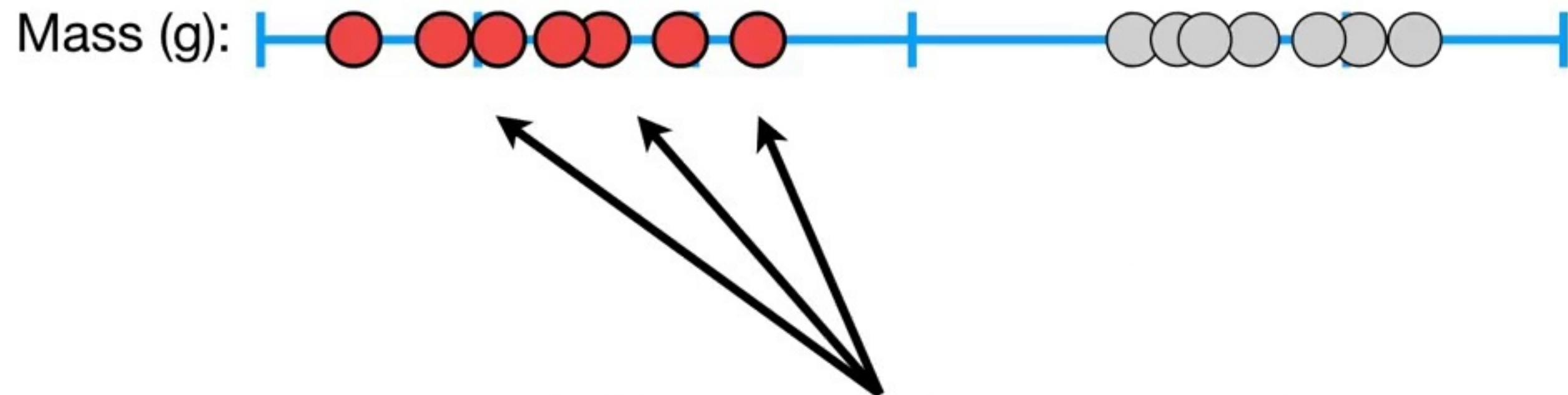
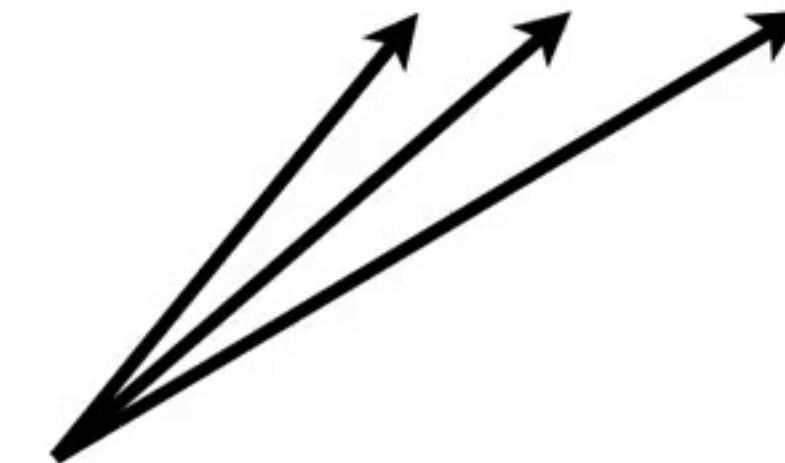
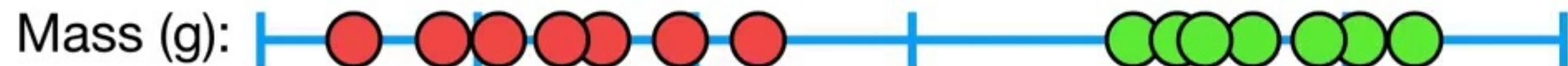


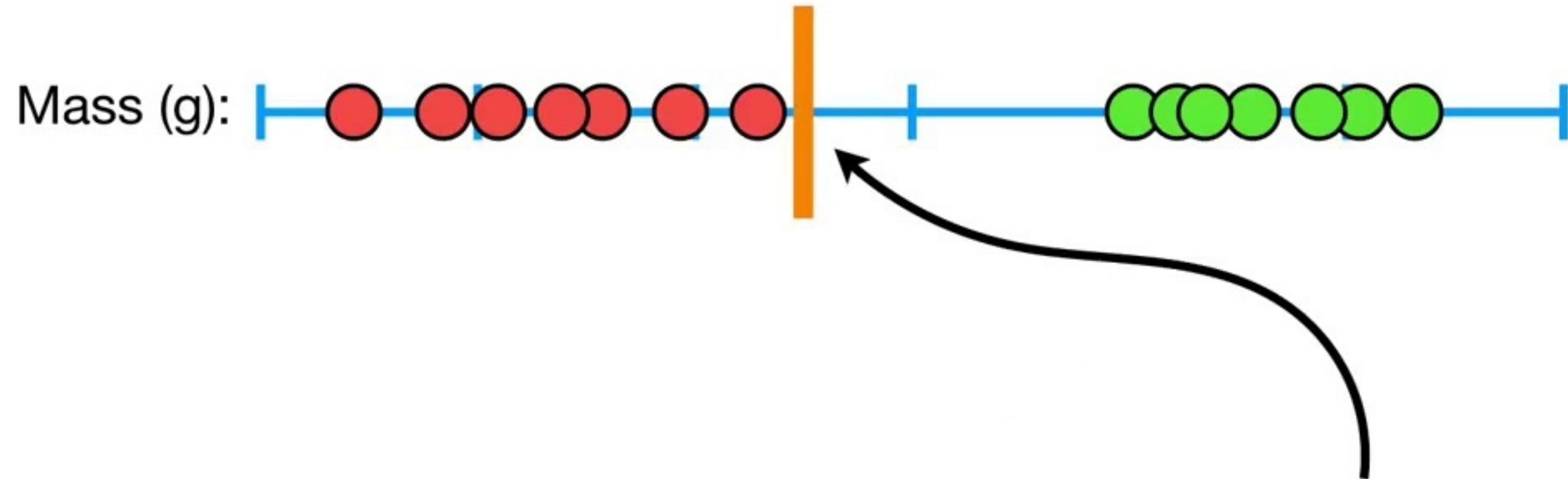
Let's start by imagining we measured
the mass of a bunch of mice...



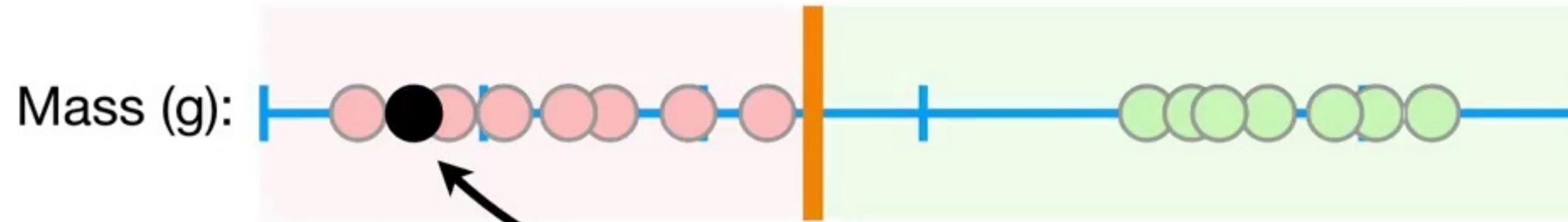
The **red dots** represent mice are *not obese*...



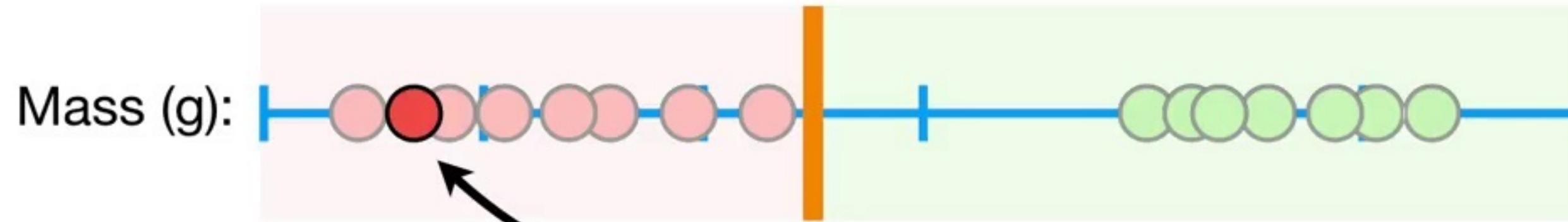
...and the **green dots** represent mice are **obese**.



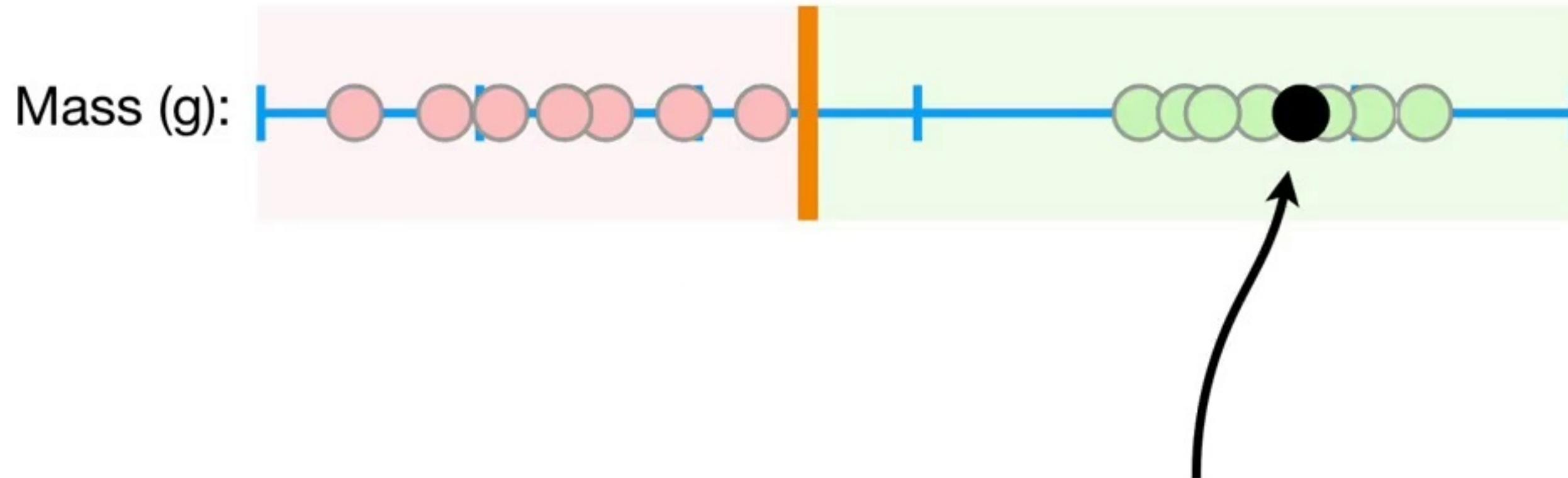
Based on these observations, we can pick a threshold...



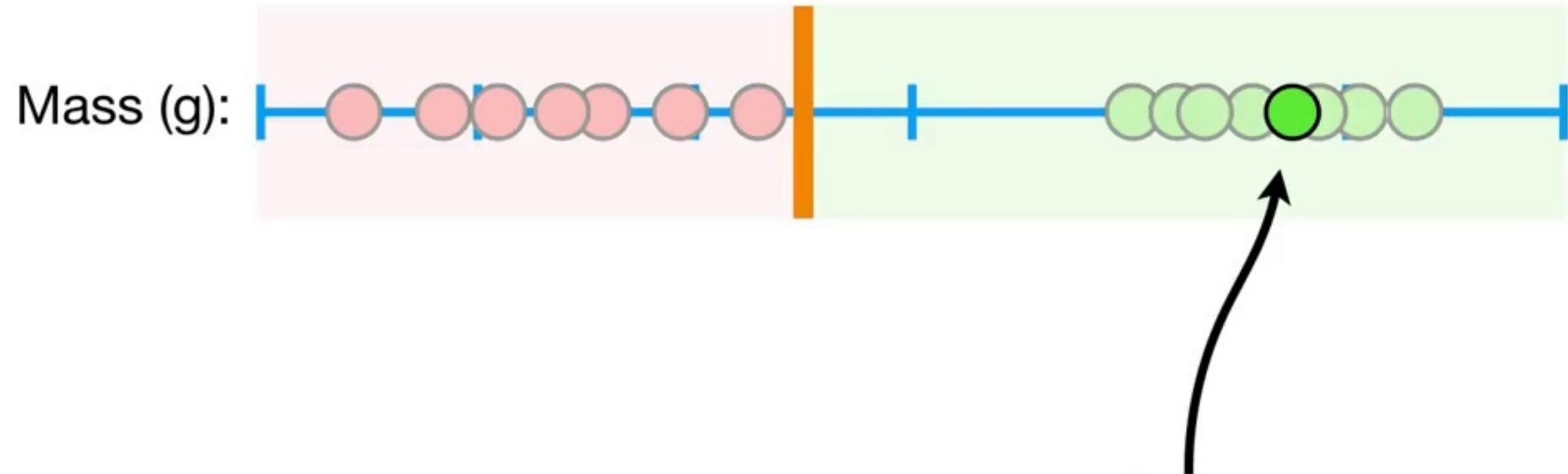
...and when we get a new observation that
has less mass than the threshold...



