Lectures on Machine Learning (Fall 2017)

Hyeong In Choi Seoul National University

Lecture 1: Overview of supervised learning (Draft: version 0.9.3)

Topics to be covered:

- Basic terminology
- Probabilistic model of data
- Overview of regression problem
- Overview of classification problem

In this lecture we present a very brief overview of supervised learning and some related topics, In particular, we explain how one views the data in machine learning and present a prevalent probabilistic model for it. This leads into many important issues like error, bias, and variance which are so pervasive in the whole of machine learning. We have found it is beneficial to expose the readers to these general ideas as early as possible. They are also intimately related to other important concepts like overfitting, underfitting, model selection, and so on, which will be covered in subsequent lectures.

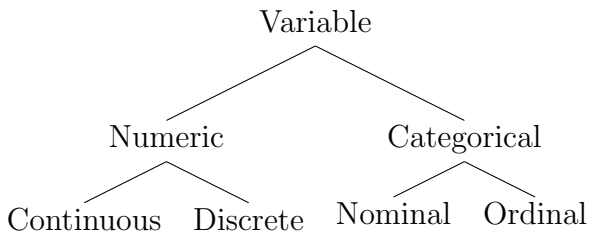
1 Bare bones of supervised learning

1.1 Basic terminology

Suppose we want to estimate the price of a housing unit, such as a condominium (apartment) or single-family house, in a certain city. The first factor

that comes in mind is the size of the unit, say, the square meters of the living space. However, other factors may also matter, such as the location of the unit in the city and the distance to subway stations. And of course, the type of housing, whether it is a single-family house or a unit in a multi-unit dwelling, must be a very important factor as well. There are many other factors we can think of that may influence the price. All these factors are generally called **features** or **input variables**. They also go by various other names like **attributes** or **covariates**. Some even use the term predictor, but it is a bad practice because it may be confused with the prediction function that is an outcome of supervised learning. Hence this term should be avoided and will not be used in this course. The quantity one wants to estimate, say, the price in this case, is called the **output**, **output variable**, or **response variable**.

Input and output variables together are generically called variables. Variables used in machine learning are of diverse types. It can be a **numeric** variable representing numerical quantities like distance or area, or it can be a **categorical** variable representing things or concepts. For instance, the type of housing is a categorical variable. The location in a city should also be considered a categorical variable. Numeric variables can be either **continuous** or **discrete**. When an output variable is categorical, its value is called **class** or **label**. Some categorical variables may have ordering property. For example, in bond credit rating, AAA bonds are considered more credit-worthy than AA bonds and so on. Such kind of variable is called an **ordinal** variable. In summary, variables are classified as below:



Nominal variables are purely categorical variables with no ordinal relation. However, in practice, unless specified otherwise, "categorical" usually means "nominal" and vice versa. So people use "nominal" and "categorical" interchangeably, and we do too. In machine learning, a categorical variable is assumed to have a finite number of values. In this sense it can be regarded as discrete. However, when we say "discrete", we usually mean numeric vari-

ables with discrete values. Of course, an ordinal variable can be continuous, but when we say "ordinal", we always mean it to have a finite number of values. The numeric variable type can be further enlarged to include the ones with values in intervals or even sets, but the above classification is good enough for our present purpose.

The goal of supervised learning is to find a way of expressing the response variable y as a function of input variables $x_1, \cdots, x_d$,. Namely, its objective is to find a function of the form

$$y = \varphi(x_1, \cdots, x_d).$$

This φ is called a **prediction function** or simply a **predictor**. It may be explicitly described by a formula or it may be computable only by a procedure (computer programming). For the input, one usually uses the vector notation to denote

$$x = (x_1, \cdots, x_d),$$

and so φ is written as

$$y = \varphi(x).$$

The set from which input data is selected is called the **input domain** or **input space**, which is denoted by $\mathfrak{X}$; the set of all possible outputs is called the **output domain** or **output space**, which is denoted by $\mathfrak{Y}$. Using this notation, predictor is a function

$$\varphi : \mathfrak{X} \to \mathfrak{Y}.$$

So after everything is said and done, a supervised learning problem is nothing but finding such a function that satisfies the suitable criteria.

