EE660 Lecture Notes: Mathematical Foundations and Methods of Machine Learning

Stephen Tu

stephen.tu@usc.edu

Last Updated: November 5, 2024

Abstract

This set of lecture notes accompanies the Fall 2024 offering of EE660 at USC.

Note: This document is a continual work in progress and will be constantly updated throughout the semester. Be sure to frequently check for updates.

Contents

1 Supervised learning

We start our study by considering an unknown distribution $\mathcal{D}$ over an event space $\mathsf{Z} = \mathsf{X} \times \mathsf{Y}$. Here, the space X denotes the *covariates*, and we will typically assume for simplicity that X is a subset of Euclidean space. On the other hand, the space Y denotes the *labels*. We will let $p(x, y) = p(x)p(y \mid x)$ denote the joint density function over Z.[1]

We first consider *binary classification*, where $\mathsf{Y} = \{-1, +1\}$. The binary classification problem is: given i.i.d. *training examples* $S_n := \{(x_1, y_1), \ldots, (x_n, y_n)\}$ from $\mathcal{D}$, learn a predictor $\hat{f}_n : \mathsf{X} \mapsto \mathsf{Y}$, so that ideally on unseen *test examples* $(x, y) \sim \mathcal{D}$, we have that $\hat{f}_n(x) = y$.

1.1 What form should the predictor take?

In order to propose an algorithm to learn the predictor $\hat{f}_n$, we need to specify two things:

(a) What form should the predictor function $\hat{f}_n$ take?

(b) What algorithm should we use to learn its representation?

To shed some insight into question (a), let us set forth a framework to study what an *optimal* predictor looks like. Let $\ell : \mathsf{Y} \times \mathsf{Y} \mapsto \mathbb{R}$ be a loss function, where we think of $\ell(\hat{y}, y)$ as the cost of predicting $\hat{y}$ when the true label is y. For a concrete example, think of $\ell(\hat{y}, y) = \mathbf{1}\{\hat{y} \neq y\}$, which is the *zero-one loss* or the *mistake loss*.

With this setup, we can now pose an optimization problem to compute the best predictor:

$$\min_{f:\mathsf{X}\to\mathsf{Y}} \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f(x), y)]. \tag{1.1}$$

(Note for what follows, we will usually drop the subscript under the expectation $\mathbb{E}$. However, please do not be afraid to ask if it is ever not clear what random variables we are taking expectations and/or probabilities with respect to.) Above, the optimization is taken over all (measurable) functions mapping covariates X to labels Y. It may seem at first that this is an impossible problem to solve, but it actually has a very simple solution.

Proposition 1.1. *Suppose the loss ℓ is* proper*, i.e.,*

$$\ell(1, -1) > \ell(-1, -1), \quad \textit{and} \quad \ell(-1, 1) > \ell(1, 1).$$

A solution $\hat{f} : \mathsf{X} \to \mathsf{Y}$ of Equation (1.1) takes on the form:[2]

$$\hat{f}(x) = \operatorname{sgn}\left\{p(y = +1 \mid x) - \frac{\ell(1, -1) - \ell(-1, -1)}{\ell(-1, 1) - \ell(1, 1)} p(y = -1 \mid x)\right\}. \tag{1.2}$$

Proof. By iterated expectations,

$$\mathbb{E}[\ell(f(x), y)] = \mathbb{E}[\mathbb{E}[\ell(f(x), y) \mid x]].$$

Let us now study the inner quantity, conditioned on x. The key is that $f(x)$, once conditioned on x, is no longer random (the jargon I will often use is that $f(x)$ is "x-measurable"). Hence, the inner conditional expectation decomposes very cleanly:

$$\mathbb{E}[\ell(f(x), y) \mid x] = \ell(f(x), 1)p(y = +1 \mid x) + \ell(f(x), -1)p(y = -1 \mid x).$$

Hence, an optimal $f(x)$, *for this value of x* that we are conditioning on, should pick either $+1$ or -1 to minimize the RHS of the above equation. That is, abbreviating $p_1 = p(y = 1 \mid x)$ and $p_{-1} = p(y = -1 \mid x)$:

$$f(x) = \arg\min_{y\in\{\pm 1\}} \{\ell(y, 1)p_1 + \ell(y, -1)p_{-1}\}$$

[1] We will not be overly concerned with measure-theoretic concerns in this course.

[2] The sgn function is $\operatorname{sgn}(a) := \mathbf{1}\{a \geqslant 0\} - \mathbf{1}\{a < 0\}$. Note the tie at $a = 0$, such that $\operatorname{sgn}(0) = 1$, is broken arbitrarily.

$$= \begin{cases} +1 & \text{if } \ell(-1,1)p_1 + \ell(-1,-1)p_{-1} \geqslant \ell(1,1)p_1 + \ell(1,-1)p_{-1} \\ -1 & \text{otherwise.} \end{cases}$$

The result now follows by re-arranging the expression above, relying on the properness of the loss ℓ to divide by $\ell(-1,1) - \ell(1,1)$. □

Let us now consider a special case of Equation (1.2) where $\ell(\hat{y}, y) = \mathbf{1}\{\hat{y} \neq y\}$ is the zero-one loss. In this case, the predictor $\hat{f}$ reduces to the *maximum a-posterior* (MAP) predictor:

$$\hat{f}(x) = \arg\max_{y \in \mathsf{Y}} p(y \mid x). \tag{1.3}$$

Note that by Bayes' rule, we have that:

$$p(y \mid x) = \frac{p(y)p(x \mid y)}{p(x)}.$$

Hence, under a uniform prior on the labels, i.e., $p(y = 1) = p(y = -1) = 1/2$, we have that $\hat{f}$ is also equal to the *maximum-likelihood estimator* (MLE):

$$\hat{f}(x) = \arg\max_{y \in \mathsf{Y}} p(x \mid y).$$

Exercise 1.2. Suppose we are in a multi-class classification setting, so that $\mathsf{Y} = \{1, \ldots, K\}$. Show that the optimal predictor $\hat{f} : \mathsf{X} \mapsto \mathsf{Y}$ minimizing the zero-one loss is still the MAP predictor:

$$\hat{f}(x) = \arg\max_{y \in \mathsf{Y}} p(y \mid x).$$

1.2 Perceptron

We now turn to answer the second question (b), what algorithm do we use to learn the representation of a decision function $f : \mathsf{X} \mapsto \mathsf{Y}$. We will start from a very simple prototypical algorithm, called the Perceptron, and understand its various properties. While the Perceptron is a very simple model, many interesting properties will arise that will inform our later study.

The Perceptron starts by making a modeling assuming regarding the form of the predictor $\hat{f}$. Conceptually, every predictor $\hat{f}$ (whether optimal or not) induces a *partition* of X (which we now assume to be a subset of $\mathbb{R}^d$):

$$T_1 := \{x \in \mathsf{X} \mid \hat{f}(x) = 1\}, \quad \text{and} \quad T_{-1} := \{x \in \mathsf{X} \mid \hat{f}(x) = -1\}.$$

(In fact, there is a one-to-one correspondence between partitions of X and predictors $\hat{f}$.) In the Perceptron, it is posited that the optimal predictor induces a *linear* partition of X: that is, we cut X in half by a hyperplane (that we will assume crosses through the origin for convenience), and assign the label $+1$ to one side of the hyperplane, and -1 to the other. Mathematically:

$$f_w(x) = \operatorname{sgn}\{\langle w, x \rangle\}, \quad w \in \mathbb{R}^d.$$

Learning here then refers to learning the weights w which parameterize the separating hyperplane. The Perceptron prescribes the following procedure to learn these weights.

We will now study the behavior of the Perceptron on *linearly separable* data.

Definition 1.3. *A dataset $S_n = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathsf{Z}$ is* linearly separable *if there exists a non-zero $w \in \mathbb{R}^d$ such that:*

$$\forall i \in \{1, \ldots, n\}, \quad y_i = \operatorname{sgn}\{\langle w, x_i \rangle\}.$$

Such a non-zero $w \in \mathbb{R}^d$ satisfying the above condition is called a linear separator.

Algorithm 1 The Perceptron

Input: Dataset $S_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.

Set $w = 0$.
while `true` **do**
  Let $\mathcal{I}$ be a (possibly random) permutation of $\{1, \ldots, n\}$.
  Set `mistake = false`.
  for $i \in \mathcal{I}$ **do**
    if $y_i \langle w, x_i \rangle \leqslant 0$ **then**
      Set $w \leftarrow w + y_i x_i$.
      Set `mistake = true`.
    end if
  end for
  if `not mistake` **then**
    return w.
  end if
end while

Exercise 1.4. Which of the following $n = 2$ datasets with $\mathsf{X} = \mathbb{R}^2$ are linearly separable?

(a) $S_2 = \{((0, 1), +1), ((0, -1), +1)\}$.

(b) $S_2 = \{((0, 1), -1), ((0, -1), -1)\}$.

(c) $S_2 = \{((0, 0), +1), ((0, 1), -1)\}$.

Next, we will define the notion of a *margin*, which quantifies to what degree a dataset is linearly seperable. Before we do this, let S be a subset of Euclidean space. The distance function $\operatorname{dist}(x, S)$ is defined as:

$$\operatorname{dist}(x, S) = \inf\{\|x - z\| \mid z \in S\}.$$

Also, for a vector $w \in \mathbb{R}^d$, we define its hyperplane H_w as:

$$H_w := \{x \in \mathbb{R}^d \mid \langle x, w \rangle = 0\}.$$

We now have the requisite notation to define the notion of margin.

Definition 1.5. *Let S_n be a linearly separable dataset (cf. Definition 1.3), and let $W \subset \mathbb{R}^d$ denote the set of linear separators of S_n. Given $w \in W$, the* margin of w, *denoted $\gamma(w, S_n)$, is defined as:*

$$\gamma(w, S_n) := \min_{i \in \{1, \ldots, n\}} \operatorname{dist}(x_i, H_w). \tag{1.4}$$

Furthermore, the margin $\gamma(S_n)$ *is defined as the maximum margin achievable by any linear separator $w \in W$:*

$$\gamma(S_n) := \sup_{w \in W} \gamma(w, S_n) = \sup_{w \in W} \min_{i \in \{1, \ldots, n\}} \operatorname{dist}(x_i, H_w). \tag{1.5}$$

A max-margin *separator is any linear separator $w_\star \in W$ such that $\gamma(w_\star, S_n) = \gamma(S_n)$.*

The idea behind margin is to assign a quantity to a linear separator which measures how "robust" it is. (Draw some pictures in class). The following exercises check your understanding of margin.

Exercise 1.6. Show that the margin defined in Equation (1.5) satisfies the following equalities:

$$\gamma(S_n) = \sup_{w \in W} \min_{i \in \{1, \ldots, n\}} \frac{|\langle x_i, w \rangle|}{\|w\|} = \sup_{\|w\| = 1} \min_{i \in \{1, \ldots, n\}} y_i \langle x_i, w \rangle.$$

Exercise 1.7. Suppose S_n is linearly separable. Consider the alternative definition of margin, which does not take into account whether or not a hyperplane separates the dataset:

$$\gamma'(S_n) = \sup_{w \neq 0} \min_{i \in \{1,\dots,n\}} \operatorname{dist}(x_i, H_w).$$

Is it possible to construct an S_n such that $\gamma'(S_n) > \gamma(S_n)$? If so, provide one construction. If not, prove it is not possible.

1.2.1 Mistake bound

With this notion of margin, we are able to prove our first non-trivial result regarding Perceptron, which is its well-known *mistake bound*. While this is a classic result, we will follow the derivations based on the presentation in Hardt and Recht [2022, Ch. 3].

To start, we will define a sequences of weights $w_0, w_1, \dots$ recursively, which captures the main Perceptron update. Let $i_0, i_1, \dots$ denote an *arbitrary* sequence of indices in $\{1, \dots, n\}$ (meaning, these indices could be repeating, etc.). Our sequence $w_0, w_1, \dots$ is defined as:

$$w_{t+1} = w_t + y_{i_t} x_{i_t} \mathbf{1}\{y_{i_t} \langle w_t, x_{i_t} \rangle \leqslant 0\}, \quad w_0 = 0. \tag{1.6}$$

Note that our notation makes implicit the dependence of the iterates $\{w_t\}$ on the indices $\{i_t\}$. In the sequel, it will be clear from context how the indices $\{i_t\}$ are specified.

Lemma 1.8 (Perceptron mistake bound)**.** *Let S_n be linearly separable. Consider the Perceptron sequence $\{w_t\}_{t \geqslant 0}$ defined in Equation (1.6). Let $m_t := \mathbf{1}\{y_{i_t} \langle w_t, x_{i_t} \rangle \leqslant 0\}$ denote the indicator of whether a mistake occured at time t, and let $M_t := \sum_{s=0}^{t-1} m_t$ denote the culmulative number of mistakes made up to time t. We have that for all $t \in \mathbb{N}$:*

$$M_t \leqslant \frac{\max_{i \in \{1,\dots,n\}} \|x_i\|^2}{\gamma^2(S_n)}. \tag{1.7}$$

Before proceeding to the proof, it is worth reflecting on the nature of this result. It is quite remarkable in that it is *dimension independent*. Furthermore, since the RHS is independent of t, this result implies there exists a t such that for all $t' \geqslant t$, we have $m_{t'} = 0$ (why?). In other words, the Perceptron eventually stops making mistakes. Furthermore, since this result holds for *any* sequence $i_0, i_1, \dots$, by setting the sequence $\{i_t\}$ to be repeated copies of $\{1, \dots, n\}$, this ensures that eventually the Perceptron indeed learns a linear separator (why?).

Proof of Lemma 1.8. Put $B := \max_{i \in \{1,\dots n\}} \|x_i\|^2$. Fix a t and suppose that $m_t = 1$. Then, by expanding the square:

$$\begin{aligned} \|w_{t+1}\|^2 &= \|w_t + y_{i_t} x_{i_t}\|^2 = \|w_t\|^2 + 2 y_{i_t} \langle w_t, x_{i_t} \rangle + \|x_{i_t}\|^2 \\ &\leqslant \|w_t\|^2 + 2 y_{i_t} \langle w_t, x_{i_t} \rangle + B \leqslant \|w_t\|^2 + B. \end{aligned}$$

The key here is the last inequality, which is a consequence of $m_t = 1$, or equivalently $y_{i_t} \langle w_t, x_{t_i} \rangle \leqslant 0$. On the other hand, if $m_t = 0$, then $\|w_{t+1}\|^2 = \|w_t\|^2$ by definition. Hence, for every t:

$$\|w_{t+1}\|^2 \leqslant \|w_t\|^2 + B m_t.$$

Unrolling this recursion to $t = 0$ and using the definition of M_t yields for every t:

$$\|w_t\|^2 \leqslant B \sum_{s=0}^{t-1} m_t = B M_t.$$

Next, let $w_\star$ denote a unit norm max-margin linear separator, i.e., a vector such that (cf. Equation (1.5)):

$$\gamma(S_n) = \min_{1 \leqslant i \leqslant n} |\langle x_i, w_\star \rangle|.$$

Now, suppose that $m_t = 1$, and hence, since $w_\star$ linearly separates S_n,

$$\langle w_\star, w_{t+1} - w_t \rangle = y_{i_t} \langle w_\star, x_{i_t} \rangle = |\langle w_\star, x_{i_t} \rangle| \geqslant \gamma(S_n).$$

Next, we use a *telescoping sum*, which is a trick often used in optimization proofs. The telescoping sum simply states (recall that $w_0 = 0$):

$$w_t = \sum_{k=1}^{t} (w_k - w_{k-1}) = \sum_{k=1}^{t} (w_k - w_{k-1}) m_{k-1}.$$

While this may appear tautological, it is actually quite useful, since:

$$\|w_t\| \geqslant \langle w_\star, w_t \rangle = \sum_{k=1}^{t} \langle w_\star, w_k - w_{k-1} \rangle m_{k-1} \geqslant \gamma(S_n) \sum_{k=1}^{t} m_{k-1} = \gamma(S_n) M_t.$$

Above, the first inequality holds by Cauchy-Schwarz (recall that $w_\star$ is unit norm). We now have upper and lower bounds on $\|w_t\|$, from which we conclude:

$$\gamma^2(S_n) M_t^2 \leqslant \|w_t\|^2 \leqslant B M_t \implies M_t \leqslant \frac{B}{\gamma^2(S_n)}.$$

□

Exercise 1.9. Suppose Algorithm 1 is called with a linearly separable dataset S_n. Let B bound the covariate norm, i.e., $B = \max_{i \in \{1,\ldots,n\}} \|x_i\|^2$. Show that Algorithm 1 terminates with a solution $w \in \mathbb{R}^d$, which linearly separates S_n, in at most $B/\gamma^2(S_n)$ passes through S_n.

1.2.2 Generalization bound

The mistake bound from Section 1.2.1 gives us a way to prove a generalization bound about Perceptron, on *unseen* instances. We will utilize a clever technique known as leave-one-out analysis. We need one more piece of notation before we proceed. Given a linearly separable dataset S_n, we let $w(S_n)$ denote the limiting value of the Perceptron sequence $\{w_t\}$ defined in Equation (1.6), where we fix the indices $\{i_t\}$ to repeatedly cycle through $\{1, \ldots, n\}$, i.e., $i_t = (t \mod n) + 1$. Note that $w(S_n)$ is well-defined by Lemma 1.8. For what follows, we will assume that the distribution $\mathcal{D}$ is linearly separable, meaning that almost surely, for any $n \in \mathbb{N}_+$, S_n sampled i.i.d. from $\mathcal{D}$ is linearly separable.

Lemma 1.10 (Perceptron generalization bound)**.** *Suppose that $\mathcal{D}$ is linearly separable. Also, suppose that $\|x\| \leqslant B$ almost surely when $x \sim p(x)$. Then, letting $(x, y) \in \mathsf{Z}$ be drawn independently from S_n,*

$$\mathbb{P}\{y \langle w(S_n), x \rangle \leqslant 0\} \leqslant \frac{B}{n+1} \mathbb{E}\left[\frac{1}{\gamma^2(S_{n+1})}\right]. \tag{1.8}$$

Note that the probability on the LHS above is taken jointly over S_n and x.

Proof. We will construct an equivalent probability space as follows. Let $S_{n+1} = \{z_1, \ldots, z_{n+1}\}$ be $n+1$ i.i.d. draws from $\mathcal{D}$, and let $S_{n+1}^{-k} := \{z_1, \ldots, z_{k-1}, z_{k+1}, \ldots z_{n+1}\}$ be the dataset where the k-th point is left out. Note that by exchangability, for any k,

$$\mathbb{P}\{y \langle w(S_n), x \rangle \leqslant 0\} = \mathbb{P}\{y_k \langle w(S_{n+1}^{-k}), x_k \rangle \leqslant 0\}.$$

Hence we can average over all indices to conclude:

$$\mathbb{P}\{y\langle w(S_n), x\rangle \leqslant 0\} = \frac{1}{n+1}\sum_{k=1}^{n+1} \mathbb{P}\{y_k\langle w(S_{n+1}^{-k}), x_k\rangle \leqslant 0\}.$$

(This type of relabeling argument is common to leave-one-out analyses, and we will see it come up again later when we study algorithmic stability.)

Now, let $I_m \subset \{1, \dots, n+1\}$ denote the indices for which the Perceptron on S_{n+1} made a mistake at any point during execution, i.e.,

$$I_m := \{(t \mod (n+1)) + 1 \mid m_t = 1,\, t \in \mathbb{N}\}.$$

Let $k \in \{1, \dots, n+1\}$, and suppose $k \notin I_m$. Then, we have the key equality:

$$w(S_{n+1}^{-k}) = w(S_{n+1}).$$

It is worth taking a moment to think why this equality is true. If $k \notin I_m$, then this means that each time Perceptron on S_{n+1} cycles through the index k, it makes no update. Hence, it is equivalent to running Perceptron on S_{n+1}^{-k}, where the k-th datapoint is altogether removed.

Now here is the final trick. Again let $k \notin I_m$. Then we have:

$$0 < y_k\langle w(S_{n+1}), x_k\rangle = y_k\langle w(S_{n+1}^{-k}), x_k\rangle.$$

(Check your understanding: why is the first inequality is true?) Therefore, chaining these arguments together and applying the mistake bound from Lemma 1.8,

$$\begin{aligned}\sum_{k=1}^{n+1} \mathbf{1}\{y_k\langle w(S_{n+1}^{-k}), x_k\rangle \leqslant 0\} &= \sum_{k\in I_m} \mathbf{1}\{y_k\langle w(S_{n+1}^{-k}), x_k\rangle \leqslant 0\} + \sum_{k\notin I_m} \mathbf{1}\{y_k\langle w(S_{n+1}^{-k}), x_k\rangle \leqslant 0\} \\ &= \sum_{k\in I_m} \mathbf{1}\{y_k\langle w(S_{n+1}^{-k}), x_k\rangle \leqslant 0\} \leqslant |I_m| \leqslant \frac{B}{\gamma^2(S_{n+1})}.\end{aligned}$$

The result now follows by taking expectations on both sides of the above inequality. □

Remark 1.11. Recall in the definition of $w(S_n)$, we specifically fixed the indices $\{i_t\}$ to repeatedly cycle through $\{1, \dots, n\}$. Where in the proof of Lemma 1.10 did we use this assumption? What modifications to how the sequence $\{i_t\}$ is generated can you think of, which allow the proof to still go through?

Remark 1.12. Lemma 1.10 states that the margin mistake error $\mathbb{P}\{y\langle w(S_n), w\rangle \leqslant 0\}$ decreases at a rate of $O(1/n)$. In the literature, this is referred to as a *fast rate*, since it is faster than the usual $O(1/\sqrt{n})$ rate which arises out of typical concentration of measure arguments (which we will see in a few lectures). This fast-rate is not to be taken for granted: obtaining a fast-rate in general settings typically involves a much more non-trivial argument.

Interpreting the margin mistake bound. How does the margin mistake bound from Lemma 1.10 relate to the zero-one loss $\mathbb{P}\{y \neq \operatorname{sgn}(\langle w(S_n), x\rangle)\}$? Let us abbreviate $w = w(S_n)$. We can upper bound the zero-one loss by the margin mistake bound via the following argument:

$$\begin{aligned}\{y \neq \operatorname{sgn}(\langle w, x\rangle)\} &= \{y = 1, \langle w, x\rangle < 0\} \cup \{y = -1, \langle w, x\rangle \geqslant 0\} \\ &= \{y = 1, y\langle w, x\rangle < 0\} \cup \{y = -1, y\langle w, x\rangle \leqslant 0\} \\ &\subset \{y\langle w, x\rangle \leqslant 0\}.\end{aligned}$$

(Note the asymmetry in the cases is due to the tie-breaking choice made in our definition of the sgn function.) Hence by monotonicity of probability combined with Lemma 1.10:

$$\mathbb{P}\{y \neq \operatorname{sgn}(\langle w(S_n), x\rangle)\} \leqslant \mathbb{P}\{y\langle w(S_n), x\rangle \leqslant 0\} \leqslant \frac{B}{n+1}\mathbb{E}\left[\frac{1}{\gamma^2(S_{n+1})}\right].$$

1.3 Support Vector Machines, kernel machines, and random features

In our study of the Perceptron, we saw (cf. Lemma 1.8) that on linearly separable data, the Perceptron algorithm converges to a linear separator. However, as we will see, the Perceptron does not necessarily converge to a *max-margin* separator. Can we design an algorithm which is guaranteed to recovery a max-margin separator on the training data? Indeed we can, which is precisely the motivation behind the class Support Vector Machine (SVM) algorithm.

Proposition 1.13. *Given a linearly separable dataset S_n, the Perceptron algorithm Algorithm 1 does not necessarily converge to a max-margin separator.*

Proof. Consider the following $n = 2$ dataset for some $\alpha \in (0, 1)$:

$$S_2 = \{((0, 1), +1), ((1, -\alpha), -1)\}.$$

Suppose that the index sequence is $i_0 = 1, i_1 = 2, i_2 = 1, i_3 = 2, \ldots$. Observe that the first update will set $w_1 = (0, 1)$. However, observe that:

$$y_1\langle w_1, x_1\rangle = 1, \quad y_2\langle w_1, x_2\rangle = \alpha.$$

Hence, w_1 satisfies $y_i\langle w_1, x_i\rangle > 0$ for $i \in \{1, 2\}$, and hence the Perceptron algorithm has already converged and will make no more updates. Furthermore, since w_1 is unit norm, we can read-off the margin $\gamma(w_1, S_2)$ of this predictor w_1 as $\gamma(w_1, S_2) = \alpha$. On the other hand, as $\alpha \to 0$, the max-magin $\gamma(S_2)$ approaches $1/\sqrt{2}$. Hence, w_1 is not a max-margin separator, and hence Perceptron is not guaranteed to converge to one. □

1.3.1 Support Vector Machine

We now design an algorithm which is ensured to recover a max-margin separator. The key is to realize that the max-margin condition can be written as a convex problem. In particular, by Exercise 1.6:

$$\begin{aligned}
\max_{w \in W} \gamma(w, S_n) &= \max_{\|w\|=1} \min_{i \in [n]} y_i\langle w, x_i\rangle \\
&= \max_{\|w\|=1, \gamma > 0} \gamma \text{ s.t. } y_i\langle w, x_i\rangle \geqslant \gamma, i \in [n] \\
&= \max_{\|w\|=1, \gamma > 0} \frac{1}{\|w/\gamma\|} \text{ s.t. } y_i\langle w/\gamma, x_i\rangle \geqslant 1, i \in [n] \\
&= \max_{w \neq 0} \frac{1}{\|w\|} \text{ s.t. } y_i\langle w, x_i\rangle \geqslant 1, i \in [n].
\end{aligned}$$

On the other hand,

$$\arg\max_{w \neq 0} \left\{ \frac{1}{\|w\|} \text{ s.t. } y_i\langle w, x_i\rangle \geqslant 1, i \in [n] \right\} = \arg\min_{w} \left\{ \frac{1}{2}\|w\|^2 \text{ s.t. } y_i\langle w, x_i\rangle \geqslant 1, i \in [n] \right\}. \tag{1.9}$$

The RHS of (1.9) is the primal form of the (hard-margin) Support Vector Machine (SVM). The SVM is a convex optimization problem—in particular a quadratic program (QP)—which means it can be solved efficiently using convex optimization solvers such as `cvxopt`.

Support vectors. Consider the primal SVM (1.9). We now argue that there must exists indices i such that the constraint $y_i\langle w, x_i\rangle \geqslant 1$ is satisfied with *equality* for the optimal solution. The indices i such that $y_i\langle w, x_i\rangle = 1$ are referred to as the *support vectors*, since these points (and only these points) determine the placement of the decision boundary. To see that such support vectors must exist, suppose towards a contradiction that $\min_{i \in [n]} y_i\langle w, x_i\rangle = C > 1$. But, we can now form a new solution $\bar{w} = w/C$, for which $\min_{i \in [n]} y_i\langle \bar{w}, x_i\rangle = 1$; in other words $\bar{w}$ is a feasible solution for (1.9). Furthermore, $\|\bar{w}\| = \|w\|/C < \|w\|$ since $C > 1$, which contradicts the optimality of w.

SVM Duality. Let us now explore what happens when we take the Lagrangian dual of the SVM optimization problem (1.9). For a brief review of Lagrangian duality, see [Boyd and Vandenberghe, 2004, Chapter 5]. If you are not familiar with convex optimization, you may simply take the following derivation for granted.

Introducing Lagrange multipliers λ_i for each of the constraints $y_i\langle w, x_i\rangle \geqslant 1$, we define the Lagrangian:

$$\mathcal{L}(w, \{\lambda_i\}) := \frac{1}{2}\|w\|^2 + \sum_{i=1}^{n} \lambda_i(1 - y_i\langle w, x_i\rangle).$$

By strong duality for quadratic programming:

$$\begin{aligned}\min_w \left[\frac{1}{2}\|w\|^2 \text{ s.t. } y_i\langle w, x_i\rangle \geqslant 1, i \in [n]\right] &= \min_w \max_{\lambda_i \geqslant 0} \left[\frac{1}{2}\|w\|^2 + \sum_{i=1}^{n} \lambda_i(1 - y_i\langle w, x_i\rangle)\right] \\ &= \min_w \max_{\lambda_i \geqslant 0} \mathcal{L}(w, \{\lambda_i\}) \\ &= \max_{\lambda_i \geqslant 0} \min_w \mathcal{L}(w, \{\lambda_i\}).\end{aligned}$$

Above, the last equality is Slater's condition for strong duality in quadratic programming; we know there exists a feasible point by the assumption that the data is linearly separable. Continuing from above,

$$\begin{aligned}\max_{\lambda_i \geqslant 0} \min_w \mathcal{L}(w, \{\lambda_i\}) &= \max_{\lambda_i \geqslant 0} \left[\sum_{i=1}^{n} \lambda_i - \frac{1}{2}\left\|\sum_{i=1}^{n} \lambda_i y_i x_i\right\|^2\right] \qquad \text{setting } w = \sum_{i=1}^{n} \lambda_i y_i x_i \\ &= \max_{\lambda_i \geqslant 0} \left[\sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i,j=1}^{n} \lambda_i \lambda_j y_i y_j \langle x_i, x_j\rangle\right]. \qquad (1.10)\end{aligned}$$

The QP (1.10) is the dual SVM problem. To recover the primal solution from (1.10), set $w = \sum_{i=1}^{n} \lambda_i y_i x_i$. Given a new test datapoint x, the model's prediction $f(x)$ is then:

$$f(x) = \langle w, x\rangle = \sum_{i=1}^{n} \lambda_i y_i \langle x_i, x\rangle. \qquad (1.11)$$

At first, the dual program may seem like just a mathematical curiosity. However, the dual program has a few nice properties. First, the dual program involves n decision variables $(\lambda_1, \ldots, \lambda_n)$, whereas the primal program involves d decision variables $w \in \mathbb{R}^d$. Hence, when the dimensionality d of the covariates $x_i \in \mathbb{R}^d$ *exceeds* the number of data points n, the dual QP can be more efficient to solve. Furthermore, the dual program only ever involves the covariates x_i through *inner products* $\langle x_i, x_j\rangle$. The latter property gives rise to kernel machines, which we will see shortly.

1.3.2 Feature maps and kernels

So far, we have considered linear hypothesis which separate the training data S_n via *linear separators*. This is admittedly a limited model, as assuming data is linearly separable is a strong assumption. However, the expressivity of linear separators can be improved by first performing a *non-linear* projection of the data into a higher dimensional space via a feature map $\Phi : \mathbb{R}^d \mapsto \mathbb{R}^D$, and then performing linear classification in the higher dimensional $\mathbb{R}^D$.

The classic example of this is the following. Suppose there is a dataset in $\mathbb{R}^2$ where the classification rule is the positive class $+1$ correspond to $\|x\| \geqslant 1$, and the negative class -1 corresponds to $\|x\| < 1$. That is, the *decision boundary* is the circle $x_1^2 + x_2^2 = 1$ of radius one. Because this decision boundary is non-linear, it cannot accurately expressed as a linear decision boundary in $\mathbb{R}^2$. However, let us consider the feature map $\Phi(x) := (1, x_1, x_2, x_1x_2, x_1^2, x_2^2) \in \mathbb{R}^6$ which enumerates all monomials of the coordinates of x up to

(and including) degree two. Note that the linear hypothesis $w = (-1, 0, 0, 0, 1, 1) \in \mathbb{R}^6$ expresses perfectly reconstructs the decision boundary.

Now let us consider what happens when we apply the dual SVM QP Equation (1.10) to a *featurized* dataset $\tilde{S}_n = \{(\Phi(x_1), y_1), \ldots, (\Phi(x_n), y_n)\}$. In particular, we obtain the following QP:

$$\max_{\lambda_i \geqslant 0} \left[\sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i,j=1}^n \lambda_i \lambda_j y_i y_j \langle \Phi(x_i), \Phi(x_j) \rangle \right].$$

Interestingly, after computing all pairwise inner products $\langle \Phi(x_i), \Phi(x_j) \rangle$, the computational complexity of solving the dual QP remains exactly the same as solving the dual QP on the original dataset S_n. Furthermore, predictions $f(x)$ are given by (cf. Equation (1.11)):

$$f(x) = \sum_{i=1}^n \lambda_i y_i \langle \Phi(x_i), \Phi(x) \rangle.$$

Hence, the dimensionality of $\Phi(x_i)$ only appears implicitly in the inner product, in both the dual SVM QP in addition to the prediction function. In particular, even if the output of Φ is infinite dimensional, as long as the inner products $\langle \Phi(x), \Phi(y) \rangle$ can be computed efficiently, then the dual SVM program and its corresponding predictor can still be solved with finite dimensional optimization. Hence, this motivates us to define a *kernel* $k : \mathsf{X} \times \mathsf{X} \mapsto \mathbb{R}$ as a generalization of an inner product:

$$k(x, y) = \langle \Phi(x), \Phi(y) \rangle.$$

With this kernel function, we define the *kernelized support vector machine*:

$$\max_{\lambda_i \geqslant 0} \left[\sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i,j=1}^n \lambda_i \lambda_j y_i y_j k(x_i, x_j) \right], \quad f(x) = \sum_{i=1}^n \lambda_i y_i k(x_i, x). \tag{1.12}$$

What properties of $k(x, y)$ do we need to ensure that $k(x, y)$ acts like an inner product?

Definition 1.14 (Positive semidefinite kernel)**.** *A function $k : \mathsf{X} \times \mathsf{X} \mapsto \mathbb{R}$ is called a* positive semidefinite kernel*, or just* kernel*, if the following two conditions holds:*

(a) Symmetric*: The kernel function must be symmetric, i.e., $k(x, y) = k(y, x)$ for every $x, y \in \mathsf{X}$.*

(b) Positive semi-definite*: Let $x_1, \ldots, x_m \in \mathsf{X}$ be $m \in \mathbb{N}_+$ arbitrary points. Define the associated* gram matrix *$K \in \mathbb{R}^{m \times m}$ with entries $K_{ij} := k(x_i, x_j)$ for $i, j \in [m]$. By property (a), the gram matrix is guaranteed to be symmetric. We impose a further requirement that the gram matrix K must be positive semi-definite (PSD), i.e., the quadratic form $\langle v, Kv \rangle \geqslant 0$ for every test vector $v \in \mathbb{R}^m$.*

The definition of a positive semidefinite kernel ensures that the kernelized SVM (1.12) is guaranteed to be a convex quadratic program (why?).

Examples of kernels: We now list a few examples of classic kernels:

(i) *Linear kernel*: $k(x, y) = \langle x, y \rangle$.

(ii) *Polynomial kernel*: $k(x, y) = (\langle x, y \rangle + \alpha)^p$ where $\alpha \geqslant 0$ and $p \in \mathbb{N}_+$ is the degree.

(iii) *Feature maps*: $k(x, y) = \langle \Phi(x), \Phi(y) \rangle$ for any (possibly infinite dimensional) feature map Φ.

(iv) *Radial basis function (RBF):* $k(x, y) = \exp(-\|x - y\|^2/(2\sigma^2))$ for $\sigma > 0$, where σ is the *bandwidth* of the RBF kernel.

(v) *Exponential kernel:* $k(x, y) = \exp(-\|x - y\|_1/\sigma)$.

Composition of kernels: It turns out that kernels also have nice compositional properties. Let k_1, k_2 denote two kernel functions over the same space X. Then, the following are also kernels:

(i) *Addition:* $k(x, y) = k_1(x, y) + k_2(x, y)$.

(ii) *Product:* $k(x, y) = k_1(x, y)k_2(x, y)$.

Exercise 1.15. Prove the addition and product kernel composition rules listed above. Hint: for the product composition rule, you may use the fact that the Hadamard (entry-wise) product of two PSD matrices is also PSD.

Exercise 1.16. Use the kernel composition rules to prove that the Gaussian kernel:

$$k(x, y) = \exp(-\|x - y\|^2/2)$$

is a valid kernel.

Hint 1: first expand:

$$\exp(-\|x - y\|^2/2) = \exp(-\|x\|^2/2)\exp(-\|y\|^2/2)\exp(\langle x, y\rangle),$$

and then use the composition rules.

Hint 2: the Taylor series expansion of $\exp(x)$ is $\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!}$.

Equivalence between feature maps and kernels: We saw above that any feature map $\Phi(x)$ induces a kernel $k(x, y) = \langle \Phi(x), \Phi(y)\rangle$. What about the reverse direction? If one is given a kernel $k(x, y)$, does there exist a corresponding (possibly infinite dimensional) inner product (Hilbert) space $\mathcal{H}$ and feature map $\Phi : \mathsf{X} \mapsto \mathcal{H}$ such that $k(x, y) = \langle \Phi(x), \Phi(y)\rangle_{\mathcal{H}}$, where the inner product is taken w.r.t. the Hilbert space? The answer turns out to be yes, that these two concepts (positive definite kernels and feature maps) are equivalent. However, we will not prove this in class, as this requires some machinery from functional analysis which is out of scope; students who are interested in further study are encouraged to read into the literature on *reproducing kernel Hilbert spaces* (RKHS) [see e.g. Smola and Schölkopf, 1998]. We conclude our discussing by noting that the corresponding feature maps can be infinite-dimensional. For instance, for the Gaussian and exponential kernels, this is indeed the case.

1.3.3 Random Fourier features

In the previous section, we saw how the inner products appearing in the dual SVM QP could be replaced with a kernel function, improving the expressive power of the a linear classifier. Furthermore, we also discussed before the computational considerations of solving the dual SVM QP. In particular, the dual SVM QP has n decision variables λ_i, and also function evaluations $f(x)$ require computation involving the entire training dataset. This is in contrast to the primal SVM problem, which only has d decision variables and evaluation involves $O(d)$ computation via $f(x) = \langle w, x\rangle$. In other words, the price we paid to increase the expressivity of linear classifiers (by lifting the ambient dimension of the data to a higher, possibly infinite dimensional space) was that computation now scales depending on n instead of d. In this section, we focus on the question: can we regain the computational benefits of working with the primal problem, without losing the expressive power of kernel methods?

The answer to this question turns out, surprisingly, to be yes. The trick is to utilize a technical property of a certain type of kernels, which are called *shift-invariant* kernels.

Definition 1.17 (Shift-invariant kernels)**.** *A kernel $k(x, y)$ is defined to be* shift-invariant *if it can be written in the form $k(x, y) = k(x - y)$.*

The shift-invariance in Definition 1.17 refers to the fact that for any vector $c \in \mathbb{R}^d$, a shift-invariant kernel satisfies:

$$k(x+c, y+c) = k((x+c)-(y+c)) = k(x-y) = k(x,y).$$

Examples of shift-invariant kernels include both the RBF kernel $k(x,y) = \exp(-\|x-y\|^2/(2\sigma^2))$ and the exponential kernel $k(x,y) = \exp(-\|x-y\|/\sigma)$.

The main idea behind random features is as follows. Recall that in Section 1.3.2 we discussed the equivalence between kernels and feature maps. In particular, for any kernel $k(x,y)$, there exists a feature map $\Phi(x)$ such that $k(x,y) = \langle \Phi(x), \Phi(y) \rangle$. If the dimensionality D of this feature map $\Phi(x)$ is less than n, then the primal SVM QP problem can be used in lieu of the dual SVM for improved computational efficiency. Unfortunately, D in general is not guaranteed to be finite. Furthermore, even if D is finite, its dimensionality is not guaranteed to be less than n. It turns out, however, that for shift-invariant kernels, we can explicitly construct a feature map $\Phi(x)$ where its dimensionality D is explicitly controlled. The key result which makes this possible is a result from functional analysis known as Bocher's theorem.

Theorem 1.18 (Bocher's theorem)**.** *A shift-invariant function $k(x,y) = k(x-y)$ is a valid kernel if and only if $k(\cdot)$ is the Fourier transform of a non-negative measure.*

In particular, Bocher's theorem implies that $k(\cdot)$ can be written as:

$$k(\Delta) = \int \exp(j\langle \omega, \Delta \rangle) \mu(\omega) \, \mathrm{d}\omega,$$

where $\mu(\omega)$ is a non-negative measure (here, $j := \sqrt{-1}$ denotes the imaginary unit). Since $\mu(\omega)$ is a non-negative measure, this means that we can normalize it and obtain a valid probability distribution $p(\omega) = \mu(\omega)/Z$, where $Z = \int \mu(\omega) \, \mathrm{d}\omega$. Hence, we can rewrite the above expression as:

$$k(\Delta) = Z \cdot \mathbb{E}_{\omega \sim p(\omega)}[\exp(j\langle \omega, \Delta \rangle)].$$

Why is this useful? Using this identity:

$$k(x,y) = k(x-y) = Z \cdot \mathbb{E}_{\omega \sim p(\omega)}[\exp(j\langle \omega, x-y \rangle)] = Z \cdot \mathbb{E}_{\omega \sim p(\omega)}[\exp(j\langle \omega, x \rangle) \exp(-j\langle \omega, y \rangle)]. \tag{1.13}$$

Now supposing we drawn K iid examples $\omega_1, \ldots, \omega_K$ from $p(\omega)$, we can form a Monte-Carlo approximation of $k(x,y)$ as:

$$k(x,y) \approx \frac{Z}{K} \sum_{i=1}^{K} \exp(j\langle \omega_i, x \rangle) \exp(-j\langle \omega_i, y \rangle). \tag{1.14}$$

Hence if we define $\Phi(x) := \sqrt{\frac{Z}{K}}(\exp(-j\langle \omega_1, x \rangle), \ldots, \exp(-j\langle \omega_K, x \rangle)) \in \mathbb{C}^K$, then we have the approximation $k(x,y) \approx \Phi(x)^* \Phi(y) = \langle \Phi(x), \Phi(y) \rangle_{\mathbb{C}^K}$. Here, $\Phi(x)^*$ denotes the conjugate transpose, and $\langle \cdot, \cdot \rangle_{\mathbb{C}^K}$ denotes the complex inner product on $\mathbb{C}^K$. Hence, we have met our goal of constructing a feature map $\Phi(x)$ where the dimensionality of the feature map is controllable. Of course, the larger K is, the more accurate the Monte-Carlo approximation (1.14) becomes. However, this comes at the expense of more computation. Hence, random features provide a tunable knob to trade off computation for expressivity. Let us now see some specific examples.

RBF kernel example: From probability theory we have the following formula for the characteristic function of a multivariate Gaussian distribution $N(\mu, \Sigma)$:

$$\mathbb{E}_{\omega \sim N(\mu,\Sigma)}[\exp(j\langle x, \omega \rangle)] = \exp\left\{ j\langle x, \mu \rangle - \frac{1}{2} x^\mathsf{T} \Sigma x \right\} \quad \forall x \in \mathbb{R}^d.$$

Hence,

$$\exp(-\|\Delta\|^2/(2\sigma^2)) = \mathbb{E}_{\omega \sim N(0, \sigma^{-2} I)}[\exp(j\langle \Delta, \omega \rangle)] \quad \forall \Delta \in \mathbb{R}^d.$$

Thus, the measure $\mu(\omega)$ from Bochner's theorem (cf. Theorem 1.18) is $\mu(\omega) = N(0, \sigma^{-2} I)$, which happens to already be a normalized probability distribution.

Exponential kernel example: From basic Fourier analysis, we know that for any $a \in \mathbb{R}$:

$$\int_{-\infty}^{\infty} \exp(-2\pi j x s) \exp(-2\pi a|x|)\, \mathrm{d}x = \frac{1}{\pi} \frac{a}{a^2 + s^2} =: \hat{f}_a(s), \quad \forall s \in \mathbb{R}.$$

Hence by Fourier inversion,

$$\exp(-a|x|) = \int_{-\infty}^{\infty} \exp(jxs) \hat{f}_a(s)\, \mathrm{d}s, \quad \forall x \in \mathbb{R}.$$

Consequently, for any $\sigma > 0$:

$$\exp(-\|\Delta\|_1/\sigma) = \int \exp(j\langle \Delta, \omega \rangle) \left[\frac{1}{\pi^d} \prod_{i=1}^{d} \frac{\sigma}{1 + \sigma^2 \omega_i^2} \right] \mathrm{d}\omega, \quad \forall \Delta \in \mathbb{R}^d.$$

Hence, we have

$$\mu(\omega) = \frac{1}{\pi^d} \prod_{i=1}^{d} \frac{\sigma}{1 + \sigma^2 \omega_i^2},$$

which is the product of d Cauchy distributions.

Real-valued features: The random features derived in (1.14) involves a complex-valued feature map $\Phi(x)$. While mathematically there is no issue, from a practical point of view it may be preferable to keep all features real-valued. We can further manipulate (1.13) so that this is the case. Since we know that $k(x, y)$ is real-valued, using Euler's identity $\exp(jx) = \cos(x) + j \sin(x)$, we can write:

$$\begin{aligned} k(x, y) &= \mathrm{Re}(k(x, y)) \\ &= Z \cdot \mathrm{Re}(\mathbb{E}_{\omega \sim p(\omega)}[\exp(j\langle \omega, x \rangle) \exp(-j\langle \omega, y \rangle)]) \\ &= Z \cdot \mathrm{Re}(\mathbb{E}_{\omega \sim p(\omega)}[(\cos(\langle \omega, x \rangle) + j \sin(\langle \omega, x \rangle))(\cos(\langle \omega, x \rangle) - j \sin(\langle \omega, x \rangle))]) \\ &= Z \cdot (\mathbb{E}_{\omega \sim p(\omega)}[\cos(\langle \omega, x \rangle) \cos(\langle \omega, y \rangle)] + \mathbb{E}_{\omega \sim p(\omega)}[\sin(\langle \omega, x \rangle) \sin(\langle \omega, y \rangle)]) \end{aligned}$$

Again letting $\omega_1, \ldots, \omega_K$ denote K iid examples from $p(\omega)$, the following is a Monte-Carlo approximation of $k(x, y)$:

$$k(x, y) \approx \frac{Z}{K} \sum_{i=1}^{K} (\cos(\langle \omega_i, x \rangle) \cos(\langle \omega_i, y \rangle) + \sin(\langle \omega_i, x \rangle) \sin(\langle \omega_i, y \rangle)).$$

Hence we can form the following real-valued feature map:

$$\Phi(x) := \sqrt{\frac{Z}{K}} (\cos(\langle \omega_1, x \rangle), \sin(\langle \omega_1, x \rangle), \ldots, \cos(\langle \omega_K, x \rangle), \sin(\langle \omega_K, x \rangle)) \in \mathbb{R}^{2K}.$$

1.4 Empirical Risk Minimization (ERM)

The Perceptron algorithm we studied in Section 1.2, while having nice theoretical properties, is quite limiting. Indeed, its theoretical analysis requires that the underlying distribution $\mathcal{D}$ is linearly separable, a fairly restrictive assumption for many real world datasets. Therefore, we desire a more general framework which does not require such limiting assumptions.

1.4.1 Framework setup

The main framework we will study in this course, which covers essentially most of modern machine learning, is the empirical risk minimization (ERM) framework. We will spend quite a bit of time studying ERM, as it is the de-facto standard workhorse for modern machine learning.

At its core, ERM is a very simple concept. There are only three core ingredients:

1. A loss function $\ell : \Lambda \times \mathsf{Y} \mapsto \mathbb{R}$, where Λ is an auxiliary space.
2. A known, deterministic selection function $\psi : \Lambda \mapsto \mathsf{Y}$.
3. A function (hypothesis) class $\mathscr{F} = \{f : \mathsf{X} \mapsto \Lambda\}$ over which to restrict our search for predictors over. Note, this class induces a set of predictors by composition with the selection function ψ.
4. A training dataset $S_n = \{z_1, \dots, z_n\}$, sampled i.i.d. from an underlying distribution $\mathcal{D}$ over $\mathsf{Z} = \mathsf{X} \times \mathsf{Y}$.

Remark 1.19. While the ERM framework above is stated somewhat abstractly, for binary classification, we will always consider models with auxiliary space $\Lambda = \mathbb{R}$ and selection rule $\psi = \mathrm{sgn}$. This corresponds to models $f(x)$ which classify examples by thresholding $\mathrm{sgn}(f(x))$. This structure will be implicitly assumed moving forward. Note that the extra generality allows for e.g., multi-class classification, regression, etc. to fit under the same framework.

The goal of a learner is to produce a hypothesis $f \in \mathscr{F}$ to minimize the *population risk*:

$$L[f] := \mathbb{E}[\ell(f(x), y)].$$

Towards this goal, the ERM prescribes to us the following predictor:

$$\hat{f}_{\mathrm{erm}} \in \arg\min_{f \in \mathscr{F}} \hat{L}_n[f] := \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) \right\}. \tag{1.15}$$

The function $\hat{L}_n[f]$ is referred to as the *empirical risk*. The main object of study in this section of the course is the behavior of the population risk of the ERM, i.e., $L[\hat{f}_{\mathrm{erm}}]$. (Note: our notation for both the population risk $L[f]$ and the empirical risk $L[f]$ makes the dependency on the loss ℓ implicit. In the sequel, it should be clear from context what ℓ is, but please do not hesitate to ask if it is not.)

1.4.2 Generalization error

Given a *fixed* predictor $f(x)$ and a dataset S_n, the empirical loss is an unbiased estimator of the true population level error, i.e.,

$$\mathbb{E}[\hat{L}_n[f]] = L[f].$$

However, when the predictor f is no longer independent of the dataset S_n (e.g., f is constructed as a function of S_n, as in the ERM (1.15)), then the empirical risk is no longer an unbiased estimator (convince yourself this is true by constructing an example). Thus, what we will need to do is understand how much the empirical risk can differ from the population risk in a worst case sense. We start with a tautological decomposition:[3]

$$L[f] = \underbrace{\hat{L}_n[f]}_{\text{empirical risk}} + \underbrace{L[f] - \hat{L}_n[f]}_{\text{generalization error}} =: \hat{L}_n[f] + \mathrm{gen}[f]. \tag{1.16}$$

In ERM, we purposefully make the first quantity, the empirical risk $\hat{L}_n[f]$, as small as possible. However, the question that remains is how does this affect $\mathrm{gen}[f]$? Again, the difficulty is that $\hat{f}_{\mathrm{erm}}$ is a function of the dataset S_n, and therefore $\mathbb{E}[L_n[\hat{f}_{\mathrm{erm}}]] \neq L[\hat{f}_{\mathrm{erm}}]$.

[3] My graduate school advisor Ben would often call this decomposition the "fundamental theorem of machine learning".

One of the key ideas behind generalization theory in machine learning is that by *restricting the function class $\mathscr{F}$ which we optimize over*, we can indeed control the generalization error by quantities which scale according to the complexity of the function class. The way this works is by doing something admittedly crude. Since characterizing the random variable $\hat{f}_{\mathrm{erm}}$ is non-trivial in general, we instead consider the *worst-case* generalization error over the class $\mathscr{F}$:

$$\mathrm{gen}[\hat{f}_{\mathrm{erm}}] \leqslant \sup_{f\in\mathscr{F}} \mathrm{gen}[f] = \sup_{f\in\mathscr{F}} \{L[f] - \hat{L}_n[f]\}.$$

Hence, combining with (1.16),

$$L[f] \leqslant \hat{L}_n[f] + \sup_{f\in\mathscr{F}} \{L[f] - \hat{L}_n[f]\}, \quad \forall f \in \mathscr{F}. \tag{1.17}$$

The RHS of the above expression gives a *uniform* bound on the population risk of any $f \in \mathscr{F}$ (including $\hat{f}_{\mathrm{erm}}$); we will focus the next few lectures on deriving explicit expressions for the RHS for various function classes. Before turning to this, we make the following observation. Plugging $\hat{f}_{\mathrm{erm}}$ into the above expression and letting $f \in \mathscr{F}$ be an arbitrary fixed function:

$$\begin{aligned}
L[\hat{f}_{\mathrm{erm}}] &\leqslant \hat{L}_n[\hat{f}_{\mathrm{erm}}] + \sup_{f\in\mathscr{F}} \{L[f] - \hat{L}_n[f]\} \\
&\leqslant \hat{L}_n[f] + \sup_{f\in\mathscr{F}} \{L[f] - \hat{L}_n[f]\} && \text{since } \hat{f}_{\mathrm{erm}} \text{ minimizes } \hat{L}_n[f] \\
&= L[f] + \hat{L}_n[f] - L[f] + \sup_{f\in\mathscr{F}} \{L[f] - \hat{L}_n[f]\} \\
&\leqslant L[f] + 2\max\left\{ \sup_{f\in\mathscr{F}} \{\hat{L}_n[f] - L[f]\}, \sup_{f\in\mathscr{F}} \{L[f] - \hat{L}_n[f]\} \right\}.
\end{aligned}$$

Now if we take $f \in \arg\min_{f\in\mathscr{F}} L[f]$ as the *best in-class* hypothesis, we see that the *excess risk* of $\hat{f}_{\mathrm{erm}}$ is controlled by:

$$L[\hat{f}_{\mathrm{erm}}] - \inf_{f\in\mathscr{F}} L[f] \leqslant 2\max\left\{ \sup_{f\in\mathscr{F}} \{\hat{L}_n[f] - L[f]\}, \sup_{f\in\mathscr{F}} \{L[f] - \hat{L}_n[f]\} \right\}. \tag{1.18}$$

While we will focus on bounds of the form (1.17), we will return to excess risk bounds (1.18) when we study agnostic PAC learning (cf. Section 1.4.12). We conclude by remarking that the argument to control $\sup_{f\in\mathscr{F}}\{\hat{L}_n[f] - L[f]\}$ is typically identical to the one to control the other direction $\sup_{f\in\mathscr{F}}\{L[f] - \hat{L}_n[f]\}$, so focusing on bounds of the form (1.17) is of no loss of generality.

1.4.3 Finite hypothesis classes

As a warmup which contains the key ideas, let us suppose that $\mathscr{F}$ is finite, i.e., $|\mathscr{F}| < \infty$, and that ℓ is the zero-one loss, i.e., $\ell(\lambda, y) = \mathbf{1}\{\mathrm{sgn}(\lambda) \neq y\}$. We will use Hoeffding's inequality combined with a union bound to control the generalization error of every single predictor in $\mathscr{F}$.

Proposition 1.20. *Let ℓ be the zero-one loss and $\mathscr{F}$ be a finite function class. Fix any $\delta \in (0,1)$. With probability at least $1-\delta$,*

$$\max_{f\in\mathscr{F}} \mathrm{gen}[f] \leqslant \sqrt{\frac{2\log(|\mathscr{F}|/\delta)}{n}}.$$

Proof. As stated above, the main tool is Hoeffding's inequality (cf. Proposition B.12). Fix any $f \in \mathscr{F}$. We apply to the sum $M_n = \sum_{i=1}^n X_i$, with $X_i = \mathbf{1}\{y_i \neq \mathrm{sgn}(f(x_i))\}$. Note that by construction $X_i \in [-1,1]$ and hence for any $t > 0$, by Hoeffding's inequality,

$$\mathbb{P}\{\mathrm{gen}[f] \geqslant t\} = \mathbb{P}\{n^{-1}(\mathbb{E}[M_n] - M_n) \geqslant t\} \leqslant \exp(-nt^2/2).$$

Hence by a union bound,

$$\mathbb{P}\left\{\max_{f\in\mathscr{F}} \text{gen}[f] \geqslant t\right\} = \mathbb{P}\left\{\bigcup_{f\in\mathscr{F}} \{\text{gen}[f] \geqslant t\}\right\} \leqslant \sum_{f\in\mathscr{F}} \mathbb{P}\{\text{gen}[f] \geqslant t\} \leqslant |\mathscr{F}| \exp(-nt^2/2).$$

Now, set the RHS above equal to the given δ and solve for t to conclude.

Note: we did not actually use any specific properties of the zero-one loss in this proof (other than that $|\ell| \leqslant 1$), so this same proof actually works for any losses which satisfy $|\ell| \leqslant 1$. □

The above bound in Proposition 1.20 is a prototypical statistical learning bound. The $1/\sqrt{n}$ rate in the denominator is generally unimprovable, as a consequence of the Central Limit Theorem. The $\sqrt{\log|\mathscr{F}|}$ scaling in the numerator states that we are allowed to test exponentially many in n hypothesis before our generalization gap dominates the error.

The generalization bound in Proposition 1.20 is quite powerful already. It tells us, via (1.16), that:

$$\forall f \in \mathscr{F},\ L[f] \leqslant \hat{L}_n[f] + \sqrt{\frac{2\log(|\mathscr{F}|/\delta)}{n}}.$$

Again, it is important to emphasize that this is quite remarkable. It implies that we can test exponentially many hypothesis in the number of data points n, until the training error $\hat{L}_n[f]$ fails to be an accurate proxy of the population error $L[f]$. Let us conclude this section with an example. Suppose that $\mathscr{F} = \{f_\theta : \mathsf{X} \mapsto \mathbb{R} \mid \theta \in \mathbb{R}^d\}$. That is, we optimize over a function class parameterized by d parameters. Let $N_{\texttt{f32}}$ denote the number of valid `float32` numbers (so $N_{\texttt{f32}} \leqslant 2^{32}$). Then, on a computer, $\mathscr{F}$ is a finite hypothesis class of cardinality $|\mathscr{F}| = N_{\texttt{f32}}^d$. So, if we perform optimization in our favorite ML framework,[4] we immediate have:

$$\forall f \in \mathscr{F},\ L[f] \leqslant \hat{L}_n[f] + \sqrt{\frac{2(d\log N_{\texttt{f32}} + \log(1/\delta))}{n}}. \tag{1.19}$$

From this inequality, we have already uncovered a core tenant of conventional machine learning wisdom: *generalization happens when one has more datapoints n than parameters d (i.e., $n \gg d$)*. Now, this wisdom has come into question in the past decade or so, with the success of deep overparmaeterized neural networks (which seem to be generalizing in ways that defy this logic). We will revisit this later on, but for now $n \gg d$ implies generalization is a good mental framework to have.

Exercise 1.21. Suppose that ℓ is the zero-one loss and $\mathscr{F}$ is a countably infinite function class of the form:

$$\mathscr{F} = \{f_\kappa \mid \kappa \in \mathbb{N}_+\}.$$

Let $\delta \in (0,1)$. Show that with probability at least $1-\delta$, we have:

$$\forall f_\kappa \in \mathscr{F},\ \text{gen}[f_\kappa] \leqslant c_0\sqrt{\frac{\log(c_1\kappa/\delta)}{n}},$$

where c_0, $c_1 > 1$ are universal positive constants.

Hint: you may use without proof the well-known solution to the Basel problem: $\sum_{i=1}^{\infty} i^{-2} = \pi^2/6$.

1.4.4 Fast rates with realizable finite hypothesis classes

In addition to the assumptions of the last section, let us now impose an additional assumption, that there exists an $f \in \mathscr{F}$ such that $L[f] = 0$, that is, no mistakes are made at the population level. This is obviously a strong assumption, but we will see that it provides fast rates (as we saw with the Perceptron when data was linearly separable).

[4] You are using `jax` (`https://github.com/google/jax`) right?

Proposition 1.22. *Suppose that ℓ is the zero-one loss, $\mathscr{F}$ is finite, and there exists $f \in \mathscr{F}$ satisfying $L[f] = 0$. Then, the empirical risk minimizer $\hat{f}_n$ satisfies, with probability at least $1 - \delta$,*

$$L[\hat{f}_n] \leqslant \frac{\log(|\mathscr{F}|/\delta)}{n}.$$

Proof. For any $t > 0$, we will study the event $\{L(\hat{f}_n) > t\}$. Define the set of sub-optimal hypothesis as $B(t) = \{f \in \mathscr{F} \mid L(f) > t\}$. Note that $\{L(\hat{f}_n) > t\} = \{\hat{f}_n \in B(t)\}$. Now observe that since there exists an $f \in \mathscr{F}$ with $L[f] = 0$, then the ERM $\hat{f}_n$ will always achieve zero training error, i.e., $\hat{L}_n[\hat{f}_n] = 0$.[5] Hence if $\hat{f}_n \in B(t)$, that means there exists some $f \in B(t)$ with $\hat{L}_n[f] = 0$. So now we compute, for a fixed $f \in B(t)$,

$$\begin{aligned}\mathbb{P}\{\hat{L}_n[f] = 0\} = \mathbb{P}\left\{\bigcap_{i=1}^n \{\operatorname{sgn}(f(x_i)) = y_i\}\right\} &= \prod_{i=1}^n (1 - \mathbb{P}\{\operatorname{sgn}(f(x_i)) \neq y_i\}) \\ &= (1 - L[f])^n \leqslant (1-t)^n \leqslant \exp(-tn),\end{aligned}$$

where we used the elementary inequality $1 - x \leqslant \exp(x)$ for any $x \in \mathbb{R}$. Therefore by a union bound,

$$\begin{aligned}\mathbb{P}\{\hat{f}_n \in B(t)\} &\leqslant \mathbb{P}\{\exists f \in B(t),\ \hat{L}_n(f) = 0\} \leqslant \sum_{f \in B(t)} \mathbb{P}\{\hat{L}_n(f) = 0\} \\ &\leqslant |B(t)| \exp(-tn) \leqslant |\mathscr{F}| \exp(-tn).\end{aligned}$$

Now set the RHS equal to δ and solve for t. □

1.4.5 Moving towards non-finite hypothesis classes: hinge loss and logistic regression

The analyses of Section 1.4.3 and Section 1.4.4 both suffer from one key issue: they only apply to finite function classes, i.e., when $|\mathscr{F}| < \infty$. This is an unnecessary restriction. Indeed we saw that the Perceptron algorithm, which operates over a continuum of linear hypothesis, also enjoys a strong generalization bound (cf. Lemma 1.10). Towards removing this restriction, we first observe that Perception is an instance of ERM.

Hinge loss and the Perceptron. We claim that the Perceptron algorithm is an instance of ERM over hypothesis class of linear function $\mathscr{F} = \{x \mapsto \langle w, x\rangle \mid w \in \mathbb{R}^d\}$. What is the loss function? Here, we slightly overload notation and consider ℓ as a univariate function $\ell(\lambda, y) = \ell(\lambda \cdot y)$. With this univariate notation, we define the *hinge loss*

$$\ell_{\text{hinge}}(s) := \max\{1 - s, 0\}.$$

Now observe that for a datapoint $(x, y) \in \mathsf{Z}$ and parameters w, we have that:

$$\nabla_w \ell_{\text{hinge}}(y\langle w, x\rangle) = -yx \cdot \mathbf{1}\{y\langle w, x\rangle < 1\}.$$

This update rule is exactly that of the Perceptron algorithm (cf. Equation (1.6)). Hence, we can see that Perceptron is precisely running stochastic gradient descent (SGD) on the following empirical risk:

$$L_n[w] = \frac{1}{n}\sum_{i=1}^n \ell_{\text{hinge}}(y_i\langle w, x_i\rangle) = \frac{1}{n}\sum_{i=1}^n \max\{1 - y_i\langle w, x_i\rangle, 0\}.$$

We will revisit SGD in more depth later on in the course (cf. Section 1.5). Next, we will see that even for linear models, the hinge loss is not the only well-motivated loss function.

[5] Technically, there could be a measure-zero event where this is not true, but we ignore this technicality.

Logistic regression. Motivated by the MAP estimator (1.3), we now consider a different type of loss function based on directly learning the posterior probability $p(y \mid x)$. Sticking with a hypothesis class of linear functions $\mathscr{F} = \{x \mapsto \langle w, x\rangle \mid w \in \mathbb{R}^d\}$, consider the following model:

$$p_w(y = 1 \mid x) \propto \exp(\langle w, x\rangle), \quad p_w(y = -1 \mid x) \propto 1.$$

Normalizing across $\mathsf{Y} = \{\pm 1\}$,

$$p_w(y = 1 \mid x) = \frac{\exp(\langle w, x\rangle)}{1 + \exp(\langle w, x\rangle)}, \quad p_w(y = -1 \mid x) = \frac{1}{1 + \exp(\langle w, x\rangle)}.$$

To construct a loss function, let us use the KL-divergence to match $p_w(y \mid x)$ with $p(y \mid x)$. Specifically, conditioned on $x \in \mathsf{X}$,

$$\begin{aligned}
&\mathrm{KL}(p(y \mid x) \,\|\, p_w(y \mid x)) - \mathbb{E}[\log p(y \mid x) \mid x] \\
&= -\mathbb{E}[\log p_w(y \mid x) \mid x] \\
&= -p(y = 1 \mid x) \log p_w(y = 1 \mid x) - p(y = -1 \mid x) \log p_w(y = -1 \mid x) \\
&= p(y = 1 \mid x) \log(1 + \exp(-\langle w, x\rangle)) + p(y = -1 \mid x) \log(1 + \exp(\langle w, x\rangle)) \\
&= \mathbb{E}[\log(1 + \exp(-y\langle w, x\rangle)) \mid x].
\end{aligned}$$

Hence, by the tower property,

$$\mathbb{E}[\mathrm{KL}(p(y \mid x) \,\|\, p_w(y \mid x))] = \mathbb{E}[\log p(y \mid x)] + \mathbb{E}[\log(1 + \exp(-y\langle w, x\rangle))].$$

Observe that the first (negative) entropy term does not depend on the parameters w. Hence, this motivates the population level estimator:

$$w_\star \in \arg\min_{w \in \mathbb{R}^d} \mathbb{E}[\ell_{\log}(y\langle w, x\rangle)], \quad \ell_{\log}(s) := \log(1 + \exp(-s)),$$

and corresponding empirical risk:

$$\hat{L}_n[w] = \frac{1}{n} \sum_{i=1}^n \ell_{\log}(y_i \langle w, x_i\rangle) = \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-y_i \langle w, x_i\rangle)).$$

(Indeed, we have $\mathbb{E}[\hat{L}_n[w]] = L[w]$ for every fixed $w \in \mathbb{R}^d$.) The loss $\ell_{\log}(s)$ is the *logistic loss*, and this setup is referred to as *logistic regression*.

