**Generalization from Margins: Connecting Boosting Theory to Neural Nets**
Kiran Vodrahalli, Columbia DSI Seminar: 09/29/17

# Contents

## 1 Generalization Theory

In machine learning theory, we are in the business of trying to identify conditions under which classes of models and algorithms for learning work in the real world. Coming up with a good theory is valuable for at least two reasons:

(a) A good theory gives intuition for what will work in practice;

(b) With theory, we can give **guarantees** if certain conditions are satisfied that we will end up with a "good" model (for some definition of "good") in a reasonable amount of time (for some definition of "reasonable").

Of course, there are many more reasons to do theory. The main point of this spiel is to motivate the study of **generalization bounds** in learning theory. Generally, we want to claim that it is possible for a model to generalize beyond a training set to all future data. No Free Lunch theorems tell us that we cannot hope to do this if we allow ourselves to consider all possible models (there will always be some model which perfectly fits the training set but is the worst possible model on everything outside of that training set). Therefore, we must restrict the class of models we consider in some way. In the following treatment, we will draw from textbooks and previous surveys (Shalev-Shwartz & Ben-David (2014); Bousquet et al. (2004); Boucheron et al. (2005)).

## 1.1 Classic Approaches to Generalization Bounds

The classic results of learning theory make this notion of restriction precise. Adopting the binary classification setting with labels $\{\pm 1\}$, our goal is to bound the difference between the the expected error on the whole distribution and the expected error on a finite data set of size $n$ sampled i.i.d. from the distribution. We will call the expected error the **risk**, denoted by $\mathcal{R}$. Specifically, we define

**Definition 1.1.** Empirical risk.
The risk for a function $f$ on a dataset $\{(X_i, Y_i)\}_{i=1}^n$ is given by

$$\mathcal{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\left\{f(X_i) = Y_i\right\}$$

We choose the $0-1$ loss to define risk in the classification setting.

For cases where we only consider a finite number of models, we have the following bound:

**Theorem 1.2.** *Learning Finite Hypothesis Classes.*
*For a finite hypothesis class $\mathcal{H} = \{h_1, \cdots, h_N\}$, for all $\delta > 0$ with probability $\geq 1-\delta$, for all $h \in \mathcal{H}$,*

$$\mathcal{R}(h) \leq \mathcal{R}_n(h) + \mathcal{O}\left(\sqrt{\frac{\log N + \log\frac{1}{\delta}}{2n}}\right)$$

*where $\mathcal{R}(h)$ denotes the population risk of hypothesis $h$ and $\mathcal{R}_n(h)$ denotes the expected risk on a dataset of size $n$ for hypothesis $h$.*

This theorem essentially follows by concentration of measure (Hoeffding's inequality/Chernoff bounds) to ensure the expected risk is close to the population risk (think Law of Large Numbers) and an application of the union bound over all possible classifications of the data. Since there are only $N$ functions in our hypothesis class, the number of possible classifications is bounded by $N$, and we get the $\log N$ term. However, we probably want to be able to consider much larger sets of potential models (for instance, hyperplanes in $\mathbb{R}^d$).

**Remark 1.3.** Bias-Variance.
Note that all that we have said so far is that we can bound the deviation of the risk on a sample compared to the risk on the population (i.e., the *variance*). We have not yet said anything about how *good* the population risk is yet. In the **realizable** setting, we assume that there is a hypothesis $h^* \in \mathcal{H}$ which attains 0 population risk (i.e., $h^*$ is *unbiased*). In the agnostic setting, we allow for some bias in the hypothesis class (perhaps due to noise). In either case, we compare the error to the **best hypothesis in the class**. Thus, we can interpret the above theorem as a kind of **bias-variance tradeoff**.

**Remark 1.4.** Computational Efficiency.
The act of choosing the best hypothesis in class is known as the **ERM algorithm**, which stands for *empirical risk minimization*. Note that finding the ERM is not necessarily efficient, and depends on the hypothesis class. This is the optimization component of machine learning. We would like to have situations where we both have efficiency in the sampling regime (statistical efficiency) and efficiency in the optimization regime (computational efficiency).

**Remark 1.5.** Regularization.
The term regularization refers to modifying the definition of risk (typically by adding some additional penalty to the loss function). It can be thought of both in terms of computational efficiency (certain regularizers can improve the speed of optimization algorithms) and statistical efficiency (defined correctly, we can think of regularization as limiting the model class).

### 1.1.1 VC Dimension

The desire to prove bounds for infinite hypothesis classes motivates the definition of a different measure of complexity, the VC dimension. What we really cared about last time with the union bound was the number of possible realizable classifications on the dataset. So, for an infinite class, we'll "project" the function class onto a finite sample of the dataset and try to bound the number of possible classifications given the function class as the sample size grows large. Intuitively, to match the bound for finite sample complexity, we hope for growth which occurs polynomially in sample size so that we get some factor of $\log n$ in the numerator as before (recall $n = N$ there). Note that things would be hopeless in the case of exponential growth: We would end up with a $\mathcal{O}(1)$ error bound, which implies that error would be constant as we saw more data points, and thus not learn.

**Definition 1.6.** VC Dimension.
The **VC dimension** of a hypothesis space $\mathcal{H}$ is the size of the largest set of data points for which it is possible to assign a hypothesis from $\mathcal{H}$ to each possible classification of the data.

**Remark 1.7.** Shattering.
We say that a function class with VC dimension $d$ *shatters* a set if $\exists h \in \mathcal{H}$ for each classification).

**Remark 1.8.** Maximality of VC dimension.
Also note that VC dimension is a maximal property: It is *not* the case that a function class with VC dimension $d$ will shatter *every* set of points of cardinality $\leq d$. Indeed, it is possible to construct finite sets which function classes of infinite VC dimension cannot shatter.

The reason the VC dimension is a good measure of complexity is because it turns out that for data sets larger than the VC-dimension, the size of the number of possible classifications of the data only grows *polynomially* rather than exponentially in the size of the dataset.

