Let $\mathcal{X}$ — this is the input space (instance space), and $\mathcal{Y}$ — is the output space (label space). In case of binary classification $\mathcal{Y}$ is identified with the multitude $\{-1, +1\}$. We denote those classes by -1 and 1. The question of learning is reduced to the question of estimating a functional relationship of the form $f : \mathcal{X} \to \mathcal{Y}$, that is a relationship between input and output.

Such a mapping f is called a classifier. In order to do this, we get access to some training points $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathcal{X} \times \mathcal{Y}$. We do not make specific assumptions on the spaces $\mathcal{X}$ or $\mathcal{Y}$, but we do make an assumption on the mechanism which generates those training points. Namely, we assume that there exists a joint probability distribution P on $\mathcal{X} \times \mathcal{Y}$, and the training examples (X_i, Y_i) are sampled independently from this distribution P.

The main measure of the quality of the classifier f is the loss function ℓ, which tells us the "cost" of classifying instance $X \in \mathcal{X}$ as $Y \in \mathcal{Y}$.

In regression, where the output variables Y take values that are real numbers rather than class labels, a well-known loss function is the squared error loss function $\ell(X, Y, f(X)) = (Y - f(X))^2$.

While the loss function measures the error of a function on some individual data point, the risk of a function is the average loss over data points generated according to the underlying distribution P,

$$R(f) := E(\ell(X, Y, f(X))).$$

To find a good classifier f we need to find one for which R(f) is as small as possible. The best classifier is the one with the smallest risk value $R(f)$.

What kind of functions f to consider? To formalize this, we consider some underlying space $\mathcal{F}$of functions which map $\mathcal{X}$ to $\mathcal{Y}$. At first glance, the most natural way would be to allow all possible functions from $\mathcal{X}$ to $\mathcal{Y}$ as classifier, that is to choose $F_{all} = \{f : \mathcal{X} \to \mathcal{Y}\}$. In this case, one can formally write down what the optimal classifier should be. Given the underlying probability distribution P, this classifier is defined as follows:

$$f_{Bayes}(x) := \begin{cases} 1, & if\ P(Y = 1 | X = x) \geq 0.5 \\ -1, & otherwise. \end{cases}$$

This is the so-called "Bayes classifier". Intuitively, what it does is as follows. For each point in the space $\mathcal{X}$, it looks at the function $\eta(x) := P(Y = 1 | X = x)$. If we assume that $P(Y = 1 | X = x) = 1$, this means that the true label Y of the point $X = x$ satisfies $Y = 1$ with certainty (probability 1). Hence, an optimal classifier should also take this value, that is it should choose $f(x) = 1$. Now assume that the classes slightly overlap, for example $P(Y = 1 | X = x) = 0.9$. This still means that in an overwhelming number of cases the label of object x is $+1$, thus this is what the classifier f should choose. The same holds as long as the overlap is so small that $\eta(x) \geq 0.5$.

In practice, it is impossible to directly compute the Bayes classifier. The reason is that, as we explained above, the underlying probability distribution is unknown to the learner. Given some training points $(X_1, Y_1), \ldots, (X_n, Y_n)$ which have been drawn independently from some unknown probability distribution P, and given some loss function, how can we construct a function $f : \mathcal{X} \to \mathcal{Y}$ which has risk $R(f)$ as close as possible to the risk of the Bayes classifier?

At this point, note that not only is it impossible to compute the Bayes error, but also the risk of a function f cannot be computed without knowledge of P. This is where SLT comes in. It provides a framework to analyze this situation, to come up with solutions, and to provide guarantees on the goodness of these solution.