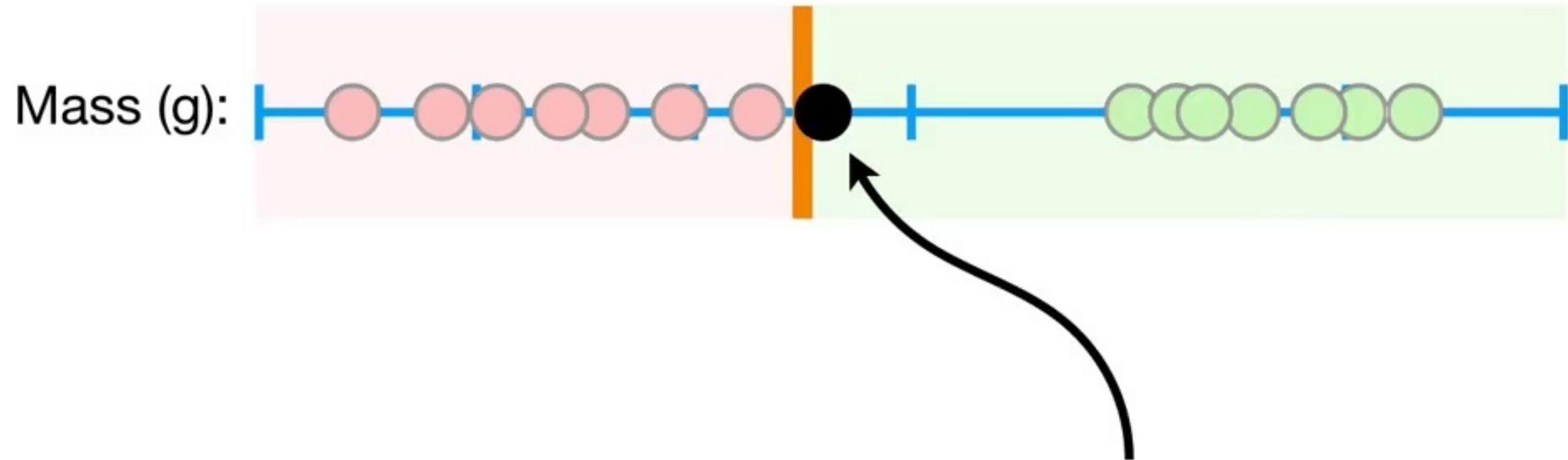
...we can classify it as ***not obese***.



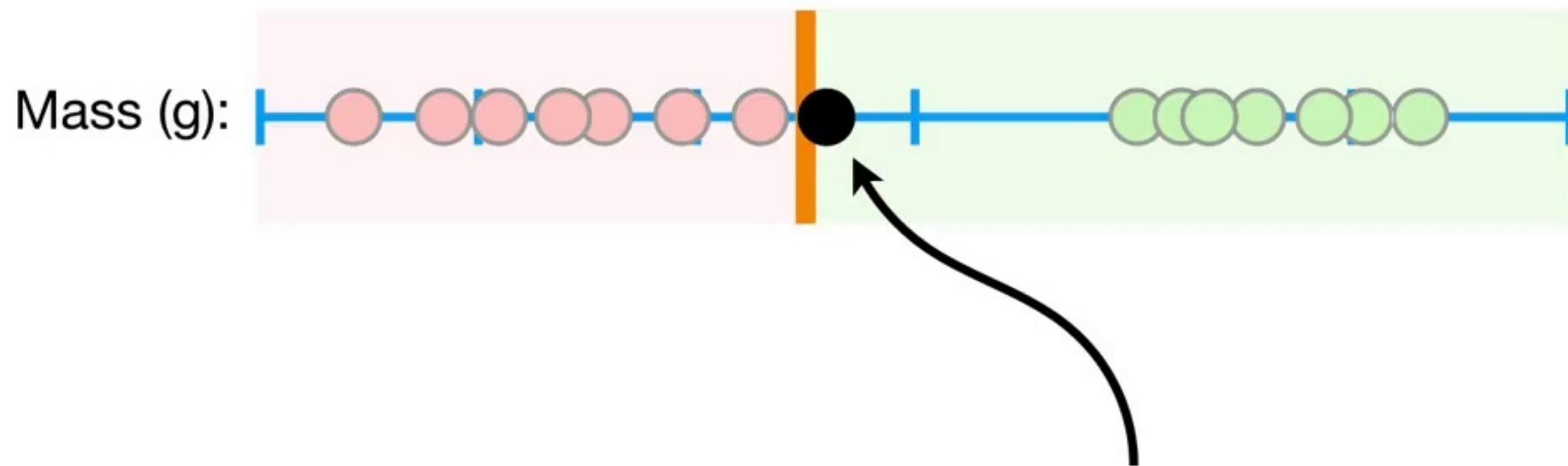
And when we get a new observation with more mass
than the threshold...



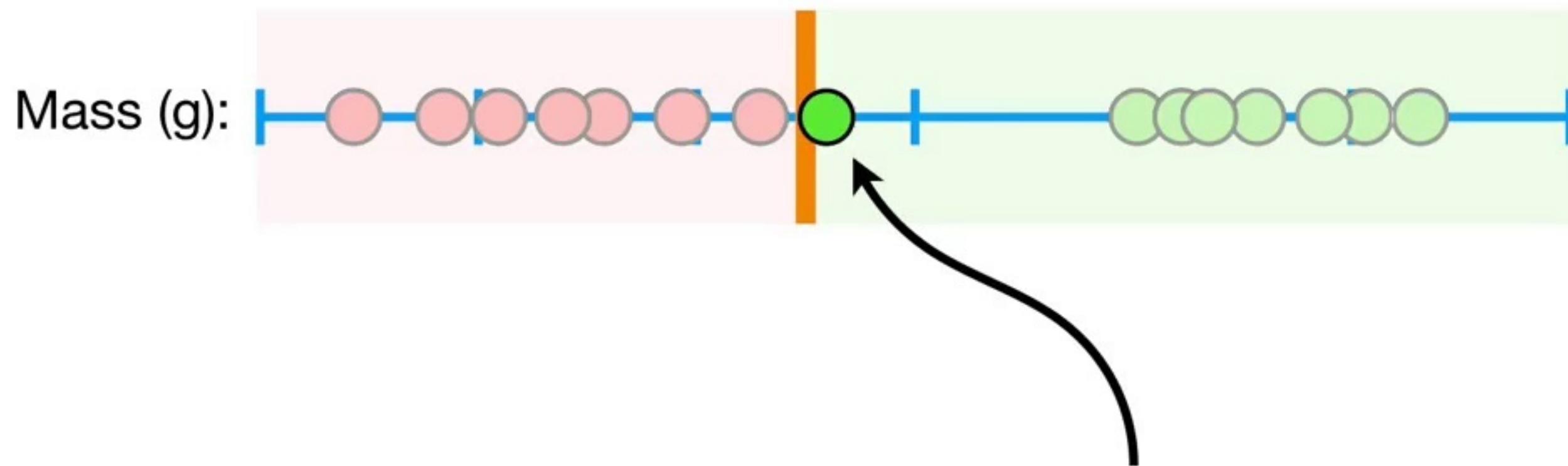
...we can classify it as ***obese***.



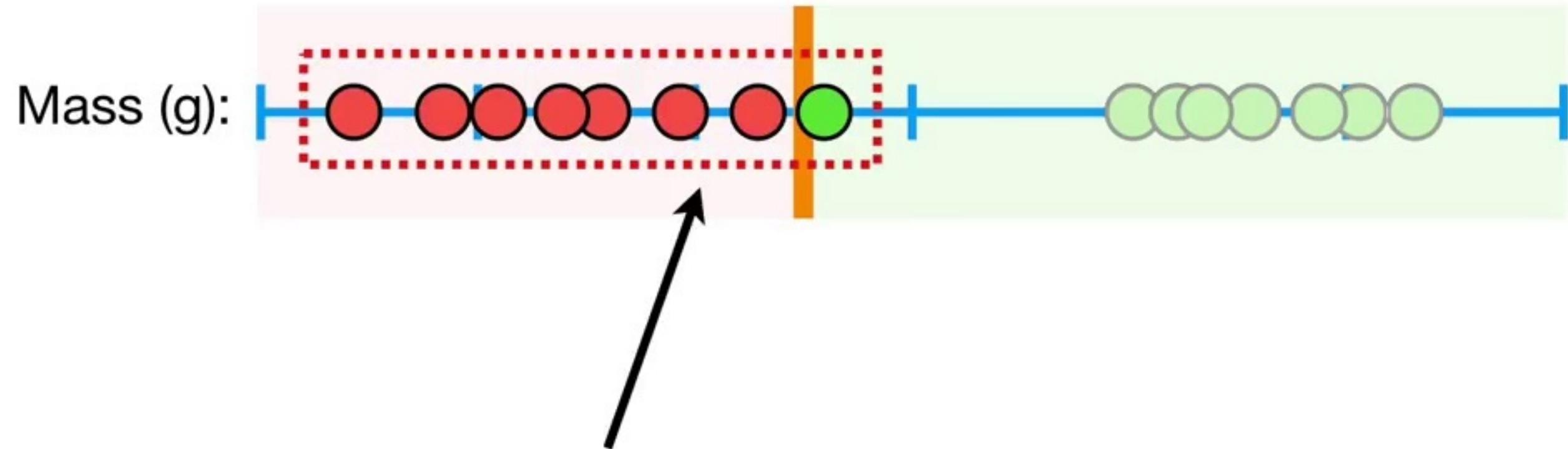
However, what if get a new observation here?



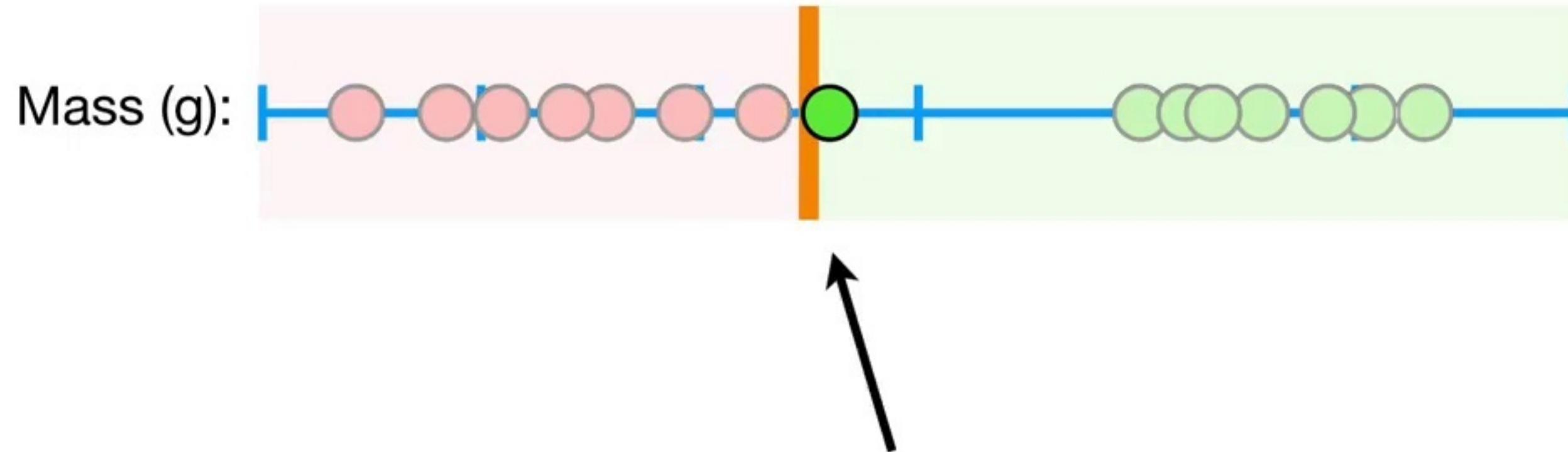
Because this observation has more mass than the threshold, we classify it as **obese**.



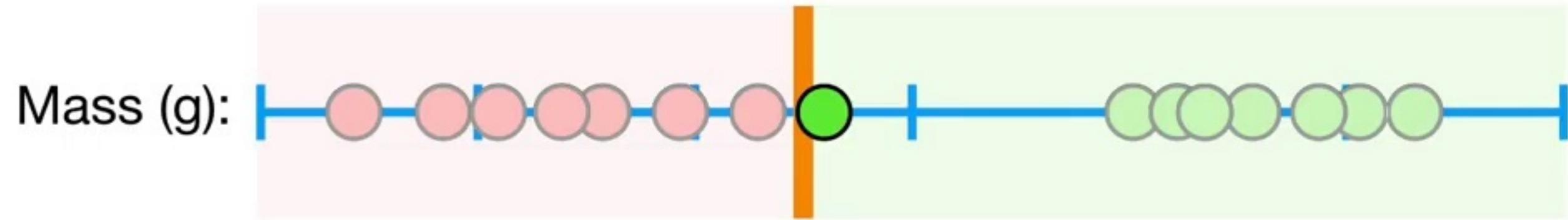
Because this observation has more mass than the threshold, we classify it as **obese**.