**Lemma 1.9.** *Growth function (Sauer-Shelah).*
*The maximum number of possible classifications for any dataset of size $n$ by a hypothesis class of VC dimension $d$ is bounded by*

$$\sum_{i=0}^{d} \binom{n}{i} \leq \left(\frac{en}{d}\right)^d$$

The other main ingredient necessary to obtain a generalization bound in the infinite hypothesis case is a method to allow us to apply a union bound on a finite set of events. It turns out we can bound the probability of the difference between $\mathcal{R}$ and $\mathcal{R}_n$ being large by a constant factor of the probability that the difference between expected error on two randomly sampled datasets is large. This trick allows us to apply concentration to the latter probability instead. Then, we can consider two datasets of size $n$ and evaluate the number of possible classifications. By Sauer-Shelah, we know this quantity is bounded polynomially, and we can apply union bound over the polynomial number of events to get the following theorem:

**Theorem 1.10.** *Learning Hypothesis Classes of Finite VC dimension.*
*For a (potentially infinite size) hypothesis class $\mathcal{H}$ with* VC-dim$(\mathcal{H}) = d < \infty$*, for all $\delta > 0$ with probability $\geq 1 - \delta$, for all $h \in \mathcal{H}$,*

$$R(h) \leq R_n(h) + \mathcal{O}\left(\sqrt{\frac{d \log n + \log \frac{1}{\delta}}{2n}}\right)$$

Notably, this theorem implies that for datasets of size $n < d$, one will not get a good learning bound (intuitively, this is because there always exists a function in the chosen hypothesis class which completely matches the dataset). Thus, the VC dimension can be seen as a quantity which controls the amount of data necessary to learn.

In particular, the VC dimension is a combinatorial notion of complexity of a hypothesis class. Said another way, the above theorem tells us that more complex function classes require more samples to learn.

### 1.2 VC dimension of Neural Networks

A very common function class used in practice nowadays is that of deep neural networks. How does generalization work if we use VC dimension?

**Theorem 1.11.** *VC dimension of deep networks.*
*As stated in Neyshabur et al. (2017a), the VC dimension of fully connected feedforward neural networks with ReLU activation, $d$ layers, and $h$ units per layer is $\tilde{\Theta}\left(d^2h^2\right)$, giving a generalization bound of*

$$R(f_w) \leq R_n(f_w) + \tilde{\mathcal{O}}\left(\sqrt{\frac{d^2h^2}{n}}\right)$$

*where $f_w$ is a neural net parametrized by weights $w$, and the $\sim$ hides log factors of failure probability $\delta$, $n$, $d$, $h$.*

Modern neural networks can be extremely deep with a lot of units per layer. Let the total number of parameters be $p = dh$. Thus, the bound essentially depends on $\sqrt{\frac{p^2}{n}}$. This VC dimension bound only implies learnability when $p \leq \sqrt{n}$. Thus, this bound is not good enough to explain the generalization of deep networks in practice, where $\frac{p}{n} >> 1$ is typical (Zhang et al. (2017)). At first glance, we might think something like convolutional networks, which share weights, could mitigate this problem. Indeed, this gap might be a little better (i.e., $p \approx \frac{dh}{c}$ for some constant $c$ based on how weight-tying proceeds). Nevertheless, $c$ would need to depend on the size of the dataset $n$ in order for $\frac{p}{n}$ to be manageable. It is also worth noting that the number of parameters even for convolutional nets in practice still satisfies $\frac{p}{n} >> 1$!

What can we do about this gap? It appears that classical learning theory is not enough to explain generalization.

## 2 Non-Uniform Learning Bounds

When theory does not predict success in practice, and we observe success in practice, we need to take a step back and check that our theoretical definition of success is actually what we want it to be.

VC dimension bounds like those given in the previous section are referred to as *uniform learning* bounds, since they simultaneously bound the fluctuation of the risk of all functions in the hypothesis class simultaneously(essentially, bounding $\sup_f \mathcal{R}(f) - \mathcal{R}_n(f)$). Furthermore, they are completely distribution independent – it does not matter what distribution the data is drawn from as long as the data are drawn i.i.d. Thus, we are not incorporating any information about the data itself into our learning bounds. For instance, it might be the case that for the particular data we use to train a neural network, it might be easy for a member of the neural network hypothesis class to generalize.

Broadly speaking, we can remedy this deficiency by drawing on techniques from non-uniform learning.

**Definition 2.1.** Non-uniform learning bounds (non-formal).
There are several cases where uniform learning may fail to provide tight bounds, since

(a) (Assumptions about data). Uniform learning provides distribution independent bounds (may be tighter bounds for most distributions, excluding only a few bad cases).

(b) (Assumptions about hypothesis class). We may want to use randomness in our definition of our hypothesis (i.e., the power of randomness in voting algorithms – we'll come back to this notion).

(c) (Assumptions about data and hypothesis class together). We may know that hypothesis class $\mathcal{H}$ and data distribution $\mathcal{D}$ together satisfy certain properties which make generalization easier.

There are other notions of non-uniform learning as well.

Roughly speaking, the solutions to the three problematic cases presented in the definition are Rademacher complexity (and more complicated variants thereof, including VC entropy and covering numbers), the PAC-Bayes framework, and margins, respectively. We will see how these are all intricately related.

## 2.1 Rademacher Complexity

We begin by discussing Rademacher complexity, drawing from both Bousquet et al. (2004); Boucheron et al. (2005).

**Definition 2.2.** Rademacher Complexity.
Let

$$g(\sigma, Z) = \sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(Z_i)$$

be the supremum of an empirical random process, where $\sigma$ is a vector in $\{\pm 1\}^n$ with i.i.d. uniform coordinates (these are called Rademacher random variables) and $Z = \{Z_i\}_{i=1}^n$ is a i.i.d. selection of data points from the data distribution. Then, the **Rademacher complexity** of a hypothesis class $\mathcal{H}$ of functions is given by

$$\mathscr{R}(\mathcal{H}) = \mathbb{E}_{\sigma, Z}\left[g(\sigma, Z)\right]$$

We use

$$\mathscr{R}_n(\mathcal{H}) = \mathbb{E}_\sigma \left[ g(\sigma, Z) | Z \right]$$

to denote the **conditional Rademacher complexity**.

Note that this definition involves an expectation over the draw of the data: Including the data distribution directly is a departure from the previous classical setting we saw before. We will give a generalization bound in terms of the Rademacher complexity, thus resolving the first issue we wanted to solve. It turns out that it is even possible to give generalization bounds in terms of the *conditional* Rademacher complexity, thus giving generalization bounds that *depend directly on the data observed*.

