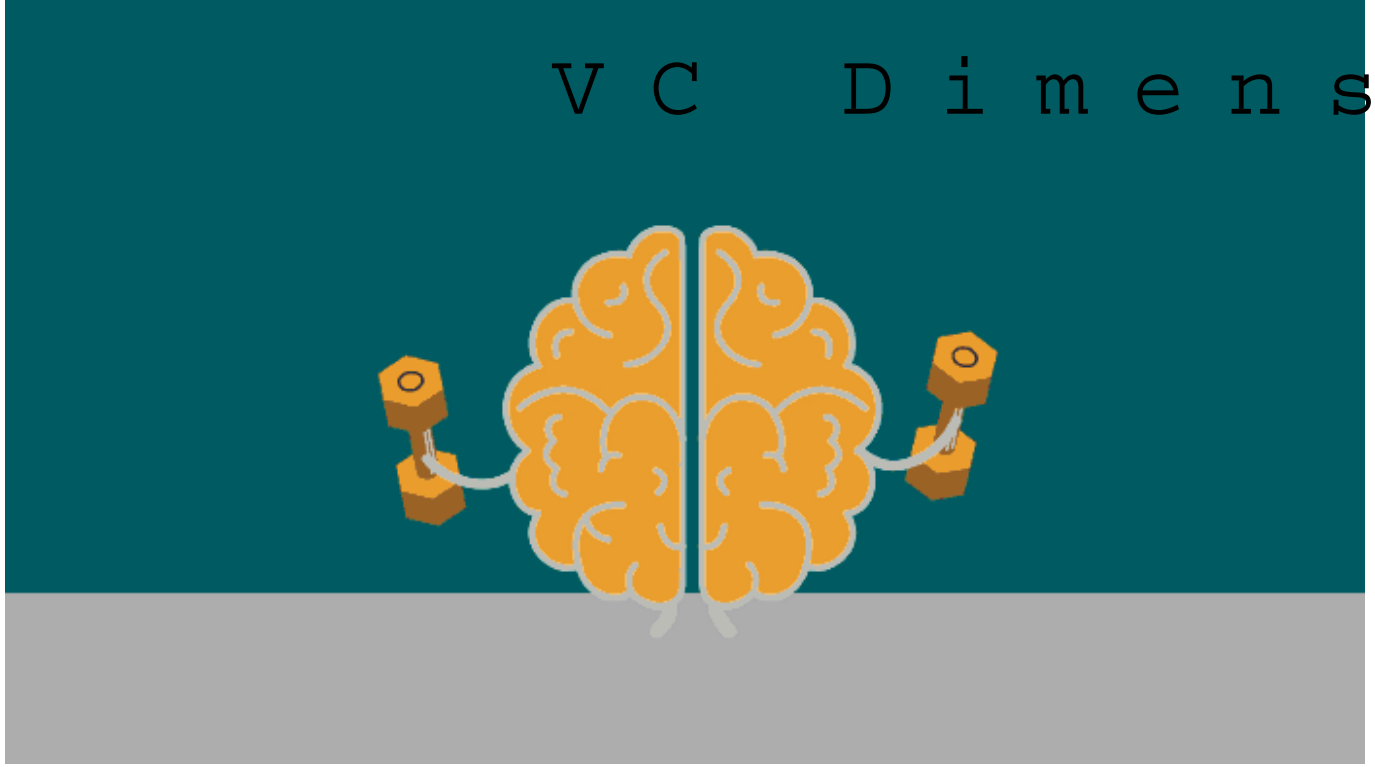


When choosing a classifier for your data, an obvious question to ask is "What kind of data can this classifier classify?". For example, if you know your points can easily be separated by a single line, you may opt to choose a simple linear classifier, whereas if you know your points will be in many separate groups, you may opt to choose a more powerful classifier such as a random forest or multilayer perceptron. This fundamental question can be answered using a classifier's **VC dimension**, which is a concept from computational learning theory that formally quantifies the power of a classification algorithm.