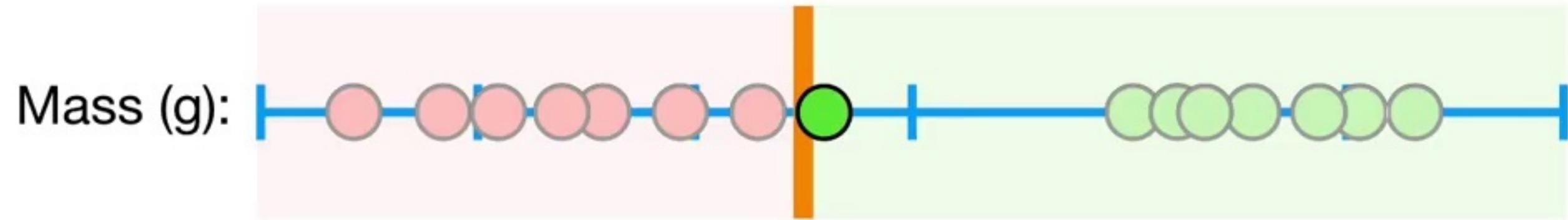
But that doesn't make sense, because it is much closer to the observations that are ***not obese***.



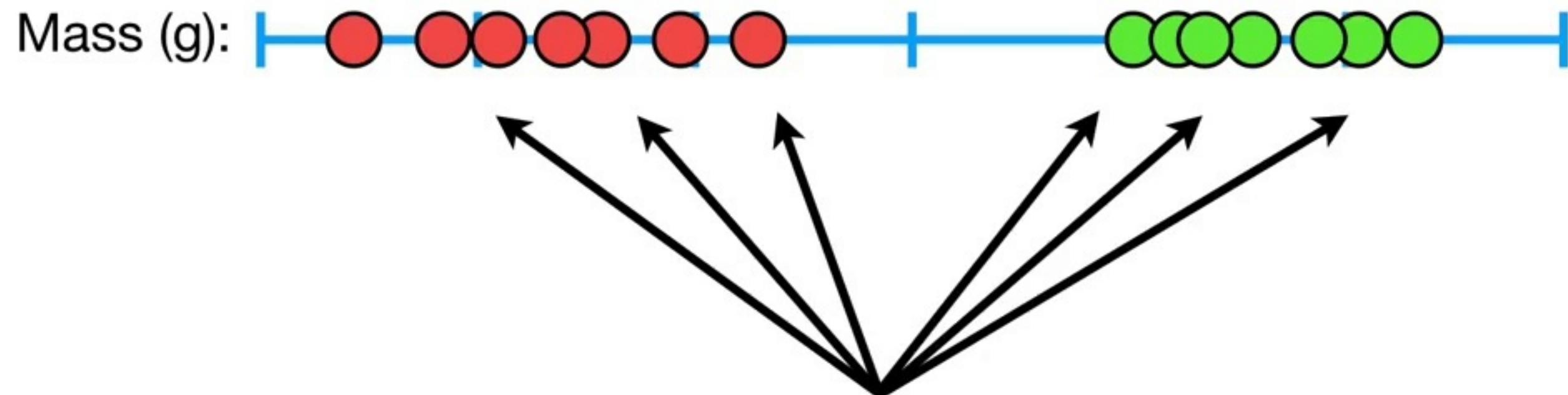
So this threshold is pretty lame.



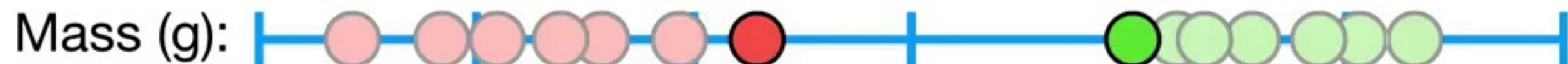
Can we do better?



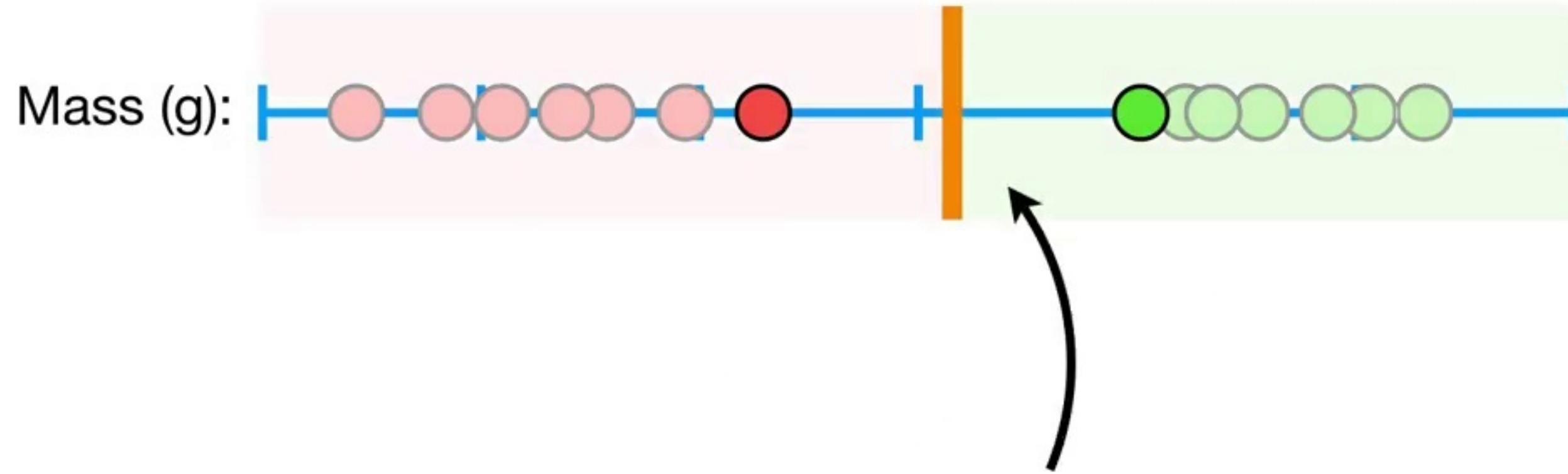
Yes!



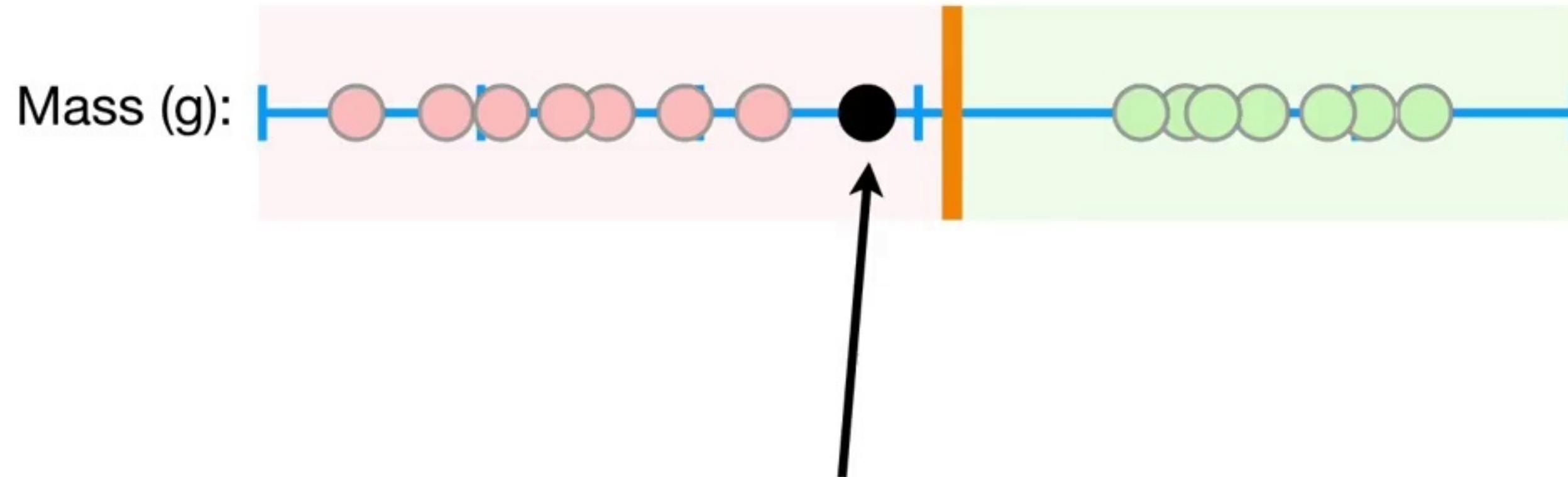
Going back to the original training dataset...



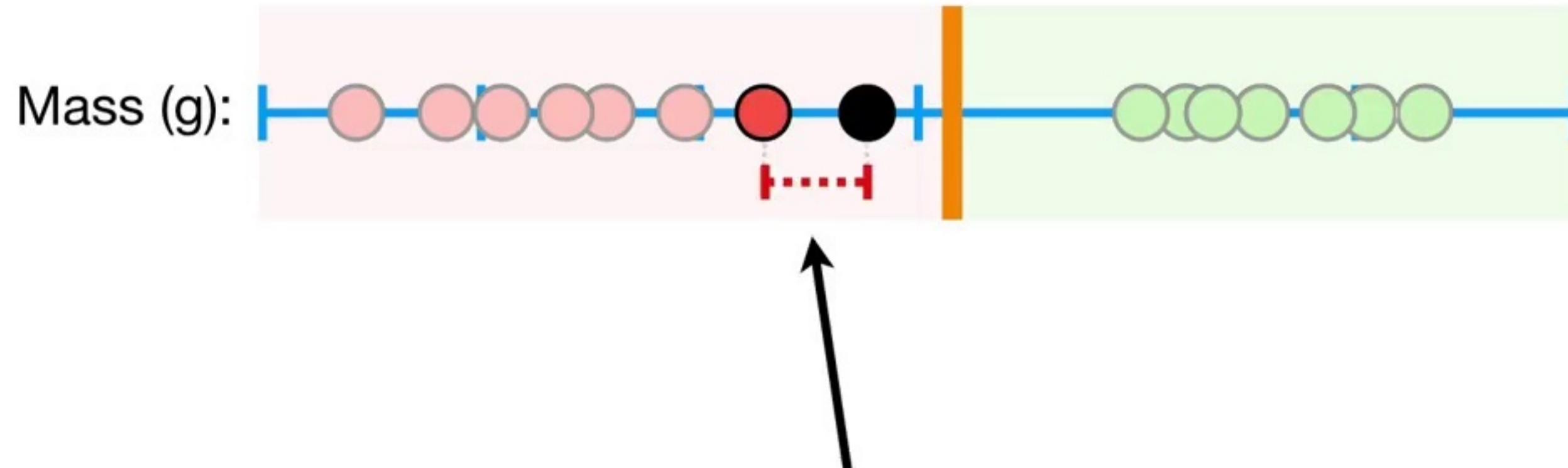
...we can focus on the observations on
the edges of each cluster...



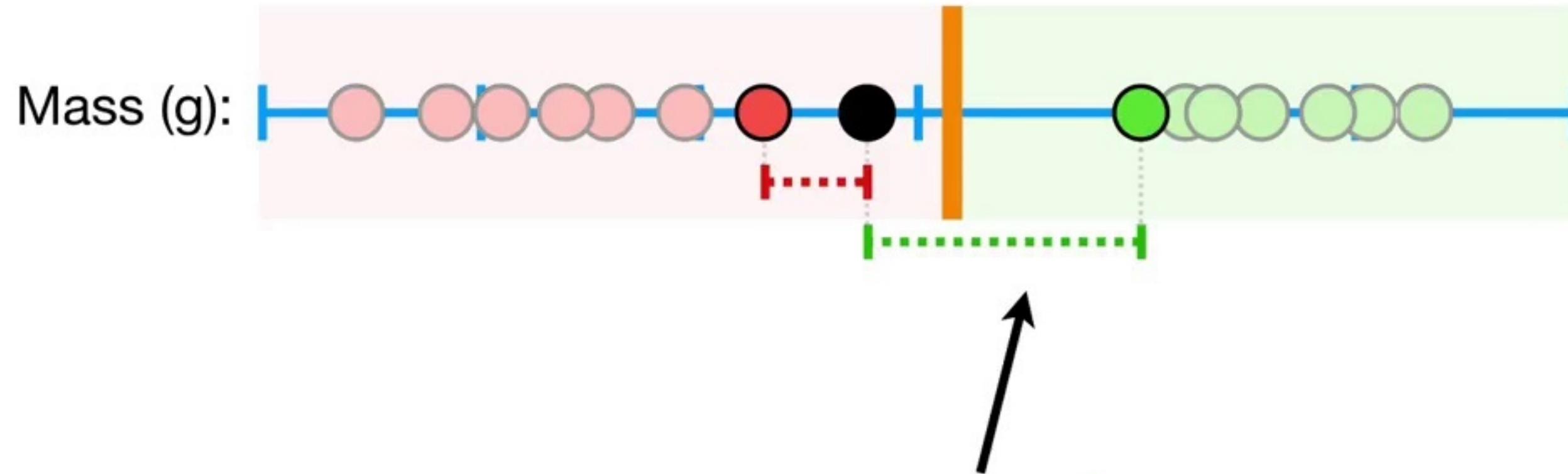
...and use the midpoint between
them as the threshold.



