Chapter 10

Ensemble learning

Chapter summary

– Combining several predictors learned on modified versions of the original dataset can have computational and/or statistical benefits.
– Averaging predictors on several reshuffled/resampled/uniformly projected data sets will typically lower the estimator's variance with a potentially limited increase in bias.
– Boosting: iteratively refining the prediction function by re-training on a reweighted dataset in a greedy fashion is an efficient way of building task-dependent features.

Given a supervised learning algorithm $\mathcal{A}$ that goes from datasets $\mathcal{D}$ to prediction rules $\mathcal{A}(\mathcal{D}) : \mathcal{X} \to \mathcal{Y}$, can we run it several times on different datasets constructed from the same original one, and combine the results to get a better overall predictor? The combination is typically a "linear" combination: like for local averaging methods, which combine labels from close-by inputs, we combine the predicted labels from the estimators learned on different datasets. For regression ($\mathcal{Y} = \mathbb{R}$), this is done by simply linearly combining predictions; for classification, this is done by a weighted majority vote or by linearly combining real-valued predictions when convex surrogates are used (such as the logistic loss). For linear models (in their parameters), such linear combinations do not lead to new functions that could not be accessed initially. Still, for non-linear models, this leads to new functions with typically better approximation properties.

The construction of a new dataset given an old one $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, is typically done by giving a different weight $v_i \in \mathbb{R}_+$ to each (x_i, y_i). When the weights are integer-valued, this can be implemented by duplicating the corresponding observations several times (as many times as the integer weight) and then using an existing algorithm for (regularized) empirical risk minimization on the enlarged dataset. In particular, for stochastic gradient descent on the empirical risk, this can be implemented by sampling each observation (x_i, y_i) according to its weight v_i (which then does not need to be an integer). Note, however, that most learning techniques, particularly those based on empirical risk minimization, can directly accommodate arbitrary weights.

In this chapter, we consider two classes of techniques:

- *Bagging / averaging techniques*: datasets are constructed *in parallel*, and the weights are typically random and "uniform" (for example, distributed uniformly or constant). A similar effect can be obtained by modifying the original dataset using random projections. This is studied in Section 10.1 and in Section 10.2.
- *Boosting techniques*: datasets are constructed *sequentially*, and these weights are adapted from previous datasets and thus not uniformly distributed. This is studied in Section 10.3.

The benefits of each combination technique will depend strongly on the original predictor, with three classes that we have considered in earlier chapters:

- Local averaging methods: they will be well-adapted to all ensemble learning techniques, in particular for predictors with high-variance, such as 1-nearest-neighbor estimation.
- Empirical risk minimization with non-linear models: from a set of functions $\varphi(w, \cdot)$, with $w \in \mathcal{W}$, then linear combinations increase the set of models to $\int_{\mathcal{W}} \varphi(w, x) d\nu(w)$, for ν a signed measure on $\mathcal{W}$. These will be adapted to boosting techniques (we already saw some of them in Chapter 9 in the context of neural networks).
- Empirical risk minimization with linear models (linearity in the model's parameters): the overall model class remains the same by taking linear combinations. Thus, these are typically not adapted to ensemble learning techniques unless some variable/feature selection is added (as we do in Section 10.2).

10.1 Averaging / bagging

In this section, for simplicity, we consider the regression case with the square loss where we have an explicit bias/variance decomposition, noting that most results extend beyond that situation. See Exercise 10.1 below.

10.1.1 Independent datasets

The idea of bagging, and more generally of averaging methods, is to average predictions from estimators learned from datasets that are as independent as possible. In an idealized situation, we have m independent datasets of size n, composed of i.i.d. observations from the same distribution $p(x, y)$ on $\mathcal{X} \times \mathcal{Y}$. We obtain for each of them an estimator $\hat{f}_\lambda^{(j)}$, where $j \in \{1, \dots, m\}$, and λ is an associated hyperparameter specific to the learning procedure. The new predictor is $\hat{f}_\lambda^{\text{bag}}$ is simply the average of all $\hat{f}_\lambda^{(j)}$, $j = 1, \dots, m$.

If we denote $\text{bias}^{(j)}(x) = \mathbb{E}[\hat{f}_\lambda^{(j)}(x)] - f^*(x)$, and $\text{var}^{(j)}(x) = \text{var}\left[\hat{f}_\lambda^{(j)}(x)\right]$ (assuming x

is fixed and only taking expectations with respect to the data), then they are the same for all $j \in \{1, \dots, m\}$, and the bias of $\hat{f}_\lambda^{\text{bag}}$ is the same as the base bias for a single dataset (and thus so is the squared bias). At the same time, the variance is divided by m because the datasets are assumed to be independent.

Thus, in the bias/variance trade-off, the selected hyperparameter will typically select a higher variance (or equivalently lower bias) estimator than for $m = 1$. We now give a few examples.

k-nearest neighbor regression. We consider the analysis from Section 6.3.2 on prediction problems over $\mathcal{X} \subset \mathbb{R}^d$, where we showed in Prop. 6.2 that the (squared) bias was upper-bounded by $8B^2 \text{diam}(\mathcal{X})^2 \big(\frac{2k}{n}\big)^{2/d}$ (for $d \geqslant 2$). At the same time, the variance was bounded by $\frac{\sigma^2}{k}$, where σ^2 is a bound on the noise variance on top of the target function f^*, while B is the Lipschitz-constant of the target function. Thus, with m replications, we get an excess risk upper-bounded by

$$\frac{\sigma^2}{km} + 8B^2 \text{diam}(\mathcal{X})^2 \Big(\frac{2k}{n}\Big)^{2/d}.$$

When optimizing the bound above with respect to k, we get that $k^{1+2/d} \propto \frac{n^{2/d}}{m}$, leading to $k \propto \frac{1}{m^{d/(2+d)}} n^{2/(2+d)}$. Compared to Section 6.3.2, we obtain a smaller number of neighbors (which is consistent with favoring higher variance estimators). The overall excess risk ends up being proportional to $1/(mn)^{2/(d+2)}$, which is exactly the rate for a dataset of $N = mn$ observations.

Thus, dividing a dataset of N observations in m chunks of $n = N/m$ observations, estimating independently, and combining linearly does not lead to an overall improved statistical behavior compared to learning all at once. Still, it can have significant computational advantages when the m estimators can be computed in parallel (and totally independently). We thus obtain a distributed algorithm with the same worst-case predictive performance as for a single machine.

Note here that there is an upper bound on the number of replications (and thus the ability for parallelization) to get the same (optimal rate), as we need k to be larger than one, and thus, m cannot grow larger than $n^{2/d}$.

Exercise 10.1 *We consider k-nearest neighbor multi-category classification with a majority vote rule. What is the optimal choice of m when using independent datasets?*

Ridge regression. Following the analysis from Section 7.6.6, the variance of the ridge regression estimator was proportional to $\frac{\sigma^2}{n}\lambda^{-1/\alpha}$ and the bias proportional to $\lambda^{t/s}$ (see precise definitions in Section 7.6.6). With m replications, we thus get an excess risk proportional to $\frac{\sigma^2}{nm}\lambda^{-1/\alpha} + \lambda^{t/s}$, and the averaged estimator behaves like having $N = nm$ observations. Again, with the proper choice of regularization parameter (lower λ than for the full dataset), there is no statistical advantage. Still, there may be a computational one, not only for parallel processing but also with a single machine, as the training time for ridge regression is super-linear in the number of observations (see the exercise below).

Exercise 10.2 *Assuming that obtaining an estimator for ridge regression has running-time complexity $O(n^\beta)$ for $\beta \geqslant 1$ for n observations, what is the complexity of using a split of the data into m chunks? What is the optimal value of m?*

Beyond independent datasets. Having independent datasets may not be possible, and one typically needs to artificially "create" such replicated datasets from a single one, which is precisely what bagging methods will do in the next section, with still a reduced variance, but this time a potentially higher bias.

10.1.2 Bagging

We consider data sets $\mathcal{D}^{(b)}$, obtained with random weights $v_i^{(b)} \in \mathbb{R}_+$, $i = 1, \dots, n$. For the bootstrap, we consider n samples from the original n data points with replacement, which corresponds to $v_i^{(b)} \in \mathbb{N}$, $i = 1, \dots, n$, that sum to n. Such sets of weights are sampled independently m times. We study $m = \infty$ for simplicity, that is, infinitely many replications (in practice, the infinite m behavior can be achieved with moderate m's). Infinitely many bootstrap replications lead to a form of stabilization, which is important for highly variable predictors (which usually imply a large estimation variance).

For linear estimators (in the definition of Section 6.2.1) with the square loss, such as kernel ridge regression or local averaging, this leads to another linear estimator. Therefore, this provides alternative ways of regularizing, which typically may not provide a strong statistical gain over existing methods but provide a computational gain, in particular when each estimator is very efficient to compute. Overall, as shown below for 1-nearest-neighbor, bagging will lower variance while increasing the bias, thus leading to trade-offs that are common in regularizing methods. See also the end of Section 10.2 for a short description of "random forests", which is also (partially) based on bagging.

For simplicity, we will consider averaging estimators obtained by randomly selecting s observations from the n available ones, doing this many times (infinitely many for the analysis), and averaging the predictions.

Exercise 10.3 *Show that when sampling n elements* with *replacement from n items, the expected fraction of distinct items is $1 - (1 - 1/n)^n$, and that it tends to $1 - 1/e$ when n tends to infinity.*

One-nearest neighbor regression. We focus on the 1-nearest neighbor estimator where the strong effect of bagging is striking. The analysis below follows from Biau et al. (2010). The key observation is that if we denote $(x_{(i)}(x), y_{(i)}(x))$ the pair of observations which is the i-th nearest neighbor of x from the dataset $x_1, \dots, x_n$ (ignoring ties), then we can write the bagged estimate as

$$\hat{f}(x) = \sum_{i=1}^{n} V_i y_{(i)}(x),$$

where the non-negative weights V_i sum to one, and *do not depend on x*. The weight V_i is the probability that the i-th nearest neighbor of x is the 1-nearest-neighbor of x in a uniform subsample of size s. We consider sampling without replacement and leave sampling with replacement as an exercise (see Biau et al., 2010, for more details). We assume $s \geqslant 2$.

To select the i-th nearest neighbor as the 1-nearest-neighbor in a subsample, we need that the i-th nearest neighbor is selected but none of the closer neighbors, which leaves $s-1$ elements to choose among $n-i$ possibilities. This shows, that if $i > n-s+1$, then $V_i = 0$, while otherwise $V_i = \binom{n}{s}^{-1}\binom{n-i}{s-1}$, as the total number of subsets of size s is $\binom{n}{s}$ and there are $\binom{n-i}{s-1}$ relevant ones.

We can now use the reasoning from Section 6.3.2. Since for any x, the weights given to each observation (once they are ordered in terms of distance to x) are $V_1, \dots, V_n$, the variance term is equal to $\sum_{i=1}^n V_i^2$. To obtain a bound, we note that for $i \leqslant n-s+1$,

$$V_i = \frac{s}{n-s+1} \frac{\prod_{j=0}^{s-2}(n-i-j)}{\prod_{j=0}^{s-2}(n-j)} = \frac{s}{n-s+1} \prod_{j=0}^{s-2} (1 - \frac{i}{n-j}) \leqslant \frac{s}{n-s+1} \prod_{j=0}^{s-2} (1 - \frac{i}{n}),$$

leading to, upper-bounding the sum by an integral:

$$\begin{aligned}\sum_{i=1}^n V_i^2 &\leqslant \frac{s^2}{(n-s+1)^2} \sum_{i=1}^n \big(1 - \frac{i}{n}\big)^{2(s-1)} \leqslant \frac{ns^2}{(n-s+1)^2} \int_0^1 (1-t)^{2(s-1)} dt \\ &\leqslant \frac{ns^2}{(n-s+1)^2} \frac{1}{2s-1} \leqslant \frac{ns}{(n-s+1)^2} = \frac{s}{n} \frac{1}{(1+1/n-s/n)^2}.\end{aligned}$$

For the bias term, we need to bound $\sum_{i=1}^n V_i \cdot \mathbb{E}\big[\|x - x_{(i)}(x)\|^2\big]$, where the expectation is with respect to the data and the test point x. We note here that by definition of V_i, and conditioning on the data and x, this is the expectation of the distance to the first nearest neighbor from a random sample of size s, and thus, by Lemma 6.1, less than $4\mathrm{diam}(\mathcal{X})^2 \frac{1}{s^{2/d}}$ if $d \geqslant 2$ (which we now assume).

