Lectures on Machine Learning (Fall 2017)

Hyeong In Choi Seoul National University

Lecture 8: Aggregation and randomization (Draft: version 0.9.1)

Topics to be covered:

- Aggregation by subsampling
  Parameter aggregation vs. value aggregation
- Aggregation
- Bagging
- Bootstrap

1 Introductory discussion

Aggregation and randomization are two big ideas going hand in hand in contemporary machine learning that are responsible for many big breakthroughs. By now they have become part of almost every top machine learning algorithms. They are especially prevalent in the so-called **ensemble learning** such as **bagging**, **random forests**, and all kinds of boosting like **AdaBoost** and **gradient boosting machine (GBM)**. We shall cover these algorithms in our subsequent lectures. As we shall see in our subsequent lectures, the idea of aggregation and randomization is also deeply embedded in contemporary deep learning.

1.1 Linear regression

Let us begin with a simple example. Figure 1 shows 50 data points gotten according to the method described in Section 2.2 of Lecture 1, i.e. according to $y = ax + b + \epsilon(x)$. The linear function $y = f(x) = ax + b$ is called the optimal regression function, which we call here the true function or the true line. In Figure 1 it is drawn as the dashed blue line. We also compute the linear regression line as described in Lecture 5 and draw it as the solid red line. We see they almost coincide with each other.

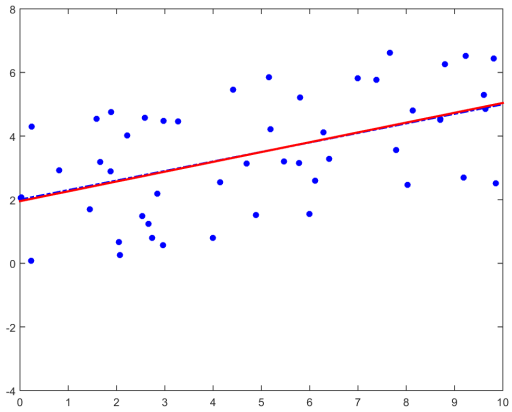

Figure 1: Data points with true line (dashed blue) and regression line (solid red)

To explain the idea of aggregation and randomization, forget about linear regression we learned in Lecture 5 and try a much simple-minded approach. In a nutshell, it goes as follows: (1) randomly select pairs of data points, (2) find line connecting each pair, and (3) average them. The result is shown in Figure 2, in which each of the cyan colored lines connects randomly chosen pairs of data points, and the dashed blue line is the line gotten by averaging. As we can see, the result of this simple-minded averaging is surprisingly good. Namely, it is very close to the regression line (solid red line) gotten by our standard linear regression method of Lecture 5.

The averaging procedure is as follows: let $P_i = \{(x_1^{(i)}, y_1^{(i)}), (x_2^{(i)}, y_2^{(i)})\}$ be M pair of points for $i = 1, \cdots .M$, and let $y = \alpha_i x + \beta_i$ be the line joining

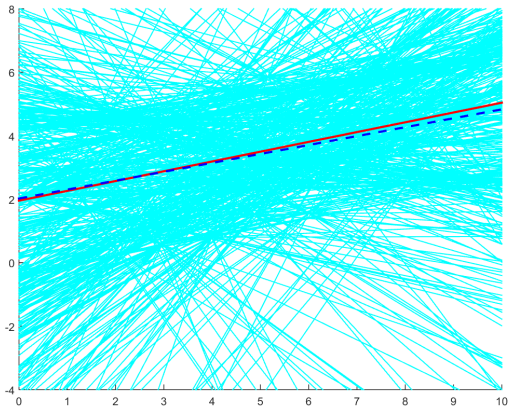

Figure 2: Linear regression line (solid red) and averaged line (dashed blue)

them. Then the averaged line is

$$y = \widehat{A}x + \widehat{B},$$

where

$$\widehat{A} = \frac{1}{M}\sum_{i=1}^{M}\alpha_i \tag{1}$$

$$\widehat{B} = \frac{1}{M}\sum_{i=1}^{M}\beta_i. \tag{2}$$

Let us see why this simple averaging works. Let (x_1, y_1) and (x_2, y_2) be two generic points. The equation of the line connecting them is

$$y = Ax + B,$$

where

$$A = \frac{y_2 - y_1}{x_2 - x_1}$$

$$B = y_1 - x_1\frac{y_2 - y_1}{x_2 - x_1}$$

are random variables due to the random nature of (x_1, y_1) and (x_2, y_2). Since $y = ax + b + \epsilon(x)$, we have

$$\begin{aligned} A &= \frac{a(x_2 - x_1) + \epsilon(x_2) - \epsilon(x_1)}{x_2 - x_1} \\ B &= ax_1 + b + \epsilon(x_1) - x_1 \frac{a(x_2 - x_1) + \epsilon(x_2) - \epsilon(x_1)}{x_2 - x_1}. \end{aligned}$$