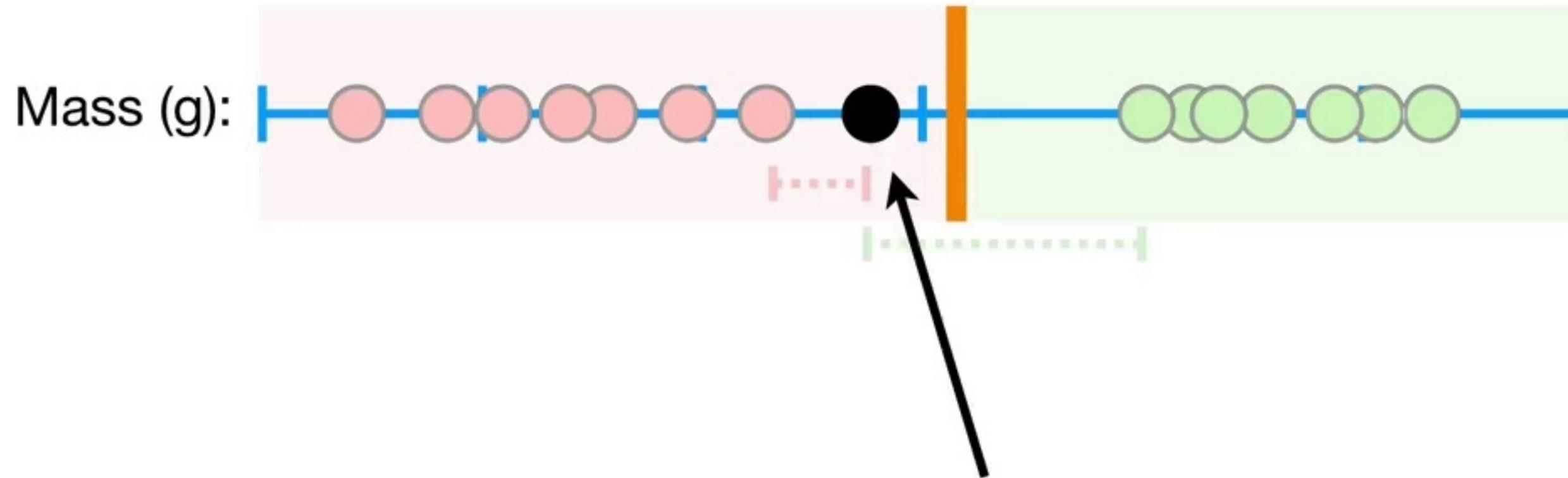
Now, when a new observation falls
on the left side of the threshold...



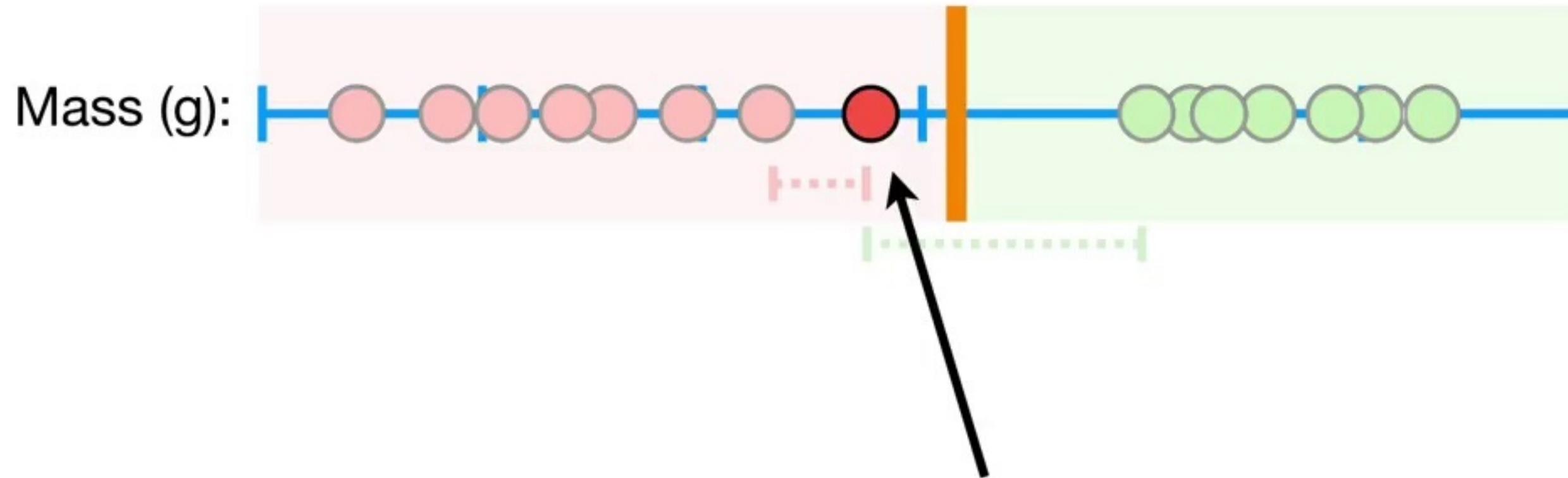
...it will be closer to the
observations that are ***not obese***...



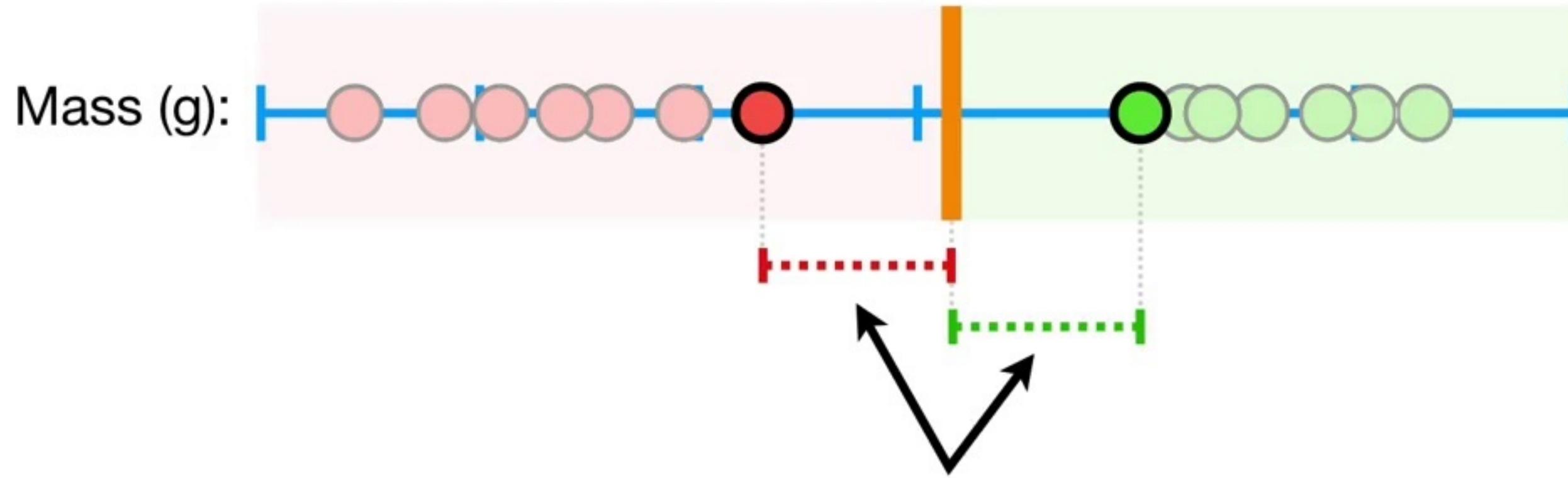
...than it is to the *obese*
observations.



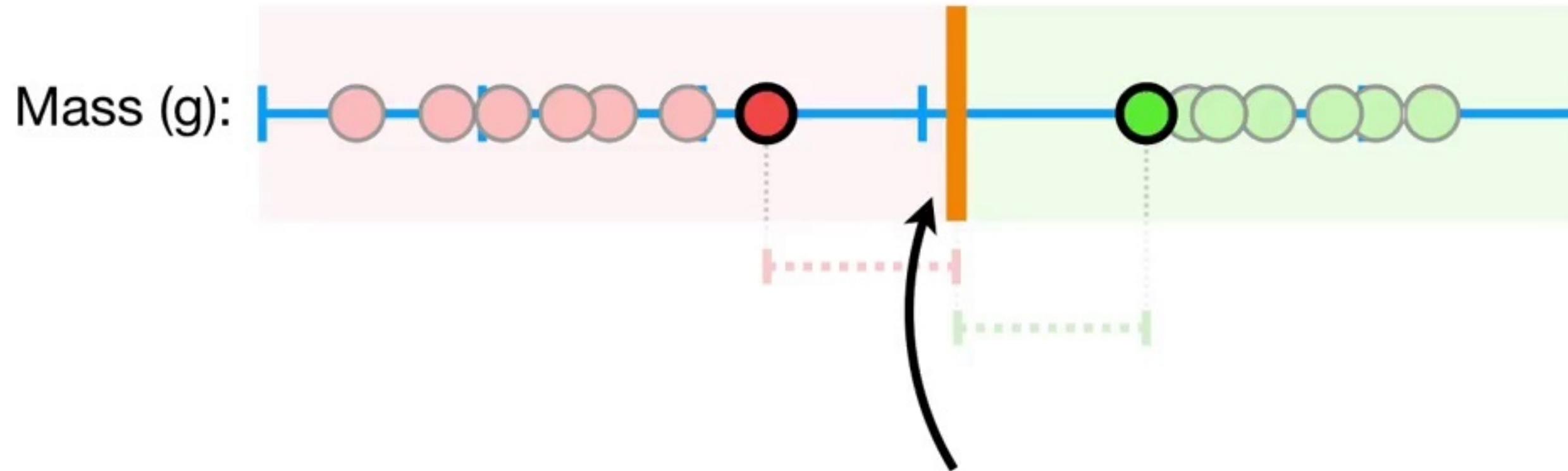
So it makes sense to classify this new observation as ***not obese***.



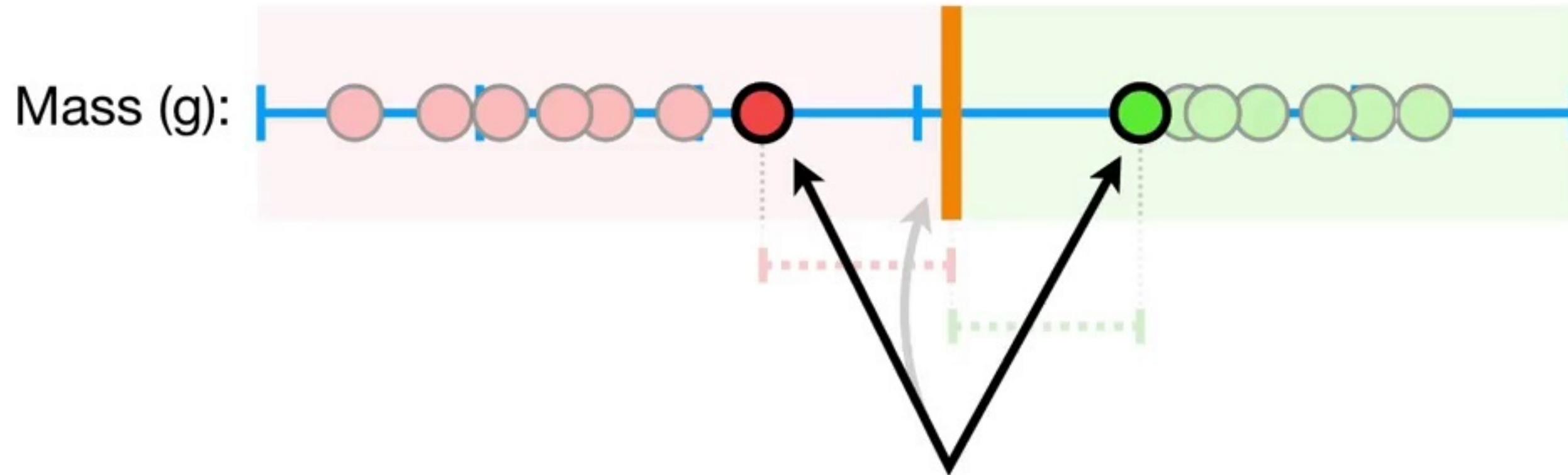
So it makes sense to classify this new observation as ***not obese***.



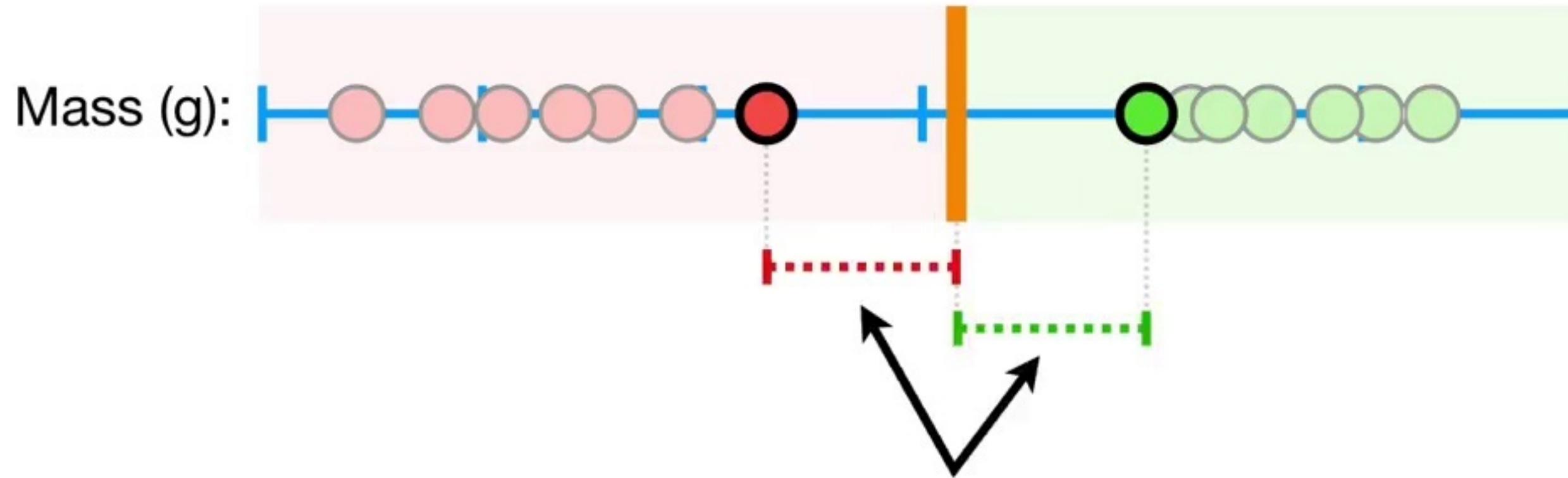
The shortest distance between
the observations and the
threshold is called the **margin**.