Thus, the overall excess risk is less than

$$4B^2 \mathrm{diam}(\mathcal{X})^2 \frac{1}{s^{2/d}} + \frac{s}{n} \frac{1}{(1+1/n-s/n)^2},$$

which we can balance by choosing $s^{1+2/d} \propto n$, leading to the same performance as k-nearest neighbor for a well chosen k, but now with a bagged esimate.

In Figure 10.1, simulations in one dimension are plotted, showing the regularizing effects of bagging; we see that when $s = n$ (no subsampling), we recover the 1-nearest neighbor estimate, and when s decreases, the variance indeed decreases, while the bias increases.

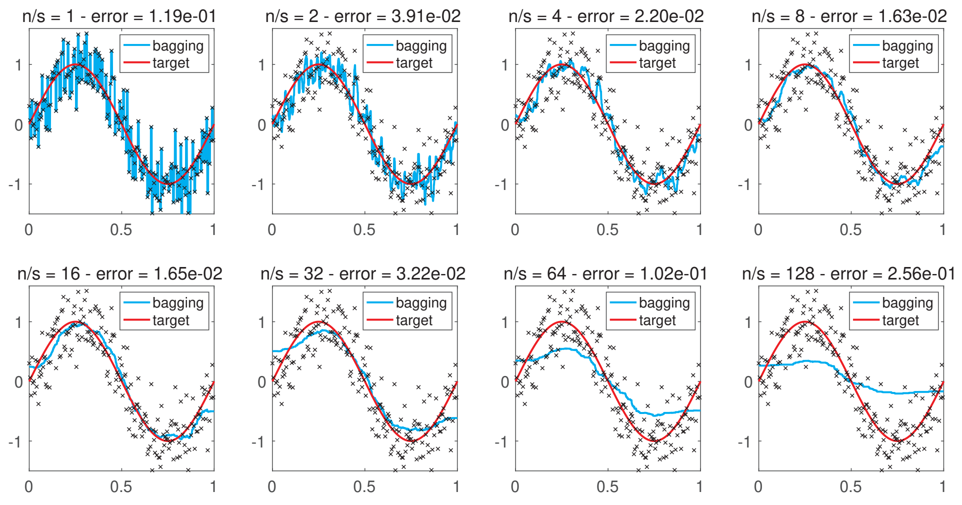

Figure 10.1: Subsampling estimates with $m = 20$ subsampled datasets, for varying subsampling ratios n/s, with an estimation of the testing error. When n/s is equal to one, we recover the 1-nearest neighbor classifier (which overfits), and when n/s grows, we get better fits until underfitting kicks in.

10.2 Random projections and averaging

In the previous section, we reweighted observations to be able to re-run the original algorithm. This can also be done through random projections of all observations. Such random projections can be performed in several ways: (a) for data in $\mathbb{R}^d$ by selecting s of the d variables, (b) still for data in $\mathbb{R}^d$, by projecting the data in a more general s-dimensional subspace, (c) for kernel methods, using random features such as presented in Section 7.4. Such random projections can also reduce the number of samples while keeping the dimension fixed.

In this section, we consider random projections for ordinary least-squares (with the same notation as in Chapter 3, with $y \in \mathbb{R}^n$ the response vector and $\Phi \in \mathbb{R}^{n\times d}$ the design matrix), in two settings:

(a) *Sketching*: replacing $\min_{\theta\in\mathbb{R}^d} \|y-\Phi\theta\|_2^2$ by $\min_{\theta\in\mathbb{R}^d} \|Sy-S\Phi\theta\|_2^2$, where $S \in \mathbb{R}^{s\times n}$ is an i.i.d. Gaussian matrix (with independent zero mean and unit variance elements). This is an idealization of subsampling done in the previous section. Here we typically have $n > s > d$ (more observations than the feature dimension), and one of the benefits of sketching is to be able to store a reduced representation of the data ($\mathbb{R}^{s\times d}$ instead of $\mathbb{R}^{n\times d}$).

(b) *Random projection*: replacing $\min_{\theta\in\mathbb{R}^d} \|y - \Phi\theta\|_2^2$ by $\min_{\eta\in\mathbb{R}^s} \|y - \Phi S\eta\|_2^2$, where $S \in \mathbb{R}^{d\times s}$ is a more general sketching matrix. Here, we typically have $d > n > s$ (high-dimensional situation). The benefits of random projection are two-fold: reduction in computation time and regularization. This corresponds to replacing the corresponding feature vectors $\varphi(x) \in \mathbb{R}^d$ by $S^\top \varphi(x) \in \mathbb{R}^s$. We will consider Gaussian matrices, but also subsampling, and draw connections with kernel methods.

In the following sections, we study these precisely for the ordinary least-squares framework (it could also be done for ridge regression). We first briefly mention a commonly used related approach.

Random forests. A popular algorithm called random forests (Breiman, 2001) mixes both dimension reduction by projection and bagging: decision trees are learned on a bootstrapped sample of the data, with selecting a random subset of features at every splitting decision. This algorithm has nice properties (invariance to rescaling of the variables, robustness in high dimension due to the random feature selection) and can be extended in many ways. See Biau and Scornet (2016) for details.

10.2.1 Gaussian sketching

Following Section 3.3 on ordinary least-squares, we consider a design matrix $\Phi \in \mathbb{R}^{n\times d}$ with rank d (that is, $\Phi^\top\Phi \in \mathbb{R}^{d\times d}$ invertible), which implies $n \geqslant d$. We consider $s > d$ Gaussian random projections, with typically $s \leqslant n$, but this is not necessary in the analysis below.

The estimator $\hat\theta^{(j)}$ is obtained by using $S^{(j)} \in \mathbb{R}^{s\times n}$, with $j = 1, \dots, m$, where m denotes the number of replications. We then consider $\hat\theta = \frac{1}{m}\sum_{j=1}^m \hat\theta^{(j)}$. When $m = 1$, this is a single sketch.

We will consider the same assumptions as in Section 3.5, that is, $y = \Phi\theta_* + \varepsilon$, where $\varepsilon \in \mathbb{R}^n$ has independent zero-mean components with variance σ^2, and $\theta_* \in \mathbb{R}^d$. Our goal is to compute the fixed design error $\frac{1}{n}\mathbb{E}_{\varepsilon,S}\|\Phi\hat\theta - \Phi\theta_*\|_2^2$, where we take both expectations, with respect to the learning problem (in the fixed design setting, the noise vector ε) and the added randomization (the sketching matrices $S^{(j)}$, $j = 1, \dots, m$).

To compute this error, we first need to compute expectations and variances with respect to the random projections, assuming that ε is fixed.

Since the Gaussian matrices $S^{(j)}$ are invariant by left and right multiplication by an orthogonal matrix, we can assume that the singular value decomposition of $\Phi = UDV^\top$, where $V \in \mathbb{R}^{d\times d}$ is orthogonal (i.e., $V^\top V = VV^\top = I$), $D \in \mathbb{R}^{d\times d}$ is an invertible diagonal matrix, and $U \in \mathbb{R}^{n\times d}$ has orthonormal columns (i.e., $U^\top U = I$), is such that $U = \binom{I}{0}$, and that we can write $S^{(j)} = \begin{pmatrix} S_1^{(j)} & S_2^{(j)} \end{pmatrix}$ with $S_1^{(j)} \in \mathbb{R}^{s\times d}$ and $S_2^{(j)} \in \mathbb{R}^{s\times(n-d)}$. We can also split y as $y = \binom{y_1}{y_2}$ for $y_1 \in \mathbb{R}^d$ and $y_2 \in \mathbb{R}^{n-d}$.

We can write down the normal equations that define $\hat\theta^{(j)} \in \mathbb{R}^d$, for each $j \in \{1, \dots, m\}$, that is, $(\Phi^\top (S^{(j)})^\top S^{(j)}\Phi)\hat\theta^{(j)} = \Phi^\top (S^{(j)})^\top S^{(j)} y$, leading to the following closed-form estimators $\hat\theta^{(j)} = (\Phi^\top (S^{(j)})^\top S^{(j)}\Phi)^{-1}\Phi^\top (S^{(j)})^\top S^{(j)} y$.[1] Using the assumptions above regarding the SVD of Φ, we have: $S^{(j)}\Phi = S_1^{(j)} DV^\top$. We can then expand the prediction

[1]If $s \geqslant d$, then $S^{(j)}\Phi$ has almost surely rank d, and thus $\hat\theta^{(j)}$ is uniquely defined.

vector in $\mathbb{R}^n$ as:

$$\begin{aligned}
\Phi\hat\theta^{(j)} &= \Phi(\Phi^\top (S^{(j)})^\top S^{(j)}\Phi)^{-1}\Phi^\top (S^{(j)})^\top S^{(j)} y \\
&= \begin{pmatrix} I \\ 0 \end{pmatrix} DV^\top (VD(S_1^{(j)})^\top S_1^{(j)} DV^\top)^{-1} VD(S_1^{(j)})^\top S^{(j)} y \\
&= \begin{pmatrix} I \\ 0 \end{pmatrix} ((S_1^{(j)})^\top S_1^{(j)})^{-1}(S_1^{(j)})^\top S^{(j)} y = \begin{pmatrix} I \\ 0 \end{pmatrix} ((S_1^{(j)})^\top S_1^{(j)})^{-1}(S_1^{(j)})^\top (S_1^{(j)} y_1 + S_2^{(j)} y_2) \\
&= \begin{pmatrix} y_1 + ((S_1^{(j)})^\top S_1^{(j)})^{-1}(S_1^{(j)})^\top S_2^{(j)} y_2 \\ 0 \end{pmatrix}.
\end{aligned}$$

Thus, since $\mathbb{E}[S_2^{(j)}] = 0$ and $S_2^{(j)}$ is independent of $S_1^{(j)}$, we get $\mathbb{E}_{S^{(j)}}\big[\Phi\hat\theta^{(j)}\big] = \binom{y_1}{0}$, which happens to be exactly the OLS estimator $\Phi\hat\theta_{\text{OLS}} = \Phi(\Phi^\top\Phi)^{-1}\Phi^\top y = \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} y$. Moreover, we have the model $y = \Phi\theta_* + \varepsilon$ and, if we split ε as $\varepsilon = \binom{\varepsilon_1}{\varepsilon_2}$, we have $y = \binom{I}{0} DV^\top \theta_* + \varepsilon$, and thus $y_2 = \varepsilon_2$. We thus get:

$$\mathbb{E}_{S^{(j)}}\Big[\big\|\Phi\hat\theta^{(j)} - \mathbb{E}_{S^{(j)}}\Phi\hat\theta^{(j)}\big\|^2\Big] = \mathbb{E}_{S^{(j)}}\big[\|((S_1^{(j)})^\top S_1^{(j)})^{-1}(S_1^{(j)})^\top S_2^{(j)}\varepsilon_2\|_2^2\big].$$

Taking the expectation with respect to ε, and using expectations for the (inverse) Wishart distribution,[2] this leads to

$$\begin{aligned}
\mathbb{E}_{\varepsilon,S^{(j)}}\Big[\big\|\Phi\hat\theta^{(j)} - \mathbb{E}_{S^{(j)}}\Phi\hat\theta^{(j)}\big\|^2\Big] &= \sigma^2 \mathbb{E}_{S^{(j)}}\big[\operatorname{tr}\big((S_2^{(j)})^\top (S_1^{(j)}((S_1^{(j)})^\top S_1^{(j)})^{-2}(S_1^{(j)})^\top S_2^{(j)}\big)\big] \\
&= (n-d)\sigma^2 \mathbb{E}_{S_1^{(j)}}\big[\operatorname{tr}\big(((S_1^{(j)})^\top S_1^{(j)})^{-1}\big)\big] = \frac{d}{s-d-1}(n-d)\sigma^2.
\end{aligned}$$