Exercise 1.23. Suppose that we are in the multi-class classification setting, with $\mathsf{Y} = \{1, \ldots, K\}$. Suppose that we posit the following probabilistic model:

$$p_{w_{1:K}}(y = k \mid x) \propto \exp(\langle w_k, x\rangle),\ k \in \{1, \ldots, K\}.$$

Show that the corresponding empirical risk for the population risk

$$L[w] = \mathbb{E}[\mathrm{KL}(p(y \mid x) \,\|\, p_{w_{1:K}}(y \mid x))]$$

is given by the following *cross-entropy loss*:

$$\hat{L}_n[w] = -\frac{1}{n} \sum_{i=1}^n \left[\langle w_{y_i}, x_i\rangle - \log \left[\sum_{j=1}^K \exp(\langle w_j, x_i\rangle) \right] \right].$$

That is, show that $\mathbb{E}[\hat{L}_n[w]] = L[w] + C$ for every fixed $w = \{w_k\}_{k=1}^K$, where C is a constant that does not depend on w.

1.4.6 Margin theory

We have seen two examples of loss functions: the hinge loss and the logistic loss. However, at this point two key questions remain to be answered:

(a) Both our derivations of the hinge loss and logistic loss were based on linear predictors. Can we apply these specific losses beyond linear models?

(b) Ultimately, our goal is to access model quality via the zero-one loss. What is the relationship between the hinge/logistic loss and the zero-one loss? Does low population risk under the former imply low classification error?

Resolving (a) is simple. Observe that both the hinge and logistic loss involve the corresponding linear model $x \mapsto \langle w, x\rangle$ only through quantities of the form $y\langle w, x\rangle$. Hence, for a possibly non-linear hypothesis $f : \mathsf{X} \mapsto \mathbb{R}$, we construct the following population and empirical loss:

$$L[f] = \mathbb{E}[\ell(yf(x))], \quad \hat{L}_n[f] = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i f(x_i)), \quad \ell \in \{\ell_{\text{hinge}}, \ell_{\log}\}.$$

We now turn to resolving (b). For both hinge and logistic loss, there is actually a simple answer. Indeed, a key property of both losses is that they both dominate the zero-one loss in the following way. Specifically, consider a model $f : \mathsf{X} \mapsto \mathbb{R}$, and following similar arguments as we did to conclude Section 1.4.3:

$$\mathbf{1}\{y \neq \operatorname{sgn}(f(x))\} \leqslant \mathbf{1}\{yf(x) \leqslant 0\}.$$

This characterization of the zero-one loss can then be used to show that both losses dominate the zero-one loss, as formalized by the following exercise.

Exercise 1.24. Show that there exists a universal constant $c > 0$ such that:

$$\forall s \in \mathbb{R},\ \mathbf{1}\{s \leqslant 0\} \leqslant \ell_{\text{hinge}}(s) \text{ and } \mathbf{1}\{s \leqslant 0\} \leqslant c \cdot \ell_{\log}(s).$$

What is the sharpest constant c?

Hence, having low population risk under both the hinge or logistic loss implies low classification error, i.e.,

$$\mathbb{P}\{\operatorname{sgn}(f(x)) \neq y\} \leqslant \mathbb{E}[\ell_{\text{hinge}}(yf(x))] \quad \text{and} \quad \mathbb{P}\{\operatorname{sgn}(f(x)) \neq y\} \leqslant c \cdot \mathbb{E}[\ell_{\log}(yf(x))].$$

We can go beyond these two losses by considering a *family* of *ramp losses* parameterized by $\gamma > 0$:

$$\ell_\gamma(s) = \begin{cases} 1 & \text{if } s < 0 \\ 1 - s/\gamma & \text{if } s \in [0, \gamma], \\ 0 & \text{if } s > \gamma \end{cases}.$$

By construction, this loss is sandwiched by the following inequalities:

$$\mathbf{1}\{s \leqslant 0\} \leqslant \ell_\gamma(s) \leqslant \mathbf{1}\{s < \gamma\}. \tag{1.20}$$

Consequently, for every $f \in \mathscr{F}$,

$$\begin{aligned}
\mathbb{P}\{y \neq \operatorname{sgn}(f(x))\} &\leqslant \mathbb{E}[\ell_\gamma(yf(x))] && \text{using (1.20)} \\
&\leqslant \frac{1}{n}\sum_{i=1}^{n} \ell_\gamma(y_i f(x_i)) + \sup_{f \in \mathscr{F}} \operatorname{gen}[f; \ell_\gamma] && \text{recall } \operatorname{gen}[f] = L[f] - \hat{L}_n[f] \\
&\leqslant \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{y_i f(x_i) < \gamma\} + \sup_{f \in \mathscr{F}} \operatorname{gen}[f; \ell_\gamma] && \text{using (1.20).}
\end{aligned} \tag{1.21}$$

(Here, we use the notation $\operatorname{gen}[f; \ell_\gamma]$ to emphasize that the ramp loss is involved in the generalization bound). This derivation states that we can control the zero-one loss in general by:

(a) Controlling the generalization error $\sup_{f \in \mathscr{F}} \mathrm{gen}[f; \ell_\gamma]$ under the ramp loss ℓ_γ, and

(b) Counting the number of mistakes $y_i f(x_i) < \gamma$ on the training dataset S_n. Recall that if f is linear, then we defined (Definition 1.5) the margin of a correctly classified point (x, y) as $y\langle w, x\rangle / \|w\|$. In the general case, the quantity $yf(x)$ serves as the *un-normalized* margin.

Note that, $\hat{f}_{\mathrm{erm}}$ can actually optimize any loss function not necessarily related to ℓ_γ and still apply the bound from Equation (1.21). We shall see shortly studying generalization of ℓ_γ in the non-finite hypothesis class case, is much simpler than directly trying to study the generalization over the zero-one loss, thanks to the Lipschitz continuity of ℓ_γ (observe that ℓ_γ is γ^{-1}-Lipschitz).

1.4.7 Rademacher complexity

Towards a theory for general function classes, we first observe that simply counting the number of parameters of a function class is insufficient. Instead we must consider the geometric structure. For instance, consider the following linear hypothesis classes:

$$\mathscr{F}_p = \{x \mapsto \langle w, x\rangle \mid w \in \mathbb{R}^d,\ \|w\|_p \leqslant 1\},\ p \in [1, \infty].$$

Note that for all p, $\mathscr{F}_p$ is parameterized by d parameters. However, in high dimension, the parameter spaces of these sets are drastically different. By a volume calculation,

$$\mathrm{Vol}(\{\|w\|_1 \leqslant 1\}) = 2^d/d!, \quad \mathrm{Vol}(\{\|w\|_2 \leqslant 1\}) = \pi^{d/2}/\Gamma(1 + d/2), \quad \mathrm{Vol}(\{\|w\|_\infty \leqslant 1\}) = 2^d.$$

Note that as $d \to \infty$, the first two volumes tend to 0, whereas the last tends to ∞. Hence, these parameter sets are geometrically very different, despite all sets being described by d parameters.

To address this issue, we now introduce a powerful tool for analyzing generalization error, which captures the geometry of the underlying hypothesis class.

Definition 1.25 (Rademacher complexity)**.** *Let $\mathscr{F}$ be a set of functions mapping $\mathsf{X} \mapsto \mathbb{R}$. The Rademacher complexity of $\mathscr{F}$ is defined as:*

$$\mathcal{R}_n(\mathscr{F}) := \mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^n \varepsilon_i f(x_i),$$

where $\{\varepsilon_i\}_{i=1}^n$ are independent Rademacher random variables (and also independent of the $\{x_i\}_{i=1}^n$). Note that the expectation is taken jointly over both the data $\{x_i\}$ and the Rademacher random variables $\{\varepsilon_i\}$.

Notation overload in Definition 1.25. Note that there is a slight overload of notation in the definition of the Rademacher complexity in Definition 1.25. There, the domain X is an abstract domain. In particular, it is *not* necessarily equal to the domain that the covariates in supervised learning reside in. In the sequel, we will often consider the Rademacher complexity of function classes taking as input covariates (i.e., functions mapping $\mathsf{X} \mapsto \mathbb{R}$), but we will also function classes taking as input labelled pairs (i.e., functions mapping $\mathsf{X} \times \mathsf{Y} \mapsto \mathbb{R}$).

The power of the Rademacher complexity comes from the following classical symmetrization lemma.

Proposition 1.26 (Symmetrization)**.** *Let $\{x_i\}_{i=1}^n$ be* independent *random variables in X, and let $\mathscr{F}$ be a set of functions mapping $\mathsf{X} \mapsto \mathbb{R}$. We have that:*

$$\mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^n (\mathbb{E}[f(x_i)] - f(x_i)) \leqslant 2\mathcal{R}_n(\mathscr{F}).$$

Proof. Let $\{x_i'\}$ be an independent copy of $\{x_i\}$, and let $\{\varepsilon_i\}$ be independent Rademacher random variables. We have:

$$\mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^n (\mathbb{E}[f(x_i)] - f(x_i))$$

$$
\begin{aligned}
&= \mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} (\mathbb{E}[f(x_i')] - f(x_i)) && \text{since } x_i \overset{d}{=} x_i' \\
&= \mathbb{E}_{x_{1:n}} \sup_{f \in \mathscr{F}} \mathbb{E}_{x_{1:n}'} \left[n^{-1} \sum_{i=1}^{n} (f(x_i') - f(x_i)) \right] && \text{linearity of expectation} \\
&\leqslant \mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} (f(x_i') - f(x_i)) && \text{since } \sup_f \mathbb{E}[Z_f] \leqslant \mathbb{E} \sup_f Z_f \\
&= \mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} \varepsilon_i (f(x_i') - f(x_i)) && \text{since } f(x_i') - f(x_i) \overset{d}{=} \varepsilon_i (f(x_i') - f(x_i)) \\
&\leqslant \mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} \varepsilon_i f(x_i') + \mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} \varepsilon_i f(x_i) && \text{since } \sup_x [f(x) + g(x)] \leqslant \sup_x f(x) + \sup_x g(x) \\
&= 2\mathcal{R}_n(\mathscr{F}) && \text{since } x_i \overset{d}{=} x_i'.
\end{aligned}
$$

(Note: observe where in the proof we used the fact that the x_i's are independent (cf. Exercise 1.27).) □

Exercise 1.27. Construct a sequence of random variables $\{x_i\}_{i=1}^n$ in X that are *not* independent, such that for there exists a set of functions $\mathscr{F}$ mapping $\mathsf{X} \mapsto \mathbb{R}$ such that

$$
\mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} (\mathbb{E}[f(x_i)] - f(x_i)) > 2\mathcal{R}_n(\mathscr{F}).
$$

Note that given $\mathscr{F}$, applying Proposition 1.26 to the function class $\{-f \mid f \in \mathscr{F}\}$ we also conclude that,

$$
\mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} (f(x_i) - \mathbb{E}[f(x_i)]) \leqslant 2\mathcal{R}_n(\{-f \mid f \in \mathscr{F}\}) = 2\mathcal{R}_n(\mathscr{F}).
$$

The last equality is a special case the positive homogeneity of Rademacher complexities (convince yourself it is true!). Therefore, we can upgrade the conclusion of Proposition 1.26 to consider both directions:

$$
\max \left\{ \mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} (\mathbb{E}[f(x_i)] - f(x_i)), \mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} (f(x_i) - \mathbb{E}[f(x_i)]) \right\} \leqslant 2\mathcal{R}_n(\mathscr{F}). \tag{1.22}
$$

Next, we prove a high probability deviation bound on the quantity $\sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} (\mathbb{E}[f(x_i)] - f(x_i))$ using the bounded-differences inequality (Proposition B.16).

Proposition 1.28. *Let $\{x_i\}_{i=1}^n$ be independent random variables, and let $\mathscr{F}$ be a set of uniformly bounded functions mapping $\mathsf{X} \mapsto [-B, B]$. Then, with probability at least $1 - \delta$ (over the randomness of $\{x_i\}_{i=1}^n$),*

$$
\sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} (\mathbb{E}[f(x_i)] - f(x_i)) \leqslant 2\mathcal{R}_n(\mathscr{F}) + 2B\sqrt{\frac{2\log(1/\delta)}{n}}.
$$

Proof. Let $h(x_{1:n}) := \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} (\mathbb{E}[f(x_i)] - f(x_i))$. Pick any index $i \in \{1, \dots n\}$, and let $x_1, \dots, x_n, x_i'$ be arbitrary. Observe that, since $\mathscr{F}$ is uniformly bounded,

$$
\begin{aligned}
&|h(x_{1:i-1}, x_i, x_{i+1:n}) - h(x_{1:i-1}, x_i', x_{i+1:n})| \\
&= \left| \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} (\mathbb{E}[f(x_i)] - f(x_i)) - \sup_{f \in \mathscr{F}} n^{-1} \left[(\mathbb{E}[f(x_i)] - f(x_i')) + \sum_{i \neq i}^{n} (\mathbb{E}[f(x_i)] - f(x_i)) \right] \right|
\end{aligned}
$$

$$\leqslant \frac{1}{n} \sup_{f \in \mathscr{F}} |f(x_i) - f(x_i')| \leqslant \frac{2B}{n}.$$

Therefore the function h satisfies bounded-differences, and we can apply the bounded-differences inequality (Proposition B.16) to conclude:

$$\mathbb{P}\{h(x_{1:n}) - \mathbb{E}[h(x_{1:n})] \geqslant t\} \leqslant \exp\left(-nt^2/(8B^2)\right).$$

The claim now follows by using Proposition 1.26 to bound $\mathbb{E}[h(x_{1:n})] \leqslant 2\mathcal{R}_n(\mathscr{F})$. □

Again, as we saw following Proposition 1.26, applying Proposition 1.28 to the negated function class $\{-f \mid f \in \mathscr{F}\}$, we conclude that with probability at least $1 - \delta$,

$$\sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} (f(x_i) - \mathbb{E}[f(x_i)]) \leqslant 2\mathcal{R}_n(\mathscr{F}) + 2B\sqrt{\frac{2\log(1/\delta)}{n}}. \tag{1.23}$$

At this point, we have seen that the Rademacher complexity is a natural notion of complexity which arises from controlling the expected value of the following supremum of the *empirical process*:

$$\sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} (\mathbb{E}[f(x_i)] - f(x_i)).$$

We now discuss some properties of this complexity measure. Immediately from the definition, we see that the Rademacher complexity has the following properties (here, $\mathscr{F}$ and $\mathscr{G}$ are arbitrary function class mapping $\mathsf{X} \mapsto \mathbb{R}$):

(a) (Monotonicity) For any $\mathscr{G}$ such that $\mathscr{F} \subseteq \mathscr{G}$, $\mathcal{R}_n(\mathscr{F}) \leqslant \mathcal{R}_n(\mathscr{G})$.

(b) (Positive homogeneous) For any $c \in \mathbb{R}$, $\mathcal{R}_n(\{x \mapsto cf(x) \mid f \in \mathscr{F}\}) = |c| \cdot \mathcal{R}_n(\mathscr{F})$.

(c) (Offset invariance) Let $f_0 : \mathsf{X} \mapsto \mathbb{R}$ be an arbitrary function. We have

$$\mathcal{R}_n(\{x \mapsto f(x) + f_0(x) \mid f \in \mathscr{F}\}) = \mathcal{R}_n(\mathscr{F}).$$

(d) (Sub-additivity) For any $\mathscr{G}$,

$$\mathcal{R}_n(\{x \mapsto f(x) + g(x) \mid f \in \mathscr{F},\, g \in \mathscr{G}\}) \leqslant \mathcal{R}_n(\mathscr{F}) + \mathcal{R}_n(\mathscr{G}).$$

(Check that these properties hold!). Furthermore, Rademacher complexities have a key "peeling" property which will end up being important for what follows.

Proposition 1.29. *Let $\ell : \mathbb{R} \mapsto \mathbb{R}$ be L-Lipschitz, i.e.,:*

$$\forall x, y \in \mathbb{R},\ |\ell(x) - \ell(y)| \leqslant L|x - y|.$$

Let $\mathscr{F}$ be a function class mapping $\mathsf{X} \mapsto \mathbb{R}$. We have that:

$$\mathcal{R}_n(\{x \mapsto \ell(f(x)) \mid f \in \mathscr{F}\}) \leqslant L \cdot \mathcal{R}_n(\mathscr{F}).$$

Unlike the previously stated properties, the proof of Proposition 1.29 is not as immediate. It is, however, a consequence of the following Ledoux and Talagrand's contraction lemma.

Lemma 1.30 (Ledoux and Talagrand Contraction)**.** *Let $\phi : \mathbb{R} \mapsto \mathbb{R}$ be a contraction, i.e.,*

$$\forall x, y \in \mathbb{R},\ |\phi(x) - \phi(y)| \leqslant |x - y|.$$

Let $T \subset \mathbb{R}^n$ be any set. We have that:

$$\mathbb{E} \sup_{t \in T} \sum_{i=1}^{n} \varepsilon_i \phi(t_i) \leqslant \mathbb{E} \sup_{t \in T} \sum_{i=1}^{n} \varepsilon_i t_i.$$

Proof. We will prove that, for every index $i \in \{1, \dots, n\}$ and any arbitrary $A : T \mapsto \mathbb{R}$,

$$\mathbb{E} \sup_{t \in T} [A(t) + \varepsilon \phi(t_i)] \leqslant \mathbb{E} \sup_{t \in T} [A(t) + \varepsilon t_i]. \tag{1.24}$$

We start by expanding the LHS of (1.24):

$$\mathbb{E} \sup_{t \in T} [A(t) + \varepsilon \phi(t_i)] = \frac{1}{2} \sup_{t \in T} [A(t) + \phi(t_i)] + \frac{1}{2} \sup_{t \in T} [A(t) - \phi(t_i)].$$

Suppose that $u \in T$ achieves the supremum for the first term on the RHS, and $v \in T$ achieves the supremum for the second term on the RHS (if both supremums are not achieved, consider sequences $\{u^{(i)}\}, \{v^{(i)}\}$ in T which approach the supremums instead). First, let us suppose that $u_i \leqslant v_i$. Then, the contraction of ϕ yields $\phi(u_i) - \phi(v_i) \leqslant v_i - u_i$, and therefore

$$\begin{aligned}
\frac{1}{2}(A(u) + \phi(u_i)) + \frac{1}{2}(A(v) - \phi(v_i)) &\leqslant \frac{1}{2}(A(u) - u_i) + \frac{1}{2}(A(v) + v_i) \\
&\leqslant \frac{1}{2} \sup_{t \in T} (A(t) - t_i) + \frac{1}{2} \sup_{t \in T} (A(t) + t_i) \qquad \text{since } u, v \in T \\
&= \mathbb{E} \sup_{t \in T} [A(t) + \varepsilon t_i].
\end{aligned}$$

On the other hand, if $v_i \leqslant u_i$, then $\phi(u_i) - \phi(v_i) \leqslant u_i - v_i$, and a nearly identical argument yields:

$$\frac{1}{2}(A(u) + \phi(u_i)) + \frac{1}{2}(A(v) - \phi(v_i)) \leqslant \frac{1}{2}(A(u) + u_i) + \frac{1}{2}(A(v) - v_i) \leqslant \mathbb{E} \sup_{t \in T} [A(t) + \varepsilon t_i].$$

Combining these two cases, we have shown that (1.24) holds. To conclude the proof, we use the independence of the Rademacher random variables $\{\varepsilon_i\}$ along with repeated applications of (1.24) to "peel" off the ϕ in a step by step manner:

$$\begin{aligned}
\mathbb{E} \sup_{t \in T} \sum_{i=1}^{n} \varepsilon_i \phi(t_i) &= \mathbb{E}_{\varepsilon_{\neg n}} \mathbb{E}_{\varepsilon_n} \sup_{t \in T} \left[\sum_{i=1}^{n-1} \varepsilon_i \phi(t_i) + \varepsilon_n \phi(t_n) \right] \\
&\leqslant \mathbb{E}_{\varepsilon_{\neg n}} \mathbb{E}_{\varepsilon_n} \sup_{t \in T} \left[\sum_{i=1}^{n-1} \varepsilon_i \phi(t_i) + \varepsilon_n t_n \right] \qquad \text{using (1.24)} \\
&= \mathbb{E}_{\varepsilon_{\neg n-1}} \mathbb{E}_{\varepsilon_{n-1}} \sup_{t \in T} \left[\left\{ \sum_{i=1}^{n-2} \varepsilon_i \phi(t_i) + \varepsilon_n t_n \right\} + \varepsilon_{n-1} \phi(t_{n-1}) \right] \\
&\leqslant \mathbb{E}_{\varepsilon_{\neg n-1}} \mathbb{E}_{\varepsilon_{n-1}} \sup_{t \in T} \left[\left\{ \sum_{i=1}^{n-2} \varepsilon_i \phi(t_i) + \varepsilon_n t_n \right\} + \varepsilon_{n-1} t_{n-1} \right] \qquad \text{using (1.24) again} \\
&\vdots \\
&\leqslant \mathbb{E} \sup_{t \in T} \sum_{i=1}^{n} \varepsilon_i t_i.
\end{aligned}$$

Above, the notation $\varepsilon_{\neg k}$ for an index $k \in \{1, \dots, n\}$ denotes all the Rademacher random variables excluding the k-th index, i.e., $\varepsilon_{\neg k} = (\varepsilon_1, \dots, \varepsilon_{k-1}, \varepsilon_{k+1}, \dots, \varepsilon_n)$. □

Exercise 1.31. Prove Proposition 1.29 using Lemma 1.30.

1.4.8 Generalization bounds via Rademacher complexity

We now have the tools in place to state our first generalization bound via Rademacher complexities.

Theorem 1.32 (Generalization bound via Rademacher complexity)**.** *Let $\ell : \mathbb{R} \mapsto \mathbb{R}$ be any L-Lipschitz loss function. Let $\mathscr{F}$ be a function class mapping $\mathsf{X} \mapsto \mathbb{R}$. Suppose that:*

$$\sup_{f \in \mathscr{F}} \sup_{(x,y) \in \mathsf{Z}} |\ell(yf(x))| \leqslant B_\ell.$$

With probability at least $1 - \delta$ (over the randomness of $\{(x_i, y_i)\}_{i=1}^n$),

$$\forall f \in \mathscr{F},\ L[f] \leqslant \frac{1}{n} \sum_{i=1}^n \ell(y_i f(x_i)) + 2L \cdot \mathcal{R}_n(\mathscr{F}) + 2B_\ell \sqrt{\frac{2\log(1/\delta)}{n}}.$$

Proof. Much as we proceeded in (1.16), we have for any $f \in \mathscr{F}$,

$$\begin{aligned} L[f] &= \frac{1}{n} \sum_{i=1}^n \ell(y_i f(x_i)) + \mathbb{E}[\ell(yf(x))] - \frac{1}{n} \sum_{i=1}^n \ell(y_i f(x_i)) \\ &\leqslant \frac{1}{n} \sum_{i=1}^n \ell(y_i f(x_i)) + \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^n (\mathbb{E}[\ell(y_i f(x_i))] - \ell(y_i f(x_i))). \end{aligned}$$

We use Proposition 1.28 to analyze the second term (noting that the empirical process we want to control is over functions mapping $\mathsf{X} \times \mathsf{Y} \mapsto \mathbb{R}$, cf. the discussion after Definition 1.25), which tells us with probability at least $1 - \delta$,

$$\sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^n (\mathbb{E}[\ell(y_i f(x_i))] - \ell(y_i f(x_i))) \leqslant 2\mathcal{R}_n(\{(x,y) \mapsto \ell(yf(x)) \mid f \in \mathscr{F}\}) + 2B_\ell \sqrt{\frac{2\log(1/\delta)}{n}}. \quad (1.25)$$

The Rademacher complexity term $\mathcal{R}_n(\{(x,y) \mapsto \ell(yf(x)) \mid f \in \mathscr{F}\})$ in (1.8) is challenging to analyze directly, since it involves the composition of two functions. However, we can use Rademacher contraction to "peel" off the loss ℓ directly exposing the Rademacher complexity of $\mathscr{F}$. Specifically, by Proposition 1.29:

$$\begin{aligned} \mathcal{R}_n(\{(x,y) \mapsto \ell(yf(x)) \mid f \in \mathscr{F}\}) &\leqslant L \cdot \mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^n \varepsilon_i y_i f(x_i) \\ &= L \cdot \mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^n \varepsilon_i f(x_i) \qquad \varepsilon_i y_i \overset{d}{=} \varepsilon_i \text{ by symmetry} \\ &= L \cdot \mathcal{R}_n(\mathscr{F}). \end{aligned}$$

□

Recalling our discussion on margin theory (Section 1.4.6), an immediate corollary of Theorem 1.32 is the following margin bound. The following bound allows one to decouple the question of what loss function to use for training from the desired zero-one loss metric. It states that no matter what loss function one uses for empirical risk minimization (more broadly, no matter what algorithm is used to select a hypothesis from the training data), one can bound the estimator's population zero-one loss by counting the number of margin mistakes the estimator makes on the training data, and then adding some extra terms that account for both the complexity of the hypothesis space $\mathscr{F}$ and the number of training data points observed.

Corollary 1.33 (Margin generalization bound)**.** *Fix any $\gamma > 0$. Let $\mathscr{F}$ be a function class mapping $\mathsf{X} \mapsto \mathbb{R}$. With probability at least $1 - \delta$ (over the randomness of $\{(x_i, y_i)\}_{i=1}^n$),*

$$\forall f \in \mathscr{F},\ \mathbb{P}\{\operatorname{sgn}(f(x)) \neq y\} \leqslant \frac{1}{n} \sum_{i=1}^n \mathbf{1}\{y_i f(x_i) < \gamma\} + \frac{2}{\gamma} \cdot \mathcal{R}_n(\mathscr{F}) + 2\sqrt{\frac{2\log(1/\delta)}{n}}.$$

Proof. Apply Theorem 1.32 to the ramp loss ℓ_γ (which satisfies $|\ell_\gamma| \leqslant 1$ and is γ^{-1}-Lipschitz) from Section 1.4.6, and then apply the ramp loss inequality $\ell_\gamma(s) \leqslant \mathbf{1}\{s < \gamma\}$ from (1.20) to conclude. □

One drawback of Corollary 1.33 is that it only applies for a *fixed* margin γ. However, this can be remedied at a small cost to the bound, as shown in the following exercise.

Exercise 1.34. Show that one can extend Corollary 1.33 to apply for all margins as follows. Suppose that $\mathscr{F}$ is uniformly bounded, so that $|f| \leqslant B$ for $f \in \mathscr{F}$. With probability at least $1-\delta$, for all $f \in \mathscr{F}$,

$$\mathbb{P}\{\operatorname{sgn}(f(x)) \neq y\} \leqslant \inf_{\gamma \in [0,B]} \left[\frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{y_i f(x_i) < \gamma\} + \frac{4}{\gamma} \cdot \mathcal{R}_n(\mathscr{F}) + c\sqrt{\frac{\log\log(B/\gamma) + \log(1/\delta)}{n}} \right],$$

where $c > 0$ is a universal constant. Note that the range of $\gamma \in [0, B]$ is not restrictive, since the assumption that $|f| \leqslant B$ implies that for any $\gamma > B$, the margin mistake $yf(x) < \gamma$ always occurs and hence the bound on the RHS becomes vacuous for $\gamma > B$.

Why the ramp loss? At this point, a natural question to ask is why Theorem 1.32 is applied to the ramp loss ℓ_γ and not directly to the zero-one loss. While the simple answer is that the zero-one loss is not Lipschitz thanks to its discontinuity at the origin, inspecting the proof of Theorem 1.32 shows that the Lipschitzness of the loss is only used to "peel" the loss out of the Rademacher complexity and directly expose the Rademacher complexity $\mathcal{R}_n(\mathscr{F})$ of the hypothesis class $\mathscr{F}$. Indeed, the proof up to and including (1.25) can be replayed with the zero-one loss substituted in. In other words, with probability at least $1-\delta$,

$$\forall f \in \mathscr{F},\ \mathbb{P}\{\operatorname{sgn}(f(x)) \neq y\} \leqslant \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{\operatorname{sgn}(f(x_i)) \neq y_i\} + 2\mathcal{R}_n(\{(x,y) \mapsto \mathbf{1}\{\operatorname{sgn}(f(x)) \neq y\} \mid f \in \mathscr{F}\}) + 2\sqrt{\frac{2\log(1/\delta)}{n}}.$$

The Rademacher complexity term to be controlled then becomes:

$$\begin{aligned}
\mathcal{R}_n(\{(x,y) \mapsto \mathbf{1}\{\operatorname{sgn}(f(x)) \neq y\} \mid f \in \mathscr{F}\}) &= \mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} \varepsilon_i \mathbf{1}\{y_i \neq \operatorname{sgn}(f(x_i))\} \\
&= \mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} \varepsilon_i \left(\frac{1 - \operatorname{sgn}(f(x_i)) y_i}{2} \right) \quad \text{using Exercise 1.35} \\
&= \frac{1}{2} \mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} \varepsilon_i (-y_i) \operatorname{sgn}(f(x_i)) \\
&= \frac{1}{2} \mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} \varepsilon_i \operatorname{sgn}(f(x_i)) \qquad \text{since } \varepsilon_i \overset{d}{=} -\varepsilon_i y_i \\
&= \frac{1}{2} \mathcal{R}_n(\{x \mapsto \operatorname{sgn}(f(x)) \mid f \in \mathscr{F}\}). \qquad (1.26)
\end{aligned}$$

We will see later (Section 1.4.10) that the Rademacher complexity in (1.26) can be controlled by the VC-dimension of the predictor class $\mathscr{H} = \{x \mapsto \operatorname{sgn}(f(x)) \mid f \in \mathscr{F}\}$. The VC-dimension, a classical object in the machine learning literature which we will study shortly, is a combinatorial quantity which indicates the cardinality of the largest dataset which can be *shattered* by a predictor. However, this approach of going through the VC-dimension has its drawbacks, one of which is the geometric structure of the underlying hypothesis class $\mathscr{F}$ is abstracted into a combinatorial quantity, rather than directly exposed via the Rademacher complexity. The latter is much intuitive in my opinion, and is why we study the latter approach first.

We conclude our discussion with the following exercise.

Exercise 1.35. Show that for all $y, y' \in \{\pm 1\}$,

$$\mathbf{1}\{y \neq y'\} = \frac{1 - y \cdot y'}{2}.$$

1.4.9 Explicit computations of Rademacher complexity

In this section, we will explicitly computer upper and lower bounds on the Rademacher complexity, in order to get a better understanding of the properties of this complexity measure. Recall that from Theorem 1.32, for any L-Lipschitz and B-bounded loss, the Rademacher complexity controls the maximum generalization error over the function class in the following way. With probability at least $1 - \delta$,

$$\sup_{f \in \mathscr{F}} \mathrm{gen}[f] \leqslant 2L \cdot \mathcal{R}_n(\mathscr{F}) + 2B\sqrt{\frac{2\log(1/\delta)}{n}}. \tag{1.27}$$

As we will see through a series of examples, the first term above is the dominant term.

Linear function classes. We will first consider Rademacher complexities of linear function classes,

$$\mathscr{F}_p = \{x \mapsto \langle x, w\rangle \mid \|w\|_p \leqslant 1\}.$$

We will also assume that $x_i \sim N(0, I)$ to simplify expressions as much as possible. Our first computation is to simplify $\mathcal{R}_n(\mathscr{F}_p)$.

Recall from linear algebra, for any norm $\|\cdot\|$, its dual norm $\|\cdot\|_*$ is defined as:

$$\|x\|_* := \sup\{\langle z, x\rangle \mid \|z\| \leqslant 1\}.$$

For ℓ_p norms, the dual norms have a simple characterization. For a given $p \in [1, \infty]$, the number q is *dual* to p if $1/p + 1/q = 1$. With this duality pairing, we have that the ℓ_q norm $\|\cdot\|_q$ is dual to the ℓ_p norm $\|\cdot\|_p$, as a consequence of Hölder's inequality. With this context in mind, we have:

$$\mathcal{R}_n(\mathscr{F}_p) = \mathbb{E} \sup_{\|w\|_p \leqslant 1} n^{-1} \sum_{i=1}^n \varepsilon_i \langle x_i, w\rangle = n^{-1}\mathbb{E} \sup_{\|w\|_p \leqslant 1} \left\langle w, \sum_{i=1}^n \varepsilon_i x_i \right\rangle = n^{-1}\mathbb{E} \left\| \sum_{i=1}^n \varepsilon_i x_i \right\|_q. \tag{1.28}$$

That is, the Rademacher complexity of $\mathscr{F}_p$ involves computing the dual ℓ_q-norm of a random vector sum. For $p = q = 2$, Jensen's inequality yields a simple upper bound:

$$\mathcal{R}_n(\mathscr{F}_2) = n^{-1}\mathbb{E}\left\| \sum_{i=1}^n \varepsilon_i x_i \right\| \leqslant n^{-1}\sqrt{\mathbb{E}\left\| \sum_{i=1}^n \varepsilon_i x_i \right\|^2} = n^{-1}\sqrt{\sum_{i=1}^n \mathbb{E}\|x_i\|^2} = \sqrt{\frac{d}{n}}. \tag{1.29}$$

This argument is actually fairly sharp, as shown in the following exercise.

Exercise 1.36. Show that $\mathcal{R}_n(\mathscr{F}_2) \geqslant c\sqrt{d/n}$ for a universal constant c. You may assume that $d \geqslant c_0$ for any fixed universal constant c_0 to simplify the argument.

Hint: You may use without proof the following fact, known as the *Gaussian Poincaré inequality*. Let $g \in \mathbb{R}^k$ be an isotropic Gaussian random vector. For any $f : \mathbb{R}^k \mapsto \mathbb{R}$, we have that

$$\mathrm{Var}(f(g)) \leqslant \mathbb{E}\|\nabla f(g)\|^2.$$

Note that this fact is dimension free.

Thus, we see that for $\mathscr{F}_2$, the Rademacher complexity scales as $\sqrt{d/n}$. Plugging into (1.27), assuming $B = L = 1$,

$$\sup_{f \in \mathscr{F}_2} \operatorname{gen}[f] \leqslant c\sqrt{\frac{d + \log(1/\delta)}{n}},$$

for a universal constant c. Note that so far, this matches up to constants the generalization bound we derived by discretizing $\mathscr{F}_2$ using `float32` numbers in Equation (1.19).

We now turn to study several other linear hypothesis spaces. First, we will see that for $\mathscr{F}_1$, the Rademacher machinery yields a significant reduction in model complexity.

Proposition 1.37. *We have that:*

$$\mathcal{R}_n(\mathscr{F}_1) \leqslant \sqrt{\frac{2\log(2d)}{n}}.$$

Proof. Fix an index $j \in \{1, \dots, d\}$. It is not hard to see that $Z_j := \sum_{i=1}^n \varepsilon_i \langle x_i, e_j \rangle \overset{d}{=} N(0, n)$ (why?). Hence by the sub-Gaussian maximal inequality (Exercise B.18),

$$\mathbb{E} \left\| \sum_{i=1}^n \varepsilon_i x_i \right\|_\infty = \mathbb{E} \max_{j=1,\dots,d} |Z_j| \leqslant \sqrt{2n \log(2d)}.$$

The claim now follows from the dual ℓ_∞-norm characterization of $\mathcal{R}_n(\mathscr{F}_1)$ (cf. Equation (1.28)). □

Thus we see that from a Rademacher complexity standpoint, the geometry of the ℓ_1-ball yields a significant reduction in model complexity compared to the ℓ_2-ball. Note that this can go the other direction, as we see for the ℓ_∞-ball.

Exercise 1.38. Show that $\mathcal{R}_n(\mathscr{F}_\infty) = \sqrt{2/\pi} \cdot d/\sqrt{n}$.

We conclude our last example of linear hypothesis by looking at sparse predictors. Here, we consider a strict subset of $\mathscr{F}_2$ where the support (non-zero entries) of the weight vector w is restricted to at most s (with $s < d$) entries. We do not, however, which coordinates the support can entail. Note that if we restricted $\mathscr{F}_2$ to one fixed subset of s non-zero coordinates, then we would have Rademacher complexity scaling as $\sqrt{s/n}$; only the coordinates with non-zero support affect the model complexity. However, it turns out that we only have to pay a modest price to allow for any support of at most s coordinates.

Proposition 1.39. *Let $\mathscr{F}_{2,s}$ denote the following sparse hypothesis class (where $1 \leqslant s < d$):*

$$\mathscr{F}_{2,s} := \{x \mapsto \langle w, x \rangle \mid w \in \mathbb{R}^d,\ \|w\|_0 \leqslant s,\ \|w\| \leqslant 1\}.$$

Here, $\|w\|_0 = \sum_{i=1}^d \mathbf{1}\{w_k \neq 0\}$ counts the number of non-zero entries of $w \in \mathbb{R}^d$. There exists a universal positive constant c_0 such that,

$$\mathcal{R}_n(\mathscr{F}_{2,s}) \leqslant c_0\sqrt{\frac{s\log(d/s)}{n}}.$$

Before we turn to the proof of this result, let us remark about its implication. Using Equation (1.27) with $B = L = 1$, for a universal c,

$$\sup_{f \in \mathscr{F}_{2,s}} \operatorname{gen}[f] \leqslant c\sqrt{\frac{s\log(d/s) + \log(1/\delta)}{n}}.$$

Thus, this states that statistically, we can learn s-sparse predictors with ambient dimension d, even when we do not know the support, by only requiring $n \gtrsim s\log(d/s)$ samples. Indeed, $n \ll d$ can hold and this condition on n can still be satisfied. This is related to the compressed sensing literature, where one studies how to recover a sparse model when the number of measurements is less than the ambient dimension of

the parameter vector. However, note that the result we prove is purely statistical: it does not provide prescribe an efficient algorithm for recovering such a sparse model. Indeed, solving the ERM problem over $\mathscr{F}_{2,s}$ requires enumerating all possible subsets of $\{1,\dots,d\}$ of size at most s, and solving an ERM problem restricted to each subset.

Proof of Proposition 1.39. The proof of this result uses the following fact. Let $g \sim N(0, I)$ be an isotropic random Gaussian vector. The random variable $\|g\|$ is c_1-sub-Gaussian, for a universal positive constant. Note that this fact is also dimension free (meaning the constant c_1 does not depend on the dimension of g). This fact is a consequence of Gaussian concentration of Lipschitz functions, which is a quite remarkable result [Vershynin, 2018, cf. Section 5.2.1].

Let $I \subseteq \{1,\dots,d\}$ denote an index set, and let $2^{[d]}$ denote the powerset (subset of all subsets) of $[d] := \{1,\dots,d\}$. Let $P_I : \mathbb{R}^{|I|} \mapsto \mathbb{R}^d$ denote the linear operator which maps a vector of size $|I|$ to a vector of size d, placing the entries of the input into the coordinates listed in I. For example if $d = 3$ and $I = \{1, 3\}$,

$$P_{\{1,3\}} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ 0 \\ x_2 \end{bmatrix}.$$

Observe that by construction, $\|P_I x\| = \|x\|$ for all $x \in \mathbb{R}^{|I|}$. Also observe that the transpose (adjoint operator) of P_I, denoted $P_I^* : \mathbb{R}^d \mapsto \mathbb{R}^{|I|}$, picks out the coordinates of the d-dimensional input vector listed in I to form an $|I|$-diemnsion output vector. For example, again if $d = 3$ and $I = \{1, 3\}$,

$$P^*_{\{1,3\}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix}.$$

Furthermore, let

$$\mathcal{I}_s := \{I \in 2^{[d]} \mid |I| \in \{1,\dots,s\}\}$$

denote the set of all non-empty subsets of $[d]$ which contain at most s elements. A simple counting argument yields that:

$$|\mathcal{I}_s| = \sum_{k=1}^{s} \binom{d}{k}.$$

With this notation in place, and letting g be a d-dimensional isotropic Gaussian random vector,

$$\begin{aligned}
n^{1/2} \cdot \mathcal{R}_n(\mathscr{F}_{2,s}) &= n^{-1/2} \cdot \mathbb{E} \sup_{w \in \mathscr{F}_{2,s}} \sum_{i=1}^{n} \varepsilon_i \langle w, x_i \rangle \\
&= \mathbb{E} \sup_{w \in \mathscr{F}_{2,s}} \left\langle w, n^{-1/2} \sum_{i=1}^{n} \varepsilon_i x_i \right\rangle \\
&= \mathbb{E}_g \sup_{w \in \mathscr{F}_{2,s}} \langle w, g \rangle \\
&= \mathbb{E}_g \sup_{I \in \mathcal{I}_s} \sup_{w \in \mathbb{R}^{|I|}, \|w\| \leqslant 1} \langle P_I w, g \rangle \\
&= \mathbb{E}_g \sup_{I \in \mathcal{I}_s} \sup_{w \in \mathbb{R}^{|I|}, \|w\| \leqslant 1} \langle w, P_I^* g \rangle \\
&= \mathbb{E}_g \sup_{I \in \mathcal{I}_s} \|P_I^* g\|.
\end{aligned}$$

Now, we observe that $P_I^* g$ has the same distribution as an $|I|$-dimensional Gaussian random vector, and therefore by Jensen's inequality $\mathbb{E}_g \|P_I^* g\| \leqslant \sqrt{|I|} \leqslant \sqrt{s}$ for any $I \in \mathcal{I}_s$. Hence,

$$\mathbb{E}_g \sup_{I \in \mathcal{I}_s} \|P_I^* g\| = \mathbb{E}_g \sup_{I \in \mathcal{I}_s} \{\|P_I^* g\| - \mathbb{E}_g \|P_I^* g\| + \mathbb{E}_g \|P_I^* g\|\}$$

$$\leqslant \mathbb{E}_g \sup_{I \in \mathcal{I}_s} \{\|P_I^* g\| - \mathbb{E}_g \|P_I^* g\|\} + \sup_{I \in \mathcal{I}_s} \mathbb{E}_g \|P_I^* g\|$$
$$\leqslant \sqrt{s} + \mathbb{E}_g \sup_{I \in \mathcal{I}_s} \{\|P_I^* g\| - \mathbb{E}_g \|P_I^* g\|\}.$$

Now we use the fact that the random variable $(\|P_I^* g\| - \mathbb{E}\|P_I^* g\|)$ is zero-mean c_1-sub-Gaussian for all I. Hence, by the sub-Gaussian maximal inequality (Proposition B.17),

$$\mathbb{E}_g \sup_{I \in \mathcal{I}_s} \{\|P_I^* g\| - \mathbb{E}_g \|P_I^* g\|\} \leqslant c_1 \sqrt{2 \log |\mathcal{I}_s|} = c_1 \sqrt{2 \log \left(\sum_{k=1}^{s} \binom{d}{k} \right)}.$$

We now use Proposition D.1 to bound:

$$\sum_{k=1}^{s} \binom{d}{k} \leqslant \sum_{k=0}^{s} \binom{d}{k} \leqslant \left(\frac{ed}{s} \right)^s.$$

This bound implies:

$$n^{1/2} \cdot \mathcal{R}_n(\mathscr{F}_{2,s}) \leqslant \sqrt{s} + c_1 \sqrt{2s \log(ed/s)} \Longrightarrow \mathcal{R}_n(\mathscr{F}_{2,s}) \leqslant c_2 \sqrt{\frac{s \log(d/s)}{n}},$$

for some universal constant c_2. □

Beyond linear function classes. We now move beyond linear hypothesis classes. First, we show a negative result, that without making further assumptions on the hypothesis class, the Rademacher complexity does not necessarily need to decay to zero as $n \to \infty$.

Exercise 1.40. Construct a hypothesis class $\mathscr{F} \subseteq [-1, 1]^{\mathsf{X}}$ such that $\mathcal{R}_n(\mathscr{F}) \geqslant 1$ for all $n \in \mathbb{N}_+$.

Next, we will study the hypothesis space of a single hidden-layer neural network with ReLU activations. The following example comes from Ma [2022, Section 5.3].

Proposition 1.41. *Let $\phi(x) = \max\{x, 0\}$ be the ReLU activation function. Consider the hypothesis space of single hidden-layer neural networks with width $m \in \mathbb{N}_+$:*

$$\mathscr{F} = \left\{ x \mapsto \sum_{h=1}^{m} w_h \phi(\langle u_h, x \rangle) \,\middle|\, \{w_h\}_{h=1}^m \subset \mathbb{R}, \{u_h\}_{h=1}^m \subset \mathbb{R}^d, \sum_{h=1}^{m} |w_h| \|u_h\| \leqslant 1 \right\}.$$

We have that:

$$\mathcal{R}_n(\mathscr{F}) \leqslant 2\sqrt{\frac{d}{n}}.$$

Before we proceed with the proof of this result, let us first remark why it is interesting. First, the number of parameters parameterizing functions in $\mathscr{F}$ is $m + m \cdot d = m(d+1)$. Thus, from a parameter counting perspective, we expect the Rademacher complexity of this class to scale as $\sqrt{m(d+1)/n}$. However, this computation shows that by imposing the extra norm constraint $\sum_{h=1}^m |w_i| \|u_h\| \leqslant 1$, the Rademacher scaling improves to $\sqrt{d/n}$, meaning it only *implicitly* depends on the hidden width dimension m via the norm constraint. In fact, if for all $h \in \{1, \ldots, m\}$ and any $s > 1$, we have $\|u_h\| \leqslant 1$ and $|w_h| \leqslant \frac{1}{\zeta(s) \cdot h^s}$ (here, $\zeta(s) := \sum_{i=1}^{\infty} i^{-s}$), then we can let $m \to \infty$ while preserving the $\sqrt{d/n}$ scaling. Indeed, this allows us to have an *infinitely wide* neural network with bounded capacity.

Proof. Let $\theta = (\{w_h\}, \{u_h\})$ denote the parameters of $\mathscr{F}$, and let Θ denote the parameter set. We have:

$$\begin{aligned}
n \cdot \mathcal{R}_n(\mathscr{F}) &= \mathbb{E} \sup_{\theta \in \Theta} \sum_{i=1}^{n} \varepsilon_i \sum_{h=1}^{m} w_h \phi(\langle u_h, x_i \rangle) \\
&= \mathbb{E} \sup_{\theta \in \Theta} \sum_{h=1}^{m} w_h \sum_{i=1}^{n} \varepsilon_i \phi(\langle u_h, x_i \rangle) && \text{by re-arranging sums} \\
&= \mathbb{E} \sup_{\theta \in \Theta} \sum_{h=1}^{m} w_h \|u_h\| \sum_{i=1}^{n} \varepsilon_i \phi(\langle u_h, x_i \rangle / \|u_h\|) && \text{since ReLU is positive homogeneous} \\
&\leqslant \mathbb{E} \sup_{\theta \in \Theta} \max_{h=1,\ldots,m} \left| \sum_{i=1}^{n} \varepsilon_i \phi(\langle u_h, x_i \rangle / \|u_h\|) \right| && \text{since } \sum_{h=1}^{m} |w_i| \|u_h\| \leqslant 1 \\
&\leqslant \mathbb{E} \sup_{\|u\| \leqslant 1} \left| \sum_{i=1}^{n} \varepsilon_i \phi(\langle u, x_i \rangle) \right| && \text{since } u_h / \|u_h\| \text{ is unit-norm.}
\end{aligned}$$

We now prove an quick fact that will allow us to remove the absolute value in the last inequality above.

Proposition 1.42. *Let $T \subset \mathbb{R}^n$ be an arbitrary set, and suppose that:*

$$\forall \varepsilon \in \{\pm 1\}^n, \ \exists\, t \in T \text{ s.t. } \langle \varepsilon, t \rangle \geqslant 0.$$

Then,

$$\mathbb{E} \sup_{t \in T} |\langle \varepsilon, t \rangle| \leqslant 2 \mathbb{E} \sup_{t \in T} \langle \varepsilon, t \rangle, \tag{1.30}$$

where $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_n)$ and the ε_i's are i.i.d. Rademacher random variables.

Proof. Observe that our assumption implies that:

$$\forall \varepsilon \in \{\pm 1\}^n, \ \sup_{t \in T} \langle \varepsilon, t \rangle \geqslant 0. \tag{1.31}$$

Hence, for any $\varepsilon \in \{\pm 1\}^n$,

$$\begin{aligned}
\sup_{t \in T} |\langle \varepsilon, t \rangle| &= \sup_{t \in T} \max_{\sigma \in \{\pm 1\}} \sigma \langle \varepsilon, t \rangle && |x| = \sup_{\sigma \in \{\pm 1\}} \sigma x \text{ for all } x \in \mathbb{R} \\
&= \max \left\{ \sup_{t \in T} \langle \varepsilon, t \rangle, \sup_{t \in T} \langle -\varepsilon, t \rangle \right\} && \text{switching order of maximization} \\
&\leqslant \sup_{t \in T} \langle \varepsilon, t \rangle + \sup_{t \in T} \langle -\varepsilon, t \rangle && \text{using (1.31).}
\end{aligned}$$

To clarify, the last inequality holds since for non-negative a, b, $\max\{a, b\} \leqslant a + b$, and by (1.31) both $\sup_{t \in T} \langle \varepsilon, t \rangle$ and $\sup_{t \in T} \langle -\varepsilon, t \rangle$ are non-negative. The claim now follows by taking expectation and noting that for ε a random Rademacher vector, ε and $-\varepsilon$ have the same distribution. □

We now define the projection set $Q(x_{1:n}) \subset \mathbb{R}^n$ as:

$$Q(x_{1:n}) := \{z = (\phi(\langle u, x_i \rangle))_{i=1}^n \mid \|u\| \leqslant 1\}.$$

Notice that $0 \in Q(x_{1:n})$ always (by setting $u = 0$), so the hypothesis of the previous proposition hold, and therefore we can apply (1.30) to conclude that:

$$n \cdot \mathcal{R}_n(\mathscr{F}) \leqslant \mathbb{E} \sup_{\|u\| \leqslant 1} \left| \sum_{i=1}^{n} \varepsilon_i \phi(\langle u, x_i \rangle) \right|$$

$$
\begin{aligned}
&= \mathbb{E}_{x_{1:n}} \mathbb{E}_\varepsilon \sup_{q \in Q(x_{1:n})} |\langle \varepsilon, q \rangle| \\
&\leqslant 2\mathbb{E}_{x_{1:n}} \mathbb{E}_\varepsilon \sup_{q \in Q(x_{1:n})} \langle \varepsilon, q \rangle && \text{using (1.30)} \\
&= 2\mathbb{E} \sup_{\|u\| \leqslant 1} \sum_{i=1}^n \varepsilon_i \phi(\langle u, x_i \rangle) \\
&\leqslant 2\mathbb{E} \sup_{\|u\| \leqslant 1} \sum_{i=1}^n \varepsilon_i \langle u, x_i \rangle && \text{using Proposition 1.29} \\
&\leqslant 2\sqrt{nd} && \text{using (1.29).}
\end{aligned}
$$

The claim now follows by dividing both sides of the above inequality by n. □

Next, we consider another possibly infinite-dimensional function class, inspired by the theory of reproducing kernel Hilbert spaces (RKHS).

Exercise 1.43. Let $\phi : \mathsf{X} \mapsto \ell_2(\mathbb{N})$ be an infinite-dimensional feature map satisfying $\|\phi(x)\|_\infty \leqslant 1$ for all $x \in \mathsf{X}$. Consider the following hypothesis class:

$$
\mathscr{F} = \left\{ x \mapsto \sum_{i=1}^\infty w_i \phi(x)_i \;\middle|\; \sum_{i=1}^\infty \frac{w_i^2}{\mu_i} \leqslant 1 \right\},
$$

where $(\mu_i)_{i \geqslant 1}$ is a positive element in $\ell_1(\mathbb{N})$ (i.e., $\mu_i \geqslant 0$ and $\sum_{i=1}^\infty \mu_i < \infty$). Show that:

$$
\mathcal{R}_n(\mathscr{F}) \leqslant \sqrt{\frac{\sum_{i=1}^\infty \mu_i}{n}}.
$$

For those familiar with the theory of RKHS, given a reproducing kernel, Mercer's theorem gives us an ellipsoidal representation (of the form written in the previous exercise) of the induced function class, where ϕ corresponds to the eigenfunctions which diagonalize the kernel operator, and $(\mu_i)_{i \geqslant 1}$ corresponds to the resulting eigenvalues.

Finite function classes. We conclude our discussion of Rademacher complexity by showing that the Rademacher complexity is well-controlled for any finite function class. The next result is often referred to as Massart's lemma.

Proposition 1.44 (Massart's lemma)**.** *Let $Q \subset \mathbb{R}^n$ be a finite set of points, and put $M = \max_{q \in Q} n^{-1/2} \|q\|$. Let $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_n)$ denote a Rademacher random vector. Then,*

$$
\mathbb{E}_\varepsilon \max_{q \in Q} n^{-1} \langle \varepsilon, q \rangle \leqslant \sqrt{\frac{2M^2 \log |Q|}{n}}.
$$

Proof. For a fixed $q \in Q$, we have that $n^{-1/2} \langle \varepsilon, q \rangle$ is a $n^{-1/2}\|q\|$-sub-Gaussian random vector. Hence by the sub-Gaussian maximal inequality (Proposition B.17),

$$
\mathbb{E}_\varepsilon \max_{q \in Q} n^{-1} \langle \varepsilon, q \rangle = n^{-1/2} \mathbb{E}_\varepsilon \max_{q \in Q} n^{-1/2} \langle \varepsilon, q \rangle \leqslant \sqrt{\frac{2M^2 \log |Q|}{n}}.
$$

□

The previous result gives us direct control over the Rademacher complexity of a finite class.

Corollary 1.45. *Let $\mathscr{F}$ be a finite function class, and suppose that there exists a finite M satisfying*

$$\sup_{x \in \mathsf{X}, f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} f(x_i)^2 \leqslant M^2.$$

Then, we have that:

$$\mathcal{R}_n(\mathscr{F}) \leqslant \sqrt{\frac{2M^2 \log |\mathscr{F}|}{n}}.$$

Proof. Let $Q(x_{1:n}) := \{(f(x_1), \dots, f(x_n)) \mid f \in \mathscr{F}\} \subset \mathbb{R}^n$ denote the empirical projection of $\mathscr{F}$ onto $\mathbb{R}^n$. By assumption, we have that for any $q \in Q(x_{1:n})$, $n^{-1/2}\|q\| \leqslant M$. Hence,

$$\begin{aligned}
\mathcal{R}_n(\mathscr{F}) &= \mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} \varepsilon_i f(x_i) \\
&= \mathbb{E}_{x_{1:n}} \mathbb{E}_{\varepsilon} \sup_{q \in Q(x_{1:n})} n^{-1} \langle \varepsilon, q \rangle \\
&\leqslant \mathbb{E}_{x_{1:n}} \sqrt{\frac{2M^2 \log |Q(x_{1:n})|}{n}} && \text{using Proposition 1.44} \\
&\leqslant \sqrt{\frac{2M^2 \log |\mathscr{F}|}{n}} && \text{since } |Q(x_{1:n})| \leqslant |\mathscr{F}|.
\end{aligned}$$

□

1.4.10 Vapnik-Chervonenkis dimension

We now return to our discussion of the Vapnik-Chervonenkis dimension (abbreviated VC-dimension) that we started in Section 1.4.8. Recall from that discussion that with probability at least $1 - \delta$, for all $f \in \mathscr{F}$,

$$\mathbb{P}\{\operatorname{sgn}(f(x)) \neq y\} \leqslant \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{\operatorname{sgn}(f(x_i)) \neq y_i\} + \mathcal{R}_n(\{x \mapsto \operatorname{sgn}(f(x)) \mid f \in \mathscr{F}\}) + 2\sqrt{\frac{2 \log(1/\delta)}{n}}. \tag{1.32}$$

Let $\mathscr{H}$ denote the predictor class $\mathscr{H} = \{x \mapsto \operatorname{sgn}(f(x)) \mid f \in \mathscr{F}\}$. At this point, it is not immediately obvious how to analyze the Rademacher complexity $\mathcal{R}_n(\mathscr{H})$ which appears in the expression above. It turns out, as we will see, that this quantity is precisely controlled by the VC-dimension of $\mathscr{H}$.

To define the VC-dimension, we start with the notion of a growth function. For a fixed set of datapoints $x_{1:n}$, let $\mathscr{H}(x_{1:n}) \subseteq \{\pm 1\}^n$ denote the projection set:

$$\mathscr{H}(x_{1:n}) = \{(h(x_1), \dots, h(x_n)) \mid h \in \mathscr{H}\}.$$

Specficially, the *Vapnik-Chervonenkis dimension* $\operatorname{VCdim}(\mathscr{H})$ is the following quantity:

$$\operatorname{VCdim}(\mathscr{H}) := \sup\{n \in \mathbb{N}_+ \mid \exists\, x_{1:n} \text{ s.t. } |\mathscr{H}(x_{1:n})| = 2^n\}. \tag{1.33}$$

Let us take a moment to unpack this abstract definition. To do this, we first discuss the notion of shattering.

Shattering. First, we make the simple observation that $|\mathscr{H}(x_{1:n})| \leqslant 2^n$ for any $\mathscr{H}$ and $x_{1:n}$, since there are at most 2^n possible values for an n-length binary string. Thus, the condition $|\mathscr{H}(x_{1:n})| = 2^n$ states that every n-length binary string can be realized by the projection set $\mathscr{H}(x_{1:n})$; this condition is referred to as *$\mathscr{H}$ shatters $x_{1:n}$*. Put in the language of classification, this condition states that for any set of labels $(y_1, \dots, y_n) \in \{\pm 1\}^n$, there exists a hypothesis $h \in \mathscr{H}$ such that its predictions on $x_{1:n}$ precisely match the given labels:

$$(h(x_1), \dots, h(x_n)) = (y_1, \dots, y_n).$$

Intuitively, we expect function classes $\mathscr{H}$ which are richer to be able to shatter $x_{1:n}$ for larger values of n.

With the notion of shattering in place, we see that $\mathrm{VCdim}(\mathscr{H})$ is defined as the largest cardinality $n \in \mathbb{N}_+$ such that there *exists* a set of data points $x_{1:n}$ that is shattered by $\mathscr{H}$. Note that if for every $n \in \mathbb{N}_+$ there exists $x_{1:n}$ that is shattered by $\mathscr{H}$, then $\mathrm{VCdim}(\mathscr{H}) = \infty$.

The choice of the existential qualifier here is important. For example, convince yourself that if we had replaced the $\exists$ quantifier with the $\forall$ quantifier in the definition of VC-dimension (Equation (1.33)), that the resulting definition would not be very useful. Also note that the definition of shattering satisfies two monotonicity properties:

(i) If $\mathscr{H}$ shatters $x_{1:n}$ for some $n \in \mathbb{N}_+$, then $\mathscr{H}$ also shatters $x_{1:k}$ for every $k \in \{1, \dots, n-1\}$.

(ii) Suppose that $\mathscr{F}' \supseteq \mathscr{F}$. Then its corresponding predictor class $\mathscr{H}' = \{x \mapsto \operatorname{sgn}(f(x)) \mid f \in \mathscr{F}'\}$ satisfies $\mathscr{H}' \supseteq \mathscr{H}$. Consequently, if $\mathscr{H}$ shatters $x_{1:n}$, then so does $\mathscr{H}'$, since $\mathscr{H}(x_{1:n}) \subseteq \mathscr{H}'(x_{1:n})$. Hence, VC-dimension is monotonic: $\mathrm{VCdim}(\mathscr{H}) \leqslant \mathrm{VCdim}(\mathscr{H}')$.

Computing the VC-dimension. In light of our discussion, in order to compute $\mathrm{VCdim}(\mathscr{H})$ for a given hypothesis class $\mathscr{H}$, one needs to check two conditions:

(a) Find an $n \in \mathbb{N}_+$ and a specific dataset $x_{1:n}$ such that $\mathscr{H}$ shatters $x_{1:n}$, i.e., $|\mathscr{H}(x_{1:n})| = 2^n$.

(b) Show that for *every* dataset $x_{1:n+1}$, $\mathscr{H}$ *cannot* shatter $x_{1:n+1}$, i.e., $|\mathscr{H}(x_{1:n+1})| < 2^{n+1}$.

If both conditions (a) and (b) are met, then that proves that $\mathrm{VCdim}(\mathscr{H}) = n$. This is because condition (a) shows that $\mathrm{VCdim}(\mathscr{H}) \geqslant n$, and condition (b) shows that $\mathrm{VCdim}(\mathscr{H}) < n+1$.

Shortly, we will compute specific examples of VC-dimension to get more intuition. But first, we will relate the VC-dimension to the Rademacher complexity of $\mathcal{R}_n(\mathscr{H})$. To do this, we introduce one last concept. The *growth function* $\tau_{\mathscr{H}}(n)$ is the following quantity:

$$\tau_{\mathscr{H}}(n) := \sup_{x_{1:n}} |\mathscr{H}(x_{1:n})|.$$

By the definition of VC-dimension, for any $n \leqslant \mathrm{VCdim}(\mathscr{H})$, we immediately have $\tau_{\mathscr{H}}(n) = 2^n$. That is, the growth function is exponential in n in this regime. However, it turns out that when $n > \mathrm{VCdim}(\mathscr{H})$, this growth function undergoes a phase transition and the growth turns to *polynomial* in n. This is the contents of the well-known Sauer-Shelah lemma. Specifically, letting $d = \mathrm{VCdim}(\mathscr{H})$,

$$\tau_{\mathscr{H}}(n) \leqslant \sum_{i=0}^{d} \binom{n}{i} \quad \text{and} \quad \sum_{i=0}^{d} \binom{n}{i} \leqslant \left(\frac{en}{d}\right)^d \text{when } n \geqslant d. \tag{1.34}$$

The proof of this result is out of scope for this course, but can be found in [Vershynin, 2018, Section 8.3.3]. The estimate on the RHS of (1.34) follows from Proposition D.1. With this result in place, we have all the tools needed to connect the Rademacher complexity $\mathcal{R}_n(\mathscr{H})$ with the growth function $\tau_{\mathscr{H}}(n)$

Proposition 1.46. *Suppose that $\mathscr{H} = \{x \mapsto \operatorname{sgn}(f(x)) \mid f \in \mathscr{F}\}$ has finite $\mathrm{VCdim}(\mathscr{H}) = d$. Then,*

$$\mathcal{R}_n(\mathscr{H}) \leqslant \sqrt{\frac{2 \log \tau_{\mathscr{H}}(n)}{n}}.$$

Proof. The idea is to use the definition of the growth function $\tau_{\mathscr{H}}(n)$ to reduce the Rademacher complexity to a maximum over a finite set of hypothesis, and appeal to Massart's lemma (Proposition 1.44). Specifically,

$$\begin{aligned}
\mathcal{R}_n(\mathscr{H}) &= \mathbb{E} \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^{n} \varepsilon_i \operatorname{sgn}(f(x_i)) \\
&= \mathbb{E} \sup_{h \in \mathscr{H}(x_{1:n})} n^{-1} \langle \varepsilon, h \rangle \qquad \text{definition of } \mathscr{H}(x_{1:n})
\end{aligned}$$

$$
\begin{aligned}
&= \mathbb{E}_{x_{1:n}}\left[\mathbb{E}_{\varepsilon} \sup_{h \in \mathscr{H}(x_{1:n})} n^{-1}\langle \varepsilon, h\rangle\right] \\
&\leqslant \mathbb{E}_{x_{1:n}} \sqrt{\frac{2\log|\mathscr{H}(x_{1:n})|}{n}} && \text{Massart's lemma (Proposition 1.44)} \\
&\leqslant \sqrt{\frac{2\log \tau_{\mathscr{H}}(n)}{n}}. && \text{definition of growth function}
\end{aligned}
$$

□

Now, combining the generalization bound (1.32), the Sauer-Shelah growth function bound (1.34), with the result from Proposition 1.46 yields the following key result.

Theorem 1.47. *Let $\mathscr{F}$ denote a function class and let $\mathscr{H} = \{x \mapsto \operatorname{sgn}(f(x)) \mid f \in \mathscr{F}\}$ denote its corresponding predictor class. Suppose that the predictor class $\mathscr{H}$ has finite $\operatorname{VCdim}(\mathscr{H}) = d$. There exists a universal constant $c > 0$ such that for any $n \geqslant d+1$, with probability at least $1-\delta$,*

$$
\forall f \in \mathscr{F},\ \mathbb{P}\{\operatorname{sgn}(f(x)) \neq y\} \leqslant \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{\operatorname{sgn}(f(x_i)) \neq y_i\} + c\sqrt{\frac{d\log(n/d)}{n}} + 2\sqrt{\frac{2\log(1/\delta)}{n}}.
$$

We now take a moment to compare the generalization bound from Theorem 1.47 with the previously derived bounds in Theorem 1.32 and Corollary 1.33, respectively.

Theorem 1.32 vs. Theorem 1.47. Theorem 1.32 bounds the generalization error of Lipschitz losses, and uses the Rademacher complexity $\mathcal{R}_n(\mathscr{F})$ to measure complexity. On the other hand, Theorem 1.47 is specific to the zero-one loss, and uses $\operatorname{VCdim}(\mathscr{H})$ of the induced predictor class $\mathscr{H}$ to measure complexity.