### 2.1.1 Intuition

Before giving the generalization bounds, let us build some intuition. With some calculations (Bousquet et al. (2004)), we can re-write the Rademacher complexity as

$$\begin{aligned}\frac{1}{2}\mathscr{R}(\mathcal{H}) &= \frac{1}{2}\mathbb{E}_{\sigma,Z}\left[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i h(Z_i)\right] \\ &= \frac{1}{2} + \mathbb{E}_{\sigma,Z}\left[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} -\frac{1-\sigma_i h(Z_i)}{2}\right] \\ &= \frac{1}{2} - \mathbb{E}_{\sigma,Z}\left[\inf_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \frac{1-\sigma_i h(Z_i)}{2}\right]\end{aligned} \tag{1}$$

Then, we recognize that treating $\{(Z_i, \sigma_i)\}_{i=1}^n$ as though they were a training set $\{(X_i, Y_i)\}_{i=1}^n$, we precisely recover the definition of the empirical risk of $h$ on a dataset with labels $\sigma$:

$$\frac{1}{2}\mathscr{R}(\mathcal{H}) = \frac{1}{2} - \mathbb{E}_{Z,\sigma}\left[\inf_{h\in\mathcal{H}} \mathcal{R}_n(h(Z), \sigma)\right]$$

Thus, we can observe that first, calculating Rademacher complexity is as hard as performing empirical risk minimization (which can be NP-hard). Second, this interpretation affords us another perspective as to what Rademacher complexity means: Essentially, $\mathscr{R}_n(\mathcal{H})$ is maximized if are able to always perfectly fit any random selection of labels $\sigma$. Thus,

**Claim 2.3.** *Rademacher complexity measures how much a function class $\mathcal{H}$ is able to fit* ***random noise****.*

Therefore, we see from another perspective the limits of uniform learning bounds. If the Rademacher complexity of a function class is $\frac{1}{2}$, we have no hope of getting any kind of uniform bound, since by the probabilistic method there will always exist a set of size $\{n : n \in \mathbb{N}\}$ which can be shattered with functions from the class.

### 2.1.2 Rademacher Calculus

There are furthermore several nice properties of Rademacher random variables which allow us to get Rademacher complexities of function classes from the Rademacher complexities of related function classes using minimal algebra. We state the rules from Boucheron et al. (2005) in terms of sets of vectors $A, B$. In practice, we work with sets $\{f(\{X_i\}_{i=1}^n) : f \in \mathcal{F}\}$ for some function class $\mathcal{F}$.

(a) Linear properties.

$$\mathscr{R}_n(A \cup B) \leq \mathscr{R}_n(A) + \mathscr{R}_n(B), \mathscr{R}_n(c \cdot A) = |c|\mathscr{R}_n(A), \mathscr{R}_n(A \oplus B) \leq \mathscr{R}_n(A) + \mathscr{R}_n(B)$$

$c \cdot A$ and $A \oplus B$ are the usual definitions for vector spaces.

(b) Convex hull.

$$\mathscr{R}_n(A) = \mathscr{R}_n(\text{AbsConv}(A))$$

where AbsConv is the absolute convex hull of $A$, $\left\{\sum_{j=1}^N c_j a_j : \sum_{j=1}^N |c_j| \leq 1, a_j \in A\right\}$.

(c) Talagrand contraction (Lipschitz property). If $\phi : \mathbb{R} \to \mathbb{R}$ has $\phi(0) = 0$ and Lipschitz constant $L_\phi$, and $\phi \circ A$ applies $\phi$ over the vector space,

$$\mathscr{R}_n(\phi \circ A) \leq L_\phi \mathscr{R}_n(A)$$

### 2.1.3 Generalization Bound

Now we state the generalization bound for Rademacher complexity.

**Theorem 2.4.** *Rademacher Generalization Bound.*
*For all* $\delta > 0$ *with probability at least* $1 - \delta$,

$$\forall f \in \mathcal{F}, \mathcal{R}(f) \leq \mathcal{R}_n(f) + 2\mathscr{R}(\mathcal{F}) + \sqrt{\frac{2 \log \frac{1}{\delta}}{n}}$$

*and also with probability at least* $1 - \delta$,

$$\forall f \in \mathcal{F}, \mathcal{R}(f) \leq \mathcal{R}_n(f) + 2\mathscr{R}_n(\mathcal{F}) + \sqrt{\frac{2 \log \frac{2}{\delta}}{n}}$$

*Proof.* We briefly describe the proof from Bousquet et al. (2004). The main new point from what we saw before is the usage of bounded difference inequalities from empirical process theory. Roughly, the idea is that you can bound the deviation of a functional of $n$ independent random variables from its mean in terms of a sum of bounded differences at each coordinate. The bounded difference for the $i^{th}$ coordinate is an upper bound on how the value of the function changes when all coordinates except the $i^{th}$ coordinate are fixed.

These bounded difference type inequalities yield subgaussian concentration due to the fact that entropy tensorizes. Here, the nonlinear functional is the supremum function. One can use McDiarmid's inequality (which follows directly from Azuma-Hoeffding) to ensure that concentration happens (think Law of Large Numbers again) because the losses are bounded. Then, we follow a similar path as in the VC dimension proof. We relate the distance between the expected supremum and the empirical supremum to the distance between the expected supremum of the difference between two randomly drawn datasets of equal size, due to the fact that supremum is convex and Jensen's inequality applies. It turns out that this value is bounded above by a constant factor of the Rademacher average, because $f(Z_i')-f(Z_i)$ is a symmetric function and multiplying by $\pm 1$ does not change the distribution of $\sigma_i(f(Z_i') - f(Z_i))$. These two steps give the first claim in generalization theorem.

In order to get the second claim (conditional Rademacher generalization), all we need to is show that the conditional Rademacher average functional is very close to the population Rademacher average functional. We simply apply McDiarmid's bounded difference inequality to the Rademacher functional to achieve this result. □

It furthermore turns out that we can recover the VC dimension bounds from before with the Rademacher complexity. Essentially, Rademacher complexity provides a route towards deriving the uniform learning bounds purely through concentration inequalities. The fact that this works allows us to improve upon the crude union bound and get tighter results. This fact is discussed more in one of the remarks.

**Remark 2.5.** Symmetrization.
The argument we used in the generalization proof for VC dimension which relates the gap between population risk and empirical risk to the gap between two different draws of empirical risk is called *symmetrization*. Essentially, the name comes from the fact that $\mathcal{R}_n(f) - \mathcal{R}_n'(f)$ is a symmetric funtion under expectation (centered at zero, multiplying by $-1$ does not change the distribution). In the VC dimension case, we used the argument to allow ourselves to consider a finite set of possible classifications on a dataset of size $2n$. This trick allowed us to apply a union bound.

In the case for the Rademacher generalization bound, symmetrization falls out much more naturally and there is no need for us to use a union bound to sneak the growth function into the bound. Instead, we are able to directly bound using the *definition* of Rademacher complexity, which enforces exactly the kind of symmetry we need.

**Remark 2.6.** Converting from loss function class to hypothesis class.
Since the Rademacher generalization theorem is in terms of the class of loss functions composed with hypothesis functions, we want to re-write the bound in terms of the Rademacher complexity of the original hypothesis class $\mathcal{H}$ of maps from the data space to $\{\pm 1\}$. Fortunately, most common classification loss functions $\ell(h(X_i), Y_i)$ can be easily expressed in terms of $Y_i h(X_i)$. If the loss function is Lipschitz in this quantity, we can apply the contraction property of Rademacher averages (Talagrand contraction) in order to only pay an extra Lipschitz constant of the loss map in the generalization bound.