Namely A and B can be viewed as random variables depending only on x_1 and x_2. Since $\mathrm{E}[\epsilon(x) \,|\, x] = 0$, we have

$$\begin{aligned} \mathrm{E}[A \,|\, x_1, x_2] &= a \\ \mathrm{E}[B \,|\, x_1, x_2] &= b. \end{aligned}$$

This shows why the averaging works. The averaging method so far described is called **parameter averaging**, because it actually averages parameters α_i and β_i. But there is another way. Suppose we have found lines $y = f_i(x) = \alpha x + \beta$. Define a new function $y = \widehat{\varphi}(x)$ by

$$\widehat{\varphi}(x) = \frac{1}{M} \sum_{i=1}^{M} f_i(x) = \frac{1}{M} \sum_{i=1}^{M} (\alpha_i x + \beta). \tag{3}$$

The resulting function $\widehat{\varphi}(x)$ is obtained by directly averaging the values, hence the name **value averaging**. Note, however, because of linearity, the value averaging and the parameter averaging turn out to be the same.

The idea of aggregation and randomization is pervasive in contemporary machine learning, and is widely applied. One should note that most of the good predictors cannot be described by simple formulas like linear regression. It thus limits the applicability of parameter averaging. Value averaging is more general in that it can be applied to anything since all we need is the output of the individual predictors. That is the reason why we are mostly concerned with value averaging in the lecture.

1.2 Nonlinear regression

To see how averaging works for a nonlinear problem, let us look at the case depicted in Figure 3. In there, 200 data points are drawn together with a true quadratic curve (dashed blue) $y = f(x) = ax^2 + bx + c$ and the quadratic

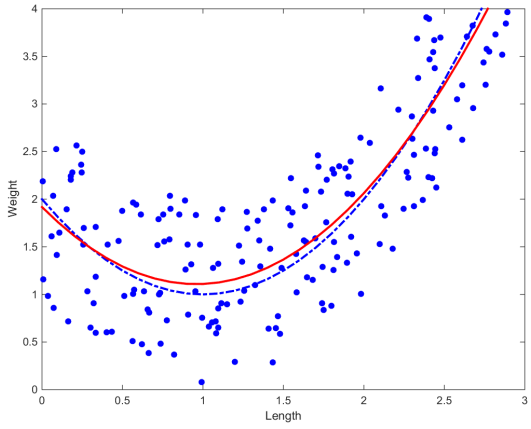

Figure 3: Data points with true curve (dashed blue) and regression curve (solid red)

regression function (solid red) gotten by the method of Lecture 5. As usual, the data point (x_i, y_i) is gotten from $y_i = f(x_i) + \epsilon(x_i)$.

Next, from the given 200 data points, we select a triplet of points $\{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$ 1000 times. After throwing away those triplets whose x co-ordinates are too close, we come up with 276 $(= M)$ triplets. For each of those M triplets $\{(x_1^{(i)}, y_1^{(i)}), (x_2^{(i)}, y_2^{(i)}), (x_3^{(i)}, y_3^{(i)})\}$, we construct a parabola $y = \alpha_i x^2 + \beta_i x + \gamma_i$. They are drawn in cyan color in Figures 4 and **??**.

The parameter averaged curve is

$$y = \widehat{A}x^2 + \widehat{B}x + \widehat{C},$$

where

$$\begin{aligned} \widehat{A} &= \frac{1}{M}\sum_{i=1}^{M} \alpha_i \\ \widehat{B} &= \frac{1}{M}\sum_{i=1}^{M} \beta_i. \\ \widehat{C} &= \frac{1}{M}\sum_{i=1}^{M} \gamma_i.. \end{aligned}$$

This parameter averaged curve is drawn as a dashed blue curve in Figure 4. Using the argument similar to those given in Section 1.1, it is not hard to

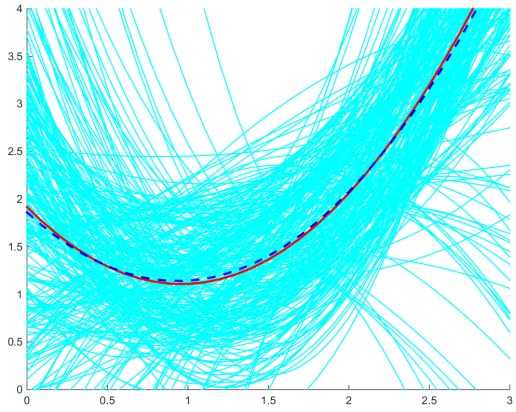

Figure 4: Linear regression curve (solid red) and parameter averaged curve (dashed blue)

define A, B, C and show that

$$\begin{aligned} \mathrm{E}[A \,|\, x_1, x_2, x_3] &= a \\ \mathrm{E}[B \,|\, x_1, x_2, x_3] &= b \\ \mathrm{E}[C \,|\, x_1, x_2, x_3] &= c. \end{aligned}$$

This again proves why the parameter averaging works in this case.