If output is continuous, say, $\mathfrak{Y} = \mathbb{R}^m$, a predictor is called a **regression function**, or simply a **regressor**; if output is categorical, it is called a **classifier**. For ordinal output, one usually devises special methods that go by the name of **ordinal regression** of one sort or another.

To find a predictor φ, supervised learning requires data, say, $\mathfrak{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, which is a set of N input-output pairs $(x^{(i)}, y^{(i)})$, for $i = 1, \cdots, N$, where

$$x^{(i)} = (x_1^{(i)}, \cdots x_d^{(i)})$$

is the i-th input data consisting of d features and $y^{(i)}$ is the i-th output value.

Throughout this lecture, **data point** means individual data element $(x^{(i)}, y^{(i)})$, $x^{(i)}$, or $y^{(i)}$, and we also frequently drop the word "point" and

simply say **data**. The parenthesized superscript always denotes the data point number. So, for example, $(x^{(i)}, y^{(i)})$ is the i-th data, and so on.

The predictor we want to find is $\varphi : \mathfrak{X} \to \mathfrak{Y}$ such that the output $\varphi(x^{(i)})$ of the input $x^{(i)}$ is as close to $y^{(i)}$ as possible for $i = 1, \cdots, N$. This "closeness" assertion requires a definition of error, and depending on how the error function is defined, one has many different supervised learning models. We will encounter many different error functions throughout this course.

When we find a predictor, we have to restrict the set of possible candidate functions, for otherwise there will be too many to choose from, which is too complex a problem for any reasonable algorithm to handle. Such a restricted class of functions is called **hypothesis space** and it is denoted by $\mathcal{H}$ or sometimes by $\mathcal{F}$. So a machine learning problem presupposes a hypothesis space

$$\mathcal{H} = \{\varphi \,|\, \varphi : \mathfrak{X} \to \mathfrak{Y}\},$$

and finds a predictor in $\mathcal{H}$ that fits the data as *best* as possible. The process of finding such predictor is generally called **learning** or **training**. These two terms are used interchangeably, although learning can be further elaborated on to include the process of validation and testing.

The ultimate utility of a predictor is not to fit a given dataset at the time of training but to continue to perform satisfactorily in the future under the assumption that data arising in the future will be of the *same nature*. To clarify what we mean by "same nature", we need the following probabilistic model of data.

1.2 Probabilistic model of data

We postulate there is random variable Z with values in $\mathfrak{X} \times \mathfrak{Y}$ whose probability (density or mass) is denoted by $P_Z(x, y)$. If clear in context, we omit the subscript Z and write it as $P(x, y)$. It is to be noted that such a probability is assumed to be there in the background, but in general we do not attempt to know (estimate) it, unless a specific machine learning algorithm calls for explicit modeling of $P(x, y)$. (The naive Bayes model in Lecture 2 is one such case.) This is in contrast with statistics, which as a matter of principle almost always endeavors to know more about $P(x, y)$ itself, directly or indirectly.

By marginalizing the variable y y in $P(x, y)$, we get a probability on $\mathfrak{X}$, which we denote by $P_X(x)$. This way, an $\mathfrak{X}$-valued random variable X is

defined. When there is no danger of confusion, we drop the subscript X and write it as $P(x)$. Similarly, we define $P_Y(y) = P(y)$ and Y. In other words, we can now view $P(x, y)$ as a joint probability distribution of X and Y. We also write $Z = (X, Y)$. In this case, $P(x, y)$ can be also viewed as a distribution of Z.

Using this probabilistic setup, we can now clarify the assumption on how the dataset $\mathfrak{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$ comes about. Simply put, the dataset $\mathfrak{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$ is a collection of independent identically distributed (IID) samples drawn from $\mathfrak{X} \times \mathfrak{Y}$ according to the probability distribution $P(x, y)$.

In academic analysis of machine learning algorithms, one frequently makes this IID assumption, for otherwise one cannot go very far. In fact, in many cases it is a reasonable approximation of reality, Sometimes, however, this IID assumption paints a distorted picture of reality, as is the case in sequentially dependent data like time series. For this kind of data one must take special care to de-correlate the sequential dependency.