We can now compute the overall expected generalization error:

$$\begin{aligned}
\frac{1}{n}\mathbb{E}_{\varepsilon,S^{(j)}}\bigg[\Big\|\frac{1}{m}\sum_{j=1}^m \Phi\hat\theta^{(j)} - \Phi\theta_*\Big\|_2^2\bigg] &= \frac{1}{n}\mathbb{E}_\varepsilon\bigg[\Big\|\frac{1}{m}\sum_{j=1}^m \mathbb{E}_{S^{(j)}}\big[\Phi\hat\theta^{(j)}\big] - \Phi\theta_*\Big\|_2^2\bigg] \\
&\quad + \frac{1}{nm}\mathbb{E}_{\varepsilon,S^{(1)}}\Big[\big\|\Phi\hat\theta^{(1)} - \mathbb{E}_{S^{(1)}}\Phi\hat\theta^{(1)}\big\|^2\Big] \\
&= \frac{1}{n}\mathbb{E}_\varepsilon\Big[\big\|\Phi\hat\theta_{\text{OLS}} - \Phi\theta_*\big\|_2^2\Big] + \sigma^2\frac{d}{nm}\frac{n-d}{s-d-1} \\
&= \sigma^2\frac{d}{n} + \sigma^2\frac{d}{nm}\frac{n-d}{s-d-1}.
\end{aligned}$$

Thus, when m or s tends to infinity, we recover the traditional OLS behavior, while for m and s finite, the performance degrades gracefully. Moreover, when $s = n$, even for $m = 1$, we get essentially twice the performance of the OLS estimator. We note that to get the same performance as OLS (up to a factor of 2), we need $m = \frac{n-d}{s-d-1} \sim \frac{n}{s}$ replications.

[2] If $S \in \mathbb{R}^{a\times b}$ has independent standard Gaussian components, then $\mathbb{E}[(S^\top S)^{-1}] = \frac{1}{a-b-1} I$ if $a > b+1$, and $\mathbb{E}[SS^\top] = bI$; see https://en.wikipedia.org/wiki/Inverse-Wishart_distribution.

As in the previous section, there is no statistical gain (here, compared to OLS), but only potentially a computational one (because some computations may be done in parallel and of reduced storage). See, e.g., Dobriban and Liu (2019) for other criteria and sketching matrices.

Beyond Gaussian sketching. In this section, we have chosen a Gaussian sketching matrix S. This made the analysis simple because of the properties of the Gaussian distribution (invariance by rotation and availability of exact expectations for inverse Wishart distributions). The analysis can be extended with more complex tools to other random sketching matrices with more attractive computational properties, such as with many zeros, leading to subsampling observations or dimensions. See Wang et al. (2018); Dobriban and Liu (2019) and references therein. For random projections below, our analysis will apply to more general sketches.

10.2.2 Random projections

We also consider the fixed design set-up, with a design matrix $\Phi \in \mathbb{R}^{n\times d}$ and a response vector of the form $y = \Phi\theta_* + \varepsilon$. We now assume that $d > n$ (high-dimensional set-up) and that the rank of Φ is n. For each $j \in \{1, \dots, n\}$, we consider a sketching matrix $S^{(j)} \in \mathbb{R}^{d\times s}$, for $s \leqslant n$ sampled independently from a distribution to be determined (we only assume that almost surely, its rank is equal to s). We then consider $\hat\eta^{(j)}$ as a minimizer of $\min_{\eta\in\mathbb{R}^s} \|y - \Phi S^{(j)}\eta\|_2^2$. For simplicity, we assume that the matrix $\Phi S^{(j)}$ has rank s, which is the case almost surely for Gaussian projections; this implies that $\hat\eta^{(j)}$ is unique, but our result applies in all situations as we are only interested in the denoised response vector. We now consider the average $\hat\theta = \frac{1}{m}\sum_{j=1}^m S^{(j)}\hat\eta^{(j)}$.

We thus consider the estimator $\hat\eta^{(j)} = \big((S^{(j)})^\top\Phi^\top\Phi S^{(j)}\big)^{-1}(S^{(j)})^\top\Phi^\top y \in \mathbb{R}^s$, obtained from the normal equation $(S^{(j)})^\top\Phi^\top\Phi S^{(j)}\hat\eta^{(j)} = (S^{(j)})^\top\Phi^\top y$, with denoised response vector

$$\hat y^{(j)} = \Phi S^{(j)}\hat\eta^{(j)} = \Phi S^{(j)}\big((S^{(j)})^\top\Phi^\top\Phi S^{(j)}\big)^{-1}(S^{(j)})^\top\Phi^\top y \in \mathbb{R}^n.$$

Denoting $\Pi^{(j)} = \Phi S^{(j)}\big((S^{(j)})^\top\Phi^\top\Phi S^{(j)}\big)^{-1}(S^{(j)})^\top\Phi^\top$, it is of the form $\hat y^{(j)} = \Pi^{(j)}y$. The matrix $\Pi^{(j)}$ is almost surely an orthogonal projection matrix into an s-dimensional vector space, and its expectation is denoted $\Delta \in \mathbb{R}^{n\times n}$, and is such that $\operatorname{tr}(\Delta) = s$. We have moreover $0 \preccurlyeq \Delta \preccurlyeq I$, that is, all eigenvalues of Δ are between 0 and 1.

We can then compute expectations and variances:

$$\begin{aligned}
\mathbb{E}_{S^{(j)}}\big[\hat y^{(j)}\big] &= \mathbb{E}_{S^{(j)}}\big[\Pi^{(j)}y\big] = \Delta y = \Delta\big[\Phi\theta_* + \varepsilon\big] = \Delta\varepsilon + \Delta\Phi\theta_* \\
\mathbb{E}_{S^{(j)}}\big[\hat y^{(j)}\big] - \Phi\theta_* &= \Delta\varepsilon + [\Delta - I]\Phi\theta_* \\
\mathbb{E}_{S^{(j)}}\big\|\hat y^{(j)} - \mathbb{E}_{S^{(j)}}\big[\hat y^{(j)}\big]\big\|_2^2 &= \mathbb{E}_{S^{(j)}}\big\|(\Pi^{(j)} - \Delta)y\big\|_2^2 = y^\top\mathbb{E}_{S^{(j)}}\Big[\big(\Pi^{(j)} - \Delta\big)^2\Big]y \\
&= y^\top\mathbb{E}_{S^{(j)}}\Big[\Pi^{(j)} - \Delta\Pi^{(j)} - \Pi^{(j)}\Delta + \Delta^2\Big]y \text{ since } \Pi^{(j)}\Pi^{(j)} = \Pi^{(j)}, \\
&= y^\top\big(\Delta - \Delta^2\big)y.
\end{aligned}$$

Thus, the overall (fixed design) expected generalization error is equal to:

$$\begin{aligned}
& \frac{1}{n}\mathbb{E}_{\varepsilon,S}\Big\|\frac{1}{m}\sum_{j=1}^{m}\hat{y}^{(j)} - \Phi\theta_*\Big\|_2^2 \\
= \ & \frac{1}{n}\mathbb{E}_{\varepsilon}\Big[\big\|\mathbb{E}_{S^{(1)}}\big[\hat{y}^{(1)}\big] - \Phi\theta_*\big\|_2^2 + \frac{1}{m}\mathbb{E}_{S^{(1)}}\big\|\hat{y}^{(1)} - \mathbb{E}_{S^{(1)}}\big[\hat{y}^{(1)}\big]\big\|_2^2\Big] \\
& \hspace{6cm} \text{by taking expectations with respect to all } S^{(j)}, \\
= \ & \frac{1}{n}\mathbb{E}_{\varepsilon}\Big[\big\|\Delta\varepsilon + [\Delta - I]\Phi\theta_*\big\|_2^2 + \frac{1}{m}y^\top(\Delta - \Delta^2)y\Big] \text{ using the expressions above,} \\
= \ & \frac{\sigma^2}{n}\operatorname{tr}(\Delta^2) + \frac{1}{n}\theta_*^\top\Phi^\top[I-\Delta]^2\Phi\theta_* + \frac{1}{nm}\big[\sigma^2(\operatorname{tr}(\Delta) - \operatorname{tr}(\Delta^2)) + \theta_*^\top\Phi^\top\big(\Delta - \Delta^2\big)\Phi\theta_*\big] \\
& \hspace{2cm} \text{using the model } y = \Phi\theta_* + \varepsilon \text{ and the fact that } \mathbb{E}[\varepsilon] = 0 \text{ and } \mathbb{E}[\varepsilon\varepsilon^\top] = \sigma^2 I, \\
= \ & \frac{\sigma^2}{n}\big(1 - \frac{1}{m}\big)\operatorname{tr}(\Delta^2) + \frac{\sigma^2 s}{nm} + \frac{1}{n}\theta_*^\top\Phi^\top[\Delta - I]^2\Phi\theta_* + \frac{1}{nm}\theta_*^\top\Phi^\top\big(\Delta - \Delta^2\big)\Phi\theta_* \\
= \ & \frac{\sigma^2}{n}\big(1 - \frac{1}{m}\big)\operatorname{tr}(\Delta^2) + \frac{\sigma^2 s}{nm} + \frac{1}{n}\theta_*^\top\Phi^\top\big[I - \Delta + (\frac{1}{m} - 1)(\Delta - \Delta^2)\big]\Phi\theta_* \\
\leqslant \ & \frac{\sigma^2 s}{n} + \frac{1}{n}\theta_*^\top\Phi^\top\big[I - \Delta\big]\Phi\theta_*, \text{ since } \Delta^2 \preccurlyeq \Delta,
\end{aligned}$$

which is the value for $m = 1$ (single replication). Note that the expectation (before taking the bound) decreases in m. We now follow Kabán (2014); Thanei et al. (2017) to bound the matrix $I - \Delta$.

Since Δ is the expectation of a projection matrix, we already know that $0 \preccurlyeq \Delta \preccurlyeq I$. We omit the superscript $^{(j)}$ for clarity, and consider $\Pi = \Phi S\big(S^\top\Phi^\top\Phi S\big)^{-1}S^\top\Phi$. For any vector $z \in \mathbb{R}^n$, we consider:

$$\begin{aligned}
z^\top(I - \Delta)z &= \mathbb{E}_S\big[z^\top(I - \Pi)z\big] = \mathbb{E}_S\big[z^\top z - z^\top\Phi S\big(S^\top\Phi^\top\Phi S\big)^{-1}S^\top\Phi^\top z\big] \\
&= \mathbb{E}_S\Big[\min_{u\in\mathbb{R}^s}\|z - \Phi S u\|_2^2\Big] \text{ by definition of projections,} \\
&\leqslant \mathbb{E}_S\Big[\min_{v\in\mathbb{R}^d}\|z - \Phi SS^\top v\|_2^2\Big] \text{ by minimizing over a smaller subspace,} \\
&\leqslant \min_{v\in\mathbb{R}^d}\mathbb{E}_S\Big[\|z - \Phi SS^\top v\|_2^2\Big] \text{ by properties of the expectation.}
\end{aligned}$$

We can expand to get:

$$\mathbb{E}_S\Big[\|z - \Phi SS^\top v\|_2^2\Big] = \|z\|_2^2 - 2z^\top\Phi\mathbb{E}_S\big[SS^\top\big]v + v^\top\mathbb{E}_S\big[SS^\top\Phi^\top\Phi SS^\top\big]v,$$

leading to, after selecting the optimal v as $v = \big(\mathbb{E}_S\big[SS^\top\Phi^\top\Phi SS^\top\big]\big)^{-1}\mathbb{E}_S\big[SS^\top\big]\Phi^\top z$,

$$z^\top(I - \Delta)z \leqslant z^\top\Big(I - \Phi\mathbb{E}_S\big[SS^\top\big]\big(\mathbb{E}_S\big[SS^\top\Phi^\top\Phi SS^\top\big]\big)^{-1}\mathbb{E}_S\big[SS^\top\big]\Phi^\top\Big)z.$$

We then need to apply to $z = \Phi\theta_*$, and get:

$$\theta_*^\top\Phi^\top\big[I - \Delta\big]\Phi\theta_* \leqslant \theta_*^\top\Phi^\top\Big(I - \Phi\mathbb{E}_S\big[SS^\top\big]\big(\mathbb{E}_S\big[SS^\top\Phi^\top\Phi SS^\top\big]\big)^{-1}\mathbb{E}_S\big[SS^\top\big]\Phi^\top\Big)\Phi\theta_*.$$

Thus, we get an overall upper bound of

$$\frac{\sigma^2 s}{n} + \frac{1}{n}\theta_*^\top \Phi^\top \Big(I - \Phi \mathbb{E}_S\big[SS^\top\big]\big(\mathbb{E}_S\big[SS^\top\Phi^\top\Phi SS^\top\big]\big)^{-1}\mathbb{E}_S\big[SS^\top\big]\Phi^\top\Big)\Phi\theta_*.$$

As shown below for special cases, we obtain a bias-variance trade-off similar to Eq. (3.6) for ridge regression in Section 3.6, but now with random projections. Note that in the fixed design setting, there is no explosion of the testing performance when $s = n$ (as opposed to the random design setting studied in Section 12.2 in the context of "double descent").