The value averaging proceeds the same way as given in Section 1.1. Since the formula is linear with respect to the coefficients, the value averaging and the parameter averaging also turn out to be the same.

2 Aggregation

As its name suggests, aggregation is a method of aggregating predictors over a family of datasets. Since this method is applicable to *any* predictors which may have no explicit representation of parameters, we only look at the *value aggregation*. To understand it, recall how we averaged regressors in Section 2.4 of Lecture 1. Namely, if $\varphi(x, \mathfrak{D})$ is a regressor constructed out of a dataset $\mathfrak{D}$, its average over the dataset $\mathfrak{D}$ is defined as $\overline{\varphi}(x) = \mathrm{E}_{\mathfrak{D}}\varphi(x, \mathfrak{D})$, where the dataset $\mathfrak{D}$ is treated as a random variable. Motivated by this example, we give the following definition. (Note that here we use $\varphi_a(x)$ instead of $\overline{\varphi}(x)$.)

Definition 1. *(Aggregation)*

(i) (Regression) Let $\varphi(x, \mathfrak{D})$ be a regression function. Then the aggregate regression function is defined as

$$\varphi_a(x) = E_{\mathfrak{D}}\varphi(x, \mathfrak{D}) = \int \varphi(x, \mathfrak{D}) dQ(\mathfrak{D}), \tag{4}$$

where $E_{\mathfrak{D}}$ is the expectation with respect to the random variable $\mathfrak{D}$ whose distribution on $\prod \mathfrak{X} \times \mathfrak{Y}$ is Q as described in Section 2.4 of Lecture 1.

(ii) (Classification) Let $\varphi(x, \mathfrak{D})$ be a classifier. Then the aggregate classifier is defined as

$$\varphi_a(x) = \operatorname*{argmax}_j P(\varphi(x, \mathfrak{D}) = j), \tag{5}$$

where $P(\varphi(x, \mathfrak{D}) = j) = Q(\{\mathfrak{D} : \varphi(x, \mathfrak{D}) = j\})$. In case of a tie, the tie is broken by some arbitrary rule.

2.1 Regression

The above definition of aggregate regressor is theoretical in nature. In practice, assume there are A datasets $\mathfrak{D}^{(1)}, \cdots, \mathfrak{D}^{(A)}$, which is supposed to be IID samples drawn from the distribution $Q = \prod P$ on $\mathfrak{Z} = \mathfrak{X} \times \mathfrak{Y}$. Then the aggregate regressor is defined by

$$\frac{1}{A} \sum_{k=1}^{A} \varphi(x, \mathfrak{D}^{(k)}).$$

Note that this is an unbiased estimator of $\varphi_a(x) = E_{\mathfrak{D}}[\varphi(x, \mathfrak{D})]$.

Let us see why the aggregation reduces the generalization error. For that, let us recall what we have done in Lecture 1. We let $\mathfrak{Z} = \mathfrak{X} \times \mathfrak{Y}$ and for $x \in \mathfrak{X}$, we let $\mathfrak{Z}_x = \{x\} \times \mathfrak{Y}$. The true function $f : \mathfrak{X} \to \mathfrak{Y}$ is defined by

$$f(x) = \int_{\mathfrak{Z}_x} y dP(y|x). \tag{6}$$

Note that the generalization error of $\varphi^{(\mathfrak{D})}$ is defined by

$$E|\varphi^{(\mathfrak{D})}(X) - Y|^2 = \int_{\mathfrak{X}} \int_{\mathfrak{Z}_x} |\varphi^{(\mathfrak{D})}(x) - y|^2 dP(y|x)\, dP(x).$$

This can be rewritten as

$$\begin{aligned} E|\varphi^{(\mathfrak{D})}(X) - Y|^2 &= E|\varphi^{(\mathfrak{D})}(X) - f(X) + f(X) - Y|^2 \\ &= E|\varphi^{(\mathfrak{D})}(X) - f(X)|^2 + 2E(\varphi^{(\mathfrak{D})}(X) - f(X)) \cdot (f(X) - Y) \\ &\quad + E|f(X) - Y|^2. \end{aligned}$$