Recall the following **chain rule of probability**.

$$P(x, y) = P(y \,|\, x)P(x). \tag{1}$$

This gives rise to an equivalent but more intuitive way of looking at sampling. Suppose (x, y) is a generic data point sampled according to $P(x, y)$. The chain rule of probability (1) implies that it is equivalent to sampling x first according to $P(x)$, and then, fixing x, sampling y according to $P(y|x)$. As we shall see below, this way of looking at data meshes very well with regression analysis.

Remark. *When we talk about probability or probability distribution, we are intentionally vague about its nature here and for the rest of this course. In fact, for continuous random variable, it is probability density function; for discrete or categorical random variable, it is probability mass function; which is which has to be understood by context.*

Similarly, we almost always use the integral notation. When it is a probability mass function, however, it should have been sum. But then, integral can be made to include summation by the use of Dirac measure. Since these fine distinctions can be easily understood in context, we use integral notation as a generic means of communication to avoid complicating notations unnecessarily.

2 Regression problem

The problem of finding the predictor when output variable is continuous is called a regression problem. In most cases, the output space is assumed to be a Euclidean space $\mathbb{R}^m$,; and in many instances, we assume $m = 1$, i.e. the scalar output. When there are constraints for output variables, the output is a subset of $\mathbb{R}^m$, typically a manifold. However, in here, we restrict ourselves to Euclidean space.

2.1 Errors of regressors

Let $\varphi : \mathfrak{X} \to R^m$ be a regression function. There are two kinds of errors relating to φ. The first is the so-called mean square error defined as:

$$\mathrm{MSE}(\varphi, \mathfrak{D}) = \frac{1}{N} \sum_{i=1}^{N} \left| \varphi(x^{(i)}) - y^{(i)} \right|^2, \tag{2}$$

which is the error of φ with respect to the dataset $\mathfrak{D}$. This kind of error is generally called **empirical error**, **in-sample error**, or **empirical risk**. It is an error only specific to the particular dataset $\mathfrak{D}$. If the dataset is changed, this kind of error will also change.

The intrinsic, or theoretical, error of φ with respect to the background probability $P(x, y)$ is defined by

$$\mathrm{R}(\varphi) = \mathrm{E} \left| \varphi(X) - Y \right|^2 = \int \left| \varphi(x) - y \right|^2 P(x, y) dx dy, \tag{3}$$

which is called **generalization error**, **out-of-sample error**, or simply **risk**. As we said above, most machine learning problems, if not all, are not concerned with finding out the probability distribution $P(x, y)$. So in this sense it is mostly a theoretical device. Nonetheless, it is an important quantity, as it represents possible error for *any future dataset* that comes from the same probability distribution $P(x, y)$. Note also that the mean square error (2) is an unbiased estimator of $R(f)$.

2.2 Optimal regression function

For the regression problem, define the deterministic function $f_P(x)$ by

$$f_P(x) = \mathrm{E}[Y \mid X = x] = \int y P(y|x) dy, \tag{4}$$

which we call the **optimal regression function** or the **true function**. When there is no danger of confusion, we drop the subscript P and use $f(x)$ instead of $f_P(x)$. It should also be noted that this optimal regression function may or may not belong in the given hypothesis space $\mathcal{H}$ of a machine learning problem at hand.

It is called *optimal* in the sense that had we known $P(x, y)$, $f_P(x)$ would have been the *best* regression function as the following theorem shows.

Theorem 1. *$f_P(x)$ is the best predictor in the sense that its generalization error is smallest among all possible predictors. Namely, for any predictor $\varphi(x)$,*

$$R(f_P) = \mathrm{E}\,|f_P(X) - Y|^2 \leq \mathrm{E}\,|\varphi(X) - Y|^2 = R(\varphi).$$

And $R(\varphi) = R(f_P)$ if and only if $\varphi = f_P$.