Gaussian projections. If we assume Gaussian random projections, with $S \in \mathbb{R}^{d\times s}$ with independent standard Gaussian components, we get, from properties of the Wishart distribution:[3]

$$\mathbb{E}_S[SS^\top] = sI \text{ and } \mathbb{E}_S\big[SS^\top\Phi^\top\Phi SS^\top\big] = s(s+1)\Phi^\top\Phi + s\,\mathrm{tr}(\Phi^\top\Phi)I.$$

We then get:

$$\begin{aligned}
\theta_*^\top\Phi^\top\big[I-\Delta\big]\Phi\theta_* &\leqslant \theta_*^\top \Phi^\top \Big(I - \Phi \mathbb{E}_S\big[SS^\top\big]\big(\mathbb{E}_S\big[SS^\top\Phi^\top\Phi SS^\top\big]\big)^{-1}\mathbb{E}_S\big[SS^\top\big]\Phi^\top\Big)\Phi\theta_* \\
&= \theta_*^\top \Phi^\top \Big(I - s^2\Phi\big(s(s+1)\Phi^\top\Phi + s\,\mathrm{tr}(\Phi^\top\Phi)I\big)^{-1}\Phi^\top\Big)\Phi\theta_* \\
&= \theta_*^\top \Phi^\top\Phi\big(\Phi^\top\Phi + \mathrm{tr}(\Phi^\top\Phi)I\big)\big((s+1)\Phi^\top\Phi + \mathrm{tr}(\Phi^\top\Phi)I\big)^{-1}\theta_* \\
&\leqslant 2\,\mathrm{tr}(\Phi^\top\Phi)\cdot\theta_*^\top\Phi^\top\Phi\big((s+1)\Phi^\top\Phi + \mathrm{tr}(\Phi^\top\Phi)I\big)^{-1}\theta_* \\
&\leqslant 2\,\mathrm{tr}(\Phi^\top\Phi)\frac{\|\theta_*\|_2^2}{s+1}.
\end{aligned}$$

The overall excess risk is then less than

$$\frac{\sigma^2 s}{n} + \frac{2}{n}\,\mathrm{tr}(\Phi^\top\Phi)\frac{\|\theta_*\|_2^2}{s+1}, \tag{10.1}$$

which is exactly of the form obtained for ridge regression in Eq. (3.6) with $s \sim \frac{\mathrm{tr}(\Phi^\top\Phi)}{\lambda}$. We can consider other sketching matrices with additional properties, such as sparsity (see the exercise below).

Exercise 10.4 *We consider a sketching matrix $S \in \mathbb{R}^{d\times s}$, where each column is equal to one of the d canonical basis vectors of $\mathbb{R}^d$, selected uniformly at random and independently. Compute $\mathbb{E}[SS^\top]$, as well as, $\mathbb{E}_S\big[SS^\top\Phi^\top\Phi SS^\top\big]$, as well as a bound similar to Eq. (10.1).*

Kernel methods. (♦) The random projection idea can be extended to kernel methods from Chapter 7. We consider the kernel matrix $K = \Phi\Phi^\top \in \mathbb{R}^{n\times n}$, and the assumption

[3]If $W = S_1 S_1^\top$ for $S_1 \in \mathbb{R}^{n\times s}$ with independent standard Gaussian components, then $\mathbb{E}[W] = sI$ and for an $n\times n$ diagonal matrix D we have $\mathbb{E}[WD^2W] = s(s+1)D^2 + s\,\mathrm{tr}(D^2)I$.

$y = \Phi\theta_* + \varepsilon$ with $\|\theta_*\|_2$ bounded, is turned into $y = y_* + \varepsilon$ with $y_*^\top K^{-1} y_*$ bounded. This corresponds to $y_* = K\alpha$, with an RKHS norm $\alpha^\top K \alpha$. We then consider a random "sketch" $\hat{\Phi} \in \mathbb{R}^{n\times s}$ and an approximate kernel matrix $\hat{K}$. We then obtain an estimate $\hat{y} = \hat{\Phi}(\hat{\Phi}^\top\hat{\Phi})^{-1}\hat{\Phi}^\top y$. The matrix Π above is then $\Pi = \hat{\Phi}(\hat{\Phi}^\top\hat{\Phi})^{-1}\hat{\Phi}^\top$, and for the analysis, we need to compute its expectation Δ. We have, following the same reasoning as above, for an arbitrary deterministic $z \in \mathbb{R}^n$:

$$\begin{aligned}
z^\top(I-\Delta)z &= \mathbb{E}_{\hat{\Phi}}\big[z^\top(I-\Pi)z\big] = \mathbb{E}_{\hat{\Phi}}\big[z^\top z - z^\top\hat{\Phi}\big(\hat{\Phi}^\top\hat{\Phi}\big)^{-1}\hat{\Phi}^\top z\big] \\
&= \mathbb{E}_{\hat{\Phi}}\Big[\min_{u\in\mathbb{R}^s}\|z-\hat{\Phi}u\|_2^2\Big] \text{ by definition of projections,} \\
&\leqslant \mathbb{E}_{\hat{\Phi}}\Big[\min_{v\in\mathbb{R}^n}\|z-\hat{\Phi}\hat{\Phi}^\top v\|_2^2\Big] \text{ by minimizing over a smaller subspace,} \\
&\leqslant \min_{v\in\mathbb{R}^n}\mathbb{E}_{\hat{\Phi}}\Big[\|z-\hat{\Phi}\hat{\Phi}^\top v\|_2^2\Big] \text{ by properties of the expectation.}
\end{aligned}$$

We can expand to get:

$$\mathbb{E}_{\hat{\Phi}}\Big[\|z-\hat{\Phi}\hat{\Phi}^\top v\|_2^2\Big] = \|z\|_2^2 - 2z^\top\mathbb{E}_{\hat{\Phi}}\big[\hat{\Phi}\hat{\Phi}^\top\big]v + v^\top\mathbb{E}_{\hat{\Phi}}\big[\hat{\Phi}\hat{\Phi}^\top\hat{\Phi}\hat{\Phi}^\top\big]v,$$

leading to, after selecting the optimal v as $v = \big(\mathbb{E}_{\hat{\Phi}}\big[\hat{\Phi}\hat{\Phi}^\top\hat{\Phi}\hat{\Phi}^\top\big]\big)^{-1}\mathbb{E}_{\hat{\Phi}}\big[\hat{\Phi}\hat{\Phi}^\top\big]z$,

$$z^\top(I-\Delta)z \leqslant z^\top\Big(I - \mathbb{E}_{\hat{\Phi}}\big[\hat{\Phi}\hat{\Phi}^\top\big]\big(\mathbb{E}_{\hat{\Phi}}\big[\hat{\Phi}\hat{\Phi}^\top\hat{\Phi}\hat{\Phi}^\top\big]\big)^{-1}\mathbb{E}_{\hat{\Phi}}\big[\hat{\Phi}\hat{\Phi}^\top\big]\Big)z.$$

We then need to apply to $z = y_*$, we get that

$$\theta_*^\top\Phi^\top\big[I-\Delta\big]\Phi\theta_* \leqslant y_*^\top\Big(I - \mathbb{E}_{\hat{\Phi}}\big[\hat{\Phi}\hat{\Phi}^\top\big]\big(\mathbb{E}_{\hat{\Phi}}\big[\hat{\Phi}\hat{\Phi}^\top\hat{\Phi}\hat{\Phi}^\top\big]\big)^{-1}\mathbb{E}_{\hat{\Phi}}\big[\hat{\Phi}\hat{\Phi}^\top\big]\Big)y_*.$$

We can, for example, consider each column of $\hat{\Phi}$ to be sampled from a normal distribution with mean zero and covariance matrix K, for which we have:

$$\mathbb{E}_{\hat{\Phi}}\big[\hat{\Phi}\hat{\Phi}^\top\big] = sK \text{ and } \mathbb{E}_{\hat{\Phi}}\big[\hat{\Phi}\hat{\Phi}^\top\hat{\Phi}\hat{\Phi}^\top\big] = s(s+1)K^2 + s\,\mathrm{tr}(K)\cdot K.$$

This leads to the bound $\dfrac{\sigma^2 s}{n} + \dfrac{2}{n}\,\mathrm{tr}(K)\dfrac{y_*^\top K^{-1}y_*}{s+1}$, which is exactly the bound in Eq. (10.1) in the kernel context. However, it is not interesting in practice as it requires the computation of the kernel matrix K and typically a square root to sample from the multivariate Gaussian distribution, which has running-time complexity $O(n^3)$.

In practice, many kernels come with a random feature expansion of the form $k(x,x') = \mathbb{E}_v\big[\varphi(x,v)\varphi(x',v)\big]$, such that $|\varphi(x,v)| \leqslant R$ almost surely (as presented in Section 7.4). We can then take for each column of $\hat{\Phi}$ the vector $(\varphi(x_1,v),\dots,\varphi(x_n,v))^\top \in \mathbb{R}^n$, for a random independent v. We have then $\mathbb{E}\big[\hat{\Phi}\hat{\Phi}^\top\big] = sK$ by construction, while a short calculation left as an exercise shows that the second-order moment can be bounded as

$$\mathbb{E}_{\hat{\Phi}}\big[\hat{\Phi}\hat{\Phi}^\top\hat{\Phi}\hat{\Phi}^\top\big] \preccurlyeq s(s-1)K + nsR^2K.$$

This leads to the bound $\dfrac{\sigma^2 s}{n} + \dfrac{2}{n}R^2\dfrac{y_*^\top K^{-1}y_*}{s+1}$, which is almost the same as above, but with now an efficient practical algorithm.

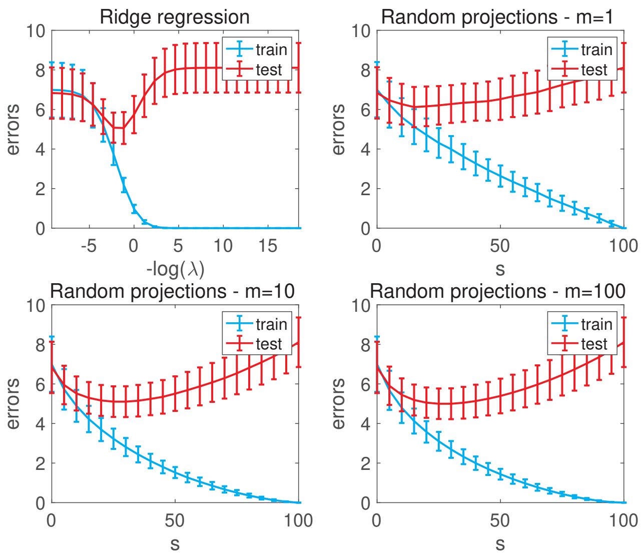

Figure 10.2: Polynomial regression in dimension 20, with polynomials of degree at most 2, with $n = 100$. Top left: training and testing errors for ridge regression in the fixed design setting (the input data are fixed, and only the noise variables are resampled for computing the test error). All other plots: training and testing errors for Gaussian random projections, with different numbers of random projections, $m = 1$ (top right), $m = 10$ (bottom left), and $m = 100$ (bottom right). All curves are averaged over 100 replications (of the noise variables and the random projections).