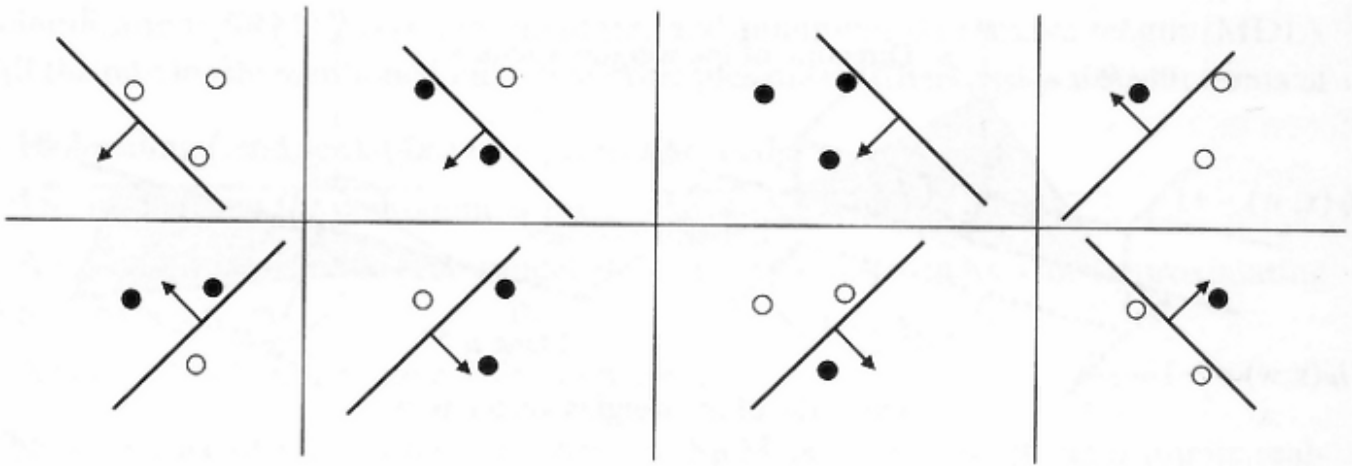
The VC dimension of a classifier is defined by Vapnik and Chervonenkis to be the cardinality (size) of the largest set of points that the classification

algorithm can **shatter** [1]. This may seem like a simple definition, but it is easy to misinterpret, so I will now go into more detail here and explain the key terms in the definition. We will use 2-D examples for simplicity, but these ideas generalize to any number of dimensions.

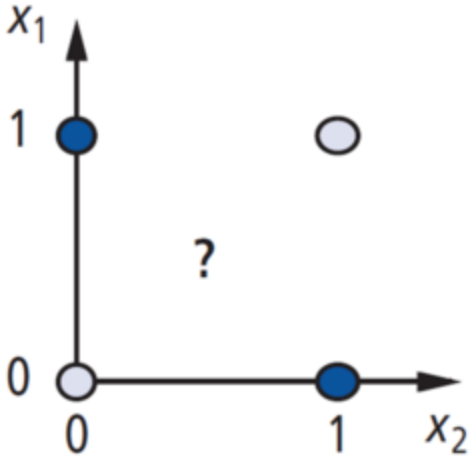
Shattering a set of points

A configuration of **N** points on the plane is just any placement of **N** points. In order to have a VC dimension of *at least* **N**, a classifier must be able to shatter a *single* configuration of **N** points. In order to **shatter** a configuration of points, the classifier must be able to, for every possible **assignment** of positive and negative for the points, perfectly partition the plane such that the positive points are separated from the negative points. For a configuration of **N** points, there are **2^N** possible assignments of positive or negative, so the classifier must be able to properly separate the points in each of these.

In the below example, we show that the VC dimension for a linear classifier is *at least* 3, since it can shatter this configuration of 3 points. In each of the $2^3 = 8$ possible assignment of positive and negative, the classifier is able to perfectly separate the two classes.