Note

$$\begin{aligned} E(\varphi^{(\mathfrak{D})}(X) - f(X)) \cdot (f(X) - Y) &= \int\int (\varphi^{(\mathfrak{D})}(x) - f(x)) \cdot (f(x) - y)\, dP(y|x)\, dP(x) \\ &= \int_{\mathfrak{X}} (\varphi^{(\mathfrak{D})}(x) - f(x)) \cdot \left\{ \int_{\mathfrak{Z}_z} (f(x) - y)\, dP(y|x) \right\} dP(x) \\ &= 0, \end{aligned}$$

where the last equality is due to (6). Therefore, we have

$$E|\varphi^{(\mathfrak{D})}(X) - Y|^2 = E|\varphi^{(\mathfrak{D})}(X) - f(X)|^2 + E|f(X) - Y|^2. \tag{7}$$

Repeating the above argument almost verbatim, we also have

$$E|\varphi_a(X) - Y|^2 = E|\varphi_a(X) - f(X)|^2 + E|f(X) - Y|^2. \tag{8}$$

This is the generalization error of the aggregate function that we want to get a hold of. By Jensen's inequality, we have

$$\begin{aligned} E|\varphi_a(X) - f(X)|^2 &= E|E_{\mathfrak{D}}\varphi^{(\mathfrak{D})}(X) - f(X)|^2 \\ &\leq E E_{\mathfrak{D}}|\varphi^{(\mathfrak{D})}(X) - f(X)|^2. \qquad (9) \\ &= E_{\mathfrak{D}} E|\varphi^{(\mathfrak{D})}(X) - f(X)|^2. \qquad (10) \end{aligned}$$

This leads to the following important theorem

Theorem 1. *The generalization error $E|\varphi_a(X) - f(X)|^2$ of aggregate regressor is on the average less than the generalization error $E|\varphi^{(\mathfrak{D})}(X) - f(X)|^2$ of typical regressor $\varphi^{(\mathfrak{D})}(x)$ constructed out of data in the sense that*

$$E|\varphi_a(X) - f(X)|^2 \leq E_{\mathfrak{D}} E|\varphi^{(\mathfrak{D})}(X) - f(X)|^2.$$

This theorem says that $E|\varphi^{(\mathfrak{D})}(X) - f(X)|^2$ is on average greater than $E|\varphi_a(X) - f(X)|^2$, i.e., aggregation reduces overall L^2 error. This is the reason aggregation works better in general.

Let us analyze (9) further. First, recall that Theorem 2 of Lecture 1 says

$$E_{\mathfrak{D}}|\varphi^{(\mathfrak{D})}(x) - f(x)|^2 = \mathrm{Var}(x) + \mathrm{Bias}^2(x).$$

Thus

$$EE_{\mathfrak{D}}|\varphi^{(\mathfrak{D})}(X) - f(X)|^2 = E_x[\mathrm{Var}(x) + \mathrm{Bias}(x)] = \mathrm{Var} + \mathrm{Bias}^2. \tag{11}$$

Hence, by combining (8), (9) and (11), we have the following general bias-variance decomposition theorem.

Theorem 2. *(Bias-variance decomposition)*

$$E_{\mathfrak{D}}E|\varphi^{(\mathfrak{D})}(X) - f(X)|^2 = \mathrm{Var} + \mathrm{Bias}^2 + E|f(X) - Y|^2. \tag{12}$$

Note that Theorem 1 and Theorem 2 together give the upper bound of the generalization error of the aggregate regressor. As we have discussed in Lecture 1, the bias term measures how far the aggregate function φ_a is from the true function f, while the variance measures how much each individual regression function $\varphi^{(\mathfrak{D})}$ varies around its aggregate (mean) function φ_a.

As for the third term, note

$$E|f(X) - Y|^2 = \int_{\mathfrak{X}} \int_{\mathfrak{Z}_z} |f(x) - y|^2 \, dP(y|x) \, dP(x).$$

Obviously $\int_{\mathfrak{Z}_z} |f(x) - y|^2 \, dP(y|x)$ is the variance of the data y over $\mathfrak{Z}_z$, and $E|f(X) - Y|^2$ is its average over $\mathfrak{X}$. Therefore $E|f(X) - Y|^2$ is the error inherent in the probability structure of $P = P_{X,Y}$ that has nothing to do with how each individual regressor function $\varphi^{(\mathfrak{D})}$ is constructed.