Proof. For simplicity of notation, we use $f(x)$ instead of $f_P(x)$. Then

$$\begin{aligned}
& \mathrm{E}\,|\varphi(X) - Y|^2 - \mathrm{E}\,|f(X) - Y|^2 \\
= & \int |\varphi(x) - y|^2 P(x, y)dxdy - \int |f(x) - y|^2 P(x, y)dxdy \\
= & \int |\varphi(x) - y|^2 P(y \,|\, x)P(x)dxdy - \int |f(x) - y|^2 P(y \,|\, x)P(x)dxdy \\
= & \int |\varphi(x)|^2 P(x)dx - 2\int \varphi(x) \cdot f(x)P(x)dx - \int |f(x)|^2 P(x)dx + 2\int |f(x)|^2 P(x)dx \\
= & \int |\varphi(x) - f(x)|^2 P(x)dx \geq 0.
\end{aligned}$$

Here, we used the fact that $\int P(y \,|\, x)dy = 1$ and the property (4) to go from the third line to the fourth. □

Next, define the **error random variable** $\epsilon(x)$ at x, i.e. a random variable with value in $\{x\} \times \mathcal{Y}$ by

$$\epsilon(x) = Z|_x - f(x), \tag{5}$$

where $Z|_x$ is a random variable gotten out of the conditional probability distribution $P(y \,|\, x)$. Note $\mathrm{E}[\epsilon(x)] = 0$ and we assume $\sigma(x)^2 = \mathrm{E}[\epsilon(x)^2] < \infty$.

Recall the alternative view on sampling explained at the end of the previous section. Namely, a generic sample point (x, y) is gotten by first sampling x according to $P(x)$, and then, fixing x, sampling y according to $P(y \,|\, x)$.

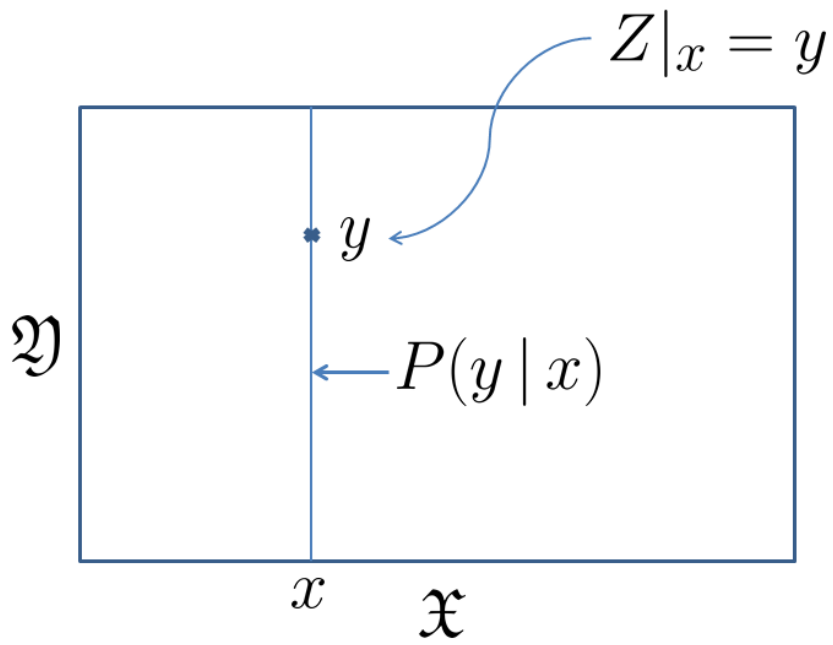


Figure 1: $P(y \mid x)$ and $Z|_x$

The last statement is equivalent to saying that y is the value of the random variable $Z|_x$. This situation is depicted in Figure 1.

Thus using (5), any sample point (x, y) can be envisioned to have been gotten by

$$y = f(x) + \epsilon(x), \tag{6}$$

i.e. y is a random perturbation by $\epsilon(x)$ of $f(x)$.

Using this, we may envision the dataset $\mathfrak{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ as gotten first by sampling $x^{(i)}$ and then by setting $y^{(i)}$ as

$$y^{(i)} = f(x^{(i)}) + \epsilon(x^{(i)}),$$

for $i = 1, \cdots, N$. This viewpoint is frequently exploited when we discuss error of regression problems.

2.3 Empirical error minimization

Although the optimal regression function $f_P(x)$ introduced above is the best, it is not realistic unless we know the probability distribution $P(x, y)$.

A typical way machine learning deals with this problem is to find a function (regressor) that minimizes the mean square error (2). Or in practice, find a function whose mean square error is as close to the minimum as possible. This kind of approach is generally called the **empirical error minimization** or **empirical risk minimization**. (There are lots of variants of this, and in due course of time, we will deal with many of them.)

