Lecture 3. Support Vector Machine and Large Margin Learning

Bao Wang
Department of Mathematics
Scientific Computing and Imaging Institute
University of Utah
Math 5750/6880, Fall 2023

- Consider the two-class linear classifier

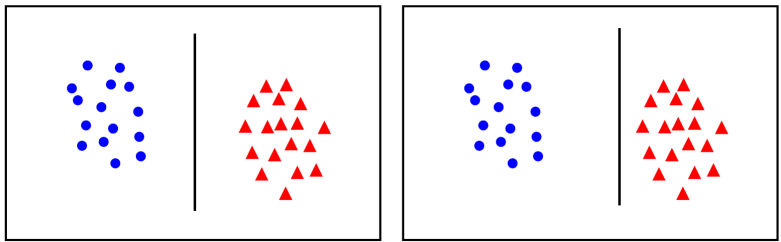
$$y(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x}) + b \tag{1}$$

where $\phi(\boldsymbol{x})$ denotes a fixed feature-space transformation, and b is the bias.

> Let the training data be $\{\boldsymbol{x}_n, t_n\}_{n=1}^N$ with $t_n \in \{-1, 1\}$.

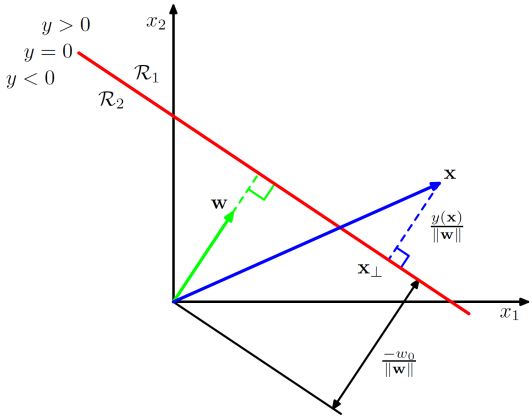
> The new data $\boldsymbol{x}$ is classified based on $sign(y(\boldsymbol{x}))$.

Assume the training data set is linearly separable in feature space, i.e. there exists at least one set of $\boldsymbol{w}$ and b s.t. a function of the form (1) satisfies $y(\boldsymbol{x}_n) > 0 (< 0)$ for points having $t_n = +1(-1)$, i.e., $t_n y(\boldsymbol{x}_n) > 0$ for all training data points.



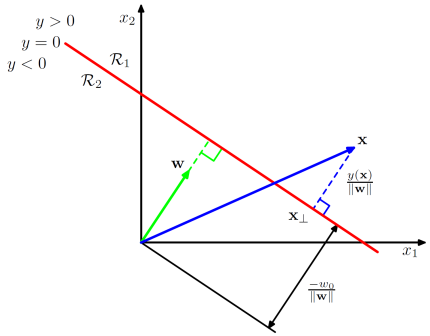
Which classifier is better?

Geometry of the linear discriminant



Signed distance: $y(\boldsymbol{x})/\|\boldsymbol{w}\|$.

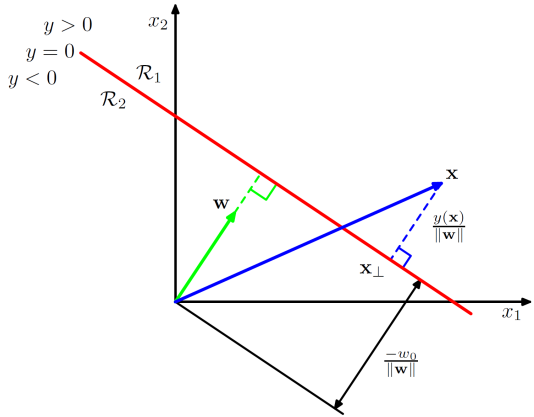
Geometry of the linear discriminant



If all data points are correctly classified, then $t_n y(\boldsymbol{x}_n) > 0$ for all n. Thus the distance of a point $\boldsymbol{x}_n$ to the decision surface is given by

$$\frac{t_n y(\boldsymbol{x}_n)}{\|\boldsymbol{w}\|} = \frac{t_n\Big(\boldsymbol{w}^\top \phi(\boldsymbol{x}_n) + b\Big)}{\|\boldsymbol{w}\|}. \tag{2}$$

Geometry of the linear discriminant



The **margin** is given by the perpendicular distance to the closest point $\boldsymbol{x}_n$ from the data set; we wish to optimize the parameters $\boldsymbol{w}$ and b in order to maximize this distance.