**Remark 2.7.** Rademacher as a Fully-Concentration Based Approach to Generalization. The main deficiency of our previous approach with the VC dimension was the fact that we had to crudely use the union bound at the end to finish. The union bound ignores dependencies between different classifications realized by the function class. However, the geometry of a given function class might exist in such a way that the resulting classifications are extremely dependent on each other, and thus actually, the growth function is much smaller than the maximum possible number of classifications according to Sauer-Shelah. Thus, we would be able to enjoy an improved bound.

The fact that the Rademacher approach exclusively uses concentration to derive the bound allows for the possibility of using more refined techniques to get the bound, following the intuition for the improvement above. This approach is the generic chaining. The main idea is as follows: We want to characterize the geometry of the function class as it relates to the projection onto data drawn from the data distribution. To do this, we can define a random metric on the function space: We randomly draw $n$ points i.i.d. from the data distribution, and define the distance metric between two functions in the space to be the average number of data points which the functions do not agree on (Hamming metric). Thus, we now have some notion of dependency between functions. We now want a notion which approximates the number of "distinct classifications". If the union bound were tight, then the maximum number of possible classifications would always exist (the upper bound in Sauer-Shelah). However, if there are some dependencies, then it may be that we can find a much smaller number of "representative functions" which are not too far away.

This notion of representative functions is encapsulated by the notion of an $\epsilon$-cover. We would like to find a finite number of functions such that an $\epsilon$-ball in the random metric around each of these functions covers the whole function class. The **covering number** of a function class, a function of $\epsilon$ and the number of data points drawn in the random metric, is the minimum size of such a cover. In other words, it acts as a proxy for the number of achievable classifications. The generic chaining is an approach which takes this value into account for all $\epsilon$. It turns out one can bound the Rademacher complexity by this value, providing another way to see how Rademacher complexity helps one take advantage of data distribution specific properties (here encoded in the random distance metric of the covering number).

### 2.1.4 Relevance of Rademacher Complexity

Now, how can we use this definition of complexity to understand neural network generalization? Rademacher complexities are often used in **margin bounds** and **PAC-Bayes** generalization bounds. We will delve briefly into each notion, and then see they are actually very closely related.

## 2.2 PAC-Bayes

The PAC-Bayes framework is the answer to the second problem we identified with uniform learning: We would like to be able to make assumptions about our hypothesis class. In particular, we want to use randomness to define a stochastic hypothesis, a non-deterministic classifier. What motivates us to use randomness in this way? Simply put, it actually makes our problem easier to solve. We can think of the problem in relation to voting-based learning algorithms (which, notably, has a LOT to do with boosting, which we will come to shortly). In particular, we compare the weighted majority (WM) algorithm to the randomized weighted majority (RWM) algorithm.

### 2.2.1 The Power of Randomness

Consider the following problem: We have a finite set of experts who we can ask to help us bet on a horse at the races (Freund & Schapire (1997); Blum (1998)). Suppose we want to do as well as the best expert, over time. The classic approach is the Weighted-Majority algorithm of Littlestone and Warmuth, which works as follows (in the 2-class setting):

(a) Assign a weight of 1 to each expert.

(b) Ask the experts to give their advice (for instance, bet on a given horse or not).

(c) Sum up the weights and normalize. Choose the choice with the larger weight.

(d) Experience the result. If you were correct, leave the weights as is. Otherwise, multiply the weights of the experts who were wrong by a factor $\beta$ where $0 < \beta < 1$. Pick $\beta = 1/2$.

If we let $M_T^*$ be the mistake bound of the best expert after $T$ rounds, then by doing a potential analysis on the total weight, it is possible to bound the number of mistakes you make.

**Theorem 2.8.** *Weighted Majority Mistake Bound (Blum (1998)).*

$$M_T \leq 2.41(M_T^* + \log N)$$

*where $N$ is the number of experts.*

This bound is good if $M_T^*$ is constant with respect to $T$, since it implies you will also make a constant number of mistakes forever. However, this bound is bad otherwise: You will do considerably worse than the best expert, no matter how much time you spend. This result basically means you are not learning from that expert. How can this unfortunate point be resolved?

It turns out it is necessary to use **randomness** to be able to truly learn. Only a small modification is necessary to the Weighted Majority algorithm: On the third step, do not just pick the choice with larger weight. Instead, choose each option with probability proportional to the weight of the expert's votes. Thus, we have derived a voting algorithm. It turns out with this simple modification, the following mistake bound becomes possible (Blum (1998)):

**Theorem 2.9.** *Randomized Weighted Majority Mistake Bound.*

$$M_T \leq \frac{M_T^* \ln(\beta) + \ln(N)}{1-\beta}$$

The power that randomness brings to the table is essentially the ability to take benefit from the average case. If there are only a few corner cases which are really bad which are not typical, these will not occur often and thus the performance of our algorithm will be much better on average. Intuitively, the randomized approach gains the most benefit when the distribution over experts tends to be close to uniform. If the distribution readily collapses to a single expert, the situation is almost identical to the deterministic algorithm case. Additionally, we can think of adding randomness over the experts in terms of defining a stochastic solution concept: We allow for convex combinations of experts now. This approach can also be viewed as convexifying the problem into a learning problem over the simplex.

### 2.2.2 Randomized Weighted Majority and PAC-Bayes

The Randomized Weighted Majority algorithm described previously illustrates why the PAC-Bayes framework is a good idea. The PAC-Bayes framework introduces a prior $\rho$ (a weight distribution) over the elements of hypothesis space (each hypothesis is an expert), and individually bounds the risk of specific hypotheses in terms of this prior. Of course, the PAC setting we are dealing with is not online, and we do not care about mistake bounds (we will make the connection between weighted majority and PAC-Bayes explicit later using the work of Langford & Shawe-Taylor (2002)).

However, the intuition that we should try to learn a stochastic hypothesis is fruitful: It enables us to end up with much tighter generalization bounds because we bound the true error rate over a *distribution* of hypotheses rather than the supremum (as we would in uniform learning) (Langford & Caruana (2001)). We want to consider the average case, not the worst case. In particular, as Langford & Caruana (2001) puts it, we can think of PAC-Bayes as re-distributing the work between theorist and experimenter, with more load placed upon the experimenter. It is easier for the theorist to prove better bounds given access to randomness, but the experimenter must now deal with a stochastic predictor function.

### 2.2.3 PAC-Bayes Generalization Bound

We follow the treatment in Boucheron et al. (2005) for explaining the non-uniform learning bounds which are possible with PAC-Bayes. We associate each hypothesis class with a prior $\pi$ (a probability measure) and a probability measure $\rho$, which defines the stochastic distribution over the hypothesis space. First, consider that the case where $\rho$ does not depend on the data is easy: The expected empirical risk with respect to $\rho$ is a sum of independent random variables whose expectation is the expected population risk with respect to $\rho$, and thus one can apply Hoeffding directly.