Experiments. In Figure 10.2, we consider a polynomial regression problem in dimension $d_{\mathcal{X}} = 20$, with polynomials of degree at most 2, and thus a feature space of dimension $d = 1 + d_{\mathcal{X}} + d_{\mathcal{X}}(d_{\mathcal{X}} + 1)/2 = 231$, and compare ridge regression with Gaussian random projections. We see a better performance as m grows, consistent with our bounds.

Johnson-Lindenstrauss lemma (♦). A related classical result in Gaussian random projections shows that n feature vectors $\varphi_1, \dots, \varphi_n \in \mathbb{R}^d$ can be "well-represented" in dimension s by Gaussian random projections, with s growing only logarithmically in n, and independently of the underlying dimension. The following lemma shows that all pairwise distances are preserved (a small modification would lead to all dot-products).

Lemma 10.1 (Johnson and Lindenstrauss, 1984) *Given $\varphi_1, \dots, \varphi_n \in \mathbb{R}^d$, let $S \in \mathbb{R}^{d\times s}$ be random matrix with independent standard Gaussian random variables. Then for any $\varepsilon \in (0, 1/2)$ and $\delta \in (0, 1)$, if $s \geqslant \frac{6}{\varepsilon^2} \log \frac{n^2}{\delta}$, with probability greater than $1 - \delta$, we have:*

$$\forall i, j \in \{1, \dots, n\},\ (1-\varepsilon)\|\varphi_i - \varphi_j\|_2^2 \leqslant \big\|s^{-1/2}S^\top \varphi_i - s^{-1/2}S^\top \varphi_j\|_2^2 \leqslant (1+\varepsilon)\|\varphi_i - \varphi_j\|_2^2. \tag{10.2}$$

Proof (♦) Let $\psi \in \mathbb{R}^d$ with ℓ_2-norm equal to one. The random variable $y = \psi^\top SS^\top \psi$ is the sum of s random variables $\psi^\top S_{\cdot j} S_{\cdot j}^\top \psi$, for $S_{\cdot j}$ the j-th column of S, $j \in \{1, \dots, s\}$. Each of these is the square of $S_{\cdot j}^\top \psi$ which is Gaussian with mean zero and variance equal to $\|\psi\|_2^2 = 1$. Thus, y is a chi-squared random variable. We can thus apply concentration results from Exercise 8.1, leading to

$$\mathbb{P}\big(|y - s| \geqslant s\varepsilon\big) \leqslant \Big(\frac{1-\varepsilon}{\exp(-\varepsilon)}\Big)^{s/2} + \Big(\frac{1+\varepsilon}{\exp(\varepsilon)}\Big)^{s/2}.$$

We can then use the inequality $\log(1+u) \leqslant u - \frac{u^2}{3}$ for any $|u| \leqslant 1/2$, leading to the probability bound $\mathbb{P}\big(|y - s| \geqslant s\varepsilon\big) \leqslant 2\exp\big(-\frac{s}{2}\frac{\varepsilon^2}{3}\big)$. We then apply the reasoning above to the $n(n-1)/2$ vectors $\varphi_i - \varphi_j$, for $i \neq j$, leading to, using a union bound, a probability that Eq. (10.2) is not satisfied with probability less than $n^2 \exp(-s\varepsilon^2/6)$, leading to the desired result. ■

In our context of least-squares regression, the Johnson-Lindenstrauss lemma shows that the kernel matrix is preserved by random projections so that predictions with the projected data should be close to predictions with the original data. The results in this section provide a direct proof aiming at characterizing directly the predictive performance of such random projections.

10.3 Boosting

In the previous section, we focused on uniformly combining the outputs (e.g., plain averaging) of estimators obtained by randomly reweighted versions of the original datasets. Reweighting was performed independently of the performance of the resulting prediction functions, and the training procedures for all predictors could be done in parallel. In this section, we explore *sequential* reweightings of the training datasets that depend on the mistakes made by the current prediction functions.

In the early boosting procedures adapted to binary classification, the original learning procedure was used directly on a reweighted version, e.g., Adaboost (see, e.g., Freund et al., 1999). Our analysis will be carried out for boosting procedures, often referred to as "gradient boosting", which are adapted to real-valued outputs, as done in the rest of the book (noting that for classification, we can use convex surrogates).

The theory of boosting is rich, with many connections, and in this section, we only provide a consistency proof in the simplest setting. See Schapire and Freund (2012) for more details.

10.3.1 Problem set-up

Given an input space $\mathcal{X}$, and n observations $(x_i, y_i) \in \mathcal{X} \times \mathbb{R}$, $i = 1, \dots, n$, we are given a set of predictors $\varphi(w, \cdot) : \mathcal{X} \to \mathbb{R}$, for $w \in \mathcal{W}$, with $\mathcal{W}$ typically a compact subset of a finite-dimensional vector space.

The main assumption is that given weights $\alpha \in \mathbb{R}^n$, one can "easily" find the function $\varphi(w,\cdot)$ that minimizes with respect to $w \in \mathcal{W}$

$$\sum_{i=1}^{n} \alpha_i \varphi(w, x_i),$$

that is, the dot-product between α and the n outputs of $\varphi(w,\cdot)$ on the n observations. In this section, for simplicity, we assume that this minimization can be done exactly. This is often referred to as the "weak learner" assumption. Many examples are available, such as:

- Linear stumps for $\mathcal{X} = \mathbb{R}^d$: $\varphi(w,x) = \pm(w_0^\top x + w_1)_+$, with sometimes the restriction that w has only non-zero components along a single coordinate (where the weak learning tractability assumption is indeed verified). This will lead to a predictor, which is a one-hidden layer neural network, but learned sequentially (rather than by gradient descent on the empirical risk). In the context of binary classification, the weak learners are sometimes thresholded to values in $\{-1, 1\}$ by taking their signs.
- Decision trees for $\mathcal{X} = \mathbb{R}^d$: we consider here the space of piecewise constant functions of x, where the pieces with constant values are obtained by recursively partitioning the input space into half-spaces with normals along one of the coordinate axes. In this situation, the set of functions is more easily characterized through the estimation algorithm. See Chen and Guestrin (2016) for an efficient implementation of a boosting algorithm based on decision trees (referred to as "XGBoost").

In this section, we assume bounded features, that is, for all $w \in \mathcal{W}$, and inputs $x \in \mathcal{X}$, $|\varphi(x,w)| \leqslant R$. Moreover, for simplicity, we assume that the set of feature functions $\{\varphi(\cdot,w),\ w \in \mathcal{W}\}$ is centrally symmetric, which is the case for the examples above.

Boosting procedures will make sequential calls to the weak learner oracle that outputs $w_1, \dots, w_t \in \mathcal{W}$ with t the number of iterations, and *linearly combine* the function $\varphi(w_1,\cdot), \dots, \varphi(w_t,\cdot)$. Therefore, the set of predictors that are explored are not only the functions $\varphi(w,\cdot)$, but all linear combinations, that is, functions of the form

$$f(x) = \int_{\mathcal{W}} \varphi(w,x) d\nu(w), \tag{10.3}$$

for ν a (signed) measure on $\mathcal{W}$, which we assume to have finite mass.

To avoid overfitting, some norm that will be explicitly or implicitly controlled needs to be defined. As done in Section 9.3.2 with neural networks, we will consider an L_1-norm, namely the total variation of ν, that is:

$$\int_{\mathcal{W}} |d\nu(w)|.$$

Note that since we have assumed that the features are centrally symmetric, assuming that ν is a positive measure does not change anything.

For functions $f : \mathcal{X} \to \mathbb{R}$ that can be represented as integrals in Eq. (10.3), the minimal value of $\int_{\mathcal{W}} |d\nu(w)|$ is referred to as the "variation norm" (Kurková and Sanguineti, 2001), or the "atomic norm" (Chandrasekaran et al., 2012), of f, and the set of functions with finite norm will be denoted $\mathcal{F}_1$, with a norm γ_1. Like in Section 9.3.2, this is to distinguish it from the squared norm $\int_{\mathcal{W}} \big| \frac{d\nu(w)}{d\tau(w)} \big|^2 d\tau(w)$ for a fixed positive measure τ, which corresponds to a reproducing kernel Hilbert space (see Chapter 7).

Note that by definition, for any $w \in \mathcal{W}$, $\gamma_1(\varphi(w,\cdot)) \leqslant 1$, since we can represent this function by the measure $\nu = \delta_w$. Since we will optimize over the realizations of the features on the data, we denote by $\psi(w) \in \mathbb{R}^n$ the vector so that $\psi(w)_i = \varphi(x_i, w)$. Since $|\varphi(x,w)| \leqslant R$ for all w and x, $\|\psi(w)\|_2 \leqslant R\sqrt{n}$ for all w. By restricting to values on $x_1, \dots, x_n$, we obtain a penalty defined on $\mathbb{R}^n$ with a definition similar to γ_1 defined on functions from $\mathcal{X}$ to $\mathbb{R}$, with more properties we will need for our proofs.

Gauge function. We define the function $\gamma : \mathbb{R}^n \to \mathbb{R}$ as the infimum of $\int_{\mathcal{W}} |d\nu(w)|$ over all positive measures such that $u = \int_{\mathcal{W}} \psi(w) d\nu(w)$. This function is usually referred to as the "gauge" function associated with the convex hull of all $\psi(w)$, $w \in \mathcal{W}$ (Rockafellar, 1997). The gauge function γ is always convex and positively homogeneous. Since we further assumed central symmetry of the features, that is, the set $\{\psi(w), w \in \mathcal{W}\} \subset \mathbb{R}^n$ is centrally symmetric, $\gamma(-u) = \gamma(u)$ for all $u \in \mathbb{R}^n$. Given our bounded norm assumption $\|\psi(w)\|_2 \leqslant R\sqrt{n}$, we have, for any u such that $\gamma(u)$ is finite (and with associated measure ν), $\|u\|_2 = \big\| \int_{\mathcal{W}} \psi(w) d\nu(w) \big\|_2 \leqslant \int_{\mathcal{W}} \|\psi(w)\|_2 |d\nu(w)| \leqslant R\sqrt{n}\gamma(u)$.

The gauge function may not be a norm since it may not be finite everywhere, that is, there may exist $u \in \mathbb{R}^n$ which cannot be expressed as a linear combination of feature vectors $\psi(w)$, $w \in \mathcal{W}$. We may, however, define a notion of dual gauge function, called a "polar" gauge $\gamma^* : \mathbb{R}^n \to \mathbb{R}$, as $\gamma^*(v) = \sup_{w \in \mathcal{W}} \psi(w)^\top v$, which leads to a form a Cauchy-Schwarz inequality, as $u^\top v \leqslant \gamma(u)\gamma^*(v)$ (see Rockafellar, 1997, Chapter 15, for more details).

Assumptions. Following our traditional empirical risk minimization framework presented in Chapter 4, we consider a loss function $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}$, both for regression and classification. Since we will need differentiable loss functions, our developments are restricted to the logistic loss, the exponential loss, and the square loss. We denote by $\ell_i : \mathbb{R} \to \mathbb{R}$ the loss the observation (x_i, y_i), that is, $\ell_i(u_i) = \ell(y_i, u_i)$. We thus consider the logistic loss $\ell_i(u_i) = \log(1 + \exp(-y_i u_i))$ and the exponential loss $\ell_i(u_i) = \exp(-y_i u_i)$ when $y_i \in \{-1, 1\}$, or the square loss $\ell_i(u_i) = \frac{1}{2}(y_i - u_i)^2$ when $y_i \in \mathbb{R}$.