The maximum margin solution is found by solving

$$\arg\max_{\boldsymbol{w},b}\left\{\frac{1}{\|\boldsymbol{w}\|}\min_{n}\left[t_n(\boldsymbol{w}^\top\phi(\boldsymbol{x}_n)+b)\right]\right\}. \tag{3}$$

Direct solution of this optimization problem would be very complex, and so we shall convert it into an equivalent problem that is much easier to solve.

Note that the decision boundary $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ is the same as $\kappa \boldsymbol{w}^\top \boldsymbol{x} + \kappa b = 0$ for any $\kappa \neq 0$.

Rescaling $\boldsymbol{w} \to \kappa \boldsymbol{w}$ and $b \to \kappa b$ does not change the distance of $\boldsymbol{x}_n$ to the decision boundary, i.e. $t_n y(\boldsymbol{x}_n)/\|\boldsymbol{w}\|$ remains unchanged. We can select κ, s.t.

$$t_n\Big(\kappa \boldsymbol{w}^\top \phi(\boldsymbol{x}_n) + \kappa b\Big) = 1 \tag{4}$$

for the point that is closest to the surface.

Let $\hat{\boldsymbol{w}} = \kappa \boldsymbol{w}$ and $\hat{b} = \kappa b$, then we have

$$\arg\max_{\boldsymbol{w},b} \left\{ \frac{1}{\|\kappa \boldsymbol{w}\|} \min_n \Big[t_n(\kappa \boldsymbol{w}^\top \phi(\boldsymbol{x}_n) + \kappa b) \Big] \right\} = \arg\max_{\boldsymbol{w},b} \left\{ \frac{1}{\|\kappa \boldsymbol{w}\|} \right\} = \arg\max_{\hat{\boldsymbol{w}},b} \left\{ \frac{1}{\|\hat{\boldsymbol{w}}\|} \right\}. \tag{5}$$

$$\arg\max_{\hat{\boldsymbol{w}},b}\left\{\frac{1}{\|\hat{\boldsymbol{w}}\|}\right\} = \arg\max_{\boldsymbol{w},b}\left\{\frac{1}{\|\boldsymbol{w}\|}\right\}$$

Using the above rescaling, all data points will satisfy the constraints

$$t_n\Big(\boldsymbol{w}^\top\phi(\boldsymbol{x}_n)+b\Big)\geq 1,\quad n=1,\cdots,N, \tag{6}$$

which is the canonical representation of the decision hyperplane. In the case of data points for which the equality holds, the constraints are *active*, whereas for the remainder they are *inactive*.

At least one active constraint, because there will always be a closest point, and once the margin has been maximized there will be at least two active constraints.

The optimization problem then simply requires that we maximize $\frac{1}{\|\boldsymbol{w}\|}$, which is equivalent to minimizing $\|\boldsymbol{w}\|^2$, and so we have to solve the optimization

$$\arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{7}$$

subject to the constraints given by (6). The factor of $1/2$ in (7) is included for later convenience. This is an example of a *quadratic programming* problem in which we are trying to minimize a quadratic function subject to a set of linear inequality constraints.

How to solve the constraint quadratic programming above?

We introduce Lagrange multipliers $a_n \geq 0$, with one multiplier a_n for each of the constraints in (6), giving the Lagrangian function[1]

$$L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{n=1}^{N} a_n\{t_n(\boldsymbol{w}^\top \phi(\boldsymbol{x}_n) + b) - 1\}, \quad \boldsymbol{a} = (a_1, \cdots, a_N)^\top. \tag{8}$$

Setting the derivatives of $L(\boldsymbol{w}, b, \boldsymbol{a})$ w.r.t. $\boldsymbol{w}$ and b equal to zero, we have

$$\boldsymbol{w} = \sum_{n=1}^{N} a_n t_n \phi(\boldsymbol{x}_n); \quad 0 = \sum_{n=1}^{N} a_n t_n. \tag{9}$$

[1] KKT condition: If we wish to minimize the function $f(\boldsymbol{x})$ s.t. $g(\boldsymbol{x}) \geq 0$, then we minimize the Lagrangian function $L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) - \lambda g(\boldsymbol{x})$ w.r.t. $\boldsymbol{x}$, subject to $\lambda \geq 0$.