Corollary 1.33 vs. Theorem 1.47. Both results bound the population zero-one loss $\mathbb{P}\{\operatorname{sgn}(f(x)) \neq y\}$. However, the main difference is on the right hand sides. For Corollary 1.33, the Rademacher complexity $\mathcal{R}_n(\mathscr{F})$ is still used to measure complexity (similar to Theorem 1.47). However, the price to pay for this is that training error measures the *margin mistakes* $\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{y_i f(x_i) < \gamma\}$ for $\gamma > 0$; having no margin ($\gamma = 0$) is not allowed in Corollary 1.33. On the other hand, Theorem 1.47 does *not* require any margin, and directly counts the number of classification mistakes $\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{\operatorname{sgn}(f(x_i)) \neq y_i\}$ made on the training dataset. The price to pay for this, however, is that the complexity measure is no longer the more intuitive Rademacher complexity $\mathcal{R}_n(\mathscr{F})$, but instead it becomes the VC-dimension $\operatorname{VCdim}(\mathscr{H})$ of the induced predictor class $\mathscr{H}$.

In light of this discussion, we see that none of the bounds (Theorem 1.32, Corollary 1.33, and Theorem 1.47) are strictly better than the others, and involve different sets of assumptions and complexity measures. Depending on the specific problem at hand, one may be better than the other.

Exercise 1.48. Let $\{\mathscr{H}_i\}_{i=1}^{k}$ be k hypothesis classes. Put $d := \max_{i=1,\dots,k} \operatorname{VCdim}(\mathscr{H}_i)$, and suppose that d is finite. Show that there exists universal constants c_0, c_1 such that

$$
\operatorname{VCdim}(\cup_{i=1}^{k}\mathscr{H}_i) \leqslant c_0 d + c_1 \log k.
$$

Hint: use the Sauer-Shelah lemma and Proposition D.2.

1.4.11 Explicit computations of VC-dimension

We now proceed to compute the VC-dimension for several specific predictor classes.

Threshold functions. Let us consider first consider the predictor class

$$\mathscr{H} = \{x \mapsto h_t(x) := \operatorname{sgn}(t - x) \mid t \in \mathbb{R}\}.$$

This is the set of all threshold functions in one-dimension; any $h_t \in \mathscr{H}$ is equal to:

$$h_t(x) = \begin{cases} 1 & \text{if } x \leqslant t \\ -1 & \text{o.w.} \end{cases}$$

We will show that $\operatorname{VCdim}(\mathscr{H}) = 1$. Following the recipe from Section 1.4.10, we first check there exists a dataset of one point can be shattered by $\mathscr{H}$. We choose this dataset as $x_1 = 0$. Note that if we choose $t = 1$, then $h_1(x_1) = h_1(0) = \operatorname{sgn}(1) = 1$. On the other hand, if we choose $t = -1$, then $h_{-1}(x_1) = h_{-1}(0) = \operatorname{sgn}(-1) = -1$, so $|\mathscr{H}(x_1)| = 2$. Now we show that no $n = 2$ points can be shattered. Without loss of generality assume that $x_1 \leqslant x_2$. Notice that the labels $(y_1, y_2) = (-1, +1)$ are not contained in $\mathscr{H}(x_{1:2})$, since for any $t \in \mathbb{R}$ such that $h_t(x_2) = 1$, then $x_2 \leqslant t$ which implies that $x_1 \leqslant t$, but which then implies $h_t(x_1) = 1$. Hence $|\mathscr{H}(x_{1:2})| < 4$.

Interval functions. Now let us consider the prediction class

$$\mathscr{H} = \{x \mapsto h_{a,b}(x) := \operatorname{sgn}(\mathbf{1}\{x \in [a, b]\} - 1/2) \mid a, b \in \mathbb{R},\ a \leqslant b\}.$$

Note that $h_{a,b} \in \mathscr{H}$ is equal to:

$$h_{a,b}(x) = \begin{cases} 1 & \text{if } x \in [a, b] \\ -1 & \text{o.w.} \end{cases}$$

We now show that $\operatorname{VCdim}(\mathscr{H}) = 2$. Again, following the recipe from Section 1.4.10, we first construct a dataset of $n = 2$ points that can be shattered by $\mathscr{H}$. Let this dataset be $x_1 = 0$ and $x_2 = 1$. Now we need to check that all four labelings $(y_1, y_2) \in \{\pm 1\}^2$ are contained in $\mathscr{H}(x_{1:2})$. We do this by enumerating all four cases and choosing values of $(a, b) \in \mathbb{R}^2$ such $(h_{a,b}(x_1), h_{a,b}(x_2)) = (y_1, y_2)$:

(i) Case $(y_1, y_2) = (-1, -1)$: select $a = b = 1/2$.

(ii) Case $(y_1, y_2) = (-1, +1)$: select $a = b = 1$.

(iii) Case $(y_1, y_2) = (+1, -1)$: select $a = b = 0$.

(iv) Case $(y_1, y_2) = (+1, +1)$: select $a = 0$, $b = 1$.

Next, we need to show that any set of $n = 3$ points cannot be shattered by $\mathscr{H}$. Note that we can assume wlog that $x_1 < x_2 < x_3$, since if there are any equalities, then clearly $|\mathscr{H}(x_{1:3})| < 8$. Now we will argue that for any $x_1 < x_2 < x_3$, the label $(y_1, y_2, y_3) = (+1, -1, +1)$ cannot be achieved. To see this, the first requirement that $h_{a,b}(x_1) = 1$ implies that $a \leqslant x_1$. On the other hand, the second requirement that $h_{a,b}(x_2) = -1$ then requires that $b \leqslant x_2$. But then this necessitates that $h_{a,b}(x_3) = -1$ since $x_3 > x_2 \geqslant b$. Hence, $|\mathscr{H}(x_{1:3})| < 8$.

Linear functions. We now consider the set of d-dimensional linear functions:

$$\mathscr{H} = \{x \mapsto h_w(x) := \operatorname{sgn}(\langle x, w \rangle) \mid w \in \mathbb{R}^d\}.$$

We will show that $\operatorname{VCdim}(\mathscr{H}) = d$. First, let $e_i \in \mathbb{R}^d$ denote the i-th standard basis vector. Set $x_i = e_i$ for $i \in \{1, \ldots, d\}$. Now for a given labeling $(y_1, \ldots, y_n) \in \{\pm 1\}^d$, set $w = (y_1, \ldots, y_n)$, and observe that:

$$h_w(x_i) = \operatorname{sgn}(\langle x_i, w \rangle) = \operatorname{sgn}(\langle e_i, w \rangle) = \operatorname{sgn}(w_i) = \operatorname{sgn}(y_i) = y_i.$$

This show that $\mathscr{H}$ shatters $x_{1:d}$.

Now we show that no set of $d+1$ points can be shattered by $\mathscr{H}$. Let $x_{1:d+1}$ be arbitrary. Since the x_i's live in a d-dimensional vector space, the points $x_{1:d+1}$ must be linearly dependent. This means there exists a non-zero $\alpha \in \mathbb{R}^{d+1}$ such that

$$\sum_{i=1}^{d+1} \alpha_i x_i = 0.$$

Let $I_+ = \{i \in \{1, \dots, d+1\} \mid \alpha_i \geqslant 0\}$ and $I_- = \{i \in \{1, \dots, d+1\} \mid \alpha_i < 0\}$. Wlog, let us assume that α has at least one negative element, that is $|I_-| \geqslant 1$ (otherwise, negate α). Re-arranging the above equation,

$$\sum_{i \in I_+} \alpha_i x_i = -\sum_{i \in I_-} \alpha_i x_i = \sum_{i \in I_-} |\alpha_i| x_i. \tag{1.35}$$

Now let us suppose $x_{1:d+1}$ is shattered by $\mathscr{H}$. Choose the labeling $(y_1, \dots, y_{d+1}) \in \{\pm 1\}^{d+1}$

$$y_i = \begin{cases} 1 & \text{if } i \in I_+, \\ -1 & \text{if } i \in I_-, \end{cases}$$

and let $w \in \mathbb{R}^d$ be such that $h_w(x_i) = y_i$ for $i \in \{1, \dots, d+1\}$; such a w must exist by the definition of shattering. Notice this means that for all $i \in I_+$, $\langle w, x_i \rangle \geqslant 0$, and for all $i \in I_-$, $\langle w, x_i \rangle < 0$. However, using (1.35),

$$0 > \sum_{i \in I_-} |\alpha_i| \langle w, x_i \rangle = \left\langle w, \sum_{i \in I_-} |\alpha_i| x_i \right\rangle = \left\langle w, \sum_{i \in I_+} \alpha_i x_i \right\rangle = \sum_{i \in I_+} \alpha_i \langle w, x_i \rangle \geqslant 0,$$

which is a contradiction. Note that the strict inequality above follows since $|I_-| \geqslant 1$. Thus, $\mathscr{H}$ cannot shatter $x_{1:d+1}$.

Finite hypothesis class. Suppose now we consider the prediction class:

$$\mathscr{H} = \{x \mapsto \operatorname{sgn}(f(x)) \mid f \in \mathscr{F}\}, \quad |\mathscr{F}| < \infty.$$

Then immediate we have that for any $x_{1:n}$, $|\mathscr{H}(x_{1:n})| \leqslant |\mathscr{F}|$. Hence for any $n > \log_2 |\mathscr{F}|$, we must have $|\mathscr{H}(x_{1:n})| \leqslant |\mathscr{F}| < 2^n$. Hence, this yields an upper bound of $\operatorname{VCdim}(\mathscr{H}) \leqslant \log_2 |\mathscr{F}|$. Note that this upper bound can be very loose though. For example, consider the subset of threshold functions:

$$\mathscr{H}' = \{x \mapsto \operatorname{sgn}(t - x) \mid t \in I\},$$

where I is a finite subset of $\mathbb{R}$. Thus, this finite class bound yields that $\operatorname{VCdim}(\mathscr{H}') \leqslant \log_2 |I|$, but since $\mathscr{H}' \subset \mathscr{H}'' = \{x \mapsto \operatorname{sgn}(x - t) \mid t \in \mathbb{R}\}$ and we know that $\operatorname{VCdim}(\mathscr{H}'') = 1$ from our previous analysis, we actually by monotonicity of VC-dimension that $\operatorname{VCdim}(\mathscr{H}') \leqslant \operatorname{VCdim}(\mathscr{H}'') = 1$. Therefore, the finite class bound can be arbitrarily loose.

An example of infinite VC-dimension. We now consider a one parameter prediction class, and show that it has infinite VC-dimension. The prediction class we consider is:

$$\mathscr{H} = \{x \mapsto h_\theta(x) := \operatorname{sgn}(\sin(\theta x)) \mid \theta \in \mathbb{R}\}.$$

Fix any $n \in \mathbb{N}_+$, and consider the dataset $x_{1:n}$ where $x_i = 10^{-i}$. We will show that $\mathscr{H}$ shatters $x_{1:n}$. Let $(y_1, \dots, y_n) \in \{\pm 1\}^n$ be any labeling, and let $I_- = \{i \in \{1, \dots, n\} \mid y_i = -1\}$. Choose the frequency parameter θ as:

$$\theta = \pi \left(1 + \sum_{i \in I_-} 10^i \right).$$

Now for any $j \in \{1, \dots, n\}$, we compute:

$$\begin{aligned}\theta x_j &= \pi 10^{-j} + \pi \sum_{i \in I_-} 10^{i-j} \\ &= \pi 10^{-j} + \pi \mathbf{1}\{j \in I_-\} + \pi \sum_{i \in I_-, i<j} 10^{i-j} + \pi \sum_{i \in I_-, i>j} 10^{i-j}.\end{aligned}$$

We next observe that, whenever $i > j$, then 10^{i-j} is an even integer, and hence $\pi \sum_{i \in I_-, i>j} 10^{i-j} = 2\pi k(j)$ for a non-negative integer $k(j)$. On the other hand, we can bound,

$$0 \leqslant \varepsilon(j) := \sum_{i \in I_-, i<j} 10^{i-j} \leqslant \sum_{i=1}^{\infty} 10^{-i} = \frac{1}{1 - 1/10} - 1 = 1/9.$$

Hence,

$$\begin{aligned}\sin(\theta x_j) &= \sin(\pi 10^{-j} + \pi \mathbf{1}\{j \in I_-\} + \pi\varepsilon(j) + 2\pi k(j)) \\ &= \sin(\pi 10^{-j} + \pi \mathbf{1}\{j \in I_-\} + \pi\varepsilon(j)) && \text{since } \sin(x + 2\pi k) = \sin(x) \text{ for } k \in \mathbb{N} \\ &= \sin(\pi(\mathbf{1}\{j \in I_-\} + 10^{-j} + \varepsilon(j))).\end{aligned}$$

Next defining $\bar{\varepsilon}(j) := 10^{-j} + \varepsilon(j)$, we observe that $0 < \bar{\varepsilon}(j) \leqslant 1/10 + 1/9 < 1$. Therefore, let us first suppose that $j \notin I_-$, so that $y_j = +1$. Then, since $\sin(x) > 0$ for $x \in (0, \pi)$,

$$\sin(\theta x_j) = \sin(\pi\bar{\varepsilon}(j)) > 0 \Longrightarrow h_\theta(x_j) = \operatorname{sgn}(\sin(\theta x_j)) = y_j.$$

On the other hand, suppose that $j \in I_-$, so that $y_j = -1$. Then, since $\sin(x) < 0$ for $x \in (\pi, 2\pi)$,

$$\sin(\theta x_j) = \sin(\pi(1 + \bar{\varepsilon}(j))) < 0 \Longrightarrow h_\theta(x_j) = \operatorname{sgn}(\operatorname{sgn}(\theta x_j)) = y_j,$$

This shows that $\mathscr{H}$ shatters $x_{1:n}$. Since the choices of n and $x_{1:n}$ are arbitrary, we have that $\mathrm{VCdim}(\mathscr{H}) = \infty$.

1.4.12 Agnostic PAC learnability and the No Free Lunch theorem

We now have all the tools in place to formally define what it means for a hypothesis to be *learnable*. Furthermore, we will consider the question of whether or not learnability is even possible in general. In this section, we will focus on the zero-one loss.

We start with a general definition of learnability.

Definition 1.49 (Agnostically PAC learnable)**.** *A hypothesis class $\mathscr{H}$ of functions mapping $\mathsf{X} \mapsto \mathsf{Y}$ is called* agnostically probably approximately correct learnable *(agnostically PAC learnable) if there exists an algorithm $\mathcal{A} : \mathsf{Z}^* \mapsto \mathscr{H}$ with the following property. For every $\varepsilon, \delta \in (0, 1)$, there exists a finite $n = n(\varepsilon, \delta) \in \mathbb{N}_+$ such that running $\mathcal{A}$ on n iid examples $S_n = \{(x_i, y_i)\}_{i=1}^n$ drawn from any distribution $\mathcal{D}$ on $\mathsf{Z} = \mathsf{X} \times \mathsf{Y}$ satisfies the following property. With probability at least $1 - \delta$ over the randomness of S_n, $\mathcal{A}$ returns a hypothesis $\hat{h}_n = \mathcal{A}(S_n)$ such that*

$$\mathbb{P}_{\mathcal{D}}\{y \neq \hat{h}_n(x)\} \leqslant \inf_{h \in \mathscr{H}} \mathbb{P}_{\mathcal{D}}\{y \neq h(x)\} + \varepsilon. \tag{1.36}$$

The function $n(\varepsilon, \delta)$ is referred to as the sample complexity *of learning.*

We now highlight a few key characteristics of Definition 1.49.

(a) Agnostic PAC learnability does *not* prescribe a rate of learning; the sample complexity $n(\varepsilon, \delta)$ can scale arbitrarily poorly on ε, δ, it just needs to be finite for every fixed ε, δ.

(b) Agnostic PAC learnability also does *not* require computational efficiency. The algorithm $\mathcal{A}$ does not have to be efficiency computable, it just has to be a well-defined mathematical function.

(c) Agnostic PAC learnability does not require realizability (i.e., $\exists h \in \mathscr{F}$ such that $\mathbb{P}_{\mathcal{D}}\{y \neq h(x)\} = 0$). It simply requires that the learning algorithm $\mathcal{A}$ emits a hypothesis $h \in \mathscr{H}$ that can compete with the best possible predictor in $\mathscr{H}$. In other words, agnostic PAC learnability is still meaningful for $\inf_{h\in\mathscr{H}} \mathbb{P}_{\mathcal{D}}\{y \neq h(x)\}$ arbitrarily close to 1. On the other hand, standard PAC learnability (without the agnostic) requires realizability.

(d) Note that the order of quantifiers in the definition is extremely important. For example, if the definition were to require instead that for every distribution $\mathcal{D}$ over Z, there exists an algorithm $\mathcal{A}$ satisfying (1.36), then the definition would be vacuous (why?).

With the definition of agnostically PAC learnable in place, we are ready to prove our first result providing a sufficient condition for agnostic PAC learnability.

Theorem 1.50 (Finite VC-dimension implies learnability)**.** *Let $\mathscr{H}$ have finite VC-dimension d. Then, $\mathscr{H}$ is agnostically PAC learnable via the empirical risk minimizer. Furthermore, it suffices to take n satisfying:*

$$n \geqslant \frac{c_0}{\varepsilon^2}\left[d\log\left(\frac{2}{\varepsilon}\right) + \log\left(\frac{2}{\delta}\right)\right],$$

where c_0 is a universal positive constant.

Proof. Fix $\varepsilon, \delta \in (0,1)$. We already saw by Theorem 1.47, that for any distribution over Z, for any $n \geqslant d+1$, with probability at least $1-\delta/2$ over the samples $\{(x_i, y_i)\}_{i=1}^n$,

$$\forall h \in \mathscr{H},\ \mathbb{P}\{y \neq h(x)\} \leqslant \frac{1}{n}\sum_{i=1}^n \mathbf{1}\{y_i \neq h(x_i)\} + c\sqrt{\frac{d\log(n/d)}{n}} + 2\sqrt{\frac{2\log(2/\delta)}{n}}. \tag{1.37}$$

Here, c is a universal constant. We will assume $c \geqslant 1$ without loss of generality. Note that by Equation (1.23), Theorem 1.47 also tells us that with probability at least $1-\delta/2$ over the samples,

$$\forall h \in \mathscr{H},\ \frac{1}{n}\sum_{i=1}^n \mathbf{1}\{y_i \neq h(x_i)\} \leqslant \mathbb{P}\{y \neq h(x)\} + c\sqrt{\frac{d\log(n/d)}{n}} + 2\sqrt{\frac{2\log(2/\delta)}{n}}. \tag{1.38}$$

By a union bound, with probability at least $1-\delta$, both (1.37) and (1.38) hold. We will work assuming this event holds for the remainder of the proof.

Setting $\hat{h}_n \in \arg\min_{h\in\mathscr{H}} \frac{1}{n}\sum_{i=1}^n \mathbf{1}\{y_i \neq h(x_i)\}$ to be the ERM and letting $h' \in \mathscr{H}$ be arbitrary,

$$\begin{aligned}
\mathbb{P}\{y \neq \hat{h}_n(x)\} &\leqslant \frac{1}{n}\sum_{i=1}^n \mathbf{1}\{y_i \neq \hat{h}_n(x_i)\} + c\sqrt{\frac{d\log(n/d)}{n}} + 2\sqrt{\frac{2\log(2/\delta)}{n}} && \text{using (1.37)} \\
&\leqslant \frac{1}{n}\sum_{i=1}^n \mathbf{1}\{y_i \neq h'(x_i)\} + c\sqrt{\frac{d\log(n/d)}{n}} + 2\sqrt{\frac{2\log(2/\delta)}{n}} && \text{since } \hat{h}_n \text{ is the ERM} \\
&\leqslant \mathbb{P}\{y \neq h'(x)\} + 2c\sqrt{\frac{d\log(n/d)}{n}} + 4\sqrt{\frac{2\log(2/\delta)}{n}}. && \text{using (1.38)}
\end{aligned} \tag{1.39}$$

Since (1.39) holds for any $h' \in \mathscr{H}$, we can take the infimum over the RHS to conclude:

$$\mathbb{P}\{y \neq \hat{h}_n(x)\} \leqslant \inf_{h\in\mathscr{H}} \mathbb{P}\{y \neq h(x)\} + 2c\sqrt{\frac{d\log(n/d)}{n}} + 4\sqrt{\frac{2\log(2/\delta)}{n}}.$$

Now we simply need to set n large enough so the RHS is $\leqslant \varepsilon$. We will choose a sufficient (but not necessary) condition instead. We will ask that n be large enough so that:

$$2c\sqrt{\frac{d\log(n/d)}{n}} \leqslant \frac{\varepsilon}{2} \quad \text{and} \quad 4\sqrt{\frac{2\log(2/\delta)}{n}} \leqslant \frac{\varepsilon}{2}. \tag{1.40}$$

Clearly if both these conditions hold, then we have achieved the agnostic PAC learnability condition (1.36):

$$\mathbb{P}\{y \neq \hat{h}_n(x)\} \leqslant \inf_{h \in \mathscr{H}} \mathbb{P}\{y \neq h(x)\} + \varepsilon.$$

We now turn to ensuring (1.40). The RHS of (1.40) is simple to ensure by requiring that:

$$n \geqslant \frac{128\log(2/\delta)}{\varepsilon^2}.$$

Turning to the LHS of Equation (1.40), we first re-arrange the condition as:

$$n \geqslant \left(\frac{4c}{\varepsilon}\right)^2 d\log(n/d). \tag{1.41}$$

Given the appearance of the n inside the logarithm on the RHS of (1.41), it is not straightforward to precisely solve for the smallest n satisfying (1.41). We can, however, derive a sufficient condition. In particular, Proposition D.2 states that if α, β are positive reals such that $\alpha\beta \geqslant 1$, then for any $x \in \mathbb{R}$:

$$x \geqslant 2\alpha\log(2\alpha\beta) \implies x \geqslant \alpha\log(\beta x). \tag{1.42}$$

We now set:

$$\alpha = \left(\frac{4c}{\varepsilon}\right)^2 d, \quad \beta = \frac{1}{d}.$$

To verify that $\alpha\beta \geqslant 1$, since $c \geqslant 1$ and $\varepsilon \in (0,1)$, we have:

$$\alpha\beta = \left(\frac{4c}{\varepsilon}\right)^2 d \cdot \frac{1}{d} = \left(\frac{4c}{\varepsilon}\right)^2 \geqslant 1.$$

Thus, by (1.42), a sufficient condition for (1.41) to hold is:

$$n \geqslant \frac{32c^2 d}{\varepsilon^2}\log\left(\frac{32c^2}{\varepsilon^2}\right) \implies n \geqslant \left(\frac{4c}{\varepsilon}\right)^2 d\log(n/d).$$

Combining our two requirements on n, we see that for the final sample complexity bound, it suffices to take

$$n \geqslant \max\left\{\frac{32c^2 d}{\varepsilon^2}\log\left(\frac{32c^2}{\varepsilon^2}\right), \frac{128\log(2/\delta)}{\varepsilon^2}\right\}.$$

□

We note that the sample complexity bound given in Theorem 1.50 is sub-optimal order-wise. It can actually be improved to:

$$n \geqslant \frac{c_0}{\varepsilon^2}\left[d + \log\left(\frac{2}{\delta}\right)\right],$$

using a technique called *chaining*. This more advanced technique is developed in Section 1.4.13, although we will not have time to discuss it in class.

At this point we have seen that finite VC-dimension implies agnostic PAC learnability. What about the converse? Is it possible that hypothesis classes with infinite VC-dimension are still agnostically PAC learnable? We will show that this is not possible. The main step towards showing this is the following classic "no free lunch" theorem.

Theorem 1.51 (No Free Lunch)**.** *Let X be a domain of infinite cardinality, and let $\mathscr{H}$ denote hypothesis class of functions mapping $\mathsf{X} \mapsto \mathsf{Y}$ with infinite VC-dimension (i.e., $\mathrm{VCdim}(\mathscr{H}) = \infty$). Fix $n \in \mathbb{N}_+$. Let $\mathcal{A} : \mathsf{Z}^n \mapsto \mathscr{H}$ be any algorithm which takes in as input n pairs $S_n = \{(x_i, y_i)\}_{i=1}^n$ and outputs a hypothesis in $\mathscr{H}$. Then, there exists a distribution $\mathcal{D}$ over Z such that the following hold:*

(i) There exists an $h \in \mathscr{H}$ satisfying $\mathbb{P}_{\mathcal{D}}\{h(x) \neq y\} = 0$.

(ii) Let S_n consist of n iid draws from $\mathcal{D}$. With probability at least $1/7$ over the randomness of S_n,

$$\mathbb{P}_{\mathcal{D}}\{\mathcal{A}(S_n)(x) \neq y\} \geqslant \frac{1}{8}.$$

Proof. The proof of this result relies on the probabilistic method. To do this, we set up distributions μ and ν over X and $\mathscr{H}$, respectively. Since $\mathrm{VCdim}(\mathscr{H}) = \infty$, there exists $2n$ distinct points $x'_{1:2n} = (x'_1, \ldots, x'_{2n}) \subset \mathsf{X}$ such that the projection set $\mathscr{H}(x'_{1:2n})$ has cardinality 2^{2n}. This implies there exists a subset of $\mathscr{H}'_{2n}$ of $\mathscr{H}$ containing 2^{2n} distinct functions such that $\mathscr{H}'_{2n}(x'_{1:2n}) = \mathscr{H}(x'_{1:2n})$. Importantly, note there is a bijection between $\mathscr{H}'_{2n}(x'_{1:2n})$ and $\{\pm 1\}^{2n}$. We set μ to be the uniform distribution over $x'_{1:2n}$, and we set ν to be the uniform distribution over $\mathscr{H}'_{2n}$.

Next, for any arbitrary sequence of points $X_n = (x_1, \ldots, x_n) \subset \mathsf{X}$ and a hypothesis $h \in \mathscr{H}$, we set $S(X_n, h) := \{(x_i, h(x_i))\}_{i=1}^n$. We will now lower bound the following expectation:

$$Q := \mathbb{E}_{X_n \sim \mu^n} \mathbb{E}_{h \sim \nu} \mathbb{E}_{x' \sim \mu} \mathbf{1}\{\mathcal{A}(S(X_n, h))(x') \neq h(x')\}.$$

For notational brevity, for what follows we will write:

$$\mathbb{E}_{X_n \sim \mu^n} \mathbb{E}_{h \sim \nu} \mathbb{E}_{x' \sim \mu} = \mathbb{E}_{\mu^n} \mathbb{E}_{\nu} \mathbb{E}_{\mu}.$$

Now for a given X_n and h, let $h(X_n) = (h(x_1), \ldots, h(x_n))$. Fix X_n, x' such that $x' \notin X_n$, and observe that:

$$\begin{aligned}
\mathbb{E}_{\nu} \mathbf{1}\{\mathcal{A}(S(X_n, h))(x') \neq h(x')\} &= \sum_{y \in \{\pm 1\}^n} \mathbb{E}_{\nu}[\mathbf{1}\{\mathcal{A}(S(X_n, h))(x') \neq h(x') \mid h(X_n) = y] \mathbb{P}_{\nu}\{h(X_n) = y\} \\
&= \sum_{y \in \{\pm 1\}^n} \frac{1}{2} \cdot \mathbb{P}_{\nu}\{h(X_n) = y\} = \frac{1}{2}.
\end{aligned}$$

This is the key idea here so let us be clear about why this equality is true. First, observe that if $h(X_n) = y$, then the quantity $\mathcal{A}(S(X_n, h))(x')$ is constant over the remaining randomness of h. Hence, since ν is uniform over $\mathscr{H}'_{2n}$, conditioned on $h(X_n) = y$, since $x' \notin X_n$, $h(x')$ takes on -1 and $+1$ both with equal probability. To make this more clear, consider the bijection between $\mathscr{H}'_{2n}$ and $\{\pm 1\}^{2n}$. If ζ is a random Rademacher random variable, the distribution of $h(x')$ conditioned on $h(X_n) = y$ for $x \notin X_n$ is equivalent to conditioning ζ on n coordinates and then considering the distribution on one of the remaining n free coordinates (i.e., it is uniform on $\{\pm 1\}$). This allows us to conclude, by swapping the order of expectations,

$$\begin{aligned}
Q &= \mathbb{E}_{\mu^n} \mathbb{E}_{\mu} \mathbb{E}_{\nu} \mathbf{1}\{\mathcal{A}(S(X_n, h))(x') \neq h(x')\} \\
&\geqslant \mathbb{E}_{\mu^n} \mathbb{E}_{\mu} \left[\mathbf{1}\{x' \notin X_n\} \cdot \mathbb{E}_{\nu} \mathbf{1}\{\mathcal{A}(S(X_n, h))(x') \neq h(x')\}\right] \\
&= \mathbb{E}_{\mu^n} \mathbb{E}_{\mu} \left[\mathbf{1}\{x' \notin X_n\} \cdot \frac{1}{2}\right] \\
&= \frac{1}{2} \cdot \mathbb{P}_{\mu^n, \mu}\{x' \notin X_n\} \\
&\geqslant \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}.
\end{aligned}$$

Above, the last inequality holds since μ is uniform over $x'_{1:2n}$. Hence, since there are at most n distinct elements in X_n, $\mathbb{P}_{\mu}\{x' \notin X_n\} \geqslant 1/2$. By swapping order of expectations again,

$$\mathbb{E}_{\nu} \mathbb{E}_{\mu^n} \mathbb{E}_{\mu} \mathbf{1}\{\mathcal{A}(S(X_n, h))(x') \neq h(x')\} \geqslant \frac{1}{4}.$$

By the probabilistic method (applied to the outermost expectation over ν only), there exists an $h \in \mathscr{H}'_{2n}$ such that

$$\mathbb{E}_{\mu^n}\mathbb{E}_{\mu}\mathbf{1}\{\mathcal{A}(S(X_n,h))(x') \neq h(x')\} \geqslant \frac{1}{4}.$$

To conclude the claim, we need a technical fact.

Proposition 1.52. *Suppose that $X \in [0,1]$ is a random variable with $\mathbb{E}[X] \geqslant \mu$. Then,*

$$\forall t \in (0,1),\ \mathbb{P}\{X > t\} \geqslant \frac{\mu - t}{1-t}.$$

Proof. Observe that $1 - X$ is non-negative, and hence we can apply Markov's inequality as follows:

$$1 - \mathbb{P}\{1 - t < X\} = \mathbb{P}\{1 - X \geqslant t\} \leqslant \frac{\mathbb{E}[1-X]}{t} \leqslant \frac{1-\mu}{t}.$$

Rearranging, this yields,

$$\mathbb{P}\{X > 1 - t\} \geqslant 1 - \frac{1-\mu}{t}.$$

Now set $t' = 1 - t$ to conclude:

$$\mathbb{P}\{X > t'\} \geqslant \frac{t - 1 + \mu}{t} = \frac{\mu - t'}{1 - t'}.$$

□

The claim now follows by applying the above proposition to the random variable

$$\psi(X_n) := \mathbb{E}_{\mu}\mathbf{1}\{\mathcal{A}(S(X_n,h))(x') \neq h(x')\}.$$

Since $\mathbb{E}_{\mu^n}[\psi(X_n)] \geqslant 1/4$, we conclude:

$$\mathbb{P}_{\mu^n}\{\psi(X_n) > 1/8\} \geqslant \frac{1/4 - 1/8}{1 - 1/8} = \frac{1}{7}.$$

Note that the distribution $\mathcal{D}$ is given by sampling $x \sim \mu$ and setting $y = h(x)$. □

The no free lunch theorem immediate gives us the following converse result for agnostic PAC learnability.

Theorem 1.53 (Infinite VC-dimension classes are not agnostically PAC learnable)**.** *Let X be a domain of infinite cardinality, and let $\mathscr{H}$ denote hypothesis class of functions mapping $\mathsf{X} \mapsto \mathsf{Y}$ with infinite VC-dimension (i.e., $\mathrm{VCdim}(\mathscr{H}) = \infty$). Then, $\mathscr{H}$ is not agnostically PAC learnable (cf. Definition 1.49).*

Proof. Suppose that $\mathscr{H}$ is agnostically PAC learnable via algorithm $\mathcal{A}$. Set $\varepsilon = 1/16$, $\delta = 1/14$, and $n = n(\varepsilon, \delta)$. Let $\mathcal{D}$ denote the distribution from Theorem 1.51 such that:

(a) there exists an $h \in \mathscr{H}$ satisfying $\mathbb{P}_{\mathcal{D}}\{h(x) \neq y\} = 0$,

(b) there exists an event $\mathcal{E}_1$ on $\mathcal{D}^n$ such that for $S_n \sim \mathcal{D}^n$, $\mathbb{P}_{\mathcal{D}^n}\{S_n \in \mathcal{E}_1\} \geqslant 1/7$, and

(c) on $\mathcal{E}_1$, $\mathbb{P}_{\mathcal{D}}\{\mathcal{A}(S_n)(x) \neq y\} \geqslant 1/8$.

However, since $\mathscr{H}$ is agnostically PAC learnable, there also exists another event $\mathcal{E}_2$ on $\mathcal{D}^n$ such that:

(a) for $S_n \sim \mathcal{D}^n$, $\mathbb{P}_{\mathcal{D}^n}\{S_n \in \mathcal{E}_2\} \geqslant 1 - \delta$,

(b) on $\mathcal{E}_2$, $\mathbb{P}_{\mathcal{D}}\{\mathcal{A}(S_n)(x) \neq y\} \leqslant \inf_{h \in \mathscr{H}} \mathbb{P}_{\mathcal{D}}\{h(x) \neq y) + \varepsilon = \varepsilon$.

Observe that for $S_n \in \mathcal{E}_1 \cap \mathcal{E}_2$, we have that:

$$\frac{1}{8} \leqslant \mathbb{P}_{\mathcal{D}}\{\mathcal{A}(S_n)(x) \neq y\} \leqslant \varepsilon = \frac{1}{16},$$

which is impossible. So we must have that $\mathbb{P}_{\mathcal{D}^n}\{\mathcal{E}_1 \cap \mathcal{E}_2\} = 0$. But, we have:

$$\begin{aligned}
\mathbb{P}_{\mathcal{D}^n}\{\mathcal{E}_1 \cap \mathcal{E}_2\} &= 1 - \mathbb{P}_{\mathcal{D}^n}\{\mathcal{E}_1^c \cup \mathcal{E}_2^c\} \\
&\geqslant 1 - \mathbb{P}_{\mathcal{D}^n}\{\mathcal{E}_1^c\} - \mathbb{P}_{\mathcal{D}^n}\{\mathcal{E}_2\} && \text{by a union bound} \\
&\geqslant 1 - (1 - 1/7) - \delta = 1/7 - 1/14 = 1/14 > 0.
\end{aligned}$$

□

Theorem 1.50 combined with Theorem 1.53 immediately yields the following characterization of when agnostic PAC learning is possible. The following corollary is a key foundational result in machine learning.

Corollary 1.54. *A hypothesis class $\mathscr{H}$ is agnostically PAC learnable if and only if* $\mathrm{VCdim}(\mathscr{H}) < \infty$.

1.4.13 Chaining and Dudley's inequality

Note: we will not have time to cover this section in this course.

Definition 1.55 (Covering number)**.** *Let (T, ρ) be a metric space. A collection of points $S = \{s_i\} \subset T$ is an* ε-covering *if for every $t \in T$ there exists an $s_i \in S$ such that $\rho(s_i, t) \leqslant \varepsilon$. The* covering number *of T at resolution ε, denoted $N(\varepsilon; T, \rho)$, is the minimum number of points needed to form an ε-cover of T.*

Definition 1.56 (Packing number)**.** *Let (T, ρ) be a metric space. A collection of points $S = \{s_i\} \subset T$ is an* ε-packing *if for any $s_i, s_j \in S$ with $s_i \neq s_j$, we have $\rho(s_i, s_j) > \epsilon$. The* packing number *of T at resolution ε, denoted $M(\varepsilon; T, \rho)$, is the maximum achievable size amongst all ε-packings of T.*

Covering and packing numbers are closely related.

Proposition 1.57. *Let (T, ρ) be a metric space. We have:*

$$M(2\varepsilon; T, \rho) \leqslant N(\varepsilon; T, \rho) \leqslant M(\varepsilon; T, \rho).$$

Proof. We will abbreviate the covering and packing numbers as $N(\varepsilon)$ and $M(\varepsilon)$, respectively. We first prove that $N(\varepsilon) \leqslant M(\varepsilon)$. Let $\{t_1, \ldots, t_M\}$ be a maximal ε-packing. We claim that $\{t_1, \ldots, t_M\}$ is also an ε-covering. To see this, suppose by contradiction there exists a $t \in T$ satisfying $\min_{i=1,\ldots,M} \rho(t, t_i) > \varepsilon$. Then, $\{t_1, \ldots, t_M, t\}$ is an ε-packing by definition with cardinality $M + 1$, a contradiction.

Now we show that $M(2\varepsilon) \leqslant N(\varepsilon)$. Let $\{t_1, \ldots, t_M\}$ be a maximal 2ε-packing and let $\{u_1, \ldots, u_N\}$ be a minimal ε-covering. Suppose that $M \geqslant N + 1$. Since $\{u_1, \ldots, u_N\}$ covers T and $M > N$, there must exists a $t_i \neq t_j$ and u_k such that $t_i, t_j \in B_\rho(u_k, \varepsilon)$ (here, $B_\rho(u, \varepsilon) = \{v \in T \mid \rho(u, v) \leqslant \varepsilon\}$.) By triangle inequality, $\rho(t_i, t_j) \leqslant \rho(t_i, u_k) + \rho(u_k, t_j) \leqslant 2\varepsilon$. But, since $\{t_i\}$ is a 2ε-packing, by definition we must have $\rho(t_i, t_j) > 2\varepsilon$, a contradiction, and hence $M < N + 1$ or equivalently $M \leqslant N$. □

Remark 1.58. The quantities $\log N(\varepsilon; T, \rho)$ and $\log M(\varepsilon; T, \rho)$ are often referred to as the *metric entropy* of T (at resolution ε).

Proposition 1.59 (Volume comparison inequalities)**.** *Let $T \subset \mathbb{R}^d$, and let $\|\cdot\|$ be a norm on $\mathbb{R}^d$ (not necessarily Euclidean). Let $B(x, r)$ denote the closed ball of radius r around x (measured using the $\|\cdot\|$ norm). Finally, let* $\mathrm{Vol}(\cdot)$ *denote the Lesbesgue measure on $\mathbb{R}^d$. For any $\varepsilon > 0$,*

$$\left(\frac{1}{\varepsilon}\right)^d \frac{\mathrm{Vol}(T)}{\mathrm{Vol}(B(0,1))} \leqslant N(\varepsilon; T, \|\cdot\|) \leqslant \frac{\mathrm{Vol}(T + B(0, \varepsilon/2))}{\mathrm{Vol}(B(0, \varepsilon/2))}.$$

Proof. We first prove the LHS inequality. Let $\{t_1, \dots, t_N\}$ be any ε-covering of T. By the definition of an ε covering, $T \subseteq \cup_{i=1}^N B(t_i, \varepsilon)$, and hence,

$$\mathrm{Vol}(T) \overset{(a)}{\leqslant} \mathrm{Vol}(\cup_{i=1}^N B(t_i, \varepsilon)) \overset{(b)}{\leqslant} \sum_{i=1}^N \mathrm{Vol}(B(t_i, \varepsilon)) \overset{(c)}{=} \sum_{i=1}^N \varepsilon^d \mathrm{Vol}(B(0,1)) = \varepsilon^d N \cdot \mathrm{Vol}(B(0,1)).$$

Above, (a) follows from monotonicity of measure, (b) follows from sub-additivity of measure, and (c) follows from the translation invariance and scaling properties of the Lebesgue measure. Rearranging the above inequality yields $N \geqslant \varepsilon^{-d} \mathrm{Vol}(T)/\mathrm{Vol}(B(0,1))$, and taking the infimum over ε-covers yields the LHS inequality.

We now proceed to the RHS inequality. Let $\{t_1, \dots, t_M\}$ be a maximal ε-packing of T. By construction,

$$\mathrm{Vol}(\cup_{i=1}^M B(t_i, \varepsilon/2)) = \sum_{i=1}^M \mathrm{Vol}(B(t_i, \varepsilon/2)) = M \cdot \mathrm{Vol}(B(0, \varepsilon/2)),$$

since the balls $B(t_i, \varepsilon/2)$ are disjoint. On the other hand, we clearly have $\cup_{i=1}^M B(t_i, \varepsilon/2) \subseteq T + B(0, \varepsilon/2)$, and hence by monotonicity of measure,

$$\mathrm{Vol}(\cup_{i=1}^M B(t_i, \varepsilon/2)) \leqslant \mathrm{Vol}(T + B(0, \varepsilon/2)).$$

Combining these two inequalities yields $M \leqslant \mathrm{Vol}(T + B(0, \varepsilon/2))/\mathrm{Vol}(B(0, \varepsilon/2))$. The claim follows by using the inequality $N(\varepsilon; T, \|\cdot\|) \leqslant M(\varepsilon; T, \|\cdot\|)$ from Proposition 1.57. □

Corollary 1.60. *Let $\|\cdot\|$ be any norm on $\mathbb{R}^d$. For any $\varepsilon > 0$,*

$$\left(\frac{1}{\varepsilon}\right)^d \leqslant N(\varepsilon; B(0,1), \|\cdot\|) \leqslant \left(1 + \frac{2}{\varepsilon}\right)^d.$$

Proof. The LHS inequality follows immediately from Proposition 1.59. For the RHS inequality, we simply observe that $B(0,1) + B(0, \varepsilon/2) \subseteq B(0, 1 + \varepsilon/2)$, in which case:

$$\mathrm{Vol}(B(0,1) + B(0, \varepsilon/2)) \leqslant (1 + \varepsilon/2)^d \mathrm{Vol}(B(0,1)).$$

The RHS inequality now follows again from Proposition 1.59. □

For a function class $\mathscr{F}$ mapping $\mathsf{X} \mapsto \mathbb{R}$ and a realization of data points $\{x_i\}_{i=1}^n$, we define the empirical $L_2(\mathsf{P}_n)$ (semi-)norm as:

$$\|f\|_{L_2(\mathsf{P}_n)}^2 := \frac{1}{n} \sum_{i=1}^n f(x_i)^2.$$

Let us make a first attempt to bound the Rademacher complexity by covering numbers. Recall Massart's lemma (Proposition 1.44), which gives a simple bound for finite function classes. Thus, we can use the idea of a covering to discretize a continuous function class, and pay a little extra for the discretization error (which is controlled by the scale of the covering).

Proposition 1.61 (One step discretization)**.** *The following holds for every realization $x_1, \dots, x_n \in \mathsf{X}$:*

$$\mathbb{E}_\varepsilon \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^n \varepsilon_i f(x_i) \leqslant \inf_{\alpha > 0} \left\{ \alpha + B_n \sqrt{\frac{2 \log N(\alpha; \mathscr{F}, L_2(\mathsf{P}_n))}{n}} \right\}, \quad B_n = \sup_{f \in \mathscr{F}} \|f\|_{L_2(\mathsf{P}_n)}.$$

Proof. We start with the projection set $Q = \{(f(x_1), \dots, f(x_n)) \in \mathbb{R}^n \mid f \in \mathscr{F}\}$. Fix any $\alpha > 0$ and let C denote a minimal α-covering of Q in the weighted $x \mapsto n^{-1/2}\|x\|$ norm. For any $q \in Q$ let $\varphi(q)$ denote the element in C of minimal distance, such that $n^{-1/2}\|q - \varphi(q)\| \leqslant \alpha$. We can now proceed by:

$$\mathbb{E} \sup_{q \in Q} n^{-1} \langle \varepsilon, q \rangle = \mathbb{E} \sup_{q \in Q} n^{-1} \langle \varepsilon, \varphi(q) + q - \varphi(q) \rangle$$

$$\leqslant \mathbb{E} \sup_{q \in Q} n^{-1} \langle \varepsilon, \varphi(q) \rangle + \mathbb{E} \sup_{q \in Q} n^{-1} \langle \varepsilon, q - \varphi(q) \rangle.$$

Now, we first observe that:

$$n^{-1} \langle \varepsilon, q - \varphi(q) \rangle \leqslant n^{-1} \|\varepsilon\| \|q - \varphi(q)\| = n^{-1/2} \|q - \varphi(q)\| \leqslant \alpha,$$

and hence the second term is bounded by α. As for the first term, we apply Massart's lemma (Proposition 1.44), noting that $n^{-1/2}\|q\| \leqslant B_n$, and therefore,

$$\mathbb{E} \sup_{q \in Q} n^{-1} \langle \varepsilon, \varphi(q) \rangle \leqslant B_n \sqrt{\frac{2 \log |C|}{n}} \leqslant B_n \sqrt{\frac{2 \log N(\alpha; \mathscr{F}, L_2(\mathsf{P}_n))}{n}}.$$

Since $\alpha > 0$ was arbitrary, we can optimize the bound over α, from which the claim follows. □

While the one step discretization argument is certainly a step in the right direction in terms of using covering numbers to bound Rademacher complexity, as we will see later it can be un-necessarily loose. The issue is that the argument is quite conservative by only choosing to discretize at one resolution. By repeatedly discretizing at finer and finer resolutions, we get a sharper result known as Dudley's inequality.

Lemma 1.62 (Dudley's entropy inequality)**.** *The following holds for every realization $x_1, \ldots, x_n \in \mathsf{X}$:*

$$\mathbb{E}_\varepsilon \sup_{f \in \mathscr{F}} n^{-1} \sum_{i=1}^n \varepsilon_i f(x_i) \leqslant 12 \int_0^{B_n} \sqrt{\frac{\log N(\varepsilon; \mathscr{F}, L_2(\mathsf{P}_n))}{n}} \, \mathrm{d}\varepsilon, \quad B_n = \sup_{f \in \mathscr{F}} \|f\|_{L_2(\mathsf{P}_n)}.$$

Proof. Put $\varepsilon_0 := B_n$ and for $i = 1, 2, \ldots$, define a sequence of decreasing resolutions $\varepsilon_i = \varepsilon_0 2^{-i}$. As before, let the projection set Q be

$$Q = \{(f(x_1), \ldots, f(x_n)) \in \mathbb{R}^n \mid f \in \mathscr{F}\}.$$

With the definition of ε_0, observe that for any $q \in Q$,

$$n^{-1/2}\|q\| = \sqrt{\frac{1}{n} \sum_{i=1}^n f(x_i)^2} \leqslant \varepsilon_0.$$

For any $i = 1, 2, \ldots$, let C_i denote a minimal ε_i-cover of Q in the weighted ℓ_2 norm $n^{-1/2}\|\cdot\|$. Furthermore, for any $q \in Q$, let $\varphi_i(q)$ denote an element in C_i satisfying $n^{-1/2}\|q - \varphi_i(q)\| \leqslant \varepsilon_i$. Fix any $q \in Q$, and note that for any finite N,

$$\varphi_1(q) + \sum_{i=1}^N (\varphi_{i+1}(q) - \varphi_i(q)) = \varphi_{N+1}(q).$$

Hence, since $\lim_{i \to \infty} \varphi_i(q) = q$,

$$q = \varphi_1(q) + \sum_{i=1}^\infty (\varphi_{i+1}(q) - \varphi_i(q)).$$

Therefore,

$$\begin{aligned}
\mathbb{E} \sup_{q \in Q} n^{-1} \langle \varepsilon, q \rangle &= \mathbb{E} \sup_{q \in Q} n^{-1} \left\langle \varepsilon, \varphi_1(q) + \sum_{i=1}^\infty (\varphi_{i+1}(q) - \varphi_i(q)) \right\rangle \\
&\leqslant \mathbb{E} \sup_{q \in Q} n^{-1} \langle \varepsilon, \varphi_1(q) \rangle + \sum_{i=1}^\infty \mathbb{E} \sup_{q \in Q} n^{-1} \langle \varepsilon, \varphi_{i+1}(q) - \varphi_i(q) \rangle.
\end{aligned}$$

Let us control the second term first. First, observe that for any $q \in Q$,

$$n^{-1/2}\|\varphi_{i+1}(q) - \varphi_i(q)\| \leqslant n^{-1/2}\|q - \varphi_{i+1}(q)\| + n^{-1/2}\|q - \varphi_i(q)\| \leqslant \varepsilon_{i+1} + \varepsilon_i = 3\varepsilon_{i+1}.$$

On the other hand, the number of unique vectors $\{\varphi_{i+1}(q) - \varphi_i(q) \mid q \in Q\}$ is clearly upper bounded by $|C_{i+1}| \times |C_i|$. Hence by Massart's lemma (Proposition 1.44),

$$\mathbb{E}\sup_{q\in Q} n^{-1}\langle \varepsilon, \varphi_{i+1}(q) - \varphi_i(q)\rangle \leqslant 3\varepsilon_{i+1}\sqrt{\frac{2\log|C_{i+1}||C_i|}{n}} \overset{(a)}{\leqslant} 6\varepsilon_{i+1}\sqrt{\frac{\log|C_{i+1}|}{n}},$$

where (a) follows since $|C_i| \leqslant |C_{i+1}|$.

Now turning to the first term, we observe that $n^{-1/2}\|\varphi_1(q)\| \leqslant \varepsilon_0$, and hence again by Massart's lemma,

$$\mathbb{E}\sup_{q\in Q} n^{-1}\langle \varepsilon, \varphi_1(q)\rangle \leqslant \varepsilon_0\sqrt{\frac{2\log|C_1|}{n}} = 2\varepsilon_1\sqrt{\frac{2\log|C_1|}{n}}.$$

Hence combining both inequalities,

$$\mathbb{E}\sup_{q\in Q} n^{-1}\langle \varepsilon, q\rangle \leqslant 6\sum_{i=1}^{\infty}\varepsilon_i\sqrt{\frac{\log|C_i|}{n}} = 6\sum_{i=1}^{\infty}\varepsilon_i\sqrt{\frac{\log N(\varepsilon_i;\mathscr{F},L_2(\mathsf{P}_n))}{n}}.$$

To finish the proof, we observe that since $\varepsilon \mapsto N(\varepsilon;\mathscr{F},L_2(\mathsf{P}_n))$ is a monotonically decreasing function of ε, we have the following inequality:

$$(\varepsilon_i - \varepsilon_{i+1})\sqrt{\log N(\varepsilon_i;\mathscr{F},L_2(\mathsf{P}_n))} \leqslant \int_{\varepsilon_{i+1}}^{\varepsilon_i}\sqrt{\log N(\varepsilon;\mathscr{F},L_2(\mathsf{P}_n))}\,\mathrm{d}\varepsilon.$$

Since $\varepsilon_i = 2(\varepsilon_i - \varepsilon_{i+1})$, we conclude that

$$\begin{aligned}
6\sum_{i=1}^{\infty}\varepsilon_i\sqrt{\log N(\varepsilon_i;\mathscr{F},L_2(\mathsf{P}_n))} &\leqslant 12\sum_{i=1}^{\infty}\int_{\varepsilon_{i+1}}^{\varepsilon_i}\sqrt{\log N(\varepsilon;\mathscr{F},L_2(\mathsf{P}_n))}\,\mathrm{d}\varepsilon \\
&= 12\int_0^{\varepsilon_1}\sqrt{\log N(\varepsilon;\mathscr{F},L_2(\mathsf{P}_n))}\,\mathrm{d}\varepsilon \\
&\leqslant 12\int_0^{\varepsilon_0}\sqrt{\log N(\varepsilon;\mathscr{F},L_2(\mathsf{P}_n))}\,\mathrm{d}\varepsilon.
\end{aligned}$$

□

Remark 1.63. Dudley's inequality (Lemma 1.62) is stated conditionally on the data $\{x_i\}_{i=1}^n$. Taking another expectation over the data, we obtain:

$$\mathcal{R}_n(\mathscr{F}) \leqslant 12\mathbb{E}\int_0^{B_n}\sqrt{\frac{\log N(\varepsilon;\mathscr{F},L_2(\mathsf{P}_n))}{n}}\,\mathrm{d}\varepsilon, \quad B_n = \sup_{f\in\mathscr{F}}\|f\|_{L_2(\mathsf{P}_n)}.$$

While Dudley's inequality is an improvement over the one step discretization argument in that it considers covering numbers at multiple resolutions, the version stated in Lemma 1.62 suffers from one key drawback, which is that in situations where $\log N(\varepsilon;\mathscr{F},L_2(\mathsf{P}_n))$ is not integrable near the origin, the bound is vacuous. On the other hand the one step argument Proposition 1.61 does not suffer from this issue. We can, however, combine both bounds and obtain the best of both—multiscale covering numbers and truncating the lower limit of the integral before $\varepsilon \to 0$.

Exercise 1.64. Modify the proof of Lemma 1.62 to prove the following refined Dudley's inequality:

$$\mathbb{E}_\varepsilon \sup_{f\in\mathscr{F}} n^{-1}\sum_{i=1}^n \varepsilon_i f(x_i) \leqslant \inf_{\alpha>0}\left\{4\alpha + 12\int_\alpha^{B_n}\sqrt{\frac{\log N(\varepsilon;\mathscr{F},L_2(\mathsf{P}_n))}{n}}\,\mathrm{d}\varepsilon\right\}.$$

Let us now apply Dudley's inequality to study linear hypothesis classes, as we did in Section 1.4.9. Specifically, let us consider $\mathscr{F} = \{x \mapsto \langle x, w\rangle \mid w \in \mathbb{R}^d, \|w\| \leqslant 1\}$. We need to estimate the $L_2(\mathsf{P}_n)$ covering number of $\mathscr{F}$. Let $X \in \mathbb{R}^{n\times d}$ denote the data matrix with row i containing example x_i. Now observe that for any $f_u, f_v \in \mathscr{F}$,

$$\|f_u - f_v\|^2_{L_2(\mathsf{P}_n)} = \frac{1}{n}\sum_{i=1}^n \langle x_i, u-v\rangle^2 = \frac{1}{n}(u-v)^\mathsf{T} X^\mathsf{T} X(u-v) \leqslant \frac{\lambda_{\max}(X^\mathsf{T}X)}{n}\|u-v\|^2.$$

Thus, we immediately see that, with $B_n := \|X/\sqrt{n}\|_{\mathrm{op}}$,

$$N(\varepsilon;\mathscr{F},L_2(\mathsf{P}_n)) \leqslant N(\varepsilon/B_n, B(0,1), \|\cdot\|) \leqslant \left(1+\frac{2B_n}{\varepsilon}\right)^d,$$

where the last estimate follows from Corollary 1.60. Now, letting B be an almost sure upper bound on B_n, we compute:

$$\begin{aligned}
\int_0^B \sqrt{\log N(\varepsilon;\mathscr{F},L_2(\mathsf{P}_n))}\,\mathrm{d}\varepsilon &\leqslant \sqrt{d}\int_0^B \sqrt{\log(1+2B/\varepsilon)}\,\mathrm{d}\varepsilon \\
&= B\sqrt{d}\int_0^1 \sqrt{\log(1+2/\varepsilon)}\,\mathrm{d}\varepsilon && \text{change of variables } \varepsilon \leftarrow \varepsilon/B \\
&\leqslant 2B\sqrt{d} && \text{using Mathematica.}
\end{aligned}$$

Hence, from Dudley's inequality,

$$\mathcal{R}_n(\mathscr{F}) \leqslant 24B\sqrt{\frac{d}{n}}.$$

Now, as a point of comparison, observe that the one-step discretization bound Proposition 1.61 yields the looser bound $\mathcal{R}_n(\mathscr{F}) \lesssim B\sqrt{\frac{d\log(n/d)}{n}}$ by choosing $\alpha \asymp \sqrt{d/n}$. Hence, the Dudley argument allows us to shave off a log factor in the bound for this example.

Let us now compare the bound from Dudley's inequality to the direct calculations made in Section 1.4.9. Recall that we computed that if x_i are i.i.d. from $N(0,I)$, then $\mathcal{R}_n(\mathscr{F}) \leqslant \sqrt{d/n}$. Immediately we see that the Dudley bound above does not apply to case, since there is no finite almost sure bound on B. However, let us suppose that instead the x_i's are i.i.d. where each entry is an independent Rademacher random variable. Then, the first and second moments $\mathbb{E}\|x\|$ and $\mathbb{E}\|x\|^2$ match the Gaussian distribution order wise, but we have that $B = \sqrt{d}$ (check!). Thus, we actually get a looser result $\mathcal{R}_n(\mathscr{F}) \lesssim d/\sqrt{n}$, which is not sharp. However, this can be fixed, as the contents of the following exercise:

Exercise 1.65. Suppose that the random vector x_i's are i.i.d. where each entry is an independent Rademacher random variable. Let $\mathscr{F} = \{x \mapsto \langle x, w\rangle \mid w \in \mathbb{R}^d, \|w\| \leqslant 1\}$. Show using Remark 1.63 that when $n \geqslant d$, there exists a universal constant c such that:

$$\mathcal{R}_n(\mathscr{F}) \leqslant c\sqrt{\frac{d}{n}}.$$

Hint: use the following random matrix deviation bound (which you may use without proof, although at this point you actually have all the tools to prove this inequality!). Let $X \in \mathbb{R}^{n\times d}$ be the data matrix with row i corresponding to x_i. There exists universal constants c_0, c_1 such that:

$$\forall t > 0,\ \mathbb{P}\left\{\|X\|_{\mathrm{op}} \geqslant \sqrt{n} + c_0\sqrt{d} + t\right\} \leqslant \exp(-c_1 t^2).$$

The real advantage of Dudley's inequality, however, is to utilize it beyond linear hypothesis classes.

Exercise 1.66. Let $\mathscr{F} = \{x \mapsto f_\theta(x) \mid \theta \in \Theta \subset \mathbb{R}^d,\ \|\theta\| \leqslant B\}$. Suppose furthermore that the following conditions hold:

$$\forall x \in \mathsf{X},\ \theta, \theta_1, \theta_2 \in \Theta,\ |f_\theta(x)| \leqslant 1,\ |f_{\theta_1}(x) - f_{\theta_2}(x)| \leqslant L\|\theta_1 - \theta_2\|.$$

Show that there exists a universal constant c such that:

$$\mathcal{R}_n(\mathscr{F}) \leqslant c\sqrt{\frac{d\log(BL)}{n}}.$$

Exercise 1.67. Let $\mathscr{F}$ denote the following 1-Lipschitz function class:

$$\mathscr{F} = \{f : [-1,1]^d \mapsto [-1,1] \mid \forall x, y \in [-1,1]^d,\ |f(x) - f(y)| \leqslant \|x - y\|_\infty\}.$$

Suppose that $d > 2$. There exists an constant c such that:

$$\mathcal{R}_n(\mathscr{F}) \leqslant \frac{c}{n^{1/d}}.$$

Hint: Use the fact that $\log N(\varepsilon; \mathscr{F}, \|\cdot\|_\infty) \leqslant (c_0/\varepsilon)^d$ for some universal constant c_0. Here, $\|f\|_\infty = \sup_{x\in[-1,1]^d} |f(x)|$ is the supremum norm over $[-1,1]^d$.

1.5 Stochastic gradient methods

In our study of ERM (cf. Section 1.4), we assumed that our estimator was the global minimizer of the empirical risk:

$$\hat{f}_{\mathrm{erm}} \in \arg\min_{f\in\mathscr{F}} \hat{L}_n[f] = \left\{\frac{1}{n}\sum_{i=1}^n \ell(f(x_i), y_i)\right\}.$$

In this section, we revisit this assumption. In particular, we now consider the algorithmic aspects of actually computing an approximate empirical risk minimizer. To set the stage, we impose some additional structure on our function class. In particular, we assume that the function class is parameterized by a set of parameters $\theta \in \Theta \subseteq \mathbb{R}^p$, i.e., $\mathcal{F} = \{f_\theta \mid \theta \in \Theta\}$. With this parameterization, the training loss because a function mapping $\Theta \mapsto \mathbb{R}$, as in $\hat{L}_n[\theta] = \frac{1}{n}\sum_{i=1}^n \ell(f_\theta(x_i), y_i)$. Furthermore, we will assume that both $\theta \mapsto f_\theta$ and $y \mapsto \ell(y, \hat{y})$ are differentiable, and hence $\hat{L}_n[\theta]$ is differentiable.

1.5.1 Gradient descent

Gradient descent in its most basic form is the following algorithm. First, choose a step size $\eta > 0$ and initial iterate θ_0. Then, iterate the following until convergence:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \hat{L}_n[\theta_t], \quad t \in \mathbb{N}_+. \tag{1.43}$$

There are many ways to measure convergence, of which here are a few:

(i) Parameter convergence: $\|\theta_{t+1} - \theta_t\| \leqslant \varepsilon$,

(ii) Function value convergence: $|\hat{L}_n[\theta_{t+1}] - \hat{L}_n[\theta_t]| \leqslant \varepsilon$,

(iii) Gradient norm convergence: $\|\nabla_\theta \hat{L}_n[\theta_t]\| \leqslant \varepsilon$.

The motivation behind gradient descent is that for a small enough step size η, the update $\theta_t - \eta \nabla_\theta \hat{L}_n[\theta_t]$ is guaranteed to *decrease* the function value $\hat{L}_n[\theta_t]$, unless θ_t is already optimal. To see this, we take a Taylor expansion of $\hat{L}_n[\theta]$:

$$\begin{aligned}\hat{L}_n[\theta_{t+1}] &= \hat{L}_n[\theta_t] + \langle \nabla_\theta \hat{L}_n[\theta_t], \theta_{t+1} - \theta_t \rangle + O(\|\theta_{t+1} - \theta_t\|^2) \\ &= \hat{L}_n[\theta_t] - \eta \|\nabla_\theta \hat{L}_n[\theta_t]\|^2 + O(\eta^2).\end{aligned}$$

Supposing that θ_t is not optimal for $\hat{L}_n$, then the gradient $\nabla_\theta \hat{L}_n[\theta_t] \neq 0$. Hence, for η small enough, this shows that $\hat{L}_n[\theta_{t+1}] < \hat{L}_n[\theta_t]$. In other words, $-\nabla_\theta \hat{L}_n[\theta_t]$ is a *descent direction*.

The issue with gradient descent in practice is that it takes $O(n)$ computation to compute one gradient $\nabla_\theta \hat{L}_n[\theta]$, since $\hat{L}_n[\theta]$ is composed of the sum of n training examples. Hence, for the large datasets often found in machine learning, this is not a practical algorithm. We now turn to stochastic gradient descent to address this issue.

1.5.2 Stochastic gradient descent

The additive structure of the empirical risk provides a solution to the computational complexity of gradient descent, known as *stochastic gradient descent* (SGD). To describe SGD, let indices $\{i_t\}_{t \in \mathbb{N}}$ be constructed so that each $i_t \sim \mathrm{Unif}(\{1, \dots, n\})$ is sampled independently across time, and let $\{\eta_t\}_{t \in \mathbb{N}}$ denote a sequence of step sizes. Fix an initial θ_0, and define the following update rule:

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta \ell(f_{\theta_t}(x_{i_t}), y_{i_t}), \quad t \in \mathbb{N}. \tag{1.44}$$

The key to SGD update rule is the following property. Let $\mathcal{H}_t = \{i_1, \dots, i_t\}$ denote the history of indices up to and including time t, so that θ_t is $\mathcal{H}_{t-1}$-measurable (that is, θ_t is fully determined by the randomness in $\mathcal{H}_{t-1}$). Then by the tower property of expectation:

$$\begin{aligned}\mathbb{E}[\nabla_\theta \ell(f_{\theta_t}(x_{i_t}), y_{i_t})] &= \mathbb{E}[\mathbb{E}[\nabla_\theta \ell(f_{\theta_t}(x_{i_t}), y_{i_t}) \mid \mathcal{H}_{t-1}]] \\ &= \mathbb{E}\left[\frac{1}{n} \sum_{i=1}^n \nabla_\theta \ell(f_{\theta_t}(x_i), y_i)\right] \\ &= \mathbb{E}[\nabla_\theta \hat{L}_n[\theta_t]].\end{aligned}$$

Hence stochastic gradient update rule follows the true gradient in expectation. On the other hand, the computational cost of each iteration of (1.44) does *not* scale with the number of training datapoints n, and is hence much more practical than (1.43) for large training datasets.

Step sizes: How should the step sizes be chosen for SGD? When the functions $\theta \mapsto \ell(f_\theta(x), y)$ are convex, then the classic theory prescribes learning rates of the form $\eta_t \propto 1/\sqrt{t}$ or $\eta_t \propto 1/t$, depending on type of convexity present. However, for the non-convex functions frequently encountered in machine learning, there exists many heuristic selections of step sizes. Covering all these choices is out of the scope of this course. For those interested, some keywords to search for include: epoch decay, cyclic learning rates, learning rate warmup, and cosine decay.

Epochs and minibatches: In practice, the random index model discussed is usually replaced with epochs and minibatching. Specifically, given n data points, we proceed in epochs, where within each epoch every data point is processed exactly once. The way to ensure this is to use a *random permutation* of the indices for each epoch. Furthermore, in order to reduce the variance of each stochastic gradient, the stochastic gradient over a minibatch of data is used.