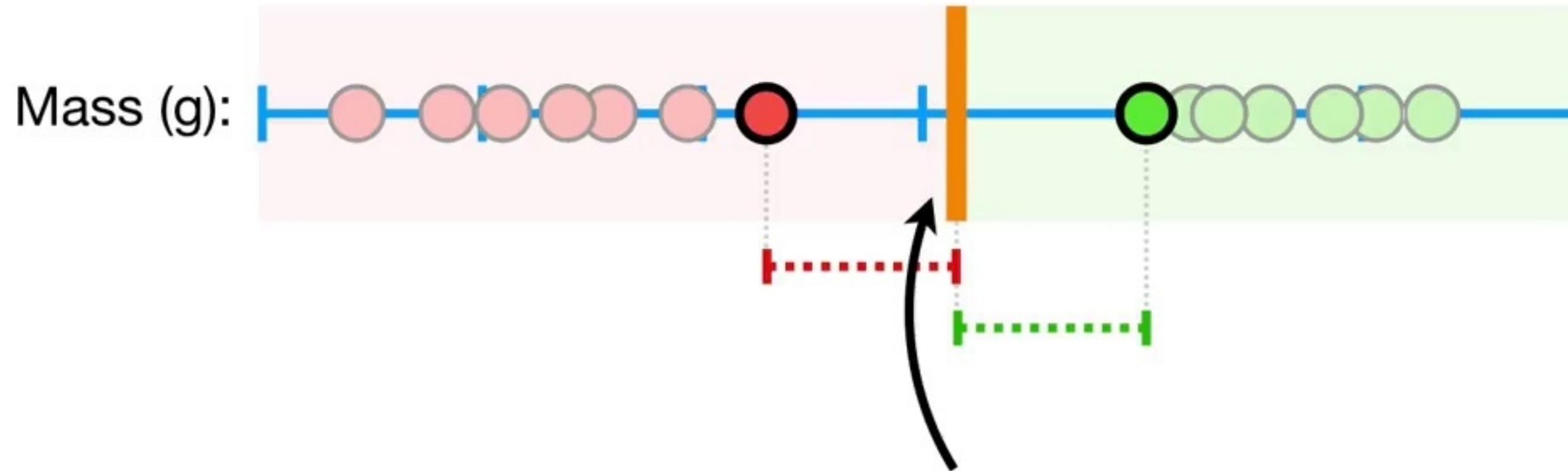
Since we put the threshold
halfway between these two
observations...



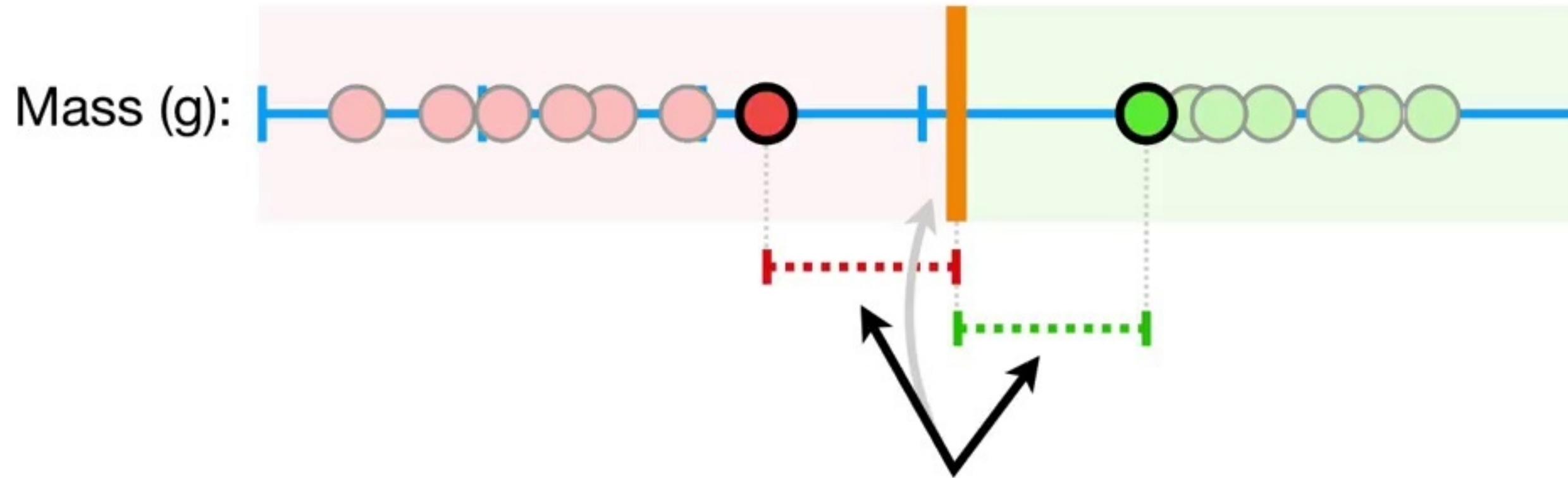
Since we put the threshold
halfway between these two
observations...



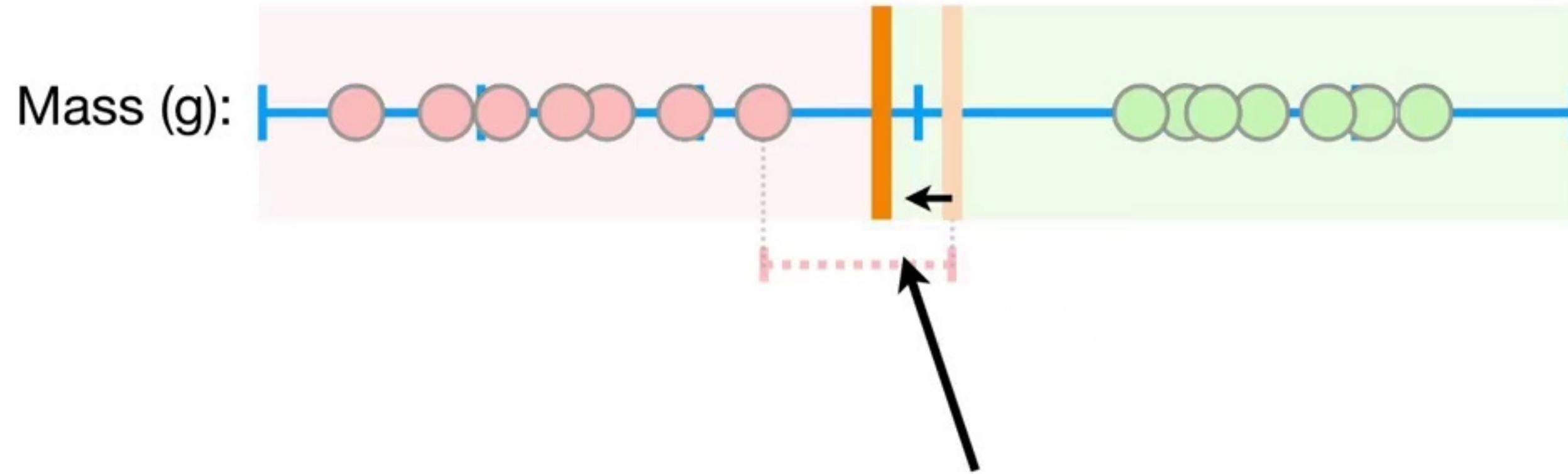
...the distances between the observations and the threshold are the same and both reflect the **margin**.



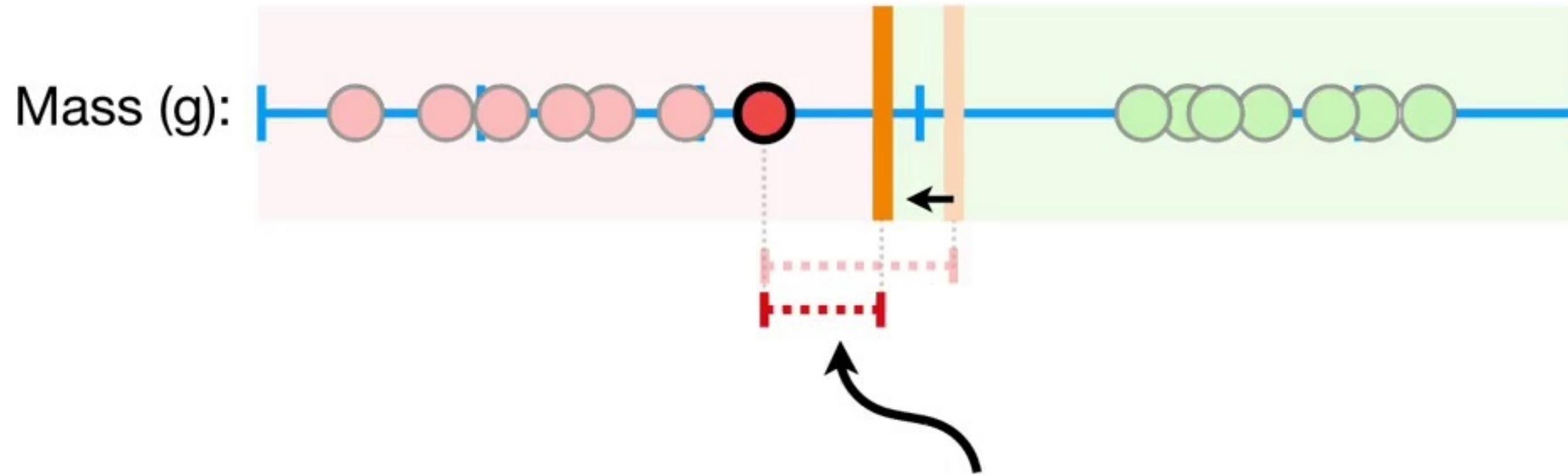
When the threshold is halfway
between the two observations, the
margin is as large as it can be.



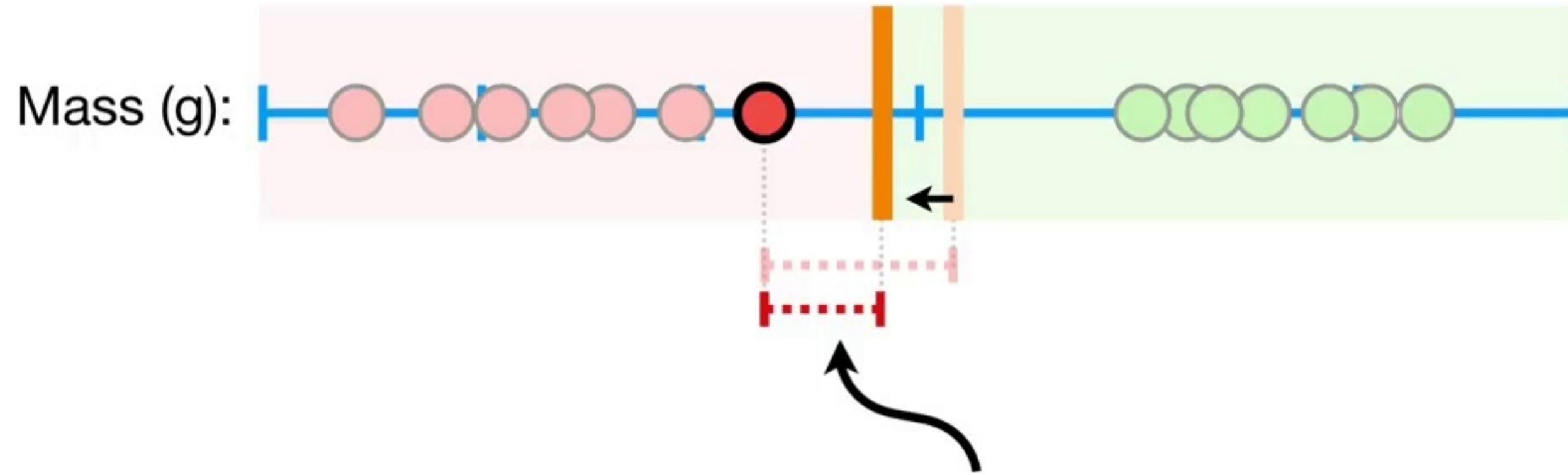
When the threshold is halfway
between the two observations, the
margin is as large as it can be.