Eliminating $\boldsymbol{w}$ and b from $L(\boldsymbol{w}, b, \boldsymbol{a})$ using these conditions then gives the *dual representation* of the maximum margin problem in which we maximize

$$\tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m), \quad k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) \text{ is the kernel.} \tag{10}$$

with respect to $\boldsymbol{a}$ subject to the constraints

$$a_n \geq 0; \quad \sum_{n=1}^{N} a_n t_n = 0. \tag{11}$$

• The solution to a quadratic programming problem in M variables in general has computational complexity that is $O(M^3)$.

• In going to the dual formulation we have turned the original optimization problem, which involved minimizing (7) over M variables (the dimension of $\boldsymbol{w}$ plus the dimension of b), into the dual problem (10), which has N variables.

• For a fixed set of basis functions whose number M is smaller than the number N of data points, the move to the dual problem is more computationally expensive.

- However, it allows the model to be reformulated using kernels, and so the maximum margin classifier can be applied efficiently to feature spaces whose dimensionality exceeds the number of data points, including infinite feature spaces.

Kernel functions: inner products in feature spaces that can have high, or even infinite, dimensionality. Working directly with kernel functions can handle very high dimensional feature space (ϕ). For instance, a second order polynomial kernel can represent six dimensional features:

$$\begin{aligned} k(\boldsymbol{x}, \boldsymbol{z}) &= (1 + \boldsymbol{x}^\top \boldsymbol{z})^2 = (1 + x_1 z_1 + x_2 z_2)^2 \\ &= 1 + 2x_1 z_1 + 2x_2 z_2 + x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1 x_2, x_2^2)(1, \sqrt{2}z_1, \sqrt{2}z_2, z_1^2, \sqrt{2}z_1 z_2, z_2^2)^\top \\ &= \phi(\boldsymbol{x})^\top \phi(\boldsymbol{z}), \end{aligned}$$

where ϕ is a six dimensional feature map, though the input space has dimension two.

To classify new data points using the trained model, we evaluate the sign of $y(\boldsymbol{x})$ defined by (1). This can be expressed in terms of the parameters $\{a_n\}$ and the kernel function by substituting for $\boldsymbol{w}$ using (9) to give

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n t_n k(\boldsymbol{x}, \boldsymbol{x}_n) + b. \tag{12}$$

• The dual constraint optimization problems satisfies the *Karush-Kuhn-Tucker* (KKT) conditions, which in this case require that the following three properties hold

$$a_n \geq 0; \quad t_n y(\boldsymbol{x}_n) - 1 \geq 0; \quad a_n\{t_n y(\boldsymbol{x}_n) - 1\} = 0. \tag{13}$$

Thus for every data point, either $a_n = 0$ or $t_n y(\boldsymbol{x}_n) = 1$.

• Any data point for which $a_n = 0$ will not appear in the sum in (12) and hence plays no role in making predictions for new data points. The remaining data points are called *support vectors*, and because they satisfy $t_n y(\boldsymbol{x}_n) = 1$, they correspond to points that lie on the maximum margin hyperplanes in feature space. This property is central to the practical applicability of SVMs. *Once the model is trained, a significant proportion of the data points can be discarded and only the support vectors retained.*

SVM solution and support vectors

• Having solved the quadratic programming and found a value for $\boldsymbol{a}$, we then determine the value of the threshold parameter b by noting that any support vector $\boldsymbol{x}_n$ satisfies $t_n y(\boldsymbol{x}_n) = 1$. Using (12) gives

$$t_n\Big(\sum_{m\in\mathcal{S}} a_m t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m) + b\Big) = 1 \tag{14}$$

where $\mathcal{S}$ is the indices of the support vectors.

