coordinate is drawn independently from $\mathcal{N}(0, 1/D)$. Measure concentration implies x_2 is distributed essentially like a random unit vector in $\Re^D$.

The classification can clearly be solved using the linear classifier $x \to \text{sgn}(w^* \cdot x)$ where the first K coordinates contain $w_1^* = u/\|u\|_2^2$, and the last D coordinates contain $w_2^* = 0$.

Let's consider a simple training objective: find linear classifer $h(x)$ that maximises $y \cdot h(x)$. Roughly speaking, this ignores the magnitude of $h(x)$ and tries to align the sign of $h(x)$ and y. Using

learning rate $\eta = 1$ and a sample S of m datapoints (x^i, y^i) for $i = 1, \ldots, m$ gradient descent produces a classifier with $w_S = (w_1, w_2)$ where

$$w_{S,1} = m \cdot u, \qquad w_{S,2} = \sum_i y^i x_2^i. \tag{5.5}$$

Notice that $w_{S,2}$ is a sum of m random unit vectors, which means its norm is fairly tightly concentrated around $\sqrt{m}$. In other words, unlike our ideal classifier w^* the learnt classifier has a lot of junk in the last D coordinates that is not relevant to the classification.

Now we describe how to set the various parameters. As usual m denotes training set size. We set

$$m\sqrt{(\log 1/\delta)} \approx D \tag{5.6}$$

$$\|u\|_2^2 = \frac{1}{m} \tag{5.7}$$

The ℓ_2 norm of the learnt classifier w_S is around $\sqrt{m^2\|u\|_2^2 + m}$, and thus (5.7) implies this norm is $\sqrt{2m}$.

Let's check that the junk coordinates do not interfere with classification for randomly chosen data points m—in other words, has good test error. Given a new data point $x = (y \cdot u, x_2)$ where x_2 is a random unit vector, the learnt classifier produces the answer $w_S \cdot x = my\|u\|_2^2 + x_2 \cdot (\sum_i y^i x_2^i)$. Since inner product between a fixed vector and a random gaussian vector $\mathcal{N}(0, 1/D)$ is a univariate gaussian with standard deviation $1/\sqrt{D}$ times the norm of the fixed vector, we see that the sign of this is correct i.e. y, with probability $1 - \delta$ so long as

$$m\|u\|_2^2 > \sqrt{\frac{m \log 1/\delta}{D}},$$

which holds from (5.6) and (5.7). Thus the learnt classifier works fine on random test data points.

But now imagine we try to explain the success of learning via a union-bound argument. Let's denote by $\mathcal{H}_0$ the set of such classifiers that could result from GD on training sets of m datapoints. The argument would have to prove that with high probability, $\Delta_S(h)$ is small for all classifiers $h \in \mathcal{H}_0$. The next result shows this is not true.

Claim 5.4.1. *For a random sample set S, whp there is a classifier w_{flip} whose generalization error is large (specifically, whose loss on full distribution $\mathcal{D}$ is small but whose loss on S is large.)*

Proof. We let w_{flip} be the classifier trained on the set S_{flip}, which is obtained by taking S and flipping the sign of the x_2 part. In other words, datapoint $z = (y^i u, x_2^i), y^i)$ of S turns into $z_{\text{flip}} = (y^i u, -x_2^i), y^i)$ in S_{flip}. Note that S_{flip} has exactly the same probability as S. By our

earlier analysis, w_S and w_{flip} agree on the first K coordinates, but have the signs of the last D coordinates flipped. Thus the absolute value of $(w_S - w_{\text{flip}}) \cdot z$ is at least $2x_2^i \cdot x_2^i = 2$. Thus we have shown that the signs of $w_S \cdot z$ and $w_{\text{flip}} \cdot z$ are different. □

Let us consider what we have shown. The classifier w_S and $w_{\textbf{flip}}$ both have excellent test error. However, on the training set S used to produce w_S, the classifier $w_{\textbf{flip}}$ has bad generalization error. This shows a stumbling block on proving good generalization of w_S on training dataset S using the naive union bound.