1.5.3 Convergence guarantees

We now develop a simple yet general convergence theory for gradient methods. We first discuss what a reasonable notion of convergence for general non-convex settings is. In general, for a non-convex loss surface $\hat{L}_n[\theta]$, we cannot expect to have strong convergence guarantees to global empirical risk minimizers of $\hat{L}_n[\theta]$, as it is possible to cast NP-hard problems in the language of non-convex ERM. However, what we can expect are weaker results regarding the norm of the gradient. In particular, we will look at the iteration complexity of computing a point θ_T such that $\|\nabla_\theta \hat{L}_n[\theta_T]\|^2 \leqslant \varepsilon$. Of course, for non-convex functions we know that small gradient norm does not imply optimality. Nevertheless, it is still a necessary condition to hold for optimality.

We start our analysis for gradient descent. For our analysis, we will impose two mild conditions: (a) the training loss is non-negative, and (b) the training loss $\hat{L}_n[\theta]$ is L-smooth (cf. Definition C.4):

$$\hat{L}_n[v] \leqslant \hat{L}_n[u] + \langle \nabla_\theta \hat{L}_n[u], v - u\rangle + \frac{L}{2}\|u - v\|^2, \quad \forall u, v \in \mathbb{R}^p.$$

With this assumption, we have the following result.

Proposition 1.68. *Suppose that $\hat{L}_n$ is non-negative and L-smooth. For any $0 < \eta \leqslant 1/L$ and $T \in \mathbb{N}_+$, the gradient descent update (1.43) satisfies:*

$$\min_{t\in\{0,\dots,T-1\}} \|\nabla_\theta \hat{L}_n[\theta_t]\|^2 \leqslant \frac{2\hat{L}_n[\theta_0]}{\eta T}.$$

Proof. By the L-smoothness of $\hat{L}_n$, we have for every $t \in \mathbb{N}$:

$$\begin{aligned}
\hat{L}_n[\theta_{t+1}] &\leqslant \hat{L}_n[\theta_t] - \eta\|\nabla_\theta \hat{L}_n[\theta_t]\|^2 + \frac{\eta^2 L}{2}\|\nabla_\theta \hat{L}_n[\theta_t]\|^2 \\
&= \hat{L}_n[\theta_t] - \eta\left(1 - \frac{\eta L}{2}\right)\|\nabla_\theta \hat{L}_n[\theta_t]\|^2 \\
&\leqslant \hat{L}_n[\theta_t] - \frac{\eta}{2}\|\nabla_\theta \hat{L}_n[\theta_t]\|^2 && \text{since } \eta \leqslant 1/L.
\end{aligned}$$

Unrolling this recursion down to $t = 0$,

$$\begin{aligned}
\hat{L}_n[\theta_T] \leqslant \hat{L}_n[\theta_0] - \frac{\eta}{2}\sum_{t=0}^{T-1}\|\nabla_\theta \hat{L}_n[\theta_t]\|^2 &\implies \frac{\eta}{2}\sum_{t=0}^{T-1}\|\nabla_\theta \hat{L}_n[\theta_t]\|^2 \leqslant \hat{L}_n[\theta_0] - \hat{L}_n[\theta_T] \\
&\implies \min_{t\in\{0,\dots,T-1\}}\|\nabla_\theta \hat{L}_n[\theta_t]\|^2 \leqslant \frac{2(\hat{L}_n[\theta_0] - \hat{L}_n[\theta_T])}{\eta T} \\
&\implies \min_{t\in\{0,\dots,T-1\}}\|\nabla_\theta \hat{L}_n[\theta_t]\|^2 \leqslant \frac{2\hat{L}_n[\theta_0]}{\eta T} && \text{since } \hat{L}_n \geqslant 0.
\end{aligned}$$

□

Proposition 1.68 has several key takeaways. First, we see that a *constant step size* suffices for gradient convergence. Furthermore, the rate of convergence is that:

$$T \geqslant \frac{2\hat{L}_n[\theta_0]}{\eta\varepsilon} \implies \min_{t\in\{0,\dots,T-1\}}\|\nabla_\theta \hat{L}_n[\theta_t]\|^2 \leqslant \varepsilon.$$

We now turn to studying stochastic gradient descent.

Proposition 1.69. *Suppose that $\hat{L}_n$ is non-negative and L-smooth. Suppose furthermore that:*

$$\sup_{\theta \in \mathbb{R}^p} \operatorname{tr} \operatorname{Cov}_\iota(\nabla_\theta \ell(f_\theta(x_\iota), y_\iota)) \leqslant B^2, \quad \iota \sim \operatorname{Unif}(\{1, \dots, n\}).$$

Finally, suppose that the step size sequence $\{\eta_t\}_{t \in \mathbb{N}}$ satisfies $\eta_t \leqslant 1/L$ for all $t \in \mathbb{N}$. Then, for all $T \in \mathbb{N}_+$:

$$\sum_{t=0}^{T-1} \eta_t \mathbb{E} \|\nabla_\theta \hat{L}_n[\theta_t]\|^2 \leqslant 2\hat{L}_n[\theta_0] + LB^2 \sum_{t=0}^{T-1} \eta_t^2.$$

Proof. Let $g_t := \nabla_\theta \ell(f_{\theta_t}(x_{i_t}), y_{i_t})$ denote the stochastic gradient. We have that $\mathbb{E}[g_t \mid \mathcal{H}_{t-1}] = \nabla_\theta \hat{L}_n[\theta_t]$. Furthermore, the variance assumption ensures that:

$$\mathbb{E}[\|g_t\|^2 \mid \mathcal{H}_{t-1}] - \|\nabla_\theta \hat{L}_n[\theta_t]\|^2 \leqslant B^2 \text{ a.s.}$$

By the L-smoothness of $\hat{L}_n$, we have for every $t \in \mathbb{N}$:

$$\begin{aligned}
\mathbb{E}[\hat{L}_n[\theta_{t+1}]] &\leqslant \mathbb{E}\left[\hat{L}_n[\theta_t] - \eta_t \langle \nabla_\theta \hat{L}_n[\theta_t], g_t \rangle + \frac{L\eta_t^2}{2} \|g_t\|^2\right] \\
&= \mathbb{E}\left[\hat{L}_n[\theta_t] - \eta_t \langle \nabla_\theta \hat{L}_n[\theta_t], \mathbb{E}[g_t \mid \mathcal{H}_{t-1}] \rangle + \frac{L\eta_t^2}{2} \mathbb{E}[\|g_t\|^2 \mid \mathcal{H}_{t-1}]\right] \\
&\leqslant \mathbb{E}\left[\hat{L}_n[\theta_t] - \eta_t \left(1 - \frac{\eta_t L}{2}\right) \|\nabla_\theta \hat{L}_n[\theta_t]\|^2 + \frac{LB^2\eta_t^2}{2}\right] && \text{variance assumption} \\
&\leqslant \mathbb{E}\left[\hat{L}_n[\theta_t] - \frac{\eta_t}{2} \|\nabla_\theta \hat{L}_n[\theta_t]\|^2 + \frac{LB^2\eta_t^2}{2}\right] && \text{since } \eta_t \leqslant 1/L.
\end{aligned}$$

Unrolling this recursion down to $t = 0$,

$$\mathbb{E}[\hat{L}_n[\theta_T]] \leqslant \hat{L}_n[\theta_0] - \frac{1}{2} \sum_{t=0}^{T-1} \eta_t \mathbb{E} \|\nabla_\theta \hat{L}_n[\theta_t]\|^2 + \frac{LB^2}{2} \sum_{t=0}^{T-1} \eta_t^2.$$

Rearranging and using the non-negativity of $\hat{L}_n$ yields the desired claim. □

Let us specify a sequence of step sizes for Proposition 1.69. Let $\eta_t = \frac{1}{L\sqrt{t+1}}$ for $t \in \mathbb{N}$. Then, since $t \mapsto 1/(t+1)$ is a decreasing function, we can bound:

$$\sum_{t=0}^{T-1} \eta_t^2 = \frac{1}{L^2} \sum_{t=0}^{T-1} \frac{1}{t+1} = \frac{1}{L^2}\left(1 + \sum_{t=1}^{T-1} \frac{1}{t+1}\right) \leqslant \frac{1}{L^2}\left(1 + \int_0^{T-1} \frac{1}{1+x} \, \mathrm{d}x\right) = \frac{\log T + 1}{L^2}.$$

On the other hand:

$$\sum_{t=0}^{T-1} \eta_t \mathbb{E} \|\nabla_\theta \hat{L}_n[\theta_t]\|^2 \geqslant \eta_{T-1} T \cdot \min_{t \in \{0, \dots, T-1\}} \mathbb{E} \|\nabla_\theta \hat{L}_n[\theta_t]\|^2 = \sqrt{T}/L \cdot \min_{t \in \{0, \dots, T-1\}} \mathbb{E} \|\nabla_\theta \hat{L}_n[\theta_t]\|^2.$$

Combining these bounds with Proposition 1.69,

$$\min_{t \in \{0, \dots, T-1\}} \mathbb{E} \|\nabla_\theta \hat{L}_n[\theta_t]\|^2 \leqslant \frac{2L \cdot \hat{L}_n[\theta_0] + B^2(\log T + 1)}{\sqrt{T}}.$$

Compared to Proposition 1.68, we see some key differences. First, the step size η_t needs to decay for a non-trivial rate, due to the appearance of the $\sum_{t=0}^{T-1} \eta_t^2$ term. This is opposed to the constant step size for gradient descent. Next we see that the convergence rate is $O(\log T/\sqrt{T})$ for SGD versus $O(1/T)$ for GD; the slower $\tilde{O}(1/\sqrt{T})$ rate is due to the decaying step size. Finally, the guarantee from Proposition 1.69 is only in expectation (over the randomness of the SGD indices) as opposed to the deterministic guarantee from Proposition 1.68. We note that with more sophisticated martingale methods, high probability bounds for SGD can also be obtained.

1.5.4 Overparmaterization

So far the convergence results in Section 1.5.3 show that the gradient norm of $\hat{L}_n[\theta]$ is small. As discussed, for non-convex functions this does not imply that $\hat{L}_n[\theta]$ is a global, nor even local, minima. Why then, does SGD work so well in modern deep learning pipelines? One of the key characteristics is that modern deep learning models are *overparameterized*, meaning that the number of parameters p far exceeds the number of training examples n. It turns out that overparameterization has a very beneficial effect on optimization. Let us give some intuition for this effect. Let $f_\theta(X) \in \mathbb{R}^n$ denote a non-linear model $f_\theta : \mathsf{X} \mapsto \mathbb{R}$ applied to every training point X, and let $Y \in \mathbb{R}^n$ denote the labels. Let the Jacobian matrix $J_\theta \in \mathbb{R}^{n \times p}$, where each i-th row contains the gradient $\nabla_\theta f_\theta(x_i)$.

Consider the square error training loss for concreteness: $\hat{L}_n[\theta] = \frac{1}{2n}\|f_\theta(X) - Y\|^2$. For simplicity we will consider what happens when we run gradient descent on $\hat{L}_n[\theta]$, i.e., $\theta_{t+1} = \theta_t - \eta \nabla_\theta \hat{L}_n[\theta_t]$. First, we compute the gradient as:

$$\nabla_\theta \hat{L}_n[\theta] = \frac{1}{n} J_\theta^\mathsf{T}(f_\theta(X) - y) =: \frac{1}{n} J_\theta^\mathsf{T} r(\theta).$$

Now let us consider how $\hat{L}_n[\theta]$ changes over the course of each iteration. Using the Taylor expansion argument from Section 1.5.1:

$$\begin{aligned}\hat{L}_n[\theta_{t+1}] &= \hat{L}_n[\theta_t] + \langle \nabla_\theta \hat{L}_n[\theta_t], \theta_{t+1} - \theta_t \rangle + O(\|\theta_{t+1} - \theta_t\|^2) \\ &= \hat{L}_n[\theta_t] - \eta \|\nabla_\theta \hat{L}_n[\theta_t]\|^2 + O(\eta^2).\end{aligned}$$

However, now plugging in the specifics of the square loss, this implies that:

$$\begin{aligned}\frac{1}{2n}\|r(\theta_{t+1})\|^2 &= \frac{1}{2n}\|r(\theta_t)\|^2 - \frac{\eta}{n^2}\|J_{\theta_t}^\mathsf{T} r(\theta_t)\|^2 + O(\eta^2) \\ &\leqslant \frac{1}{2n}\|r(\theta_t)\|^2 - \frac{\eta \lambda_{\min}(J_{\theta_t} J_{\theta_t}^\mathsf{T})}{n^2}\|r(\theta_t)\|^2 + O(\eta^2) \\ &= \frac{1}{2n}\left(1 - \frac{2\eta \lambda_{\min}(J_{\theta_t} J_{\theta_t}^\mathsf{T})}{n}\right)\|r(\theta_t)\|^2 + O(\eta^2).\end{aligned}$$

Thus, for η sufficiently small we have approximately that:

$$\hat{L}_n[\theta_{t+1}] \leqslant \left(1 - \frac{2\eta \lambda_{\min}(J_{\theta_t} J_{\theta_t}^\mathsf{T})}{n}\right)\hat{L}_n[\theta_t].$$

Thus, if one can show that $\lambda_{\min}(J_{\theta_t} J_{\theta_t}^\mathsf{T})$ is uniformly bounded away from zero for all t, then this gives us an *exponential* rate of convergence to zero training error, even if $\hat{L}_n[\theta]$ is not a convex function. The question remains whether or not $J_{\theta_t} J_{\theta_t}^\mathsf{T}$ remains non-degenerate. Notice that until this point, we have not used the assumption that f_θ is overparameterized ($p \ll n$). Here is where this property comes into play. Observe that if we had an underparameterized model ($p < n$), then it would be *impossible* for $J_{\theta_t} J_{\theta_t}^\mathsf{T}$ to be non-degenerate, since $\operatorname{rank}(J_{\theta_t} J_{\theta_t}^\mathsf{T}) \leqslant \min\{n, p\}$. Hence, overparameterization is actually *necessary* for this argument to go through. Now, assuming overparameterization actually proving that $J_{\theta_t} J_{\theta_t}^\mathsf{T}$ remains non-degenerate is out of scope for this course, but has been the subject of recent research in the machine learning literature st: TODO: cite. One thing we will mention is that this condition highlights the important of *random initialization* of the parameters. By random initialization of θ_0, one can prove using random matrix theory arguments that $J_{\theta_0} J_{\theta_0}^\mathsf{T}$ starts off non-degenerate. Proving that $J_{\theta_t} J_{\theta_t}^\mathsf{T}$ remains non-degenerate throughout the course of gradient descent then involves a careful analysis of the learning dynamics. Long story short, this condition indeed holds for many types of neural networks actually used in practice.

1.6 Algorithmic stability

We now study alternative techniques for ensuring generalization. One particular method is based on algorithmic stability. The main idea is that algorithms which are more robust to small changes in the data (i.e., small perturbations to the data do not drastically alter the result) generalize better.

We will slightly redefine notation in this section to aide the analysis. As before, we fix a hypothesis class $\mathscr{F}$ and a loss function ℓ. However, in this second, the loss function maps $\ell : \mathscr{F} \times \mathsf{Z} \to \mathbb{R}$. That is, the loss function takes the pair of hypothesis and labelled data point, and produces a score (think $\ell(f, (x, y)) = \ell(f(x), y)$). Finally, we will introduce the notion of an algorithm, which is a mapping $A : \mathsf{Z}^n \times \Xi \to \mathscr{F}$. The set Ξ encapsulates any randomness used by the algorithm.

Definition 1.70 (Uniform stability)**.** *An algorithm $A : \mathsf{Z}^n \times \Xi \to \mathscr{F}$ is* ε-uniformly-stable *if for every pair of datasets $S, S' \in \mathsf{Z}^n$ which differ in at most one example,*

$$\sup_{z \in \mathsf{Z}} \mathbb{E}_\xi[\ell(A(S, \xi), z) - \ell(A(S', \xi), z)] \leqslant \varepsilon.$$

It is worth noting that the trivial algorithm $A(S, \xi) = f_0$ for some fixed f_0 (e.g., an algorithm which is *not* a function of the data) is trivially ε stable for any $\varepsilon \geqslant 0$. Of course this is not actually useful for learning, but it demonstrates that algorithmic stability only deals with generalization error and not whether the algorithm A has capacity to fit data.

For what follows we need some extra notation. Let $\{z_i\}_{i=1}^n$ and $\{z_i'\}_{i=1}^n$ be i.i.d. draws. Put $S = (z_1, \ldots, z_n)$, $S' = (z_1', \ldots, z_n')$, and $S^{(i)} = (z_1, \ldots, z_{i-1}, z_i', z_{i+1}, \ldots, z_n)$ for $i = 1, \ldots, n$. We slightly overload the $\hat{L}_n[f]$ notation with:

$$\hat{L}_S[f] := \frac{1}{n} \sum_{i=1}^n \ell(f, z_i).$$

1.6.1 Uniform stability implies generalization

Now, we show that uniform stability directly implies generalization.

Lemma 1.71 (Uniform stability implies generalization in expectation)**.** *Let $A : \mathsf{Z}^n \times \Xi \to \mathscr{F}$ be an ε-uniformly-stable algorithm (cf. Definition 1.70). Then,*

$$|\mathbb{E}_{S,\xi}[L[A(S, \xi)] - \hat{L}_S[A(S, \xi)]]| \leqslant \varepsilon.$$

Proof. This argument is essentially an exercise in exchangability. For what follows, we will abbreviate $A(S, \xi) = A(S)$ to reduce the notional overhead. The first trick is to observe that for any i, because (S, z) and $(S^{(i)}, z_i)$ are exchangeable pairs (since $(S, z) \overset{d}{=} (S^{(i)}, z_i)$),

$$\mathbb{E}_{S,\xi}[L[A(S)]] = \mathbb{E}_{S,z,\xi}[\ell(A(S), z)] = \mathbb{E}_{S',z_i,\xi}[\ell(A(S^{(i)}), z_i)] = \mathbb{E}_{S,S',\xi}[\ell(A(S^{(i)}), z_i)].$$

Since this identity holds for any $i = 1, \ldots, n$, we can average the RHS expression as follows:

$$\mathbb{E}_{S,\xi}[L[A(S)]] = \mathbb{E}_{S,S',\xi}\left[\frac{1}{n} \sum_{i=1}^n \ell(A(S^{(i)}), z_i)\right].$$

Therefore by ε-uniform-stability,

$$\mathbb{E}_{S,\xi}[L[A(S)] - \hat{L}_S[A(S)]] = \mathbb{E}_{S,S',\xi}\left[\frac{1}{n} \sum_{i=1}^n (\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i))\right] \leqslant \varepsilon.$$

Bounding $\mathbb{E}_{S,\xi}[\hat{L}_S[A(S)] - L[A(S)]]$ proceeds in a nearly identical way, from which the claim follows. □

With this result in hand, we can utilize the bounded differences inequality (Proposition B.16) to prove a high probability bound on the generalization error.

Lemma 1.72. *Suppose that $|\ell| \leqslant B$, and that $A : \mathsf{Z}^n \times \Xi \to \mathscr{F}$ is an ε-uniformly-stable algorithm. With probability at least $1 - \delta$,*

$$\mathbb{E}_\xi[L[A(S,\xi)]] \leqslant \mathbb{E}_\xi[\hat{L}_S[A(S,\xi)]] + \varepsilon + 2(\varepsilon n + B)\sqrt{\frac{2\log(1/\delta)}{n}}.$$

Proof. Again as before, we will abbreviate $A(S,\xi) = A(S)$ to reduce notational burden. Define the random variable $f(S) = \mathbb{E}_\xi[L[A(S)] - \hat{L}_S[A(S)]]$. We will show that $f(S)$ satisfies the bounded differences property. Let $S^{(i)}$ be a dataset which differs in S in the i-th location. We first observe that,

$$\mathbb{E}_\xi[L[A(S)]] - \mathbb{E}_\xi[L(A(S^{(i)}))] = \mathbb{E}_z[\mathbb{E}_\xi[\ell(A(S),z) - \ell(A(S^{(i)}),z)]].$$

Hence by the ε-uniform-stability assumption, $\mathbb{E}_\xi[L[A(S)]] - \mathbb{E}_\xi[L(A(S^{(i)}))] \leqslant \varepsilon$. Furthermore, bounding $\mathbb{E}_\xi[L(A(S^{(i)}))] - \mathbb{E}_\xi[L[A(S)]] \leqslant \varepsilon$ proceeds identically.

Next, we observe that,

$$\begin{aligned}
&\mathbb{E}_\xi[\hat{L}_S[A(S)] - \hat{L}_{S^{(i)}}[A(S^{(i)})]] \\
&= \mathbb{E}_\xi\left[\frac{1}{n}\sum_{j\neq i}[\ell(A(S),z_j) - \ell(A(S^{(i)}),z_j)] + \frac{1}{n}[\ell(A(S),z_i) - \ell(A(S^{(i)}),z_i')]\right].
\end{aligned}$$

Hence by the ε-uniform-stability assumption and the boundedness of ℓ,

$$\mathbb{E}_\xi[\hat{L}_S[A(S)] - \hat{L}_{S^{(i)}}[A(S^{(i)})]] \leqslant \varepsilon + \frac{2B}{n}.$$

Again, bounding $\mathbb{E}_\xi[\hat{L}_{S^{(i)}}[A(S^{(i)})] - \hat{L}_S[A(S)]] \leqslant \varepsilon + 2B/n$ proceeds identically. Putting these inequalities together, we see that

$$|f(S) - f(S^{(i)})| = |\mathbb{E}_\xi[L[A(S)]] - \mathbb{E}_\xi[L(A(S^{(i)}))] + \mathbb{E}_\xi[\hat{L}_S[A(S)] - \hat{L}_{S^{(i)}}[A(S^{(i)})]]| \leqslant 2\varepsilon + \frac{2B}{n}.$$

This shows that f satisfies the bounded differences inequality. Hence, by Proposition B.16, with probability at least $1 - \delta$,

$$f(S) \leqslant \mathbb{E}[f(S)] + \sqrt{8n\left(\varepsilon + \frac{B}{n}\right)^2 \log(1/\delta)} = \mathbb{E}[f(S)] + 2(\varepsilon n + B)\sqrt{\frac{2\log(1/\delta)}{n}}.$$

Finally, by Lemma 1.71, we have that $\mathbb{E}[f(S)] \leqslant \varepsilon$. The claim now follows. □

An immediate consequence of Lemma 1.72 is that we require that the uniform stability constant ε to scale as $\varepsilon = O(n^{-1})$ in order for the generalization bound from Lemma 1.72 to scale as the expected $O(n^{-1/2})$.

Remark 1.73. Note that Lemma 1.72 only prescribes a high probability bound over the randomness of the training data, and *not* over the algorithm's internal randomness. This is because our definition of uniform stability (Definition 1.70) only considers the average stability over the algorithm randomness (for a fixed (S, S', z)).

Exercise 1.74. Modify the definition of Definition 1.70 and the corresponding results in Lemma 1.71 and Lemma 1.72, so that the high probability bound holds jointly over drawing both the training data and the specific algorithm randomness. Note that there is quite a bit of flexibility in altering the definitions (as in there is no one right answer), so be sure to also discuss the pros/cons of your approach.

1.6.2 Stability of regularized least-squares regression

We now consider regularized least-squares regression. Given a dataset $S = \{(x_i, y_i)\}_{i=1}^n$ (here, y_i is not necessarily restricted to $\{\pm 1\}$), consider the algorithm, parameterized by $\lambda > 0$:

$$A_\lambda(S) = \left(\frac{1}{n}\sum_{i=1}^n x_i x_i^\mathsf{T} + \lambda I\right)^{-1} \frac{1}{n}\sum_{i=1}^n x_i y_i.$$

Note that $A_\lambda(S)$ is the unique solution to the following optimization problem:

$$\arg\min_{\theta\in\mathbb{R}^d} L_S[\theta] := \frac{1}{2n}\sum_{i=1}^n (\langle x_i, \theta\rangle - y_i)^2 + \frac{\lambda}{2}\|\theta\|^2.$$

(Convince yourself this is true!). Now let us study the stability of $A_\lambda(S)$. To do this, we will consider the absolute value loss instead of the square loss:

$$\ell(\theta, (x, y)) = |\langle x, \theta\rangle - y|.$$

This is merely for simplicity, the analysis also goes through for the square loss. Next, let us impose some regularity conditions. We assume that $\|x_i\| \leqslant 1$ and $|y_i| \leqslant 1$ for all i. Observe that under these conditions, for any $\theta_1, \theta_2 \in \mathbb{R}^d$ and (x, y),

$$\begin{aligned} ||\ell(\theta_1, (x, y))| - |\ell(\theta_2, (x, y))|| &= ||\langle x, \theta_1\rangle - y| - |\langle x, \theta_2\rangle - y|| \\ &\leqslant |\langle x, \theta_1 - \theta_2\rangle| \leqslant \|x\|\|\theta_1 - \theta_2\| \leqslant \|\theta_1 - \theta_2\|. \end{aligned}$$

Let S, S' be two datasets differing in the i-th location. To show that A is ε-uniformly-stable, it suffices to bound $\|A(S) - A(S')\|$. To do this, we start with a simple algebraic inequality. Let A_1, A_2 be two invertible matrices, and b_1, b_2 be two vectors. We have:

$$A_1^{-1}b_1 - A_2^{-1}b_2 = A_1^{-1}(b_1 - b_2) + [A_1^{-1} - A_2^{-1}]b_2.$$

Therefore,

$$\begin{aligned} \|A_1^{-1}b_1 - A_2^{-1}b_2\| &\leqslant \|A_1^{-1}\|_{\mathrm{op}}\|b_1 - b_2\| + \|A_1^{-1} - A_2^{-1}\|_{\mathrm{op}}\|b_2\| \\ &= \frac{1}{\sigma_{\min}(A_1)}\|b_1 - b_2\| + \|A_1^{-1} - A_2^{-1}\|_{\mathrm{op}}\|b_2\|. \end{aligned}$$

Next, we note another simple algebraic fact:

$$A_1^{-1} - A_2^{-1} = -A_2^{-1}(A_1 - A_2)A_1^{-1}.$$

Plugging this into the inequality above yields,

$$\|A_1^{-1}b_1 - A_2^{-1}b_2\| \leqslant \frac{1}{\sigma_{\min}(A_1)}\|b_1 - b_2\| + \frac{1}{\sigma_{\min}(A_1)\sigma_{\min}(A_2)}\|A_1 - A_2\|_{\mathrm{op}}\|b_2\|. \tag{1.45}$$

Note this generalizes the scalar identity $\frac{1}{a_1} - \frac{1}{a_2} = \frac{a_2 - a_1}{a_1 a_2}$ for non-zero $a_1, a_2 \in \mathbb{R}$. To proceed, we now set:

$$A(S) = A_1^{-1}b_1, \quad A_1 = \frac{1}{n}\sum_{i=1}^n x_i x_i^\mathsf{T} + \lambda I, \quad b_1 = \frac{1}{n}\sum_{i=1}^n x_i y_i.$$

Similarly,

$$A(S') = A_2^{-1}b_2, \quad A_2 = A_1 + \frac{1}{n}(x_i' x_i'^\mathsf{T} - x_i x_i^\mathsf{T}), \quad b_2 = b_1 + \frac{1}{n}(x_i' y_i' - x_i y_i).$$

Note that since $A_1 \succcurlyeq \lambda I$, we have that $\sigma_{\min}(A_1) = \lambda_{\min}(A_1) \geqslant \lambda$. By the same reasoning, we also have $\sigma_{\min}(A_2) \geqslant \lambda$. Furthermore, by triangle inequality and our assumptions on (x_i, y_i), we have $\|b_2\| \leqslant 1$. Plugging these estimates into (1.45), and again using our assumptions on (x_i, y_i),

$$\begin{aligned}\|A(S) - A(S')\| &\leqslant \frac{1}{\lambda}\left\|\frac{1}{n}(x_i y_i - x_i' y_i')\right\| + \frac{1}{\lambda^2}\left\|\frac{1}{n}(x_i x_i^\mathsf{T} - x_i' x_i'^\mathsf{T})\right\| \\ &\leqslant \frac{2}{\lambda n} + \frac{2}{\lambda^2 n} = \frac{2}{n}\left[\frac{1}{\lambda} + \frac{1}{\lambda^2}\right].\end{aligned}$$

Thus, we see that A is ε-uniformly-stable for $\varepsilon = 2n^{-1}(1/\lambda + 1/\lambda^2)$. Here we see the influence of the regularization parameter λ on stability: the larger λ is, the more stable A is. Indeed, as $\lambda \to \infty$, we have that the stability parameter $\varepsilon \to 0$. However, there is a trade-off: since $\|A(S)\| \leqslant 1/\lambda$, we see that λ reduces the possible hypothesis that the algorithm can select; in the limit as $\lambda \to \infty$, the algorithm is forced to choose $A(S) = 0$. Thus, we see again the trade-off between expressivity and generalization error.

1.6.3 Stability of stochastic gradient descent

With the previous section in place, we now turn to the studying the stability of specific algorithms. We will first consider the classical stochastic gradient descent (SGD) algorithm. In order to simplify the analysis, we will study a stylized version with no batching and where the data is drawn uniformly at random (instead of random shuffling of the data). Specifically, suppose we have a loss function $\ell(w, z)$, where now we let $w \in \mathbb{R}^d$. The SGD algorithm uses the following recipe for $t = 0, 1, \ldots,$

1. Let $i_t \sim \mathrm{Unif}(\{1, \ldots, n\})$.
2. Set $w_{t+1} = w_t - \eta_t \nabla_w \ell(w_t, z_{i_t})$.

To study this update rule, let us define the update operator $G_{\ell,\eta}(w, z)$ as:

$$G_{\ell,\eta}(w, z) := w - \eta \nabla_w \ell(w, z),$$

so that the SGD update rule is $w_{t+1} = G_{\ell,\eta_t}(w_t, z_{i_t})$.

Proposition 1.75 (Update operator is non-expansive)**.** *Suppose for every $z \in \mathsf{Z}$, we have $w \mapsto \ell(w, z)$ is convex and β-smooth. Then for any $u, v \in \mathbb{R}^d$, as long as $\eta \in [0, 2/\beta]$, we have that the update operator $G_{\ell,\eta}$ is non-expansive, i.e.,*

$$\forall z \in \mathsf{Z},\ \|G_{\ell,\eta}(u, z) - G_{\ell,\eta}(v, z)\| \leqslant \|u - v\|.$$

Proof. From Proposition C.6, since for every z the function $w \mapsto \ell(w, z)$ is convex and β-smooth, we have the following co-coercivity condition:

$$\langle \nabla_w \ell(u, z) - \nabla_w \ell(v, z), u - v\rangle \geqslant \frac{1}{\beta}\|\nabla_w \ell(u, z) - \nabla_w \ell(v, z)\|^2.$$

Therefore,

$$\begin{aligned}&\|G_{\ell,\eta}(u, z) - G_{\ell,\eta}(v, z)\|^2 \\ &= \|(u - v) - \eta(\nabla_w \ell(u, z) - \nabla_w \ell(v, z))\|^2 \\ &= \|u - v\|^2 - 2\eta\langle u - v, \nabla_w \ell(u, z) - \nabla_w \ell(v, z)\rangle + \eta^2\|\nabla_w \ell(u, z) - \nabla_w \ell(v, z)\|^2 \\ &\leqslant \|u - v\|^2 - (2\eta/\beta - \eta^2)\|\nabla_w \ell(u, z) - \nabla_w \ell(v, z)\|^2 \\ &\leqslant \|u - v\|^2. \qquad \text{since } \eta \in [0, 2/\beta]\end{aligned}$$

□

Lemma 1.76 (SGD is stable on convex functions)**.** *Suppose that for every $z \in \mathsf{Z}$, we have $w \mapsto \ell(w, z)$ is convex, L-Lipschitz, and β-smooth. Then, as long as $\eta_t \in [0, 2/\beta]$ for all t, running the SGD algorithm for T iterations is ε-uniformly-stable for*

$$\varepsilon \leqslant \frac{2L^2}{n} \sum_{t=0}^{T-1} \eta_t.$$

Proof. Let us consider running SGD on two datasets $S = \{z_i\}_{i=1}^n$ and $S' = \{z_i'\}_{i=1}^n$, where S, S' differ in exactly one data point. Let $\kappa \in \{1, \dots, n\}$ denote the index which the datasets differ. We are going to use what is known as a *coupling argument*. We let the iterates $\{u_t\}$ denote SGD running on S, and $\{v_t\}$ denote SGD running on S'. The coupling argument works by using the *same* randomness ξ for both executions (in this case, the same indices $\{i_t\}$ are drawn), and then bounding $\mathbb{E}_\xi \|u_T - v_T\|$ (the dependence of the u_t's and v_t's on ξ is suppressed in our notation). To understand why this is correct, recall the definition of uniform stability (Definition 1.70):

$$\sup_{z \in \mathsf{Z}} \mathbb{E}_\xi[\ell(u_T, z) - \ell(v_T, z)] \leqslant \varepsilon.$$

To control the LHS, we use the L-Lipschitz assumption on ℓ, which implies

$$\sup_{z \in \mathsf{Z}} \mathbb{E}_\xi[\ell(u_T, z) - \ell(v_T, z)] \leqslant L\mathbb{E}_\xi \|u_T - v_T\|.$$

Thus it suffices to focus on the difference of the iterates, averaged over the randomness of ξ.

To proceed, observe that since SGD initializes its weighs independently of the data (but possibly using the internal randomness ξ), we start with $u_0 = v_0$. Let $\delta_t := \|u_t - v_t\|$. We will write a recursion for δ_t by decomposing as follows:

$$\delta_{t+1} = \delta_{t+1}\mathbf{1}\{i_t = \kappa\} + \delta_{t+1}\mathbf{1}\{i_t \neq \kappa\}.$$

Let us first consider $\delta_{t+1}\mathbf{1}\{i_t \neq \kappa\}$, for which we have:

$$\begin{aligned}
\delta_{t+1}\mathbf{1}\{i_t \neq \kappa\} &= \|G_{\ell,\eta_t}(u_t, z_{i_t}) - G_{\ell,\eta_t}(v_t, z'_{i_t})\|\mathbf{1}\{i_t \neq \kappa\} \\
&= \|G_{\ell,\eta_t}(u_t, z_{i_t}) - G_{\ell,\eta_t}(v_t, z_{i_t})\|\mathbf{1}\{i_t \neq \kappa\} && \text{since } z_{i_t} = z'_{i_t} \text{ when } i_t \neq \kappa \\
&\leqslant \|u_t - v_t\|\mathbf{1}\{i_t \neq \kappa\} && \text{using Proposition 1.75} \\
&= \delta_t\mathbf{1}\{i_t \neq \kappa\}.
\end{aligned}$$

On the other hand,

$$\begin{aligned}
\delta_{t+1}\mathbf{1}\{i_t = \kappa\} &= \|u_t - v_t - \eta_t(\nabla_w \ell(u_t, z_{i_t}) - \nabla_w \ell(v_t, z'_{i_t}))\|\mathbf{1}\{i_t = \kappa\} \\
&\leqslant (\|u_t - v_t\| + 2\eta_t L)\mathbf{1}\{i_t = \kappa\} && \text{since } \ell \text{ is } L\text{-Lipschitz} \\
&= (\delta_t + 2\eta_t L)\mathbf{1}\{i_t = \kappa\}.
\end{aligned}$$

Combining these two inequalities,

$$\delta_{t+1} \leqslant \delta_t + 2\eta_t L\mathbf{1}\{i_t = \kappa\} \Longrightarrow \delta_T \leqslant 2L \sum_{t=0}^{T-1} \eta_t \mathbf{1}\{i_t = \kappa\}.$$

Since this inequality holds for every ξ, by linearity of expectation,

$$\mathbb{E}_\xi[\delta_T] \leqslant \frac{2L}{n} \sum_{t=0}^{T-1} \eta_t.$$

The claim now follows since ℓ is L-Lipschitz:

$$\sup_{z \in \mathsf{Z}} \mathbb{E}_\xi[\ell(u_T, z) - \ell(v_T, z)] \leqslant L\mathbb{E}_\xi \|u_T - v_T\| = L\mathbb{E}_\xi[\delta_T] \leqslant \frac{2L^2}{n} \sum_{t=0}^{T-1} \eta_t.$$

□

Remark 1.77. Note that the proof of Lemma 1.76 illustrates why the definition of uniform stability in Definition 1.70 requires that stability only hold in expectation over the internal randomness of the algorithm. If, for instance, the definition required that the algorithm was stable for almost every realization of the randomness, then SGD would not be stable under this definition, since the realization ξ which repeatedly picks the index corresponding to the differing point κ is not stable (despite being very unlikely) .

Sub-optimal rates for typical SGD step sizes. Note that one drawback of Lemma 1.76 is that for the typical $\eta_t \asymp t^{-1/2}$ step sizes usually prescribed for running SGD on convex functions, the ε-uniform-stability constant scales as $\varepsilon = O(\sqrt{T}/n)$. So for any non-trivial amount of steps T, we have that ε is slower than the n^{-1} scaling required so that Lemma 1.72 yields $O(n^{-1/2})$ generalization bounds. On the other hand, if we use a more aggressive step size $\eta_t \asymp t^{-1}$, then we have that $\varepsilon = O(\log T/n)$ which improves the overall generalization rate, but is generally too aggressive. If we impose extra conditions on the function ℓ, specifically that it is strongly convex (Definition C.7), then it turns out we can get a stability bound which is independent of the number of iterations run.

Exercise 1.78. Suppose that for every $z \in \mathsf{Z}$, we have that $w \mapsto \ell(w, z)$ is μ-strongly-convex (cf. Definition C.7), L-Lipschitz, and β-smooth.

(a) Show that if $\eta \leqslant 2/(\mu + \beta)$, the update operator $G_{\ell,\eta}$ becomes *contractive*, i.e., for any $u, v \in \mathbb{R}^d$:

$$\forall z \in \mathsf{Z},\ \|G_{\ell,\eta}(u, z) - G_{\ell,\eta}(u, z)\| \leqslant \left(1 - \frac{\eta\mu\beta}{\mu + \beta}\right) \|u - v\|.$$

Hint: you may assume the following fact about μ-strongly-convex and β-smooth functions without proof. For all $u, v \in \mathbb{R}^d$,

$$\langle \nabla f(u) - \nabla f(v), u - v\rangle \geqslant \frac{\mu\beta}{\mu + \beta}\|u - v\|^2 + \frac{1}{\mu + \beta}\|\nabla f(u) - \nabla f(v)\|^2.$$

(b) Modify the proof of Lemma 1.76 to show the following recursion, valid whenever $\eta_t \leqslant 2/(\mu + \beta)$:

$$\delta_{t+1} \leqslant (1 - \eta_t m)\delta_t + 2\eta_t L \mathbf{1}\{i_t = \kappa\}, \quad m := \frac{\mu\beta}{\mu + \beta}.$$

(c) Now suppose we use a *constant* step size $\eta_t = \eta \leqslant 2/(\mu + \beta)$. Conclude that:

$$\mathbb{E}[\delta_T] \leqslant \frac{2L}{mn},$$

and hence SGD is ε-uniformly-stable for $\varepsilon \leqslant \frac{2L^2}{mn}$.

1.6.4 Stability of Gibbs ERM

We now study the stability of an idealized algorithm which returns a sample from a particular weighted Gibbs distribution. In particular, let $p_\beta(w; S)$ be the density of a distribution on $\mathbb{R}^d$ defined as:

$$p_\beta(w; S) = \exp(-\beta \hat{L}_S[w])/Z_\beta[w; S], \quad Z_\beta[w; S] = \int \exp(-\beta \hat{L}_S[w])\, \mathrm{d}w. \tag{1.46}$$

The parameter $\beta \in \mathbb{R}$ is referred to as the *temperature* parameter. The idea behind the Gibbs ERM is to select hypothesis $w \in \mathbb{R}^d$ which correspond to low empirical risk $\hat{L}_S[w]$. However, by not just selecting the true ERM, but instead allowing for a randomized selection rule, we actually do not require any convexity assumptions on $\hat{L}_S$ to ensure uniform stability.

Proposition 1.79 (Gibbs ERM is uniformly-stable)**.** *Let $A(S,\xi)$ denote the algorithm which returns a sample from the Gibbs distribution defined by Equation (1.46). Suppose that $|\ell| \leqslant M$. We have that A is ε-uniformly-stable with $\varepsilon \leqslant (\exp(4\beta M/n) - 1)M$.*

Proof. Let S, S' differ in only the i-th index. Note that:

$$\begin{aligned}\mathbb{E}_\xi[\ell(A(S,\xi),z) - \ell(A(S',\xi),z)] &= \int \ell(w,z)p_\beta(w;S)\,\mathrm{d}w - \int \ell(w,z)p_\beta(w;S')\,\mathrm{d}w \\ &= \int \ell(w,z)\left(\frac{p_\beta(w;S)}{p_\beta(w;S')} - 1\right)p_\beta(w;S')\,\mathrm{d}w \\ &\leqslant M\left|\frac{p_\beta(w;S)}{p_\beta(w;S')} - 1\right|.\end{aligned}$$

To bound the likelihood ratio, we first observe that,

$$\frac{\exp(-\beta\hat{L}_S[w])}{\exp(-\beta\hat{L}_{S'}[w])} = \exp\left(-\frac{\beta}{n}(\ell(w,z_i) - \ell(w,z_i'))\right) \leqslant \exp(2\beta M/n).$$

On the other hand,

$$\frac{Z_\beta[w;S]}{Z_\beta[w;S']} = \frac{\int \exp(-\beta\hat{L}_S[w])\,\mathrm{d}w}{\int \exp(-\beta\hat{L}_{S'}[w])\,\mathrm{d}w} = \frac{\int \exp\left(-\frac{\beta}{n}(\ell(w,z_i) - \ell(w,z_i'))\right)\exp(-\beta\hat{L}_{S'}[w])\,\mathrm{d}w}{\int \exp(-\beta\hat{L}_{S'}[w])\,\mathrm{d}w} \leqslant \exp(2\beta M/n).$$

Thus, we have:

$$\max\left\{\frac{\exp(-\beta\hat{L}_S[w])}{\exp(-\beta\hat{L}_{S'}[w])}, \frac{Z_\beta[w;S]}{Z_\beta[w;S']}\right\} \leqslant \exp(2\beta M/n).$$

Similar arguments show that

$$\min\left\{\frac{\exp(-\beta\hat{L}_S[w])}{\exp(-\beta\hat{L}_{S'}[w])}, \frac{Z_\beta[w;S]}{Z_\beta[w;S']}\right\} \geqslant \exp(-2\beta M/n).$$

Hence,

$$\begin{aligned}\left|\frac{p_\beta(w;S)}{p_\beta(w;S')} - 1\right| &= \left|\frac{\exp(-\beta\hat{L}_S[w])}{\exp(-\beta\hat{L}_{S'}[w])}\cdot\frac{Z_\beta[w;S']}{Z_\beta[w;S]} - 1\right| \\ &= \max\left\{\frac{\exp(-\beta\hat{L}_S[w])}{\exp(-\beta\hat{L}_{S'}[w])}\cdot\frac{Z_\beta[w;S']}{Z_\beta[w;S]} - 1, 1 - \frac{\exp(-\beta\hat{L}_S[w])}{\exp(-\beta\hat{L}_{S'}[w])}\cdot\frac{Z_\beta[w;S']}{Z_\beta[w;S]}\right\} \\ &\leqslant \max\left\{\exp\left(\frac{4\beta M}{n}\right) - 1, 1 - \exp\left(-\frac{4\beta M}{n}\right)\right\} \\ &= \exp(4\beta M/n) - 1.\end{aligned}$$

The last equality holds since for all $x \in \mathbb{R}$,

$$\frac{e^x + e^{-x}}{2} = \cosh(x) \geqslant 1.$$

□

Remark 1.80. Note that for β, M fixed, as n grows larger, we have that:

$$(\exp(4\beta M/n) - 1)M = \frac{4\beta M^2}{n} + o(1/n).$$

Hence the Gibbs-ERM has $O(1/n)$ uniform stability scaling.

We conclude this section with a discussion on the computation aspects of sampling from the Gibbs distribution (1.46). Without any particular structure imposed on $\hat{L}_S$, the most general algorithm for sampling from the Gibbs distribution is a type of Markov Chain Monte Carlo (MCMC) algorithm known as *stochastic gradient Langevin dynamics (SGLD)*. To derive the SGLD update, the first step is to consider using *Langevin dynamics* to sample from (1.46):

$$w_{t+1} = w_t - \eta\beta\nabla\hat{L}_S[w_t] + \sqrt{2\eta}\xi_t, \quad \xi_t \sim N(0, I).$$

We will return to discussing Langevin dynamics in more detail when we study energy-based models. The SGLD update simply uses an unbiased estimator g_t such that $\mathbb{E}[g_t \mid w_t] = \nabla\hat{L}_S[w_t]$ (as is done in standard SGD):

$$w_{t+1} = w_t - \eta\beta g_t + \sqrt{2\eta}\xi_t, \quad \xi_t \sim N(0, I).$$

The convergence rate of SGLD is quite an advanced topic, and we will only briefly touch on it later in this course.

1.7 PAC-Bayes inequalities

We now study an alternative approach to controlling generalization error. When we studied the generalization of ERM, we essentially proved uniform convergence of training and population risk over the entire hypothesis space $\mathscr{F}$. One downside of this approach is that it uniformly weights every hypothesis in the function class. An alternative approach is to construct a prior distribution over hypothesis, and take a "weighted" union bound over the entire function class which takes into account the prior. This is what the PAC-Bayes approach does. After we derive the main PAC-Bayes deviation inequality, we will instantiate it for a few special cases, including what are referred to as compression-based bound in the literature.

Perhaps the main conceptual difference of PAC-Bayes analysis compared with ERM analysis is that the learning algorithm is now a *distribution* over hypothesis $\mathscr{F}$, which we will refer to as the posterior (hence where the "Bayes" part of the name stems from). In PAC-Bayes, we start with a prior π over $\mathscr{F}$, data $\{z_i\}_{i=1}^n$ is observed, and then a posterior distribution ρ_n over $\mathscr{F}$ is formed by incorporating the observed data. Unlike ERM analysis, the main quantity which governs the generalization error is the KL-divergence between the posterior ρ_n and the prior π.

While there are many variants of PAC-Bayes inequalities, we will study one due to Cantoni, described by Alquier [2021].

Theorem 1.81 (PAC-Bayes deviation inequality)**.** *Suppose that $|\ell| \leqslant B$. For any $\lambda > 0$ and $\delta \in (0, 1)$, we have with probability at least $1 - \delta$,*

$$\forall \rho \in \mathcal{P}(\mathscr{F}),\ \mathbb{E}_{f\sim\rho}[L[f]] \leqslant \mathbb{E}_{f\sim\rho}[\hat{L}_n[f]] + \frac{\lambda B^2}{2n} + \frac{\mathrm{KL}(\rho \,\|\, \pi) + \log(1/\delta)}{\lambda}.$$

Here, $\mathcal{P}(\mathscr{F})$ refers to the set of probability measures over the hypothesis class $\mathscr{F}$.

Proof. The main tool used to derive the PAC-Bayes inequality is the Donsker-Varadhan representation of KL-divergence (Lemma A.3). We first set things up in order to apply Donsker-Varadhan.

We start with a fixed $f \in \mathscr{F}$. By Hoeffding's inequality (Proposition B.12),

$$\mathbb{E}_S \exp(\lambda(L[f] - \hat{L}_n[f])) \leqslant \exp(\lambda^2 B^2/(2n)).$$

(Here $\mathbb{E}_S$ denotes expectation w.r.t. the data $S = \{z_i\}$.) Note that since this inequality holds for every fixed $f \in \mathscr{F}$, it also holds if we integrate it w.r.t. the prior π (this works because the prior *does not depend on* the data S). Hence,

$$\mathbb{E}_{f\sim\pi}\mathbb{E}_S \exp(\lambda(L[f] - \hat{L}_n[f])) \leqslant \exp(\lambda^2 B^2/(2n)).$$

Now by Fubini's theorem, we exchange the order of expectations on the LHS,

$$\mathbb{E}_S \mathbb{E}_{f\sim\pi} \exp(\lambda(L[f] - \hat{L}_n[f])) \leqslant \exp(\lambda^2 B^2/(2n)).$$

The next step is to apply the Donsker-Varadhan lemma (Lemma A.3). Specifically, we will use the following identity which holds for any fixed g,

$$\log \mathbb{E}_\pi[\exp(g)] = \sup_{\rho\in\mathcal{P}(\mathscr{F})} \{\mathbb{E}_\rho[g] - \mathrm{KL}(\rho \,\|\, \pi)\}.$$

Applying this identity to $g = \lambda(L[f] - \hat{L}_n[f])$ and taking an expectation w.r.t. S on both sides,

$$\mathbb{E}_S \mathbb{E}_\pi \exp\left(\lambda(L[f] - \hat{L}_n[f])\right) = \mathbb{E}_S \exp\left(\sup_{\rho\in\mathcal{P}(\mathscr{F})} \left\{\lambda \mathbb{E}_\rho[L[f] - \hat{L}_n[f]] - \mathrm{KL}(\rho \,\|\, \pi)\right\}\right) \leqslant \exp(\lambda^2 B^2/(2n)).$$

Hence for any $t > 0$, by the Laplace transform method (Proposition B.2),

$$\begin{aligned}
&\mathbb{P}_S\left\{\sup_{\rho\in\mathcal{P}(\mathscr{F})} \left\{\lambda \mathbb{E}_\rho[L[f] - \hat{L}_n[f]] - \mathrm{KL}(\rho \,\|\, \pi)\right\} - \frac{\lambda^2 B^2}{2n} \geqslant t\right\} \\
&\leqslant e^{-t} \mathbb{E}_S \exp\left(\sup_{\rho\in\mathcal{P}(\mathscr{F})} \left\{\lambda \mathbb{E}_\rho[L[f] - \hat{L}_n[f]] - \mathrm{KL}(\rho \,\|\, \pi)\right\} - \frac{\lambda^2 B^2}{2n}\right) \leqslant e^{-t}.
\end{aligned}$$

The claim now follows by setting $t = \log(1/\delta)$. □

Remark 1.82. The order of the quantifier for λ in Theorem 1.81 is important. The statement requires that you choose a λ first, and then the resulting deviation inequality holds. Note in particular, that the following is in general **not true** as a direct consequence of Theorem 1.81: for all $\rho \in \mathcal{P}(\mathscr{F})$,

$$\begin{aligned}
\mathbb{E}_{f\sim\rho}[L[f]] &\leqslant \mathbb{E}_{f\sim\rho}[\hat{L}_n[f]] + \inf_{\lambda>0}\left[\frac{\lambda B^2}{2n} + \frac{\mathrm{KL}(\rho \,\|\, \pi) + \log(1/\delta)}{\lambda}\right] \\
&= \mathbb{E}_{f\sim\rho}[\hat{L}_n[f]] + B\sqrt{\frac{2(\mathrm{KL}(\rho \,\|\, \pi) + \log(1/\delta))}{n}}. \qquad (1.47)
\end{aligned}$$

This is because the optimizing λ above, which is

$$\lambda = \sqrt{\frac{\mathrm{KL}(\rho \,\|\, \pi) + \log(1/\delta)}{B^2/(2n)}},$$

is a function of the posterior ρ. Hence (1.47) does not hold for every posterior ρ generally. However, there are some special cases where an inequality of the form (1.47) does hold over a subset of posteriors $\mathcal{P}' \subset \mathcal{P}(\mathscr{F})$. In particular, if every $\rho \in \mathcal{P}'$ has the same value of $\mathrm{KL}(\rho \,\|\, \pi)$, then (1.47) does indeed hold for every $\rho \in \mathcal{P}'$ (convince yourself why). Furthermore, we can optimize λ over any fixed range with only a small penalty, as shown next.

Exercise 1.83. Suppose that $|\ell| \leqslant B$. Fix constants $0 < \alpha \leqslant \beta < \infty$ and $\delta \in (0, 1)$. Show that with probability at least $1 - \delta$,

$$\forall \rho \in \mathcal{P}(\mathscr{F}),\ \mathbb{E}_{f\sim\rho}[L[f]] \leqslant \mathbb{E}_{f\sim\rho}[\hat{L}_n[f]] + \inf_{\lambda\in[\alpha,\beta]} \left\{\frac{\lambda B^2}{2n} + \frac{\mathrm{KL}(\rho \,\|\, \pi) + \log((\lfloor \log(\beta/\alpha) \rfloor + 1)/\delta)}{\lambda/e}\right\}.$$

Gibbs posterior. Since Theorem 1.81 holds for every $\rho \in \mathcal{P}(\mathscr{F})$, it is natural to ask the question which posterior minimizes the bound on the RHS. In particular, what is the solution to:

$$\begin{aligned}\arg\min_{\rho\in\mathcal{P}(\mathscr{F})}\left\{\mathbb{E}_{f\sim\rho}[\hat{L}_n[f]]+\frac{\lambda B^2}{2n}+\frac{\mathrm{KL}(\rho\,\|\,\pi)+\log(1/\delta)}{\lambda}\right\} &= \arg\min_{\rho\in\mathcal{P}(\mathscr{F})}\left\{\mathbb{E}_{f\sim\rho}[\hat{L}_n[f]]+\frac{\mathrm{KL}(\rho\,\|\,\pi)}{\lambda}\right\}\\ &= \arg\max_{\rho\in\mathcal{P}(\mathscr{F})}\left\{-\mathbb{E}_{f\sim\rho}[\hat{L}_n[f]]-\frac{\mathrm{KL}(\rho\,\|\,\pi)}{\lambda}\right\}\\ &= \arg\max_{\rho\in\mathcal{P}(\mathscr{F})}\left\{\mathbb{E}_{f\sim\rho}[-\lambda\hat{L}_n[f]]-\mathrm{KL}(\rho\,\|\,\pi)\right\}.\end{aligned}$$

But this question is precisely answered by the Donsker-Varadhan lemma (Lemma A.3), which states that the solution to the maximization above has density w.r.t. the prior π defined as:

$$\frac{\mathrm{d}\rho_\lambda}{\mathrm{d}\pi}(f)=\frac{\exp(-\lambda\hat{L}_n[f])}{\mathbb{E}_{f\sim\pi}[\exp(-\lambda\hat{L}_n[f])]}.$$

Compare this with the Gibbs ERM distribution introduced in Section 1.6.4 (cf. Equation (1.46)), which is essentially a special case of the above display with a uniform prior.

We now consider some specific instantiations of Theorem 1.81.

Finite hypothesis classes. Let us first consider to what extent PAC-Bayes generalizes the uniform convergence analysis of ERM. First, we see that the PAC-Bayes inequality implies the uniform convergence result from Proposition 1.20. Indeed, suppose $\mathscr{F}$ is finite, and put $M = |\mathscr{F}|$. Let us consider the uniform prior on $\mathscr{F}$, i.e.,

$$\pi(f)=\frac{1}{M}.$$

Now consider a posterior $\rho_f = \delta_f$, which is a Dirac mass on a fixed $f \in \mathscr{F}$. An immediate computation yields that:

$$\mathrm{KL}(\rho_f\,\|\,\pi)=\log\left(\frac{1}{\pi(f)}\right)=\log M.$$

Note that since $\mathrm{KL}(\rho_f \,\|\, \pi)$ is independent of f, then we can apply the argument from Remark 1.82 (using the subset of posteriors $\mathcal{P}' = \{\rho_f \mid f \in \mathscr{F}\}$) to conclude that, with probability at least $1-\delta$,

$$\forall f\in\mathscr{F},\ L[f]\leqslant\hat{L}_n[f]+B\sqrt{\frac{2\log(M/\delta)}{n}}.$$

This is precisely the statement of Proposition 1.20.

Compression based generalization. Let us now explore an idea from the deep learning literature, known as compression-based bounds. There is a vast literature here, so we will only cover the very basics. The following discussion is motivated by Zhou et al. [2019]. The intuition is that many deep learning models, while vastly overparameterized, can actually be compressed (in a lossy manner) but still retain much of its performance. The PAC-Bayes framework suggests a way to utilize this insight in order to derive sharper generalization bounds, by building a prior distribution which upweights compressible models. Let us describe a simplistic model of this. Let M denote a max bound on the hypothesis we will consider, denoting the number of bits used to represent the model (hence there are $|\mathscr{F}| = \sum_{i=1}^{M} 2^i = 2(2^M - 1)$ possible models in total). For each model $f \in \mathscr{F}$, we let $|f| \in \{1, \dots, M\}$ denote the number of bits used to represent f. With this notation, let us consider a prior π on $\mathscr{F}$ such that:

$$\pi(f)=\frac{2^{-|f|}}{M}.$$

This prior distribution is described by the following sampling scheme: (a) choose an index $i \sim \mathrm{Unif}(\{1, \dots, M\})$, and (b) choose an f with $|f| = i$ uniformly at random.

Now if we let ρ_f denote a point mass on model f, a quick computation yields:

$$\mathrm{KL}(\rho_f \parallel \pi) = \log\left(\frac{1}{\pi(f)}\right) = |f| \log 2 + \log M.$$

This prior can now be used to construct the following compression-bound.

Exercise 1.84. Suppose that $\mathscr{F}$ is as described above. Show that the PAC-Bayes deviation bound (Theorem 1.81) implies that with probability at least $1 - \delta$,

$$\forall f \in \mathscr{F},\ L[f] \leqslant \hat{L}_n[f] + cB\sqrt{\frac{|f| + \log(M/\delta)}{n}}.$$

Here, $c > 0$ is a universal constant.

Applying the standard uniform convergence bound for finite hypothesis classes (Proposition 1.20) to this setting yields a bound of:

$$\max_{f \in \mathscr{F}} \mathrm{gen}[f] \leqslant B\sqrt{\frac{2\log(|\mathscr{F}|/\delta)}{n}} \leqslant cB\sqrt{\frac{M + \log(1/\delta)}{n}},$$

for an absolute constant c. Compared this with Exercise 1.84 where compressible models (e.g. $|f| \ll M$) enjoy a much tighter generalization bound. Of course, in the worst case when $|f| \asymp M$, the Exercise 1.84 yields orderwise the same bound.

Gaussian priors. We now consider the case when our function class $\mathscr{F}$ is parametric of the form:

$$\mathscr{F} = \{f_\theta(x) \mid \theta \in \mathbb{R}^d\},$$

where f_θ is some arbitrary parametric function of θ. In this case, a very natural prior is to set $\pi = N(\mu, \sigma^2 I)$, for some fixed μ, σ. Then, considering posteriors of the form $\rho_\theta = N(\theta, \sigma^2 I)$, we can compute, using the formula for the KL divergence of two Gaussians (cf. Proposition A.4):

$$\mathrm{KL}(\rho_\theta \parallel \pi) = \frac{1}{2\sigma^2}\|\theta - \mu\|^2.$$

Therefore, the PAC-Bayes guarantee tells us that for any $\lambda > 0$, with probability at least $1 - \delta$,

$$\forall \theta \in \mathbb{R}^d,\ \mathbb{E}_{w \sim N(0,I)}[L[f_{\theta + \sigma w}]] \leqslant \mathbb{E}_{w \sim N(0,I)}[\hat{L}_n[f_{\theta+\sigma w}]] + \frac{\lambda B^2}{2n} + \frac{\frac{1}{2\sigma^2}\|\theta - \mu\|^2 + \log(1/\delta)}{\lambda}. \tag{1.48}$$

Thus, we see that the ℓ_2 distance from our prior parameter vector μ controls our generalization error. Indeed, there is *no a-prior* bound on θ here (which is typically needed for Rademacher type analysis). Instead, we simply pay a larger price for generalization the more our weight vector θ deviates from μ.

One way to utilize the guarantee in (1.48) is to use the RHS of (1.48) as the training objective. Indeed, since the RHS gives us a bound on the generalization error, and since it holds for every $\theta \in \mathbb{R}^d$, then we can minimize it as a way of selecting the final model θ. This approach was taken recently in Dziugaite and Roy [2017] which allowed them to compute a non-trivial (i.e., a resulting bound less than one for the zero-one loss) generalization bound for a deep network trained on MNIST. Many more recent follow-up works have employed more PAC-Bayes tricks (beyond the score of this course) to extend the models and datasets for which a non-trivial generalization bound can be computed.

Data splits and trainable priors. We now turn to the question of how to choose the prior. One approach is to split the dataset in a way that the prior is actually learnable from data. Let us again consider the parametric family $\mathscr{F} = \{f_\theta(x) \mid \theta \in \mathbb{R}^d\}$, and consider Gaussian priors of the form $\pi = N(\mu, \sigma^2 I)$. Suppose we are given n training examples $S_n = \{(x_i, y_i)\}_{i=1}^n$. One valid way to choose the μ is to split the dataset into two. That is, for any $m \in \{1, \dots, n-1\}$, let us split S_n into two sets:

$$S_{1:m} = \{(x_i, y_i)\}_{i=1}^m, \quad S_{m+1:n} = \{(x_i, y_i)\}_{i=m+1}^n.$$

We use $S_{1:m}$ to learn μ, for example by solving an ERM problem:

$$\hat{\mu}_{1:m} \in \arg\min_{\theta \in \mathbb{R}^d} \hat{L}_{S_{1:m}}[f_\theta] = \frac{1}{m} \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i).$$

We then apply Theorem 1.81 using the prior $\pi = N(\hat{\mu}_{1:m}, \sigma^2 I)$, to conclude that with probability at least $1 - \delta$ over $S_{m+1:n}$,

$$\forall \theta \in \mathbb{R}^d, \ \mathbb{E}_{w \sim N(0,I)}[L[f_{\theta+\sigma w}]] \leqslant \mathbb{E}_{w \sim N(0,I)}[\hat{L}_n[f_{\theta+\sigma w}]] + \frac{\lambda B^2}{2(n-m)} + \frac{\frac{1}{2\sigma^2}\|\theta - \hat{\mu}_{1:m}\|^2 + \log(1/\delta)}{\lambda}.$$

2 Unsupervised learning and generative modeling

2.1 Principal components analysis

In this section, we develop principal components analysis (PCA) from first principles. We first review the details of the PCA algorithm in Algorithm 2.

Algorithm 2 Principle components analysis (PCA)

Input: A data matrix $X \in \mathbb{R}^{n \times d}$, a dimension $k \in \{1, \dots, d\}$.
Output: A orthogonal basis $V \in \mathbb{R}^{d \times k}$.
`/* Center the data */`
$\bar{X} = X - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\mathsf{T} X$.
`/* Compute singular value decomposition (SVD) */`
$\_, \_, V^\mathsf{T} = \mathsf{SVD}(\bar{X})$. `//` $\bar{X} = U\Sigma V^\mathsf{T}$
`/* Return the right singular vectors corresponding to the top k singular values */`
return $V[:, :k]$. `// numpy notation`

The orthogonal basis $V \in \mathbb{R}^{d \times k}$ output from PCA (Algorithm 2) can then be used to transform the original d-dimensional data into a k-dimensional space via $V^\mathsf{T} x$; if $X \in \mathbb{R}^{n \times d}$ is the data matrix, then $XV \in \mathbb{R}^{n \times k}$ is the new lower-dimensional data matrix.

At its very core, dimensionality reduction is a lossy transformation; information about the original dataset generally will be lost after transforming the dataset into a lower dimensional space. Furthermore, there are an infinite number of transforms one can apply to a dataset. Thus, we now turn to the question of in what sense does PCA compute an optimal projection.

To motivate our discussion, suppose we want to design a dimesionality reduction scheme from scratch. The first question we might ask ourselves is: what type of projection are we going to consider? For expressivity we might want to consider non-linear projections, such as those that arise from the output of a neural network, but let us suppose we restrict ourselves to linear projections so we can access the tools of linear algebra. Also, from a practical perspective, linear projections have many computational advantages.

Given that we consider only linear projections, we can now define what it means for a projection to be optimal, and discuss how we solve the resulting optimization problem. As a starting point, fix a k-dimensional subspace W of $\mathbb{R}^d$. Given a data point $x \in \mathbb{R}^d$, we compute its projection onto W by finding

the point $y \in W$ such that $\|x - y\|$ is minimized: $\arg\min_{y \in W} \|x - y\|$. From linear algebra, we know that this optimization problem is solved as:

$$P_W x = \arg\min_{y \in W} \|x - y\|,$$

where $P_W \in \mathbb{R}^{d \times d}$ is the *orthogonal projector* onto the subspace W. Next, we consider the error introduced by such a projection. Indeed, the *reconstruction error* is

$$x - P_W x = (I - P_W)x = P_W^{\perp} x,$$

which is the projection of x onto the *orthogonal complement* of W. Thus, a very natural metric to minimize when choosing W is to minimize the reconstruction error. For a given x, the W which minimizes the reconstruction error is to choose $W = \text{span}\{x\}$, in which case the reconstruction error is zero. However, this is a fairly poor choice since in general we have many x in mind not just a single x. Hence, a better choice is to place a distribution over x, and minimize the expected square error of the reconstruction error:

$$W \in \arg\min_{\dim(W)=k} \mathbb{E}\|P_W^{\perp} x\|^2.$$

The solution to this optimization problem is precisely given by the the top-k eigenvectors of the second moment matrix $\mathbb{E}[xx^{\mathsf{T}}]$, as shown below.