The other case, where $\rho$ does depend on the data, is harder. We proceed by performing a modified union bound.

The prior assigns weight $\pi(f)$ for each $f \in \mathcal{F}$, such that $\sum_{f\in\mathcal{F}} \pi(f) = 1$. In this particular case, one can apply a *weighted union bound*:

$$\begin{aligned}\mathbb{P}\left\{\exists f \in \mathcal{F} : \mathcal{R}(f) - \mathcal{R}_n(f) \geq \sqrt{\frac{\log(1/\pi(f)\delta)}{2n}}\right\} &\leq \sum_{f\in\mathcal{F}} \mathbb{P}\left\{\mathcal{R}(f) - \mathcal{R}_n(f) \geq \sqrt{\frac{\log(1/\pi(f)\delta)}{2n}}\right\} \\ &\leq \sum_{f\in\mathcal{F}} \pi(f)\delta = \delta\end{aligned} \tag{2}$$

the idea being that we pick the parameter $t$ in Hoeffding's inequality as a *function of $f$*. This in turn leads to the following bound:

**Lemma 2.10.** *PAC-Bayes Generalization Bound for individual $f$.*
*With probability at least $1 - \delta$, for $0 < \delta < 1$,*

$$\forall f \in \mathcal{F}, \mathcal{R}(f) - \mathcal{R}_n(f) \leq \sqrt{\frac{\log(1/\pi(f)) + \log(1/\delta)}{2n}}$$

Notably, the error bound for each $f$ now depends on the prior weight for that $f$! We can now take expectation with respect to $\rho$ and apply Jensen's inequality. Let us quickly recall the definitions of KL-divergence and entropy:

**Definition 2.11.** Entropy.
For probability distribution $\rho$,

$$H(\rho) = \sum_{f\in\mathcal{F}} -\rho(f)\log\rho(f)$$

**Definition 2.12.** KL-Divergence.
For probability distributions $\rho, \pi$,

$$D(\rho\|\pi) = \sum_{f\in\mathcal{F}} \rho(f)\log(\frac{\rho(f)}{\pi(f)})$$

Then, noting that $\mathbb{E}_\rho\left[\log(1/\pi(f))\right] = D(\rho\|\pi) + H(\rho)$, we get the following theorem.

**Theorem 2.13.** *PAC-Bayes Bound (Inefficient).*
*With probability at least $1 - \delta$, for $0 < \delta < 1$,*

$$\forall \rho, \rho(\mathcal{R}(f)) - \rho(\mathcal{R}_n(f)) \leq \sqrt{\frac{D(\rho\|\pi) + H(\rho) + \log(1/\delta)}{2n}}$$

*where $D(\cdot\|\cdot)$ is the KL-Divergence and $H$ is the entropy.*

The dependence on the entropy term is in fact unnecessary, and an improvement essentially follows from the convex duality of relative entropy:

**Lemma 2.14.** *Convex-Duality of Relative Entropy.*
*For random variable $X_f$,*

$$\rho(X_f) \leq \inf_{\lambda > 0} \frac{\log \pi(e^{\lambda X_f}) + D(\rho \| \pi)}{\lambda}$$

We can thus bound the Applying $X_f = [\mathcal{R}(f) - \mathcal{R}_n(f)]_+^2$ and using Markov's inequality on $\mathbb{P}\left\{\pi(e^{\lambda X_f}) \geq \epsilon\right\}$ and then switching order of integration between $\pi$ and the expectation allows us to bound $\mathbb{E}\left[e^{\lambda[\mathcal{R}(f)-\mathcal{R}_n(f)]_+^2}\right] \leq 2n$ choosing $\lambda = 2n - 1$ and integrating. This gives us the bound

$$\mathbb{P}\left\{\pi(e^{\lambda X_f}) \geq \epsilon\right\} \leq \frac{2n}{\epsilon}$$

which gives us the following result after selecting $\epsilon = \frac{2n}{\delta}$:

**Theorem 2.15.** *PAC-Bayesian Bound.*
*With probability at least $1 - \delta$, for $0 < \delta < 1$,*

$$\forall \rho, \rho(\mathcal{R}(f)) - \rho(\mathcal{R}_n(f)) \leq \sqrt{\frac{D(\rho\|\pi) + \log(2n) + \log(1/\delta)}{2n-1}}$$

*where $D(\cdot\|\cdot)$ is the KL-Divergence and $H$ is the entropy.*

We can intuitively think of this bound as a refined union bound. There are some difficulties with using this bound however. As we found intuitvely in the Randomized Weighted Majority algorithm, we would get no benefit in cases where the distribution was very peaked on given experts. The same is the case here: When $\rho$ is concentrated on a single function, this yields the deterministic case and the complexity of the bound blows up (for instance, if the prior were uniform, we suddenly get a term proportional to the size of the function class in the numerator). It is possible to fix this by adding more non-uniformity and assumptions by allowing the prior to depend on the data. It is also possible to have a PAC-Bayes take on the generic chaining mentioned earlier. Essentially, we look at the $\epsilon$-covers for all scales of $\epsilon$ and apply PAC-Bayes bounds at each level.

## 2.3 Margin Theory

The margin is a quantity that imposes constraints on both the function class and the data simultaneously, and is an answer to our third issue with uniform learning. We follow the introduction to margins given by Boucheron et al. (2005).

There are two main issues which we will see the margin helps us solve:

(a) In uniform learning, we require the VC dimension to be small to keep variance small. However, limiting our hypothesis class may pose a problem with bias (there may be no good $h \in \mathcal{H}$).

(b) Tangentially for us, ERM optimization is hard. We would like a way to approximate this quantity in a computationally efficient manner.

These concerns motiviate the introduction of a *cost function*, mentioned briefly before. We have been so far considering classifiers $h \in \mathcal{H}$ which are functions which map from the input space to $\{\pm 1\}$, and we have evaluated the risk directly in terms of $\mathbb{E}_{\mathcal{D}}\left[\mathbf{1}\left\{h(X_i) \neq Y_i\right\}\right]$. Now we consider the relaxation of this notion.