Note that the limitations shown above do not hold if we are allowed to modify/prune the classifier obtained at the end of training. One can imagine identifying non-influential coordinates in the learnt classifier via some simple test and realizing that the last D coordinates can be zero-ed out without greatly affecting accuracy. Then all learnt classifiers become scalar multiples of the ideal classifier w^*. In other words, the limitations shown here do not apply to the approach we describe in the next Section.

5.5 *A Compression-based framework*

Now we described a simple compression-based technique[12] from Arora et al [13] that formalizes a very simple idea. Suppose the training dataset S contains m samples, and h is a classifier from a complicated class (e.g., deep nets with much more than m parameters) that incurs very low empirical loss. We are trying to understand from looking at h and S how well h will generalize. Now suppose we can compute a classifier g with discrete trainable parameters much fewer than m and which incurs similar loss on the training data as h. We call this an *approximator* for h. Then if g has sufficiently low description length, its generalization follows by simple union bound argument. [14]

This framework has the advantage of staying with intuitive parameter counting and to avoid explicitly dealing with the hypothesis class that includes h (see note after Theorem 5.5.3). Notice, the mapping from f to g merely needs to *exist*, not to be efficiently computable. But in all our examples the mapping will be explicit and fairly efficient. Now we formalize the notions. The proofs are elementary via concentration bounds and appear in the appendix.

Definition 5.5.1 ((γ,S)-compressible)**.** *Let f be a classifier and $G_{\mathcal{A}} = \{g_A | A \in \mathcal{A}\}$ be a class of classifiers. We say f is (γ, S)-compressible via $G_{\mathcal{A}}$ if there exists $A \in \mathcal{A}$ such that for any $x \in S$, we have for all y*

$$|f(x)[y] - g_A(x)[y]| \leq \gamma.$$

[12] Do not confuse this with another older and unrelated technique in generalization theory based upon *data compression*, which is not applicable to deep learning.

[13] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *Proc. ICML 2018*, pages 254–263, 2018

[14] This scenario is quite reminiscent of empirical work in network pruning, whereby trained deep nets are compressed using one of a long list of methods that prune away lots of parameters and retrain the rest. If network left after pruning is compact enough, one can conceivably prove generalization bounds for the pruned net. See

We also consider a different setting where the compression algorithm is allowed a"helper string" s, which is arbitrary but fixed before looking at the training samples. Often s will contain random numbers. [15]

[15] A simple example is to let s be the random initialization used for training the deep net. Then one could compress the *difference* between the final weights and s; this can give better generalization bounds.

Definition 5.5.2 ((γ,S)-compressible using helper string s)**.** *Suppose $G_{\mathcal{A},s} = \{g_{A,s} | A \in \mathcal{A}\}$ is a class of classifiers indexed by trainable parameters A and fixed strings s. A classifier f is (γ, S)-compressible with respect to $G_{\mathcal{A},s}$ using helper string s if there exists $A \in \mathcal{A}$ such that for any $x \in S$, we have for all y*

$$|f(x)[y] - g_{A,s}(x)[y]| \leq \gamma.$$

The following theorem is a simple application of the union bound method above.

Theorem 5.5.3. *Suppose $G_{\mathcal{A},s} = \{g_{A,s} | A \in \mathcal{A}\}$ where A is a set of q parameters each of which can have at most r discrete values and s is a helper string. Let S be a training set with m samples. If the trained classifier f is (γ, S)-compressible via $G_{\mathcal{A},s}$ with helper string s, then there exists $A \in \mathcal{A}$ with high probability over the training set,*

$$L(g_A) \leq \widehat{L}_\gamma(f) + O\left(\sqrt{\frac{q \log r}{m}}\right),$$

where $L(f) = \mathbb{E}_{(x,y)\in\mathcal{D}}[f(x)[y] \leq \max_{j\neq y} f(x)[j]]$ is the expected error and $\widehat{L}_\gamma(f)$ is the proportion of data (x, y) satisfying $f(x)[y] \leq \max_{j\neq y} f(x)[j]$ in the training set S.