Proposition 2.1. *Let $\Sigma := \mathbb{E}[xx^{\mathsf{T}}] \in \mathbb{R}^{d \times d}$, and let $v_1, \ldots, v_k$ correspond to a basis for the eigenspace of the top-k eigenvalues of Σ (ties can be broken arbitrarily). We have that:*

$$\text{span}\{v_1, \ldots, v_k\} \in \arg\min_{\dim(W)=k} \mathbb{E}\|P_W^{\perp} x\|^2.$$

Proof. Let $O(d, k)$ denote:

$$O(d, k) := \{P \in \mathbb{R}^{d \times k} \mid P^{\mathsf{T}} P = I\},$$

There is a one-to-one correspondance between $O(d, k)$ and k-dimensional subspaces of $\mathbb{R}^d$, i.e.,

$$\begin{aligned}
\min_{\dim(W)=k} \mathbb{E}\|P_W^{\perp} x\|^2 &= \min_{P \in O(d,k)} \mathbb{E}\|(I - PP^{\mathsf{T}})x\|^2 \\
&= \min_{P \in O(d,k)} \text{tr}((I - PP^{\mathsf{T}})\mathbb{E}[xx^{\mathsf{T}}]) \\
&= \text{tr}(\Sigma) - \max_{P \in O(d,k)} \text{tr}(PP^{\mathsf{T}}\Sigma).
\end{aligned}$$

We now focus on the optimization problem $\max_{P \in O(d,k)} \text{tr}(PP^{\mathsf{T}}\Sigma)$. Let $U\Lambda U^{\mathsf{T}} = \Sigma$ denote the eigendecomposition of Σ, so that the columns of U are listed in decreasing order corresponding to the eigenvalues (the first column of U corresponds to the maximum eigenvalue λ_1, the second to λ_2, and so on). The solution is to set $P = U[:, :k]$ (this is using `numpy` notation). To see this, let us first upper bound the value of $\text{tr}(PP^{\mathsf{T}}\Sigma)$ for any $P \in O(d, k)$. Note that by Von Neumann's trace inequality, for any two $d \times d$ positive semidefinite matrices A, B, we have:[6]

$$\text{tr}(AB) \leqslant \sum_{i=1}^{d} \lambda_i(A)\lambda_i(B),$$

where $\lambda_i(A)$ lists the eigenvalues of A in decreasing order (similarly for $\lambda_i(B)$). Hence,

$$\text{tr}(PP^{\mathsf{T}}\Sigma) \leqslant \sum_{i=1}^{d} \lambda_i(PP^{\mathsf{T}})\lambda_i(\Sigma) = \sum_{i=1}^{k} \lambda_i(\Sigma).$$

[6]It would be nice to have a proof of this result that uses only elementary linear algebra arguments and does not rely on these type of trace inequalities. Unfortunately I could not come up with one.

The last equality holds since a projection matrix PP^T onto a k-dimensional subspace has precisely k eigenvalues equal to 1 and the remaining $d-k$ eigenvalues equal to 0. Thus, it remains to show that choosing $P = U[:, :k]$ achieves this value. Indeed with this choice of P,

$$\operatorname{tr}(PP^\mathsf{T}\Sigma) = \operatorname{tr}\left[\left(\sum_{i=1}^{k} u_i u_i^\mathsf{T}\right)\left(\sum_{i=1}^{d} \lambda_i u_i u_i^\mathsf{T}\right)\right] = \operatorname{tr}\left(\sum_{i=1}^{k} \lambda_i u_i u_i^\mathsf{T}\right) = \sum_{i=1}^{k} \lambda_i.$$

□

2.2 K-means clustering

2.3 Kernel density estimation

2.4 Energy based models

We now turn our attention towards generative modeling. As before, we have covariates $x \in \mathsf{X}$ and an underlying distribution $p(x)$ over these covariates. (Again, we will typically assume Euclidean structure on X, namely $\mathsf{X} \subseteq \mathbb{R}^d$, although many of the ideas we will discuss transfer over the case when X is countable.) However, we no longer have any labels $y \in \mathsf{Y}$, making this an unsupervised learning setup. The goal then is no longer prediction, but rather sampling. Indeed, given a set of samples $x_1, \ldots, x_n$, the goal is to learn a sampler $\hat{p}_n$ such that samples drawn from $\hat{p}_n$ approximate samples drawn from $p(x)$ as close as possible.

Our first attempt at solving this problem will be using classic energy based models. This is a classic idea dating back to Yann LeCun's earlier work [LeCun et al., 2006]. Suppose that the distribution $p(x)$ over X has a density, and with slight abuse of notation we let $p(x)$ denote this density. Energy based models (EBMs) model the *un-normalized density*, i.e., for a function class $\mathscr{F}$ mapping $\mathsf{X} \mapsto \mathbb{R}$,

$$p(x) \propto \exp(f(x)), \quad f \in \mathscr{F}. \tag{2.1}$$

The important property of this model is that there is no constraint placed on f that enforces normalization, i.e., generically we have that

$$Z[f] := \int \exp(f(x))\,\mathrm{d}x \neq 1.$$

In reference to the statistical physics motivations behind EBMs, the function $Z[f]$ is called the *partition function*. Of course for a given $f \in \mathscr{F}$, the resulting normalized density is hence:

$$p_f(x) := \exp(f(x))/Z[f].$$

Since computing $Z[f]$ is typically not tractable, as it involves high dimensional integration, the exact form p_f is to be thought of as a purely analytical tool. As stated before, the loss function we want to minimize is now the KL-divergence. This motivates the following maximum likelihood (MLE) procedure at the population level:

$$f_\star = \arg\min_{f \in \mathscr{F}} \mathrm{KL}(p \,\|\, p_f) = \arg\min_{f \in \mathscr{F}} \{-H[p] - \mathbb{E}_p[\log p_f]\} = \arg\min_{f \in \mathscr{F}} \{-\mathbb{E}_p[\log p_f]\} =: \arg\min_{f \in \mathscr{F}} L[f]. \tag{2.2}$$

Here, $H[p] := -\mathbb{E}_p[\log p]$ is the (differential) entropy of p. Note the loss $L[f]$ is referred to as the *cross-entropy loss*. Let us now suppose that $\mathscr{F} = \{f_\theta \mid \theta \in \Theta\}$ where Θ is a subset of Euclidean space, and compute what the gradient of the cross-entropy loss is. Slightly overloading notation with $L[\theta] = L[f_\theta]$, $p_\theta = p_{f_\theta}$, and $Z[\theta] = Z[f_\theta]$, a quick computation yields:

$$\begin{aligned}\nabla_\theta L[\theta] &= -\nabla_\theta \int \log p_\theta(x) p(x)\,\mathrm{d}x \\ &\overset{(a)}{=} -\int \nabla_\theta \log p_\theta(x) p(x)\,\mathrm{d}x\end{aligned}$$

$$= -\int \nabla_\theta [f_\theta(x) - \log Z[\theta]] p(x)\,\mathrm{d}x$$
$$= -\mathbb{E}_p[\nabla_\theta f_\theta] + \nabla_\theta \log Z[\theta].$$

In (a), we exchanged the order of differentiation and integration.[7] Next we compute:

$$\nabla_\theta \log Z[\theta] = \frac{1}{Z[\theta]} \nabla_\theta \int \exp(f_\theta(x))\,\mathrm{d}x \overset{(a)}{=} \frac{1}{Z[\theta]} \int \nabla_\theta \exp(f_\theta(x))\,\mathrm{d}x$$
$$= \frac{1}{Z[\theta]} \int \nabla_\theta f_\theta(x) \exp(f_\theta(x))\,\mathrm{d}x = \mathbb{E}_{p_\theta}[\nabla_\theta f_\theta].$$

Again in (a) we exchanged the order of differentiation and integration. Above, the blue color p_θ is used to emphasize that the sampling is done w.r.t. the model p_θ and not the true data generating distribution p. Combining these calculations,

$$\nabla_\theta L[\theta] = -(\mathbb{E}_p[\nabla_\theta f_\theta] - \mathbb{E}_{p_\theta}[\nabla_\theta f_\theta]). \tag{2.3}$$

Thus, we see that to form a stochastic estimate of this gradient, we need to *sample* from the model p_θ. We now turn to how to generate these samples. Note that if X is finite, then sampling is simple, since computing $p_f(x)$ exactly for each $x \in \mathsf{X}$ is feasible. So we focus on the setting when X is not finite. In particular, we will focus on the case when X has Euclidean structure.

2.4.1 Markov Chain Monte Carlo sampling

Here, suppose that $\mathsf{X} \subseteq \mathbb{R}^d$. The standard algorithm for sampling from p_θ is Langevin sampling, which sets up a Markov chain with a stationary distribution equal to p_θ. A classic fact from statistical physics (by way of the Fokker-Planck equation) states that the following Itô stochastic differential equation (SDE), know as *Langevin dynamics*, has its stationary measure equal to p_θ:

$$\mathrm{d}X_t = \nabla_x \log p_\theta(X_t)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t. \tag{2.4}$$

Here, $(B_t)_{t\geqslant 0}$ is standard Brownian motion in $\mathbb{R}^d$. For those unfamiliar with SDEs, we can think of the process $(X_t)_{t\geqslant 0}$ described in Equation (2.4) as the continuous time limit as $\eta \to 0$ of the following discrete-time stochastic process parameterized by $\eta > 0$, which does not require any fancy machinery to describe:[8]

$$X_{t+1} = X_t + \eta \nabla_x \log p_\theta(X_t) + \sqrt{2\eta} W_t. \tag{2.5}$$

Here, $W_t \sim N(0, I)$ is drawn independently for $t \in \mathbb{N}$. Note the appearance of $\log p_\theta$ above is critical, since it allows the partition function, which is not a function of x, to disappear in the gradient:

$$\nabla_x \log p_\theta(x) = \nabla_x [f_\theta(x) - \log Z[\theta]] = \nabla_x f_\theta(x).$$

The object $\nabla_x \log p_\theta(x)$ is an important enough quantity that we give it a particular name: the *score function*. Let us denote X_t from the discrete-time process (2.5) as:

$$X_t = \mathsf{MCMC}_\eta(f_\theta, X_0, t),$$

with the dependence on the random $\{W_t\}$'s made implicit.

[7] Technically, we should check some integrability conditions to ensure this is valid (using Lebesgue's dominated convergence theorem), but we will ignore this technicality and assume this step holds.

[8] This is known as the Euler–Maruyama discretization of the Langevin SDE.

Stochastic gradient estimate for $L[\theta]$. Recall that the population level gradient for the cross-entropy loss $L[\theta] = -\mathbb{E}_p[\log p_\theta]$ is given by $\nabla_\theta L[\theta] = -(\mathbb{E}_p[\nabla_\theta f_\theta] - \mathbb{E}_{p_\theta}[\nabla_\theta f_\theta])$. We now have the necessary notation to write down a stochastic estimator for this gradient. This estimator has two hyperparameters, $\eta > 0$ and $\kappa \in \mathbb{N}_+$. Consider the following estimator

$$\hat{g}_n := -(\nabla_\theta f_\theta(x_i) - \nabla_\theta f_\theta(\mathsf{MCMC}_\eta(f_\theta, 0, \kappa))), \quad i \sim \mathrm{Unif}(\{1, \dots, n\}).$$

This estimator approximately satisfies $\mathbb{E}[\hat{g}_n] \approx \nabla_\theta L[\theta]$, although quantifying the bias of $\hat{g}_n$ is non-trivial and out of scope for this course (we will briefly discuss the sources of error in a moment). Furthermore, it is not necessary to start $X_0 = 0$ in the MCMC; more sophisticated variants use warm-starting techniques (e.g. using the last sample generated) to improve MCMC convergence. See Song and Kingma [2021] for more advanced techniques.

Gaussian example: Langevin sampling. To make Langevin sampling more concrete, let us consider what happens when we apply (2.5) to a Gaussian score function. Of course, for a Gaussian distribution, we do not require Langevin sampling to generate samples, but nevertheless we can calculate what Langevin sampling applied to a Gaussian score function does. Specifically, suppose that $p(x) = N(\mu, \sigma^2 I)$. A quick calculation yields that the score function is:

$$\nabla_x \log p(x) = -\sigma^{-2}(x - \mu).$$

Therefore, the discrete-time Langevin sampling process (2.5) simplifies to:

$$\begin{aligned} x_{t+1} &= x_t + \eta \nabla_x \log p(x) + \sqrt{2\eta} g_t && [g_t \sim N(0, I),\ x_0 = 0] \\ &= (1 - \eta\sigma^{-2}) x_t + \eta\sigma^{-2}\mu + \sqrt{2\eta} g_t. \end{aligned}$$

Unrolling this recursion yields:

$$x_t = \sum_{i=0}^{t-1} (1 - \eta\sigma^{-2})^{t-1-i} [\eta\sigma^{-2}\mu + \sqrt{2\eta} g_i].$$

Immediately, we see that x_t is the result of an affine transformation of Gaussian random variables, and is hence Gaussian. Thus, to characterize the distribution of x_t, it suffices to compute its mean and covariance. Let us assume $\eta < \sigma^2$. Starting with its mean,

$$\mathbb{E}[x_t] = \sum_{i=0}^{t-1} (1 - \eta\sigma^{-2})^i \eta\sigma^{-2}\mu = [1 - (1 - \eta\sigma^{-2})^t]\mu.$$

On the other hand, the covariance is:

$$\mathrm{Cov}(x_t) = 2\eta \sum_{i=0}^{t-1} (1 - \eta\sigma^{-2})^{2i} = 2\eta \frac{1 - (1 - \eta\sigma^{-2})^{2t}}{1 - (1 - \eta\sigma^{-2})^2} = \frac{2\eta[1 - (1 - \eta\sigma^{-2})^{2t}]}{2\eta\sigma^{-2} - (\eta\sigma^{-2})^2}.$$

As $t \to \infty$, the process x_t converges to its stationary measure $N(\mu, \sigma_\eta^2 I)$ with

$$\sigma_\eta^2 := \frac{2\sigma^2}{2 - \eta\sigma^{-2}}.$$

Furthermore, we also have that $\lim_{\eta \to 0} \sigma_\eta^2 = \sigma^2$, and hence we see that $\mathrm{Law}(x_t)$ converges to the correct distribution $N(\mu, \sigma^2 I)$ as both the number of Langevin steps $t \to \infty$, and the Langevin step size $\eta \to 0$. Of course, we see that for finite $t < \infty$ and non-zero $\eta > 0$, the distribution $\mathrm{Law}(x_t)$ is not equal to the target $N(\mu, \sigma^2 I)$, and hence this introduces bias in the sampling.

Gaussian example: EBM learning. Let us instantiate an idealized version of EBM learning for clarity, where we ignore the bias in MCMC sampling. Suppose the true distribution is an isotropic Gaussian with unknown mean $\mu \in \mathbb{R}^d$, i.e., $p(x) = N(\mu, I)$ (we set $\sigma^2 = 1$ here to simplify the following expressions). Suppose furthermore that our energy model class is:

$$\mathscr{F} = \left\{ x \mapsto f_\theta(x) := -\frac{1}{2}\|x - \theta\|^2 \mid \theta \in \mathbb{R}^d \right\}.$$

That is, we model the log density of an isotropic Gaussian with a non-zero mean up to the normalization constant. Now, let us suppose we are in an idealized streaming setting, where $x_0, x_1, \ldots$ are a stream of iid examples from $p(x)$. Following Equation (2.3), we set:

$$\theta_{t+1} = \theta_t + \eta_t(\nabla_\theta f_{\theta_t}(x_t) - \nabla_\theta f_{\theta_t}(\tilde{x}_t)), \quad \tilde{x}_t \sim N(\theta_t, I), \quad \theta_0 = 0.$$

Note in this iteration above, we are using $-(\nabla_\theta f_{\theta_t}(x_t) - \nabla_\theta f_{\theta_t}(\tilde{x}_t))$ as an estimate of the cross-entropy gradient (2.3); this equation assumes the MCMC sampling is exact (i.e., $t \to \infty$ and $\eta \to 0$). Now, using the fact that $\nabla_\theta f_\theta(x) = x - \theta$, $x_t = \mu + g_t$, and $\tilde{x}_t = \theta_t + \tilde{g}_t$, we have that:

$$\begin{aligned}
\theta_{t+1} - \mu &= \theta_t - \mu + \eta_t(x_t - \theta_t - (\tilde{x}_t - \theta_t)) \\
&= \theta_t - \mu + \eta_t(\mu - \theta_t + g_t - \tilde{g}_t) \\
&= (1 - \eta_t)(\theta_t - \mu) + \eta_t \hat{g}_t. && \text{defining } \hat{g}_t := g_t - \tilde{g}_t.
\end{aligned}$$

Letting $\Delta_t := \theta_t - \mu$, we have the identity

$$\Delta_{t+1} = (1 - \eta_t)\Delta_t + \eta_t \hat{g}_t.$$

Unrolling the recursion,

$$\Delta_t = \sum_{i=0}^{t-1} \left[\prod_{j=i+1}^{t-1} (1 - \eta_j) \right] \eta_i \hat{g}_i.$$

Therefore, since all the $\hat{g}_i$'s are mutually independent,

$$\mathbb{E}\|\Delta_t\|^2 = 2 \sum_{i=0}^{t-1} \left[\prod_{j=i+1}^{t-1} (1 - \eta_j) \right]^2 \eta_i^2.$$

Now let us choose $\eta_i = 1/(i+1)$ for all $i \in \mathbb{N}$. With this choice,

$$\prod_{i=k}^{t-1} (1 - \eta_i) = \frac{k}{t}$$

Hence we can compute,

$$\sum_{i=0}^{t-1} \left[\prod_{j=i+1}^{t-1} (1 - \eta_j) \right]^2 \eta_i^2 = \sum_{i=0}^{t-1} \left(\frac{i+1}{t} \right)^2 \frac{1}{(i+1)^2} = \frac{1}{t}.$$

Therefore,

$$\mathbb{E}\|\Delta_t\|^2 = \frac{2}{t}.$$

So we see that the estimator θ_t converges to the true mean μ at a $O(1/t)$ rate.

Divergence, Laplacian, and integration by parts. For what follows, we will make use of the following notation and tools from multivariable calculus. For a vector field $v : \mathbb{R}^d \mapsto \mathbb{R}^d$, the divergence $\nabla \cdot v$ is:

$$(\nabla \cdot v)(x) = \sum_{i=1}^{d} \frac{\partial v_i(x)}{\partial x_i}.$$

Next, for a function $f : \mathbb{R}^d \mapsto \mathbb{R}$, the Laplacian Δf is:

$$(\Delta f)(x) = \sum_{i=1}^{d} \frac{\partial^2 f(x)}{\partial x_i^2}.$$

(Check the following identity that $\nabla \cdot \nabla f = \Delta f$.)

A key technical tool we will use is the classic integration by parts identity, which states that for a smooth function $f : \mathbb{R}^d \mapsto \mathbb{R}$ and smooth vector field $v : \mathbb{R}^d \mapsto \mathbb{R}^d$ whose behavior at infinity vanishes,[9]

$$\int \langle v(x), \nabla f(x) \rangle \, \mathrm{d}x = -\int \nabla \cdot v(x) f(x) \, \mathrm{d}x. \tag{2.6}$$

Stationarity of Langevin dynamics (2.4). To check that p_θ is the stationary measure of the Langevin dynamics (2.4), we will appeal to the Fokker-Planck (FP) equation, which states the following. Suppose we have a SDE of the form:

$$\mathrm{d}X_t = v(X_t) \, \mathrm{d}t + \sqrt{2}\mathrm{d}B_t.$$

Then, the probability density function ρ_t of X_t satisfies the following partial differential equation (PDE):[10]

$$\frac{\partial \rho_t}{\partial t} = -\nabla \cdot (\rho_t v) + \Delta \rho_t. \tag{2.7}$$

Now, plugging (2.4) into the FP equation (2.7), we have:

$$\begin{aligned}
\frac{\partial p_\theta}{\partial t} &= -\nabla \cdot (p_\theta \nabla \log p_\theta) + \Delta p_\theta \\
&= -\nabla \cdot (p_\theta \nabla \log p_\theta) + \nabla \cdot \nabla p_\theta && \text{since } \Delta p_\theta = \nabla \cdot \nabla p_\theta \\
&= -\nabla \cdot (p_\theta \nabla \log p_\theta) + \nabla \cdot (p_\theta \nabla \log p_\theta) && \text{since } p_\theta \nabla \log p_\theta = \nabla p_\theta \\
&= 0.
\end{aligned}$$

Therefore, tells us that the density of p_θ remains invariant over time under the SDE (2.4). In other words, if we start $X_0 \sim p_\theta$, then $\mathrm{Law}(X_t) = p_\theta$ for all $t \geqslant 0$.

Sources of MCMC error. We briefly describe the sources of error which arise in using MCMC to generate samples from p_θ. We first consider the continuous-limit SDE (2.4). Via the Fokker-Planck equation, we previous showed that p_θ is the stationary measure of the Langevin dynamics (2.4). As a consequence of this, we have that:

$$\lim_{t \to \infty} \mathrm{KL}(\mathrm{Law}(X_t) \parallel p_\theta) = 0. \tag{2.8}$$

However, for any *finite* $t > 0$, $\mathrm{KL}(\mathrm{Law}(X_t) \parallel p_\theta)$ is not necessarily zero. Furthermore, the convergence speed of (2.8), without any further assumptions on p_θ, is known to be prohibitively slow in the worst case. This happens in particular when the distribution p_θ is *multi-modal*. As an extreme case, consider a distribution with two modes that have equal probability, which are well-separated. Since both modes are equally likely,

[9] We will not be overly concerned with the technical conditions here.

[10] The fact that p_t has a well-defined density respect to the Lebesgue measure on $\mathbb{R}^d$ is a consequence of the fact that the Brownian motion driving the process $(X_t)_{t \geqslant 0}$ is full dimensional.

the Langevin dynamics should spend equal time traversing between the two modes, in order to accurately reflect the underlying distribution. However, since Langevin dynamics noisily follow the gradient of the log density (cf. Equation (2.4)), traversing across models can take a long time (exponential in the ambient dimension d). Thus, in the worst case, convergence of Langevin dynamics to the stationary distribution p_θ requires running the SDE for very large t. This presents one source of error.

The next source of error involves the relationship between the continuous-limit SDE (2.4) and the discrete-time stochastic process (2.5). To avoid confusion, let $\{Z_t\}_{t\in\mathbb{N}}$ denote the discrete-time process (2.5). Conceptually, we can think of there being a natural correspondence between $Z_k \leftrightarrow X_{k\eta}$. However, the discrepancy between Z_k and $X_{k\eta}$ grows as η increases; for any finite $\eta > 0$ there is always discretization error. This presents another source of error.

2.4.2 Convergence of Langevin dynamics in Fisher information

Despite our previous discussion on the convergence of Langevin dynamics, it turns out we can prove a rather general positive result for a particular divergence measure. This result will give us some intuition for the type of resulting distributions which we can reasonably expect Langevin dynamics to converge to. This section is motivated by the results in Balasubramanian et al. [2022].

We define the following divergence over smooth measures (that is, measures with differentiable densities). For two smooth measures μ, ν, the *Fisher information* of μ with respect to ν is defined as:

$$\mathrm{FI}(\mu \parallel \nu) := \mathbb{E}_\mu \|\nabla \log(\mu/\nu)\|^2. \tag{2.9}$$

We consider the following Langevin dynamics (here p is an arbitrary smooth density),

$$\mathrm{d}X_t = \nabla \log p(X_t)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t.$$

Recall that p is the stationary measure of the process $(X_t)_{t\geqslant 0}$. Let $p_t = \mathrm{Law}(X_t)$ denote both the measure and density of the Langevin dynamics at time t. It turns out that under the Fisher information divergence, we have that $\mathrm{FI}(p_t \parallel p) \to 0$ almost immediately.

The key identity is that the negative Fisher information is the time derivative of the KL-divergence $\mathrm{KL}(p_t \parallel p)$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(p_t \parallel p) = -\mathrm{FI}(p_t \parallel p). \tag{2.10}$$

We will prove this fundamental identity in a moment. But taking it for granted, we first see that (2.10) confirms the stationarity of p, since if we set $p_t = p$, then $\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(p_t \parallel p) = 0$. Furthermore, by the fundamental theorem of calculus, we conclude from (2.10) that for any $T > 0$,

$$\mathrm{KL}(p_T \parallel p) = \mathrm{KL}(p_0 \parallel p) - \int_0^T \mathrm{FI}(p_t \parallel p)\,\mathrm{d}t.$$

Since the Fisher information is always non-negative, we immediately see a nice property of Langevin dynamics, which is that $\mathrm{KL}(p_T \parallel p) \leqslant \mathrm{KL}(p_0 \parallel p)$ (the KL-divergence never increases as a result of running Langevin). Furthermore, since $\mathrm{KL}(p_T \parallel p) \geqslant 0$, re-arranging the above display and dividing by T yields:

$$\frac{1}{T}\int_0^T \mathrm{FI}(p_t \parallel p)\,\mathrm{d}t \leqslant \frac{\mathrm{KL}(p_0 \parallel p)}{T}.$$

Now consider the mixture measure $\bar{p}_T := \frac{1}{T}\int_0^T p_t\,dt$. By the convexity of $\mu \mapsto \mathrm{FI}(\mu \parallel \nu)$,[11] Jensen's inequality yields that:

$$\mathrm{FI}(\bar{p}_T \parallel p) \leqslant \frac{1}{T}\int_0^T \mathrm{FI}(p_t \parallel p)\,\mathrm{d}t \leqslant \frac{\mathrm{KL}(p_0 \parallel p)}{T}.$$

[11] This follows by first writing $\mathrm{FI}(\mu \parallel \nu) = \int \frac{\|\nabla\rho\|^2}{\rho}\,\mathrm{d}\nu$ for $\rho = \mathrm{d}\mu/\mathrm{d}\nu$, and then using the convexity of $(x, y) \mapsto \|x\|^2/y$ on $\mathbb{R}^d \times \mathbb{R}_{>0}$.

Note that this mixture measure $\bar{p}_T$ can be sampled from by sampling $t \sim \mathrm{Unif}([0,T])$, and outputting X_t. Hence, we see that if we set $T \geqslant \mathrm{KL}(p_0 \parallel p)/\varepsilon$, then $\mathrm{FI}(\bar{p}_T \parallel p) \leqslant \varepsilon$.

The question now becomes, what exactly does this guarantee of $\mathrm{FI}(\bar{p}_T \parallel p) = O(1/T)$ mean. To shed some insight, Balasubramanian et al. [2022] constructed an example of two distributions, parameterized by an scalar parameter m, such that the TV-distance is bounded away from zero uniformly for all m, but the Fisher information tends to zero. The example is:

$$\begin{aligned}\pi_m &= \frac{1}{2}N(-m,1) + \frac{1}{2}N(m,1),\\ \mu_m &= \frac{3}{4}N(-m,1) + \frac{1}{4}N(m,1).\end{aligned}$$

Balasubramanian et al. [2022, Proposition 1] shows that:

$$\inf_{m \geqslant 1/80} \|\mu_m - \pi_m\|_{\mathrm{tv}} \geqslant \frac{1}{800}, \quad \mathrm{FI}(\mu_m \parallel \pi_m) \leqslant 4m^2 \exp(-m^2/2).$$

The intuition here is that, for the Fisher information to be small, it suffices to get the local structure correct (i.e., the $N(\pm m, 1)$ modes), but the global weights can be incorrect. On the other hand, TV-distance requires that the global weights are correct (in addition to the local structure).

Proof of Equation (2.10). We now finish this section with a proof the time derivative identity (2.10). The key is to use the Fokker-Planck equation (2.7) in conjunction with integration by parts:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(p_t \parallel p) &= \partial_t \int p_t \log \frac{p_t}{p}\,\mathrm{d}x\\ &= \int \partial_t \left(p_t \log \frac{p_t}{p}\right)\mathrm{d}x\\ &= \int (\partial_t p_t)\log\frac{p_t}{p}\,\mathrm{d}x + \int p_t \cdot \partial_t \log p_t\,\mathrm{d}x\\ &\overset{(a)}{=} \int (\partial_t p_t)\log\frac{p_t}{p}\,\mathrm{d}x\\ &\overset{(b)}{=} \int \nabla\cdot\left(p_t \nabla \log\frac{p_t}{p}\right)\log\frac{p_t}{p}\,\mathrm{d}x && \text{Fokker-Planck equation}\\ &= -\int \left\|\nabla \log\frac{p_t}{p}\right\|^2 p_t\,\mathrm{d}x && \text{integration by parts}\\ &= -\mathbb{E}_{p_t}\left\|\nabla\log\frac{p_t}{p}\right\|^2\\ &= -\mathrm{FI}(p_t \parallel p). \end{aligned} \tag{2.11}$$

To see (a) above, note that:

$$\int p_t \cdot \partial_t \log p_t\,\mathrm{d}x = \int \partial_t p_t\,\mathrm{d}x = \partial_t \int p_t\,\mathrm{d}x = 0.$$

To see how (b) arises from the Fokker-Planck equation, from (2.7),

$$\begin{aligned}\partial_t p_t &= -\nabla\cdot(p_t \nabla \log p) + \Delta p_t\\ &= -\nabla\cdot[p_t\nabla\log p - \nabla p_t] && \text{since } \Delta p_t = \nabla\cdot\nabla p_t\\ &= -\nabla\cdot(p_t\nabla\log p - p_t\nabla\log p_t)\\ &= \nabla\cdot\left(p_t\nabla\log\frac{p_t}{p}\right).\end{aligned}$$

2.4.3 Exponential convergence of Langevin dynamics under log-Sobolev inequalities

Note: this is an advanced topic we will not discuss in the course, but it is included for readers who are interested in diving further into this topic.

In the previous section, we saw that with no extra assumptions on p, we could show that Langevin dynamics drives the Fisher information to zero at a $O(1/T)$ rate. However, we also saw that having small Fisher information does not necessarily mean the global structure of the distribution is preserve. In this section, we introduce an extra assumption on the distribution p to ensure that we have convergence of Langevin in KL-divergence, and furthermore at an exponetial rate.

Equation (2.10) gives us a hint as to what condition to enforce on p_t and p to ensure convergence. In particular, suppose we are assured that for some $\alpha > 0$ and all $t \geqslant 0$,

$$\mathrm{FI}(p_t \parallel p) = \mathbb{E}_{p_t}\left\|\nabla \log \frac{p_t}{p}\right\|^2 \geqslant \alpha \cdot \mathrm{KL}(p_t \parallel p). \tag{2.12}$$

Then, combining inequality (2.12) with (2.10),

$$\forall t \geqslant 0,\ \frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(p_t \parallel p) \leqslant -\alpha \cdot \mathrm{KL}(p_t \parallel p),$$

and therefore by the comparison lemma for ODEs,

$$\forall t \geqslant 0,\ \mathrm{KL}(p_t \parallel p) \leqslant e^{-\alpha t} \cdot \mathrm{KL}(p_0 \parallel p).$$

Now the question remains, what assumption on p_t and p ensures the condition (2.12) holds? It turns out that this condition is ensured by a standard functional inequality in probability theory known as the *log-Sobolev* inequality. Log-Sobolev inequalities have a rich history in probability theory which we will not have time to discuss. We will simply state its definition and see how it implies (2.12). The first definition we need is that of *concentration entropy* (not to be confused with the information theoretic notion of entropy). For a measure μ and a non-negative random variable X, the concentration entropy $\mathrm{Ent}_\mu(X)$ (or entropy for short) of X is defined as:

$$\mathrm{Ent}_\mu(X) := \mathbb{E}_\mu[X \log X] - \mathbb{E}_\mu[X] \cdot \log \mathbb{E}_\mu[X].$$

A measure μ satisfies the *log-Sobolev inequality* with constant $C > 0$ if for all smooth functions $f : \mathsf{X} \mapsto \mathbb{R}$,

$$\mathrm{Ent}_\mu(f^2) \leqslant 2C \cdot \mathbb{E}_\mu\|\nabla f\|^2. \tag{2.13}$$

While this definition is seemingly arbitrary, it turns out that it directly implies (2.12). To see this, first let us compute the entropy of the function $f = \sqrt{p_t/p}$ under the measure p:

$$\mathrm{Ent}_p\left(\frac{p_t}{p}\right) = \mathbb{E}_p\left[\frac{p_t}{p}\log\frac{p_t}{p}\right] - \mathbb{E}_p\left[\frac{p_t}{p}\right]\log\mathbb{E}_p\left[\frac{p_t}{p}\right] = \mathbb{E}_p\left[\frac{p_t}{p}\log\frac{p_t}{p}\right] = \mathrm{KL}(p_t \parallel p).$$

Therefore, if we suppose that the stationary measure p satisfies the log-Sobolev inequality (LSI) with constant C, then applying the LSI inequality to $f = \sqrt{p_t/p}$,

$$\mathrm{Ent}_p\left(\frac{p_t}{p}\right) = \mathrm{KL}(p_t \parallel p) \leqslant 2C \cdot \mathbb{E}_p\left\|\nabla\sqrt{\frac{p_t}{p}}\right\|^2 = \frac{C}{2} \cdot \mathbb{E}_{p_t}\left\|\nabla \log \frac{p_t}{p}\right\|^2.$$

Hence, this implies (2.12) with $\alpha = 2/C$. That is, we have shown that if p satisfies the LSI inequality with constant C, then for all $t > 0$,

$$\mathrm{KL}(p_t \parallel p) \leqslant e^{-2t/C} \cdot \mathrm{KL}(p_0 \parallel p).$$

That is, if we want to ensure that $\mathrm{KL}(p_t \parallel p) \leqslant \varepsilon$, it suffices to ensure that $t \geqslant \frac{C}{2}\log\left(\frac{\mathrm{KL}(p_0 \parallel p)}{\varepsilon}\right)$.

Which distributions satisfy log-Sobolev inequalities? In general, verifying the functional inequality (2.13) for a given distribution p is not a simple task. There is, however, a very clean sufficient condition, known as the Bakry-Émery criterion. Suppose that $p = \exp(-U)$, and suppose that U is α strongly-convex. Then p satisfies the LSI inequality with constant $C = 1/\alpha$. The proof of this result is non-trivial and out of the scope of this course: see e.g., Bakry et al. [2014] for a proof. Distributions of this form are known as log-concave distributions, and include e.g., Gaussian distributions.

2.5 Score matching

The MCMC sampling dynamics (2.5) motivate an alternative loss which does not rely on the the cross-entropy loss (2.2). Indeed, the idea is to *directly* learn the score function $\nabla_x \log p(x)$. It turns out that we can setup a very simple loss function, called the score matching loss, which bypasses the requirement to sample from the current model in order to implement the gradient of the cross-entropy loss (cf. Equation (2.3)).

Proposition 2.2 (Score matching loss)**.** *Let $p(x)$ denote a distribution over $\mathsf{X} \subset \mathbb{R}^d$, and suppose its density is sufficiently smooth. For any smooth vector field $f : \mathbb{R}^d \mapsto \mathbb{R}^d$ vanishing at infinity,*

$$\mathbb{E}_p \|\nabla \log p - f\|^2 = \mathbb{E}_p \left[\|f\|^2 + 2\nabla \cdot f + \|\nabla \log p\|^2 \right].$$

Proof. First, we use the identity integration by parts to derive,

$$\begin{aligned}\mathbb{E}_p[\langle \nabla \log p, f \rangle] &= \int \langle \nabla \log p(x), f(x) \rangle p(x) \, \mathrm{d}x \\ &= \int \langle \nabla p(x), f(x) \rangle \, \mathrm{d}x = -\int \nabla \cdot f(x) p(x) \, \mathrm{d}x = -\mathbb{E}_p[\nabla \cdot f].\end{aligned}$$

We now complete the proof as follows:

$$\mathbb{E}_p \|\nabla \log p - f\|^2 = \mathbb{E}_p \left[\|f\|^2 - 2\langle \nabla \log p, f \rangle + \|\nabla \log p\|^2 \right] = \mathbb{E}_p \left[\|f\|^2 + 2\nabla \cdot f + \|\nabla \log p\|^2 \right].$$

□

An immediate consequence of this result is that:

$$\arg\min_{f \in \mathscr{F}} \mathbb{E}_p \|\nabla \log p - f\|^2 = \arg\min_{f \in \mathscr{F}} L[f] := \mathbb{E}_p \left[\|f\|^2 + 2\nabla \cdot f \right].$$

The loss $L[f]$ is called the (population) score matching loss. Given samples $x_1, \ldots, x_n \sim p(x)$ iid, we can form the empirical score matching loss:

$$\hat{L}_n[f] := \frac{1}{n} \sum_{i=1}^{n} \left[\|f(x_i)\|^2 + 2\nabla \cdot f(x_i) \right]. \tag{2.14}$$

Note that unlike the cross-entropy loss, computing gradients of the score matching loss is straightforward given a reasonable auto-differentiation library.

Curl-free vector fields. One issue which arises from directly representing the vector field corresponding to the score function, is that in general it is difficult to directly ensure that a vector field is *curl-free*, meaning that the Jacobian of the vector field is a symmetric matrix. However, we do know that the true score function has this property, since

$$\frac{\mathrm{d}\nabla \log p(x)}{\mathrm{d}x} = \nabla^2 \log p(x),$$

and Hessian matrices are symmetric. One simple way to enforce this symmetry condition is to indirectly parameterize a vector field through the gradient of a scalar potential, e.g. using a function class of the form:

$$\mathscr{F} = \{ x \mapsto \nabla \phi(x) \mid \phi : \mathsf{X} \mapsto \mathbb{R}, \ \phi \in \mathscr{G} \}, \tag{2.15}$$

where $\mathscr{G}$ is a function class of smooth potential functions (functions mapping $\mathsf{X} \mapsto \mathbb{R}$). Note that this parameterization also comes with the advantage that it directly searches for an energy model (2.1), showing that we can actually think of the score matching loss as an alternative loss for learning an energy model (e.g., score matching and EBMs are not incompatible).

While the parameterization (2.15) does enforce the curl-free condition, it is not often used in practice since it comes with additional computational overhead. Specifically, we see that the empirical loss $\hat{L}_n[f]$ (cf. Equation (2.14)) now involves the Laplacian of ϕ. Hence, taking the gradient of $\hat{L}_n[f]$ requires taking *three* derivatives, which can become a computational bottleneck. Therefore in practice the lack of curl-free vector fields is often simply ignored, and it does not seem to result in any performance loss.

2.5.1 Score matching with Gaussians

Let us now work through a specific example of score matching for intuition, instantiating it on Gaussian distributions.

Isotropic Gaussians. Suppose that $p(x) = N(\mu, I)$, where $\mu \in \mathbb{R}^d$ is unknown (but the covariance I is known). A quick calculation yields that the score function of $p(x)$ is the following affine function:

$$\nabla \log p(x) = -(x - \mu).$$

Suppose our family of score functions $\mathscr{F}$ is the set of Gaussian score functions corresponding to different means, e.g.,

$$\mathscr{F} = \{x \mapsto -(x - \theta) \mid \theta \in \mathbb{R}^d\}.$$

Then we have that:

$$L[\theta] = \mathbb{E}_{x \sim N(\mu, I)} \left[\|x - \theta\|^2 - 2\Delta \cdot (x - \theta)\right] = \mathbb{E}_{x \sim N(\mu, I)} \|x - \theta\|^2 - 2d = \|\theta - \mu\|^2 - d.$$

Clearly at the population level, $\arg\min_{\theta \in \mathbb{R}^d} L[\theta] = \mu$, which is correct. Now at the empirical level, we have:

$$\hat{L}_n[\theta] = \frac{1}{n} \sum_{i=1}^{n} \|x_i - \theta\|^2 - 2d.$$

Therefore, the empirical risk minimizer of $\hat{L}_n[\theta]$ is the sample mean (check this!):

$$\hat{\theta}_n = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

General multivariate Gaussians. Now let us suppose that $p(x) = N(\mu, \Sigma)$, where Σ is a positive definite matrix. Here, we assume both μ and Σ are unknown. The true score function of $p(x)$ now takes the form:

$$\nabla \log p(x) = -\Sigma^{-1}(x - \mu).$$

Consider now using the family of score functions $\mathscr{F}$ as:

$$\mathscr{F} = \{x \mapsto -G(x - \theta) \mid \theta \in \mathbb{R}^d,\ G \in \mathbb{R}^{d \times d}\}.$$

Note that even though G needs to be symmetric and positive definite to be correct, we will not constrain it to be symmetric positive definite in our formulation. A quick computation yields that the divergence of an $f \in \mathscr{F}$:

$$\nabla \cdot (-G(x - \theta)) = -\nabla \cdot (Gx) = -\operatorname{tr}(G).$$

So the empirical score matching loss is:

$$\hat{L}_n[\theta, G] = \frac{1}{n} \sum_{i=1}^{n} \|G(x_i - \theta)\|^2 - 2\operatorname{tr}(G).$$

Let us look at the stationary points of this objective $\hat{L}_n[\theta, G]$. Computing the gradients,

$$\nabla_\theta \hat{L}_n[\theta, G] = \frac{2}{n}\sum_{i=1}^n G^\mathsf{T} G(x_i - \theta) = 2G^\mathsf{T} G \cdot \left[\frac{1}{n}\sum_{i=1}^n x_i - \theta\right],$$
$$\nabla_G \hat{L}_n[\theta, G] = \frac{2}{n}\sum_{i=1}^n G(x_i - \theta)(x_i - \theta)^\mathsf{T} - 2I = 2G \cdot \left[\frac{1}{n}\sum_{i=1}^n (x_i - \theta)(x_i - \theta)^\mathsf{T}\right] - 2I.$$

Thus, we see that the following pair $(\hat{\theta}_n, \hat{G}_n)$ is a stationary point, assuming the inverse is well defined,

$$\hat{\theta}_n = \frac{1}{n}\sum_{i=1}^n x_i, \quad \hat{G}_n = \left[\frac{1}{n}\sum_{i=1}^n (x_i - \hat{\theta}_n)(x_i - \hat{\theta}_n)^\mathsf{T}\right]^{-1}.$$

This essentially reduces to a "plug-in estimate" of the mean and covariance of the data points $x_1, \ldots, x_n$.

2.5.2 Score matching with exponential families

We can generalize beyond the Gaussian example we saw previously. Suppose that $p(x)$ has the form of an exponential family:

$$p(x) = \exp(\langle \phi(x), \theta_\star \rangle)/Z[\theta_\star], \quad Z[\theta] = \int \exp(\langle \phi(x), \theta \rangle)\, \mathrm{d}x, \quad \theta \in \Theta \subseteq \mathbb{R}^p.$$

Here, we assume the sufficient statistic $\phi : \mathsf{X} \mapsto \mathbb{R}^p$ is known. Let the component-wise functions of ϕ be denoted as $\phi = (\phi_1, \ldots, \phi_p)$, and let $J_\phi(x) \in \mathbb{R}^{p \times d}$ denote the Jacobian of ϕ, i.e., $J_\phi = \frac{\mathrm{d}\phi}{\mathrm{d}x}$. A quick calculation yields:

$$\nabla \log p(x) = J_\phi^\mathsf{T}(x)\theta_\star.$$

As we did in the realizable case, let us consider the following function class:

$$\mathscr{F} = \{x \mapsto J_\phi^\mathsf{T}(x)\theta \mid \theta \in \Theta\}.$$

The score matching loss has the following least-squares form given as follows:

$$L[\theta] = \mathbb{E}_p\left[\|J_\phi^\mathsf{T}(x)\theta\|^2 + 2\langle h_\phi(x), \theta \rangle\right], \quad h_\phi(x) = (\Delta\phi_1(x), \ldots, \Delta\phi_p(x)).$$

To see this, we observe that $J_\phi^\mathsf{T}(x)\theta = \sum_{i=1}^p \nabla\phi_i(x)\theta_i$, and therefore:

$$\nabla \cdot J_\phi^\mathsf{T}(x)\theta = \sum_{i=1}^p \nabla \cdot \nabla\phi_i(x)\theta_i = \sum_{i=1}^p \Delta\phi_i(x)\theta_i = \langle h_\phi(x), \theta \rangle.$$

Hence the empirical loss $\hat{L}_n[\theta]$ has the following form:

$$\hat{L}_n[\theta] = \frac{1}{n}\sum_{i=1}^n \left[\|J_\phi^\mathsf{T}(x_i)\theta\|^2 + 2\langle h_\phi(x_i), \theta \rangle\right].$$

This is a least-squares problem with one solution given by:

$$\hat{\theta}_n = -\left(\sum_{i=1}^n J_\phi(x_i) J_\phi^\mathsf{T}(x_i)\right)^\dagger \left(\sum_{i=1}^n h_\phi(x_i)\right).$$

Here, $(\cdot)^\dagger$ denotes the pseudo-inverse of a matrix.

2.5.3 Sliced score matching

Despite the empirical score matching loss $\hat{L}_n[f]$ not requiring MCMC sampling to differentiate, computing the divergence of f does introduce some computational overhead. However, we can utilize randomized estimators to mitigate this issue. These ideas fall under the family of sliced score matching.

Let $f : \mathbb{R}^d \mapsto \mathbb{R}^d$ be a smooth function, and let $\partial f(x) \in \mathbb{R}^{d\times d}$ denote its Jacobian matrix evaluated at x. The first observation is the following identity which holds for every x:

$$\mathbb{E}_v \langle v, \partial f(x) \cdot v\rangle = \nabla \cdot f(x), \quad v \sim N(0, I).$$

To see this, observe that:

$$\mathbb{E}_v \langle v, \partial f(x) \cdot v\rangle = \mathbb{E}_v \operatorname{tr}(v^\mathsf{T} \partial f(x) \cdot v) = \mathbb{E}_v \operatorname{tr}(\partial f(x) \cdot vv^\mathsf{T}) = \operatorname{tr}(\partial f(x)) = \sum_{i=1}^d \frac{\partial f_i}{\partial x_i}(x) = \nabla \cdot f(x).$$

Thus, we can replace the empirical score matching loss with a randomized variant, while still preserving the expectation:

$$\hat{L}_{n,1}[f] := \frac{1}{n}\sum_{i=1}^n [\|f(x_i)\|^2 + 2\langle v_i, \partial f(x_i) \cdot v_i\rangle], \quad v_i \sim N(0, I).$$

The computational advantages of this loss over $\hat{L}_n[f]$ are as follows. In order to compute $\nabla \cdot f(x)$, one has to make d Jacobian vector product (JVP) calls. However, in order to compute $\langle v, \partial f(x_i) \cdot v\rangle$, only one JVP call is needed. Hence, this sliced score matching loss offers a factor of d savings in computation, which is non-trivial when d is large. Furthermore, we can actually create a family of losses $\hat{L}_{n,k}$:

$$\hat{L}_{n,k}[f] := \frac{1}{n}\sum_{i=1}^n \left[\|f(x_i)\|^2 + 2\sum_{j=1}^k \langle v_i^{(j)}, \partial f(x_i) \cdot v_i^{(j)}\rangle\right], \quad v_i^{(j)} \sim N(0, I).$$

Choosing $k \in \{1, \ldots, d\}$ allows us to trade off computation for variance reduction.

Exercise 2.3. Sliced score matching takes advantage of the fact that for a fixed matrix $M \in \mathbb{R}^{d\times d}$, the following identity holds:

$$\mathbb{E}_v \langle v, Mv\rangle = \operatorname{tr}(M), \quad v \sim N(0, I).$$

Therefore, $\langle v, Mv\rangle$ can be used as an unbiased estimate of the trace of M. Come with a few other non-Gaussian distributions such that the same identity holds, and run some computer simulations to compare the relative performance of the $\langle v, Mv\rangle$ eostimator using different distributions.

2.5.4 Disadvantages of score matching

While the score matching loss is an improvement over the cross-entropy loss, there are a few disadvantages:

(a) The divergence calculation in the score matching loss (2.14) adds non-negligible computational overhead. Indeed, computing a gradient of $\hat{L}_n[f]$ requires at least two derivatives (three if using the scalar potential parameterization (2.15)). While the score matching loss is a clear improvement over the cross-entropy loss, there is still some computational burden.

(b) Score matching requires that the underlying distribution $p(x)$ to be learned is absolutely continuous w.r.t. the Lebesgue measure. This means that $p(x)$ is not allowed to be supported on a lower dimensional manifold. However, when the event space X corresponds to say, RGB images, this may not be a realistic assumption. Note that this disadvantage is also present with EBMs, which also starts by postulating the existence of a density on X.

(c) Despite the improve score matching loss over the cross-entropy loss, sampling from a learned score function still involves MCMC sampling (cf. Section 2.4.1), and therefore comes with all the disadvantages of MCMC sampling.

Our goal in the subsequent sections will be to address these concerns. Foreshadowing a bit, it turns out denoising diffusion models actually address all these shortcomings (quite cleverly actually). However, in order to fully appreciate the ingenuity behind diffusion models, we will proceed more or less in chronological order and consider the solutions which came before in the literature (which all contain fundamental ideas in generative modeling which are still extremely relevant).

2.6 Denoising score matching

One important step towards addressing the issues brought up in Section 2.5.4 regarding score matching is the idea of *denoising score matching*, which we will cover here. The idea is quite straightforward and intuitive. Regardless of what properties the distribution $p(x)$, the noised distribution $p_\sigma = p \star N(0, \sigma^2 I)$, where $\star$ denotes convolution, has the following properties:

(a) For any $\sigma > 0$, p_σ is fully supported on $\mathbb{R}^d$, meaning that it has a well-defined density function on $\mathbb{R}^d$. This holds even if $p(x)$ does not.

(b) As long as σ is not too large, then p and p_σ should not be too different in some sense.

Given these observations, denoising score matching simply applies score matching to p_σ instead of p. However, because of the special structure of p_σ, the resulting loss function simplifies even more compared to the vanilla score matching loss. To see this, we let $q_\sigma(\tilde{x} \mid x)$ denote the density of the distribution $N(x, \sigma^2 I)$, so that we can write the density of p_σ as:

$$p_\sigma(\tilde{x}) = \int q_\sigma(\tilde{x} \mid x) p(x) \, \mathrm{d}x = \mathbb{E}_{x \sim p}[q_\sigma(\tilde{x} \mid x)].$$

With this notation, letting $(x, \tilde{x}) \sim p \times q_\sigma$ denote sampling from the joint distribution with $x \sim p(x)$ and $\tilde{x} \sim q_\sigma(\tilde{x} \mid x)$, observe that for any $f : \mathsf{X} \mapsto \mathbb{R}$,

$$\begin{aligned}
\mathbb{E}_{p_\sigma} \langle f, \nabla \log p_\sigma \rangle &= \int \langle f(\tilde{x}), \nabla \log p_\sigma(\tilde{x}) \rangle p_\sigma(\tilde{x}) \, \mathrm{d}\tilde{x} \\
&= \int \langle f(\tilde{x}), \nabla p_\sigma(\tilde{x}) \rangle \, \mathrm{d}\tilde{x} \\
&= \int \langle f(\tilde{x}), \nabla \mathbb{E}_{x \sim p}[q_\sigma(\tilde{x} \mid x)] \rangle \, \mathrm{d}\tilde{x} \\
&= \int \langle f(\tilde{x}), \mathbb{E}_{x \sim p}[\nabla q_\sigma(\tilde{x} \mid x)] \rangle \, \mathrm{d}\tilde{x} \\
&= \mathbb{E}_{x \sim p} \int \langle f(\tilde{x}), \nabla q_\sigma(\tilde{x} \mid x) \rangle \, \mathrm{d}\tilde{x} \\
&= \mathbb{E}_{x \sim p} \int \langle f(\tilde{x}), \nabla \log q_\sigma(\tilde{x} \mid x) \rangle q_\sigma(\tilde{x} \mid x) \, \mathrm{d}\tilde{x} \\
&= \mathbb{E}_{(x, \tilde{x}) \sim p \times q_\sigma} \langle f(\tilde{x}), \nabla \log q_\sigma(\tilde{x} \mid x) \rangle. \qquad (2.16)
\end{aligned}$$

With this calculation, letting $(x, \tilde{x}) \sim p \times q_\sigma$, the score matching loss on p_σ simplifies to:

$$\begin{aligned}
\mathbb{E}_{p_\sigma} \| f - \nabla \log p_\sigma \|^2 &= \mathbb{E}_{p_\sigma}[\|f\|^2 + \|\nabla \log p_\sigma\|^2] - 2\mathbb{E}_{p_\sigma} \langle f, \nabla \log p_\sigma \rangle \\
&= \mathbb{E}_{p_\sigma}[\|f\|^2 + \|\nabla \log p_\sigma\|^2] - 2\mathbb{E}_{(x, \tilde{x})} \langle f(\tilde{x}), \nabla \log q_\sigma(\tilde{x} \mid x) \rangle \qquad \text{using (2.16)} \\
&= \mathbb{E}_{p_\sigma}[\|f\|^2 + \|\nabla \log p_\sigma\|^2] - 2\mathbb{E}_{(x, \tilde{x})} \langle f(\tilde{x}), \nabla \log q_\sigma(\tilde{x} \mid x) \rangle \\
&\quad + \mathbb{E}_{(x, \tilde{x})}[\|\nabla \log q_\sigma(\tilde{x} \mid x)\|^2 - \|\nabla \log q_\sigma(\tilde{x} \mid x)\|^2]
\end{aligned}$$

$$= \mathbb{E}_{(x,\tilde{x})}\|f(\tilde{x}) - \nabla \log q_\sigma(\tilde{x} \mid x)\|^2 \qquad \text{Law}(\tilde{x}) = p_\sigma$$
$$+ \mathbb{E}_{(x,\tilde{x})}[\|\nabla \log p_\sigma(\tilde{x})\|^2 - \|\nabla \log q_\sigma(\tilde{x} \mid x)\|^2].$$

Hence for any function class $\mathscr{F}$,

$$\arg\min_{f\in\mathscr{F}} \mathbb{E}_{p_\sigma}\|f - \nabla \log p_\theta\|^2 = \arg\min_{f\in\mathscr{F}} L[f] := \mathbb{E}_{(x,\tilde{x})}\|f(\tilde{x}) - \nabla \log q_\sigma(\tilde{x} \mid x)\|^2. \tag{2.17}$$

The loss $L[f]$ is known as the *denoising loss*. Its empirical counterpart $\hat{L}_n[f]$ can be constructed by taking $\{x_i\}_{i=1}^n$ drawn i.i.d. from $p(x)$, taking $\{w_i\}_{i=1}^n$ drawn i.i.d. from $N(0, I)$ (and independent of the x_i's), and then setting:

$$\hat{L}_n[f] = \frac{1}{n}\sum_{i=1}^n \|f(x_i + \sigma w_i) - \nabla \log q_\sigma(x_i + \sigma w_i \mid x_i)\|^2.$$

Observe that the denoising loss resolves disadvantages (a) and (b) described in Section 2.5.4. Indeed, (a) is resolved since the denoising loss is a standard least-squares loss; no divergence calculations are needed anymore. Furthermore, (b) is resolved since adding Gaussian noise makes p_σ full dimensional even with p is not.

Actually, we can create a family of losses $\hat{L}_{n,k}[f]$ by drawing more noisy samples $\tilde{x}$. That is, letting $\{w_i^{(j)}\}_{i=1,j=1}^{n,k}$ be i.i.d. from $N(0, I)$, let

$$\hat{L}_{n,k}[f] = \frac{1}{nk}\sum_{i=1}^n\sum_{j=1}^k \|f(x_i + \sigma w_i^{(j)}) - \nabla \log q_\sigma(x_i + \sigma w_i^{(j)} \mid x_i)\|^2.$$

Let us see where the term "denoising" comes from. A quick calculation yields that (recall from the derivation that the gradient below is with respect to $\tilde{x}$):

$$\nabla \log q_\sigma(\tilde{x} \mid x) = -\frac{1}{\sigma^2}(\tilde{x} - x),$$

which is proportional to the (negative) noise vector added to make the noised sample $\tilde{x}$ from the pure sample x. Thus, we can interpret the denoising loss as trying to, given a noised sample $\tilde{x}$, predict the direction $f(\tilde{x})$ such that $\tilde{x} + \sigma^2 f(\tilde{x}) \approx x$ is a denoised version of $\tilde{x}$.

Exercise 2.4. Use the identity established in Equation (2.17) to prove *Tweedie's formula*:

$$\nabla \log p_\sigma(\tilde{x}) = \frac{\mathbb{E}[x \mid \tilde{x}] - \tilde{x}}{\sigma^2}.$$

To be clear, here $\mathbb{E}[x \mid \tilde{x}]$ is understood to be the function $\tilde{x} \mapsto \mathbb{E}[X \mid \tilde{X} = \tilde{x}]$ with $(X, \tilde{X}) \sim p \times q_\sigma$.

2.7 Latent models and the Expectation Maximization Algorithm

We now turn to the problem of latent model learning. A latent variable model is a probabilistic model of the following form:

$$z \sim p(z), \qquad \text{(Latent)}$$
$$x \sim p(x \mid z). \qquad \text{(Observation)}$$

The generating process samples latents $z_1, \dots z_n$ iid from $p(z)$, and then each observed x_i is sampled independently from $p(x_i \mid z_i)$.

Now suppose we have a family of probability models:

$$\mathscr{F} = \{p_\theta(x, z) \mid \theta \in \Theta\}.$$

Let us compute the log likelihood $\log p_\theta(x_{1:n})$ for n samples $x_1, \ldots, x_n$ (remember, we do not have access to the latents $z_1, \ldots, z_n$ which generated the data). Specifically, for any $\theta \in \Theta$,

$$\log p_\theta(x_{1:n}) = \sum_{i=1}^n \log p_\theta(x_i) = \sum_{i=1}^n \log \left(\int p_\theta(x_i \mid z_i) p_\theta(z_i) \, \mathrm{d}z_i \right).$$

Now, in the case when $z \in \mathsf{Z}$ is continuous, then the log likelihood requires non-trivial integration. On the other hand, when the latent space Z is discrete, it is possible to run a standard gradient based algorithm (assuming that both $p_\theta(x \mid z)$ and $p_\theta(z)$ can be easily computed). Nevertheless, we will explore an alternative algorithm which takes advantage of the latent structure.

We now state the Expectation Maximization (EM) algorithm for learning latent models, which provides an alternative algorithm to direct gradient based optimization of the log likelihood. The EM algorithm is an iterative algorithm which generates a sequences of parameters $\{\theta_t\}$. Starting with an arbitrary $\theta_0 \in \Theta$, for $t = 0, 1, 2, \ldots$:

(a) **(E-step.)** Compute $Q(\theta, \theta_t) := \mathbb{E}_{p_{\theta_t}(z_{1:n} \mid x_{1:n})}[\log p_\theta(x_{1:n}, z_{1:n})]$.

(b) **(M-step.)** Select θ_{t+1} as follows:

$$\theta_{t+1} \in \arg\max_{\theta \in \Theta} Q(\theta, \theta_t).$$

We will work through some specific examples of the EM algorithm in a moment. However, at first glance, the EM algorithm is a bit mysterious. For instance, where does the definition of $Q(\theta, \theta_t)$ come from? Why do we maximize it to select θ_{t+1}?

2.7.1 Expectation Maximization Improves Log-Likelihood

We now show the following guarantee for the EM algorithm:

$$\log p_{\theta_{t+1}}(x_{1:n}) \geqslant \log p_{\theta_t}(x_{1:n}). \tag{2.18}$$

For what follows, we abbreviate $x = x_{1:n}$ and $z = z_{1:n}$. By Bayes' rule,

$$p_{\theta_{t+1}}(x) = \frac{p_{\theta_{t+1}}(x, z)}{p_{\theta_{t+1}}(z \mid x)} \Longrightarrow \log p_{\theta_{t+1}}(x) = \log p_{\theta_{t+1}}(x, z) - \log p_{\theta_{t+1}}(z \mid x).$$

Since this identity holds for every z, we can take the expectation of the RHS over $z \sim p_{\theta_t}(z \mid x)$:

$$\begin{aligned}
\log p_{\theta_{t+1}}(x) &= \mathbb{E}_{z \sim p_{\theta_t}(z|x)}[\log p_{\theta_{t+1}}(x, z)] - \mathbb{E}_{z \sim p_{\theta_t}(z|x)} \log p_{\theta_{t+1}}(z \mid x)] \\
&= Q(\theta_{t+1}, \theta_t) - \mathbb{E}_{z \sim p_{\theta_t}(z|x)}[\log p_{\theta_{t+1}}(z \mid x)] \\
&\geqslant Q(\theta_t, \theta_t) - \mathbb{E}_{z \sim p_{\theta_t}(z|x)}[\log p_{\theta_{t+1}}(z \mid x)] && \text{EM optimizes } Q(\cdot, \theta_t) \\
&= Q(\theta_t, \theta_t) - \mathbb{E}_{z \sim p_{\theta_t}(z|x)} \left[\log \frac{p_{\theta_{t+1}}(z \mid x) p_{\theta_t}(z \mid x)}{p_{\theta_t}(z \mid x)} \right] \\
&= Q(\theta_t, \theta_t) - \mathbb{E}_{z \sim p_{\theta_t}(z|x)}[\log p_{\theta_t}(z \mid x)] + \mathrm{KL}(p_{\theta_t}(\cdot \mid x) \parallel p_{\theta_{t+1}}(\cdot \mid x)) \\
&\geqslant Q(\theta_t, \theta_t) - \mathbb{E}_{z \sim p_{\theta_t}(z|x)}[\log p_{\theta_t}(z \mid x)] && \text{KL is non-negative} \\
&= \mathbb{E}_{z \sim p_{\theta_t}(z|x)}[\log p_{\theta_t}(x, z) - \log p_{\theta_t}(z \mid x)] \\
&= \mathbb{E}_{z \sim p_{\theta_t}(z|x)}[\log p_{\theta_t}(x)] && \text{Bayes' rule} \\
&= \log p_{\theta_t}(x).
\end{aligned}$$

2.7.2 Expectation Maximization as Coordinate Ascent

It turns out that EM can be understood as a form of coordinate ascent. Our starting point is the Donsker-Varadhan lemma (Lemma A.3), which appears again, but this time in a completely different context than PAC-Bayes. The specific form of Donsker-Varadhan we will use is the CGF form, which states that for a probability measure q and a function g, we have:

$$\log \mathbb{E}_q[\exp(g)] = \sup_{p \ll q} \{\mathbb{E}_p[g] - \mathrm{KL}(p \parallel q)\}.$$

The notation $p \ll q$ above means that p is absolutely continuous w.r.t. q, that is for any set A, if $q(A) = 0$, then $p(A) = 0$ as well (this ensures that the $\mathrm{KL}(p \parallel q)$ term is well-defined). Furthermore, Donsker-Varadhan states that the supremum on the RHS is achieved by the Gibbs measure:

$$\frac{\mathrm{d}\pi}{\mathrm{d}q} = \frac{\exp(g)}{\mathbb{E}_q[\exp(g)]}.$$

Fixing an x and applying Donsker-Varadhan to $g = \log p_\theta(x \mid z)$ and $q = p_\theta(z)$, we have that:

$$\log p_\theta(x) = \log \mathbb{E}_{p_\theta(z)}[p_\theta(x \mid z)] = \sup_{\mu \ll p_\theta(z)} \{\mathbb{E}_{\mu(z)}[\log p_\theta(x \mid z)] - \mathrm{KL}(\mu(z) \parallel p_\theta(z))\},$$

and furthermore, this supremum on the RHS is achieved by the posterior measure:

$$\frac{\mathrm{d}\mu(z)}{\mathrm{d}p_\theta(z)} = \frac{p_\theta(x \mid z)}{p_\theta(x)} \Longrightarrow \mu(z) = p_\theta(z \mid x).$$

(Since the DV representation holds for a fixed x, the distribution μ is allowed to depend on x.) Let us now define a function $F(\mu, \theta; x) : \mathcal{P}(\mathsf{Z}) \times \Theta \to \mathbb{R}$ as:

$$F(\mu, \theta; x) := \mathbb{E}_{\mu(z)}[\log p_\theta(x \mid z)] - \mathrm{KL}(\mu(z) \parallel p_\theta(z)),$$

so that by Donsker-Varadhan we have:

$$\log p_\theta(x) = \sup_{\mu \ll p_\theta(z)} F(\mu, \theta; x) = F(p_\theta(z \mid x), \theta; x). \tag{2.19}$$

Furthermore, observe that for $\theta' \in \Theta$,

$$F(\theta', \theta; x) = \mathbb{E}_{p_{\theta'}(z|x)}[\log p_\theta(x, z)] + H(p_{\theta'}(z \mid x)) = Q(\theta, \theta') + H(p_{\theta'}(z \mid x)), \tag{2.20}$$

where $H(\mu) = \mathbb{E}_\mu[-\log \mu(x)]$ is the entropy of μ.