However, without restricting the candidates for empirical error minimization, the problem may get unwieldy. So to make it a manageable problem, first set up the hypothesis space

$$\mathcal{H} = \{\varphi \,|\, \varphi : \mathfrak{X} \to \mathfrak{Y}\},$$

which has a beneficial effect of confining the search for the final predictor to a well defined set of candidates.

Now let $\varphi(x) = \varphi(x, \mathfrak{D}) \in \mathcal{H}$ be a regressor found by applying the empirical error minimization method to the dataset $\mathfrak{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$. (We will study many different models and methods of finding such a regressor in the subsequent lectures of this course.) This regressor has to have not just small empirical error but also small enough generalization error as well so that it can work as accurately for all possible future datasets of the same nature. This is one of the central issues of machine learning. To understand it, we present the bias and variance problem below. There are also several other related important issues. For more details, the reader is referred to Lecture 6: Model selection.

2.4 Bias and variance

To gain perspective on what is involved, let us look at the following dataset which is created as a random perturbation of a function $f(x) = x^3 - 3x^2 + 2x$. It goes as follows: first sample $N(= 30)$ points uniformly in the interval $[0, 2]$ and form the x-part $\{x^{(i)}\}_{i=1}^N$ of the dataset; second, repeatedly run the Gaussian random variable ϵ with mean 0 and $\sigma \approx 0.26$ and set $y^{(i)} = f(x^{(i)}) + \epsilon$ for $i = 1, \cdots, N$. The resulting dataset and $f(x)$ is depicted in Figure 2. Note this way of a creating dataset is in line with the description given at the end of the previous section.

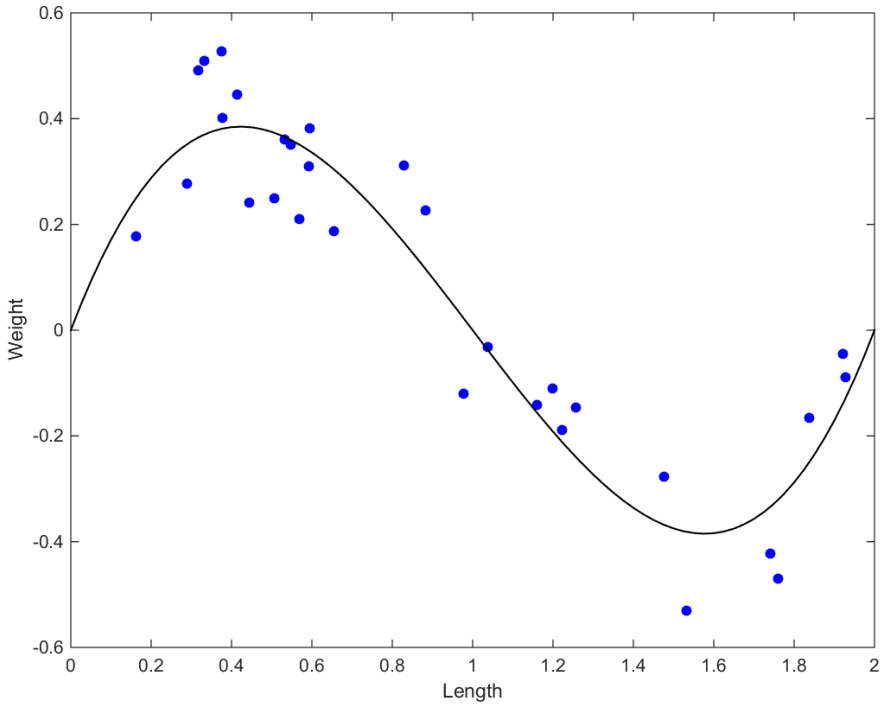


Figure 2: The dataset with true $f(x)$

Since this way of creating a dataset is itself a random process, the dependence of $\varphi(x, \mathfrak{D})$ on $\mathfrak{D}$ is also random. Namely, for each $x \in \mathfrak{X}$ we may view $\varphi(x, \mathfrak{D})$ as a random variable with respect to $\mathfrak{D}$. (See Remark below on how to make this concept mathematically palatable.)