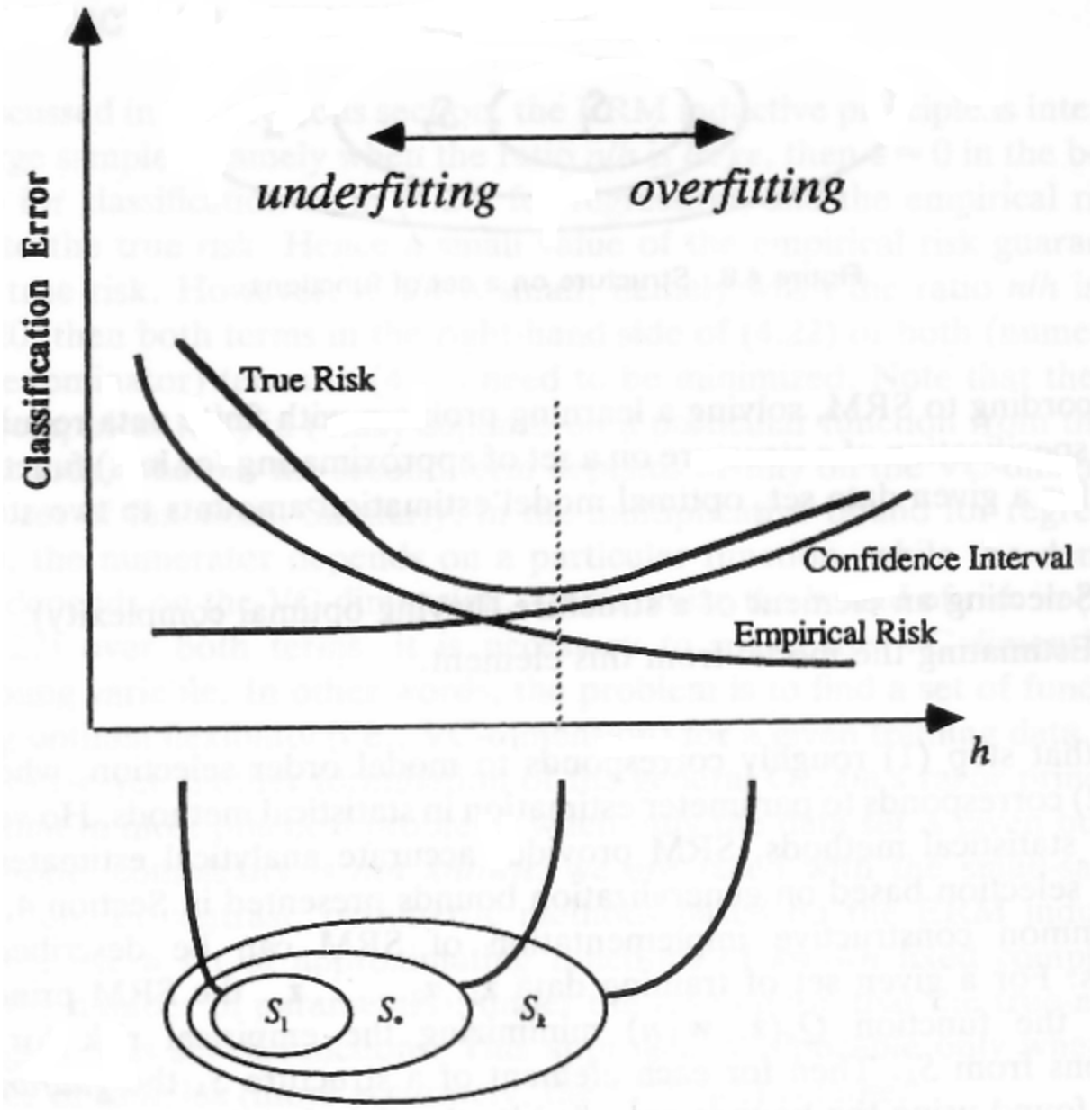
Now, we show that a linear classifier is *lower than* 4. In this configuration of 4 points, the classifier is unable to segment the positive and negative classes in at least one assignment. Two lines would be necessary to separate the two classes in this situation. We actually need to prove that there *does not exist* a 4 point configuration that can be shattered, but the same logic applies to other configurations, so, for brevity's sake, this example is good enough.



Since we have now shown that the linear classifier's VC dimension is *at least* 3, and *lower than* 4, we can finally conclude that its VC dimension is *exactly* 3. Again, remember that in order to have a VC dimension of **N**, the classifier must only shatter *a single* configuration of **N** points — there will likely be many other configurations of **N** points that the classifier cannot shatter.

Applications of VC dimension

Now that you know what the VC dimension is, and how to find it, it is important to also understand what its practical implications are. In most cases, the exact VC dimension ofa classifier is not so important. Rather, it is used more so to classify different types of algorithms by their complexities; for example, the class of simple classifiers could include basic shapes like lines, circles, or rectangles, whereas a class of complex classifiers could include classifiers such as multilayer perceptrons, boosted trees, or other nonlinear classifiers. The complexity of a classification algorithm, which is directly related to its VC dimension, is related to the trade-off between bias and variance.



In this image, we visualize the effects of model complexity. On the bottom, each **S_i** represents a set of models that are similar in VC dimension, or complexity. On the graph above, VC dimension is measured on the x-axis as **h**. Observe that as complexity increases, you transition from underfitting to

overfitting; adding complexity is good up until a certain point, after which you begin to overfit on the training data.

Another way of thinking about this is through bias and variance. A low complexity model will have a high bias and low variance; while it has low expressive power leading to high bias, it is also very simple, so it has very predictable performance leading to a low variance. Conversely, a complex model will have a lower bias since it has more expressiveness, but will have a higher variance as there are more parameters to tune based on the sample training data. Generally, a model with a higher VC dimension will require more training data to properly train, but will be able to identify more complex relationships in the data.

At some level of model complexity there will exist an ideal balance between bias and variance, denoted by the dotted vertical line, at which you are neither underfitting nor overfitting to your data. In other words, you should aim to choose a classifier with a level of complexity that is *just* enough for your classification task — any less would lead to underfitting, and any more would lead to overfitting.
