

## Complexity and generalization

Our goal here is to understand what type of performance guarantees we can give classification methods that base their predictions on finite training data. This is a core theoretical question in machine learning. For this purpose we will need to understand in detail what “complexity” means in terms of classifiers. A single classifier is never complex or simple; complexity is a property of the set of classifiers or the model. Each model selection criterion we have encountered provided a slightly different definition of “model complexity”.

Our focus here is on performance guarantees that will eventually relate the margin we can attain on the training set to the generalization error, especially for linear classifiers (geometric margin). Later we will also cover ensemble classifiers where each classifier votes for one label versus another.

Let’s start by motivating the complexity measure we need for this type of analysis. We will use a very simple classifier known as a decision stump for this purpose. A stump can be seen as a linear classifier that only depends on a single coordinate of the input vector. For example, in two dimensions, where  $\underline{x} = [x_1, x_2]^T$ , the discriminant function of a vertical stump is given by  $f(\underline{x}; \theta) = \theta_1 x_1 + \theta_0$ . The resulting decision boundary is  $\theta_1 x_1 + \theta_0 = 0$  which is a vertical line (a line in  $x_2$  direction). We could also write the stump as  $f(\underline{x}; \theta) = \theta_1(x_1 - \mu)$  so as to make the location of the stump  $\mu = \theta_0/\theta_1$  more explicit. When we refer to the resulting classifier (sign of the discriminant function) we use

$$h(\underline{x}; \theta) = \text{sign}(f(\underline{x}; \theta)) = \text{sign}(\theta_1(x_1 - \mu)) \quad (1)$$

A horizontal stump is defined analogously. Note that, in general, the coordinate that the stump depends on has to be included as a parameter. In the example below, however, we will focus only on vertical stumps or the set  $\mathcal{F}_{S,1}$ . We’ll use  $\mathcal{H}_{S,1}$  for the corresponding set of binary  $\pm 1$  valued functions (classifiers). Example decision boundaries are displayed in Figure 1.

Suppose now that we are getting the data points in a sequence and we are interested in seeing when our predictions for future points become constrained by the labeled points we have already seen. Such constraints pertain to both the data and the set of classifiers  $\mathcal{H}_{S,1}$ . If we are given only a single labeled training example, then we can always find a stump  $h_1, h_2 \in \mathcal{H}_{S,1}$  such that they classify the single training example in the same way but will differ on the new point. So a single training example offers no help in terms of constraining our predictions. Now, consider Figure 1e. Having seen the labels for the first two points, labeled  $-1$  and  $+1$ , all classifiers  $h \in \mathcal{H}_{S,1}$  that are consistent with these two labeled points have to predict  $+1$  for the next (third) point in the figure. Since the labels we

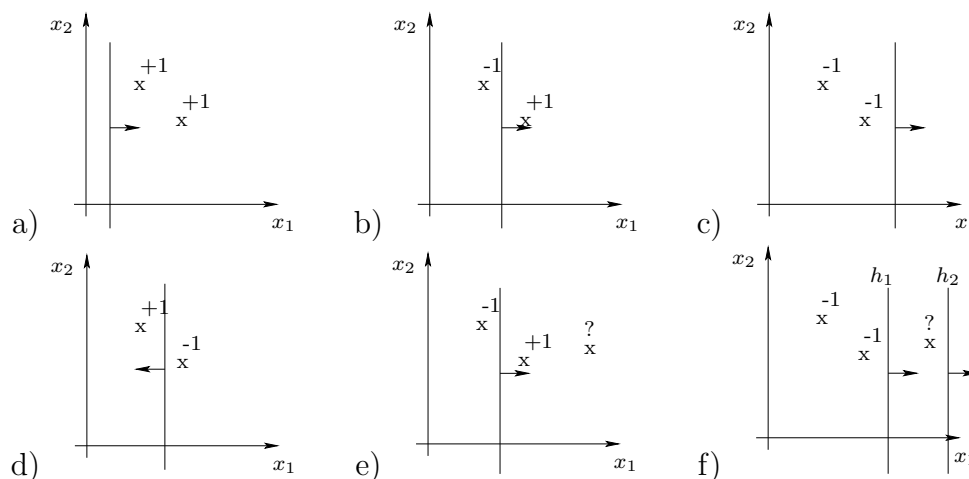


Figure 1: Possible decision boundaries corresponding to decision stumps that rely only on  $x_1$  coordinate. The arrow (normal) to the boundary specifies the positive side.