To illustrate the random nature of datasets, we repeat the above process multiple times to get, say, $\mathfrak{D}_1, \cdots, \mathfrak{D}_5$, and find the cubic regression functions $\varphi(x, \mathfrak{D}_1) \cdots, \varphi(x, \mathfrak{D}_5)$, which, together with the optimal regression function $f(x)$, are depicted in Figure 3. This of course is not a realistic picture, as we have only one dataset in practice, not multiple ones. (Generating multiple datasets out of one can be done, but we have to wait until we come to the bootstrapping method.)

A natural question to ask now is how close is $\varphi(x, \mathfrak{D})$ to $f(x)$ on average. The following theorem answers that question.

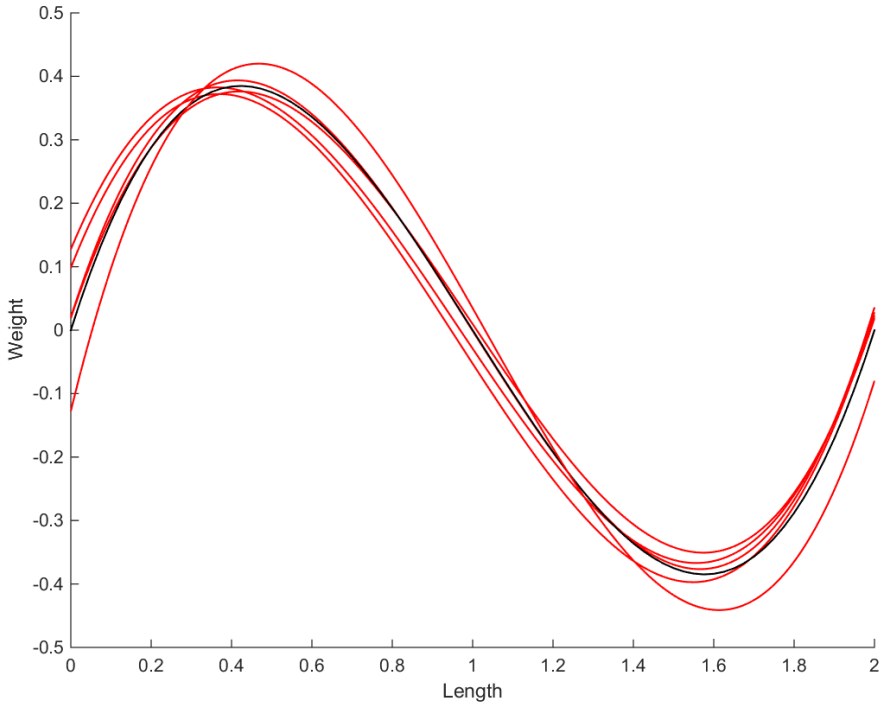


Figure 3: $\varphi(x, \mathfrak{D}_1) \cdots, \varphi(x, \mathfrak{D}_5)$ together with $f(x)$

Theorem 2. *(Bias-variance decomposition) Define* $\overline{\varphi}(x) = \mathrm{E}_{\mathfrak{D}}\varphi(x, \mathfrak{D})$. *Then*

$$\mathrm{E}_{\mathfrak{D}} \left|\varphi(x, \mathfrak{D}) - f(x)\right|^2 = \mathrm{Var}(x) + \mathrm{Bias}^2(x),$$

where the bias *is defined by*

$$\mathrm{Bias}(x) = \left|\overline{\varphi}(x) - f(x)\right|,$$

and the variance *by*

$$\mathrm{Var}(x) = \mathrm{E}_{\mathfrak{D}} \left[\varphi(x, \mathfrak{D}) - \overline{\varphi}(x)\right]^2.$$

Proof.

$$\begin{aligned} & \mathrm{E}_{\mathfrak{D}} \left|\varphi(x,\mathfrak{D}) - f(x)\right|^2 \\ = \; & \mathrm{E}_{\mathfrak{D}} \left|\varphi(x,\mathfrak{D})\right|^2 - 2f(x)\mathrm{E}_{\mathfrak{D}}\varphi(x,\mathfrak{D}) + f(x)^2 \\ = \; & \mathrm{Var}(x) + \left|\mathrm{E}_{\mathfrak{D}}\varphi(x,\mathfrak{D})\right|^2 - 2f(x)\cdot \mathrm{E}_{\mathfrak{D}}\varphi(x,\mathfrak{D}) + \left|f(x)\right|^2 \\ = \; & \mathrm{Var}(x) + \left|\mathrm{E}_{\mathfrak{D}}\varphi(x,\mathfrak{D}) - f(x)\right|^2 \\ = \; & \mathrm{Var}(x) + \mathrm{Bias}^2(x) \end{aligned}$$