Remarks: (1) The framework proves the generalization not of f but of its compression g_A. (An exception is if the two are shown to have similar loss at every point in the domain, not just the training set. This is the case in Theorem 5.5.6.)
(2) The previous remark highlights the difference from what we called the union bound earlier (Theorem 5.2.1). There, one needs to fix a hypothesis class *independent* of the training set. By contrast we have no hypothesis class, only a *single* neural net that has some specific properties on a *single* finite training set. But if we can compress this specific neural net to a simpler neural nets with fewer parameters then we can use covering number argument on this simpler class to get the generalization of the compressed net.
(3) Issue (1) exists also in how researchers often apply the standard PAC-Bayes framework for deep nets (Section 5.6).

5.5.1 Example 1: Linear classifiers with margin

To illustrate the above compression method we use linear classifiers with high margins. Consider a simple family of linear classifiers,

consisting of unit vectors $c \in \mathbb{R}^d$ whose ± 1 output on input x is given by $\mathrm{sgn}(c \cdot x)$ (i.e., sign of the inner product with the datapoint). Assume that all data points are also unit vectors. Say c has *margin* γ if for all training pairs (x, y) we have $y(c^\top x) \geq \gamma$.

We show how to compress such a classifier with margin γ to one that has only $O(1/\gamma^2)$ non-zero entries. First, assume all c_i have absolute value less than $\gamma^2/8$.

For each coordinate i, toss a coin with $\Pr[heads] = p_i = 8c_i^2/\gamma^2$ and if it comes up heads set the coordinate to equal to $c_i/p_i = \gamma^2/8c_i$. This yields a vector $\widehat{c}$. The expected number of non-zero entries in $\widehat{c}$ is $\sum_{i=1}^d p_i = 8/\gamma^2$. By Chernoff bound we know with high probability the number of non-zero entries is at most $O(1/\gamma^2)$.

Furthermore, variance of coordinate i of $\widehat{c}$ is $2p_i(1-p_i)\frac{c_i^2}{p_i^2} \leq \frac{2c_i^2}{p_i} \leq \gamma^2/4$. Therefore, for any unit vector u that is independent with the choice of $\widehat{c}$, we have $\mathbb{E}[\widehat{c}^\top u] = c^\top u$. Now we estimate variance of the random variable $\widehat{c}^\top u$. It is $\leq \gamma^2/4 \cdot \|u\|^2 \leq \gamma^2/4$. By Chebyshev's inequality we know $\Pr[|\widehat{c}^\top u - c^\top u| \geq \gamma] \leq 1/4$, so $\widehat{c}$ and c will make the same prediction for all u satisfying $|c^\top u| \geq \gamma$. We can then apply Theorem 5.5.3 on a discretized version of $\widehat{c}$ (via trivial rounding) to show that the sparsified classifier has good generalization with $O(\log d/\gamma^2)$ samples.

Problem 5.5.4. *Redo the proof above when some coordinates have absolute value more than $\gamma^2/8$.*

This compressed classifier works correctly for a fixed input x with constant probability but not high probability. To fix this, one can recourse to the "compression with fixed string" model. The fixed string is a random linear transformation. When applied to unit vector c, it tends to equalize all coordinates and the guarantee $|\widehat{c}^\top u - c^\top u| < \gamma$ can hold with high probability. This random linear transformation can be fixed before seeing the training data.

Problem 5.5.5. *Prove the above property of random linear transformations. That is, let M be a random matrix of size $O(1/\gamma^2) \times d$, drawn from a suitable distribution you choose before seeing the unit vector c and the training data. Then, show that the following holds for fixed unit vectors c and u with high probability*

$$\|Mc\|_\infty = O(1), \qquad |\langle Mc, Mu\rangle - \langle c, u\rangle| < \gamma.$$

This means we can compress a unit vector c to $\widehat{c} = M^\top Mc$. Finally, Apply Theorem 5.5.3 on a discretized version of $\widehat{c}$ to show a good generalization bound with $\widetilde{O}(1/\gamma^2)$ samples, where $\widetilde{O}$ can hide polylog factors of d and $1/\gamma$.

5.5.2 *Example 2: Generalization bounds for deep nets using low rank approximations*

Some of the early generalization bounds for fully connected nets used the fact that layer matrices are often found to be low rank. (Or perhaps the final matrix minus the initialization.) We give a simple proof of such a result.