• We can solve this equation for b using an arbitrarily chosen support vector $\boldsymbol{x}_n$, but a numerically more stable solution is obtained by first multiplying through by t_n, note $t_n^2 = 1$, and then averaging these equations over all support vectors and solving for b to give (equation (14) holds for any $n \in \mathcal{S}$)

$$b = \frac{1}{N_{\mathcal{S}}}\sum_{n\in\mathcal{S}}\left(t_n - \sum_{m\in\mathcal{S}} a_m t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)\right), \quad N_{\mathcal{S}} = \#S. \tag{15}$$

We can express the maximum-margin classifier in terms of the minimization of an error function, with a simple quadratic regularizer, in the form

$$\sum_{n=1}^{N} E_\infty\Big(y(\boldsymbol{x}_n)t_n - 1\Big) + \lambda\|\boldsymbol{w}\|^2 \tag{16}$$

where $E_\infty(z)$ is a function that is zero if $z \geq 0$ and ∞ otherwise and ensures that the constraints (6) are satisfied. $(y(\boldsymbol{x}_n)t_n - 1 \geq 0)$

Overlapping Class Distributions

Overlapping class distributions

• So far, we have assumed that the training data points are linearly separable in the feature space $\phi(\boldsymbol{x})$. The resulting support vector machine will give exact separation of the training data in the original input space $\boldsymbol{x}$, although the corresponding decision boundary will be nonlinear.

• In practice, however, the class-conditional distributions may overlap, in which case exact separation of the training data can lead to poor generalization.

• We therefore need a way to modify the SVM so as to allow some of the training points to be misclassified.

- From the following SVM objective function

$$\sum_{n=1}^{N} E_{\infty}\Big(y(\boldsymbol{x}_n)t_n - 1\Big) + \lambda\|\boldsymbol{w}\|^2, \tag{17}$$

we see that in the case of separable classes, we implicitly used an error function that gave infinite error if a data point was misclassified and zero if it was classified correctly, and then optimized the model parameters to maximize the margin.

- We now modify this approach so that data points are allowed to be on the 'wrong side' of the margin boundary, but with a penalty that increases with the distance from the boundary. For the subsequent optimization problem, it is convenient to make this penalty a linear function of this distance.

• We introduce *slack variables*, $\xi_n \geq 0$ where $n = 1, \cdots, N$, with one slack variable for each training data point. These are defined by $\xi_n = 0$ for data points that are on or inside the correct margin boundary and $\xi_n = |t_n - y(\boldsymbol{x}_n)|$ for other points. Thus a data point that is on the decision boundary $y(\boldsymbol{x}_n) = 0$ will have $\xi_n = 1$, and points with $\xi_n > 1$ will be misclassified.

• Data points that are on the correct side of the margin boundary, and which therefore satisfy $y_n t_n \geq 1$, we have $\xi_n = 0$, and for the remaining points we have $\xi_n = 1 - y_n t_n$.

- The exact classification constraints (6) are then replaced with

$$t_n y(\boldsymbol{x}_n) \geq 1 - \xi_n, \quad n = 1, \cdots, N \tag{18}$$

in which the slack variables are constrained to satisfy $\xi_n \geq 0$.

- Data points for which $\xi_n = 0$ are correctly classified and are either on the margin or on the correct side of the margin. Points for which $0 < \xi_n \leq 1$ lie inside the margin, but on the correct side of the decision boundary, and those data points for which $\xi_n > 1$ lie on the wrong side of the decision boundary and are misclassified, as illustrated in Figure 1.

- This is sometimes described as relaxing the hard margin constraint to give a *soft margin* and allows some of the training set data points to be misclassified.

Relaxed SVM: Slack variables

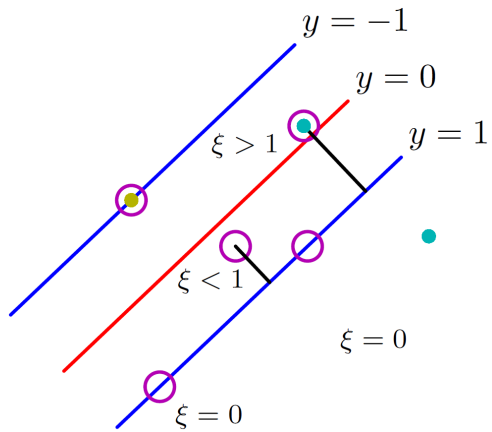


Figure: Illustration of the slack variables $\xi_n \geq 0$. Data points with circles around them are support vectors.