In our optimization convergence proofs in Section 10.3.5, we will need that each loss ℓ_i is smooth, with smoothness constant G_2 (e.g., $1/4$ for the logistic loss and 1 for the square loss, and $+\infty$ for the exponential loss). This leads to a loss function $F : \mathbb{R}^n \to \mathbb{R}$, defined as $F(u) = \frac{1}{n}\sum_{i=1}^n \ell_i(u_i)$, which is (G_2/n)-smooth. For the statistical consistency proof, we will also need that the loss functions are G_1-Lipschitz continuous, which only applies to logistic regression, and that $\ell_i(0)$ has a uniform bound G_0 (for logistic regression, $G_0 = \log 2$).

Finite $\mathcal{W}$. While boosting methods can be applied for any compact set $\mathcal{W}$ (as long as the minimization oracle is available), an interesting special case corresponds to finite sets

$\mathcal{W} = \{w_1, \dots, w_d\}$. The optimization problem that we aim to solve is the minimization of $F(u)$, for u in the span of all $\psi(w_1), \dots, \psi(w_d)$, which we can rewrite as:

$$\min_{u \in \mathbb{R}^n} F(u) \text{ such that } \exists \alpha \in \mathbb{R}^d,\ u = \sum_{j=1}^d \alpha_j \psi(w_j) = \min_{\alpha \in \mathbb{R}^d} F\Big(\sum_{j=1}^d \alpha_j \psi(w_j)\Big). \tag{10.4}$$

We can thus either see this as an optimization problem in u *or* in α, and, given our assumptions regarding central symmetry, the norm γ on u is upper-bounded by the ℓ_1-norm of α. Seeing Eq. (10.4) as a problem in u may be advantageous because of strong-convexity properties that could be lost for the problem in α (in particular when $n \leqslant d$): for example, for the square loss, where F is strongly-convex, the optimization problem in u is strongly-convex, and thus exhibits linear convergence, while the problem in α is not strongly-convex (but still exhibits linear convergence for other reasons, see Section 12.1.1).

10.3.2 Incremental learning

The simplest version of boosting-like algorithms aims to construct linear combinations of functions of the form $x \mapsto \varphi(w_t, x)$ by selecting incrementally $w_t \in \mathcal{W}$. Starting from the function $g_0 = 0$, we thus consider the simplest update

$$g_t = g_{t-1} + b_t \varphi(w_t, \cdot), \tag{10.5}$$

where the linear combination coefficients for $\varphi(w_1, \cdot), \dots, \varphi(w_{t-1}, \cdot)$ are not changed once they are computed. Given the empirical risk $\widehat{\mathcal{R}}(f) = \frac{1}{n}\sum_{i=1}^n \ell(y_i, f(x_i))$, a natural criterion for the choice of $b_t \in \mathbb{R}$ and $w_t \in \mathcal{W}$ is to solve the optimization problem

$$\min_{b_t \in \mathbb{R}_+,\ w_t \in \mathcal{W}} \widehat{\mathcal{R}}\big(g_{t-1} + b_t \varphi(w_t, \cdot)\big). \tag{10.6}$$

With our notations, and since only values at $x_1, \dots, x_n$ are used for the functions g_t, we can represent them with their values on these points, that is, by a vector $u_t \in \mathbb{R}^n$ such that $(u_t)_i = g_t(x_i)$ for all $i \in \{1, \dots, n\}$. The update in Eq. (10.5) then becomes

$$u_t = u_{t-1} + b_t \psi(w_t),$$

and the update in Eq. (10.6) becomes

$$\min_{b_t \in \mathbb{R}_+,\ w_t \in \mathcal{W}} F\big(u_{t-1} + b_t \psi(w_t)\big). \tag{10.7}$$

This minimization is easily done in two situations: for the square loss, leading to matching pursuit (Mallat and Zhang, 1993) and for the exponential loss, leading to Adaboost (Freund and Schapire, 1996). We now present these two classical algorithms and some elements of analysis of their convergence rates for optimizing the empirical risk. We then analyze the expected risk, which is more involved when the goal is to obtain a convergence rate with early stopping.

10.3.3 Matching pursuit

Matching pursuit corresponds to the iteration in Eq. (10.7) for the square loss, with applications beyond machine learning, in particular in signal processing (Mallat and Zhang, 1993). For simplicity, only in this section, we assumed that each $x \mapsto \varphi(x, w)$, for $w \in \mathcal{W}$, is normalized on the data, that is $\sum_{i=1}^n \varphi(x_i, w)^2 = \|\psi(w)\|_2^2 = n$. This implies that for all $u \in \mathbb{R}^n$, $\|u\|_2 \leqslant \sqrt{n}\gamma(u)$.

In our context of empirical risk minimization, the square loss corresponds to $F(u) = \frac{1}{2n}\|y - u\|_2^2$, and, because of the normalization, we have:

$$\begin{array}{rcl} F(u_t) & = & F(u_{t-1}) + F'(u_{t-1})^\top (u_t - u_{t-1}) + \dfrac{1}{2n}\|u_t - u_{t-1}\|_2^2 \\ & = & F(u_{t-1}) + F'(u_{t-1})^\top b_t \psi(w_t) + \dfrac{b_t^2}{2}. \end{array}$$

Optimizing with respect to $b_t \in \mathbb{R}$ leads to $b_t = -F'(u_{t-1})^\top \psi(w_t)$, leading to the optimal value

$$F(u_{t-1}) - \frac{1}{2}|F'(u_{t-1})^\top \psi(w_t)|^2.$$

Since $F'(u_{t-1}) = \frac{1}{n}(u_{t-1} - y)$, the iteration can then be written as, initialized with $u_0 = 0$, for $t \geqslant 1$:

$$\left\{ \begin{array}{rcl} w_t & = & \arg\max_{w \in \mathcal{W}} \big|(u_{t-1} - y)^\top \psi(w)\big| \\ u_t & = & u_{t-1} - \frac{1}{n}\big|(u_{t-1} - y)^\top \psi(w_t)\big|\psi(w_t) = u_{t-1} - \frac{1}{n}\gamma^*(u_{t-1} - y)\psi(w_t), \end{array} \right.$$

by definition of the polar gauge function γ^*.

Slow convergence. The minimizer of $F(u) = \frac{1}{2n}\|u - y\|_2^2$ is $u_* = y$. It may or may not be such that $\gamma(y)$ is finite. In this section on matching pursuit, we assume it is, but we consider the general case in Section 10.3.5. It turns out that the penalty $\gamma(y)$ provides an explicit control of the convergence rate of u_t towards y. Indeed, it can be shown that the matching pursuit algorithm converges with a rate proportional to $\gamma(y)$, that is,

$$\frac{1}{n}\|y - u_t\|_2^2 \leqslant \gamma(y)^2 t^{-1/3}.$$

See DeVore and Temlyakov (1996) for a detailed result (proved in Exercise 10.6), Section 10.3.5 for a related result for all smooth loss functions (and with a detailed proof), and Sil'nichenko (2004) for an improved dependence on t, and Klusowski and Siegel (2023) for lower bounds.

Fast convergence of the empirical risk. As already obtained by Mallat and Zhang (1993), exponential rates can be obtained with the stronger assumption that γ is a norm on $\mathbb{R}^n$, and then we have by equivalence of norms: $\sqrt{n}\kappa\gamma(u) \leqslant \|u\|_2$, and $\gamma^*(v) \geqslant \kappa\sqrt{n}\|v\|_2$, for a constant $\kappa > 0$ which has to be less than 1, since $\|u\|_2 \leqslant \sqrt{n}\gamma(u)$. For finite sets $\mathcal{W} = \{w_1, \dots, w_d\}$, this corresponds to the kernel matrix $\sum_{i=1}^d \psi(w_i)\psi(w_i)^\top \in$

$\mathbb{R}^{n\times n}$ being invertible. As shown below, this ensures constant multiplicative progress across matching pursuit iterations. Indeed, we then have:

$$\frac{1}{2n}\|y-u_t\|_2^2 = \frac{1}{2n}\|y-u_{t-1}\|_2^2 - \frac{1}{2n^2}\gamma^*(u_{t-1}-y)^2 \leqslant (1-\kappa^2)\cdot\frac{1}{2n}\|y-u_{t-1}\|_2^2,$$

leading to exponential convergence.

Exercise 10.5 *(♦) Orthogonal matching pursuit is a modification of matching pursuit which, once $w_t \in \mathcal{W}$ has been selected, defines u_t as the minimizer of F over the span of all previously selected feature vectors $\psi(w_1),\dots,\psi(w_t)$. Show that $\frac{1}{n}\|y-u_t\|_2^2 \leqslant \gamma(y)^2 t^{-1}$.*

10.3.4 Adaboost

Adaboost (Freund and Schapire, 1996) corresponds to the binary classification case, where we assume that $\varphi(x,w)\in\{-1,1\}$, that is, all weak learners are already classification functions, or, equivalently, $\psi(w)\in\{-1,1\}^n$, and we use the exponential loss, that is,

$$F(u) = \frac{1}{n}\sum_{i=1}^n \exp(-y_i u_i).$$

We can then implement Eq. (10.7) by solving

$$\min_{b_t\in\mathbb{R}, w_t\in\mathcal{W}} F(u_{t-1}+b_t\psi(w_t)) = \min_{b_t\in\mathbb{R}, w_t\in\mathcal{W}} \frac{1}{n}\sum_{i=1}^n \exp(-y_i(u_{t-1})_i)\exp(-b_t y_i\psi(w_t)_i).$$

Using the fact that $y_i\psi(w_t)_i\in\{-1,1\}$ for all $i\in\{1,\dots,n\}$, this is equivalent to

$$\min_{b_t\in\mathbb{R}, w_t\in\mathcal{W}} \sum_{i=1}^n \Big\{\frac{e^{-b_t}}{n}\frac{1+y_i\psi(w_t)_i}{2} + \frac{e^{b_t}}{n}\frac{1-y_i\psi(w_t)_i}{2}\Big\}e^{-y_i(u_{t-1})_i},$$

with an objective function proportional to $\sum_{i=1}^n \big\{\frac{e^{-b_t}}{n}\frac{1+y_i\psi(w_t)_i}{2} + \frac{e^{b_t}}{n}\frac{1-y_i\psi(w_t)_i}{2}\big\}\pi_i$, where π is a vector in the simplex defined as $\pi_i = \frac{e^{-y_i(u_{t-1})_i}}{\sum_{j=1}^n e^{-y_j(u_{t-1})_j}}$.

Given $w_t\in\mathcal{W}$, the optimal b_t is obtained by minimizing a function of the form $e^{-b_t}a_- + e^{b_t}a_+$, for some constants a_+ and a_-, which is attained as $b_t = \frac{1}{2}\log\frac{a_-}{a_+}$, with optimal value equal to $2\sqrt{a_-a_+}$. Thus, the optimal b_t is equal to

$$b_t = \frac{1}{2}\log\frac{1+\sum_{i=1}^n y_i\psi(w_t)_i\pi_i}{1-\sum_{i=1}^n y_i\psi(w_t)_i\pi_i},$$

and the resulting objective function (that depends on w_t) is equal to

$$F(u_{t-1})\Big[1-\Big(\sum_{i=1}^n y_i\psi(w_t)_i\pi_i\Big)^2\Big].$$

We can thus obtain w_t by maximizing $\big|\sum_{i=1}^n y_i \psi(w_t)_i \pi_i\big|$. Since we have assumed central symmetry of the weights, we can equivalently maximize $\sum_{i=1}^n y_i \psi(w_t)_i \pi_i$, which corresponds to finding the weak learner with minimal 0-1 classification error weighted by π. We thus get the following iteration:

$$\begin{cases} \pi_i = \dfrac{e^{-y_i (u_{t-1})_i}}{\sum_{j=1}^n e^{-y_j (u_{t-1})_j}} \text{ for } i \in \{1,\dots,n\} \\ w_t \in \arg\max\limits_{w \in \mathcal{W}} \displaystyle\sum_{i=1}^n y_i \psi(w_t)_i \pi_i \\ u_t = u_{t-1} + \dfrac{1}{2} \log \dfrac{1 + \sum_{i=1}^n y_i \psi(w_t)_i \pi_i}{1 - \sum_{i=1}^n y_i \psi(w_t)_i \pi_i} \psi(w_t). \end{cases}$$

After this iteration, we have

$$F(u_t) = F(u_{t-1}) \Big[1 - \Big(\sum_{i=1}^n y_i \psi(w_t)_i \pi_i \Big)^2 \Big].$$

Therefore, the empirical risk (with the exponential loss) strictly decreases if the weak learner gets an empirical weighted 0-1 loss strictly less than 1/2 (corresponding to the dot product with y being strictly positive). If the error rate is always less than a constant, an assumption referred to as weak learnability, we obtain a linear convergence. Note that if we make the same assumption as in matching pursuit in Section 10.3.3, then $\sum_{i=1}^n y_i \psi(w_t)_i \pi_i = \gamma^*(\pi \circ y) \geqslant \rho \sqrt{n} \|\pi\|_2 \geqslant \rho \|\pi\|_1 \geqslant \rho$, and we have a similar exponential convergence rate.