Realize that an $h \times h$ matrix of rank r has effectively $2hr$ parameters despite having h^2 entries. We recall that for a square matrix A the spectral norm (i.e., largest singular value) is denoted $\|A\|_2$ and sum of squares of singular values is denoted denoted $\|A\|_F^2$ where $\|\cdot\|_F$ is also called *Frobenius norm*. The ratio $\|A\|_F^2/\|A\|_2^2$ is called *stable rank*, and it is clearly upper bounded by the rank. Often the layers of the trained net have low stable rank even though rank per se is high.

Theorem 5.5.6. *([16]) For a depth-d ReLU net with hidden layers of equal width h and single coordinate output, let $A^1, A^2, \ldots A^d$ be weight matrices and γ be the output margin on a training set S of size m. Then the generalization error can be bounded by*

$$\widetilde{O}\left(\sqrt{\frac{hd^2 \max_{x\in S}\|x\| \prod_{i=1}^d \|A^i\|_2^2 \sum_{i=1}^d \frac{\|A^i\|_F^2}{\|A^i\|_2^2}}{\gamma^2 m}}\right).$$

[16] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *ICLR*, 2018

The second part of this expression ($\sum_{i=1}^d \frac{\|A^i\|_F^2}{\|A^i\|_2^2}$) is sum of stable ranks of the layers, a natural measure of their true parameter count. The first part ($\prod_{i=1}^d \|A^i\|_2^2$) is related to the Lipschitz constant of the network, namely, the maximum norm of the vector it can produce if the input is a unit vector. The Lipschitz constant of a matrix operator B is just its spectral norm $\|B\|_2$. Since the network applies a sequence of matrix operations interspersed with ReLU, and ReLU is 1-Lipschitz we conclude that the Lipschitz constant of the full network is at most $\prod_{i=1}^d \|A^i\|_2$.

To prove Theorem 5.5.6 we use the following lemma to compress the matrix at each layer to a matrix of smaller rank. Since a matrix of rank r can be expressed as the product of two matrices of inner dimension r, it has $2hr$ parameters (instead of the trivial h^2). (Furthermore, the parameters can be discretized via trivial rounding to get a compression with discrete parameters as needed by Definition 5.5.1.)

Lemma 5.5.7. *For any matrix $A \in \mathbb{R}^{m\times n}$, let $\widehat{A}$ be the truncated version of A where singular values that are smaller than $\delta\|A\|_2$ are removed. Then $\|\widehat{A} - A\|_2 \le \delta\|A\|_2$ and $\widehat{A}$ has rank at most $\|A\|_F^2/(\delta^2\|A\|_2^2)$.*

Proof. Let r be the rank of $\widehat{A}$. By construction, the maximum singular

value of $\widehat{A} - A$ is at most $\delta\|A\|_2$. Since the remaining singular values are at least $\delta\|A\|_2$, we have $\|A\|_F \geq \|\widehat{A}\|_F \geq \sqrt{r}\delta\|A\|_2$. □

For each i replace layer i by its compression using the above lemma, with $\delta = \gamma(3d\|x\|\prod_{i=1}^{d}\|A^i\|_2)^{-1}$. How much error does this introduce at each layer and how much does it affect the output after passing through the intermediate layers (and getting magnified by their Lipschitz constants)? Since $A - \widehat{A}^i$ has spectral norm (i.e., Lipschitz constant) at most $\delta\|A^i\|_2$, the error at the output due to changing layer i in isolation is at most $\prod_{j=i+1}^{d}\|A^j\|_2 \cdot \delta\|A^i\|_2 \cdot \prod_{j=1}^{i-1}\|A^j\|_2 \cdot \|x\| \leq \gamma/3d$. Rest of the proof is left to the reader and generalization bound follows immediately from Theorem 5.5.3.