□

For more general bias-variance decomposition theorem, see Lecture 8. Note that in Theorem 2, $\overline{\varphi}$ is a typical regressor one can expect to get from a *generic* dataset. $\mathrm{Bias}(x)$ measures how far this regressor is from the optimal regressor at x, and $\mathrm{Var}(x)$ measures how much variance will occur in finding such typical regressor $\overline{\varphi}$. Now one must not forget that regressor $\varphi(x,\mathfrak{D})$ is found among those in the hypothesis space $\mathcal{H}$. So the choice of $\mathcal{H}$ affects the computation of $\varphi(x,\mathfrak{D})$ and hence $\overline{\varphi}$.

If $\mathcal{H}$ is very small, say, a singleton set, then $\varphi(x,\mathfrak{D})$ and $\overline{\varphi}$ must be identical for any dataset $\mathfrak{D}$, which means the variance is zero. On the other hand, in this case, the bias must be in general very big, unless we happen to have chosen the optimal regressor f as that single element. On the other hand, if $\mathcal{H}$ is very big, there is a better chance for $\varphi(x,\mathfrak{D})$ to be close to the optimal regressor f, which means the bias will tend to be small, while the variance may get larger.

Model complexity	Bias	Variance
low	high	low
high	low	high

Table 1: Bias-variance trade-off

In here, model complexity simply means how big the hypothesis set $\mathcal{H}$ is. We say $\mathcal{H}_1$ is more complex than $\mathcal{H}_2$ if $\mathcal{H}_2$ is a subset of $\mathcal{H}_1$.

Note also that Theorem 2 says that on average the sum of (the square of) the bias and variance are controlled by $\mathrm{E}_{\mathfrak{D}} \left|\varphi(x,\mathfrak{D}) - f(x)\right|^2$ which is inherent in the setup of the model. This is a typical phenomenon called the **bias-variance trade-off**, which can be summed up in Table 1. This

bias-variance trade-off has been held as a guiding principle for a long time in machine learning in the era when data was scarce. But with the advent of big data and especially with deep learning, this long-held belief is now being looked at from a different viewpoint. Recently, there has been a flood of reports attesting that with bigger data, the accuracy of any predictor increases no matter what it is. The reader may consult the paper by Banko and Brill as one such example [1].

Remark. *In Theorem 2, the dataset $\mathfrak{D}$ is treated as a random variable when we take the expectation. However, to do so, we need a probability measure with which to take the expectation. Here is a sketch of how one can do it.*

First, let $\mathfrak{Z} = \mathfrak{X} \times \mathfrak{Y}$. Then any dataset $\mathfrak{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ can be thought of as a finite sequence of elements of $\mathfrak{Z}$, although its length may vary as N can vary. So the set of all datasets can be identified with a set of finite sequence of elements of $\mathfrak{Z}$ which is a subset of the set $\mathfrak{S}$ of infinite sequence of elements of $\mathfrak{Z}$.

The essential point of the celebrated Kolomogorov extension theorem is that the probability measure $P(x, y)$ on $\mathfrak{Z}$ can be extended to $\mathfrak{S}$. Its rigorous proof can be found in [3] and [4].

In fact, Theorem 2 is only a theoretical device that does not require any concrete calculation of expectation with respect to $\mathfrak{D}$ in machine learning. Thus this mathematical construction may be of theoretical interest only.

3 Classification problem

If output or response variable is categorical, the problem of finding the predictor is called a classification problem. Suppose there are K elements in $\mathfrak{Y}$, we usually write $\mathfrak{Y} = \{1, \cdots, K\}$. These numbers, however, are just symbolic representations of output labels or classes that have nothing to do with actual numbers, nor do they represent any order or ordinal relation between the labels. This standard convention should be understood throughout this course.

3.1 Errors of classifiers

Let $h : \mathfrak{X} \to \mathfrak{Y}$ be a classifier and let $\mathfrak{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ be a dataset. As is the case for regressors, there are two kinds of errors relating to classifiers.