First, consider a hypothesis space $\mathcal{G}$ whose elements no longer map to $\{\pm 1\}$, but now map to the reals. Then, consider the hypothesis space $\mathcal{H}$ whose elements map to $\{\pm 1\}$ by *composing* $\mathcal{G}$ with the sign indicator function $\mathbb{R} \to \{\pm 1\}$

$$\mathbf{sgn}\,(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Formally, we are now considering hypothesis in the function space

$$\mathcal{H} = \mathbf{sgn}\,(\mathcal{G})$$

We write elements $h_g \in \mathcal{H}$ to associate them with their original function $g \in \mathcal{G}$. The risk (and its empirical version) remain the same under this definition, but we can now write the risk in terms of function class $\mathcal{G}$ instead of $\mathcal{H}$:

$$\mathcal{R}(h_g) = \mathbb{P}_{(X,Y)\sim\mathcal{D}}\left\{\mathbf{sgn}(g(X) \neq Y)\right\} \leq \mathbb{E}_{(X,Y)\sim\mathcal{D}}\left[\mathbf{1}\left\{g(X)Y < 0\right\}\right] = \mathbb{E}_{(X,Y)\sim\mathcal{D}}\left[\mathbf{sgn}\,(-g(X)Y)\right]$$

where the term $g(X)Y < 0$ iff the signs of $g(X)$ and $Y$ are different (i.e., we make an error). Given this definition of risk, we can formulate yet another function class $\mathcal{F}$ whose elements are functions $f_g : \mathcal{X} \times \{\pm 1\} \to \mathbb{R}$ where $f_g(X, Y) = -g(X)Y$, where $\mathcal{X}$ is the input space.

**Definition 2.16.** Cost function.
Considering the setting described above, we are ready to define **cost function** $\phi : \mathbb{R} \to \mathbb{R}^+$ such that $\phi(x) \geq \mathbf{1}\left\{x > 0\right\}$. We are then concerned with the **cost functional**

$$\mathcal{C}_\phi(g) = \mathbb{E}_{(X,Y)\sim\mathcal{D}}\left[\phi(-g(X)Y)\right]$$

and its empirical version $\mathcal{C}_{\phi_n}(g)$ (defined analogously to the empirical risk). We see that the cost functionals are upper bounds on the risk functionals $\mathcal{R}(g) \leq \mathcal{C}_\phi(g)$ and $\mathcal{R}_n(g) \leq \mathcal{C}_{\phi_n}(g)$. We can now associate with cost $\mathcal{C}$ the composed function class $\phi \circ \mathcal{F}$ in a manner analogous to the way the risk $\mathcal{R}$ is associated with function class $\mathcal{H}$.

**Example 2.17.** Examples of cost functions.
$\phi_1(x) = e^x, \phi_2(x) = \log(1 + e^x), \phi_3(x) = [1 + x]_+$ are all common choices for loss functions. Notably, they are all convex and correspond to widely used settings in machine learning. $\phi_1$ corresponds to boosting, $\phi_2$ corresponds to logistic regression, and $\phi_3$ corresponds to the hinge loss (used in SVM).

As you might expect, we can now prove a generalization bound for cost functions using Rademacher variables. The main point to focus on is that the Lipschitz constant $L_\phi$ of the cost will pop out thanks to the Rademacher calculus, as we alluded to before. Thus, we have an explicit notion of how smoothness affects sample complexity.

**Theorem 2.18.** *Generalization for cost functions.*
*Consider choosing a hypothesis $g_n \in \mathbb{G}$ after having seen dataset $\{(X_i, Y_i)\}_{i=1}^n$ drawn i.i.d. from data distribution $\mathcal{D}$. Suppose that $\phi(-g(x)y) \leq B$, a constant and let $L_\phi$ be the Lipschitz constant of the class. Then, with probability at leaset $1-\delta$,*

$$\mathcal{R}(g_n) \leq \mathcal{C}_{\phi_n}(g_n) + 2L_\phi \mathscr{R}(\mathcal{G}) + B\sqrt{\frac{2\log(1/\delta)}{n}}$$

*Proof.* We can see that the theorem essentially directly follows from the Rademacher complexity bounds we saw before in addition to an application of the contraction lemma. Specficially,

$$\begin{aligned} \mathcal{R}(g_n) &\leq \mathcal{C}_\phi(g_n) \\ &\leq \mathcal{C}_{\phi_n}(g_n) + \sup_{g\in\mathcal{G}}(\mathcal{C}_\phi(g) - \mathcal{C}_{\phi_n}(g)) \\ &\leq \mathcal{C}_{\phi_n}(g_n) + 2\mathscr{R}(\phi \circ \mathcal{F}) + B\sqrt{\frac{2\log(1/\delta)}{n}} \end{aligned} \tag{3}$$

The $B$ comes outside the square root because McDiarmid's inequality has concentration proportional to $\exp(-1/B^2)$, and we see from contraction that $L_\phi$ will come out of the Rademacher term. There is no issue converting $\mathscr{R}(\mathcal{F})$ back to $\mathscr{R}(\mathcal{H})$ as that was just the re-organization of what is essentially the same function space. Since the **sgn** function is Lipschitz bounded as well, we get the desired result. □

Now let us consider how we might use these facts to do better than the uniform learning setting. We saw with PAC-Bayes how it is a good idea to consider stochastic classifiers. In the case that it is possible to linearly combine members of the hypothesis class $\mathcal{G}$ that we choose, we can do something very similar and combine classifiers in a weighted voting scheme as we saw earlier.

**Definition 2.19.** Weighted Voting in Hypothesis Class.
Take linearly-combinable hypothesis class $\mathcal{G}$ and define for parameter $\lambda > 0$

$$\mathcal{G}_\lambda = \left\{ g(x) = \sum_{j=1}^N c_j g_j(x) : \sum_{j=1}^N |c_j| \leq \lambda, g_j \in \mathcal{G} \forall j \in [N] \right\}$$

Note that this is just the absolute convex hull which we saw before. Using the Rademacher calculus (and normalizing by $\lambda$), we see that we only pay an extra factor of $\lambda$ in the Rademacher complexity term for this class.

Moreover, we saw before that the Rademacher complexity is bounded by the VC dimension generalization error term. Thus, it follows that

**Lemma 2.20.** *Weighted voting generalization term bound.*

$$\mathscr{R}_n(\mathcal{G}_\lambda) \leq \lambda \mathscr{R}_n(\mathcal{G}) \leq \lambda\sqrt{\frac{2d\log(n+1)}{n}}$$

*where $d$ is the VC dimension of $\mathcal{G}$.*

We can plug this all in to get the full generalization bound:

**Theorem 2.21.** *Weighted Voting Classifier Generalization Bound.*

$$\mathcal{R}(g_n) \leq \mathcal{C}_{\phi_n}(g_n) + 2L_\phi\lambda\sqrt{\frac{2d\log(n+1)}{n}} + B\sqrt{\frac{2\log(1/\delta)}{n}}$$

Now we can finally motivate the definition of margin.