Problem 5.5.8. *Complete the above proof using a simple induction (see* [17] *if needed) to show the total error incurred in all layers is strictly bounded by γ. That is, for an input x, the change in the deep net output is smaller than γ after replacing every weight matrix A^i with its truncated version $\widehat{A}^i$.*

[17] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *ICLR*, 2018

5.6 PAC-Bayes bounds

These bounds due to McAllester (1999) [McA99] are in principle the tightest, meaning previous bounds in this chapter are its subcases. They are descended from an old philosophical tradition of considering the logical foundations for belief systems, which often uses Bayes' Theorem. For example, in the 18th century, Laplace sought to give meaning to questions like "*What is the probability that the sun will rise tomorrow?*" The answer to this question depends upon the person's prior beliefs (e.g., degree of scientific knowledge) as well as their empirical observation that the sun has risen every day in their lifetime. This philosophical connection sometimes helps students improve their understanding of generalization.

In ML context, PAC-Bayes bounds assume that experimenter (i.e., ML expert) has some prior distribution P over the hypothesis $\mathcal{H}$. If asked to classify without seeing any concrete training data, the experimenter would pick a hypothesis h according to P (denoted $h \sim P$) and classify using it h. After seeing the training data and running computations, the experimenter's distribution changes to the posterior Q, meaning now if asked to classify they would pick $h \sim Q$ and use that. Thus the expected test loss is

$$\mathop{\mathbb{E}}_{h\sim Q}[L_{\mathcal{D}}(h)].$$

The theory requires Q to be a *valid posterior* with respect to P, meaning every hypothesis h that gets zero probability under P also must have zero probability under Q. The following form of PAC-Bayes bound is from [18].

[18] John Langford. *Quantitatively tight sample complexity bounds*. PhD Thesis CMU, 2002

Theorem 5.6.1 (PAC-Bayes bound). *Let $\mathcal{D}$ be the data distribution and P be a prior distribution over hypothesis class $\mathcal{H}$ and $\delta > 0$. If S is a set of i.i.d. samples of size m from $\mathcal{D}$ and Q is any valid posterior (possibly depending arbitrarily on S) then $\Delta_S(Q) = \mathbb{E}_{h\sim Q}[L_{\mathcal{D}}(h) - \widehat{L}_S(h)]$ satisfies the following bound with probability $1-\delta$,*

$$\Delta_S(Q) \leq \sqrt{\frac{D(Q||P) + \ln(2m/\delta)}{2(m-1)}},$$

where $D(Q||P) = \mathbb{E}_{h\sim Q}[\ln(Q(h)/P(h))]$ is the so-called KL-divergence[19].

[19] This is a measure of distance between distributions, meaningful when P dominates Q, in the sense that every h with nonzero probability in Q also has nonzero probability in P. Note that in this definition, $0 \ln 0$ is interpreted as 0.

In other words, generalization error can be upper bounded using the (square root of) KL-divergence of the distributions, plus some terms that arise from concentration bounds.

Example 5.6.2. *P could be the standard normal distribution, which assigns nonzero probability to every vector. For any sample set S, we could let Q be the distribution on parameter vectors obtained by vanilla deep learning using S: that is, initialize parameters using random Gaaussian, and train with SGD with a predetermined learning rate schedule. Since SGD is a stochastic process (due to randomness of batches) it leads to a natural distribution Q on trained classifiers at the end of training. Notice, Q is a valid posterior of P because P assigns nonzero probability to every classifier h. As this example emphasizes, one can consider various P and Q for the same classification setup (e.g., by changing some aspect of training) and the generalization bound will hold for every fixed choice.*

Example 5.6.3. *Suppose h is any classifier and P, Q are the distribution that assigns probability 1 to h and zero to all other hypotheses. Then $D(Q||P) = 0$, and by Hoeffding bound we have $\Delta_S(Q) = \Delta_S(h) \leq \sqrt{\frac{\log(1/\delta)}{2m}}$. The inequality in PAC-Bayes bound is satisfied.*