The first is the **empirical error**, **in-sample error** or **empirical risk** which is defined as

$$E_{emp}(h) = E_{emp}(h, \mathfrak{D}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(h(x^{(i)}) \neq y^{(i)}). \tag{7}$$

This error counts mislabeled outcomes and reports their ratio over the total number of data. This count is done with respect to a particular dataset $\mathfrak{D}$; if the dataset is changed, this error will also change.

The **generalization error**, **out-of-sample** error, or **risk** $R(h)$ of h is defined as

$$R(h) = P(h(X) \neq Y) = E\mathbb{I}(h(X) \neq Y) = \int \mathbb{I}(h(x) \neq y) P(x, y) dx\, dy. \tag{8}$$

This is the intrinsic, or theoretical, error of h with respect to the background probability $P(x, y)$. Since most machine learning problems, if not all, are not concerned with finding out the probability distribution $P(x, y)$, it is only a theoretical device. Nonetheless, it is an important quantity, as it represents all possible errors for any future datasets of the nature coming from the probability distribution $P(x, y)$. Note also that empirical error (7) is an unbiased estimator of $R(h)$.

3.2 Bayes classifier

As we shall see shortly, the Bayes classifier is an optimal classifier. However, since categorical variables do not represent numbers, it does not make sense to define it as a (theoretical) average of outcomes, which is what we did for the regression problem. Instead, we define it by

$$\beta(x) = \underset{j \in \mathfrak{Y}}{\operatorname{argmax}} P(Y = j \,|\, x). \tag{9}$$

This β is called a **Bayes classifier**. (This way of choosing the most frequent label is called **majority voting**.) Since in most cases we do not know the probability $P(x, y)$, Bayes classifier is also only a theoretical construct.

Let $h : \mathfrak{X} \to \mathfrak{Y}$ be any classifier. The definition (9) is equivalent to saying that

$$P(Y = \beta(x) \,|\, x) \geq P(Y = h(x) \,|\, x).$$

which can be rewritten as

$$\int \mathbb{I}(y = \beta(x) \,|\, x)\, P(y \,|\, x) dy \geq \int \mathbb{I}(y = h(x) \,|\, x)\, P(y \,|\, x) dy.$$

Thus, equivalently, we have

$$\int \mathbb{I}(y \neq \beta(x) \,|\, x)\, P(y \,|\, x) dy \leq \int \mathbb{I}(y \neq h(x) \,|\, x)\, P(y \,|\, x) dy.$$

Since $P(y \,|\, x)\, P(x) dx\, dy = P(x, y) dx\, dy$, upon integrating this with $P(x)dx$, we have

$$\int \mathbb{I}(y \neq \beta(x) \,|\, x)\, P(x, y) dx\, dy \leq \int \mathbb{I}(y \neq h(x) \,|\, x)\, P(x, y) dx\, dy.$$

This proves the following theorem.

Theorem 3. *The Bayes classifier is the* best *among all possible classifiers in the sense that*

$$R(\beta) = P(\beta(X) \neq Y) \leq P(h(X) \neq Y) = R(h),$$

for any classifier $h : \mathfrak{X} \to \mathfrak{Y}$.

The risk or the generalization error of the Bayes classifier is called the **Bayes error**, which of course is denoted by $R(\beta)$. The theorem above says that the Bayes error is the lower bound of the risk or the generalization error of any classifier.

As for the bias-variance decomposition, it is quite involved to be presented here. An interested reader may consult Domingos' paper [2].

References

[1] Banko, M. and Brill, E., *Scaling to Very Very Large Corpora for Natural Language Disambiguation*, Proceeding ACL '01 Proceedings of the 39th Annual Meeting on Association for Computational Linguistics Pages 26-33; http://ucrel.lancs.ac.uk/acl/P/P01/P01-1005.pdf

[2] Domingos, P. *A Unified Bias-Variance Decomposition and its Applications*, Proc. 17th International Conf. on Machine Learning (2000)

[3] *Kakutani, S, Notes on infinite product measure spaces I*, Proc. Imp. Acad. 19 (1943), no. 3, 148–151

[4] *Kakutani, S. Notes on infinite product measure spaces II*, Proc. Imp. Acad. 19 (1943), no. 4, 184–188