With this notation, we can rewrite EM as the following algorithm for $t = 0, 1, 2, ...$

(a) **(E-step).** Set μ_t to be:

$$\mu_t \in \arg\max_{\mu \in \mathcal{P}(\mathsf{Z}^n)} F(\mu, \theta_t; x_{1:n}).$$

(b) **(M-step).** Set θ_{t+1} to be:

$$\theta_{t+1} \in \arg\max_{\theta \in \Theta} F(\mu_t, \theta; x_{1:n}).$$

We will now prove that EM satisfies for all t and $x_{1:n}$:

$$\log p_{\theta_{t+1}}(x_{1:n}) \geqslant \log p_{\theta_t}(x_{1:n}).$$

Hence, the EM algorithm monotonically increases the likelihood of the observed data $x_{1:n}$. The proof is quite simple given our current notation:

$$\log p_{\theta_{t+1}}(x_{1:n}) = F(p_{\theta_{t+1}}(z_{1:n} \mid x_{1:n}), \theta_{t+1}; x_{1:n}) \qquad \text{(Equation (2.19))}$$

$$
\begin{aligned}
&\geqslant F(p_{\theta_t}(z_{1:n} \mid x_{1:n}), \theta_{t+1}; x_{1:n}) && \text{(E-step)} \\
&= Q(\theta_{t+1}, \theta_t) + H(p_{\theta_t}(z \mid x)) && \text{(Equation (2.20))} \\
&\geqslant Q(\theta_t, \theta_t) + H(p_{\theta_t}(z \mid x)) && \text{(M-step)} \\
&= F(p_{\theta_t}(z_{1:n} \mid x_{1:n}), \theta_t; x_{1:n}) && \text{(Equation (2.20))} \\
&= \log p_{\theta_t}(x_{1:n}). && \text{(Equation (2.19))}
\end{aligned}
$$

2.7.3 Gaussian Mixture Models

Let us now instantiate the EM algorithm for learning a Gaussian mixture model (GMM). A GMM works by first sampling a latent index $z \sim \mathrm{Cat}(\pi_1, \ldots, \pi_k)$, and then sampling $x \mid z \sim N(\mu_z, \Sigma_z)$. Thus, a GMM's parameters are:

$$\theta = (\{\pi_j\}_{j=1}^k, \{(\mu_j, \Sigma_j)\}_{j=1}^k),$$

and its log density is:

$$\log p_\theta(x, z{=}j) = \log \pi_j + \log \phi(x; \mu_j, \Sigma_j),$$

where $\phi(x; \mu, \Sigma)$ is the density of a multivariate Gaussian:

$$\phi(x; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d \det \Sigma}} \exp\left(-\frac{1}{2}\|x - \mu\|^2_{\Sigma^{-1}}\right).$$

E-step. We first compute the E-step, by computing the conditional expectation $p_\theta(z \mid x)$. This is accomplished by Bayes rule. Given a single observation x,

$$r_\theta(j; x) := p_\theta(z{=}j \mid x) = \frac{p_\theta(x, z{=}j)}{p_\theta(x)} = \frac{\pi_j \phi(x; \mu_j, \Sigma_j)}{\sum_{j'=1}^k \pi_{j'} \phi(x; \mu_{j'}, \Sigma_{j'})}.$$

Furthermore, let us define a shorthand notation:

$$r_\theta(j) := \sum_{i=1}^n r_\theta(j; x_i).$$

Now by independence, the E-step $Q(\theta, \theta')$ simplifies:

$$
\begin{aligned}
Q(\theta, \theta') &= \mathbb{E}_{p_{\theta'}(z_{1:n} \mid x_{1:n})}[\log p_\theta(x_{1:n}, z_{1:n})] \\
&= \sum_{i=1}^n \mathbb{E}_{p_{\theta'}(z_i \mid x_i)}[\log p_\theta(x_i, z_i)] \\
&= \sum_{i=1}^n \sum_{j=1}^k p_{\theta'}(z_i{=}j \mid x_i) \log p_\theta(x_i, z_i{=}j) \\
&= \sum_{i=1}^n \sum_{j=1}^k r_{\theta'}(j; x_i)[\log \pi_j + \log \phi(x_i; \mu_j, \Sigma_j)] \\
&= \sum_{j=1}^k r_{\theta'}(j) \log \pi_j + \sum_{j=1}^k \sum_{i=1}^n r_{\theta'}(j; x_i) \log \phi(x_i; \mu_j, \Sigma_j).
\end{aligned} \tag{2.21}
$$

M-step. The E-step gives us a function $Q(\theta, \theta')$ to optimize over θ. We will optimize the function $\theta \mapsto Q(\theta, \theta')$ separately over $\{\pi_j\}$ and each pair (μ_j, Σ_j). First, let us optimize over the mixture parameters $\{\pi_j\}$. Let S^{k-1} denote the $(k-1)$-dimensional probability simplex in $\mathbb{R}^k$:

$$S^{k-1} = \left\{ \pi \in \mathbb{R}^k_{\geqslant 0} \;\middle|\; \sum_{j=1}^k \pi_j = 1 \right\}.$$

The optimization problem over $\pi \in S^{k-1}$ is:

$$\hat{\pi} = \arg \max_{\pi \in S^{k-1}} \sum_{j=1}^{k} r_{\theta'}(j) \log \pi_j. \tag{2.22}$$

This is a negative cross-entropy maximization between the (unnormalized distribution) of $\{r_{\theta'}(j)\}$ and $\{\pi_j\}$. Therefore, the optimal $\hat{\pi}$ is given by normalizing $\{r_{\theta'}(j)\}$ to form a valid probability distribution:

$$\hat{\pi}_j = \frac{r_{\theta'}(j)}{\sum_{j'=1}^{k} r_{\theta'}(j')} = \frac{r_{\theta'}(j)}{n}. \tag{2.23}$$

To see that the normalizing constant in the denominator of (2.23) equals n, note that:

$$\sum_{j=1}^{k} r_{\theta'}(j) = \sum_{j=1}^{k} \sum_{i=1}^{n} r_{\theta'}(j; x_i) = \sum_{i=1}^{n} \sum_{j=1}^{k} r_{\theta'}(j; x_i) = \sum_{i=1}^{n} \sum_{j=1}^{k} p_{\theta'}(z{=}j \mid x_i) = n.$$

As for (2.23) itself, (2.23) can be shown by a standard Lagrange multiplier argument. Consider the Lagrangian:

$$L(\pi, \lambda) = \sum_{j=1}^{k} r_{\theta'}(j) \log \pi_j + \lambda \left(\sum_{j=1}^{k} \pi_j - 1 \right).$$

The optimization problem (2.22) maximizes a concave function over a convex set S^{k-1}, so the KKT conditions are both necessary and sufficient. We have that the optimal solution is a stationary point of the Lagrangian:

$$0 = \frac{\partial L(\pi, \lambda)}{\partial \pi_j} = \frac{r_{\theta'}(j)}{\pi_j} + \lambda, \quad j = 1, \ldots, k.$$

We can solve for λ by using the simplex constraint $\sum_{j=1}^{k} \pi_j = 1$,

$$1 = \sum_{j=1}^{k} \pi_j = -\sum_{j=1}^{k} \frac{r_{\theta'}(j)}{\lambda} \Longrightarrow \lambda = -\sum_{j=1}^{k} r_{\theta'}(j).$$

Therefore, the optimal solution satisfies:

$$\pi_j = \frac{r_{\theta'}(j)}{\sum_{j'=1}^{k} r_{\theta'}(j')}.$$

Now let us optimize for each mixture parameter (μ_j, Σ_j). Note by symmetry, the optimization for each mixture $j \in \{1, \ldots, k\}$ is the same. For mixture j, the optimization problem is:

$$\begin{aligned}(\hat{\mu}_j, \hat{\Sigma}_j) &= \arg \max_{\mu_j, \Sigma_j \succ 0} \sum_{i=1}^{n} r_{\theta'}(j; x_i) \log \phi(x_i; \mu_j, \Sigma_j) \\ &= \arg \max_{\mu_j, \Sigma_j \succ 0} L(\mu_j, \Sigma_j) := \sum_{i=1}^{n} r_{\theta'}(j; x_i) \left[-\frac{1}{2} \|x_i - \mu_j\|^2_{\Sigma_j^{-1}} - \frac{1}{2} \log \det \Sigma_j \right].\end{aligned}$$

Let us first differentiate $L(\mu_j, \Sigma_j)$ w.r.t. μ_j:

$$\nabla_{\mu_j} L(\mu_j, \Sigma_j) = -\sum_{i=1}^{n} r_{\theta'}(j; x_i) \Sigma_j^{-1} (\mu_j - x_i).$$

Setting this gradient equal to zero we obtain:

$$\hat{\mu}_j = \frac{\sum_{i=1}^n r_{\theta'}(j; x_i) x_i}{r_{\theta'}(j)}.$$

Next, we compute the gradient w.r.t. Σ_j. To do this, we rely on the following facts, valid for X and M symmetric (and X invertible):

$$\nabla_X \operatorname{tr}(X^{-1}M) = -X^{-1}MX^{-1}, \quad \nabla_X \log\det X = X^{-1}.$$

With these facts, we have:

$$\begin{aligned}\nabla_{\Sigma_j} L(\mu_j, \Sigma_j) &= \nabla_{\Sigma_j} \left\{ -\frac{1}{2} \sum_{i=1}^n r_{\theta'}(j; x_i) \left[\operatorname{tr}(\Sigma_j^{-1}(x_i - \mu_j)(x_i - \mu_j)^\mathsf{T}) + \log\det \Sigma_j \right] \right\} \\ &= -\frac{1}{2} \sum_{i=1}^n r_{\theta'}(j; x_i) \left[-\Sigma_j^{-1}(x_i - \mu_j)(x_i - \mu_j)^\mathsf{T}\Sigma_j^{-1} + \Sigma_j^{-1} \right].\end{aligned}$$

Setting $\nabla_{\Sigma_j} L(\mu_j, \Sigma_j) = 0$ yields:

$$\Sigma_j = \sum_{i=1}^n \frac{r_{\theta'}(j; x_i)}{r_{\theta'}(j)} (x_i - \mu_j)(x_i - \mu_j)^\mathsf{T}.$$

Hence, the following pair $(\hat{\mu}_j, \hat{\Sigma}_j)$ is the unique stationary point of $L(\mu_j, \Sigma_j)$ assuming that $\hat{\Sigma}_j$ is invertible:

$$\hat{\mu}_j = \sum_{i=1}^n \frac{r_{\theta'}(j; x_i)}{r_{\theta'}(j)} x_i, \quad \hat{\Sigma}_j = \sum_{i=1}^n \frac{r_{\theta'}(j; x_i)}{r_{\theta'}(j)} (x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^\mathsf{T}.$$

Putting it together. We now summarize the full EM algorithm for learning a GMM. We start with an initial set of model parameters

$$\theta_0 = (\{\pi_j^{(0)}\}_{j=1}^k, \{(\mu_j^{(0)}, \Sigma_j^{(0)})\}_{j=1}^k),$$

and for $t \in \mathbb{N}$, we generate $\theta_{t+1} = (\{\pi_j^{(t+1)}\}_{j=1}^k, \{(\mu_j^{(t+1)}, \Sigma_j^{(t+1)})\}_{j=1}^k)$ by the following updates for $j \in \{1, \ldots, k\}$:

$$\pi_j^{(t+1)} = \frac{r_{\theta_t}(j)}{n}, \quad \mu_j^{(t+1)} = \sum_{i=1}^n \frac{r_{\theta_t}(j; x_i)}{r_{\theta_t}(j)} x_i, \quad \Sigma_j^{(t+1)} = \sum_{i=1}^n \frac{r_{\theta_t}(j; x_i)}{r_{\theta_t}(j)} (x_i - \mu_j^{(t+1)})(x_i - \mu_j^{(t+1)})^\mathsf{T}.$$

One can interpret EM as a form of *soft* k-means clustering. In standard k-means clustering, each observation x_i is given a hard latent assignment z_i, and the means and covariances are updated based on this hard assignment. On the other hand in the EM algorithm, for each observation x_i, a probability distribution

$$\left\{ \frac{r_{\theta_t}(j; x_i)}{r_{\theta_t}(j)} \right\}_{j=1}^k$$

over cluster assignments is used to update the new means and covariances of each cluster. That is, no hard cluster assignments are made during EM.

2.8 Variational Inference

One key assumption in the EM algorithm is that the posterior distribution $p_\theta(z \mid x)$ can be efficiently computed. In the case of finite mixture models, we used Bayes' rule to compute this posterior. However, in the case when z is continuous, in general it is not tractable to compute $p_\theta(z \mid x)$. Variational inference (VI) gives us an alternative algorithm to handle these situations. We now review two approaches to deriving the VI objective, often referred to as the evidence lower bound (ELBO).

Jensen's inequality approach: Let $q(z \mid x)$ be any posterior distribution. We have:

$$\begin{aligned}
\log p_\theta(x) &= \log \left[\int p_\theta(x \mid z) p_\theta(z) \, \mathrm{d}z \right] \\
&= \log \left[\int \frac{p_\theta(x \mid z) p_\theta(z)}{q(z \mid x)} q(z \mid x) \, \mathrm{d}z \right] \\
&= \log \mathbb{E}_{q(z|x)} \left[\frac{p_\theta(x \mid z) p_\theta(z)}{q(z \mid x)} \right] \\
&\geqslant \mathbb{E}_{q(z|x)} \log \left[\frac{p_\theta(x \mid z) p_\theta(z)}{q(z \mid x)} \right] && \text{Jensen's inequality} \\
&= \mathbb{E}_{q(z|x)} [\log p_\theta(x \mid z)] - \mathbb{E}_{q(z|x)} \log \left[\frac{q(z \mid x)}{p_\theta(z)} \right] \\
&= \mathbb{E}_{q(z|x)} [\log p_\theta(x \mid z)] - \mathrm{KL}(q(z \mid x) \parallel p_\theta(z)).
\end{aligned}$$

Donsker-Varadhan approach: By the Donsker-Varadhan representation of the log-likelihood:

$$\log p_\theta(x) = \sup_{\mu \ll p_\theta(z)} \{ \mathbb{E}_{\mu(z)} [\log p_\theta(x \mid z)] - \mathrm{KL}(\mu(z) \parallel p_\theta(z)) \}.$$

Furthermore, from Donsker-Varadhan, we know that the optimal μ in the supremum above is given by the posterior $\mu = p_\theta(z \mid x)$. However, the following ELBO lower bound is valid for any posterior $q(z \mid x)$ (not just the optimal one):

$$\log p_\theta(x) \geqslant \mathbb{E}_{q(z|x)} [\log p_\theta(x \mid z)] - \mathrm{KL}(q(z \mid x) \parallel p_\theta(z)). \tag{2.24}$$

Hence a reasonable choice of algorithm is to maximize the ELBO over posteriors $q(z \mid x)$. This gives rise to the population variational inference objective:

$$\arg\max_{\theta, \phi} L(\theta, \phi) := \mathbb{E}_{p(x)} \left[\mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x \mid z)] - \mathrm{KL}(q_\phi(z \mid x) \parallel p_\theta(z)) \right].$$

One word about terminology. The posterior distribution $q_\phi(z \mid x)$ is often called the *encoder*, the forward model $p_\theta(x \mid z)$ is often called the *decoder*, and the distribution $p_\theta(z)$ is often called the *prior*. Since the loss function $L(\theta, \phi)$ proceeds by first encoding x via $q_\phi(z \mid x)$ and then decoding it via $p_\theta(x \mid z)$, this type of model is often called a *variational autoencoder* (VAE).

One natural question remains how to compute the KL-divergence term. In general computing the KL-divergence between two arbitrary continuous distributions is intractable (again requiring numerical integration). But there is a special case when the KL-divergence between two distributions has a simple closed form expression, and that is when both distributions are Gaussians (cf. Proposition A.4). That is, let us set:

$$\begin{aligned}
q_\phi(z \mid x) &= N(e_{\mu,\phi}(x), e_{\Sigma,\phi}(x)), \\
p_\theta(z) &= N(\mu_p, \Sigma_p).
\end{aligned}$$

Here, the maps $e_{\mu,\phi}(x)$ and $e_{\Sigma,\phi}(x)$ are arbitrary non-linear maps sending the observation x to a mean and covariance matrix. With this setting, we can compute the KL-divergence term directly as:

$$\mathrm{KL}(q_\phi(z \mid x) \parallel p_\theta(z)) = \frac{1}{2} \left\{ \mathrm{tr}(\Sigma_p^{-1} \cdot e_{\Sigma,\phi}(x)) + \|\mu_p - e_{\mu,\phi}(x)\|^2_{\Sigma_p^{-1}} - \ell + \log \frac{\det \Sigma_p}{\det e_{\Sigma,\phi}(x)} \right\},$$

where we let ℓ denote the dimensionality of z (i.e., $z \in \mathbb{R}^\ell$).

We now turn to the term $\mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x \mid z)]$ in the ELBO. Computing the value of the expectation in closed form is generally not tractable (depending on the form of $p_\theta(x \mid z)$). However, we can easily create a stochastic estimator using what is known as the *re-parameterization trick*.

Re-parameterization trick. Consider optimizing the following objective:

$$F(\phi) := \mathbb{E}_{q_\phi(z)}[f(z)],$$

where $q_\phi(z)$ is some distribution over z. Suppose we proceed by drawing samples $z_1, \ldots, z_n \sim q_\phi(z)$ and forming a stochastic estimate:

$$\hat{F}(\phi) := \frac{1}{n} \sum_{i=1}^{n} f(z_i).$$

Unfortunately, if we take the gradient of $\hat{F}(\phi)$ with respect to ϕ, we have $\nabla_\phi \hat{F}(\phi) = 0$, and hence this does not properly capture the dependency of $\hat{F}(\phi)$ on ϕ. The reason is that $\hat{F}(\phi)$ depends on ϕ implicitly via the sampling process, and so once the samples $z_1, \ldots, z_n$ are drawn, there is no further dependence on ϕ. How can we deal with this issue then?

Suppose that $q_\phi(z)$ is a Gaussian with $q_\phi(z) = N(\mu_\phi, \Sigma_\phi)$. Then we have that:

$$z \stackrel{d}{=} \mu_\phi + \Sigma_\phi^{1/2} g, \quad z \sim q_\phi, \quad g \sim N(0, I).$$

Hence,

$$F(\phi) = \mathbb{E}_{z \sim q_\phi(z)}[f(z)] = \mathbb{E}_{g \sim N(0,I)}[f(\mu_\phi + \Sigma_\phi^{1/2} g)],$$

where importantly on the RHS, the expectation no longer depends on ϕ. Hence, we can form a stochastic approximation by drawing $g_1, \ldots, g_n \sim N(0, I)$, and setting

$$\hat{F}(\phi) = \frac{1}{n} \sum_{i=1}^{n} f(\mu_\phi + \Sigma_\phi^{1/2} g_i).$$

This stochastic approximation now has the property that

$$\mathbb{E}[\nabla_\phi \hat{F}(\phi)] = \nabla_\phi F(\phi).$$

Now, applying the re-parameterization trick to the first term of the ELBO:

$$\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] = \mathbb{E}_g[\log p_\theta(x \mid z = e_{\mu,\phi}(x) + e_{\Sigma,\phi}^{1/2}(x) g)],$$

where $g \sim N(0, I)$. Hence, the population VI objective equals:

$$L(\theta, \phi) = \mathbb{E}_{p(x),g}\left[\log p_\theta(x \mid z = e_{\mu,\phi}(x) + e_{\Sigma,\phi}^{1/2}(x) g) - \mathrm{KL}(q_\phi(z \mid x) \parallel p_\theta(z))\right].$$

We can easily form a finite sample objective, given samples $x_1, \ldots, x_n$ and random Gaussian vectors $g_1, \ldots, g_n$:

$$\hat{L}_n(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} \left[\log p_\theta(x_i \mid z = e_{\mu,\phi}(x_i) + e_{\Sigma,\phi}^{1/2}(x_i) g_i) - \mathrm{KL}(q_\phi(z \mid x_i) \parallel p_\theta(z))\right].$$

This finite sample objective can then be optimized by standard gradient descent methods.

Reconstruction loss. The loss term $\log p_\theta(x_i \mid z = e_{\mu,\phi}(x_i) + e_{\Sigma,\phi}^{1/2}(x_i) g_i)$ is often referred to as the reconstruction loss. Let us motivate why this is the case. Suppose we make an extra assumption that the decoder $p_\theta(x \mid z)$ is also Gaussian:

$$p_\theta(x \mid z) = N(d_{\mu,\theta}(z), d_{\Sigma,\theta}(z)),$$

where again like in the encoder, $d_{\mu,\theta}(z), d_{\Sigma,\theta}(z)$ are arbitrary non-linear functions which take a latent z and produce a mean and covariance in the observation space. With this parameterization, we have:

$$\log p_\theta(x \mid z) = -\frac{1}{2}\|x - d_{\mu,\theta}(z)\|^2_{d^{-1}_{\Sigma,\theta}(z)} - \frac{1}{2}\log((2\pi)^d \det d_{\Sigma,\theta}(z)).$$

If we plug in the reparameterization trick

$$z = e_{\mu,\phi}(x) + e^{1/2}_{\Sigma,\phi}(x)g,$$

we see that the square loss term of $\log p_\theta(x \mid z)$ takes into account the difference of:

$$x - d_{\mu,\theta}(e_{\mu,\phi}(x) + e^{1/2}_{\Sigma,\phi}(x)g).$$

We can interpret this as a reconstruction error. Given an x, we first encode it into a latent z via $z = e_{\mu,\phi}(x) + e^{1/2}_{\Sigma,\phi}(x)g$, and then we decode the latent back into the observation via $d_{\mu,\theta}(z)$. We then ask that the original observation z matches the encoding/decoding process $d_{\mu,\theta}(z)$.

ELBO gap. The ELBO (2.24) is a lower bound on $\log p_\theta(x)$, valid for every posterior $q(z \mid x)$. It turns out we can actually quantify the gap in the ELBO. In particular, the proof of the Donsker-Varadhan lemma actually implies:

$$\log p_\theta(x) = \left[\mathbb{E}_{q(z|x)}[\log p_\theta(x \mid z)] - \mathrm{KL}(q(z \mid x) \parallel p_\theta(z))\right] + \mathrm{KL}(q(z \mid x) \parallel p_\theta(z \mid x)).$$

Thus the distance of $q(z \mid x)$ from the true posterior $p_\theta(z \mid x)$, measured in the KL-divergence, quantifies the gap in the ELBO.

2.8.1 Markovian latent variable models

We now consider the following generalization involving Markovian latent variable models. Suppose now that instead of a single latent variable z as before, there are now T latent variables $z_1, \ldots, z_T$. Consider the following forward model:

$$z_T \sim p_\theta(z_T), \quad z_{t-1} \sim p_\theta(z_{t-1} \mid z_t),\ t \in \{2, \ldots, T\}, \quad x \sim p_\theta(x \mid z_1).$$

The ELBO gives us a lower bound on $\log p_\theta(x)$, valid for any posterior $q_\phi(z_{1:T} \mid x)$:

$$\begin{aligned}\log p_\theta(x) &\geqslant \mathbb{E}_{q_\phi(z_{1:T}|x)}[\log p_\theta(x \mid z_{1:T})] - \mathrm{KL}(q_\phi(z_{1:T} \mid x) \parallel p_\theta(z_{1:T})) \\ &= \mathbb{E}_{q_\phi(z_{1:T}|x)}\left[\log \frac{p_\theta(x, z_{1:T})}{q_\phi(z_{1:T} \mid x)}\right].\end{aligned}$$

While the above ELBO is valid for any posterior $q_\phi(z_{1:T} \mid x)$, we will further impose Markovian structure on the posterior via the following form:

$$z_1 \sim q_\phi(z_1 \mid x), \quad z_t \sim q_\phi(z_t \mid z_{t-1}),\ t \in \{2, \ldots, T\}.$$

With the imposed Markovian structures, we can factorize both the forward model $p_\theta(x, z_{1:T})$ and the posterior $q_\phi(z_{1:T} \mid x)$ as follows:

$$\begin{aligned}p_\theta(x, z_{1:T}) &= p_\theta(z_T)\left[\prod_{t=2}^{T} p_\theta(z_{t-1} \mid z_t)\right] p_\theta(x \mid z_1), \\ q_\phi(z_{1:T} \mid x) &= q_\phi(z_1 \mid x)\prod_{t=2}^{T} q_\phi(z_t \mid z_{t-1}).\end{aligned}$$

Let us now plug in these factorizations into the ELBO and try to simplify the lower bound in terms of sum of KL-divergences across t. Reindexing to align the timesteps and dropping the subscripts θ, ϕ for ease of notation:

$$\log \frac{p(x, z_{1:T})}{q(z_{1:T} \mid x)} = \log \frac{p(z_T)p(x \mid z_1)}{q(z_T \mid z_{T-1})} + \sum_{t=1}^{T-1} \log \frac{p(z_t \mid z_{t+1})}{q(z_t \mid z_{t-1})},$$

with the understanding that $z_0 := x$. Taking expectation w.r.t. $q(z_{1:T} \mid x)$,

$$\begin{aligned}\mathbb{E}_{q(z_{1:T}|x)}\left[\log \frac{p(x, z_{1:T})}{q(z_{1:T} \mid x)}\right] &= \mathbb{E}_{q(z_1|x)}[\log p(x \mid z_1)] - \mathbb{E}_{q(z_{T-1}|x)}[\mathrm{KL}(q(z_T \mid z_{T-1}) \parallel p(z_T))] \\ &\quad - \sum_{t=1}^{T-1} \mathbb{E}_{q(z_{t-1}, z_t, z_{t+1}|x)}\left[\log \frac{q(z_t \mid z_{t-1})}{p(z_t \mid z_{t+1})}\right]. \end{aligned} \tag{2.25}$$

At this point, (2.25) yields an ELBO which we can optimize. Above, the first two terms are familiar: we have the reconstruction loss and the KL-divergence term. However, the last terms can be rewritten to reduce variance of the objective.

Towards rewriting the terms in the summation of (2.25), it is tempting to factorize

$$q(z_{t-1}, z_t, z_{t+1} \mid x) = q(z_{t-1}, z_{t+1} \mid x)q(z_t \mid z_{t-1}, z_{t+1}),$$

which make the last terms almost look like a KL-divergence:

$$\mathbb{E}_{q(z_{t-1}, z_t, z_{t+1}|x)}\left[\log \frac{q(z_t \mid z_{t-1})}{p(z_t \mid z_{t+1})}\right] = \mathbb{E}_{q(z_{t-1}, z_{t+1}|x)}\mathbb{E}_{q(z_t|z_{t-1}, z_{t+1})}\left[\log \frac{q(z_t \mid z_{t-1})}{p(z_t \mid z_{t+1})}\right].$$

Unfortunately, the issue is that in general

$$q(z_t \mid z_{t-1}, z_{t+1}) \neq q(z_t \mid z_{t-1}),$$

and therefore this approach does not quite work. However, we can rewrite $q(z_t \mid z_{t-1})$ using Bayes' rule as follows:

$$\begin{aligned} q(z_t \mid z_{t-1}) &= q(z_t \mid z_{t-1}, x) && \text{(Markov property)} \\ &= \frac{q(z_{t-1} \mid z_t, x)q(z_t \mid x)}{q(z_{t-1} \mid x)}. && \text{(Bayes' rule)} \end{aligned}$$

A nice telescoping cancellation now occurs if we plug this identity into the factorization of $q(z_{1:T} \mid x)$:

$$\begin{aligned} q(z_{1:T} \mid x) &= q(z_1 \mid x) \prod_{t=2}^{T} q(z_t \mid z_{t-1}) \\ &= \left[\prod_{t=2}^{T} q(z_{t-1} \mid z_t, x)\right] q(z_T \mid x). \end{aligned}$$

Now with this identity, the density ratio simplifies to:

$$\log \frac{p(x, z_{1:T})}{q(z_{1:T} \mid x)} = \log \frac{p(z_T)p(x \mid z_1)}{q(z_T \mid x)} + \sum_{t=2}^{T} \log \frac{p(z_{t-1} \mid z_t)}{q(z_{t-1} \mid z_t, x)}.$$

Taking expectation w.r.t. $q(z_{1:T} \mid x)$ yields:

$$\mathbb{E}_{q(z_{1:T}|x)}\left[\log \frac{p(x, z_{1:T})}{q(z_{1:T} \mid x)}\right] = \mathbb{E}_{q(z_1|x)}[\log p(x \mid z_1)] - \mathrm{KL}(q(z_T \mid x) \parallel p(z_T))$$

$$- \sum_{t=2}^{T} \mathbb{E}_{q(z_{t-1}, z_t | x)} \left[\log \frac{q(z_{t-1} \mid z_t, x)}{p(z_{t-1} \mid z_t)} \right].$$

We now factorize the last terms as follows:

$$\begin{aligned} \mathbb{E}_{q(z_{t-1}, z_t | x)} \left[\log \frac{q(z_{t-1} \mid z_t, x)}{p(z_{t-1} \mid z_t)} \right] &= \mathbb{E}_{q(z_t | x)} \mathbb{E}_{q(z_{t-1} | z_t, x)} \left[\log \frac{q(z_{t-1} \mid z_t, x)}{p(z_{t-1} \mid z_t)} \right] \\ &= \mathbb{E}_{q(z_t | x)} [\mathrm{KL}(q(z_{t-1} \mid z_t, x) \parallel p(z_{t-1} \mid z_t))]. \end{aligned}$$

Hence,

$$\begin{aligned} \mathbb{E}_{q(z_{1:T} | x)} \left[\log \frac{p(x, z_{1:T})}{q(z_{1:T} \mid x)} \right] &= \mathbb{E}_{q(z_1 | x)} [\log p(x \mid z_1)] - \mathrm{KL}(q(z_T \mid x) \parallel p(z_T)) \\ &\quad - \sum_{t=2}^{T} \mathbb{E}_{q(z_t | x)} [\mathrm{KL}(q(z_{t-1} \mid z_t, x) \parallel p(z_{t-1} \mid z_t))]. \end{aligned} \tag{2.26}$$

This identity above in (2.26) will be our starting point for deriving diffusion models. Compared with (2.25), using Bayes' rule has allowed us write part of the expectation in the summation terms as a KL-divergence. Hence if $q(z_{t-1} \mid z_t, x)$ and $p(z_{t-1} \mid z_t)$ are both Gaussians, then this yields an analytical expression, reducing variance compared with (2.25).

2.9 Denoising diffusion probabilistic models

In this section, we present denoising diffusion probabilistic models (DDPM), due to Ho et al. [2020]. We heavily follow the mathematical derivation presented in Luo [2022]. A DDPM is an instance of a Markovian latent variable model (cf. Section 2.8.1) with a particular non-learnable posterior q. A Markov latent variable model has an observation x and latents $z_{1:T}$. One of the defining features of a DDPM is that the latents and the observation live in the same space (i.e., $\mathsf{X} = \mathsf{Z}$). We use the notation $x_0 = x$ and $x_t = z_t$ for $t \in \{1, \ldots, T\}$, so that the random variables involved are $x_0, x_1, \ldots, x_T$.

The main idea of DDPM is, we construct the posterior q to take the observation x_0 and gradually transform it into noise at x_T. We then use variational inference to learn the reverse process $p(x_{t-1} \mid x_t)$, which reverses the noising process, turning noise into samples.

For what follows, fix a $T \in \mathbb{N}_+$ and a sequence of positive scalars $\{\alpha_t\}_{t=1}^T \subset (0, 1)$. For shorthand, define:

$$\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s, \quad \bar{\alpha}_0 := 1.$$

To make these values concrete, the original DDPM paper set $T = 1000$, $\alpha_1 = 1 - 10^{-4}$, $\alpha_T = 0.98$, and α_t interpolating linearly between α_1 and α_T, although follow up work has shown that this selection can be improved [Nichol and Dhariwal, 2021].

Training algorithm. DDPM parameterizes a model $f_\theta(x, t)$ mapping $\mathbb{R}^d \times \{1, \ldots, T\} \mapsto \mathbb{R}^d$, and optimizes:

$$L(\theta) = \mathbb{E}_{t \in \mathrm{Unif}(\{1, \ldots, T\})} \mathbb{E}_{(x_0, w_t)} \| w_t - f_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} w_t, t) \|^2. \tag{2.27}$$

We now describe how to form a stochastic gradient for $L(\theta)$. Suppose for simplicity we consider a batch size of one. Given an observation $x^{(i)}$, we draw a random index $t^{(i)} \in \mathrm{Unif}(\{1, \ldots, T\})$ and a random vector $w^{(i)} \sim N(0, I)$, and form the following stochastic gradient (we ignore the scaling factor of T in the stochastic gradient below):

$$g(\theta) = \nabla_\theta \| w^{(i)} - f_\theta(\sqrt{\bar{\alpha}_{t^{(i)}}} x^{(i)} + \sqrt{1 - \bar{\alpha}_{t^{(i)}}} w^{(i)}, t^{(i)}) \|^2.$$

Sampling algorithm. Now given a trained $f_\theta(x,t)$ model (from optimizing (2.27)), the sampling algorithm is:

$$x_T \sim N(0,I), \quad x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} f_\theta(x_t,t)\right) + \sqrt{\frac{(1-\bar{\alpha}_{t-1})(1-\alpha_t)}{1-\bar{\alpha}_t}} w_t, \; t \in \{1,\dots,T\}, \quad (2.28)$$

with $w_t \sim N(0,I)$ for $t \in \{1,\dots,T\}$. This looks very much like a form of Langevin sampling, with $f_\theta(x_t,t)$ playing the role of a scaled version of the score function.

2.9.1 Deriving the DDPM loss and sampler

We now discuss how the DDPM loss (2.27) and DDPM sampler (2.28) are derived.

Defining the noising process $q(x_{1:T} \mid x_0)$. Let us specify the noising process q, parameterized by the scalars $\{\alpha_t\}_{t=1}^T$ as:

$$q(x_t \mid x_{t-1}) = N(\sqrt{\alpha_t} x_{t-1}, (1-\alpha_t)I), \quad t \in \{1,\dots,T\}.$$

We now show that the noising process q satisfies:

$$q(x_t \mid x_0) = N(\sqrt{\bar{\alpha}_t} x_0, (1-\bar{\alpha}_t)I), \quad t \in \{0,\dots,T\}.$$

We proceed by induction. The base case is satisfied by the definition of $\bar{\alpha}_0 = 1$. Now supposing the identity holds at t, let us consider $q(x_{t+1} \mid x_0)$. Let w_t, w_{t+1} denote $N(0,I)$ random vectors independent of x_0. We can write:

$$\begin{aligned} x_{t+1} &= \sqrt{\alpha_{t+1}} x_t + \sqrt{1-\alpha_{t+1}} w_{t+1} \\ &= \sqrt{\alpha_{t+1}}(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t} w_t) + \sqrt{1-\alpha_{t+1}} w_{t+1} \\ &= \sqrt{\bar{\alpha}_{t+1}} x_0 + \sqrt{\alpha_{t+1}(1-\bar{\alpha}_t)} w_t + \sqrt{1-\alpha_{t+1}} w_{t+1}. \end{aligned}$$

Hence, we see that

$$\begin{aligned} q(x_{t+1} \mid x_0) &= N(\sqrt{\bar{\alpha}_{t+1}} x_0, (\alpha_{t+1}(1-\bar{\alpha}_t) + 1 - \alpha_{t+1})I) \\ &= N(\sqrt{\bar{\alpha}_{t+1}} x_0, (1-\bar{\alpha}_{t+1})I), \end{aligned}$$

which finishes the induction step.

Note that as long as $\bar{\alpha}_t \to 0$ as $t \to \infty$, then if T is large enough we have that $q(x_T \mid x_0) \approx N(0,I)$, and hence the data x_0 eventually becomes random noise at the end of the noising process.

Reversing the noising process and computing the ELBO. Towards being able to evaluate the ELBO (cf. Equation (2.26)), we need to be able to compute the reverse of the denoising process, conditioned on the observation x. Specifically, we need to compute for $t \geqslant 2$:

$$q(x_{t-1} \mid x_t, x_0).$$

We do this by first computing the joint distribution of $(x_{t-1}, x_t) \mid x_0$. Letting w_{t-1}, w_t be independent $N(0,I)$ random vectors (also independent of x_0), we write:

$$\begin{aligned} x_{t-1} &= \sqrt{\bar{\alpha}_{t-1}} x_0 + \sqrt{1-\bar{\alpha}_{t-1}} w_{t-1}, \\ x_t &= \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t} w_t \\ &= \sqrt{\bar{\alpha}_t} x_0 + \sqrt{\alpha_t(1-\bar{\alpha}_{t-1})} w_{t-1} + \sqrt{1-\alpha_t} w_t. \end{aligned}$$

Therefore,

$$\begin{bmatrix} x_{t-1} \\ x_t \end{bmatrix} = \begin{bmatrix} \sqrt{\bar{\alpha}_{t-1}} x_0 \\ \sqrt{\bar{\alpha}_t} x_0 \end{bmatrix} + \begin{bmatrix} \sqrt{1-\bar{\alpha}_{t-1}} I \\ \sqrt{\alpha_t(1-\bar{\alpha}_{t-1})} I \end{bmatrix} w_{t-1} + \begin{bmatrix} 0 \\ \sqrt{1-\alpha_t} I \end{bmatrix} w_t.$$

Hence, we have that the joint distribution $(x_{t-1}, x_t) \mid x_0$ is:

$$\begin{bmatrix} x_{t-1} \\ x_t \end{bmatrix} \mid x_0 = N\left(\begin{bmatrix} \sqrt{\bar{\alpha}_{t-1}} x_0 \\ \sqrt{\bar{\alpha}_t} x_0 \end{bmatrix}, \begin{bmatrix} (1-\bar{\alpha}_{t-1}) I & (1-\bar{\alpha}_{t-1})\sqrt{\alpha_t} I \\ (1-\bar{\alpha}_{t-1})\sqrt{\alpha_t} I & (1-\bar{\alpha}_t) I \end{bmatrix} \right).$$

We now use the following fact.

Proposition 2.5. *Suppose that (X_1, X_2) are jointly Gaussian:*

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12}^\mathsf{T} & \Sigma_{22} \end{bmatrix} \right).$$

Then,

$$X_1 \mid X_2 = x \sim N\left(\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{12}^\mathsf{T}\right).$$

Therefore, using Proposition 2.5, we see that

$$q(x_{t-1} \mid x_t, x_0) = N(\mu_q(x_0, x_t), \Sigma_q(t)),$$

where the mean is:

$$\begin{aligned} \mu_q(x_0, x_t) &= \mathbb{E}[x_{t-1} \mid x_t, x_0] \\ &= \sqrt{\bar{\alpha}_{t-1}} x_0 + \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}(x_t - \sqrt{\bar{\alpha}_t} x_0) \\ &= \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}} x_0 + (1-\bar{\alpha}_{t-1})\sqrt{\alpha_t} x_t}{1-\bar{\alpha}_t}, \end{aligned}$$

and covariance is:

$$\begin{aligned} \Sigma_q(t) &= \mathrm{Cov}(x_{t-1} \mid x_t, x_0) \\ &= \left[(1-\bar{\alpha}_{t-1}) - \frac{(1-\bar{\alpha}_{t-1})^2 \alpha_t}{1-\bar{\alpha}_t}\right] I \\ &= \frac{(1-\bar{\alpha}_{t-1})(1-\alpha_t)}{1-\bar{\alpha}_t} I \\ &=: \sigma_q^2(t) I. \end{aligned}$$

With this characterization of $q(x_{t-1} \mid x_t, x_0)$, we are pretty much ready to plug into the ELBO bound in Equation (2.26). The only thing left to do is to choose what the parameterization of $p_\theta(x_{t-1} \mid x_t)$ should be. Taking inspiration from the form of $q(x_{t-1} \mid x_t, x_0)$, we choose:

$$p_\theta(x_{t-1} \mid x_t) = N(\mu_\theta(x_t, t), \sigma_q^2(t) I),$$

where

$$\mu_\theta(x_t, t) := \mu_q(f_\theta(x_t, t), x_t).$$

Here, $f_\theta(x_t, t)$ is a model which seeks to predict the original observation x_0 from the noised observation x_t at timestep t. With this choice of $p_\theta(x_{t-1} \mid x_t)$, we have that:

$$\mathbb{E}_{q(x_t|x_0)}[\mathrm{KL}(q(x_{t-1} \mid x_t, x_0) \parallel p_\theta(x_{t-1} \mid x_t))] = \frac{1}{2\sigma_q^2(t)} \mathbb{E}_{q(x_t|x_0)} \|\mu_q(x_0, x_t) - \mu_\theta(x_t, t)\|^2$$

$$= \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2 \bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2} \mathbb{E}_{q(x_t|x_0)} \|x_0 - f_\theta(x_t, t)\|^2$$
$$= \frac{(1-\alpha_t)\bar{\alpha}_{t-1}}{2(1-\bar{\alpha}_t)(1-\bar{\alpha}_{t-1})} \mathbb{E}_{q(x_t|x_0)} \|x_0 - f_\theta(x_t, t)\|^2.$$

It turns out there is an alternative parameterization that allows us to draw a connection with denoising score matching, and is also more numerically stable. The key to this is to write, using $q(x_t \mid x_0)$,

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t} w_t,$$

where w_t is $N(0, I)$ Gaussian noise independent of x_0. Now we solve the above equation for x_0:

$$x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} [x_t - \sqrt{1-\bar{\alpha}_t} w_t].$$

Plugging this form of x_0 into $\mu_q(x_0, x_t)$, we get after some algebra:

$$\mu_q(x_0, x_t) = \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}} w_t.$$

Thus, using this representation, we choose $\mu_\theta(x_t, t)$ to be:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}} f_\theta(x_t, t), \tag{2.29}$$

where $f_\theta(x_t, t)$ now predicts the *noise* added to x_0 to form x_t. With this choice of $\mu_\theta(x_t, t)$, we now have:

$$\begin{aligned}
\mathbb{E}_{q(x_t|x_0)}[\mathrm{KL}(q(x_{t-1} \mid x_t, x_0) \parallel p_\theta(x_{t-1} \mid x_t))] &= \frac{1}{2\sigma_q^2(t)} \mathbb{E}_{q(x_t|x_0)} \|\mu_q(x_0, x_t) - \mu_\theta(x_t, t)\|^2 \\
&= \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} \mathbb{E}_{w_t} \|w_t - f_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t} w_t, t)\|^2 \\
&= \frac{1-\alpha_t}{2(1-\bar{\alpha}_{t-1})\alpha_t} \mathbb{E}_{w_t} \|w_t - f_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t} w_t, t)\|^2.
\end{aligned}$$

Notice the form of this loss is precisely what occurs in denoising score matching: the denoiser is trained to predict the noise from the corrupted sample. Thus, we can interpret each one of these loss terms as learning the score function corresponding to the intermediate latent x_t.

Plugging this KL equality into Equation (2.26):

$$\begin{aligned}
\mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \right] &= \mathbb{E}_{q(x_1|x_0)}[\log p_\theta(x_0 \mid x_1)] - \mathrm{KL}(q(x_T \mid x_0) \parallel p_\theta(x_T)) \\
&\quad - \sum_{t=2}^{T} \frac{1-\alpha_t}{2(1-\bar{\alpha}_{t-1})\alpha_t} \mathbb{E}_{w_t} \|w_t - f_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t} w_t, t)\|^2.
\end{aligned} \tag{2.30}$$

Let us now discuss the first two terms which remain. First, we know that when T is large, $q(x_T \mid x_0) \sim N(0, I)$, so we can simply set $p_\theta(x_T) = N(0, I)$ (e.g., no learnable parameters). As for the first term (the reconstruction error term), there is actually some degree of freedom here in how to choose $p_\theta(x_0 \mid x_1)$. For simplicity, we consider the following parameterization:

$$p_\theta(x_0 \mid x_1) = N(\mu_\theta(x_1, 1), \sigma_1^2 I),$$

where $\sigma_1 > 0$ is also a learnable parameter. With this parameterization,

$$\log p_\theta(x_0 \mid x_1) = -\frac{1}{2\sigma_1^2} \|\mu_\theta(x_1, 1) - x_0\|^2 - \frac{d}{2} \log(2\pi\sigma_1^2)$$

$$= -\frac{1}{2\sigma_1^2}\left\|\frac{1}{\sqrt{\alpha_1}}x_1 - \frac{\sqrt{1-\alpha_1}}{\sqrt{\alpha_1}}f_\theta(x_1,1) - \frac{1}{\sqrt{\alpha_1}}[x_1 - \sqrt{1-\alpha_1}w_1]\right\|^2 - d\log\sigma_1 + \text{const}$$
$$= -\frac{1-\alpha_1}{2\sigma_1^2\alpha_1}\|w_1 - f_\theta(\sqrt{\alpha_1}x_0 + \sqrt{1-\alpha_1}w_1, 1)\|^2 - d\log\sigma_1 + \text{const}.$$

Here, the const refers to a constant that does not depend on the learnable parameters θ of p_θ. Plugging this identity into (2.30) and taking expectation w.r.t. x_0, we have that:

$$\mathbb{E}_{x_0}\mathbb{E}_{q(x_{1:T}|x_0)}\left[-\log\frac{p_\theta(x_{0:T})}{q(x_{1:T}\mid x_0)}\right] = \sum_{t=1}^{T}\frac{1-\alpha_t}{2\beta_t\alpha_t}\mathbb{E}_{(x_0,w_t)}\|w_t - f_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}w_t, t)\|^2 + d\log\sigma_1 + \text{const},$$

where $\beta_1 := \sigma_1^2$ and $\beta_t := 1 - \bar{\alpha}_{t-1}$ for $t \in \{2,\dots,T\}$. Thus, the DDPM population loss is:

$$L(\theta) := \sum_{t=1}^{T}\frac{1-\alpha_t}{2\beta_t\alpha_t}\mathbb{E}_{(x_0,w_t)}\|w_t - f_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}w_t, t)\|^2 + d\log\sigma_1.$$

Note that in practice, one often ignores the scaling $\frac{1-\alpha_t}{2\beta_t\alpha_t}$, and also discards the σ_1 parameter, yielding the approximate population loss:

$$L(\theta) = \sum_{t=1}^{T}\mathbb{E}_{(x_0,w_t)}\|w_t - f_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}w_t, t)\|^2.$$

Up to scaling, this loss matches the DDPM loss (2.27).

2.9.2 Inpainting images with DDPM

The particular form of the DDPM sampler (2.28) lends itself to some interesting extensions. We will study one particular extension, known as *image inpainting*.

The concept of image inpainting is simple. Given an image and a set of missing pixels, fill in the missing pixels with plausible values to complete the image. This is a standard problem in computer vision, but DDPM models provide a novel approach by modifying the sampling procedure. The following algorithm is due to Lugmayr et al. [2022].

First, some notation. Let x denote an image and m denote a mask. Let $x \odot m$ denote the image with the pixels *not in* the mask m zeroed-out, and let $x \odot (1-m)$ denote the image with pixels *in* the mask m zeroed-out.

Now given an image x_0^k, a mask m of valid pixels, and a DDPM model $f_\theta(x,t)$ trained over images, consider the following algorithm for generating a full image x_0 (with the missing pixels in $1-m$ filled in):

Algorithm 3 DDPM inpainting sampler

1: Set $x_T \sim N(0, I)$.
2: **for** $t = T, \dots, 1$ **do**
3: `/* Sample known values at noise level t-1. */`
4: Set $x_{t-1}^k \sim N(\sqrt{\bar{\alpha}_{t-1}}x_0^k, (1-\bar{\alpha}_{t-1})I)$.
5: `/* Denoise the unknown values to step t-1. */`
6: Set $x_{t-1}^u = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}f_\theta(x_t,t)\right) + \sigma_q(t)w_t$, $w_t \sim N(0,I)$.
7: `/* Combine the known and unknown parts together. */`
8: Set $x_{t-1} = x_{t-1}^k \odot m + x_{t-1}^u \odot (1-m)$.
9: **end for**
10: **return** x_0.

Algorithm 4 DDPM inpainting sampler with consistency resampling

Set $x_T \sim N(0, I)$.
for $t = T, \dots, 1$ **do**
  for $j = 1, \dots, U$ **do**
    Set $x_{t-1}^k \sim N(\sqrt{\bar{\alpha}_{t-1}} x_0^k, (1 - \bar{\alpha}_{t-1})I)$.
    Set $x_{t-1}^u = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} f_\theta(x_t, t) \right) + \sigma_q(t) w_t$, $w_t \sim N(0, I)$.
    Set $x_{t-1} = x_{t-1}^k \odot m + x_{t-1}^u \odot (1 - m)$.
    if $j < U$ and $t >$ **then**
      `/* Renoise back to step t. */`
      $x_t \sim N(\sqrt{\alpha_{t-1}} x_{t-1}, (1 - \alpha_{t-1})I)$.
    end if
  end for
end for
return x_0.

As written the algorithm suffers from serious artifacts in the sampling (see [Lugmayr et al., 2022, Figure 3]). However, this can be fixed by trying to enforce some notion of consistency across the known and unknown pixels at every timestep of the sampling.

This inner renoising step above forces the image to be more consistent. While the inner step adds overhead (a factor of U times more expensive than generating one DDPM sample), empirically $U = 10$ renoising steps suffice (again see [Lugmayr et al., 2022, Figure 3]).

2.10 Normalizing flows and neural ODEs

When we studied energy models, we considered modeling $p_\theta(x)$ only up to a normalization constant, i.e., $p_\theta(x) \propto \exp(f_\theta(x))$. In this section, we consider an alternative parameterization where we parameterize a properly normalized model. These types of models, which are called *normalizing flows*, are important when precise likelihood values are necessary.

The key idea behind normalizing flows is that we will represent a distribution by transforming a simple known distribution (typically a Gaussian distribution) into a more complex distribution by "pushing" the simple distribution through a continuously differentiable, invertible transformation (i.e., a *diffeomorphism*). More specifically, suppose that Z is a random variable on $\mathbb{R}^d$ which has density $p_Z(z)$. Let $f : \mathbb{R}^d \mapsto \mathbb{R}^d$ be a diffeomorphism, and define the random variable $X = f(Z)$. The density of X can be computed via the following *change of variables* formula:

$$\begin{aligned} p_X(x) &= p_Z(f^{-1}(x)) |\det[\partial f^{-1}](x)| \\ &= p_Z(f^{-1}(x)) |\det [\partial f](f^{-1}(x))|^{-1}. \end{aligned}$$

Here, the notation $[\partial f](x)$ denotes the Jacobian of f evaluated at x, and similarly $[\partial f^{-1}](x)$ denotes the Jacobian of f^{-1} evaluated at x (in particular, $[\partial f](x) \in \mathbb{R}^{d \times d}$). This formula states that to compute the density of X, we first map the input x back to z through the inverse f^{-1} and compute the density of Z. However, we need to multiply this density with a volume adjustment factor, given by the determinant of the Jacobian of the inverse map f^{-1}.

Now, given samples $x_1, \dots, x_n$, we can optimize log-likelihood in a very straightforward manner, for a given base density ϕ:

$$\min_{f \in \mathscr{F}} \hat{L}_n[f] = -\frac{1}{n} \sum_{i=1}^n \log p(x_i) = \frac{1}{n} \sum_{i=1}^n \left[-\log \phi(f^{-1}(x_i)) + \log |\det[\partial f](f^{-1}(x_i))| \right].$$

At a glance it seems like we are done, since we have a tractable way to minimize negative log-likelihood. The

trick, however, is to come up with a parameterization of diffeomorphisms $\mathscr{F}$ such that the following criteria are satisfied:

(a) **Computation:** Computing f and f^{-1}, and their Jacobians, should be efficient.

(b) **Expressivity:** $\mathscr{F}$ has to be expressive enough so that the pushforward distribution $X = f(Z)$ is sufficiently rich.

These constraints are at odds with each other. Coming up with a function class $\mathscr{F}$ which satisfies both constraints is still an open research question, although considerable progress has been made in the past decade. To focus our discussion, we will look at two examples that are essentially at the extreme ends of the trade-off curve.

2.10.1 Affine transforms

Consider the following affine transform parameterized by an invertible $d \times d$ matrix A, and a vector b:

$$f_\sigma(z) = \sigma(Az + b),$$

where σ is a fixed diffeomorphism (e.g. $\sigma(x) = x$, $\sigma(x) = x^3$ (coordinate-wise)). The inverse and Jacobian of f_σ are easily computed as

$$f_\sigma^{-1}(z) = A^{-1}[\sigma^{-1}(z) - b], \quad [\partial f_\sigma](z) = [\partial\sigma](Az + b)A.$$

Furthermore, these transforms can be composed in the following way. Suppose that $f_1, \ldots, f_k$ are k diffeomorphisms. Then the composition $f := f_k \circ f_{k-1} \circ \cdots \circ f_1$ is also a diffeomorphism, with

$$f^{-1} = f_1^{-1} \circ f_2^{-1} \circ \cdots \circ f_k^{-1}, \quad [\partial f](z) = \prod_{j=k}^{1} [\partial f_j]((f_{j-1} \circ \cdots \circ f_1)(z)).$$

If we now compose f_σ's together, this looks very much like a fully connected neural network. Some key differences, however, are:

(a) Activation functions: popular activation functions for NNs include `ReLU`, `GeLU`, `swish`, etc. Unfortunately, these standard activation functions are not invertible.

(b) Hidden unit dimension: for typical NN architectures, usually the hidden units have different dimensionality than the inputs. Being able to vary the width of the hidden units in a NN is fairly important for deep learning in practice. However, this breaks the composition of diffeomorphisms assumption.

These differences limit the expressivity of these types of diffeomorphism networks. Consequently, state of the art transformations are often more complex than the composition of affine transforms presented above.

We next turn to studying a class of diffeomorphisms which are very expressive but still admit tractable log probability computations.

2.10.2 Continuous normalizing flows

We start with the following observation. Let $h : \mathbb{R}^d \mapsto \mathbb{R}^d$ be an arbitrary continuously differentiable function, and for a fixed $\varepsilon > 0$, define the function $f_\varepsilon : \mathbb{R}^d \mapsto \mathbb{R}^d$ as:

$$f_\varepsilon(z) := z + \varepsilon h(z).$$

By the inverse function theorem, for every z, there exists a sufficiently small ε and neighborhood B of z such that the function $f_\varepsilon(z)$ is invertible over B. Now let us consider what happens if we compose f_ε with itself k times, which we denote by the notation $f_\varepsilon^{(k)}$:

$$f_\varepsilon^{(k)}(z) = z + \varepsilon h(z) + \varepsilon h(f_\varepsilon(z)) + \varepsilon h(f_\varepsilon^{(2)}(z)) + \cdots + \varepsilon h(f_\varepsilon^{(k-1)}(z)).$$

Note that as $\varepsilon \to 0$, we have that:

$$f_\varepsilon^{(k)}(x) \approx z + \int_0^{k\varepsilon} h(z(t))\,\mathrm{d}t,$$

where $z(t)$ solves the ordinary differential equation (ODE):

$$\dot{z}(t) = h(z(t)), \quad z(0) = z.$$

This motivates the use of ODEs as diffeomorphisms. Given an $f(z,t) : \mathbb{R}^d \times \mathbb{R} \mapsto \mathbb{R}^d$, initial condition $z_0 \in \mathbb{R}^d$, and times t_0, t_1, let the notation $\texttt{ODESolve}(f, z_0, t_0, t_1)$ denote the solution $z(t_1)$ of the following ODE:[12]

$$\dot{z}(t) = f(z(t), t), \quad z(t_0) = z_0.$$

Note that in our notation for `ODESolve`, we do not require that $t_0 \leqslant t_1$. In particular, integrating backwards in time shows that the solution to the ODE is invertible:

$$\texttt{ODESolve}(f, \texttt{ODESolve}(f, z_0, t_0, t_1), t_1, t_0) = z_0.$$

It turns out that $\texttt{ODESolve}(f, z_0, t_0, t_1)$ is also differentiable in z_0 (and also in t_0, t_1, but we will not need this fact).

Log probability calculation. Let us now consider using `ODESolve` as a normalizing flow. Specifically, fix a $T > 0$ and a vector field g_θ, and define $f_\theta(z) := \texttt{ODESolve}(g_\theta, z, 0, T)$. We now turn to the question of computing:

$$\log p_X(x),$$

where $X = f_\theta(Z)$ and Z is a simple distribution with known density $p_Z(z)$. It turns out we can actually write another ODE to compute this log probability. To describe this ODE, let $z(t) = \texttt{ODESolve}(g_\theta, z, 0, t)$ for $t \in [0, T]$ (so in particular, $z(0) = z$ and $z(T) = f_\theta(z)$). Let $Z(t)$ be the random variable $\texttt{ODESolve}(g_\theta, Z, 0, t)$. With this notation, we have the following result.

Proposition 2.6. *With the notation above,*

$$\frac{\partial}{\partial t} \log p_{Z(t)}(z(t)) = -\operatorname{tr}([\partial_z g_\theta](z(t), t)).$$

Armed with Proposition 2.6, we are now ready to compute $\log p_X(x)$ for a given x. The key idea is to create an *augmented ODE* which integrates backwards in time to compute $f^{-1}(x)$ and simultaneously accumulates the difference between $\log p_X(x)$ and $\log p_Z(f^{-1}(x))$ using Proposition 2.6. Specifically:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} z(t) \\ \log p_X(x) - \log p_{Z(t)}(z(t)) \end{bmatrix} = \begin{bmatrix} g_\theta(z(t), t) \\ -\operatorname{tr}([\partial_z g_\theta](z(t), t)) \end{bmatrix}, \quad \begin{bmatrix} z(T) \\ \log p_X(x) - \log p_{Z(T)}(z(T)) \end{bmatrix} = \begin{bmatrix} x \\ 0 \end{bmatrix}.$$

Hence this yields the following recipe for computing $\log p_X(x)$:

With Algorithm 5, we can evaluate the negative log-likelihood loss, given example $x_1, \ldots, x_n \in \mathbb{R}^d$:

$$\hat{L}_n[\theta] = -\frac{1}{n}\sum_{i=1}^{n} \log p_X(x_i), \quad X = f_\theta(Z). \tag{2.31}$$

But one question remains: how do we compute gradients of $\hat{L}_n[\theta]$?

[12] We will assume that the conditions for Picard's existence and uniqueness theorem hold, so that the ODEs we consider are well defined.

[13] The notation _ in the function argument of g_θ^{aug} means the same as it does in Python: the argument is not used.

Algorithm 5 Log probability computation for continuous normalizing flows

1: **Input:** $x \in \mathbb{R}^d$.
2: **Output:** $\log p_X(x)$.
3: Define the augmented dynamics $g_\theta^{\text{aug}} : \mathbb{R}^{d+1} \times \mathbb{R} \mapsto \mathbb{R}^{d+1}$ as:[13]

$$g_\theta^{\text{aug}}\left(\begin{bmatrix} z \\ \_ \end{bmatrix}, t\right) = \begin{bmatrix} g_\theta(z(t), t) \\ -\operatorname{tr}([\partial_z g_\theta](z, t)) \end{bmatrix}.$$

4: Compute (integrating *backwards* in time):

$$\begin{bmatrix} z(0) \\ \Delta \end{bmatrix} = \texttt{ODESolve}\left(g_\theta^{\text{aug}}, \begin{bmatrix} x \\ 0 \end{bmatrix}, T, 0\right).$$

5: **return** $\log p_Z(z(0)) + \Delta$.

Adjoint method for gradient calculation. The loss in (2.31) is formed by solutions to ODEs. In order to differentiate $\hat{L}_n[\theta]$, we need to differentiate through `ODESolve`. Fortunately, there is a well-known solution to this problem, which we will describe in more generality than needed for (2.31).

Let ℓ be a scalar loss function, and consider taking the gradient:

$$\nabla_\theta L[\theta], \quad L[\theta] := \ell(\texttt{ODESolve}(f_\theta, x, t_0, t_1)). \tag{2.32}$$

Here, we think of $\ell, f_\theta, x, t_0, t_1$ as being arbitrary (which we can readily instantiate for continuous normalizing flows). It turns out that there is another augmented ODE which can be used to compute the gradient above. The way this works is as follows. First, let us denote

$$x(t) = \texttt{ODESolve}(f_\theta, x, t_0, t_1), \quad t \in [t_0, t_1],$$

as the intermediate solutions of the ODE in Equation (2.32). The key quantity we need is the *adjoint* variable

$$a(t) := [\nabla_x \ell_t](x(t)), \quad \ell_t(x) := \ell(\texttt{ODESolve}(f_\theta, x, t, t_1)).$$

The dynamics of $a(t)$ can be shown to satisfy:

$$\frac{\mathrm{d}a(t)}{\mathrm{d}t} = -([\partial_x f_\theta](x(t), t))^\mathsf{T} a(t).$$

An informal derivation of this identity works as follows. Suppose we discretize the ODE solution $x(t)$ as:

$$x_{t+1} = x_t + \varepsilon f_\theta(x_t, \varepsilon t).$$

Now let us compute, as we do in the derivation of the backpropagation algorithm:

$$\frac{\partial L}{\partial x_t} = \frac{\partial L}{\partial x_{t+1}} \frac{\partial x_{t+1}}{\partial x_t} = \frac{\partial L}{\partial x_{t+1}} \left[I + \varepsilon [\partial_x f_\theta](x_t, \varepsilon t)\right].$$

Rearranging,

$$\frac{1}{\varepsilon}\left[\frac{\partial L}{\partial x_{t+1}} - \frac{\partial L}{\partial x_t}\right] = -\frac{\partial L}{\partial x_{t+1}} \cdot [\partial_x f_\theta](x_t, \varepsilon t).$$

As $\varepsilon \to 0$ this tends to the expression for $\frac{\mathrm{d}a(t)}{\mathrm{d}t}$.

The reason the adjoints $a(t)$ are useful is they allow us to compute the gradient of L as follows, via backwards in time integration:

$$\nabla_\theta L[\theta] = -\int_{t_1}^{t_0} ([\partial_\theta f_\theta](x(t), t))^\mathsf{T} a(t) \, \mathrm{d}t.$$

Again an information derivation of the above equation with the discretized ODE works as follows:

$$\frac{\mathrm{d}L}{\mathrm{d}\theta} = \sum_{t=1}^{\lfloor T/\varepsilon \rfloor} \frac{\partial L}{\partial x_t} \frac{\partial}{\partial \theta} \left[x_{t-1} + \varepsilon f_\theta(x_{t-1}, (t-1)\varepsilon) \right] = \sum_{t=1}^{\lfloor T/\varepsilon \rfloor} \varepsilon \frac{\partial L}{\partial x_t} \cdot [\partial f_\theta](x_{t-1}, (t-1)\varepsilon).$$

Taking $\varepsilon \to 0$ we obtain the integral representation.

Combining these arguments together in Algorithm 6 yields a reverse-mode derivative algorithm (which generalizes the standard backpropagation algorithm) to compute the value and gradient of L at θ.

Algorithm 6 Reverse-model differentiation using the adjoint method

Input: $\theta \in \mathbb{R}^p$.
Output: $L[\theta]$ and $\nabla_\theta L[\theta]$.
`/* Forward Pass. */`
Compute $x(t_1) = \texttt{ODESolve}(f_\theta, x, t_0, t_1) \in \mathbb{R}^d$
`/* Backwards Pass. */`
Define the augmented dynamics $f_\theta^{\text{aug}} : \mathbb{R}^{d+d+p} \times \mathbb{R} \mapsto \mathbb{R}^{d+d+p}$ as:

$$f_\theta^{\text{aug}}((x, a, \_), t) := \left(f_\theta(x,t), -([\partial_z f](x,t))^\mathsf{T} a, -([\partial_\theta f](x,t))^\mathsf{T} a \right).$$

Set augmented state initial condition $s \in \mathbb{R}^{d+d+p}$ as:

$$s = (x(t_1), [\nabla_x \ell](x(t_1)), 0).$$

Solve:

$$(\_, \_, g) = \texttt{ODESolve}(f_\theta^{\text{aug}}, s, t_1, t_0).$$

return $(\ell(x(t_1)), g)$.

2.11 Generative adversarial networks

We now study our last type of generative model, the Generative Adversarial Network (GAN). The idea is to parameterize two different models:

$$G : \mathsf{Z} \mapsto \mathsf{X} \qquad \text{(Generator)}$$
$$D : \mathsf{X} \mapsto [0,1]. \qquad \text{(Discriminator)}$$

The idea here is to play an adversarial game. The generator's job is to generate samples in X by pushing forward a simple distribution $z \sim p_z(z)$ over Z through $G(z)$ (note that, unlike with normalizing flows, we do not require that G is a diffeomorphism). The discriminator's job is to, given an input x, output 1 if the input was sampled from the true underlying distribution $p(x)$, or output 0 if it was sampled from the generator G.

Based on these definitions, GANs solve the following two-player game:

$$\min_G \max_D L(G,D) := \mathbb{E}_{x \sim p(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]. \tag{2.33}$$

To understand this objective, observe that for a fixed generator G, the objective $D \mapsto L(G,D)$ is simply the log-likelihood when treating $D(x)$ as the probability that x was sampled from $p(x)$.

Let us now try to understand some basic properties of this objective. First, for a fixed generator G, we can compute the *optimal* discriminator $D_G^*(x)$ as follows.

Proposition 2.7. *Fix a generator G and let $p_G(x)$ denote the density of the pushforward $G(z)$ with $z \sim p_z(z)$. We have that:*

$$\arg\max_D L(G,D) = D_G^*(x) := \frac{p(x)}{p(x) + p_G(x)}.$$

Proof. We have:

$$\begin{aligned} L(G, D) &= \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \\ &= \int (p(x) \log D(x) + p_G(x) \log(1 - D(x)))\, \mathrm{d}x. \end{aligned}$$

Now for $p_1, p_2 > 0$, we define

$$f(a) := p_1 \log a + p_2 \log(1 - a).$$

Computing $f'(a) = 0$:

$$0 = f'(a) = \frac{p_1}{a} - \frac{p_2}{1-a} \Longrightarrow a = \frac{p_1}{p_1 + p_2}.$$

Furthermore, one can check that $f(a)$ is concave on $(0, 1)$, and hence $f'(a) = 0$ is a maximizer. We now apply this calculation to:

$$p_1 \leftarrow p(x),\ p_2 \leftarrow p_G(x),\ a \leftarrow D(x),$$

and conclude. □

Now, we consider optimizing the generator G over $G \mapsto L(G, D^*_G)$. To do this, we define a symmetrized version of the KL-divergence.