Problem 5.6.4. *Derive the union bound Theorem 5.2.1 using PAC-Bayes.*

Now we're ready to prove Theorem 5.6.1. In interest of exposition, we prove a weaker statement that is qualitatively similar but not quite correct:

$$\Delta_S(Q) \leq \sqrt{\frac{2(D(Q||P) + \ln(2/\delta))}{m}} \tag{5.8}$$

The incorrectness arises due to a simplifying assumption about the quantity $z = \sqrt{m}\Delta_S(h)$ where h is a fixed classifier and S is a random subset of m samples. Since $\Delta_S(h)$ is an average of m iid variables taking values in $[-1, 1]$ and with mean 0, we assume z behaves *exactly* like a normal distribution $\mathcal{N}(0, 1)$. Of course, in truth z is dominated in distribution by $\mathcal{N}(0, 1)$ in the limit $m \to \infty$. This assumption can be removed by using a more quantitative argument with Hoeffding

bound. The assumption allows us to assume that expected value of $e^{z^2/(2+\epsilon)}$ approaches $\sqrt{2}$ when x is drawn from $\mathcal{N}(0,1)$ and ϵ is an arbitrarily small constant. For simplicity we will assume $e^{z^2/2} = \sqrt{2}$. It is possible to fix the proof using concentration bounds.

Proof. (Theorem 5.6.1, weaker version (5.8)) Rearranging the expression in the theorem statement, we see that it gives an upper bound of $\ln(2/\delta)$ on $(m/2)\,\mathbb{E}_{h\sim Q}[\Delta_S(h)]^2 - D(Q||P)$. By Jensen's inequality [20] applied to the square function $f(x) = x^2$, this expression is at most $(m/2)\,\mathbb{E}_{h\sim Q}[\Delta_S(h)^2] - D(Q||P)$. We show this is upper bounded by $\ln(2/\delta)$. The steps are:

$$\begin{aligned}
&= \mathop{\mathbb{E}}_{h\sim Q}\left[(m/2)\Delta_S(h)^2 - \ln(Q(h)/P(h))\right] \\
&= \mathop{\mathbb{E}}_{h\sim Q}\left[\ln\left(\exp((m/2)\Delta_S(h)^2)\cdot P(h)/Q(h)\right)\right] \\
&\le \ln\left(\mathop{\mathbb{E}}_{h\sim Q}\left[\exp((m/2)\Delta_S(h)^2)\cdot P(h)/Q(h)\right]\right),
\end{aligned}$$

where the last inequality uses Jensen's inequality along with the concavity of ln. Also, since taking expectation over $h \sim Q$ is effectively like a weighted sum with term for h weighted by $Q(h)$, we have [21]

$$\ln \mathop{\mathbb{E}}_{h\sim Q}\left[\exp((m/2)\Delta(h)^2)\cdot P(h)/Q(h)\right] = \ln \mathop{\mathbb{E}}_{h\sim P}\left[\exp((m/2)\Delta(h)^2)\right].$$

Recapping, we have thus shown the following for a fixed dataset S:

$$(m/2)\mathop{\mathbb{E}}_{h\sim Q}[\Delta_S(h)]^2 - D(Q||P) \le \ln\left(\mathop{\mathbb{E}}_{h\sim P}\left[e^{(m/2)\Delta_S(h)^2}\right]\right) \tag{5.9}$$

Note that the RHS has no dependence on posterior Q. Using the fact that S is a random sample of size m and that prior belief P was fixed before seeing S (i.e., is independent of S):

$$\mathop{\mathbb{E}}_{S}\left[\mathop{\mathbb{E}}_{h\sim P}\left[\, e^{(m/2)\Delta_S(h)^2}\right]\right] = \mathop{\mathbb{E}}_{h\sim P}\left[\mathop{\mathbb{E}}_{S}\left[e^{(m/2)\Delta_S(h)^2}\right]\right] = \sqrt{2} \le 2.$$

Simple averaging implies that with probability $1-\delta$ over S,

$$\mathop{\mathbb{E}}_{h\sim P}\left[e^{(m/2)\Delta_S(h)^2}\right] \le 2/\delta \tag{5.10}$$

and now by taking logarithm of both sides the proof is completed. □

[20] Jensen's Inequality: For a concave function f and random variable X, $\mathbb{E}[f(X)] \le f(\mathbb{E}[X])$. For convex function the inequality is reversed.

[21] Often when you see KL-divergence in machine learning, you will see this trick being used to switch the distribution over which expectation is taken!

5.6.1 *Gemini 2.5 adapted above proof for original statement!!*

Now we're ready to prove Theorem 5.6.1. The standard proof can feel like a series of unmotivated algebraic tricks. Instead, we present a proof sketch that is mathematically sound but aims for greater intuition. It relies on two key ingredients which we will take as given.