Soft-margin SVM

• Our goal is now to maximize the margin while softly penalizing points that lie on the wrong side of the margin boundary. We therefore minimize

$$C \sum_{n=1}^{N} \xi_n + \frac{1}{2} \|\boldsymbol{w}\|^2, \tag{19}$$

where the parameter $C > 0$ controls the trade-off between the slack variable penalty and the margin. Because any point that is misclassified has $\xi_n > 1$, it follows that $\sum_n \xi_n$ is an upper bound on the number of misclassified points.

• Constraints $t_n(\boldsymbol{w}^\top \phi(\boldsymbol{x}_n) + b) \geq 1 - \xi_n$ instead of $t_n(\boldsymbol{w}^\top \phi(\boldsymbol{x}_n) + b) \geq 1$.

• The parameter C is therefore analogous to (the inverse of) a regularization coefficient because it controls the trade-off between minimizing training errors and controlling model complexity. In the limit $C \to \infty$, we will recover the earlier SVM for separable data.

Soft-margin SVM

We now wish to minimize (19) subject to the constrains (18) together with $\xi_n \geq 0$. The corresponding Lagrangian is given by

$$L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}a_n\Big\{t_n y(\boldsymbol{x}_n) - 1 + \xi_n\Big\} - \sum_{n=1}^{N}\mu_n\xi_n, \tag{20}$$

where $\{a_n \geq 0\}$ and $\{\mu_n \geq 0\}$ are Lagrange multipliers. The corresponding KKT conditions are

$$\begin{aligned} a_n &\geq 0 \\ t_n y(\boldsymbol{x}_n) - 1 + \xi_n &\geq 0 \\ a_n(t_n y(\boldsymbol{x}_n) - 1 + \xi_n) &= 0 \\ \mu_n &\geq 0 \\ \xi_n &\geq 0 \\ \mu_n\xi_n &= 0 \end{aligned} \tag{21}$$

where $n = 1, \cdots, N$.

Soft-margin SVM: dual formulation

We now optimize out $\boldsymbol{w}, b$, and $\{\xi_n\}$ making use of the definition (1) of $y(\boldsymbol{x})$ to give

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{x}} = 0 &\Rightarrow \boldsymbol{w} = \sum_{n=1}^{N} a_n t_n \phi(\boldsymbol{x}_n) \\ \frac{\partial L}{\partial b} = 0 &\Rightarrow \sum_{n=1}^{N} a_n t_n = 0 \\ \frac{\partial L}{\partial \xi_n} = 0 &\Rightarrow a_n = C - \mu_n. \end{aligned} \tag{22}$$

Using these results to eliminate $\boldsymbol{w}, b$, and $\{\xi_n\}$ from the Lagrangian, we obtain the dual Lagrangian in the form

$$\tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m), \tag{23}$$

which is identical to the separable case, except that the constraints are somewhat different.

What are the constraints now?

> $a_n \geq 0$ is required because these are Lagrange multipliers.
> Furthermore, the last equation of (22) together with $\mu_n \geq 0$ implies $a_n \leq C$.

We therefore have to minimize (23) with respect to the dual variables $\{a_n\}$ subject to

$$0 \leq a_n \leq C; \quad \sum_{n=1}^{N} a_n t_n = 0 \tag{24}$$

for $n = 1, \cdots, N$, where $0 \leq a_n \leq C$ are known as *box constraints*. This again represents a quadratic programming problem. If we substitute the first equation of (22) into (1), we see that predictions for new data points are again made by using (12).

A subset of the data points may have $a_n = 0$, in which case they do not contribute to the predictive model (12). The remaining data points constitute the support vectors. These have $a_n > 0$ and hence from (21) (the third equation) must satisfy

$$t_n y(\boldsymbol{x}_n) = 1 - \xi_n. \tag{25}$$

If $a_n < C$, then (22) (the third equation) implies that $\mu_n > 0$, which from (21) (the last equation) requires $\xi_n = 0$ and hence points lie on the margin. Points with $a_n = C$ can lie inside the margin and can either be correctly classified if $\xi \leq 1$ or misclassified if $\xi_n > 1$.