2.2 Classification

Let $\varphi(x, \mathfrak{D})$ be a classifier. To begin, define

$$Q(j \,|\, x) = P(\varphi(x, \mathfrak{D}) = j). \tag{13}$$

In this definition $\mathfrak{D}$ is regarded as random, and $Q(j \,|\, x)$ measures the proportion of $\mathfrak{D}$'s that give output j given x, namely it represents the probability of predicting j given x by *generic* $\varphi(x, \mathfrak{D})$. Thus in view of the definition of Q, $Q(j \,|\, x)$ measures the size of the set $\{\mathfrak{D} : \varphi(x, \mathfrak{D}) = j\}$, namely,

$$Q(j \,|\, x) = Q\left(\{\mathfrak{D} : \varphi(x, \mathfrak{D}) = j\}\right).$$

Empirically, $Q(j \,|\, x)$ is estimated as follows. As before, let $\mathfrak{D}^{(1)}, \cdots, \mathfrak{D}^{(A)}$ be IID samples of datasets drawn from the distribution $Q = \prod P$ on $\prod \mathfrak{Z} = \prod \mathfrak{X} \times \mathfrak{Y}$. Then for each x,

$$\frac{1}{A} |\{k : \varphi(x, \mathfrak{D}^{(k)}) = j\}|$$

is an estimator of $Q(j \,|\, x)$.

Let us see now how the individual classifier $\varphi(x, \mathfrak{D})$ fares on average. First, let $P(j \,|\, x) = P(y = j \,|\, x)$ be the conditional probability outputting j, given input x, according to $P = P_{X,Y}$. (Of course it has nothing to do with φ; and for P, see Lecture 1.) Then

$$\sum_j Q(j \,|\, x) P(j \,|\, x)$$

is the probability of the prediction by *generic* $\varphi(x, \mathfrak{D})$ coinciding with that by P given x, and thus

$$r = \int \sum_j Q(j \,|\, x) P(j \,|\, x) \, dP(x)$$

is the overall probability of correct classification by *generic* $\varphi(x, \mathfrak{D})$. Since $\sum_j Q(j \,|\, x) = 1$,

$$\sum_j Q(j \,|\, x) P(j \,|\, x) \le \max_i P(i|x),$$

and the equality holds, if

$$Q(j \,|\, x) = \begin{cases} 1 & \text{if } P(j \,|\, x) = \max\limits_i P(i|x) \\ 0 & \text{else.} \end{cases}$$

Recall that in Lecture 1, we defined the Bayes predictor (classifier) by

$$\beta(x) = \max_j P(j \,|\, x),$$

Theorem 3 of Lecture 1 says that β has the highest correct classification rate, which is

$$r^* = \int \max_j P(j \,|\, x) \, dP(x).$$

Definition 2. *We say the predictor φ is order-correct for the input x if*

$$\operatorname*{argmax}_j Q(j \mid x) = \operatorname*{argmax}_j P(j \mid x).$$

Remark. *Order-correct predictor can be quite worse than the Bayes predictor. For example, let*

$$\begin{aligned} P(1|x) &= 0.9 \qquad P(2|x) = 0.1 \\ Q(1|x) &= 0.6 \qquad Q(2|x) = 0.4. \end{aligned}$$

Then φ is order-correct, but the correct classification rate by Q given x is seen to be

$$Q(1|x)P(1|x) + Q(2|x)P(2|x) = 0.58.$$

But for the Bayes classifier, the correct classification rate is 0.9.

Let us now see why the aggregate classifier reduces the classification error. First, recall the definition

$$\varphi_a(x) = \operatorname*{argmax}_j Q(j \mid x). \tag{14}$$

Empirically, it is constructed by the majority vote:

$$\varphi_a(x) = \operatorname*{argmax}_j |\{k : \varphi(x, \mathfrak{D}^{(k)}) = j\}|.$$

As usual the tie is broken by some arbitrary rule. Note that since the randomness concerning $\mathfrak{D}$ is aggregated out, the aggregated classifier is a deterministic function $\varphi_a(x) : \mathfrak{X} \to \mathfrak{Y}$.

For the aggregate predictor φ_a, the probability of correct prediction given x is obviously

$$\sum_j \mathbb{I}(\operatorname*{argmax}_i Q(i|x) = j) P(j \mid x),$$

and

$$r_A = \int \sum_j \mathbb{I}(\operatorname*{argmax}_i Q(i|x) = j) P(j \mid x)\, dP(x)$$

is its overall probability of correct classification.

Let C be the set of $x \in \mathfrak{X}$ at which the individual classifier $\varphi(x, \mathfrak{D})$ is order-correct and C' be its complement $\in \mathfrak{X}$. Then

$$\begin{aligned} r_A &= \int_C \sum_j \mathbb{I}(\operatorname*{argmax}_i Q(i|x) = j) P(j \mid x)\, dP(x) \\ &+ \int_{C'} \sum_j \mathbb{I}(\operatorname*{argmax}_i Q(i|x) = j) P(j \mid x)\, dP(x). \end{aligned}$$

Now on C,

$$\operatorname*{argmax}_i Q(i|x) = \operatorname*{argmax}_i P(i|x).$$

Therefore

$$\begin{aligned} r_A &= \int_C \max_j P(j \mid x) dP(x) \\ &+ \int_{C'} \sum_j \mathbb{I}(\operatorname*{argmax}_i Q(i|x) = j) P(j \mid x)\, dP(x). \end{aligned}$$

Suppose individual classifiers are reasonable in the sense that they are order-correct for most of inputs (i.e., C is big), then r_A is close to the optimal rate r^*. This way, as long as individual classifiers are order-correct over a big input set, the overall correct classification rate of the aggregate classifier is very high, even if the order-correctness is far from being optimal, as our example above shows.