The first is a fundamental inequality that connects an expectation over the posterior Q to one over the prior P. For any function $f(h)$, it states:

$$\mathop{\mathbb{E}}_{h\sim Q}[f(h)] \leq D(Q||P) + \ln\left(\mathop{\mathbb{E}}_{h\sim P}\left[e^{f(h)}\right]\right). \tag{5.11}$$

The second ingredient is **Hoeffding's Lemma**, the tool used to prove the Hoeffding inequality you have already seen. It gives a tight bound on the moment-generating function of an average. For an average A of m i.i.d. random variables with mean 0, each bounded in $[-1, 1]$, and any $\lambda > 0$:

$$\mathbb{E}\left[e^{\lambda A}\right] \leq e^{\lambda^2/2m}. \tag{5.12}$$

This lemma formalizes the idea that the average A is strongly concentrated around its mean in a sub-Gaussian manner. With these two tools, the proof becomes a straightforward sequence of steps.

Proof. (Theorem 5.6.1, sketch) Let's apply the key inequality (5.11) by setting $f(h) = \lambda\Delta_S(h)$ for some $\lambda > 0$ that we will choose later to get the tightest bound. For any *fixed* training set S, we have:

$$\lambda \mathop{\mathbb{E}}_{h\sim Q}[\Delta_S(h)] \leq D(Q||P) + \ln\left(\mathop{\mathbb{E}}_{h\sim P}\left[e^{\lambda\Delta_S(h)}\right]\right).$$

The right-hand side still depends on the random sample S. To handle this, we take the expectation over S on both sides. Since P is independent of S, and using Jensen's inequality for the concave ln function ($E[\ln X] \leq \ln E[X]$), we can move the expectation inside the logarithm:

$$\begin{aligned}\mathop{\mathbb{E}}_{S}\left[\lambda \mathop{\mathbb{E}}_{h\sim Q}[\Delta_S(h)]\right] &\leq D(Q||P) + \mathop{\mathbb{E}}_{S}\left[\ln\left(\mathop{\mathbb{E}}_{h\sim P}\left[e^{\lambda\Delta_S(h)}\right]\right)\right] \\ &\leq D(Q||P) + \ln\left(\mathop{\mathbb{E}}_{S}\left[\mathop{\mathbb{E}}_{h\sim P}\left[e^{\lambda\Delta_S(h)}\right]\right]\right).\end{aligned}$$

Now for the crucial step, we swap the order of expectations. [22]

[22] This is permitted by Fubini's theorem, as the terms are non-negative.

$$\mathop{\mathbb{E}}_{S}\left[\lambda \mathop{\mathbb{E}}_{h\sim Q}[\Delta_S(h)]\right] \leq D(Q||P) + \ln\left(\mathop{\mathbb{E}}_{h\sim P}\left[\mathop{\mathbb{E}}_{S}\left[e^{\lambda\Delta_S(h)}\right]\right]\right).$$

Let's focus on the inner term $\mathbb{E}_S\left[e^{\lambda\Delta_S(h)}\right]$. For any *fixed* h, $\Delta_S(h)$ is an average of m mean-zero variables (the per-sample losses) bounded in $[-1, 1]$. This is exactly the setup for Hoeffding's Lemma (5.12). Applying it gives:

$$\mathop{\mathbb{E}}_{S}\left[e^{\lambda\Delta_S(h)}\right] \leq e^{\lambda^2/2m}.$$

Substituting this powerful and simple bound back into our main inequality, we find that the expectation over P becomes trivial, as the

bound is independent of h:

$$\begin{aligned}\mathop{\mathbb{E}}_{S}\left[\lambda \mathop{\mathbb{E}}_{h\sim Q}[\Delta_S(h)]\right] &\leq D(Q||P) + \ln\left(\mathop{\mathbb{E}}_{h\sim P}\left[e^{\lambda^2/2m}\right]\right)\\ &= D(Q||P) + \ln\left(e^{\lambda^2/2m}\right)\\ &= D(Q||P) + \frac{\lambda^2}{2m}.\end{aligned}$$