Parameter b

To determine the parameter b in (1), we note that those support vectors for which $0 < a_n < C$ have $\xi_n = 0$ so that $t_n y(\boldsymbol{x}_n) = 1$ and hence will satisfy

$$t_n\Big(\sum_{m\in\mathcal{S}} a_m t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m) + b\Big) = 1. \tag{26}$$

Again, a numerically stable solution is obtained by averaging to give

$$b = \frac{1}{N_{\mathcal{M}}}\sum_{n\in\mathcal{M}}\Big(t_n - \sum_{m\in\mathcal{S}} a_m t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)\Big), \tag{27}$$

where $\mathcal{M}$ denotes the set of indices of data points having $0 < a_n < C$.

We can re-cast the SVM for nonseparable distributions in terms of minimizing a regularized error function. We have seen that for data points that are on the correct side of the margin boundary, and which therefore satisfy $y_n t_n \geq 1$, we have $\xi_n = 0$, and for the remaining points we have $\xi_n = 1 - y_n t_n$. thus the objective function (19) ($C \sum_{n=1}^{N} \xi_n + \frac{1}{2}\|\boldsymbol{w}\|^2$) can be written (up to an overall multiplicative constant) in the form

$$\sum_{n=1}^{N} E_{SV}(y_n t_n) + \lambda \|\boldsymbol{w}\|^2, \tag{28}$$

where $\lambda = (2C)^{-1}$, and $E_{SV}(\cdot)$ is the *hinge error* function defined by

$$E_{SV}(y_n t_n) = [1 - y_n t_n]_+ \tag{29}$$

where $[\cdot]_+$ denotes the positive part.

Probably approximately correct (PAC) learning framework. The goal of the PAC framework is to understand how large a data set needs to be in order to give good generalization. It also gives bounds for the computational cost of learning.
Suppose that a data set $\mathcal{D}$ of size N is drawn from some joint distribution $p(\boldsymbol{x}, \boldsymbol{t})$ where $\boldsymbol{x}$ is the input variable and $\boldsymbol{t}$ represents the class label, and that we restrict attention to 'noise free' situations in which the class labels are determined by some (unknown) deterministic function $\boldsymbol{t} = \boldsymbol{g}(\boldsymbol{x})$. In PAC learning we say that a function $\boldsymbol{f}(\boldsymbol{x}; \mathcal{D})$, drawn from a space $\mathcal{F}$ of such functions on the basis of the training set $\mathcal{D}$, has good generalization if its expected error rate is below some pre-specified threshold ϵ, so that

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{t}}[I(f(\boldsymbol{x}; D) \neq \boldsymbol{t})] < \epsilon \tag{30}$$

where $I(\cdot)$ is the indicator function, and the expectation is with respect to the distribution $p(\boldsymbol{x}, \boldsymbol{t})$.

The quantity on the left-hand side is a random variable, because it depends on the training set $\mathcal{D}$, and the PAC framework requires that (30) holds, with probability greater than $1 - \delta$, for a data set $\mathcal{D}$ drawn randomly from $p(\boldsymbol{x}, \boldsymbol{t})$. Here δ is another pre-specified parameter, and the terminology 'probably approximately correct' comes from the requirement that with high probability (greater than $1 - \delta$), the error rate be smaller (less than ϵ). For a given choice of model space $\mathcal{F}$, and for given parameters ϵ and δ, PAC learning aims to provide bounds on the minimum size N of data set needed to meet this criterion. A key quantity in PAC learning is the *Vapnik-Chervonenkis* dimension, or VC dimension, which provides a measure of the complexity of a space of functions, and which allows the PAC framework to be extended to spaces containing an infinite number of functions.

The bounds derived within the PAC framework are often described as worst case, because they apply to any choice for the distribution $p(\boldsymbol{x}, \boldsymbol{t})$, so long as both the training and the test examples are drawn (independently) from the same distribution, and for *any* choice for the function $\boldsymbol{f}(\boldsymbol{x})$ so long as it belongs to $\mathcal{F}$. In real-world applications of machine learning, we deal with distributions that have significant regularity, for example in which large regions of input space carry the same class label. As a consequence of the lack of any assumptions about the form of the distribution, the PAC bounds are very conservative, in other words they strongly over-estimate the size of data sets required to achieve a given generalization performance. For this reason, PAC bounds have found few, if any, practical applications.