3 Bagging

Bagging is a method, or rather, a precept of machine learning invented by Breiman. In a nutshell, it is based on the idea or belief that it is almost always better to *aggregate* things over multiple data sets. Bagging consists of three ingredients: aggregation, bootstrap and bagging itself. In this lecture, we follow closely the paper by Breiman [1].

The trouble with aggregation described above is that we have only one dataset in practice. So to apply aggregation, one somehow has to create multiple datasets out of the given one. One way is to take several subsamples as we did in Sections 1.1 and 1.2. Another is to use the so-called bootstrap resampling method, which is described in Section 4.1.

Bagging is in essence the same as aggregation except that the datasets $\mathfrak{D}^{(1)}, \cdots, \mathfrak{D}^{(B)}$ are created by the bootstrap resampling procedure. To be specific, let $\mathfrak{D} = \{z_i\}_{i=1}^N$ be a given dataset, where $z_i = (x_i, y_i) \in \mathfrak{X} \times \mathfrak{Y}$. From this, we create B bootstrap resamples $z^*(1), \cdots, z^*(B)$, where $z^*(k) = \{z_1^*(k), \cdots, z_N^*(k)\}$ for $k = 1, \cdots, B$. As usual, we identify $z^*(k)$ with the dataset $\mathfrak{D}^{(k)} = \{(x_i^{(k)}, y_i^{(k)})\}_{i=1}^N$.

We can then construct the bagging predictor φ_b (regressor or classifier) out of these datasets $\mathfrak{D}^{(1)}, \cdots, \mathfrak{D}^{(B)}$. In case of regression, φ_b is defined by

$$\varphi_b(x) = \frac{1}{B} \sum_{k=1}^{B} \varphi(x, \mathfrak{D}^{(k)})$$

for $x \in \mathfrak{X}$. In case of classification, the bagging classifier φ_b is defined by

$$\varphi_b(x) = \operatorname*{argmax}_j |\{k : \varphi(x, \mathfrak{D}^{(k)}) = j\}|$$

for $x \in \mathfrak{X}$. In case of a tie, the tie is broken, as usual, by some arbitrary rule.

How well this bagging works depends on how well the datasets are created. In particular, if the datasets $\mathfrak{D}^{(1)}, \cdots, \mathfrak{D}^{(B)}$ are less correlated to each other, the more beneficial the bagging becomes. In general, it is observed in many experiments that bagging reduces errors on the average.

Homework. *Suppose there are n sample points $z_1, \cdots, z_n$. We are to create bootstrap samples that consist of n points picked with replacement from the set $S = \{z_1, \cdots, z_n\}$.*

(1) Let N_n be the total number of distinct bootstrap samples. Show that N_n is asymptotic to $\dfrac{1}{\sqrt{\pi n}} 2^{2n-1}$ as $n \to \infty$. (Hint: Use String's formula.)

(2) In any bootstrap sample some points from S are picked up and some are not. Show that the limit of the average (expected) percentage of points in S that are not picked up in the bootstrap samples converges to $e^{-1} \approx 36.8\%$ as $n \to \infty$.

4 Mathematical supplements

4.1 Bootstrap: a brief introduction

Bootstrap is a statistical technique invented by Efron that is now being widely used in a whole variety of statistical applications [2]. What makes

it so appealing is that it is not predicated, at least in principle, on the full knowledge of the probability model of the population. It is in a nutshell a precept of creating new datasets from a given dataset by the method of resampling with replacement. Since it relies on repeated random resampling, it can also be viewed as belonging to the family of Monte-Carlo simulation techniques.

In this section, we introduce a rudiment of bootstrap and indicate why it works. An interested reader is advised to further consult the book by Efron and Tibshirani [2]. As the method of bootstrap itself is a purely statistical one that has nothing to do with machine learning, we present it, for the sake of simplicity, for the data $\mathfrak{D} = \{x_1, \cdots, x_n\}$, where x_i's are IID samples drawn from $\mathfrak{X}$ according to some probability distribution F. When it comes to bagging, we will revert back to $\mathfrak{D} = \{(x_i, y_i)\}_{i=1}^N$. But for now, $\mathfrak{D} = \{x_1, \cdots, x_n\}$ will do.