**Definition 2.22.** Margin.
Define the cost function

$$\phi_\gamma(x) = \begin{cases} 0 & \text{if } x \leq -\gamma \\ 1 & \text{if } x \geq 0 \\ 1 + x/\gamma & \text{otherwise} \end{cases}$$

for some fixed positive parameter $\gamma$, called the **margin**. Here, $B = 1$ and $L_{\phi_\gamma} = 1/\gamma$. Visually and intuitively, we can think of this cost function as a smoother version of the $0-1$ loss. We essentially pay linearly more cost the more positive distance we move away from $x = -\gamma$ until we get to $x = 0$, the boundary. If $x = g(x)y$, then we can interpret $x = 0$ as the point where our classifier starts misclassifying things. The term margin thus refers to this gap between correct and incorrect classification. Thus, we are now in some sense *sensitive to how sure we are about our classification.*

Armed with this definition, we can define an upper bound to the cost functional $\mathcal{C}_\phi$ called the **margin error**.

**Definition 2.23.** Empirical margin error.
For margin $\gamma$, we define

$$\mathcal{R}_n^\gamma(g) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{sgn}\left(-(g(X_i)Y_i - \gamma)\right)$$

Note that the margin error is increasing in the margin $\gamma$. We can intuitively think of the margin error as classifying the misclassified pairs $(X_i, Y_i)$ in addition to those which are correctly classified with small confidence.

Finally, we can connect the margin back to Rademacher complexity theory and plug in the Lipschitz constant to get a margin bound.

**Theorem 2.24.** *Margin Generalization Bound.*
*For any $\gamma > 0$ with probability at least $1 - \delta$,*

$$\mathcal{R}(g_n) \leq \mathcal{R}_n^{\gamma}(g_n) + 2\frac{\lambda}{\gamma}\sqrt{\frac{2d\log(n+1)}{n}} + \sqrt{\frac{2\log(1/\delta)}{n}}$$

*As $\gamma$ grows, the first term increases while the second decreases. This bound is very useful whenever a classifier has a small margin error for a large $\gamma$, and can be checked empirically.*

Thus, we have been able to combine notions of data dependence and assume things about the function class in order to come up with a generalization bound in terms of the margin.

**Remark 2.25.** Margin as a Lipschitz notion.
We can think intuitively about the notion of margin as a measure of inverse Lipschitzness, as we see in the definition of $\phi$. We are only dealing with $0-1$ classification here, but it is possible to define analogues in a whole host of other settings, including linear regression.

Consider linear regression, where we try to learn a hyperplane parametrized by normal vector $w$. We have that $\nabla_x\langle w, x\rangle = w^T$, and thus $\|\nabla_x\langle w, x\rangle\| = \|w\|_2$. if $\|w\|_2 \leq L$, then $L$ can function as the Lipschitz constant, and we thus get that the margin is proportional to $\frac{1}{L} \leq \frac{1}{\|w\|_2}$. The support vector machine (SVM) literature focuses on the classification with such hyperplanes, and can be thought of as a maximum margin technique: Part of the objective is to ensure there is a large margin from the decision boundary.

**Remark 2.26.** Connecting PAC-Bayes and Margins.
We already might have some notion that PAC-Bayes and margins are connected based on the way we ended up coming up with a margin bound. Langford & Shawe-Taylor (2002) demonstrate that building a stochastic classifier for getting a PAC-Bayes bound is always possible if some margin $\gamma$ holds for the dataset.

## 3 Explaining Boosting

The various non-uniform learning concepts we have discussed are very applicable to explaining the performance of popular successful classifiers. Back in the late 1990s, the hot algorithm of the day was AdaBoost, and the boosting framework in general. In particular, it shares several similarities with the deep network craze going on presently. The most common central theme between these two eras is the at-first baffling mystery: Boosting algorithms tended to fit the training data extremely well while still generalizing. Notably, we tend to see this is the case today on large image datasets with deep neural networks (refuting a sometimes common belief that the difficulty of deep learning is due to difficulty of optimization, at least in several common vision settings).

### 3.1 Freund & Schapire's Original Result

First, we briefly describe the AdaBoost algorithm Freund & Schapire (1997); Schapire et al. (1998), also making reference to Feng et al. (2012). It can be thought of as a generalization of the Randomized Weighted Majority Algorithm we saw earlier. We think of our experts as "weak learners", which satisfy PAC learning guarantees only for error $\epsilon \geq 1/2 - \gamma$ (that is, the best error they can get is $1/2 - \gamma$) for some $1/2 > \gamma > 0$. A weak learning algorithm is guaranteed to output a hypothesis which is a weak learner. We assume i.i.d. draws from the distribution of the data, unlike RWM. But very similarly to RWM, AdaBoost combines the weak hypotheses by summing up their probabilistic predictions, and then downweighting the weights of the incorrect hypotheses by a given factor. The full algorithm proceeds as follows, given a sequence of $n$ labeled examples $\{(X_i, Y_i)\}_{i=1}^N$, a distribution $\mathcal{D}$ over the examples, a weak learning algorithm, and an integer $T$ corresponding to a number of iterations (Schapire et al. (1998)):

(a) Let $D_1(i) = \mathcal{D}(i)$ for $i \in [N]$. We initialize weights according to a prior over the examples. Now proceed for $t = 1 \to T$:

(b) Give the weak learning algorithm the data set and distribution $D_t$ to generate a weak hypothesis $h_t$.

(c) Calculate the error at time $t$:

$$\epsilon_t = \mathbb{P}_{i \sim D_t}\{Y_i \neq h_t(X_i)\}$$

(d) Multiplicatively update the prior distribution over examples:

$$D_{t+1}(i) = D_t(i)\frac{\exp(-Y_i\alpha_t h_t(X_i))}{\sum_{i=1}^N D_t(i)\exp(-Y_i\alpha_t h_t(X_i))} = D_t(i)\frac{\exp(-Y_i\alpha_t h_t(X_i))}{2\sqrt{\epsilon_t(1-\epsilon_t)}}$$

where $\alpha_t = \frac{1}{2}\ln((1-\epsilon_t)/\epsilon_t)$.

(e) After $T$ iterations, output the hypothesis

$$h(x) = \frac{\sum_{t=1}^T \alpha_t h_t(x)}{\sum_{t=1}^T \alpha_t}$$

There are a few things which are immediately obvious: 1) Boosting takes advantage of the power of probability, 2) there is a clear notion of margin, 3) the connection to randomized weighted majority is in the multiplicative updates of the distribution. Also notably, AdaBoost can be viewed as minimizing the convex exponential cost function. Another thing that is interesting to note is that we add a $h_t$ at each time iteration, increasing the complexity of the hypothesis. The fact that we do a convex combination of these hypotheses should be very reminiscent of the Rademacher margin bounds we recently proved.

Furthermore note that at round $t$, AdaBoost places the most weight on the data points $(X, Y)$ such that the margin of the combined classifier $y\sum_{t'=1}^{t-1}\alpha_{t'}h_{t'}(x)$ is smallest (margin can be written as $yf(x) = \sum_{i:y=h_i(x)}\alpha_i - \sum_{i:y\neq h_i(x)}\alpha_i$).

