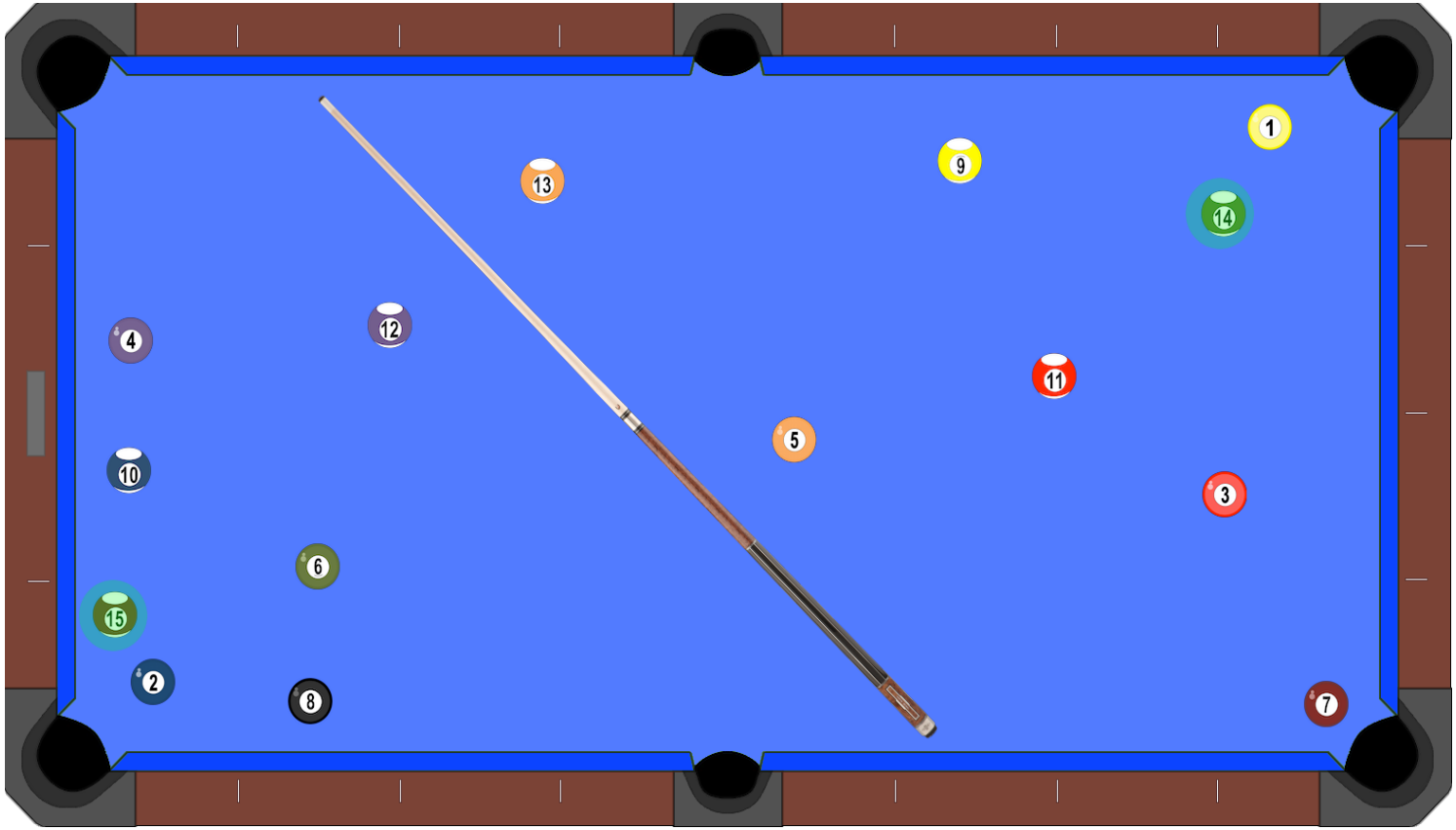
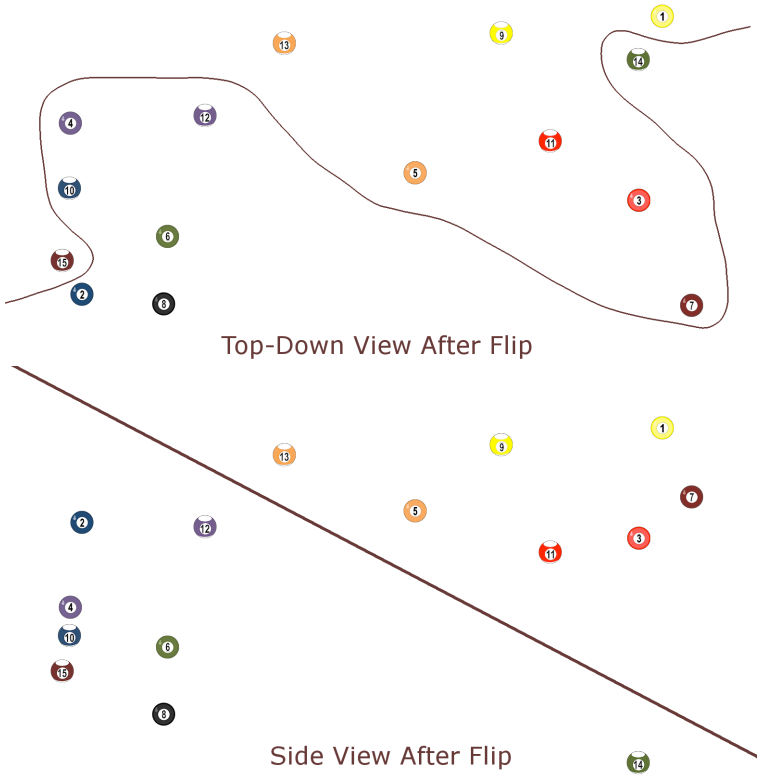
The Kernel Trick