Definition 2.8 (Jensen-Shannon Divergence)**.** *Given two measures μ, ν, let the Jensen-Shannon Divergence* $\mathrm{JSD}(\mu \parallel \nu)$ *be defined as:*

$$\mathrm{JSD}(\mu \parallel \nu) := \frac{1}{2}\mathrm{KL}(\mu \parallel (\mu + \nu)/2) + \frac{1}{2}\mathrm{KL}(\nu \parallel (\mu + \nu)/2). \tag{2.34}$$

A basic fact about KL-divergence between μ, ν is that $\mathrm{KL}(\mu \parallel \nu) = 0$ iff $\mu = \nu$. This fact immediately implies that $\mathrm{JSD}(\mu \parallel \nu) = 0$ iff $\mu = \nu$. Furthermore, by definition we see that $\mathrm{JSD}(\mu \parallel \nu) \leqslant \log 2$.

Proposition 2.9. *For a generator G, let D^*_G denote the optimal discriminator. Put*

$$G^* := \arg\min_G L(G, D^*_G).$$

We have that $p_{G^*} = p$, that is the optimal generator G^* generates the data distribution p. Furthermore, the optimal value of the game $\min_G \max_D L(G, D)$ is $-2 \log 2$.

Proof. A quick calculation yields:

$$\begin{aligned} L(G, D^*_G) &= \mathbb{E}_{x\sim p(x)}\left[\log \frac{p(x)}{p(x) + p_G(x)}\right] + \mathbb{E}_{x\sim p_G}\left[\log \frac{p_G(x)}{p(x) + p_G(x)}\right] \\ &= -2\log 2 + \mathbb{E}_{x\sim p(x)}\left[\log \frac{p(x)}{(p(x) + p_G(x))/2}\right] + \mathbb{E}_{x\sim p_G}\left[\log \frac{p_G(x)}{(p(x) + p_G(x))/2}\right] \\ &= -2\log 2 + \mathrm{KL}(p \parallel (p + p_G)/2) + \mathrm{KL}(p_G \parallel (p + p_G)/2) \\ &= -2\log 2 + 2\mathrm{JSD}(p \parallel p_G). \end{aligned}$$

By the basic properties of the Jensen-Shannon divergence, we now see that $L(G, D^*_G)$ is minimized by setting G such that $p_G = p$, which yields $L(G^*, D^*_G) = -2 \log 2$. □

Algorithm 7 summarizes how one performs a joint min/max training step of a parameterized generator G_θ and parameterized discriminator D_ϕ.

2.11.1 Difficulties of training GANs

While conceptually speaking training a GAN seems relatively straightforward (cf. Algorithm 7), there are some well-known pitfalls to be aware of with GAN training. We note that this is still an active area of research, so here we only summarize some of the key ideas.

Algorithm 7 GAN min/max training step

Input: k inner discriminator steps, current generator G_θ and discriminator D_ϕ.
Output: updated generator $G_{\theta'}$ and discriminator $D_{\phi'}$.
for $j = 1, \ldots, k$ **do**
    Sample m noise samples $\{z_1, \ldots, z_m\}$ from $p_z(z)$.
    Sample m examples $\{x_1, \ldots, x_m\}$ from the data distribution $p(x)$.
    Take a gradient *ascent* step to update the discriminator to $D_{\phi'}$, using discriminator gradient:

$$g_\phi = \nabla_\phi \left\{ \sum_{i=1}^{m} [\log D_\phi(x_i) + \log(1 - D_\phi(G_\theta(z_i)))] \right\}.$$

end for
Sample m noise samples $\{z_1, \ldots, z_m\}$ from $p_z(z)$.
Take a gradient *descent* step to update the generator to $G_{\theta'}$, using generator gradient:

$$g_\theta = \nabla_\theta \left\{ \sum_{i=1}^{m} \log(1 - D_{\phi'}(G_\theta(z_i))) \right\}.$$

return $(G_{\theta'}, D_{\phi'})$.

Vanishing gradients. One issue that arises is when the discriminator D has nearly perfect accuracy: when $x \sim p(x)$, $D(x) \approx 1$, and when $x \sim p_G(x)$, $D(x) \approx 0$. One scenario in which this tends to arise is early on in training. As an extreme example, suppose that the support of p and p_G are disjoint. Then, the optimal discriminator $D^*_G(x) = 1$ when $x \sim p(x)$ and $D^*_G(x) = 0$ when $x \sim p_G(x)$. Let us consider what happens to the generator gradients when this happens. In particular, given a set of generator parameters θ, and letting $\mathsf{sg}(\cdot)$ denote the stop gradient operator:

$$\begin{aligned}
&\nabla_\theta L(G_\theta, \mathsf{sg}(D^*_{G_\theta})) \\
&= \nabla_\theta \mathbb{E}_{z \sim p_z(z)} [\log(1 - D^*_{\mathsf{sg}(G_\theta)}(G_\theta(z)))] \\
&= -\mathbb{E}_{z \sim p_z(z)} \left[\frac{1}{1 - D^*_{G_\theta}(G_\theta(z))} [\partial_\theta G_\theta](z)^\mathsf{T} [\nabla_x D^*_{G_\theta}](G_\theta(z)) \right] \\
&= -\mathbb{E}_{z \sim p_z(z)} \left[[\partial_\theta G_\theta](z)^\mathsf{T} [\nabla_x D^*_{G_\theta}](G_\theta(z)) \right]. \qquad \text{(since the support of } p \text{ and } p_G \text{ are disjoint)}
\end{aligned}$$

Furthermore, depending on the nature of the disjointness of the supports of p and p_G, there exists an optimal $D^*_{G_\theta}$ such that $\nabla_x D^*_{G_\theta}(x) = 0$ for all x in the support of p_G. This happens, for instance, if the support of p and p_G are contained in disjoint compact sets [Arjovsky and Bottou, 2017]. If $\nabla_x D^*_{G_\theta}(x) = 0$ occurs, then we see from this calculation that the generator gradient is zero (and hence training stalls) even when the generator is nowhere near optimal (remember $p_G \neq p$ here).

Mode collapse. Model collapse happens when the generator does a good job of fooling the discriminator, but the generator does not do a good job of covering the support of the data distribution p. For instance, a GAN trained in MNIST that suffers from mode collapse might only be capable of generating one or two digits. There are a variety of reasons for mode collapse, but typically it is a symptom of imbalance in training, either through the capacity of the generator versus the discriminator, or the amount of training the discriminator receives per a fixed generator. Mathematically, for a fixed discriminator D, it is straightforward to see that the optimal generator $G^*_D = \arg\min_G L(G, D)$ is simply given by returning a point mass at the point $x \in \mathsf{X}$ which the discriminator assigns the highest probability to: $G^*_D(x) = \arg\max_x D(x)$.

2.11.2 Wasserstein GAN

The Wasserstein GAN (W-GAN) [Arjovsky et al., 2017] is designed to fix some of the shortcomings of GAN training discussed in Section 2.11.1. We saw in the proof of Proposition 2.9 that:

$$L(G, D_G^*) = 2\text{JSD}(p \parallel p_G) - 2\log 2.$$

In other words, minimizing $G \mapsto L(G, D_G^*)$ is equivalent to minimizing the Jensen-Shannon divergence between p and p_G. While this objective is minimized at setting the generator G such that $p_G = p$, the landscape of the Jensen-Shannon divergence is not necessarily the best suitable for optimization. Indeed, the issues outlined previously with vanishing gradients and mode collapse can be related to the fact that the JSD saturates at $\log 2$, which means gradient signal can be easily lost away from optimality. The W-GAN was designed to fix some of these issues with the original GAN. While we will not have time to dive into the mathematical details of why the W-GAN fixes the original GAN issues, we will still present the W-GAN algorithm.

First, we need to define the Wasserstein distance.

Definition 2.10 (Wasserstein distance)**.** *Let μ, ν be two probability distributions over a normed space X. The Wasserstein distance $W(\mu, \nu)$ is defined as:*

$$W(\mu, \nu) = \inf_{\gamma \in \Pi(\mu,\nu)} \mathbb{E}_{(X,Y)\sim\gamma} \|X - Y\|,$$

where $\Pi(\mu, \nu)$ denotes the set of couplings *between μ and ν, that is joint distributions (X, Y) over $\mathsf{X} \times \mathsf{X}$ such that $\text{Law}(X) = \mu$ and $\text{Law}(Y) = \nu$.*

The W-GAN proposes to learn the generator G so that $W(p, p_G)$ is minimized. It is not immediately clear, based on the definition of Wasserstein distance, how to do this. The key idea is to use the Kantorovich-Rubinstein dual characterization of Wasserstein distance.

Definition 2.11 (Kantorovich-Rubinstein duality)**.** *For any measures μ, ν over X, we have:*

$$W(\mu, \nu) = \sup_{f \in \mathscr{F}_{\text{Lip}}} \left\{ \mathbb{E}_{x\sim\mu}[f(x)] - \mathbb{E}_{x\sim\nu}[f(x)] \right\},$$

where $\mathscr{F}_{\text{Lip}}$ denotes the set of 1-Lipschitz functions mapping $\mathsf{X} \mapsto \mathbb{R}$.

Using Kantorovich-Rubinstein duality, the Wasserstein GAN min/max game is the following:

$$\min_G \max_f L_W(G, f) := \mathbb{E}_{x\sim p(x)}[f(x)] - \mathbb{E}_{z\sim p_z(z)}[f(G(z))].$$

This objective leads to the following Wasserstein GAN update algorithm (Algorithm 8).

Arjovsky et al. [2017, Figure 2] illustrates how the W-GAN still provides useful gradients early on in training when the generator does not match the true data distribution, but the critic (discriminator) is trained until convergence.

3 Miscellaneous topics

3.1 Conformal prediction

We now turn to the following question: given a trained machine learning model $f(x)$, can we use a small calibration dataset (say of size $n = 1000$) to calibrate the model's predictions so that we can trust the model's outputs?

Specifically, consider a model $f(x)$ which solves a multi-class classification problem over K classes. Suppose that the model's output $f(x)$ is a probability distribution over $\{1, \ldots, K\}$, so that $f(x)_i$ gives the

Algorithm 8 Wasserstein GAN min/max training step

Input: k inner discriminator steps, c clip parameter, current generator G_θ and discriminator f_ϕ.
Output: updated generator $G_{\theta'}$ and discriminator $f_{\phi'}$.
for $j = 1, \ldots, k$ **do**
  Sample m noise samples $\{z_1, \ldots, z_m\}$ from $p_z(z)$.
  Sample m examples $\{x_1, \ldots, x_m\}$ from the data distribution $p(x)$.
  Take a gradient *ascent* step to update the discriminator to $f_{\phi'}$, using discriminator gradient:

$$g_\phi = \nabla_\phi \left\{ \sum_{i=1}^{m} [f_\phi(x_i) - f_\phi(G_\theta(z_i))] \right\}.$$

  Clip the weights of ϕ' to enforce the Lipschitz constraint:

$$\phi' = \text{clip}(\phi', -c, c).$$

end for
Sample m noise samples $\{z_1, \ldots, z_m\}$ from $p_z(z)$.
Take a gradient *descent* step to update the generator to $G_{\theta'}$, using generator gradient:

$$g_\theta = \nabla_\theta \left\{ -\sum_{i=1}^{m} f_{\phi'}(G_\theta(z_i)) \right\}.$$

return $(G_{\theta'}, f_{\phi'})$.

model's belief of $\mathbb{P}(y = i \mid x)$.[14] However, there is no reason to believe that $f(x)_i$ is actually *calibrated*: $f(x)_i \neq \mathbb{P}(y = i \mid x)$ in general. How can we then trust the outputs of our model?

Conformal prediction (CP) gives us a method to address this problem with the use of a *calibration dataset* $D_c := \{(x_i, y_i)\}_{i=1}^n$.[15] We will use CP to design a set-valued function $C : \mathsf{X} \mapsto 2^K$ (which depends on the model f and the calibration set D_c), where the notation 2^K refers to the set of all subsets of $\{1, \ldots, K\}$. The property that we desire out of our set-valued C is that:

$$\mathbb{P}_{D_c}\mathbb{P}_{(x,y)}\{y \in C(x)\} \approx 1 - \alpha, \tag{3.1}$$

where $\alpha \in (0, 1)$ is a user-specified error rate.

The condition (3.1), often referred to as a *marginal coverage condition*, is by itself under-specified. In particular, here is a simple construction of $C(x)$ which achieves exact $1 - \alpha$ marginal coverage: with probability $1 - \alpha$, return $C(x) = \{1, \ldots, K\}$, and with probability α, return $C(x) = \{\}$. Of course, such a $C(x)$ is not useful, with the reason being that the set $C(x)$ is not minimal in any sense. Thus, the real goal of CP is to construct $C(x)$ to ensure marginal coverage (3.1), while simultaneously restricting $C(x)$ to be as minimal as possible. The algorithms we will discuss today empirically yield minimal sets in some sense, but quantifying the optimality of such procedure is beyond the scope of this course.

Our presentation of CP is based heavily on the excelled tutorial paper of Angelopoulos and Bates [2021]. We start with our first CP algorithm. First, a piece of notation. Given scalars $a_1, \ldots, a_m \in \mathbb{R}$ and $q \in [0, 1]$ we define quantile$(\{a_1, \ldots, a_m\}, q)$ to be the q-th quantile of $\{a_1, \ldots, a_m\}$, i.e.,

$$\text{quantile}(\{a_1, \ldots, a_m\}, q) := \inf \left\{ v \;\middle|\; \frac{|\{i \mid a_i \leqslant v\}|}{m} \geqslant q \right\}.$$

[14] Usually, $f(x)$ returns the *logits* of distribution, and the actual distribution over $\{1, \ldots, K\}$ is formed via the softmax operation. But we assume this softmax postprocessing has already been applied.

[15] Note that we will only discuss *split conformal prediction*, where the n here is not to be confused with the number of *training* points; the training dataset will make no appearance in this section. But there also exists a *full conformal prediction* procedure which makes use of all the training data.

Recall that we have a calibration dataset D_c, which the model f has *not* been trained on. For each $i = 1, \ldots, n$ in the calibration set, we define a *conformal score* $s_i \in [0, 1]$ as:

$$s_i := 1 - f(x_i)_{y_i},$$

where the notation $f(x_i)_{y_i}$ denotes the model's probability that x_i equals the true class y_i. Note that the conformal score s_i is high (close to one) when the model is wrong (i.e., it assigns low probability of x_i belonging to the true class y_i) and is low (close to zero) when the model is correct. We now define $\hat{s}$ to be:

$$\hat{s} = \text{quantile}(\{s_1, \ldots, s_m\}, \lceil (n+1)(1-\alpha) \rceil / n).$$

(Note that if $\alpha < 1/(n+1)$, we define $\hat{s} = +\infty$). Finally, we set $C(x)$ to be:

$$C(x) = \{y \mid f(x)_y \geqslant 1 - \hat{s}\}.$$

See Angelopoulos and Bates [2021, Figure 2] for an illustration of this $C(x)$. It turns out such a construction satisfies our marginal coverage condition as follow.

Proposition 3.1. *Let $C(x)$ be constructed as described above. Then,*

$$\mathbb{P}_{D_c}\mathbb{P}_{(x,y)}\{y \in C(x)\} \geqslant 1 - \alpha.$$

Proof. We assume that $\alpha \geqslant 1/(n+1)$, since otherwise $\hat{s} = \infty$ and therefore $C(x) = \{1, \ldots, K\}$, for which there is nothing to prove. Let $s_{(1)}, \ldots, s_{(n)}$ denote the conformal scores sorted in increasing order, and for simplicity let us assume that the conformal scores are disjoint with probability one. We have that:

$$\hat{s} = s_{(\lceil (n+1)(1-\alpha) \rceil)}.$$

Now given a test point (x, y), we define $s_{\text{test}} := 1 - f(x)_y$, and we observe that:

$$\begin{aligned}
\{y \in C(x)\} &= \{f(x)_y \geqslant 1 - \hat{s}\} \\
&= \{\hat{s} \geqslant 1 - f(x)_y\} \\
&= \{s_{(\lceil (n+1)(1-\alpha) \rceil)} \geqslant s_{\text{test}}\}.
\end{aligned}$$

Now here comes the important observation. Observe that s_{test} has the same distribution as $s_1, \ldots, s_n$, and all these conformal scores are iid. Thus, by exchangability, s_{test} has equal probability of landing in the $n+1$ intervals:

$$(-\infty, s_{(1)}], (s_{(1)}, s_{(2)}], \ldots, (s_{(n-1)}, s_{(n)}], (s_{(n)}, +\infty).$$

Therefore, for any $k \in \{1, \ldots, n\}$:

$$\mathbb{P}\{s_{\text{test}} \leqslant s_{(k)}\} = \frac{k}{n+1}.$$

Hence,

$$\begin{aligned}
\mathbb{P}\{y \in C(x)\} &= \mathbb{P}\{s_{\text{test}} \leqslant s_{(\lceil (n+1)(1-\alpha) \rceil)}\} \\
&= \frac{\lceil (n+1)(1-\alpha) \rceil}{n+1} \\
&\geqslant 1 - \alpha.
\end{aligned}$$

□

Now, upon examining the proof of Proposition 3.1, we see that the specific form of $s_i = 1 - f(x_i)_{y_i}$ plays essentially no role. Therefore, we can define a general scalar-valued score function $s(x, y)$ (which depends on f). The job of the score function is to assign a high value when the model's prediction $f(x)$ is incorrect, and low value when the model's prediction is correct. Based on this score function $s(x, y)$, we have the following conformal procedure:

(a) Set $s_i = s(x_i, y_i)$ for every $(x_i, y_i) \in D_c$.

(b) Set $\hat{s} = \text{quantile}(\{s_1, \ldots, s_n\}, \lceil (n+1)(1-\alpha) \rceil / n)$.

(c) Construct $C(x)$ to be:

$$C(x) = \{y \mid \hat{s} \geqslant s(x, y)\}.$$

Note that this conformal procedure has the exact same guarantee as the procedure in Proposition 3.1, using the same proof:

$$\mathbb{P}_{D_c}\mathbb{P}_{(x,y)}\{y \in C(x)\} \geqslant 1 - \alpha.$$

Adaptive prediction sets. We now consider a different calibration score function $s(x, y)$ for classification, known as *adaptive prediction sets* (APS). Empirically, it is often the case that the previous procedure does not provide very good conditional coverage. The following APS procedure, which takes into account the probabilities of the other classes $y' \neq y$, performs quite well in practice. The APS score function proposes the following score. To define it, first let $\pi(x)$ denote a permutation of $\{1, \ldots, K\}$, which orders the probabilities in $f(x)$ in decreasing order (highest to lowest). Hence, the notation $y = \pi_k(x)$ means that the true label y has the k-th most likely probability under $f(x)$. With this notation,

$$s(x, y) = \sum_{j=1}^{k} f(x)_{\pi_j(x)}, \quad y = \pi_k(x).$$

In words, this score sums together all the probabilities of the classes which have probability at least the true label, that is summing the following set:

$$\{f(x)_i \mid f(x)_i \geqslant f(x)_y\}.$$

This scoring function gives rise to the following $C(x)$:

$$C(x) = \{\pi_1(x), \ldots, \pi_k(x)\}, \quad k = \max\left\{k' \;\middle|\; \sum_{j=1}^{k'} f(x)_{\pi_j(x)} < \hat{s}\right\} + 1.$$

(Note this $C(x)$ has been slightly modified to avoid empty prediction sets $C(x)$.) See Angelopoulos and Bates [2021, Figure 4] for an illustration of $C(x)$.

3.1.1 Marginal versus conditional coverage

The coverage guarantee that

$$\mathbb{P}_{D_c}\mathbb{P}_{(x,y)}\{y \in C(x)\} \geqslant 1 - \alpha$$

is very general in that there is no assumption on the distribution (other than iid draws), and also there is no assumption on the score function $s(x, y)$. There is a price to pay for this generality though. Even assuming the score function $s(x, y)$ is well designed, the double probability $\mathbb{P}_{D_c}\mathbb{P}_{(x,y)}$ needs some careful consideration. What this says is that if one runs many experiments drawing a new calibration set D_c and a new test example (x, y), then over this joint randomness we will have $y \in C(x)$ holding at least $1 - \alpha$ of the time. But what it does *not* imply is that if I hold my calibration set D_c fixed, that $\mathbb{P}_{(x,y)}\{y \in C(x)\} \geqslant 1 - \alpha$. What if we want a guarantee that holds conditionally on D_c?

It turns out that for every D_c [Vovk, 2012]:

$$\mathbb{P}_{(x,y)}\{y \in C(x) \mid D_c\} \sim \text{Beta}(n + 1 - \ell, \ell), \quad \ell := \lfloor (n+1)\alpha \rfloor.$$

Hence, if we set $\alpha = 0.9$ and $n = 1000$, we have that:

$$\mathbb{P}_{(x,y)}\{y \in C(x) \mid D_c\} \geqslant 0.88.$$

This addresses removing the outer probability over D_c in the coverage statement. However, a subtle issue remains. The remaining statement takes a joint probability over (x, y). Again, this means that if we run many trials where we draw new data points x and labels y, that our coverage condition will hold jointly over these (x, y) draws. But in many applications, what we really want is to condition on x as well, i.e.,

$$\mathbb{P}\{y \in C(x) \mid D_c, x\} \geqslant 1 - \alpha.$$

Above, the remaining probability is over the label y conditioned on x and the calibration set D_c. Essentially this says that for a specific example x, the coverage guarantee holds. See Angelopoulos and Bates [2021, Figure 10] for an illustration of marginal versus conditional coverage.

Unfortunately, in the general case when x is continuous it turns out that achieving this guarantee in general is not possible [Vovk, 2012]. However, there are some special cases that can be handled. The most simple one is to partition the feature space x into G groups $\{1, \ldots, G\}$. Let $h(x) \in \{1, \ldots, G\}$ be the function which partitions the space. Then, we can achieve the following conditional group coverage guarantee:

$$\forall g \in \{1, \ldots, G\},\ \mathbb{P}\{y \in C(x) \mid h(x) = g\} \geqslant 1 - \alpha,$$

by simply running conformal prediction for each group g separately. Specifically, given the calibration set D_c, we first partition D_c into its groups using $h(x)$, and then for each group we compute a separate threshold $\hat{s}^{(g)}$. Then, at test time, we simply set:

$$C(x) = \{y \mid \hat{s}^{(h(x))} \geqslant s(x, y)\}.$$

3.1.2 Uncertainty intervals for regression

We now discuss how to build uncertainty intervals for regression. We consider the case when $f(x) \in \mathbb{R}$ is a real-valued prediction. In this case, our prediction set $C(x)$ is a subset of $\mathbb{R}$ (usually a contiguous interval).

Quantile regression. Recall that for square loss, we have that the optimal solution to the MSE is the conditional mean:

$$\arg \min_{f : \mathsf{X} \mapsto \mathbb{R}} \mathbb{E}(f(x) - y)^2 = \mathbb{E}[y \mid x].$$

We can generalize this to quantiles of the conditional distribution of $\mathbb{P}(y \mid x)$. For $\gamma \in (0, 1)$, define the *pinball loss* ℓ_γ as:

$$\ell_\gamma(s) := \begin{cases} \gamma s & \text{if } s > 0, \\ -(1 - \gamma)s & \text{if } s \leqslant 0. \end{cases}$$

We now consider what $\arg\min_{f : \mathsf{X} \mapsto \mathbb{R}} \mathbb{E}[\ell_\gamma(f(x) - y)]$ solves for. To do this, we need some notation. For any distribution μ over $\mathbb{R}$, let $F_\mu(s) := \mathbb{P}_{x \sim \mu}\{x \leqslant s\}$ denote the CDF. With this notation, we have the following fact.

Proposition 3.2. *Suppose that $\mathbb{P}(y \mid x)$ has an invertible CDF almost surely. We have that:*

$$\arg \min_{f : \mathsf{X} \mapsto \mathbb{R}} \mathbb{E}[\ell_\gamma(y - f(x))] = F^{-1}_{\mu_x}(\gamma), \quad \mu_x = \mathbb{P}(y \mid x).$$

Proof. By the tower property:

$$\mathbb{E}[\ell_\gamma(y - f(x))] = \mathbb{E}[\mathbb{E}[\ell_\gamma(y - f(x)) \mid x]].$$

Hence, it suffices to solve the following problem, for an arbitrary distribution μ over $\mathbb{R}$:

$$\arg \min_{f \in \mathbb{R}} \varphi(f) := \mathbb{E}_{y \sim \mu}[\ell_\gamma(y - f)].$$

We observe that $\varphi(f)$ is a convex function, so we simply take the derivative of $\varphi(f)$ and set it equal to zero:[16]

$$\begin{aligned}\varphi'(f) &= \mathbb{E}_{y\sim\mu}[\ell'_\gamma(y-f)] \\ &= \mathbb{E}_{y\sim\mu}[(-1)\gamma\mathbf{1}\{y-f>0\} + (1-\gamma)\mathbf{1}\{y-f\leqslant 0\}] \\ &= -\gamma\mathbb{P}_{y\sim\mu}\{y>f\} + (1-\gamma)\mathbb{P}_{y\sim\mu}\{y\leqslant f\} \\ &= -\gamma(1-F_\mu(f)) + (1-\gamma)F_\mu(f).\end{aligned}$$

Setting $\varphi'(f)=0$, we have

$$F(f)=\gamma \Longrightarrow f = F^{-1}(\gamma).$$

□

Conformalized quantile regression. From our discussion of quantile regression, we see that learning a model $f_{\alpha/2}$ minimizing $\ell_{\alpha/2}$, combined with learning a model $f_{1-\alpha/2}$ minimizing $\ell_{1-\alpha/2}$ will give the prediction interval $[f_{\alpha/2}(x), f_{1-\alpha/2}(x)]$ the right asymptotic coverage:

$$\mathbb{P}\{y\in[f_{\alpha/2}(x), f_{1-\alpha/2}(x)] \mid x\} \to 1-\alpha,$$

as the amount of training data goes to ∞. However, with finite samples, this interval will not necessarily be correctly calibrated. Hence, we can, as we did for classification, use a conformal procedure to calibrate the interval. We consider the score $s(x,y)$ to be:

$$s(x,y) := \max\{f_{\alpha/2}(x)-y, y-f_{1-\alpha/2}(x)\}.$$

This score measures the maximum violation of y below the lower quantile $f_{\alpha/2}(x)$, and y above the upper quantile $f_{1-\alpha/2}(x)$. With this score, our conformal procedure is:

(a) Set $s_i = s(x_i, y_i)$ for $(x_i, y_i) \in D_c$.

(b) Set $\hat{s} = \text{quantile}(\{s_1,\ldots,s_n\}, \lceil (n+1)(1-\alpha)\rceil/n)$.

(c) Set $C(x)$ to be the interval:

$$C(x) = [f_{\alpha/2}(x)-\hat{s}, f_{1-\alpha/2}(x)+\hat{s}].$$

3.2 Influence functions

We now turn to the topic of understanding the predictions of black-box models. The key question we want to address is as follows:

Which training datapoints are the most influential for this model?

An understanding of this question would ideally allow us to better understand, given a particular prediction, why the model made such a prediction. We consider a line of work to answer this question based on *influence functions*. While influence functions can trace back their roots to robust statistics, the work of Koh and Liang [2017] is credited with bringing this literature to the machine learning community, and these notes are based heavily on this paper. The main idea is to answer to the following counterfactual question:

How would the model's predictions change if this particular training datapoint were to be removed from the training set?

[16] Technically, $\varphi(f)$ is not differentiable at zero. But because $\varphi(f)$ is convex, we can set the *sub-gradient* equal to zero. We will ignore this technicality.

There is a simple algorithm which answers this question: letting $D = \{(x_i, y_i)\}_{i=1}^n$ denote the training dataset, and letting $D^{\neg i} = \{(x_j, y_j)\}_{j \neq i}$ denote the training dataset with the i-th point removed, simply learn n models $f^{\neg i}$ trained on $D^{\neg i}$ and compare with the original model f trained on all of D. Of course, the computational cost of this algorithm is prohibitive in general. Influence functions give us a way of approximating this computational expensive procedure in a much more tractable way.

Let us establish some notation. We have a twice-differentiable loss function $\ell : \mathsf{Z} \times \Theta \mapsto \mathbb{R}$ and a training dataset $D = \{z_i\}_{i=1}^n$. We consider models trained with standard ERM:

$$\hat{\theta} \in \arg\min_{\theta \in \Theta} \left\{ \sum_{i=1}^n \ell(z_i, \theta) \right\}, \quad \hat{\theta}^{\neg i} \in \arg\min_{\theta \in \Theta} \left\{ \sum_{j \neq i} \ell(z_j, \theta) \right\}.$$

Now, given an $\varepsilon \in \mathbb{R}$ and $z \in \mathsf{Z}$, we consider the following *perturbation* to the ERM solution $\hat{\theta}$:

$$\hat{\theta}_{\varepsilon,z} \in \arg\min_{\theta \in \Theta} \left\{ \frac{1}{n} \sum_{i=1}^n \ell(z_i, \theta) + \varepsilon \ell(z, \theta) \right\}. \tag{3.2}$$

Note that removing the i-th datapoint and retraining the model can be achieved by setting $\varepsilon = -1/n$ and $z = z_i$. in (3.2), i.e., $\hat{\theta}^{\neg i} = \hat{\theta}_{-1/n, z_i}$. Hence, by a first-order Taylor series approximation:

$$\hat{\theta}^{\neg i} \approx \hat{\theta} - \frac{1}{n} \frac{\partial \hat{\theta}_{\varepsilon, z_i}}{\partial \varepsilon}\Big|_{\varepsilon=0}.$$

Thus, if we can compute the derivative of the solution $\hat{\theta}_{\varepsilon,z}$ of (3.2) with respect to ε, then we can use this approximation to approximate how the model changes when removing the i-th datapoint. Furthermore, by the chain rule, for any $F : \Theta \mapsto \mathbb{R}$,

$$F(\hat{\theta}^{\neg i}) \approx F(\hat{\theta}) - \frac{1}{n} \left\langle \nabla F(\hat{\theta}), \frac{\partial \hat{\theta}_{\varepsilon, z_i}}{\partial \varepsilon}\Big|_{\varepsilon=0} \right\rangle. \tag{3.3}$$

We now show how to compute $\frac{\partial \hat{\theta}_{\varepsilon,z}}{\partial \varepsilon}\Big|_{\varepsilon=0}$.

Proposition 3.3. *Suppose that $\hat{\theta}$ is a strict optimizer.*[17] *Then,*

$$\frac{\partial \hat{\theta}_{\varepsilon,z}}{\partial \varepsilon}\Big|_{\varepsilon=0} = -H(\hat{\theta})^{-1} \nabla_\theta \ell(z, \hat{\theta}), \quad H(\theta) := \frac{1}{n} \sum_{i=1}^n \nabla_\theta^2 \ell(z_i, \theta).$$

Proof. The rigorous proof of this statement uses the implicit function theorem. We give a heuristic proof which conveys the idea. Since $\hat{\theta}_{\varepsilon,z}$ is optimal, we have that the gradient of the objective is stationary:

$$0 = \frac{1}{n} \sum_{i=1}^n \nabla_\theta \ell(z_i, \hat{\theta}_{\varepsilon,z}) + \varepsilon \nabla_\theta \ell(z, \hat{\theta}_{\varepsilon,z}).$$

Taking the derivative of both sides w.r.t. ε:

$$0 = \frac{1}{n} \sum_{i=1}^n \nabla_\theta^2 \ell(z_i, \hat{\theta}_{\varepsilon,z}) \frac{\partial \hat{\theta}_{\varepsilon,z}}{\partial \varepsilon} + \varepsilon \nabla_\theta^2 \ell(z, \hat{\theta}_{\varepsilon,z}) \frac{\partial \hat{\theta}_{\varepsilon,z}}{\partial \varepsilon} + \nabla_\theta \ell(z, \hat{\theta}_{\varepsilon,z}).$$

Evaluating this expression at $\varepsilon = 0$, we obtain:

$$H(\hat{\theta}) \frac{\partial \hat{\theta}_{\varepsilon,z}}{\partial \varepsilon}\Big|_{\varepsilon=0} = \left[\frac{1}{n} \sum_{i=1}^n \nabla_\theta^2 \ell(z_i, \hat{\theta}) \right] \frac{\partial \hat{\theta}_{\varepsilon,z}}{\partial \varepsilon}\Big|_{\varepsilon=0} = -\nabla_\theta \ell(z, \hat{\theta})$$

Since $\hat{\theta}$ is a strict optimizer, $H(\hat{\theta})$ is invertible. The claim follows. □

[17] This means there exists a ball $B \subseteq \Theta$ containing $\hat{\theta}$ such that for every $\theta' \in B$ with $\theta' \neq \theta$, we have $\sum_{i=1}^n \ell(z_i, \theta') > \sum_{i=1}^n \ell(z_i, \hat{\theta})$.

Plugging Proposition 3.3 into Equation (3.3):

$$F(\hat{\theta}^{\neg i}) \approx F(\hat{\theta}) + \frac{1}{n}\nabla F(\hat{\theta})^\intercal H(\hat{\theta})^{-1}\nabla_\theta \ell(z_i, \hat{\theta}).$$

Specifically, if $F(\theta) = \ell(z, \theta)$, measuring the loss of model θ on a test point z,

$$\ell(z, \hat{\theta}^{\neg i}) \approx \ell(z, \hat{\theta}) + \frac{1}{n}\nabla_\theta \ell(z, \hat{\theta})^\intercal H(\hat{\theta})^{-1}\nabla_\theta \ell(z_i, \hat{\theta}). \tag{3.4}$$

Equation (3.4) is the starting point for several applications of influence functions. It answers the question of, if one removes datapoint z_i from training and retrains the models, how would its performance on an arbitrary test point z change. Furthermore, it does not require one to actually retrain the model on the dataset $D^{\neg i}$ to answer the question.

Perturbing a training datapoint. Previously, we considered the effect of removing a training datapoint z_i. What if we now consider, instead of removing a training datapoint, instead perturbing the training point $z_i \mapsto z_\delta$:

$$\hat{\theta}_{z_i \mapsto z_\delta} \in \arg\min_{\theta \in \Theta} \left\{ \sum_{j \neq i} \ell(z_j, \theta) + \ell(z_\delta, \theta) \right\}.$$

We will define another perturbed ERM solution:

$$\hat{\theta}_{\varepsilon, z \mapsto z_\delta} \in \arg\min_{\theta \in \Theta} \left\{ \frac{1}{n}\sum_{i=1}^{n} \ell(z_i, \theta) + \varepsilon(\ell(z_\delta, \theta) - \ell(z, \theta)) \right\}.$$

Note that similar to before, $\hat{\theta}_{z_i \mapsto z_\delta} = \hat{\theta}_{1/n, z_i \mapsto z_\delta}$, and therefore

$$\hat{\theta}_{z_i \mapsto z_\delta} \approx \hat{\theta} + \frac{1}{n}\frac{\partial \hat{\theta}_{\varepsilon, z_i \mapsto z_\delta}}{\partial \varepsilon}\Big|_{\varepsilon=0}.$$

Also as before, we can characterize $\frac{\partial \hat{\theta}_{\varepsilon, z \mapsto z_\delta}}{\partial \varepsilon}\Big|_{\varepsilon=0}$ quite easily.

Proposition 3.4. *Suppose that $\hat{\theta}$ is a strict optimizer. Then,*

$$\frac{\partial \hat{\theta}_{\varepsilon, z \mapsto z_\delta}}{\partial \varepsilon}\Big|_{\varepsilon=0} = -H(\hat{\theta})^{-1}(\nabla_\theta \ell(z_\delta, \hat{\theta}) - \nabla_\theta \ell(z, \hat{\theta})).$$

Proof. The proof is nearly identical to Proposition 3.3, so we omit it. □

We will use this fact later in an application for constructing adversarial training dataset perturbations.

3.2.1 Computational considerations and non-optimality of models

Influence functions, while much more efficient than repeated training the model on leave-one-out datasets, does unfortunately come with its own set of computational challenges.

Computing $H(\hat{\theta})^{-1}\nabla_\theta \ell(z, \hat{\theta})$. A main part of influence functions computation is to calculate the expression $H(\hat{\theta})^{-1}\nabla_\theta \ell(z, \hat{\theta})$. The naïve way to do this is to first materialize the Hessian $H(\hat{\theta})$, invert it, and then compute the final matrix-vector product. This is unfortunately not practical in many situations: for example, the Hessian requires $O(p^2)$ space to store, which is not tractable for large models. However, this is actually a

well-studied problem in the optimization literature. The key insight is to realize that for a positive definite matrix M and vector v,

$$M^{-1}v = \arg\min_x \left\{ \frac{1}{2} x^\mathsf{T} M x - x^\mathsf{T} v \right\}.$$

Therefore, an iterative algorithm, such as the conjugate gradient (CG) algorithm, can be run on the loss function above to obtain $M^{-1}v$. The reason this is more efficient is that only matrix-vector products are ever needed to compute the gradient of the quadratic objective above. Hence, if we set $M = H(\hat{\theta})$, then these matrix-vector products become *Hessian-vector-products*, which can be efficiently implemented *without* needing to materialize the Hessian.[18]

Non-optimality of $\hat{\theta}$. The bigger issue regarding influence function computation is that the derivation in Proposition 3.3 assumes that $\hat{\theta}$ is a strict optimizer of the objective. In practice, however, especially for deep networks, models are not trained to optimality, nor even stationarity. Thus, Proposition 3.3 is often invalid. The way this manifests itself is that the Hessian $H(\hat{\theta})$ may not be invertible, or have negative eigenvalues. A simple heuristic to fix this issue is to instead compute:

$$\frac{\partial \hat{\theta}_{\varepsilon,z}}{\partial \varepsilon}\Big|_{\varepsilon=0} = -(H(\hat{\theta}) + \lambda I)^{-1} \nabla_\theta \ell(z, \hat{\theta}),$$

where $\lambda > 0$ is selected to ensure that $H(\hat{\theta}) + \lambda I$ is positive definite. This heuristic, however, performs quite poorly in practice. It turns out that a Gauss-Newton approximation works better. To state this approximation, we need to put some structure into our loss. Suppose that $\ell(z, \theta) = \psi(f_\theta(x), y)$, where $z = (x, y)$. The Gauss-Newton approximation to $H(\hat{\theta})$ is:

$$G(\hat{\theta}) := \frac{1}{n} \sum_{i=1}^{n} (\partial_\theta f_{\hat{\theta}}(x_i))^\mathsf{T} \nabla^2_{\hat{y}} \psi(f_{\hat{\theta}}(x_i), y_i)(\partial_\theta f_{\hat{\theta}}(x_i)).$$

Here, $\nabla^2_{\hat{y}} \psi(\hat{y}, y)$ denotes the Hessian of the loss function w.r.t. the first argument. In particular, if the function $\hat{y} \mapsto \psi(\hat{y}, y)$ is convex, then the Gauss-Newton approximation $G(\hat{\theta})$ is always positive-semidefinite (even when $H(\hat{\theta})$ is not). With the Gauss-Newton approximation, we have the following approximation for the influence $\frac{\partial \hat{\theta}_{\varepsilon,z}}{\partial \varepsilon}\Big|_{\varepsilon=0}$:

$$\frac{\partial \hat{\theta}_{\varepsilon,z}}{\partial \varepsilon}\Big|_{\varepsilon=0} = -(G(\hat{\theta}) + \lambda I)^{-1} \nabla_\theta \ell(z, \hat{\theta}),$$

Bae et al. [2022] go into depth about understanding what the Gauss-Newton approximation is computing the influence of.

3.2.2 Applications of influence functions

With our influence function machinery in place we now work through a few examples from Koh and Liang [2017].

Most helpful training point for test prediction. Given a test example $z \in \mathsf{Z}$, the model $\hat{\theta}$ incurs loss $\ell(z, \hat{\theta})$ on the test example z (think of this as a prediction error loss). The question we answer now is: which training datapoint $\{z_1, \ldots, z_n\}$ has the most influence on the value of $\ell(z, \hat{\theta})$? An answer to this question has several uses. For example, suppose that $\ell(z, \hat{\theta})$ is small, so the model's prediction is accurate. Computing influence can help us assess if the model is overfitting or generalizing. On the flip side, suppose that $\ell(z, \hat{\theta})$

[18] See e.g. `https://iclr-blogposts.github.io/2024/blog/bench-hvp/`.

is large, meaning the model's prediction is inaccurate. Then computing influence can help to assess what training data the model is possibly being confused by, or what features the model is erroneously relying on.

To do this, we rely on Equation (3.4), which states that:

$$\ell(z, \hat{\theta}^{\neg i}) - \ell(z, \hat{\theta}) \approx \frac{1}{n} \nabla_\theta \ell(z, \hat{\theta})^\intercal H(\hat{\theta})^{-1} \nabla_\theta \ell(z_i, \hat{\theta}), \quad i \in \{1, \dots, n\}.$$

The algorithm is as follows:

(a) For every z_i, $i = 1, \dots, n$ in the training data, compute:

$$\Delta_i := \frac{1}{n} \nabla_\theta \ell(z, \hat{\theta})^\intercal H(\hat{\theta})^{-1} \nabla_\theta \ell(z_i, \hat{\theta}).$$

To do this efficiently, first compute $s_{\text{test}} = H(\hat{\theta})^{-1} \nabla_\theta \ell(z, \hat{\theta})$ (possibly using conjugate gradients or another iterative procedure), and then compute $\Delta_i = \langle s_{\text{test}}, \nabla_\theta \ell(z_i, \hat{\theta}) \rangle / n$.

(b) Compute either the highest and/or lowest values of Δ_i:

$$i^+ := \arg \max_{i \in \{1, \dots, n\}} \Delta_i, \quad i^- := \arg \min_{i \in \{1, \dots, n\}} \Delta_i.$$

The way to interpret i^+ and i^- is as follows. We start with i^+. Generally speaking, we expect that Δ_{i^+} is positive. With these sign assumption, we can interpret the example z_{i^+} to be such that if it were to be *removed* from the training dataset ($\hat{\theta}^{\neg i^+}$), this causes the loss $\ell(z, \hat{\theta})$ to increase the most relative to the removal operation. Thus, z_{i^+} can be understood to be the datapoint that is the most helpful to the model $\hat{\theta}$ in making the prediction on z. On the other hand, if Δ_{i^+} is negative, then this alerts us to a potential issue, since its interpretation is that by removing any training datapoint, the models loss $\ell(z, \hat{\theta}^{\neg i})$ *decreases*. This suggests that there is an issue in either the (a) influence function approximations, (b) the example z is out of distribution, or (c) the model has overfit.

On the flip side let us now consider i^-. Let us first consider if i^- is negative. In this case, this tells us that example z_{i^-}, if removed, would maximally *decrease* $\ell(z, \hat{\theta}^{\neg i})$ amongst all $i \in \{1, \dots, n\}$. Intuitively, we would imagine if we were say solving a classification problem, then i^- would correspond to an example right along the decision boundary. Finally, if i^- is positive, then this tells us that every training example is useful in predicting z.

Koh and Liang [2017, Figure 4] works through a specific example of this application.

Adversarial training set generation. We now consider the use of influence function machinery to generate adversarial perturbations to the training set which will influence the model's predictions on a test point z. Here, we assume that $z = (x, y)$. Let δ be a perturbation to x. We are interested in solving for the following perturbation:

$$\arg \max_{\|\delta\| \leqslant 1} \ell(z, \hat{\theta}_{z_i \mapsto z_\delta}), \quad z_\delta = (x_i + \delta, y_i), \quad i = 1, \dots, n.$$

Using Proposition 3.4,

$$\begin{aligned}
\ell(z, \hat{\theta}_{z_i \mapsto z_\delta}) &\approx \ell(z, \hat{\theta}) - \frac{1}{n} \left\langle \nabla_\theta \ell(z, \hat{\theta}), H(\hat{\theta})^{-1} (\nabla_\theta \ell(z_\delta, \hat{\theta}) - \nabla_\theta \ell(z_i, \hat{\theta})) \right\rangle \\
&= \ell(z, \hat{\theta}) - \frac{1}{n} \left\langle \nabla_\theta \ell(z, \hat{\theta}), H(\hat{\theta})^{-1} (\nabla_\theta \ell((x_i + \delta, y_i), \hat{\theta}) - \nabla_\theta \ell((x_i, y_i), \hat{\theta})) \right\rangle \\
&\approx \ell(z, \hat{\theta}) - \frac{1}{n} \left\langle \nabla_\theta \ell(z, \hat{\theta}), H(\hat{\theta})^{-1} \partial_x \nabla_\theta \ell((x_i, y_i), \hat{\theta}) \delta \right\rangle.
\end{aligned}$$

Hence, the approximate solution to the maximal direction is to set δ to be:

$$\delta_z(x_i, y_i) := \frac{[\partial_x \nabla_\theta \ell((x_i, y_i), \hat{\theta})]^\mathsf{T} H(\hat{\theta})^{-1} \nabla_\theta \ell(z, \hat{\theta})}{\|[\partial_x \nabla_\theta \ell((x_i, y_i), \hat{\theta})]^\mathsf{T} H(\hat{\theta})^{-1} \nabla_\theta \ell(z, \hat{\theta})\|}.$$

With this direction δ_z, we can run the following sign gradient algorithm to generate an adversarial perturbation. In particular, we set $\tilde{x}_i = x_i$ to start with, and we iterate the map

$$\tilde{x}_i \leftarrow \Pi_{x_i}(\tilde{x}_i + \alpha \operatorname{sgn}(\delta_z(\tilde{x}_i, y_i))),$$

where $\alpha > 0$ is a step size, and $\Pi_{x_i}(\cdot)$ projects the argument so that the perturbed $\tilde{x}_i$ is bounded in the amount of corruption allowed (e.g., if x_i is an image then Π_{x_i} can be the set of all images with the same 8-bit representation as x_i). Training a model $\tilde{\theta}$ on this dataset with $(\tilde{x}_i, y_i)$ in place of (x_i, y_i), we can then see if the model's prediction $\ell(z, \tilde{\theta})$ is substantially different from $\ell(z, \hat{\theta})$; for instance for a classification problem, whether or not $\tilde{\theta}$ predicts a different class on the test example z than $\hat{\theta}$. Koh and Liang [2017, Figure 5] shows the result of running such an experiment to create adversarial perturbations to training data on an image classification model.

Outlier/mislabeled data detection. The final application we discuss is a simple modification to the first application. Here, we compute the influence each training point z_i has on its own prediction $\ell(z_i, \hat{\theta})$. Specifically,

$$\ell(z_i, \hat{\theta}^{\neg i}) - \ell(z_i, \hat{\theta}) \approx \frac{1}{n} \nabla_\theta \ell(z_i, \hat{\theta})^\mathsf{T} H(\hat{\theta})^{-1} \nabla_\theta \ell(z_i, \hat{\theta}), \quad i \in \{1, \ldots, n\}.$$

Here, the RHS is guaranteed to be non-negative. The proposal heuristic is that training examples which, if you were to remove them from the dataset, cause the largest change in prediction error *on that datapoint*, are candidates for further manual inspection for outlier/mislabeled data. While this is simply a heuristic, Koh and Liang [2017, Figure 6] shows that it works well for a spam classification problem where a fraction of the training data labels are corrupted.

3.3 Domain adaptation

One of the core tenants of the learnability of supervised learning problems is that the distribution that the training data is drawn from equals the distribution that the model will be deployed on. When this assumption is violated, our theory is invalid. However, in many situations, this assumption is not valid:

(a) There may be a sampling bias that happens in data collection versus the real world.

(b) Collecting training data for another task may be very expensive: for example if you want to solve a particular image classification problem, you would most likely want to use a pre-existing model as a starting point.

(c) Domain shifts over time: data collected for a specific task may become outdated over time, or the specific features of the task may adapt over time. An example is again in image classification, where the quality of camera lenses get better over time; a model trained on a corpus of 2014 images will experience a domain shift when applied to images taken from the most up-to-date smartphone.

(d) *Sim2real* in robotics/reinforcement learning, where you train a model on a simulator and deploy it on a real robot– a simulator can never fully capture the real world dynamics.

Our desiderata is that training a model on a *source* domain and deploying it on a *target* domain, which is not necessarily the same as the source, still provides non-trivial generalization. This idea is referred to as domain adaptation. There is a rich literature here that we will not be able to fully cover. Today, we will focus on a set of key results from Ben-David et al. [2010].

To develop a theory for domain adaption, we first introduce notation setup. Let X denote the covariate domain, and $\mathsf{Y} = \{\pm 1\}$ (binary classification). A domain $(\mathcal{D}, f)$ is parameterized by a distribution $\mathcal{D}$ over X, and a target function $f : \mathsf{X} \mapsto \{\pm 1\}$ (for simplicity in these notes we assume deterministic labeling functions; more generally instead of the target function f we parameterize a domain with a conditional label distribution $p(y \mid x)$). The *source domain* is denoted $(\mathcal{D}_S, f_S)$, and the *target domain* is denoted $(\mathcal{D}_T, f_T)$. As before, our notion of risk will be the zero-one error. We define source and target risks:

$$\ell_S(h) := \mathbb{P}_{x\sim\mathcal{D}_S}\{h(x) \neq f_S(x)\}, \quad \ell_T(h) := \mathbb{P}_{x\sim\mathcal{D}_T}\{h(x) \neq f_T(x)\}.$$

Throughout our development, we fix a hypothesis class $\mathscr{H}$ which we will search for hypothesis over. Note that we do not assume $f_S, f_T \in \mathscr{H}$. For simplicity, we will assume that $|\mathscr{H}| < \infty$. However, all the results presented here can be readily generalized to $\mathscr{H}$ with finite VC-dimension.

Measures of source/target discrepancy. Without any assumption relating the source domain $(\mathcal{D}_S, f_S)$ to the target domain $(\mathcal{D}_T, f_T)$, then we cannot hope for any non-trivial generalization. Therefore, we need to introduce some measure of discrepancy. Our main measure of discrepancy will be the following divergence:

$$d_{\mathscr{H}}(\mathcal{D}_S, \mathcal{D}_T) := 2 \sup_{h,h'\in\mathscr{H}} \left|\mathbb{P}_{x\sim\mathcal{D}_S}\{h(x) \neq h'(x)\} - \mathbb{P}_{x\sim\mathcal{D}_T}\{h(x) \neq h'(x)\}\right|.$$

Furthermore, we will define a notion of an *ideal joint hypothesis*:

$$h_\star \in \arg\min_{h\in\mathscr{H}} \left\{\ell_S(h) + \ell_T(h)\right\}, \quad \lambda_\star := \ell_S(h_\star) + \ell_T(h_\star).$$

This hypothesis $h_\star$ performs the best in $\mathscr{H}$ jointly over the source and target domains, and obtains joint risk $\lambda_\star$.

3.3.1 Non-adaptive classification

We now prove a deterministic result relating the performance of a hypothesis h under the target domain $\ell_T(h)$ with its source domain performance $\ell_S(h)$.

Proposition 3.5. *Fix an $h \in \mathscr{H}$. We have:*

$$\ell_T(h) \leqslant \ell_S(h) + \gamma_\star + \frac{1}{2} d_{\mathscr{H}}(\mathcal{D}_S, \mathcal{D}_T).$$

Proof. Note that for scalars $a, b \in \{\pm 1\}$, we have:

$$\mathbf{1}\{a \neq b\} = \frac{1}{2}|a - b|.$$

Hence, for any distribution $\mathcal{D}$ over X and labeling functions f, g,

$$\mathbb{P}_{x\sim\mathcal{D}}\{f(x) \neq g(x)\} = \frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}}|f(x) - g(x)|.$$

Therefore,

$$\begin{aligned}
\ell_T(h) &= \mathbb{P}_{x\sim\mathcal{D}_T}\{h(x) \neq f_T(x)\} \\
&= \frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}_T}|h(x) - f_T(x)| \\
&\leqslant \frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}_T}|h(x) - h_\star(x)| + \frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}_T}|h_\star(x) - f_T(x)| \\
&= \frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}_S}|h(x) - h_\star(x)| + \frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}_T}|h_\star(x) - f_T(x)|
\end{aligned}$$

$$
\begin{aligned}
&\quad + \left(\frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}_T}|h(x) - h_\star(x)| - \frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}_S}|h(x) - h_\star(x)|\right) \\
&\leqslant \frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}_S}|h(x) - h_\star(x)| + \frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}_T}|h_\star(x) - f_T(x)| \\
&\quad + \sup_{h,h'\in\mathscr{H}} |\mathbb{P}_{x\sim\mathcal{D}_T}\{h(x) \neq h'(x)\} - \mathbb{P}_{x\sim\mathcal{D}_S}\{h(x) \neq h'(x)\}| \\
&= \frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}_S}|h(x) - h_\star(x)| + \frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}_T}|h_\star(x) - f_T(x)| + \frac{1}{2}d_{\mathscr{H}}(\mathcal{D}_S, \mathcal{D}_T) \\
&\leqslant \frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}_S}|h(x) - f_S(x)| + \frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}_S}|h_\star(x) - f_S(x)| + \frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}_T}|h_\star(x) - f_T(x)| + \frac{1}{2}d_{\mathscr{H}}(\mathcal{D}_S, \mathcal{D}_T) \\
&= \ell_S(h) + \ell_S(h_\star) + \ell_T(h_\star) + \frac{1}{2}d_{\mathscr{H}}(\mathcal{D}_S, \mathcal{D}_T) \\
&= \ell_S(h) + \gamma_\star + \frac{1}{2}d_{\mathscr{H}}(\mathcal{D}_S, \mathcal{D}_T).
\end{aligned}
$$

□

3.3.2 Adaptive classification

Now suppose that, in addition to training examples $\{(x_i, y_i)\}_{i=1}^{n_S}$ from the source domain $(\mathcal{D}_S, f_S)$, we are also given additional training examples $\{(x_i, y_i)\}_{i=1}^{n_T}$ from the target domain $(\mathcal{D}_T, f_T)$.[19] How should we best train a model using the source and target domain examples? Can using the source domain examples provide better generalization than just training on the target domain examples?

In this example, we will study an estimator which can be thought of as a bias-variance tradeoff. In particular, let $\alpha \in [0, 1]$ be a parameter, and consider training on the following mixed loss function:

$$
\hat{h}_\alpha \in \arg\min_{h\in\mathscr{H}} \hat{\ell}_\alpha(h) := \left\{\alpha\hat{\ell}_T(h) + (1-\alpha)\hat{\ell}_S(h)\right\},
$$

where

$$
\hat{\ell}_S(h) = \frac{1}{n_S}\sum_{i=1}^{n_S} \mathbf{1}\{h(x_i) \neq y_i\}, \quad \hat{\ell}_T(h) = \frac{1}{n_T}\sum_{i=1}^{n_T} \mathbf{1}\{h(x_i) \neq y_i\}.
$$

Above, the training data (x_i, y_i) for $\hat{\ell}_S$ corresponds to the source domain data, and for $\hat{\ell}_T$ corresponds to the target domain data. The intuition here is simple: if $\alpha = 1$, then this corresponds to only using the target domain data and training on $\hat{\ell}_T(h)$. On the other hand, if $\alpha = 0$, the this corresponds to ignoring the target domain data and only using the source domain data, training on $\hat{\ell}_S(h)$. Setting $\alpha \in (0, 1)$ mixes both the source and target domain data, which higher values biasing towards the target domain, and lower values biasing towards the source domain. Furthermore, let us denote the mixed population loss:

$$
\ell_\alpha(h) := \alpha\ell_T(h) + (1-\alpha)\ell_S(h).
$$

We first quantify the bias of $\ell_\alpha(h)$ with regards to the target $\ell_T(h)$.

Proposition 3.6. *For any $h \in \mathscr{H}$,*

$$
|\ell_\alpha(h) - \ell_T(h)| \leqslant (1-\alpha)\left(\gamma_\star + \frac{1}{2}d_{\mathscr{H}}(\mathcal{D}_S, \mathcal{D}_T)\right).
$$

Proof. First, we observe that:

$$
\ell_T(h) - \ell_\alpha(h) = (1-\alpha)(\ell_T(h) - \ell_S(h)).
$$

[19] Note here that $y_i = f_S(x_i)$ for the source domain and $y_i = f_T(x_i)$ for the target domain.

By Proposition 3.5, we immediately have

$$\ell_T(h) - \ell_S(h) \leqslant \gamma_\star + \frac{1}{2} d_{\mathscr{H}}(\mathcal{D}_S, \mathcal{D}_T).$$

One can quickly check that the same proof also yields the same bound in the opposite direction, due to the symmetry of the divergence measure $d_{\mathcal{H}}$ and since $\gamma_\star$ is define by minimizing $\ell_S(h) + \ell_T(h)$ (that is, we can swap the role of S, T in the proof). Specifically:

$$|\ell_T(h) - \ell_S(h)| \leqslant \gamma_\star + \frac{1}{2} d_{\mathscr{H}}(\mathcal{D}_S, \mathcal{D}_T),$$

from which the claim follows. □

Next, we prove a concentration inequality for $|\hat{\ell}_\alpha(h) - \ell_\alpha(h)|$.

Proposition 3.7. *Fix an $h \in \mathscr{H}$. For all $t > 0$,*

$$\mathbb{P}\left\{|\hat{\ell}_\alpha(h) - \ell_\alpha(h)| \geqslant t\right\} \leqslant 2\exp\left\{-\frac{t^2}{2(\alpha^2/n_T + (1-\alpha)^2/n_S)}\right\}.$$

Proof. We first compute the MGF of $\hat{\ell}_\alpha(h) - \ell_\alpha(h)$. Let $X_i := \mathbf{1}\{h(x_i) \neq y_i\} - \ell_T(h)$ for $i \in \{1, \ldots, n_T\}$ in the target domain, and $Z_i := \mathbf{1}\{h(x_i) \neq y_i\} - \ell_S(h)$ for $i \in \{1, \ldots, n_S\}$ in the source domain. Observe that both X_i and Z_i are 1-sub-Gaussian by Hoeffding's lemma (Proposition B.7), and therefore $\sum_{i=1}^{n_T} X_i$ is $\sqrt{n_T}$-sub-Gaussian (similarly, $\sum_{i=1}^{n_S} Z_i$ is $\sqrt{n_S}$-sub-Gaussian). For any $\lambda \in \mathbb{R}$,

$$\begin{aligned}
&\mathbb{E}\exp\left(\lambda(\hat{\ell}_\alpha(h) - \ell_\alpha(h))\right) \\
&= \mathbb{E}\exp\left(\lambda\alpha(\hat{\ell}_T(h) - \ell_T(h)) + \lambda(1-\alpha)(\hat{\ell}_S(h) - \ell_S(h))\right) \\
&= \mathbb{E}\exp\left(\lambda\alpha(\hat{\ell}_T(h) - \ell_T(h))\right)\mathbb{E}\exp\left(\lambda(1-\alpha)(\hat{\ell}_S(h) - \ell_S(h))\right) \\
&= \mathbb{E}\exp\left(\frac{\lambda\alpha}{n_T}\sum_{i=1}^{n_T} X_i\right)\mathbb{E}\exp\left(\frac{\lambda(1-\alpha)}{n_S}\sum_{i=1}^{n_S} Y_i\right) \\
&\leqslant \exp\left(\frac{\lambda^2}{2}\left[\frac{\alpha^2}{n_T} + \frac{(1-\alpha)^2}{n_S}\right]\right).
\end{aligned}$$

Hence by the Laplace transform method (Proposition B.2):

$$\begin{aligned}
\mathbb{P}\left\{\hat{\ell}_\alpha(h) - \ell_\alpha(h) \geqslant t\right\} &\leqslant \inf_{\lambda > 0} e^{-\lambda t}\mathbb{E}\exp\left(\lambda(\hat{\ell}_\alpha(h) - \ell_\alpha(h))\right) \\
&\leqslant \inf_{\lambda > 0}\exp\left(-\lambda t + \frac{\lambda^2}{2}\left[\frac{\alpha^2}{n_T} + \frac{(1-\alpha)^2}{n_S}\right]\right) \\
&= \exp\left(-\frac{t^2}{2(\alpha^2/n_T + (1-\alpha)^2/n_S)}\right).
\end{aligned}$$

Note that bounding the reverse direction $\mathbb{P}\left\{\ell_\alpha(h) - \hat{\ell}_\alpha(h) \geqslant t\right\}$ uses an identical argument, from which the claim follows by a union bound. □

We now give a bound on the risk of $\hat{h}_\alpha$ under the target domain.

Proposition 3.8. *Let $h_T^\star \in \arg\min_{h \in \mathscr{H}} \ell_T(h)$. With probability at least $1 - \delta$,*

$$\ell_T(\hat{h}_\alpha) \leqslant \ell_T(h_T^\star) + 2(1-\alpha)\left(\gamma_\star + \frac{1}{2} d_{\mathscr{H}}(\mathcal{D}_S, \mathcal{D}_T)\right) + \sqrt{2\left[\frac{\alpha^2}{n_T} + \frac{(1-\alpha)^2}{n_S}\right]\log\left(\frac{2|\mathscr{H}|}{\delta}\right)}.$$

Proof. First, by Proposition 3.7 followed by a union bound over $h \in \mathscr{H}$, with probability at least $1-\delta$:

$$\forall h \in \mathscr{H},\ |\hat{\ell}_\alpha(h) - \ell_\alpha(h)| \leqslant \sqrt{2\left[\frac{\alpha^2}{n_T} + \frac{(1-\alpha)^2}{n_S}\right]\log\left(\frac{2|\mathscr{H}|}{\delta}\right)} =: C(n_T, n_S, \alpha). \tag{3.5}$$

Let us assume this event holds for the remainder of the proof. Then, abbreviating $d_{\mathscr{H}} = d_{\mathscr{H}}(\mathcal{D}_S, \mathcal{D}_T)$,

$$\begin{aligned}
\ell_T(\hat{h}_\alpha) &\leqslant \ell_\alpha(\hat{h}_\alpha) + (1-\alpha)(\gamma_\star + d_{\mathscr{H}}/2) && \text{(Proposition 3.6)}\\
&\leqslant \hat{\ell}_\alpha(\hat{h}_\alpha) + (1-\alpha)(\gamma_\star + d_{\mathscr{H}}/2) + C(n_T, n_S, \alpha) && \text{(Equation (3.5))}\\
&\leqslant \hat{\ell}_\alpha(h_T^\star) + (1-\alpha)(\gamma_\star + d_{\mathscr{H}}/2) + C(n_T, n_S, \alpha) && (\hat{h}_\alpha \text{ is an ERM})\\
&\leqslant \ell_\alpha(h_T^\star) + (1-\alpha)(\gamma_\star + d_{\mathscr{H}}/2) + 2C(n_T, n_S, \alpha) && \text{(Equation (3.5))}\\
&\leqslant \ell_T(h_T^\star) + 2(1-\alpha)(\gamma_\star + d_{\mathscr{H}}/2) + 2C(n_T, n_S, \alpha). && \text{(Proposition 3.6)}
\end{aligned}$$

□

Is $\alpha < 1$ ever optimal in Proposition 3.8? Given Proposition 3.8, a natural question to ask is if setting $\alpha < 1$, which amounts to a non-zero upweighting of the source domain training error for training $\hat{h}_\alpha$, ever gives a better generalization bound than just training only on the target domain data (i.e., setting $\alpha = 1$). While the precise answer to this question depends on the specific values, let us give some intuition for when this is the case. The bound in Proposition 3.8 reads:

$$\ell_T(\hat{h}_\alpha) - \ell_T(h_T^\star) \leqslant (1-\alpha)A + \sqrt{\left[\frac{\alpha^2}{n_T} + \frac{(1-\alpha)^2}{n_S}\right]B},$$

for some A, B. Note that setting $\alpha = 1$ yields the bound:

$$\ell_T(\hat{h}_{\alpha=1}) - \ell_T(\hat{h}_T^\star) \leqslant \sqrt{\frac{B}{n_T}}.$$

So we are looking for a regime where $\alpha < 1$, and where:

$$(1-\alpha)A + \sqrt{\left[\frac{\alpha^2}{n_T} + \frac{(1-\alpha)^2}{n_S}\right]B} < \sqrt{\frac{B}{n_T}}.$$

Suppose we are in a setting where the source dataset n_S is much larger than n_T. Neglecting the $\frac{(1-\alpha)^2}{n_S}$ term, this inequality is:

$$(1-\alpha)A + \alpha\sqrt{\frac{B}{n_T}} < \sqrt{\frac{B}{n_T}} \iff A < \sqrt{\frac{B}{n_T}} \iff n_T < B/A^2.$$

Thus there becomes a phase transition for n_T at which point it always makes sense to set $\alpha = 1$. Below this threshold, depending on the exact values of A, B, some value of $\alpha < 1$ will yield a lower generalization bound.[20] Ben-David et al. [2010, Figure 1] shows a numerical example plotting the optimal value of α across a range of (n_S, n_T) values, with A, B fixed to some reasonable values.

3.3.3 Likelihood ratio estimation

We now address a special case of domain adaption where we assume the difference between the source domain $(\mathcal{D}_S, f_S)$ and target domain $(\mathcal{D}_T, f_T)$ is restrained to a *covariate shift*: that is $\mathcal{D}_S \neq \mathcal{D}_T$, but $f = f_S = f_T$.

[20] This is not precisely true as we still need to account for the exact value of n_S; see Ben-David et al. [2010, Equation 2] for the precise optimal value of α.