We have shown that the expected generalization error (averaged over all possible datasets S) is bounded. A final application of Markov's inequality implies that with probability at least $1-\delta$ over the choice of S, the generalization error is not much larger than this expectation. A more careful derivation yields:

$$\mathop{\mathbb{E}}_{h\sim Q}[\Delta_S(h)] \leq \frac{D(Q||P)+\ln(1/\delta)}{\lambda} + \frac{\lambda}{2m}.$$

This bound holds for any $\lambda > 0$. We can choose λ to minimize the right-hand side. The optimal choice is $\lambda = \sqrt{2m(D(Q||P)+\ln(1/\delta))}$, which leads to the final bound in the theorem. This completes the sketch, showing that the generalization error scales as $O\left(\sqrt{\frac{D(Q||P)+\ln(1/\delta)}{m}}\right)$. □

5.7 *Exercises*

1. Assume the loss function ℓ is 1-Lipschitz. Consider the kernel classifier of the form $h(x) = z^\top G^{-1} y$ we studied in Section 2.2 where G is the $n \times n$ kernel matrix, y is the labels and z is the column vector whose i-th coordinate is $K(x, x_i)$. Prove that its Rademacher complexity is upper bounded by $\sqrt{2y^\top Gy \cdot \mathrm{Tr}(G)}/n$. (Hint: view the kernel classifier as a linear classifier in a Reproducing Kernel Hilbert Space.)

2. **Rademacher Complexity of a Finite Class.** Let $\mathcal{H}$ be a finite hypothesis class with $|\mathcal{H}| = N$. Show that its Rademacher complexity (as defined in Eq. 5.3) is bounded by:

$$\mathcal{R}_{m,D}(\mathcal{H}) \leq \sqrt{\frac{2\ln(2N)}{m}}$$

   (Hint: Use Hoeffding's inequality on the inner term for a fixed h, then apply a union bound. A more advanced hint is to use Massart's finite lemma.) This shows that for finite classes, the Rademacher complexity bound is closely related to the direct union bound of Theorem 5.2.1.

3. **PAC-Bayes with Gaussian Distributions.** Consider a simple hypothesis class of linear classifiers in $\mathbb{R}^k$, where a hypothesis is

just a vector $w \in \mathbb{R}^k$. Let the prior distribution P be a standard Gaussian, $w \sim \mathcal{N}(0, I_k)$. After training on a dataset S, we find an optimal weight vector $\widehat{w}$. Let the posterior distribution Q be a Gaussian centered at $\widehat{w}$ with the same variance, $w \sim \mathcal{N}(\widehat{w}, I_k)$.

(a) Compute the KL-divergence $D(Q||P)$. Recall that the KL-divergence between two multivariate Gaussians $Q = \mathcal{N}(\mu_Q, \Sigma_Q)$ and $P = \mathcal{N}(\mu_P, \Sigma_P)$ is given by

$$D(Q||P) = \frac{1}{2}\left(\text{tr}(\Sigma_P^{-1}\Sigma_Q) + (\mu_P - \mu_Q)^\top \Sigma_P^{-1}(\mu_P - \mu_Q) - k + \ln\left(\frac{\det \Sigma_P}{\det \Sigma_Q}\right)\right).$$

(b) Substitute your result into the PAC-Bayes bound (Theorem 5.6.1). How does the bound depend on the squared norm of the learned weight vector, $\|\widehat{w}\|_2^2$?

(c) What does this suggest about the relationship between finding low-norm solutions (as in L2 regularization) and obtaining good generalization bounds?

4. **Generalization via Quantization.** Recall the quantization idea from Example 5.2.2. Suppose you have trained a deep network f with k parameters, $w_1, \ldots, w_k$. You find that you can quantize each weight w_i to the nearest multiple of some small $\epsilon > 0$ within the range $[-M, M]$, creating a new network g. The number of possible discrete values for each weight is thus $2M/\epsilon + 1$. Assume this quantization is (γ, S)-compressible, meaning the output of g differs from f by at most γ on the training set S.

(a) Let the set of all such quantized networks be G. What is an upper bound on the size of this class, $|G|$?

(b) Use the compression framework (Theorem 5.5.3) to derive a generalization bound for the quantized network g. Your bound should depend on k, m, and the number of quantization levels.