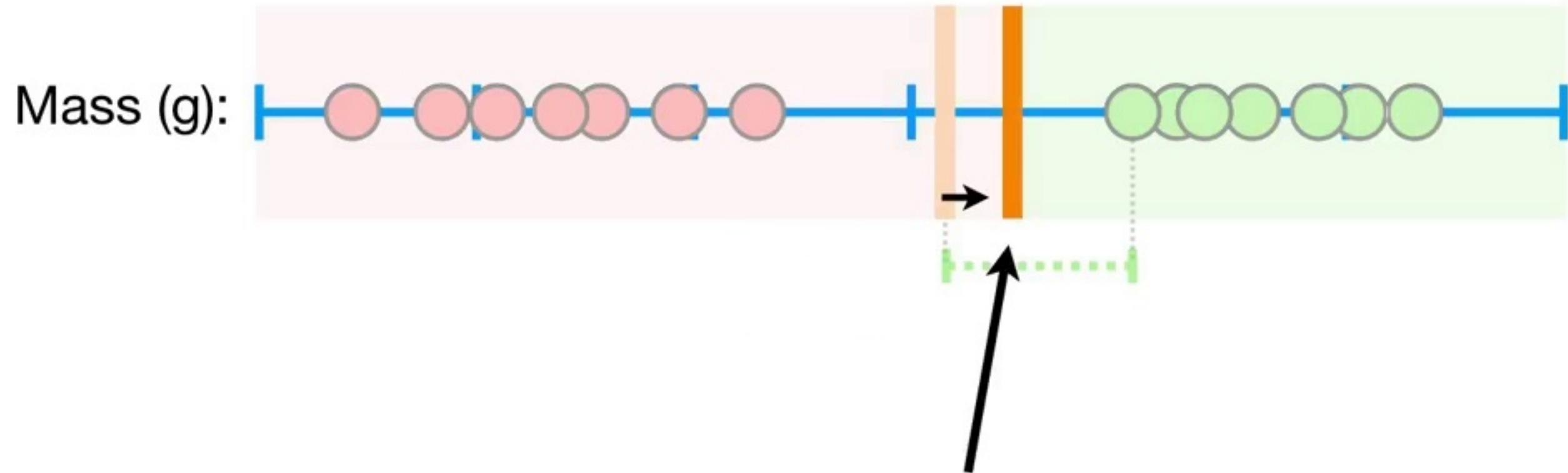
For example, if we moved the threshold to the left a little bit...



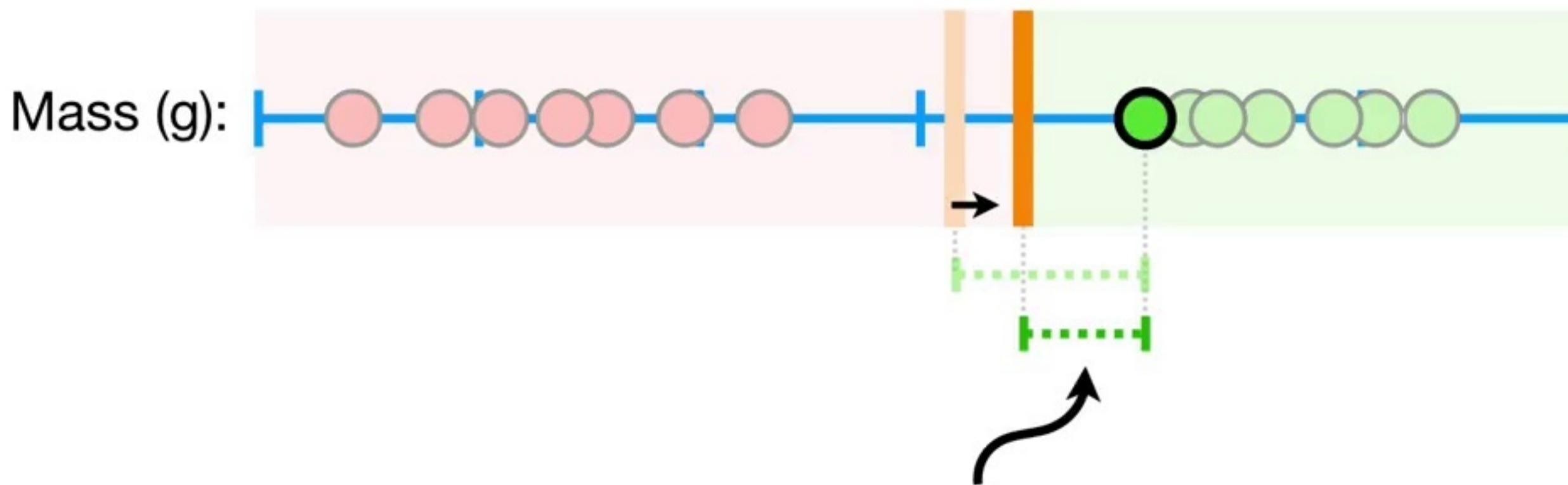
...then the distance between the threshold and the observation
that is **not obese** would be
smaller...



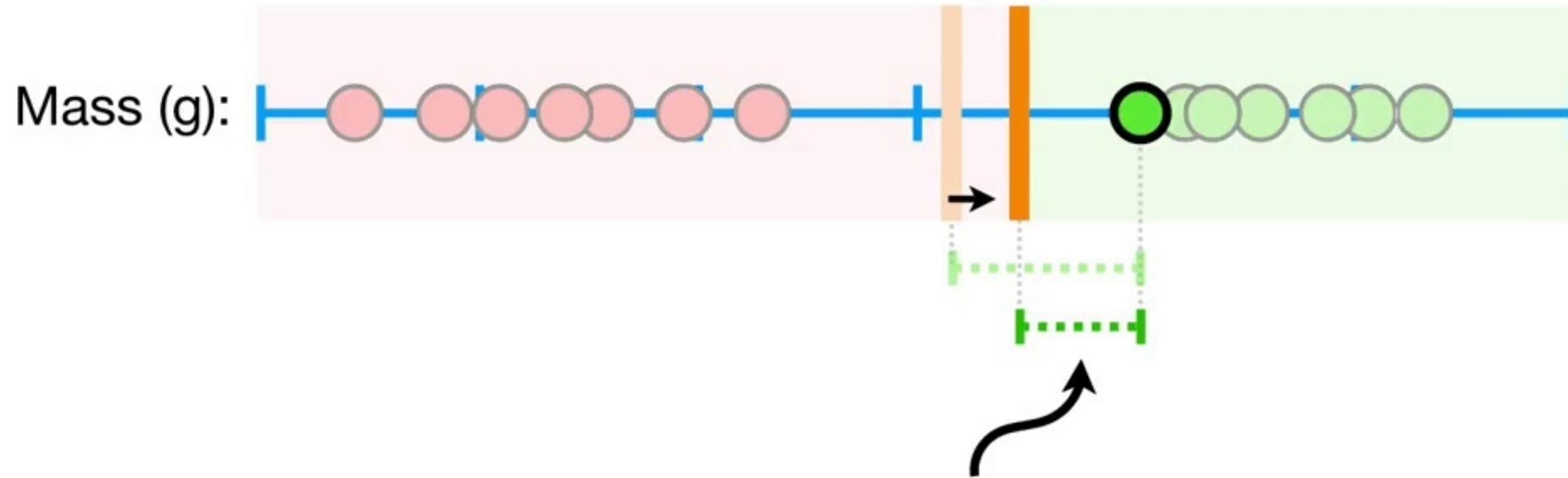
...and thus, the **margin**
would be smaller than it
was before.



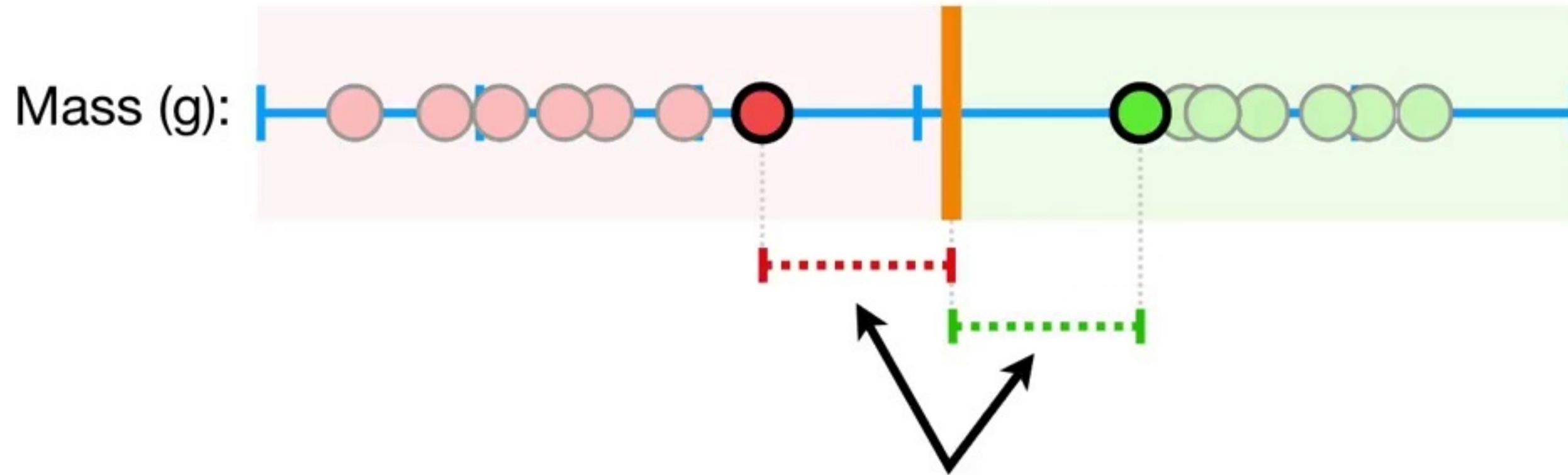
And if we moved the threshold to
the right a little bit...



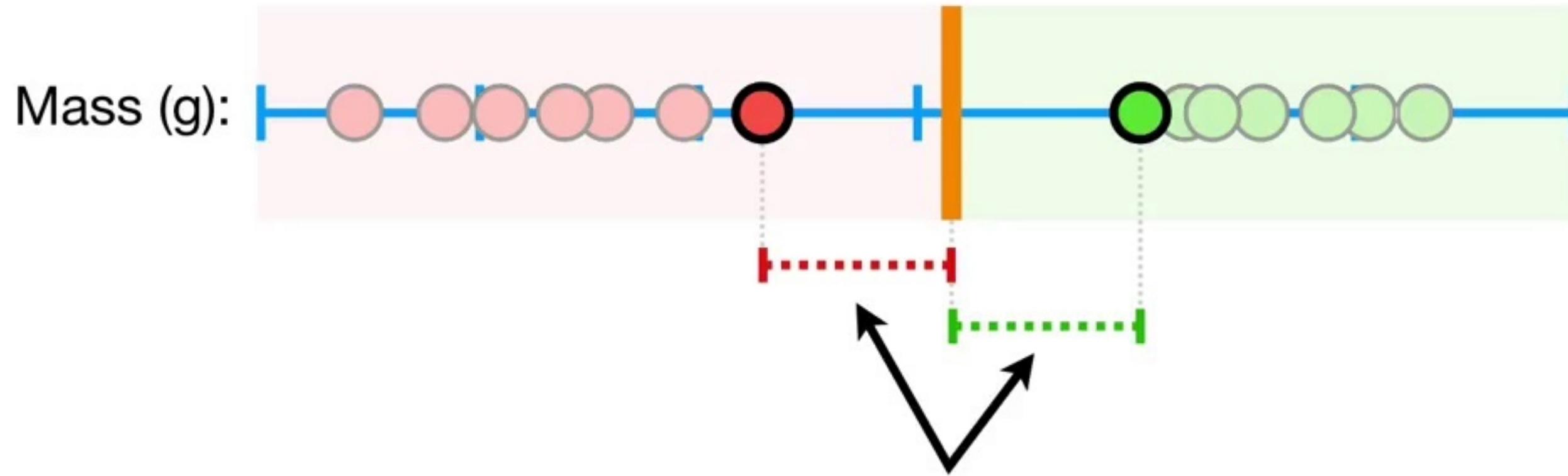
...then the distance between the
obese observation and the
threshold would get smaller...



...and again, the **margin**
would be smaller.

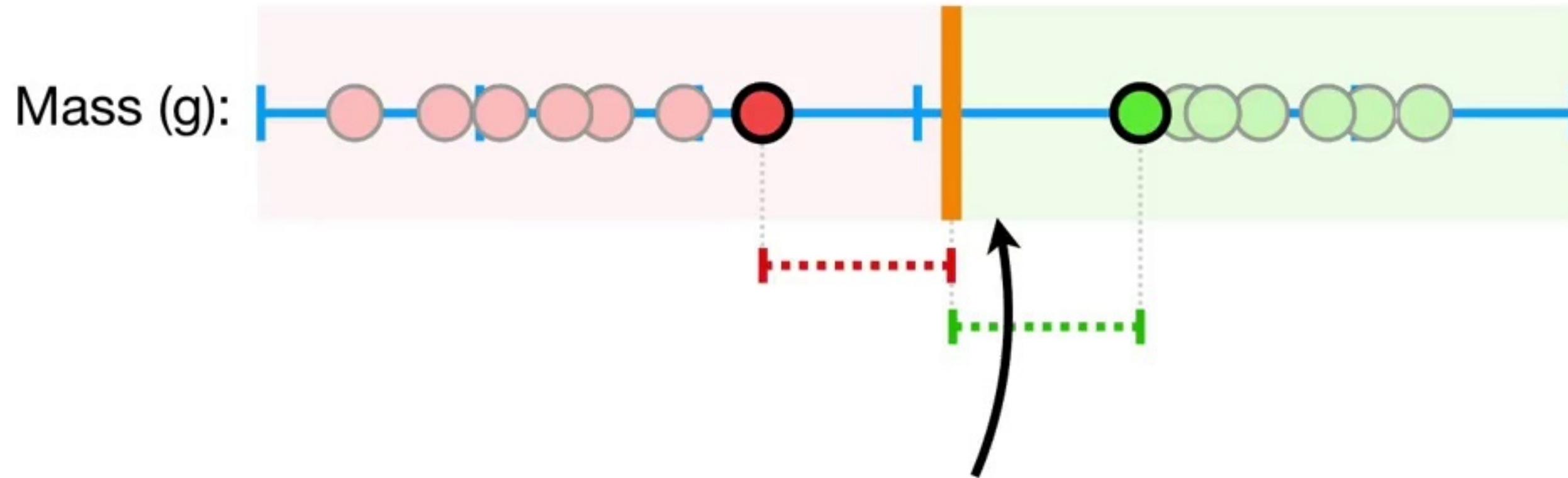


When we use the threshold that gives us the largest **margin** to make classifications...