We assume that we get access to a source domain training set $\{(x_i, y_i)\}_{i=1}^{n_S}$, but regarding the target domain, we only assume access to *unlabelled* points $\{x_i\}_{i=1}^{n_T}$.

The key idea in this section is that the target loss $\ell_T(h)$ can be rewritten as an expectation over $\mathcal{D}_S$ instead of $\mathcal{D}_T$, with a likelihood ratio correction term:

$$\begin{aligned}\ell_T(h) &= \mathbb{P}_{x\sim\mathcal{D}_T}\{h(x) \neq f(x)\} = \int \mathbf{1}\{h(x) \neq f(x)\} p_T(x)\,\mathrm{d}x \\ &= \int \mathbf{1}\{h(x) \neq f(x)\} \frac{p_T(x)}{p_S(x)} p_S(x)\,\mathrm{d}x = \mathbb{E}_{x\sim\mathcal{D}_S}\left[\mathbf{1}\{h(x) \neq f(x)\}\frac{p_T(x)}{p_S(x)}\right].\end{aligned}$$

Hence, this suggest that we should optimize the following *weighted* empirical risk:

$$\hat{\ell}_{T,w}(h) := \frac{1}{n_S}\sum_{i=1}^{n_S} \mathbf{1}\{h(x_i) \neq y_i\} w(x_i),$$

where $w(x) = \frac{p_T(x)}{p_S(x)}$. The main question how becomes, how do we estimate $w(x)$ from data? This is where our unlabelled target points $\{x_i\}_{i=1}^{n_T}$ come into play.

Least-squares likelihood ratio estimation. Suppose that $w_\theta(x)$ is a parameterization of a class of non-negative weight functions. Let us design a loss function to minimize for learning the weights θ of w_θ. One natural loss function to choose is:

$$\begin{aligned}D_{\mathrm{LS}}[\theta] &:= \mathbb{E}_{x\sim D_S}\left(w_\theta(x) - \frac{p_T(x)}{p_S(x)}\right)^2 = \int \left(w_\theta(x) - \frac{p_T(x)}{p_S(x)}\right)^2 p_S(x)\,\mathrm{d}x \\ &= \int w_\theta^2(x) p_S(x)\,\mathrm{d}x - 2\int w_\theta(x) p_T(x)\,\mathrm{d}x + \int \frac{p_T^2(x)}{p_S(x)}\,\mathrm{d}x.\end{aligned}$$

Since the last term does not depend on w_θ, we have:

$$\arg\min_\theta D_{\mathrm{LS}}[\theta] = \arg\min_\theta \int w_\theta^2(x) p_S(x)\,\mathrm{d}x - 2\int w_\theta(x) p_T(x)\,\mathrm{d}x.$$

This loss has a nice property that we can form a finite sample version of very naturally using our source and target covariates:

$$\hat{D}_{\mathrm{LS}}[\theta] := \frac{1}{n_S}\sum_{i=1}^{n_S} w_\theta^2(x_i) - \frac{2}{n_T}\sum_{i=1}^{n_T} w_\theta(x_i).$$

One additional regularization constraint which helps in practice is to utilize the following observation:

$$1 = \int p_T(x)\,\mathrm{d}x = \int \frac{p_T(x)}{p_S(x)} p_S(x)\,\mathrm{d}x \approx \frac{1}{n_S}\sum_{i=1}^{n_S}\frac{p_T(x_i)}{p_S(x_i)}.$$

In order for the learned weights $w_\theta(x_i)$ to be more regular (e.g., no extreme variance amongst the weights), the following regularization constraint is helpful:

$$\left|\frac{1}{n_S}\sum_{i=1}^{n_S} w_\theta(x_i) - 1\right| \leqslant \gamma,$$

where γ is a hyperparameter. To summarize, the following optimization problem can be used to search for θ to model the likelihood-ratio:

$$\arg\min_\theta \hat{D}_{\mathrm{LS}}[\theta] \;\text{ subj. to }\; \left|\frac{1}{n_S}\sum_{i=1}^{n_S} w_\theta(x_i) - 1\right| \leqslant \gamma.$$

Suppose we parameterize $w_\theta(x) = \langle \alpha, \phi(x) \rangle$, where $\phi(x) \geqslant 0$ coordinate wise. Then this least-squares optimization becomes a constrained quadratic program (QP):

$$\begin{aligned} &\arg\min_{\alpha \geqslant 0} \left[\alpha^\mathsf{T} \left(\frac{1}{n_S} \sum_{i=1}^{n_S} \phi(x_i) \phi(x_i)^\mathsf{T} \right) \alpha - \left\langle \alpha, \frac{2}{n_T} \sum_{i=1}^{n_T} \phi(x_i) \right\rangle \right] \\ &\text{subj. to} \left| \left\langle \alpha, \frac{1}{n_S} \sum_{i=1}^{n_S} \phi(x_i) \right\rangle - 1 \right| \leqslant \gamma. \end{aligned}$$

The least squares loss $D_{\mathrm{LS}}[\theta]$ is not the only loss we can use.

KL estimation. We now consider a loss based on the KL-divergence. The motivation is that since w_θ models the density ratio $\frac{p_T(x)}{p_S(x)}$, then $p_S w_\theta$ should model p_T. Therefore, this suggests to minimize the loss:

$$D_{\mathrm{KL}}[\theta] := \mathrm{KL}(p_T \parallel p_S w_\theta) = \int \log \frac{p_T(x)}{p_S(x) w_\theta(x)} p_T(x) \, \mathrm{d}x = \mathrm{KL}(p_T \parallel p_S) - \mathbb{E}_{x \sim p_T}[\log w_\theta(x)].$$

Therefore,

$$\arg\min_\theta D_{\mathrm{KL}}[\theta] = \arg\min_\theta \mathbb{E}_{x \sim p_T}[-\log w_\theta(x)].$$

The empirical version of this loss is:

$$\hat{D}_{\mathrm{KL}}[\theta] := -\frac{1}{n_T} \sum_{i=1}^{n_T} \log w_\theta(x_i).$$

Again using the specific parameterization $w_\theta(x) = \langle \alpha, \phi(x) \rangle$, in addition to the regularization constraint, we obtain the following constrained convex optimization problem:

$$\arg\min_{\alpha \geqslant 0} \left[-\frac{1}{n_T} \sum_{i=1}^{n_T} \log \langle \alpha, \phi(x_i) \rangle \right] \text{ subj. to } \left| \left\langle \alpha, \frac{1}{n_S} \sum_{i=1}^{n_S} \phi(x_i) \right\rangle - 1 \right| \leqslant \gamma.$$

The specific KL loss allows one to choose a different parameterization of $w_\theta(x)$ which no longer requires any constraints. In particular, we can parameterize $w_\theta(x)$ as:

$$w_\theta(x) = \frac{\exp(\langle \alpha, \phi(x) \rangle)}{\frac{1}{n_S} \sum_{i=1}^{n_S} \exp(\langle \alpha, \phi(x_i) \rangle)}.$$

This parameterization automatically ensures that:

$$w_\theta(x) \geqslant 0, \quad \frac{1}{n_S} \sum_{i=1}^{n_S} w_\theta(x_i) = 1.$$

So we can simply optimize $\hat{D}_{\mathrm{KL}}$ unconstrained over α:

$$\begin{aligned} &\arg\min_\alpha \left[-\frac{1}{n_T} \sum_{i=1}^{n_T} \left\{ \langle \alpha, \phi(x_i) \rangle - \log \left(\frac{1}{n_S} \sum_{i=1}^{n_S} \exp(\langle \alpha, \phi(x_i) \rangle) \right) \right\} \right] \\ &= \arg\min_\alpha \left[-\left\langle \alpha, \frac{1}{n_T} \sum_{i=1}^{n_T} \langle \alpha, \phi(x_i) \rangle \right\rangle + \log \left(\frac{1}{n_S} \sum_{i=1}^{n_S} \exp(\langle \alpha, \phi(x_i) \rangle) \right) \right]. \end{aligned}$$

This is an unconstrained convex optimization problem over α.

3.3.4 Subspace alignment

We now look at an alternative approach to domain adaptation based on subspace alignment. We will work in the same model as the last section, where we have source domain training examples $\{(x_i, y_i)\}_{i=1}^{n_S}$, and unlabelled target domain examples $\{x_i\}_{i=1}^{n_T}$. The main idea is to learn an alignment map which does a coordinate transform on the source data to align with the target data. After the coordinate transform, one can learn a model on the (transformed) labeled source data, and use this model to make predictions. Let us state the subspace alignment algorithm first, and then we will see where it comes from.

The main subspace alignment algorithm is described in Algorithm 9. It uses PCA (cf. Section 2.1) as a sub-routine.

Algorithm 9 Subspace alignment

1: **Input:** Source domain data $\{(x_i, y_i)\}_{i=1}^{n_S} \subset \mathbb{R}^d \times \mathsf{Y}$, target domain data $\{x_i\}_{i=1}^{n_T} \subset \mathbb{R}^d$. dimension $k \in \{1, \ldots, d\}$,
2: **Output:** Labels $\{y_i\}_{i=1}^{n_T}$ for target domain data.
3: Let $X_S \in \mathbb{R}^{n_S \times d}$ be the data matrix for the source data, and $Y_S \in \mathsf{Y}^{n_S}$ denote the labels for the source data.
4: Let $X_T \in \mathbb{R}^{n_T \times d}$ be the data matrix for the target data.
5: $V_S = \mathrm{PCA}(X_S, k)$.
6: $V_T = \mathrm{PCA}(X_T, k)$.
7: $V_A = V_S V_S^\mathsf{T} V_T$. `/* Coordinate alignment matrix */`
8: $\hat{f} = \mathrm{TrainClassifier}(X_S V_A, Y_S)$.
9: **return** $\{\hat{f}(V_T^\mathsf{T} x_i)\}_{i=1}^{n_T}$.

Let us now see where the alignment matrix $V_S V_S^\mathsf{T} V_T$ arises from in Algorithm 9. Recall that PCA computes a projection matrix V from X, projecting the d-dimensional rows of X onto a k-dimensional subspace. Recall that we compute separate projection matrices V_S, V_T for X_S, X_T, respectively. The resulting dimensionality reduced data matrices are $X_S V_S$ for the source domain, and $X_T V_T$ for the target domain. The missing link is to relate these two coordinate systems to each other, so that they are directly comparable. This is done by the following optimization:

$$\arg \min_{R \in \mathbb{R}^{k \times k}} \|V_S R - V_T\|_F^2.$$

A quick calculation yields that the optimal $R = V_S^\mathsf{T} V_T$, since

$$\nabla_R \|V_S R - V_T\|_F^2 = V_S^\mathsf{T}(V_S R - V_T) = R - V_S^\mathsf{T} V_T.$$

Let us motivate why we use this optimization. Fix an $x \in \mathbb{R}^d$, and suppose we transform the same x into the source coordinate system via $V_S^\mathsf{T} x$ and the target coordinate system $V_T^\mathsf{T} x$. In general, we do not expect $V_S^\mathsf{T} x$ and $V_T x$ to be close in ℓ_2 distance, because the subspaces are not aligned. Now we ask the question, let us find the best affine transform so that $R^\mathsf{T} V_S^\mathsf{T} x$ is as close to $V_T^\mathsf{T} x$ as possible in ℓ_2 distance:

$$\arg \min_{R \in \mathbb{R}^{k \times k}} \|R^\mathsf{T} V_S^\mathsf{T} x - V_T^\mathsf{T} x\|^2.$$

Of course, one issue with this optimization is that x is a free variable here. We have many choices as to what x to pick (e.g., worst case over some distribution or average case). We simply pick the average of $x \sim N(0, I)$, in which case we obtain:

$$\arg \min_{R \in \mathbb{R}^{k \times k}} \mathbb{E}_x \|R^\mathsf{T} V_S^\mathsf{T} x - V_T^\mathsf{T} x\|^2 = \arg \min_{R \in \mathbb{R}^{k \times k}} \|R^\mathsf{T} V_S^\mathsf{T} - V_T\|_F^2 = \arg \min_{R \in \mathbb{R}^{k \times k}} \|V_S R - V_T\|_F^2.$$

3.4 Multi-task learning

We now turn our attention to the problem of multi-task learning. This problem is related to domain adaptation. The problem setup is as follows. Suppose we are working with T domains $(\mathcal{D}_1, f_1), \ldots, (\mathcal{D}_T, f_T)$, and we have access to labelled training data from each domain:

$$S_t := \{(x_{t,i}, y_{t,i})\}_{i=1}^n, \quad t = 1, \ldots, T.$$

Suppose we have prior belief that the T tasks are related in some way:

(a) *Computer vision:* suppose that our input to the model is an image, and we have several image related tasks: (a) classification, (b) object localization (e.g., locating bounding boxes around an object), and (c) depth estimation. In principle, a model which solves task (a) must have something in common with models (b) and (c).

(b) *Natural language processing:* suppose one wants to build a model to translate English into a variety of different languages. There, it also stands to reason that e.g., an English to Spanish translation model may share common feature representations as a model that translates English to Portuguese.

(c) *Robotics:* consider a robotic manipulation application, where one wants to build a policy to control a mobile manipulator to load a dishwasher, and one to unload a dishwasher. While these two tasks are the inverse of each other, they do share common features (as in needing to reason about how to manipulate dishes and bowls, etc.).

Given belief that the T tasks are related, the question is can we exploit this information to more efficiently learn a model for each of the T tasks compared with learning T separate models. This is a form of *transfer learning*, where knowledge in one domain is used to reduce the amount of data collected in separate, but related domain. While there are many algorithms to implement multi-task learning, and more broadly transfer learning, we consider a simple setup where the model for each task is the composition of a shared feature representation, followed by task-specific models. Specifically, let Φ denote a set of functions mapping $\mathsf{X} \mapsto \mathbb{R}^p$, and let $\mathscr{G}$ denote a set of functions mapping $\mathbb{R}^p \mapsto \mathsf{Y}$. We suppose that the labeling function for each task $t \in \{1, \ldots, T\}$ takes on the form:

$$f_t(x) = g_t(\phi(x)), \quad g_t \in \mathscr{G},\ \phi \in \Phi.$$

Note that this model is useful in settings where $|\mathscr{G}| \ll |\Phi|$, so that the complexity of the learning problem is dominated by learning the shared featurizer ϕ.

Linear shared representations. The most basic example of a shared representation is where both ϕ and g_t are linear maps. In particular, for a feature dimension $p \in \mathbb{N}_+$,

$$\phi(x) = Ux,\ U \in \mathbb{R}^{p \times d}, \quad g_t(x) = \langle w_t, x \rangle,\ w_t \in \mathbb{R}^p.$$

Note that there is some redundancy in the above parameterization, since given any (U, w_t), for any invertible matrix $R \in \mathbb{R}^{p \times p}$,

$$\langle w_t, Ux \rangle = \langle RR^{-1} w_t, Ux \rangle = \langle R^{-1} w_t, R^\mathsf{T} Ux \rangle,$$

showing that the pair $(R^\mathsf{T} U, R^{-1} w_t)$ implements the same function f_t. To remove this redundancy, assuming $p \leqslant d$, we can restrict $U^\mathsf{T} \in O(d, p) := \{M \in \mathbb{R}^{d \times p} \mid M^\mathsf{T} M = I\}$ to the set of p-dimensional orthogonal basis in $\mathbb{R}^d$. Thus, we can think of Ux as a p-dimensional projection of x.

Neural network representations. A more general example of a shared representation involves using the first k layers of a neural network as ϕ, and setting the remaining layers as g_t. For instance, if we model $f_t(x)$ with the following model architecture:

$$f_t(x) = (f_H \circ f_{H-1} \circ \cdots \circ f_1)(x),$$

then we can share the first k layers of this model across all the tasks, setting:

$$\phi(x) = (f_k \circ \cdots \circ f_1)(x), \quad g_t(x) = (f_H \circ \cdots \circ f_{k+1})(x).$$

3.4.1 Multi-task ERM

Given our setup, a natural ERM to solve is the following *multi-task* ERM:

$$\{\hat{g}_t\}, \hat{\phi} \in \arg\min_{g_t \in \mathscr{G}, \phi \in \Phi} \frac{1}{nT} \sum_{t=1}^{T} \sum_{i=1}^{n} \ell(g_t(\phi(x_{t,i})), y_{t,i}).$$

Let us now consider the generalization error of this ERM. We will work in the setting where both $\mathscr{G}$ and Φ are finite, and $|\ell| \leqslant 1$. Let us define $r(\{g_t\}, \phi)$ as:

$$r(\{g_t\}, \phi) := \frac{1}{nT} \sum_{t=1}^{T} \sum_{i=1}^{n} \ell(g_t(\phi(x_{t,i})), f_t(x_{t,i})) - \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{x \sim \mathcal{D}_t}[\ell(g_t(\phi(x)), f_t(x))].$$

By Hoeffding's inequality,

$$\mathbb{P}\{|r(\{g_t\}, \phi)| \geqslant \alpha\} \leqslant 2\exp(-nT\alpha^2/2), \quad \alpha > 0.$$

Let $\mathscr{H} := \{(\{g_t\}, \phi) \mid g_t \in \mathscr{G}, \phi \in \Phi\}$. It is straightforward to see that $|\mathscr{H}| = |\mathscr{G}|^T |\Phi|$. Hence by a union bound,

$$\mathbb{P}\left\{\exists\, (\{g_t\}, \phi) \in \mathscr{H} \text{ s.t. } |r(\{g_t\}, \phi)| \geqslant \alpha\right\} \leqslant 2|\mathscr{G}|^T |\Phi| \exp(-nT\alpha^2/2).$$

Setting the RHS equal to $\delta \in (0, 1)$, with probability at least $1 - \delta$,

$$\max_{g_t \in \mathscr{G}, \phi \in \Phi} |r(\{g_t\}, \phi)| \leqslant \sqrt{\frac{2\log|\mathscr{G}|}{n} + \frac{2(\log|\Phi| + \log(2/\delta))}{nT}}.$$

Hence by the usual argument, with probability at least $1 - \delta$,

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{x \sim \mathcal{D}_t}[\ell(\hat{g}_t(\hat{\phi}(x)), f_t(x))] \leqslant \frac{1}{T} \min_{\phi \in \Phi} \sum_{t=1}^{T} \min_{g_t \in \mathscr{G}} \mathbb{E}_{x \sim \mathcal{D}_t}[\ell(g_t(\phi(x)), f_t(x))] + 2\sqrt{\frac{2\log|\mathscr{G}|}{n} + \frac{2(\log|\Phi| + \log(2/\delta))}{nT}}.$$

The RHS of this generalization bound has the following properties:

(a) The total amount of data collected between all the tasks is nT datapoints. This quantity divides the $\log|\Phi|$ complexity of the shared feature class Φ. In order words, all the data pooled amongst the tasks contributes to reducing the generalization error attributed to Φ.

(b) On the other hand, since each individual task only has n datapoints, the $\log|\mathscr{G}|$ complexity of the task specific predictor class is only divided by n and not nT. This is because there is no shared transfer across different tasks regarding the task specific function (as we expect).

Let us compare this bound to one which we separately solve T ERM problems:

$$\hat{g}_t, \hat{\phi} \in \arg \min_{g_t \in \mathscr{G}, \phi \in \Phi} \frac{1}{n} \sum_{i=1}^{n} \ell(g_t(\phi(x_{t,i})), y_{t,i}).$$

Using the standard finite hypothesis class generalization bound, with probability at least $1 - \delta$,

$$\mathbb{E}_{x \sim \mathcal{D}_t}[\ell(g_t(\phi(x)), f_t(x))] \leqslant \min_{g_t \in \mathscr{G}, \phi \in \Phi} \mathbb{E}_{x \sim \mathcal{D}_t}[\ell(g_t(\phi(x)), f_t(x))] + 2\sqrt{\frac{2(\log|\mathscr{G}| + \log|\Phi| + \log(2/\delta))}{n}}.$$

Therefore applying a union bound over all t tasks, with probability at least $1 - \delta$,

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{x \sim \mathcal{D}_t}[\ell(\hat{g}_t(\hat{\phi}(x)), f_t(x))] \leqslant \frac{1}{T}\sum_{t=1}^{T} \min_{g_t \in \mathscr{G}, \phi \in \Phi} \mathbb{E}_{x \sim \mathcal{D}_t}[\ell(g_t(\phi(x)), f_t(x))] + 2\sqrt{\frac{2(\log|\mathscr{G}| + \log|\Phi| + \log(2T/\delta))}{n}}.$$

Unlike the multi-task ERM generalization bound, this generalization bound does not have the property that the $\log|\Phi|$ complexity term of the shared feature class is divided by nT.

3.5 Few-shot learning, model fine-tuning, meta-learning

We now study a modification to the multi-task setup from before. We consider the problem of few-shot learning: given a few labelled examples $\{(x_i, y_i)\}_{i=1}^{K}$ of a new task, can we learn a model to generalize in this task from only K examples? For example, we might want an image classification model dogs to be able to classify a new dog breed that the model has not been trained on in only a few examples, or a character recognition model to quickly learn a new character from only a few examples.

The question then becomes, what sort of prior knowledge can be make into a model so that it can be quickly fine-tuned on a few data points? We will assume that at training time, we have access to a wide variety of different tasks. Our hope is that the diversity of tasks seen at training time will allow for fast test-time generalization. Let us now posit a mathematical model, in addition to an algorithm, for this few-show learning problem.

For this section, we will use the word *task* to denote a joint distribution over $\mathsf{Z} = \mathsf{X} \times \mathsf{Y}$. That is, a task $\mathcal{T} = p(x)p(y \mid x)$ for some $p(x)$ over X, and for some conditional $p(y \mid x)$ (note that this was previously referred to as a domain in the domain adaptation section). Furthermore, we assume there is a distribution $\mathcal{D}$ over *tasks* (that is, a distribution over distributions). We also assume we have a model class $\mathcal{F} = \{f_\theta \mid \theta \in \Theta\}$, where $f_\theta : \mathsf{X} \mapsto \mathsf{Y}$ and a loss function $\ell : \mathsf{Y} \times \mathsf{Y} \mapsto \mathbb{R}$. Note that we will overload:

$$\ell(\theta, (x, y)) = \ell(f_\theta(x), y).$$

Our goal is to learn a *meta-model* f_θ which can be quickly fine-tuned. By fine-tuned, we mean an algorithm $\mathcal{A}$ that, presented with the parameters θ a meta-model f_θ and labelled examples $\{(x_i, y_i)\}_{i=1}^{K}$ from a new task $\mathcal{T}$, quickly computes a new set of parameters $\hat{\theta} = \mathcal{A}(\theta, \mathcal{T})$ so that the task specific risk

$$\mathbb{E}_{(x,y)\sim\mathcal{T}}[\ell(f_{\hat{\theta}}(x), y)]$$

is as minimal as possible. A natural example of a fine-tuning algorithm $\mathcal{A}$ is to run one (or several) steps of gradient descent:

$$\mathcal{A}(\theta, \mathcal{T}) = \theta - \eta \mathbb{E}_{(x,y)\sim\mathcal{T}}[\ell(f_\theta(x), y)].$$

With this fine-tuning algorithm, a natural meta-model loss function is [Finn et al., 2017]:

$$L[\theta] = \mathbb{E}_{\mathcal{T}\sim\mathcal{D}}[\mathbb{E}_{(x,y)\sim\mathcal{T}}[\ell(\mathcal{A}(\theta, \mathcal{T}), (x, y))]].$$

That is, we want to find meta-parameters θ such that, *after* fine-tuning via $\mathcal{A}(\theta, \mathcal{T})$, the resulting model $\mathcal{A}(\theta, \mathcal{T})$ performs well on the task $\mathcal{T}$. See Parnami and Lee [2022, Figure 2] for a figure depicting this problem setup.

3.5.1 Model-agnostic meta-learning (MAML) algorithm

We are now ready to state a practical meta-learning algorithm based on the meta-model loss function $L[\theta]$ from before. Here, we consider a finite sample version of the fine-tuning algorithm $\mathcal{A}$:

$$\mathcal{A}(\theta, \{(x_i, y_i)\}_{i=1}^K) = \theta - \frac{\eta}{K} \sum_{i=1}^{n} \nabla_\theta \ell(f_\theta(x_i), y_i).$$

Algorithm 10 describes a practical meta-learning algorithm, called *Model-agnostic meta-learning (MAML)*. This algorithm is due to Finn et al. [2017].

Algorithm 10 Model-agnostic meta-learning (MAML)

Input: Task distribution $\mathcal{D}$, fine-tuning algorithm $\mathcal{A}$, K samples per fine-tune update, K_1 samples per meta update, B task batch size.
Initialize θ.
while not converged **do**
  Sample $\mathcal{T}_1, \ldots, \mathcal{T}_B$ tasks from $\mathcal{D}$.
  Sample K examples $\{(x_{t,i}, y_{t,i})\}_{i=1}^K \sim \mathcal{T}_i$ for each task, used for the fine-tuning update.
  Sample K_1 examples $\{(x'_{t,i}, y'_{t,i}\}_{i=1}^{K_1} \sim \mathcal{T}_i$ for each task, used for the meta-loss gradient.
  Update θ using gradient

$$g = \sum_{t=1}^{B} \sum_{j=1}^{K_1} \nabla_\theta \ell(\mathcal{A}(\theta, \{(x_{t,i}, y_{t,i})\}_{i=1}^K), (x'_{t,j}, y'_{t,j})).$$

end while
return θ.

Computational considerations of MAML. The main computational overhead of MAML, compared with regular gradient based training, is that the gradient of the meta-learning loss involves differentiation through the fine-tuning algorithm $\mathcal{A}$. In particular, if the fine-tuning algorithm $\mathcal{A}$ implements one step of gradient descent, then the meta-learning loss gradient requires taking a second derivative of the model f_θ. If multiple gradient steps are taken in the fine-tuning algorithm, then this adds further computation each extra fine-tuning gradient step adds another derivative to the meta-learning loss gradient.

Regression case study. We now consider the regression example given in Finn et al. [2017, Section 5.1]. Here, the input space X and output space Y are both equal to $\mathbb{R}$. The task $\mathcal{T}$ is parameterized by two scalar parameters $\alpha = (A, \phi)$, such that the response function $f_\alpha(x) = A \sin(x + \phi)$. For every task, $x \sim \text{Unif}([-5.0, 5.0])$. The distribution $\mathcal{D}$ draws the task parameters $\alpha \sim \text{Unif}([0.1, 5.0]) \times \text{Unif}([0, \pi])$. MAML is run with the square loss $\ell(y, \hat{y}) = (y - \hat{y})^2$, and with the fine-tuning update rule to be one gradient descent step on $K = 10$ labelled datapooints, with step size $\eta = 0.01$.

MAML is compared to two baselines: one where a single model is trained on *all* the training data *across* the different tasks, and an oracle baseline where the true parameters α of the tasks are fed into the model as an extra input. The results of this experiment are shown in Finn et al. [2017, Figures 2 and 3].

Classification case study. We now consider a case study where MAML is run in a classification setting. We consider a problem known as *N-way K-shot classification.* Here, the input domain X is arbitrary, and the output domain $\mathsf{Y} = \{1, \ldots, N\}$. The N-way K-shot classification problem works as follows. N random classes are chosen, and for each class, K instances of that class are drawn. All NK datapoints with their associated labels are given to the learner. The learner is then evaluated on how well it can predict novel, unseen examples of each of the N classes.

In Finn et al. [2017, Section 5.2], MAML is evaluated on two few-show classification benchmark datasets: Omniglot (a character dataset consisting of characters from many different alphabets), and MiniImagenet (a preprocessed version of Imagenet). Finn et al. [2017, Table 1] contains the details of this evaluation, compared with several other benchmarks in the few-show learning literature.

References

Pierre Alquier. User-friendly introduction to pac-bayes bounds. *arXiv preprint arXiv:2110.11216*, 2021.

Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.

Martin Arjovsky and Leon Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 214–223. PMLR, 2017.

Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger B. Grosse. If influence functions are the answer, then what is the question? In *Advances in Neural Information Processing Systems*, volume 35, pages 17953–17967, 2022.

Dominique Bakry, Ivan Gentil, and Michel Ledoux. *Analysis and Geometry of Markov Diffusion Operators*. Springer, 2014.

Krishna Balasubramanian, Sinho Chewi, Murat A. Erdogdu, Adil Salim, and Shunshi Zhang. Towards a theory of non-log-concave sampling: First-order stationarity guarantees for langevin monte carlo. In *Proceedings of Thirty Fifth Conference on Learning Theory*, volume 178, pages 2896–2923. PMLR, 2022.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Vaughan. A theory of learning from different domains. *Machine Learning*, 79:151–175, 2010.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Gintare Karolina Dziugaite and Daniel M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence*, 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1126–1135. PMLR, 06–11 Aug 2017.

Moritz Hardt and Benjamin Recht. *Patterns, predictions, and actions: Foundations of machine learning*. Princeton University Press, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1885–1894. PMLR, 06–11 Aug 2017.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc Aurelio Ranzato, and Fu-Jie Huang. *A tutorial on energy-based learning*. MIT Press, 2006.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11451–11461, 2022.

Calvin Luo. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022.

Tengyu Ma. Lecture notes for machine learning theory (CS229M/STATS214). `https://raw.githubusercontent.com/tengyuma/cs229m_notes/main/master.pdf`, 2022.

Alexander Q. Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 2021.

Archit Parnami and Minwoo Lee. Learning from few examples: A summary of approaches to few-shot learning. *arXiv preprint arXiv:2203.04291*, 2022.

Alex J Smola and Bernhard Schölkopf. *Learning with kernels*, volume 4. Citeseer, 1998.

Yang Song and Diederik P. Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge University Press, 2018.

Vladimir Vovk. Conditional validity of inductive conformal predictors. In *Proceedings of the Asian Conference on Machine Learning*, volume 25, pages 475–490. PMLR, 04–06 Nov 2012.

Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P. Adams, and Peter Orbanz. Non-vacuous generalization bounds at the imagenet scale: A pac-bayesian compression approach. In *International Conference on Learning Representations*, 2019.

A Basic properties of probability divergences

We first state the definition of the *Kullback–Leibler Divergence*. We state it in a general measure-theoretic form, but we will instantiate the definition for some special cases as follows.

Definition A.1 (KL-Divergence)**.** *Let μ, ν be two probability measures on the same measure space. Suppose that μ is absolutely continuous w.r.t. ν, i.e., $\mu \ll \nu$, and let $\frac{\mathrm{d}\mu}{\mathrm{d}\nu}$ denote the Radon-Nikodym derivative of μ w.r.t. ν. The KL-divergence* $\mathrm{KL}(\mu \parallel \nu)$ *is defined as:*

$$\mathrm{KL}(\mu \parallel \nu) = \mathbb{E}_{\mu}\left[\log \frac{\mathrm{d}\mu}{\mathrm{d}\nu}\right].$$

Proposition A.2. *Suppose that $\mu \ll \nu$. We have that* $\mathrm{KL}(\mu \parallel \nu) \geqslant 0$.

Proof. This proof uses the f-divergence representation of KL-divergence to deal with the case when $\nu \ll \mu$ does not hold. Let $f(t) = t \log t$. One can check that this function is convex on $\mathbb{R}_+$. Furthermore, by a change of measure, and Jensen's inequality

$$\mathrm{KL}(\mu \parallel \nu) = \int \log \frac{\mathrm{d}\mu}{\mathrm{d}\nu}\,\mathrm{d}\mu = \int \frac{\mathrm{d}\mu}{\mathrm{d}\nu} \log \frac{\mathrm{d}\mu}{\mathrm{d}\nu}\,\mathrm{d}\nu = \int f\left(\frac{\mathrm{d}\mu}{\mathrm{d}\nu}\right)\mathrm{d}\nu \geqslant f\left(\int \frac{\mathrm{d}\mu}{\mathrm{d}\nu}\,\mathrm{d}\nu\right) = f(1) = 0.$$

□

Lemma A.3 (Donsker-Varadhan)**.** *Let p, q be two probability measures on the same measure space, and suppose that p is absolutely continuous w.r.t. q. We have that:*

$$\mathrm{KL}(p \parallel q) = \sup_g \left\{ \mathbb{E}_p[g] - \log \mathbb{E}_q[\exp(g)] \right\},$$

where the supremum is taken over all measurable $g : \Theta \to \mathbb{R}$ satisfying $\mathbb{E}_q[\exp(g)] < \infty$.

Equivalently, let q be a probability measure and let g be a measurable function satisfying $\mathbb{E}_q[\exp(g)] < \infty$. We have that:

$$\log \mathbb{E}_q[\exp(g)] = \sup_p \left\{ \mathbb{E}_p[g] - \mathrm{KL}(p \parallel q) \right\},$$

where the supremum is taken over probability measures p which are absolutely continuous w.r.t. q.

Proof. Let us define the Gibbs measure with density:

$$\frac{\mathrm{d}\pi}{\mathrm{d}q}(\theta) = \frac{\exp(g(\theta))}{\mathbb{E}_q[\exp(g)]}.$$

Computing the KL-divergence between p and π,

$$\begin{aligned}
\mathrm{KL}(p \parallel \pi) &= \int \log \left[\frac{\mathrm{d}p}{\mathrm{d}\pi} \right] \mathrm{d}p = \int \log \left[\frac{\mathrm{d}p}{\mathrm{d}q} \frac{\mathrm{d}q}{\mathrm{d}\pi} \right] \mathrm{d}p \\
&= \mathrm{KL}(p \parallel q) + \int \log \left[\frac{\mathbb{E}_q[\exp(g)]}{\exp(g)} \right] \mathrm{d}p \\
&= \mathrm{KL}(p \parallel q) + \log \mathbb{E}_q[\exp(g)] - \mathbb{E}_p[g].
\end{aligned}$$

Since $\mathrm{KL}(p \parallel \pi) \geqslant 0$, this shows:

$$\mathrm{KL}(p \parallel q) \geqslant \mathbb{E}_p[g] - \log \mathbb{E}_q[\exp(g)] \iff \log \mathbb{E}_q[\exp(g)] \geqslant \mathbb{E}_p[g] - \mathrm{KL}(p \parallel q).$$

Furthermore, equality is achieved when $p = \pi$, i.e., when $g = \log \frac{\mathrm{d}p}{\mathrm{d}q}$. □

Our next result is the well-known formula for the KL-divergence between two non-degenerate multivariate Gaussians.

Proposition A.4 (KL-divergence for multivariate Gaussians)**.** *Let $\mu_1, \mu_2 \in \mathbb{R}^d$ and Σ_1, Σ_2 be $d \times d$ positive definite matrices. We have:*

$$\mathrm{KL}(N(\mu_1, \Sigma_1) \parallel N(\mu_2, \Sigma_2)) = \frac{1}{2} \left\{ \mathrm{tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)^\top \Sigma_2^{-1}(\mu_2 - \mu_1) - d + \log \frac{\det \Sigma_2}{\det \Sigma_1} \right\}.$$

Proof. Let $\phi(x; \mu, \Sigma)$ denote the density of $N(\mu, \Sigma)$. We know that:

$$\log \phi(x; \mu, \Sigma) = -\frac{1}{2}\|x - \mu\|^2_{\Sigma^{-1}} - \frac{1}{2}\log((2\pi)^d \det \Sigma).$$

Hence, letting $x \sim N(\mu_1, \Sigma_1)$ and $g \sim N(0, I)$,

$$\begin{aligned}
\mathrm{KL}(N(\mu_1, \Sigma_1) \parallel N(\mu_2, \Sigma_2)) &= \mathbb{E}_x[\log \phi(x; \mu_1, \Sigma_1) - \log \phi(x; \mu_2, \Sigma_2)] \\
&= \frac{1}{2}\mathbb{E}_x \left[-\|x - \mu_1\|^2_{\Sigma_1^{-1}} + \|x - \mu_2\|^2_{\Sigma_2^{-1}} + \log \frac{\det \Sigma_2}{\det \Sigma_1} \right] \\
&= \frac{1}{2}\mathbb{E}_x \left[-d + \|x - \mu_2\|^2_{\Sigma_2^{-1}} + \log \frac{\det \Sigma_2}{\det \Sigma_1} \right]
\end{aligned}$$

$$= \frac{1}{2}\mathbb{E}_g\left[-d + \|\mu_1 + \Sigma_1^{1/2} g - \mu_2\|^2_{\Sigma_2^{-1}} + \log\frac{\det\Sigma_2}{\det\Sigma_1}\right]$$
$$= \frac{1}{2}\left[-d + \|\mu_1 - \mu_2\|^2_{\Sigma_2^{-1}} + \mathrm{tr}(\Sigma_1\Sigma_2^{-1}) + \log\frac{\det\Sigma_2}{\det\Sigma_1}\right].$$

□

Another often used probability divergence is the total-variation distance.

Definition A.5. *Let μ, ν be two probability measures on the same measure space $(\Omega, \mathcal{B})$. The total-variation distance (TV-distance) is defined as:*[21]

$$\|\mu - \nu\|_{\mathrm{tv}} = \sup_{A\in\mathcal{B}} |\mu(A) - \nu(A)|.$$

B Concentration inequalities

We give a brief primer on concentration equalities needed for this course. First some motivation. Suppose that $X_1, \ldots, X_n$ are i.i.d. random variables drawn from some underlying distribution. Put $\mu := \mathbb{E}[X_1]$ and $\bar{X}_n := n^{-1}\sum_{i=1}^n X_i$. By the Central Limit Theorem, we know that:

$$\sqrt{n}(\bar{X}_n - \mu) \rightsquigarrow N(0, \mathrm{Var}(X_1)).$$

Here, $\rightsquigarrow$ denotes convergence in distribution. Hence for large enough n, we expect that with high probability,

$$\left|\frac{1}{n}\sum_{i=1}^n (X_i - \mathbb{E}[X_i])\right| \lessapprox \sqrt{\frac{\mathrm{Var}(X_1)}{n}}. \tag{B.1}$$

Here, the notation $\lessapprox$ is just to remind us that this inequality is an approximation, since the CLT is an asymptotic result that is not exact for any finite n. The purpose of this section is to derive analogous non-asymptotic bounds to Equation (B.1), which hold for any $n \geqslant 1$.

Our starting point is Markov's inequality.

Proposition B.1 (Markov's inequality)**.** *Let X be a non-negative random variable. Then,*

$$\forall t > 0,\ \mathbb{P}\{X \geqslant t\} \leqslant \frac{\mathbb{E}[X]}{t}.$$

Proof. Using the fact that X is non-negative,

$$\mathbb{E}[X] = \mathbb{E}[X\mathbf{1}\{X \geqslant t\}] + \mathbb{E}[X\mathbf{1}\{X < t\}] \geqslant \mathbb{E}[t\mathbf{1}\{X \geqslant t\}] = t\mathbb{P}\{X \geqslant t\}.$$

□

Markov's inequality is often used in conjunction with a monotonically increasing bijection, i.e., $\{X \geqslant t\} = \{\phi(X) \geqslant \phi(t)\}$ for any monotonically increasing bijection ϕ. A very useful transform is $\phi(x) = \exp(\lambda x)$ for any $\lambda > 0$:

Proposition B.2 (Laplace transform)**.** *Let X be a random variable. Then,*

$$\forall t \in \mathbb{R},\ \mathbb{P}\{X \geqslant t\} \leqslant \inf_{\lambda>0} e^{-\lambda t}\mathbb{E}\exp(\lambda X).$$

Proof. For any $\lambda > 0$, $\{X \geqslant t\} = \{\exp(\lambda X) \geqslant \exp(\lambda t)\}$. Now apply Markov's inequality to the RHS event. Since $\lambda > 0$ is arbitrary, take the infimum over all $\lambda > 0$ to conclude. □

[21] For those unfamiliar with measure theory, you can think of the supremum as being taken over all subsets of Ω.

Note that the Laplace transform is often referred to in the literature as the "Chernoff bound".

The quantity $\mathbb{E}\exp(\lambda X)$ is referred to as the *moment generating function* (MGF) of X. It has a very useful property that it "tensorizes" in the following away for independent sums. If $S_n = X_1 + \cdots + X_n$ where $X_i \perp X_j$ for $i \neq j$, then

$$\mathbb{E}\exp(\lambda S_n) = \prod_{i=1}^{n} \mathbb{E}\exp(\lambda X_i).$$

Notice that the RHS is the product of the individual MGFs, a property we will soon utilize.

Our next order of business will be to obtain control of MGFs. Without any assumptions, the MGF does not even need to be finite. So, we will need to impose some conditions on our random variables so that the MGFs do exist. The starting point of study will be the Gaussian distribution. Suppose that $X \sim N(0, \sigma^2)$. Then, one can show that:

$$\forall \lambda \in \mathbb{R},\ \mathbb{E}\exp(\lambda X) = \exp(\lambda^2 \sigma^2 / 2). \tag{B.2}$$

Exercise B.3. Show that (B.2) is true.
Hint: write out $\mathbb{E}\exp(\lambda X)$ in integral form and complete the square.

The Gaussian MGF motivates the following definition of a random variable, which we will utilize quite extensively in this course.

Definition B.4 (sub-Gaussian random variable)**.** *Let X be a random variable with finite expectation. The random variable X is σ-sub-Gaussian if:*

$$\forall \lambda \in \mathbb{R},\ \mathbb{E}\exp(\lambda(X - \mathbb{E}[X])) \leqslant \exp(\lambda^2 \sigma^2 / 2).$$

Clearly, a $N(\mu, \sigma^2)$ Gaussian random variable is σ-sub-Gaussian by definition. Also, it is not hard to see that constant scalings of sub-Gaussian random variables remain sub-Gaussian. That is, if $a \in \mathbb{R}$ is fixed and X is σ-sub-Gaussian, then aX is $|a|\sigma$-sub-Gaussian, since:

$$\mathbb{E}\exp(\lambda(aX - \mathbb{E}[aX])) \leqslant \exp(\lambda^2 a^2 \sigma^2 / 2) = \exp(\lambda^2 (a\sigma)^2 / 2),$$

Next, we will show that bounded random variables are also sub-Gaussian. To do this, we start with a definition of a *Rademacher random variable*, which will play a critical role in this class.

Definition B.5 (Rademacher random variable)**.** *A Rademacher random variable ε satisfies:*

$$\mathbb{P}\{\varepsilon = +1\} = \mathbb{P}\{\varepsilon = -1\} = 1/2.$$

Proposition B.6. *A Rademacher random variable is 1-sub-Gaussian.*

Proof. Let ε be a Rademacher random variable, and let $\lambda \in \mathbb{R}$ be fixed. We have:

$$\begin{aligned}
\mathbb{E}\exp(\lambda\varepsilon) &= \frac{1}{2}(\exp(\lambda) + \exp(-\lambda)) \\
&= \frac{1}{2}\left[\sum_{k=0}^{\infty} \frac{\lambda^k}{k!} + \sum_{k=0}^{\infty} \frac{(-\lambda)^k}{k!}\right] && \text{Taylor series of } \exp(x) \\
&= \sum_{k=0}^{\infty} \frac{\lambda^{2k}}{(2k)!} && \text{odd terms cancel} \\
&\leqslant \sum_{k=0}^{\infty} \frac{\lambda^{2k}}{2^k k!} && \text{since } (2k)! \geqslant 2^k k!
\end{aligned}$$

$$= \sum_{k=0}^{\infty} \frac{(\lambda^2/2)^k}{k!} = \exp(\lambda^2/2).$$

□

We can now show that bounded random variables are sub-Gaussian.

Proposition B.7 (Hoeffding's lemma)**.** *Let X be a random variable satisfying $X \in [-1, 1]$ a.s. Then, X is 1-sub-Gaussian.*

Proof. Note: we will prove this result with a slightly worse constant. Specifically, we will prove that X is 2-sub-Gaussian. However, the proof technique we will used illustrates a lot of the key ideas in studying concentration inequalities. The proof for the sharper constant is somewhat less intuitive.

This is our first example of a symmetrization argument, which will be a technique we return to many times. Let ε be a Rademacher random variable, and let X' be a copy of X (all of X, X', ε are mutually independent). The key observation is that $(X - X')$ has the same distribution as $\varepsilon(X - X')$, due to symmetry. We now let $\lambda \in \mathbb{R}$ be fixed, and proceed as follows:

$$\begin{aligned}
\mathbb{E}\exp(\lambda(X - \mathbb{E}[X])) &= \mathbb{E}\exp(\lambda(X - \mathbb{E}[X'])) && \text{since } X \overset{(d)}{=} X' \\
&\leqslant \mathbb{E}\exp(\lambda(X - X')) && \text{Jensen's inequality} \\
&= \mathbb{E}\exp(\lambda\varepsilon(X - X')) && \text{since } X - X' \overset{(d)}{=} \varepsilon(X - X') \\
&= \mathbb{E}[\mathbb{E}[\exp(\lambda\varepsilon(X - X')) \mid X, X']] && \text{tower property} \\
&= \mathbb{E}\exp(\lambda^2(X - X')^2/2) && \text{since } \varepsilon \text{ is 1-sub-Gaussian} \\
&\leqslant \exp(\lambda^2 2^2/2) && \text{since } X \in [-1, 1] \text{ a.s.}
\end{aligned}$$

□

Exercise B.8. Let X be a random variable satisfying $X \in [-a, a]$ a.s. for some $a > 0$. Show that X is a-sub-Gaussian.

Exercise B.9. Let $X_1, \ldots, X_n$ be independent random variables, and suppose that X_i is σ_i-sub-Gaussian for $i = 1, \ldots, n$. Show that $S_n = \sum_{i=1}^n X_i$ is $\sqrt{\sum_{i=1}^n \sigma_i^2}$-sub-Gaussian.

We are now in a position to prove our first concentration inequality.

Proposition B.10 (sub-Gaussian tail bound)**.** *Let X be a σ-sub-Gaussian random variable. We have that:*

$$\forall t > 0,\ \mathbb{P}\{X - \mathbb{E}[X] \geqslant t\} \leqslant \exp(-t^2/(2\sigma^2)).$$

Consequently,

$$\forall t > 0,\ \mathbb{P}\{|X - \mathbb{E}[X]| \geqslant t\} \leqslant 2\exp(-t^2/(2\sigma^2)).$$

Proof. This is a simple consequence of the Laplace transform in addition to the sub-Gaussian MGF bound. For any $\lambda > 0$,

$$\begin{aligned}
\mathbb{P}\{X - \mathbb{E}[X] \geqslant t\} &\leqslant e^{-\lambda t}\mathbb{E}\exp(\lambda(X - \mathbb{E}[X])) \\
&\leqslant \exp\left\{-\lambda t + \lambda^2\sigma^2/2\right\}.
\end{aligned}$$

Now choosing $\lambda = t/\sigma^2$ to minimize the quadratic form inside the exponential, we obtain $\mathbb{P}\{X - \mathbb{E}[X] \geqslant t\} \leqslant \exp(-t^2/(2\sigma^2))$. Applying the same argument to the random variable $-X$ yields $\mathbb{P}\{X - \mathbb{E}[X] \leqslant -t\} \leqslant \exp(-t^2/(2\sigma^2))$ (why is $-X$ also σ-sub-Gaussian?). Now we conclude by a union bound:

$$\begin{aligned}
\mathbb{P}\{|X - \mathbb{E}[X]| \geqslant t\} &= \mathbb{P}\{\{X - \mathbb{E}[X] \geqslant t\} \cup \{X - \mathbb{E}[X] \leqslant -t\}\} \\
&\leqslant \mathbb{P}\{X - \mathbb{E}[X] \geqslant t\} + \mathbb{P}\{X - \mathbb{E}[X] \leqslant -t\} \leqslant 2\exp(-t^2/(2\sigma^2)).
\end{aligned}$$

□

Exercise B.11. Let X be a zero-mean, 1-sub-Gaussian random variable. Show that there exists a universal constant $c > 0$ such that for all $p \geqslant 1$, $\|X\|_{L_p} := (\mathbb{E}|X|^p)^{1/p} \leqslant c\sqrt{p}$.

Hint: You will need to utilize the following facts (which you can use without proof):

(a) For a non-negative random variable X, $\mathbb{E}[X] = \int_0^\infty \mathbb{P}(X > t)\,\mathrm{d}t$.

(b) For all $s \geqslant 1/2$, $\Gamma(s) \leqslant 3s^s$, where $\Gamma(s) := \int_0^\infty t^{s-1}e^{-t}\,\mathrm{d}t$ is the Gamma function.

(c) The quantity $\sup_{p\geqslant 1} p^{1/p} < \infty$.

A nearly immediate consequence is Hoeffding's inequality.

Proposition B.12 (Hoeffding's inequality)**.** *Let $X_1, \ldots, X_n$ be independent random variables with $X_i \in [-a_i, a_i]$ a.s. Put $S_n = \sum_{i=1}^n X_i$. We have that:*

$$\forall t > 0,\ \mathbb{P}\{S_n - \mathbb{E}[S_n] \geqslant t\} \leqslant \exp\left(-\frac{t^2}{2\sum_{i=1}^n a_i^2}\right).$$

Consequently,

$$\forall t > 0,\ \mathbb{P}\{|S_n - \mathbb{E}[S_n]| \geqslant t\} \leqslant 2\exp\left(-\frac{t^2}{2\sum_{i=1}^n a_i^2}\right).$$

Proof. By Exercise B.8, we have that each X_i is a_i-sub-Gaussian. Furthermore, by Exercise B.9, we have that S_n is $\sqrt{\sum_{i=1}^n a_i^2}$-sub-Gaussian. Hence by the sub-Gaussian tail bound Proposition B.10, we have:

$$\forall t > 0,\ \mathbb{P}\{S_n - \mathbb{E}[S_n] \geqslant t\} \leqslant \exp\left(-\frac{t^2}{2\sum_{i=1}^n a_i^2}\right).$$

To control $\mathbb{P}\{|S_n - \mathbb{E}[S_n]| \geqslant t\}$, use the same argument in Proposition B.10. □

Our next result will be the well known bounded differences inequality. To do this, we will need the concept of a martingale.

Definition B.13. *Let $\{X_t\}_{t\geqslant 0}$ be a stochastic process adapted to the filtration $\{\mathcal{F}_t\}_{t\geqslant 0}$. The process $\{X_t\}$ is a martingale if*

$$\forall t \geqslant 0,\ \mathbb{E}[X_{t+1} \mid \mathcal{F}_t] = X_t.$$

Definition B.14. *Let $\{X_t\}_{t\geqslant 0}$ be a martingale. We say that $\{X_t\}_{t\geqslant 0}$ is a $\{\sigma_t\}_{t\geqslant 0}$-sub-Gaussian martingale if for all $t \geqslant 0$, the random variable $(X_{t+1} - X_t) \mid \mathcal{F}_t$ is σ_t-sub-Gaussian almost surely, i.e.,*

$$\forall t \geqslant 0, \lambda \in \mathbb{R},\ \mathbb{E}[\exp(\lambda(X_{t+1} - X_t)) \mid \mathcal{F}_t] \leqslant \exp(\lambda^2\sigma_t^2/2) \text{ a.s.}$$

Proposition B.15 (sub-Gaussian martingale tail inequality)**.** *Let $\{X_t\}_{t\geqslant 0}$ be a $\{\sigma_t\}_{t\geqslant 0}$-sub-Gaussian martingale. We have that:*

$$\forall t > 0,\ \mathbb{P}\{X_n - X_0 \geqslant t\} \leqslant \exp\left(-t^2 / \left(2\sum_{i=0}^{n-1}\sigma_t^2\right)\right).$$

Consequently,

$$\forall t > 0,\ \mathbb{P}\{|X_n - X_0| \geqslant t\} \leqslant 2\exp\left(-t^2 / \left(2\sum_{i=0}^{n-1}\sigma_t^2\right)\right).$$

Proof. By the Laplace transform method (cf. Proposition B.2) and the tower property, for any $\lambda > 0$:

$$\begin{aligned}
\mathbb{P}\{X_n - X_0 \geqslant t\} &\leqslant e^{\lambda t}\mathbb{E}\exp(\lambda(X_n - X_0)) \\
&= e^{-\lambda t}\mathbb{E}\exp\left(\sum_{i=0}^{n-1}\lambda(X_{i+1} - X_i)\right) \\
&= e^{-\lambda t}\mathbb{E}\left[\exp\left(\sum_{i=0}^{n-2}\lambda(X_{i+1} - X_i)\right)\mathbb{E}[\exp(\lambda(X_n - X_{n-1})) \mid \mathcal{F}_{n-1}]\right] \\
&\leqslant \exp\{-\lambda t + \lambda^2\sigma_{n-1}^2/2\}\mathbb{E}\exp\left(\sum_{i=0}^{n-2}\lambda(X_{i+1} - X_i)\right) \\
&\vdots \\
&\leqslant \exp\left\{-\lambda t + \lambda^2\left(\sum_{i=0}^{n-1}\sigma_i^2\right)/2\right\}.
\end{aligned}$$

Now choose $\lambda = t/\sum_{i=0}^{n-1}\sigma_i^2$, from which the claim follows. □

B.1 Bounded differences inequality

Proposition B.16 (Bounded differences inequality)**.** *Let $f : \mathsf{X}^n \mapsto \mathbb{R}$ satisfy the following bounded differences property:*

$$\forall i \in \{1, \dots, n\}, \quad \sup_{x_1,\dots,x_n,x' \in \mathsf{X}} |f(x_{1:i-1}, x_i, x_{i+1:n}) - f(x_{1:i-1}, x_i', x_{i+1:n})| \leqslant c_i.$$

Let $x_1, \dots, x_n \in \mathsf{X}$ be independent random variables. Then,

$$\forall t > 0,\ \mathbb{P}\{f(x_1, \dots, x_n) - \mathbb{E}[f(x_1, \dots, x_n)] \geqslant t\} \leqslant \exp\left(-t^2/\left(2\sum_{i=1}^{n} c_i^2\right)\right).$$

Consequently,

$$\forall t > 0,\ \mathbb{P}\{|f(x_1, \dots, x_n) - \mathbb{E}[f(x_1, \dots, x_n)]| \geqslant t\} \leqslant 2\exp\left(-t^2/\left(2\sum_{i=1}^{n} c_i^2\right)\right).$$

Proof. The idea here is to construct a martingale, known as the Doob martingale. Let $\mathcal{F}_i = \sigma(x_{1:i})$ denote the minimal σ-algebra for $x_{1:i}$,[22] and set $Z_i = \mathbb{E}[f(x_{1:n}) \mid \mathcal{F}_i]$. Let us first check this is a martingale. We have by the tower property of conditional expectations:

$$\mathbb{E}[Z_{i+1} \mid \mathcal{F}_i] = \mathbb{E}[\mathbb{E}[f(x_{1:n}) \mid \mathcal{F}_{i+1}] \mid \mathcal{F}_i] = \mathbb{E}[f(x_{1:n}) \mid \mathcal{F}_i] = Z_i.$$

Second, observe that:

$$Z_n = f(x_{1:n}),\ Z_0 = \mathbb{E}[f(x_{1:n})] \Longrightarrow Z_n - Z_0 = f(x_{1:n}) - \mathbb{E}[f(x_{1:n})].$$

Now let us bound, conditioned on $\mathcal{F}_{i-1}$,

$$|Z_i - Z_{i-1}| = \left|\int f(x_{1:n})\,\mathrm{d}x_{i+1:n} - \int f(x_{1:n})\,\mathrm{d}x_{i:n}\right|$$

[22] If you are unfamiliar with this terminology, just think of $\mathcal{F}_i$ as encoding the information available in the first i datapoints.

$$
\begin{aligned}
&= \left| \int f(x_{1:i-1}, x_i, x_{i+1:n}) \, \mathrm{d}x_{i+1:n} - \int f(x_{i:i-1}, x_i', x_{i+1:n}) \, \mathrm{d}x_i' \mathrm{d}x_{i+1:n} \right| \\
&= \left| \int [f(x_{1:i-1}, x_i, x_{i+1:n}) - f(x_{i:i-1}, x_i', x_{i+1:n})] \, \mathrm{d}x_i' \mathrm{d}x_{i+1:n} \right| \\
&\leqslant \int |f(x_{1:i-1}, x_i, x_{i+1:n}) - f(x_{i:i-1}, x_i', x_{i+1:n})| \, \mathrm{d}x_i' \mathrm{d}x_{i+1:n} && \text{Jensen's inequality} \\
&\leqslant c_i.
\end{aligned}
$$

(Check your understanding: where did we use independence of the x_i's above?) Hence, we have that for all $i \in \mathbb{N}_+$, that $(Z_i - Z_{i-1}) \mid \mathcal{F}_{i-1}$ is a c_i-sub-Gaussian random variable almost surely (cf. Exercise B.8). Therefore, the Doob martingale $\{Z_i\}$ is a $\{c_i\}$-sub-Gaussian martingale. By Proposition B.15, we conclude that for all $t > 0$,

$$
\mathbb{P}\{f(x_{1:n}) - \mathbb{E}[f(x_{1:n})] \geqslant t\} = \mathbb{P}\{Z_n - Z_0 \geqslant t\} \leqslant \exp\left(-t^2 / \left(2 \sum_{i=1}^n c_i^2 \right) \right).
$$

□

B.2 Maximal inequalities

Proposition B.17 (sub-Gaussian maximal inequality)**.** *Let X_i, $i = 1, \ldots, n$ be zero-mean σ-sub-Gaussian random variables (not necessarily independent). Then,*

$$
\mathbb{E} \max_{i=1,\ldots,n} X_i \leqslant \sigma \sqrt{2 \log n}.
$$

Proof. Fix any $\lambda > 0$. We have:

$$
\begin{aligned}
\mathbb{E} \max_{i=1,\ldots,n} X_i &= \lambda^{-1} \mathbb{E} \max_{i=1,\ldots,n} \lambda X_i && \text{since } \lambda > 0 \\
&= \lambda^{-1} \mathbb{E} \log \exp \left(\max_{i=1,\ldots,n} \lambda X_i \right) \\
&\leqslant \lambda^{-1} \log \mathbb{E} \exp \left(\max_{i=1,\ldots,n} \lambda X_i \right) && \text{Jensen's inequality} \\
&= \lambda^{-1} \log \mathbb{E} \max_{i=1,\ldots,n} \exp(\lambda X_i) && \exp \text{ is monotonically increasing} \\
&\leqslant \lambda^{-1} \log \left(\sum_{i=1}^n \mathbb{E} \exp(\lambda X_i) \right) && \max_i a_i \leqslant \sum_i a_i \text{ if } \forall i,\ a_i \geqslant 0 \\
&\leqslant \lambda^{-1} \log \left(n \exp(\lambda^2 \sigma^2 / 2) \right) && \text{each } X_i \text{ is } \sigma\text{-sub-Gaussian} \\
&= \frac{\log n}{\lambda} + \frac{\sigma^2 \lambda}{2}.
\end{aligned}
$$

Since the inequality holds for any $\lambda > 0$, we can optimize over the bound by choosing $\lambda = \sqrt{2 \log n / \sigma^2}$. □

Note that this result is actually sharp: if $\{X_i\}_{i=1}^\infty$ is a sequence of i.i.d. $N(0,1)$ random variables, then one can show:

$$
\lim_{n \to \infty} \frac{\mathbb{E} \max_{i=1,\ldots,n} X_i}{\sqrt{2 \log n}} = 1.
$$

Exercise B.18. Let X_i, $i = 1, \ldots, n$ be zero-mean σ-sub-Gaussian random variables. Show that

$$
\mathbb{E} \max_{i=1,\ldots,n} |X_i| \leqslant \sigma \sqrt{2 \log(2n)}.
$$

C Convex functions

We give a very brief primer on basic properties of convex functions.

Definition C.1. *A set D is convex if for all $x, y \in D$, $\lambda x + (1-\lambda)y \in D$ for all $\lambda \in [0,1]$.*

Definition C.2. *A function $f : D \mapsto \mathbb{R}$ is convex if the domain D is convex and the functions satisfies for all $x, y \in D$ and $\theta \in [0,1]$,*

$$f(\theta x + (1-\theta)y) \leqslant \theta f(x) + (1-\theta) f(y)$$

Proposition C.3. *Let D be a convex domain and $f : D \mapsto \mathbb{R}$:*

(a) Suppose that $f \in C^1(D)$. Then, f is convex iff for every $x, y \in D$,

$$f(y) \geqslant f(x) + \langle \nabla f(x), y - x \rangle.$$

(b) Suppose that $f \in C^2(D)$. Then, f is convex iff for every $x \in D$,

$$\nabla^2 f(x) \succcurlyeq 0.$$

Definition C.4. *A differentiable function $f : D \mapsto \mathbb{R}$ has L-Lipschitz gradients (alternatively, L-smooth), if ∇f is L-Lipschitz,*

$$\|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\| \ \forall x, y \in D.$$

Proposition C.5. *Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be L-smooth. Then for all $x \in \mathbb{R}^n$,*

$$\inf_{x \in \mathbb{R}^n} f(x) := f_\star \leqslant f(x) - \frac{1}{2L}\|\nabla f(x)\|^2.$$

Proof. Taking a second order Taylor series expansion, we have that for all x, y:

$$f(y) \leqslant f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|^2.$$

Choosing $y = x - \frac{1}{L}\nabla f(x)$ (which minimizes the RHS over $y \in \mathbb{R}^n$), we obtain:

$$f(y) \leqslant f(x) - \frac{1}{2L}\|f(x)\|^2.$$

By definition, $f_\star \leqslant f(y)$, which yields the claim. □

Proposition C.6 (Co-coercive gradients)**.** *Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be convex and L-smooth. Then,*

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geqslant \frac{1}{L}\|\nabla f(x) - \nabla f(y)\|^2 \ \forall x, y \in \mathbb{R}^n.$$

Proof. Fix an $x \in \mathbb{R}^n$, and define the function:

$$f_x(z) = f(z) - \langle \nabla f(x), z \rangle.$$

Since f is convex, so is $f_x(z)$ (why?). Hence, we can study its stationary points $\nabla f_x(z) = 0$ to understand its minimizers. Taking $\nabla f_x(z) = \nabla f(z) - \nabla f(x)$, we find that $z = x$ minimizes $f_x(z)$. Furthermore, since f is L-smooth, so is $f_x(z)$ (why?). Therefore, applying Proposition C.5, we conclude that for any $y \in \mathbb{R}^n$,

$$f(y) - f(x) - \langle \nabla f(x), y - x \rangle = f_x(y) - f_x(x) \geqslant \frac{1}{2L}\|\nabla f_x(y)\|^2 = \frac{1}{2L}\|\nabla f(y) - \nabla f(x)\|^2.$$

Next, we can flip the role of x and y in the above argument and conclude

$$f(x) - f(y) - \langle \nabla f(y), x - y \rangle \geqslant \frac{1}{2L}\|\nabla f(y) - \nabla f(x)\|^2.$$

Combining these two inequalities yields the claim. □

Definition C.7 (Strong convexity)**.** *A function $f : D \mapsto \mathbb{R}$ is* μ-strongly-convex *if the domain D is convex and the following holds for all $\theta \in [0,1]$:*

$$f(\theta x + (1-\theta)y) \leqslant \theta f(x) + (1-\theta) f(y) - \frac{\theta(1-\theta)\mu}{2}\|x - y\|^2 \ \forall x, y \in D.$$

D Miscellaneous results

Proposition D.1. *Let $n, d \in \mathbb{N}_+$ such that $n \geqslant d$. Then,*

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

Proof. Since $d/n \leqslant 1$, this implies that $(d/n)^a \leqslant (d/n)^b$ for any $a, b \in \mathbb{N}$ with $a \geqslant b$. Therefore,

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

Now recall that the binomial theorem states for every $x \in \mathbb{R}$,

$$(1+x)^n = \sum_{i=0}^{n} \binom{n}{i} x^i$$

Thus, applying the binomial theorem with $x = d/n$, we have:

$$\sum_{i=0}^{n} \left(\frac{d}{n}\right)^i \binom{n}{i} = \left(1 + \frac{d}{n}\right)^n \leqslant e^d.$$

The claim now follows. □

Proposition D.2. *Let b, c be positive reals satisfying $bc \geqslant 1$. For every $n > 0$, we have that*

$$n \geqslant 2b \log(2bc) \implies n \geqslant b \log(cn).$$

Proof. For $x > 0$, define $f(x) := x - b\log(cx)$. A quick computation shows that $f'(x) = 1 - b/x$, and hence for $x \geqslant b$ we have that $f'(x) \geqslant 0$, or equivalently that $f(x)$ is increasing. Thus, if we can find an n_0 satisfying (a) $n_0 \geqslant b$ and (b) $f(n_0) \geqslant 0$, then $f(n) \geqslant 0$ for all $n \geqslant n_0$. We select $n_0 = 2b\log(2bc)$. We first check that (a) holds:

$$n_0 = 2b\log(2bc) = b \cdot (2\log(2bc)) \geqslant b \cdot (2\log 2) \geqslant b.$$

We now check that (b) holds:

$$\begin{aligned}
b\log(cn_0) &= b\log(2bc\log(2bc)) \\
&= b\log(2\log(2)bc + 2bc\log(bc)) \\
&\leqslant b\log(2\log(2)bc + 2(bc)^2) && \text{since } \log x \leqslant x \ \forall x > 0 \\
&\leqslant b\log(2(1+\log 2)(bc)^2) && \text{since } bc \geqslant 1 \\
&= 2b\log(\sqrt{2(1+\log 2)}bc) \\
&\leqslant 2b\log(2bc) = n_0.
\end{aligned}$$

Hence, $f(n_0) = n_0 - b\log(cn_0) \geqslant 0$. We now conclude that for any $n \geqslant n_0$, then $f(n) \geqslant 0$. □