10.3.5 Greedy algorithm based on gradient boosting

In this section, we describe a boosting algorithm which, at each iteration, performs a first-order Taylor expansion at the current point (which requires to compute derivatives of the loss functions) and find the weak learner $x \mapsto \varphi(w, x)$ that makes most progress for this approximation of the risk. We thus consider the following "greedy" algorithm, starting from the zero function $g_0 = 0$, and iterating over $t \geqslant 1$:

- Loss gradient computations: compute $\alpha_i = \ell_i'(g_{t-1}(x_i))$ for $i \in \{1,\dots,n\}$.
- Weak learner: compute $w_t \in \mathcal{W}$ that minimizes $\sum_{i=1}^n \alpha_i \varphi(w, x_i)$ with respect to $w \in \mathcal{W}$. Equivalently, using our notations in $\mathbb{R}^n$, we minimize $F'(u_{t-1})^\top \psi(w)$ with respect to $w \in \mathcal{W}$.
- Function update: take $g_t = g_{t-1} + b_t \varphi(w_t, \cdot)$ for a coefficient $b_t \in \mathbb{R}_+$ that optimizes an upper-bound on the empirical risk. This corresponds to $u_t = u_{t-1} + b_t \psi(w_t)$.

After time t, the prediction function g_t will be a linear combination of the functions $\varphi(w_u, \cdot)$, for $u \in \{1,\dots,t\}$, with only t atoms, thus leading to sparse combinations (in other words, the estimated measure ν is a sum of Diracs). For the square loss, this will exactly be the matching pursuit algorithm presented in Section 10.3.3. In general, these algorithms are often referred to as "gradient boosting" procedures (Friedman, 2001).

We first provide a generic convergence result for the empirical risk (which goes beyond machine learning problems), before proving a convergence rate for the expected risk in Section 10.3.6. We focus on smooth loss functions for the optimization result, while we require a smooth and Lipschitz-continuous loss function for the statistical analysis (such as the logistic loss). For consistency results for the exponential loss, see Bartlett and Traskin (2007).

With our smoothness assumption, we can define the upper-bound on $F(u_t)$ as (using the definition of smoothness in Eq. (5.10)):

$$\begin{aligned} F(u_t) &\leqslant F(u_{t-1}) + F'(u_{t-1})^\top (u_t - u_{t-1}) + \frac{L}{2}\|u_t - u_{t-1}\|_2^2 \\ &\leqslant F(u_{t-1}) + b_t F'(u_{t-1})^\top \psi(w_t) + \frac{L}{2} b_t^2 \|\psi(w_t)\|_2^2 \\ &\qquad\qquad \text{using the expression } u_t = u_{t-1} + b_t\psi(w_t), \\ &\leqslant F(u_{t-1}) + b_t F'(u_{t-1})^\top \psi(w_t) + \frac{L}{2} b_t^2 C^2, \end{aligned} \tag{10.8}$$

if L is the smoothness constant of F and C an uniform upper-bound on all $\|\psi(w)\|_2$, $w \in \mathcal{W}$. This naturally leads to the iteration

$$\left\{ \begin{array}{l} w_t \in \arg\max_{w\in\mathcal{W}} \ F'(u_{t-1})^\top \psi(w) \\ u_t = u_{t-1} - \frac{1}{LC^2} F'(u_{t-1})^\top \psi(w_t), \end{array} \right. \tag{10.9}$$

which we can now analyze, to obtain upper-bounds on both function values and the gauge functions of the iterates.

Proposition 10.1 (convergence of gradient boosting algorithm) *We consider an L-smooth function $F : \mathbb{R}^n \to \mathbb{R}$; we assume that $\psi : \mathcal{W} \to \mathbb{R}^n$ is such that $\|\psi(w)\|_2 \leqslant C$ for all $w \in \mathcal{W}$, and the associated gauge function γ is centrally symmetric. Consider the iteration in Eq. (10.9). Then for any $v \in \mathbb{R}^n$, and $t > 0$, we have:*

$$\big(F(u_t) - F(v)\big)_+ \leqslant \Big(\frac{2LC^2\gamma(u_0 - v)^2 (F(u_0) - F(v))_+^4}{t}\Big)^{1/5},$$

and

$$\gamma(u_t) \leqslant \gamma(u_0) + \frac{\sqrt{t}}{LC^2}\Big(2LC^2[F(u_0) - F(u_t)]\Big)^{1/2}.$$

Proof We have by construction of the iteration and from Eq. (10.8):

$$F(u_t) - F(v) \leqslant F(u_{t-1}) - F(v) - \frac{1}{2LC^2}\gamma^*(F'(u_{t-1}))^2. \tag{10.10}$$

Moreover, using the convexity of F and properties of gauge functions, we have:

$$F(u_t) - F(v) \leqslant F'(u_t)^\top (u_t - v) \leqslant \gamma^*(F'(u_t))\gamma(u_t - v). \tag{10.11}$$

Finally, using the triangular inequality for γ, we obtain, from Eq. (10.9), $\gamma(u_t - v) \leqslant \gamma(u_{t-1} - v) + \frac{1}{LC^2}\gamma^*(F'(u_{t-1}))$, leading to, by recursion, $\gamma(u_t - v) \leqslant \Gamma_t$, where $\Gamma_t =$

$\gamma(u_0 - v) + \frac{1}{LC^2}\gamma^*(F'(u_{t-1})) + \cdots + \frac{1}{LC^2}\gamma^*(F'(u_0))$. We define $\Delta_t = (F(u_t) - F(v))_+$. From Eq. (10.10), we get:

$$\Delta_t \leq \big(\Delta_{t-1} - \tfrac{1}{2LC^2}\gamma^*(F'(u_{t-1}))^2\big)_+, \tag{10.12}$$

and from Eq. (10.11), we get $\Delta_t \leqslant \Gamma_t \gamma^*(F'(u_t))$. Thus,

$$\begin{aligned} \Delta_t \Gamma_t^{-2} &\leqslant \Delta_t \Gamma_{t-1}^{-2} \leqslant \big(\Delta_{t-1}\Gamma_{t-1}^{-2} - \tfrac{1}{2LC^2}\Gamma_{t-1}^{-2}\gamma^*(F'(u_{t-1}))^2\big)_+ \\ &\leqslant \big(\Delta_{t-1}\Gamma_{t-1}^{-2} - \tfrac{1}{2LC^2}(\Delta_{t-1}\Gamma_{t-1}^{-2})^2\big)_+. \end{aligned}$$

This leads to[4]

$$\Delta_t \Gamma_t^{-2} \leqslant \frac{1}{\frac{t}{2LC^2} + \Gamma_0^2 \Delta_0^{-1}} \leqslant \frac{2LC^2}{t}. \tag{10.13}$$

Moreover, by definition of Γ_t and using Eq. (10.11), we have:

$$\Gamma_t = \Gamma_{t-1} + \frac{1}{LC^2}\gamma^*(F'(u_{t-1})) \leqslant \Gamma_{t-1}\Big(1 + \frac{1}{LC^2}\frac{\gamma^*(F'(u_{t-1})^2}{\Delta_{t-1}}\Big).$$

Thus, by taking the product of the square of Eq. (10.12) and the previous inequality, we get:

$$\Gamma_t \Delta_t^2 \leqslant \Gamma_{t-1}\Delta_{t-1}^2 \Big(1 + \frac{1}{LC^2}\gamma^*(F'(u_{t-1}))^2\Big)\Big(1 - \frac{1}{2}\frac{1}{LC^2}\gamma^*(F'(u_{t-1}))^2\Big)_+^2.$$

Since $(1 - \varepsilon/2)_+^2(1+\varepsilon) \leqslant 1$ for all $\varepsilon \geqslant 0$, this leads to $\Gamma_t\Delta_t^2 \leqslant \Gamma_{t-1}\Delta_{t-1}^2$, and thus $\Gamma_t\Delta_t^2 \leqslant \Gamma_0\Delta_0^2$. Taking the product of the square of the inequality above with Eq. (10.13), this leads to

$$\Delta_t^5 = (\Gamma_t\Delta_t^2)^2 \cdot \Delta_t\Gamma_t^{-2} \leqslant (\Gamma_0\Delta_0^2)^2 \frac{2LC^2}{t},$$

which leads to the first result.

We can also bound the norm $\gamma(u_t)$ as follows, using Eq. (10.10):

$$\begin{aligned} \gamma(u_t) &\leqslant \gamma(u_0) + \frac{1}{LC^2}\sum_{i=1}^{t}\gamma^*(F'(u_{i-1})) \leqslant \gamma(u_0) + \frac{\sqrt{t}}{LC^2}\Big(\sum_{i=1}^{t}\gamma^*(F'(u_{i-1}))^2\Big)^{1/2} \\ &\leqslant \gamma(u_0) + \frac{\sqrt{t}}{LC^2}\Big(2LC^2[F(u_0) - F(u_t)]\Big)^{1/2}. \end{aligned}$$

■

We will need the flexibility of having an arbitrary $v \in \mathbb{R}^n$ in the statistical consistency proof, but when v is chosen as the minimizer u_* of F (then assumed to exist), we get a more traditional optimization bound:

$$F(u_t) - F(u_*) \leqslant \Big(\frac{2LC^2\gamma(u_0 - u_*)^2(F(u_0) - F(u_*))^4}{t}\Big)^{1/5} \leqslant \frac{LC^2\gamma(u_0 - u_*)^2}{t^{1/5}},$$

[4]We can use the following lemma, whose proof is left as an exercise: if (a_t) is a non-increasing non-negative sequence such that $a_t \leqslant (a_{t-1} - ca_{t-1}^2)_+$ for all $t \geqslant 1$, then $a_t \leqslant \frac{1}{t/c + 1/a_0}$ for all $t \geqslant 0$.

which can be compared to the bound for regular gradient descent (Prop. 5.5), which is $\frac{L}{2t}\|u_0 - u_*\|_2^2 \leqslant \frac{LC^2}{2t}\gamma(u_0 - u_*)$, with a better dependence on t, but with iterates that cannot be expressed as linear combinations of at most t iterates.

As done in the next section, assuming that $u_0 = 0$ and F is non-negative everywhere, this leads to:

$$\big(F(u_t) - F(v)\big)_+ \leqslant \Big(\frac{2LC^2\gamma(v)^2F(0)^4}{t}\Big)^{1/5} \quad \text{and} \quad \gamma(u_t) \leqslant \frac{\sqrt{2t}}{\sqrt{LC^2}}F(0)^{1/2}.$$

The proposition above shows that the gauge function γ controls the convergence of the gradient-boosting algorithm, in the same way that the Euclidean norm controls the convergence of gradient descent. For finite sets $\mathcal{W}$, where the gauge function is essentially an ℓ_1-norm in a reparameterization, the link with ℓ_1-norm penalization can be made explicit (see, e.g., Rosset et al., 2004, for details).