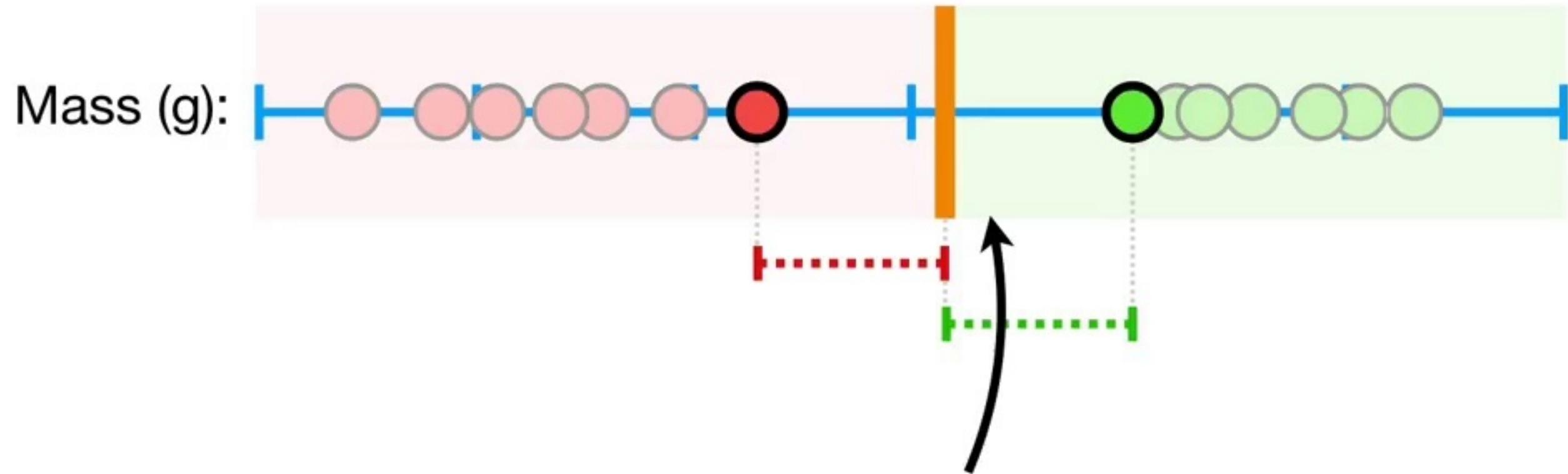


When we use the threshold that gives us the largest **margin** to make classifications...

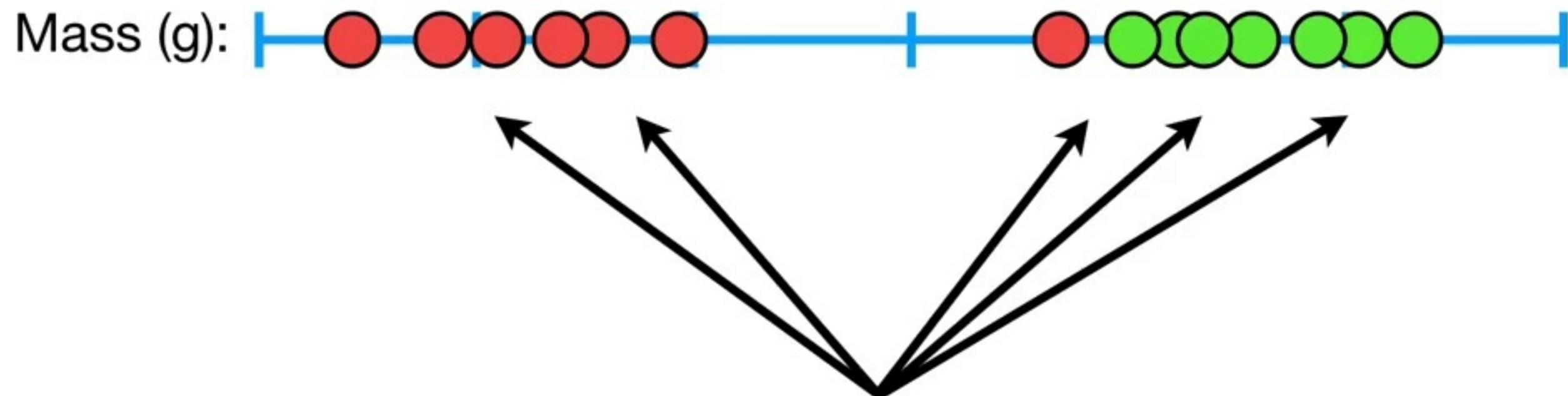
(terminology alert!)



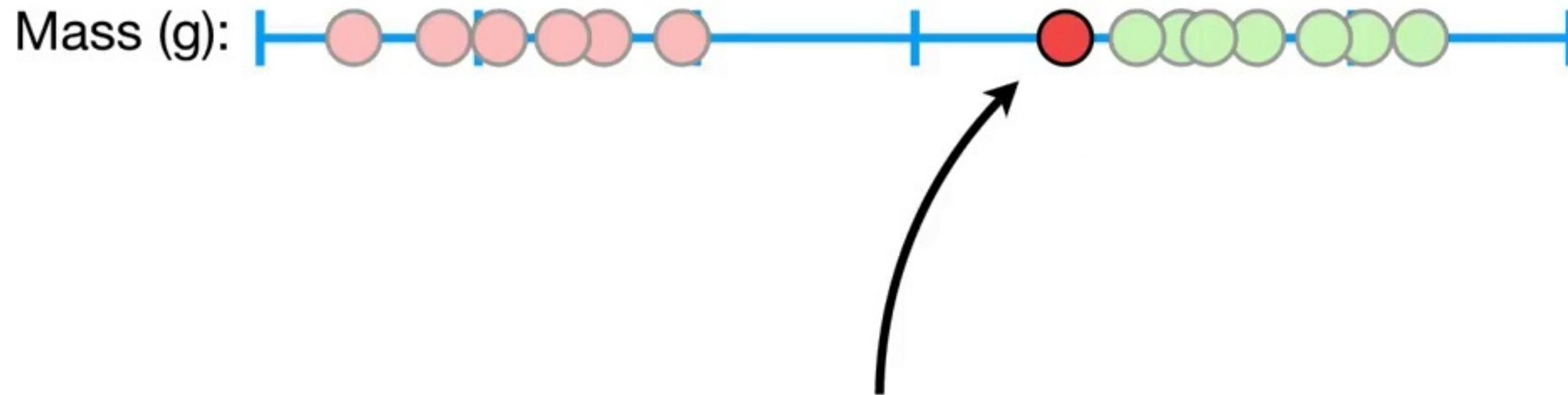
...we are using a
Maximal Margin Classifier.



Maximal Margin Classifiers
seem pretty cool...

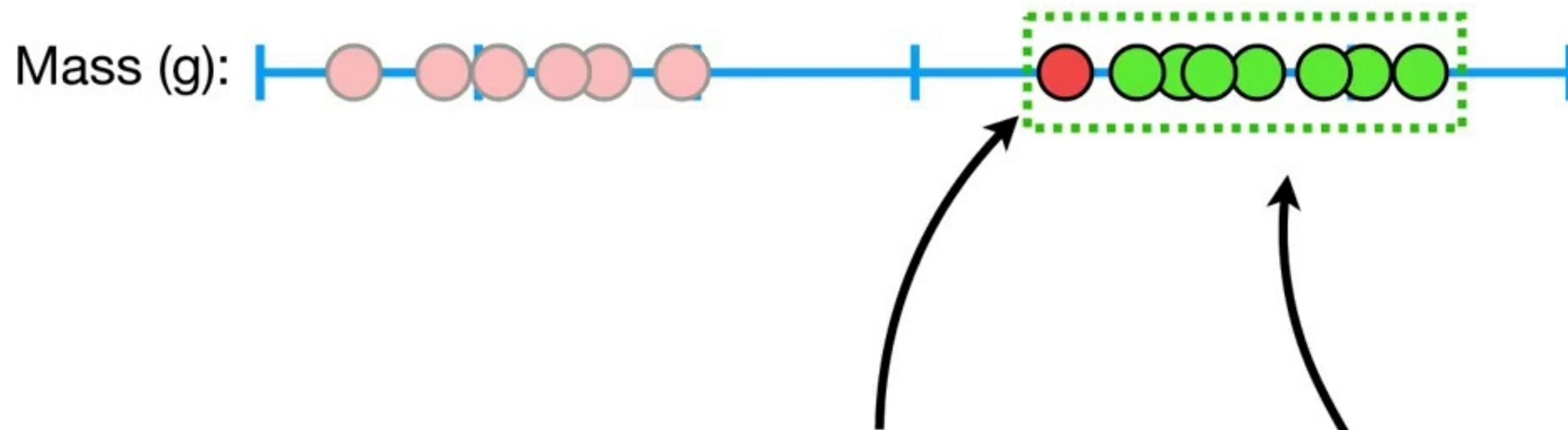


...but what if our training data
looked like this....

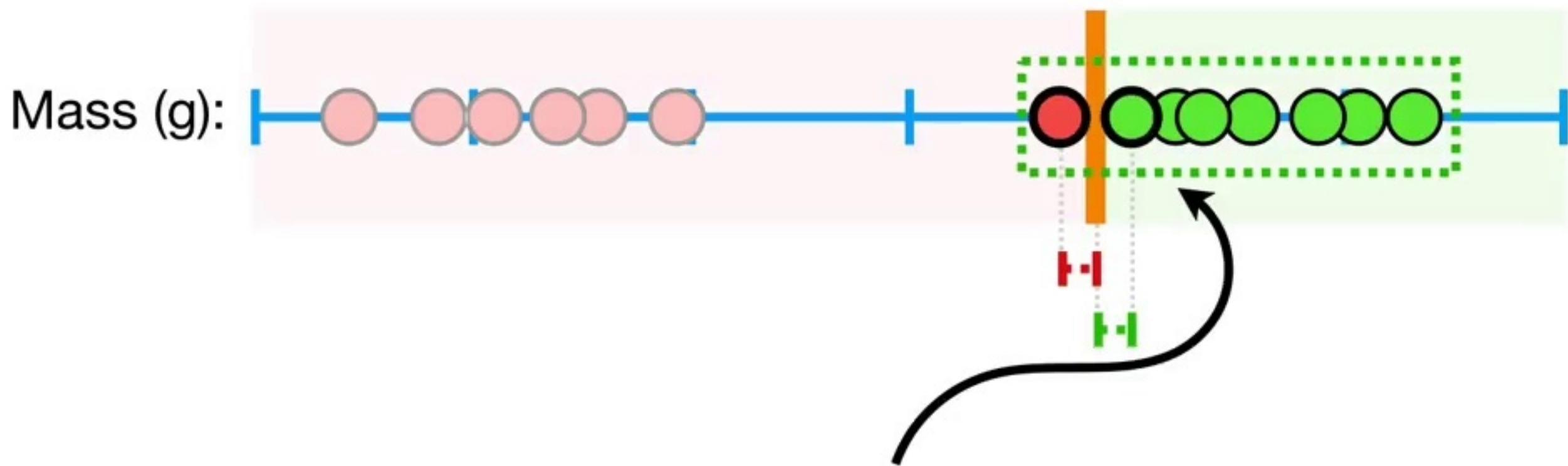


...and we had an outlier
observation that was classified as
not obese, but was much closer
to the ***obese*** observations.