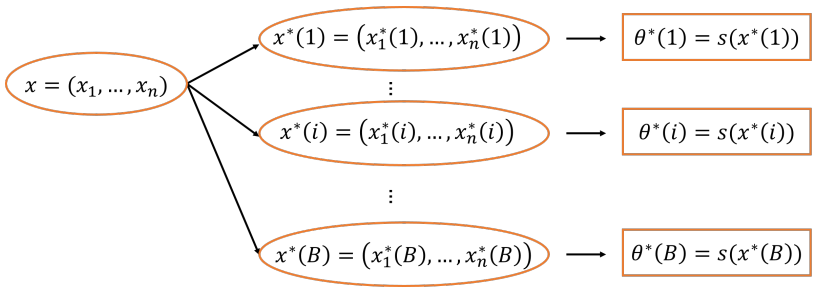

Figure 5: Bootstrap procedure

For the simplicity of notation, we interchangeably use $\mathfrak{D} = \{x_1, \cdots, x_n\}$ and $x = (x_1, \cdots, x_n)$ to denote the dataset. As depicted in Figure 5, $x_1, \cdots, x_n$ are IID samples coming from the distribution F. (Our early terminology for F is P.) It is generally assumed that F is not known. A statistic s is a function of $x = (x_1, \cdots, x_n)$ that is supposed to estimate a parameter θ, whose estimate is denoted by $\hat{\theta} = s(x)$. But if one goes one step further to seek to find out, say, the confidence interval, one need to know the nature of the distribution F. In traditional statistics, one assumes a certain probability structure (model) of F from which such questions can be answered. In the absence of such information, estimating the confidence interval is a very difficult task. But bootstrap provides a general framework to answer such questions also.

Let us now describe how bootstrap is done. First, as depicted in Figure

5, we are given a data $x = (x_1, \cdots, x_n)$. We then select n elements out of the set $\{x_1, \cdots, x_n\}$. But the selection is done with repetition so that some element may be selected more than once and some not at all. (In fact, the ratio of elements not so selected over the whole set is on the average roughly 37% for large n. See Homework above.) As in Figure 5, we call such set $x^*(1) = (x_1^*(1), \cdots, x_n^*(1))$. And then we repeat this procedure B times to create B sets of such data $x^*(1), \cdots, x^*(B)$. They are depicted in Figure 5. Each $x^*(i)$ is called a **bootstrap sample** or **bootstrap resample** (both are used interchangeably) and B is a predefined number that specifies how many bootstrap resampled datasets we want to create. Applying the statistic s, we get an estimator $\hat{\theta}^*(i) = s(x^*(i))$ for $i = 1, \cdots, B$.

These n statistics can be used to create the bootstrap estimate $\hat{\theta}^*$ of θ by the formula:

$$\hat{\theta}^* = \frac{1}{B} \sum_{i=1}^{B} \hat{\theta}^*(i).$$

One measure of how accurate this bootstrap estimate is the following bootstrap estimate $\widehat{se}_B$ of standard error se_F:

$$\widehat{se}_B = \sqrt{\frac{1}{B-1} \sum_{i=1}^{B} |\hat{\theta}^*(i) - \hat{\theta}^*|^2}.$$

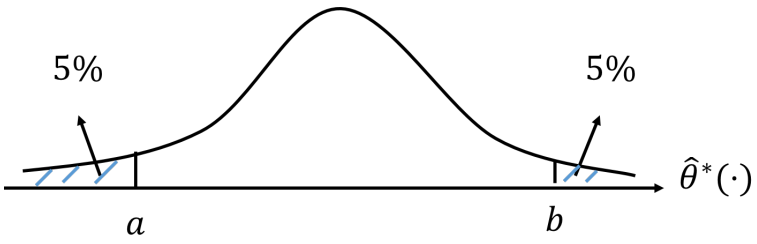

Figure 6: Histogram of $\hat{\theta}^*(\cdot)$

Bootstrap resampling can be used for many other purposes. For example, suppose we want to find the 90% confidence interval of the parameter θ. It is done as follows. First, draw the histogram of $\hat{\theta}^*(1), \cdots, \hat{\theta}^*(B)$ as in Figure 6. In there, a represents the lower threshold under which 5% of $\hat{\theta}^*(1), \cdots, \hat{\theta}^*(B)$ lie and b the upper threshold over which 5% of them lie. Then the interval $[a, b]$ is defined to be the 90% bootstrap confidence interval.