Let's revisit the dojo billiards ball example. Let's say your instructor notices how awesome you are at separating even and odd samples so decides to throw you a curve, by placing the remaining two balls at very difficult to separate spots on the table:



Your original instructions only were to use any *straight surface* to separate a mixed group of billiard balls into even and odd sets. At this point, it doesn't matter how you orient the pool stick: unless you're next to a black hole where space and time get warped out, you'll never be able to get the flat pool-stick, or any linear structure for that matter, to separate between even and odd featured balls now. So how do you handle the situation? The most mature way possible, of course—by flipping the table! Not out of anger, but a very *calculated* and *coordinated* table-flip that sends all of the balls flying into the air.

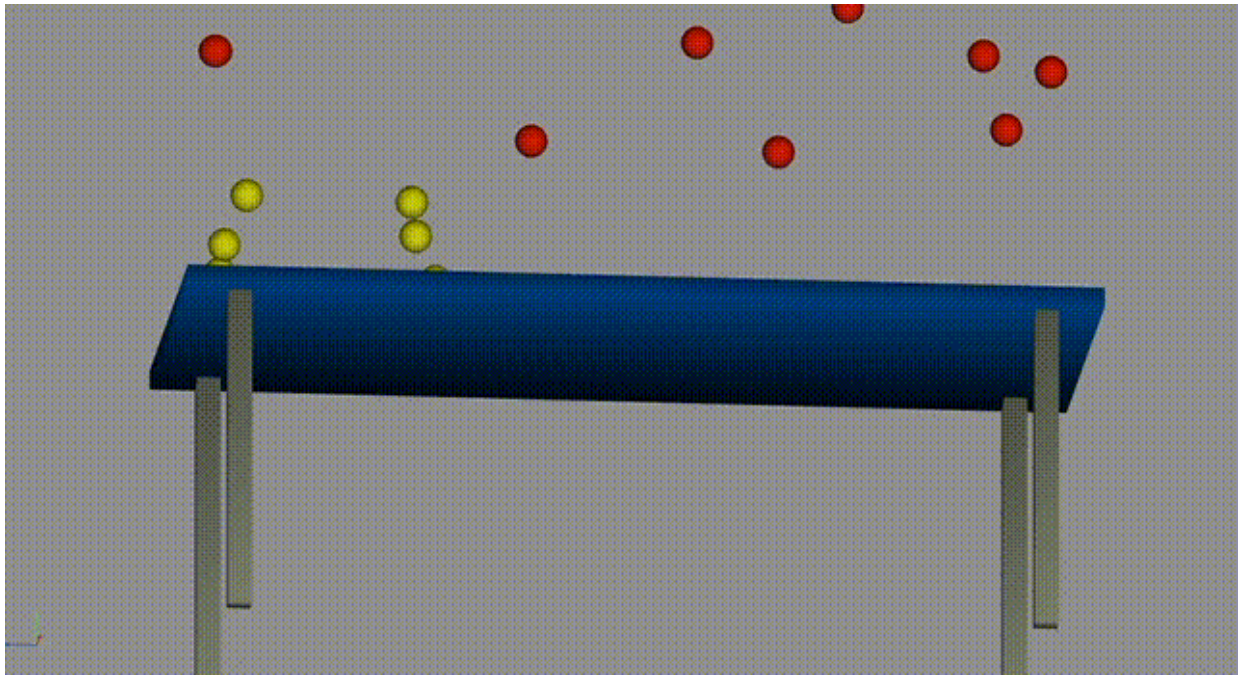


For the split second second all the balls are in the air, you use your coding-ninja skills to slide a large, *straight*, sheet of paper clean right between even and odd balls. From the perspective of the instructor (who just so happens to be floating directly above the pool table as you can tell from the first few images), it looks like the splitting of the ball samples was done using a curvy line. But from your perspective, looking at the situation from the side of the table and seeing all the balls floating in the air, a very straight, single, linear-sheet of paper was able to split the two sets of balls, properly fulfilling the objective. This trick of yours is ***almost*** called the kernel trick, used in SVC and many other mathematical algorithms.

(ノ °Д°）ノ ┻━┻

The Trick

To be able to place the decision boundary between the billiard balls, you had to get them into the air by transforming them from their original three-dimensional feature-space (table position x, table position y, even/odd status) to a higher, 4D space, where the table position z-coordinate was added. Once the balls were in the air, you could see where the flat paper sheet needed to be slipped between the two types of balls.



The kernel trick actually uses a nifty geometric shortcut to *skip* your table-flipping, transformation step, and just jump straight to the answer. Without delving too deeply into the math behind it since that's covered in-depth in the Dive Deeper section, the kernel is essentially a similarity function. You give it two samples and it lets you know how similar they are. This is pretty important because in SVC, your first priority is finding out which samples of opposite class are nearest to one another, so that you can position your decision boundary optimally. What the kernel function essentially tells you is, given

your current ball positions, how far apart *would the balls have been*, had you flipped the table. This saves you the trouble of cleaning up the balls, and the embarrassment of (possibly) not having the strength to flip the table. In machine learning, some kernels are able to answer the similarity question in an infinite-dimensional space. No machine is capable of flipping that table.