have seen force us to classify the new point in only one way, we can claim to have learned something. We can also understand this as a limitation of the complexity of our set of classifiers. Figure 1f illustrates an alternative scenario where we can find two classifiers  $h_1 \in \mathcal{H}_{S,1}$  and  $h_2 \in \mathcal{H}_{S,1}$ , both consistent with the first two labels in the figure, but make different predictions for the new point. We could therefore classify the new point either way. So, after two training points, depending on how they are labeled, our classifiers are partially constrained in terms of what they can predict for the new point.

We can summarize our observations so far in the following way. The stumps  $\mathcal{H}_{S,1}$  can classify any two points (in general position) in all possible ways (Figures 1a-d) but are already partially constrained in how they assign labels to three points (Figure 1e). More formally, we say that  $\mathcal{H}_{S,1}$  *shatters* – can generate all possible labels – for two points (in general position) but not three. In other words, we can pick any labeling of two points and there is a classifier  $h \in \mathcal{H}_{S,1}$  that reproduces those labels for the two points. Shattering is a combinatorial property of a set of classifiers, not a single classifier. A single fixed classifier cannot shatter even a single point since it always classifies the point in one way.

Similarly, for linear classifiers in 2–dimensions, we can obtain all the eight possible labelings of three points by using linear classifiers (Figure 2a). Thus linear classifiers in two dimensions shatter three points. However, there are labels over four points that no linear classifier can produce (Figure 2b).

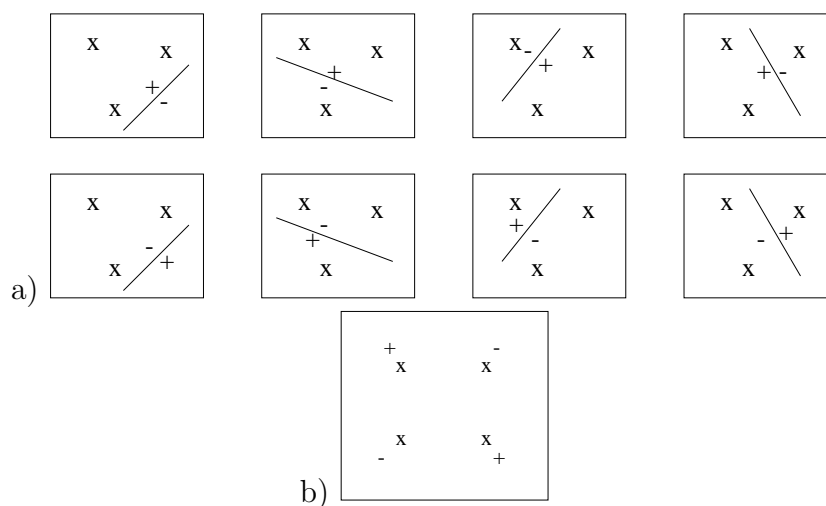


Figure 2: Linear classifiers on the plane can shatter three points a) but not four b).

**VC-dimension.** As we increase the number of data points, the set of classifiers we are considering may no longer be able to label the points in all possible ways. Such emerging constraints are critical to be able to predict labels for new points. This motivates a key measure of complexity of the set of classifiers, the *Vapnik-Chervonenkis dimension*. The VC-dimension is defined as the maximum number of points that a set of classifiers can shatter. So, the VC-dimension of  $\mathcal{H}_{S,1}$  is two, denoted as  $d_{VC}(\mathcal{H}_{S,1})$ , and the VC-dimension of linear classifiers on the plane is three.