To analyze this algorithm, Schapire et al. (1998) bounds $\mathbb{P}\{yh(x) \leq 0\}$ to show generalization.

**Theorem 3.1.** *Boosting Generalization (Schapire et al. (1998)).*
*For any $\delta > 0$ with probability at least $1-\delta$ over the random choice of training set with $n$ examples, for base classifier hypothesis class $\mathcal{H}$ with VC dimension $d$, every voting classifier $f\sum_i \alpha_i h_i$ satisfies:*

$$\mathbb{P}_{\mathcal{D}}\{yf(x) \leq 0\} \leq \inf_{\theta\in[0,1]}\left[\mathbb{P}_{\{x_i,y_i\}_{i=1}^n}\{yf(x) \leq \theta\} + \mathcal{O}\left(\frac{1}{\sqrt{n}}\sqrt{\frac{d\log^2(n/d)}{\theta^2} + \log(1/\delta)}\right)\right]$$

*This can be summarized as the generalization error being at most*

$$\mathbb{P}\{\text{margin}(x,y) \leq \theta\} + \tilde{O}\left(\sqrt{\frac{d}{n\theta^2}}\right)$$

*(from Reyzin & Schapire (2006)).*

In this theorem, we again see the choice of tradeoff for what margin $\theta$ to choose is at the heart of the problem. We can interpret the theorem as stating that if the voting classifier generates a good margin distribution (most training examples have large margins so that $\mathbb{P}\{yf(x) \leq \theta\}$ is small for not too small $\theta$), then the generalization error is also small.

We can view this generalization theorem in terms of a particular quantity: the margin distribution.

**Definition 3.2.** Margin Distribution.
The quantity $\mathbb{P}\{yf(x) \leq \theta\}$ is known as the **margin distribution**. It is a distribution over margin values $\theta$ since that probability is the fraction of training examples whose margin is at most $\theta$.

However, it is possible (see Feng et al. (2012)) to prove a generalization theorem which only depends on the minimum margin:

**Definition 3.3.** Minimum Margin.
The **minimum margin** of voting classifier $f$ is the smallest margin over the training examples. It can be represented by

$$\max_{\theta}\mathbb{P}\{yf(x) \leq \theta\} = 0$$

In fact, a paper by Breimen pointed out that the resulting generalization bound based on minimum margin distributions is sharper than the original margin bound which we saw here (Feng et al. (2012)). If $\theta$ in the Schapire theorem is taken to be the minimum margin, Breimen's theorem is much better. Thus, he claimed his bound implied that the minimum margin is what governs the generalization error. Yet, the minimum margin algorithm (called arc-gv) he developed (which essentially modifies the way $\alpha_t$ is set to take into account the minimum margin) performs much worse than AdaBoost, calling the margin explanation of boosting into doubt.

### 3.2 Why the Margin Explanation Still Holds

It turns out that the main reason Breimen was wrong is due to the fact that his algorithm was subtly increasing the complexity of the base classifiers, and thus of the VC dimension, in a manner that counteracted potential benefits from boosting (Reyzin & Schapire (2006)). Thus, it is truely the *margin distribution* which controls the generalization error of boosting (Gao & Zhou (2013)).

## 4 How do Neural Networks Generalize?

### 4.1 Rethinking Generalization

Zhang et al. (2017)

### 4.2 Margins to the Rescue

Bartlett et al. (2017)

### 4.3 PAC-Bayes

Neyshabur et al. (2017a)

### 4.4 What Remains: Missing Pieces

#### 4.4.1 Connections to Optimization

Neyshabur et al. (2015, 2016, 2017c)
Neyshabur et al. (2017b)

#### 4.4.2 Properties of the Data

## 5 Further work

### 5.1 Residual Nets == Boosting?

Huang et al. (2017)

### 5.2 Regression Based Efforts

## 6 Conclusion: Why do Deep Nets Generalize?

Here are some main takeaways:

(a) Always be aware of the limitations of your theoretical framework with respect to practice.

(b) Data matters for generalization power of neural nets.

(c) Don't forget the past!!

## References

Peter Bartlett, Dylan J. Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. 2017. URL `https://arxiv.org/pdf/1706.08498.pdf`. arXiv.

Avrim Blum. On-line algorithms in machine learning. *LNCS*, 1442, 1998.

Stéphane Boucheron, Olivier Bousquet, and Gábor Lugosi. Theory of classification: A survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323–375, November 2005.

Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. pp. 169–207, 2004.

Jufu Feng, Liwei Wang, Masashi Sugiyama, Cheng Yang, Zhi-Hua Zhou, and Chicheng Zhang. Boosting and margin theory. *Front. Electr. Electron. Eng*, 7:127–133, 2012.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

Wei Gao and Zhi-Hua Zhou. On the doubt about margin explanation of boosting. *Artificial Intelligence*, 203:1–18, October 2013. URL `https://arxiv.org/pdf/1009.3613.pdf`.

Furong Huang, Jordan Ash, John Langford, and Robert Schapire. Learning deep resnet blocks sequentially using boosting theory. 2017. URL `https://arxiv.org/pdf/1706.04964.pdf`. arXiv.

John Langford and Rich Caruana. (not) bounding the true error. 14, 2001. URL `https://papers.nips.cc/paper/1968-not-bounding-the-true-error`.

John Langford and John Shawe-Taylor. Pac-bayes and margins. pp. 439–446, 2002. URL `http://papers.nips.cc/paper/2317-pac-bayes-margins.pdf`.

Behnam Neyshabur, Ruslan Salakhutdinov, and Nathan Srebro. Path-sgd: Path-normalized optimization in deep neural networks. pp. 2422–2430, 2015. URL https://arxiv.org/pdf/1506.02617.pdf.

Behnam Neyshabur, Yuhuai Wu, Ruslan Salakhutdinov, and Nathan Srebro. Path-normalized optimization of recurrent neural networks with relu activations. 2016. URL https://arxiv.org/pdf/1605.07154.pdf.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. 2017a. URL https://arxiv.org/pdf/1707.09564.pdf. arXiv.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning. 2017b. URL https://arxiv.org/abs/1706.08947. arXiv.

Behnam Neyshabur, Royta Tomioka, Ruslan Salakhutdinov, and Nathan Srebro. Geometry of optimization and implicit regularization in deep learning. 2017c. URL https://arxiv.org/abs/1705.03071. arXiv.

Lev Reyzin and Robert Schapire. How boosting the margin can also boost classifier complexity. pp. 753–760, 2006. URL http://rob.schapire.net/papers/boost_complexity.pdf.

Robert E. Schapire, Yoav Freund, Peter Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26: 1651–1686, 1998.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, 2014.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires re-thinking generalization. 2017. URL https://arxiv.org/pdf/1611.03530.pdf.