Exercise 10.6 *(♦) Show that when the function F is quadratic, then have the guarantee: $F(u_t) - F(u_*) \leqslant \frac{LC^2}{2t^{1/3}}\gamma(u_0 - u_*)^2$. Hint: replace Eq. (10.11) by $F(u_t) - F(u_*) = \frac{1}{2}F'(u_t)^\top(u_t - u_*)$.*

10.3.6 Convergence of expected risk

In order bound the expected risk, we need to relate empirical risk $\widehat{\mathcal{R}}$ and expected risk $\mathcal{R}$, for functions f with bounded penalty $\gamma_1(f)$. To study the generalization performance of constraining or penalizing by the variation norm defined above, we can naturally use the general framework of Rademacher complexities presented in Section 4.5.

Statistical performance through Rademacher complexities. The uniform deviations for the set of predictors $g : \mathcal{X} \to \mathbb{R}$ such that $\gamma_1(g) \leqslant D$ on i.i.d. data $x_1, \dots, x_n$ are controlled by the quantity

$$\mathbb{E}\Big[\sup_{\gamma_1(g)\leqslant D} \frac{1}{n}\sum_{i=1}^n \varepsilon_i g(x_i)\Big] = D \cdot \mathbb{E}\Big[\sup_{w\in\mathcal{W}} \frac{1}{n}\sum_{i=1}^n \varepsilon_i \varphi(x_i, w)\Big], \tag{10.14}$$

where the expectation is taken both with respect to the data $x_1, \dots, x_n$ and the independent Rademacher random variables $\varepsilon_1, \dots, \varepsilon_n \in \{-1, 1\}$.

In Section 9.2.3, we computed an upperbound proportional to $DR/\sqrt{n}$ for $\varphi(x, w)$ of the form $\sigma(x^\top w)$ (which corresponds to learning a one-hidden layer neural network), with an extra factor of $\sqrt{\log d}$ for ℓ_1-norm constraint on neural network weights, showing that although the set $\mathcal{W}$ is infinite, we can bound the uniform deviations. See another example in the exercise below. In the following, we will assume that

$$\mathbb{E}\Big[\sup_{\gamma_1(g)\leqslant D} \frac{1}{n}\sum_{i=1}^n \varepsilon_i g(x_i)\Big] \leqslant \frac{DR}{\sqrt{n}}\rho_\varphi, \tag{10.15}$$

for a universal constant $\rho_\varphi > 0$.

Exercise 10.7 *Given a metric space $\mathcal{X}$ with distance d and finite diameter, we consider $\varphi(w) = \sigma(d(x,w))$, for $w \in \mathcal{W} = \mathcal{X}$. Compute an upper-bound on the Rademacher complexity in Eq. (10.14).*

Generalization bound. We can now state our main statistical result about gradient boosting. To obtain such bounds, an additional norm (see, e.g., Lugosi and Vayatis, 2004) or cardinality (Barron et al., 2008) constraint is often added. In this section, we show how early stopping is enough to obtain rates of convergences for the gradient-boosting procedures defined in Section 10.3.5.

Proposition 10.2 *Assume the feature maps φ form a centrally symmetric set and that they are uniformly bounded by R, and satisfy Eq. (10.15). Assume the loss function ℓ is non-negative, G_2-smooth and G_1-Lipschitz-continuous with respect to the second variable, and that $\ell(y,0) \leqslant G_0$ almost surely. If g_t denotes the t-th iterate of the gradient boosting procedure, then, for any function $f : \mathcal{X} \to \mathbb{R}$,*

$$\mathbb{E}\big[\mathcal{R}(g_t)\big] \leqslant \mathcal{R}(f) + \Big[\sqrt{2t}\frac{G_0^{1/2}}{G_2^{1/2}} + R\gamma_1(f)\Big]\cdot 2G_1 \cdot \frac{\rho_\varphi}{\sqrt{n}} + \frac{(R\gamma_1(f))^{2/5}}{t^{1/5}}(2G_2G_0^4)^{1/5}. \quad (10.16)$$

Proof For any function f such that $\gamma_1(f)$ is finite, we have:

$$\begin{aligned}\mathcal{R}(g_t) - \mathcal{R}(f) &= \mathcal{R}(g_t) - \widehat{\mathcal{R}}(g_t) + \widehat{\mathcal{R}}(g_t) - \widehat{\mathcal{R}}(f) + \widehat{\mathcal{R}}(f) - \mathcal{R}(f) \\ &\leqslant \sup_{\gamma_1(g)\leqslant\gamma_1(g_t)} \big\{\mathcal{R}(g) - \widehat{\mathcal{R}}(g)\big\} + \sup_{\gamma_1(g)\leqslant\gamma_1(f)} \big\{\widehat{\mathcal{R}}(g) - \mathcal{R}(g)\big\} + \widehat{\mathcal{R}}(g_t) - \widehat{\mathcal{R}}(f).\end{aligned}$$

We then apply Proposition 10.1 with $C = R\sqrt{n}$ and $L = G_2/n$, with $F(0) \leqslant G_0$. This leads to $\gamma_1(g_t) \leqslant \frac{\sqrt{2t}}{\sqrt{G_2R^2}}G_0^{1/2}$, and $\widehat{\mathcal{R}}(g_t) - \widehat{\mathcal{R}}(f) \leqslant \big(\frac{2G_2R^2\gamma_1(f)^2G_0^4}{t}\big)^{1/5}$. Thus, using properties of Rademacher averages from Section 4.5,

$$\mathbb{E}\big[\mathcal{R}(g_t) - \mathcal{R}(f)\big] \leqslant \Big[\frac{\sqrt{2t}}{\sqrt{G_2R^2}}G_0^{1/2} + \gamma_1(f)\Big]\cdot 2G_1 \cdot \frac{\rho_\varphi R}{\sqrt{n}} + \Big(\frac{2G_2R^2\gamma_1(f)^2G_0^4}{t}\Big)^{1/5},$$

which leads to the desired result. ■

Up to constants, the bound in Eq. (10.16) is of the form $\mathcal{R}(f) + \frac{\sqrt{t}}{\sqrt{n}} + \frac{R\gamma_1(f)}{\sqrt{n}}\cdot\rho_\varphi + \frac{(R\gamma_1(f))^{2/5}}{t^{1/5}}$. We can optimize with respect to the number t of iterations, and if we take it of order $t \sim n^{5/7}(R\gamma_1(f))^{4/7}$, then this leads to $\mathcal{R}(f) + \frac{R\gamma_1(f)}{\sqrt{n}}\cdot\rho_\varphi + \big(\frac{R\gamma_1(f)}{\sqrt{n}}\big)^{2/7}$.

Assuming for simplicity that ρ_φ is a constant (like for neural networks), the dominant term is $\mathcal{R}(f) + \big(\frac{R\gamma_1(f)}{\sqrt{n}}\big)^{2/7}$. If the Bayes predictor f_* is such that $\gamma_1(f^*)$ is finite, we immediately get an excess risk that goes to zero as $\big(\frac{R\gamma_1(f_*)}{\sqrt{n}}\big)^{2/7}$. If the model we consider is misspecified, then, like in Section 7.5.1 for kernel methods, and Section 9.4 for neural networks, we could compute the resulting approximation error to obtain precise rates depending on properties of the Bayes predictor f^*.

Comparison with explicit constraint on γ_1. The bound above is obtained by early-stopping the boosting algorithms before they overfit. An alternative method is to minimize the empirical risk subject to the constraint $\gamma_1(f) \leqslant D$, which can be done with the Frank-Wolfe algorithm described in Section 9.3.6, with the same access to the weak-learner oracle and an optimization error proportional to R^2D^2/t after t iterations. Together with the estimation error in $\rho_\varphi \frac{RD}{\sqrt{n}}$, we can take $t = RDn^{1/2}$ steps of the Frank-Wolfe algorithms to get an excess risk less than $\mathcal{R}(f)$ plus a constant times $\frac{\rho_\varphi RD}{\sqrt{n}}$ for any f such that $\gamma_1(f) \leqslant D$. With the optimal choice of D, this leads to $\mathcal{R}(f)$ plus a constant times $\frac{\rho_\varphi R\gamma_1(f)}{\sqrt{n}}$, which is significantly better than for boosting. This, however, requires to set the constant D.

Comparison with early stopping for gradient descent. Compared to Section 5.3, where we used gradient descent, and the rates obtained for early stopping were the same as for constrained optimization, our analysis of boosting leads to consistent estimation, but with slightly worse rates.

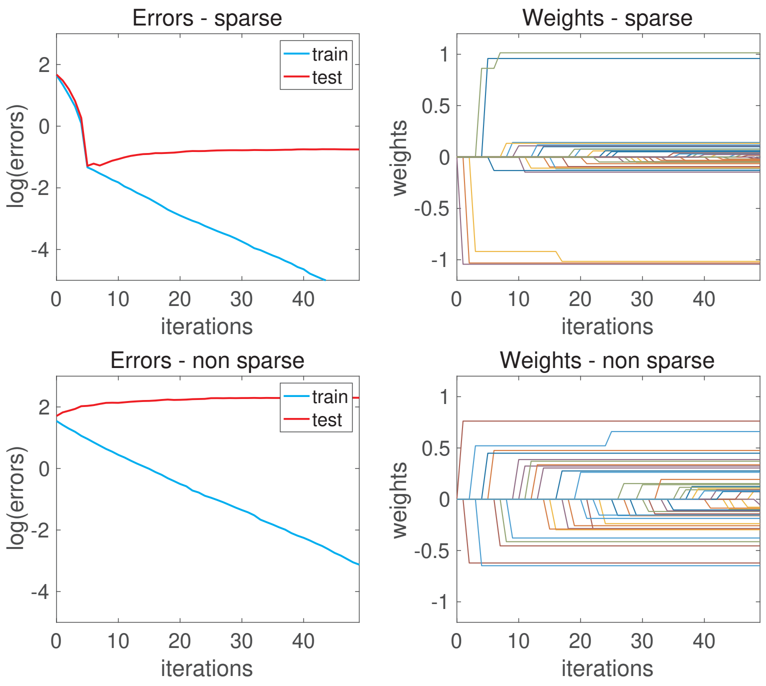

Figure 10.3: Matching pursuit algorithm on a problem with a sparse solution (top) and a non-sparse solution (bottom). Left: plots of training and testing errors. Right: plots of weights.

10.3.7 Experiments

In this section, we compare the boosting / conditional gradient algorithm described earlier on a simple linear regression task with feature selection. This corresponds to using $F(u) = \frac{1}{2n}\|y - u\|_2^2$, which is $(1/n)$-smooth and $(1/n)$-strongly convex, and $\gamma(u) = \inf_{w \in \mathbb{R}^d} \|w\|_1$ such that $u = \Phi w$.

We consider $n = 100$ observations in dimension $d = 1000$, sampled from a standard Gaussian random vector. A predictor β_* with $k = 5$ non-zero values in $\{-1, 1\}$ and data are generated from a linear model with Gaussian noise. We then compare the iterates of the boosting algorithm in terms of prediction errors (left plots) and variations of weights across iterations (right plots).

We also consider a rotation of the data so that this is no longer a sparse problem (bottom plot). We observe linear convergence of the training errors, as proved above, but with overfitting at convergence, strong for the non-sparse case and weak for the sparse case.

10.4 Conclusion

In this chapter, we have presented a brief overview of ensemble learning procedures, which rely on using the same "base" learning procedures on several datasets. Bagging procedures consider several often parallel and independent runs on randomly modified datasets, while boosting changes the weight on each observation sequentially. Moreover, boosting is an instance of computational regularization, where overfitting is avoided by early stopping an optimization algorithm that would converge to a minimizer of the empirical risks if not stopped. The implicit bias in boosting is that of an ℓ_1-norm; in Section 12.1, we analyze the implicit bias of gradient decent, when run to convergence, with a link to ℓ_2-penalties.