One should note that if one adheres to the usual method of statistics, one needs to know the probabilistic model of F to estimate the standard error and the confidence interval. But our basic tenet in machine learning is that we like to assume as little knowledge on F as possible. As the above methods attest, the estimates of bootstrap standard error and the bootstrap confidence interval can be done with virtually no knowledge on the nature of F. This illustrates the power and the flexibility of the bootstrap method.

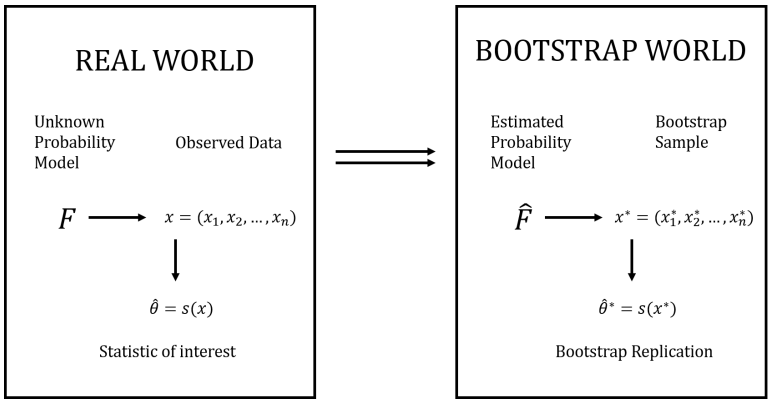

Figure 7: Real versus Bootstrap world

Let us look at the nature of the bootstrap resampling method from a slightly more theoretical angle. Figure 7 illustrates how bootstrap is done. The left panel is the usual sampling from the distribution F to get the sample $x = (x_1, \cdots, x_n)$. Let $\hat{F}$ be the empirical distribution made out of $x = (x_1, \cdots, x_n)$. Namely,

$$\hat{F} = \frac{1}{n} \sum_i \delta_{x_i}(x).$$

So for any $A \subset \mathfrak{X}$

$$\hat{F}(A) = \frac{1}{n} |\{i : x_i \in A\}|.$$

Thus $\hat{F}$ is a discrete probability (mass) distribution with point mass $1/n$ at each x_i. This discrete probability mass is counted with multiplicity. Thus for example, if $x_1 = x_2 = x'$, then $p(x') = 2/n$, and so on. This empirical distribution $\hat{F}$ is shown in the right panel of Figure 7. Then the bootstrap resample is nothing but the usual IID sample according to $\hat{F}$. The key point here is that unlike the original distribution F, which is not known, this empirical distribution $\hat{F}$ is completely known, once the sample $x = (x_1, \cdots, x_n)$

is given. So one can apply the usual Monte-Carlo simulation of repeated sampling. That is the essence of the bootstrap method described above.

There remains questions on how well this bootstrap method represents (approximates) the statistics of the original distribution F. In short, bootstrap works well for the usual estimators like the mean or median as long as the population distribution F has finite variance. But if the variance is infinite as in the case of power law distributions, the bootstrap method fails. It also fails if the data has correlations like the time series data. (For that, one can devise a trick called the block bootstrap.) There are many theoretical results concerning bootstrap and the reader is referred to many excellent books for more extensive discussions [2, 3]. But to illustrate the point, we quote a result by Bickel and Freedman [4].

Theorem 3. (Bickel and Freedman)
Assume that F has a finite variance σ^2. Let $x_1, \cdots, x_n$ be an IID sample drawn from a distribution F. Define the usual sample mean and variance by

$$\begin{aligned} \mu_n &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ s_n^2 &= \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_n)^2. \end{aligned}$$

Let $x_1^*, \cdots, x_m^*$ be a bootstrap resample $\sim \hat{F}$. (Note that in here m may differ from n.) Define the mean and the variance of the resample by

$$\begin{aligned} \mu_m^* &= \frac{1}{m}\sum_{i=1}^{m} x_i^* \\ (s_m^*)^2 &= \frac{1}{m}\sum_{i=1}^{m}(x_i^* - \mu_m^*)^2. \end{aligned}$$

Then,

(a) $\sqrt{m}(\mu_m^* - \mu_n)$ weakly converges to $N(0, \sigma^2)$, as $n, m \to \infty$;

(b) $P(|s_m^* - \sigma| > \epsilon)$ almost surely converges to 0, as $m \to \infty$.

References

[1] Breiman, L., *Bagging Predictors*, Machine Learning 24, 123-140 (1996)

[2] Efron, B. and Tibshirani, R.J., *An Introduction to the Bootstrap*, Chapman & Hall/CRC (1998)

[3] Shao, J. and Tu, D., *The Jackknife and Bootstrap*, Springer-Verlag (1995)

[4] Bickel, P.J. and Freedman, D., *Some Asymptotic Theory for Bootstrap*, The Ann. Stat. 9, 1196-1217 (1981)