The VC-dimension of the set of linear classifiers in  $d$ -dimensions is  $d + 1$ , i.e., the same as the number of parameters. This correspondence with the degrees of freedom and the VC-dimension is often but not always correct. A classifier with a single real parameter can have an infinite VC-dimension (the labelings can be encoded into the parameter at different resolutions).

Note that VC-dimension is a combinatorial property associated with a set of classifiers  $\mathcal{H}$ . Since we derive classifiers using discriminant functions by taking the sign, we can also talk about the VC-dimension of the set of discriminant functions  $\mathcal{F}$ . We will use these interchangeably.

**VC-dimension and the number of labelings.** Let  $N_{\mathcal{F}}(\underline{x}_1, \dots, \underline{x}_n)$  be the number of distinct ways that we can label  $n$  points  $\underline{x}_1, \dots, \underline{x}_n$  by choosing classifiers from set  $\mathcal{F}$ . If  $\mathcal{F}$

shatters these points, then necessarily

$$N_{\mathcal{F}}(\underline{x}_1, \dots, \underline{x}_n) = 2^n \quad (2)$$

The number of possible labelings may depend crucially on how the points are chosen. For example, we would seriously limit the possible ways to label the points if the points all lie on a line for a set of linear classifiers. We can get around this dependence by choosing the points so as to reveal the real (labeling) power in the set of classifiers. In other words, we can define

$$N_{\mathcal{F}}(n) = \max_{\underline{x}_1, \dots, \underline{x}_n} N_{\mathcal{F}}(\underline{x}_1, \dots, \underline{x}_n) \quad (3)$$

which is known as the growth function. The VC-dimension of the set of classifiers  $\mathcal{F}$  is now easily defined on the basis of  $N_{\mathcal{F}}(n)$ :  $d_{VC}(\mathcal{F}) = \max\{n : N_{\mathcal{F}}(n) = 2^n\}$ . Note that we may be able to find a smaller set of specifically chosen points that the set of classifiers cannot shatter but there cannot be any set of more than  $d_{VC}$  points that the classifier can shatter.

The effective size of a set of classifiers can be understood in terms of the number of ways that the classifiers can label  $n$  points, i.e., on the basis of the growth function. We can bound the growth function using the VC-dimension: if we let  $d_{VC} = d_{VC}(\mathcal{F})$ , then

$$N_{\mathcal{F}}(\underline{x}_1, \dots, \underline{x}_n) \leq N_{\mathcal{F}}(n) = \begin{cases} 2^n, & \text{if } n \leq d_{VC} \\ \leq \left(\frac{ne}{d_{VC}}\right)^{d_{VC}}, & \text{when } n > d_{VC} \end{cases} \quad (4)$$

Note that the number of possible labelings we can generate grows only polynomially after  $n$  exceeds the VC-dimension. This is the regime where we can hope to learn something based on the training set.

**Generalization guarantees.** The VC-dimension immediately generalizes our previous results for bounding the expected error from a finite number of classifiers. There are a number of technical steps involved that we won't get into, however. Roughly speaking, the log of the number of possible labelings that a set of classifiers can produce (specified by  $d_{VC}$ ) takes the place of the logarithm of the number of classifiers in our set. In other words, we are counting the number of classifiers on the basis of how they can label points, not based on their identities in the set. More precisely, we have for any set of classifiers  $\mathcal{F}$ : with probability at least  $1 - \delta$  over the choice of the training set,

$$R(f) \leq R_n(f) + \epsilon(n, d_{VC}, \delta), \quad \text{uniformly for all } f \in \mathcal{F} \quad (5)$$

where the complexity penalty is now a function of  $d_{VC} = d_{VC}(\mathcal{F})$ :

$$\epsilon(n, d_{VC}, \delta) = \sqrt{\frac{d_{VC}(\log(2n/d_{VC}) + 1) + \log(4/\delta)}{n}} \quad (6)$$