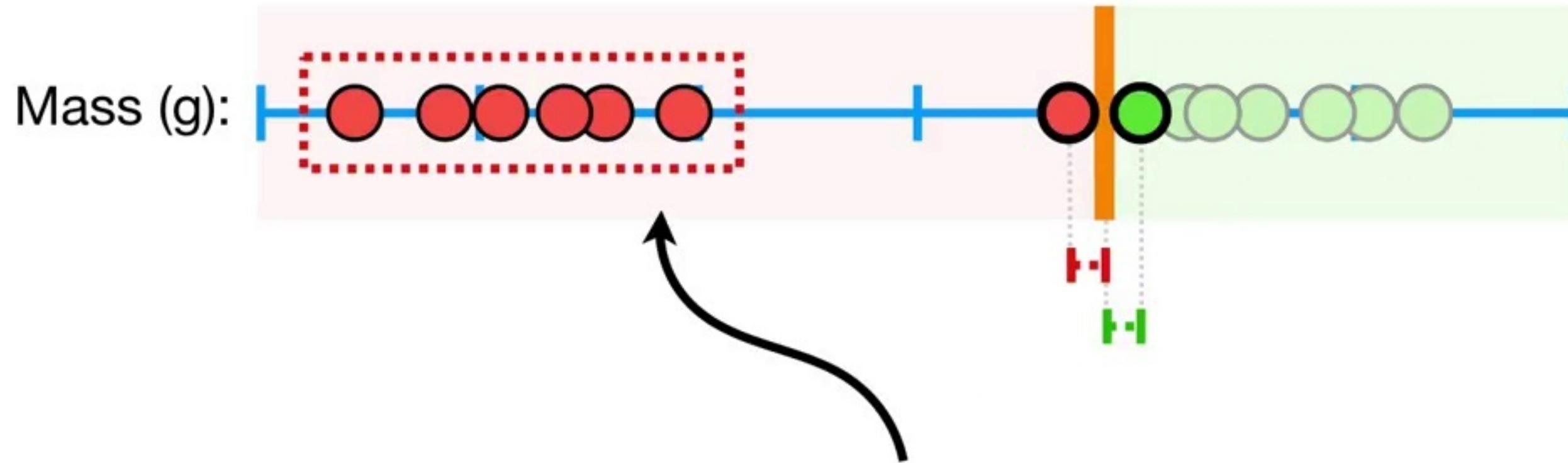
A black curved arrow points from the text "...and we had an outlier" towards the red circle on the number line.



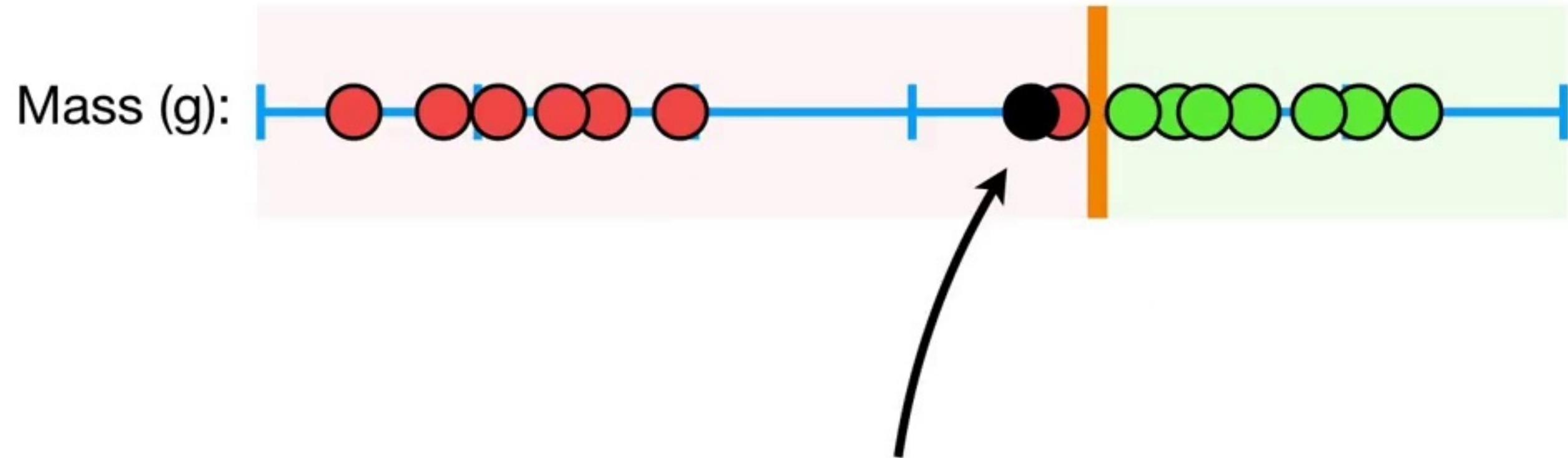
...and we had an outlier
observation that was classified as
not obese, but was much closer
to the ***obese*** observations.



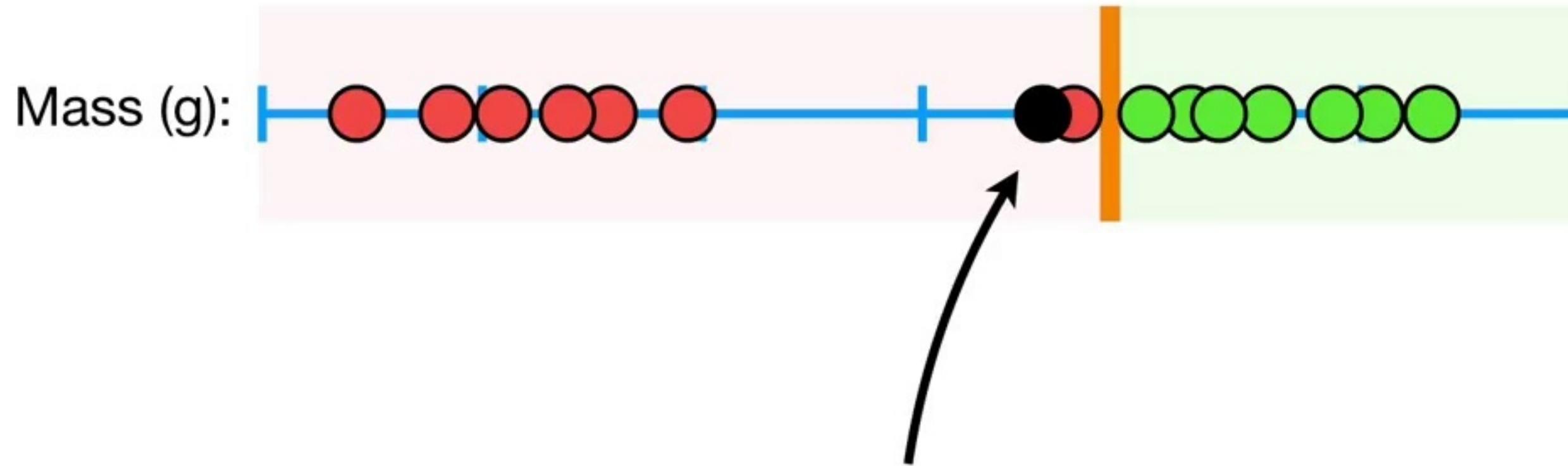
In this case, the **Maximum Margin Classifier** would be super close to the *obese* observations...



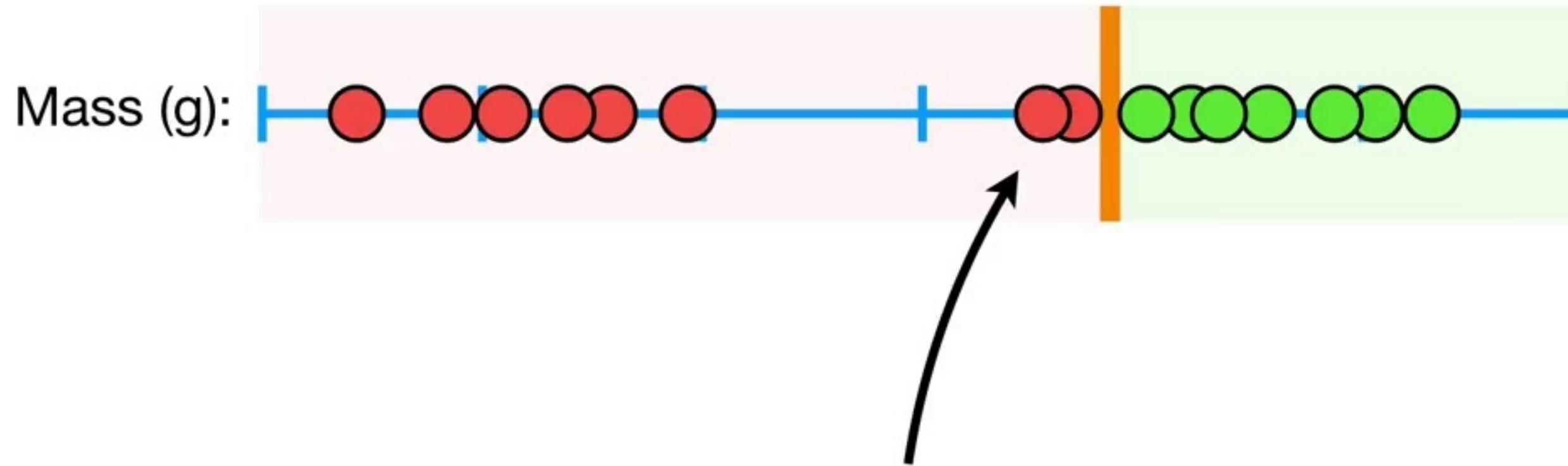
...and really far from the majority
of the observations that are **not**
obese.



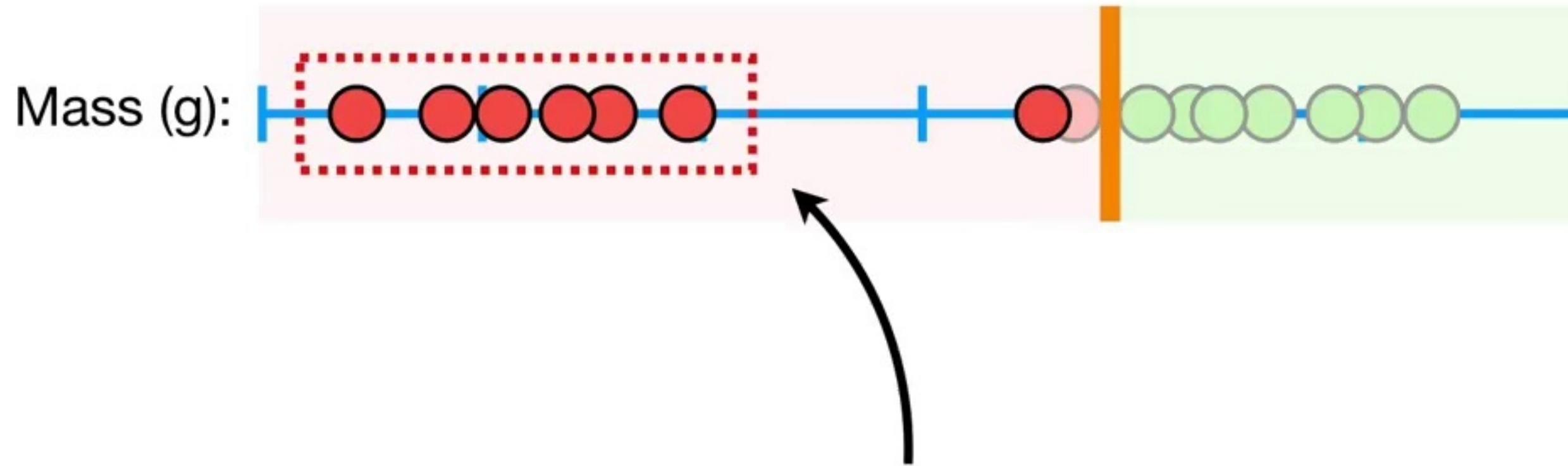
Now, if we got this new
observation...



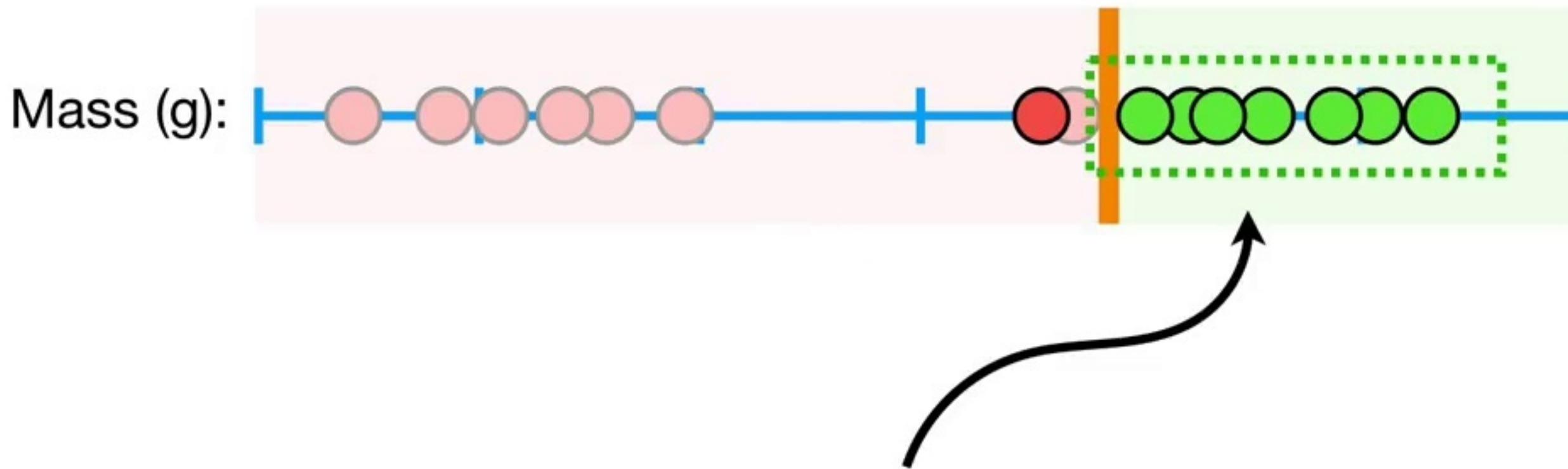
...we would classify it as ***not obese***, even though most of the ***not obese*** observations are much further away than the ***obese*** observations.



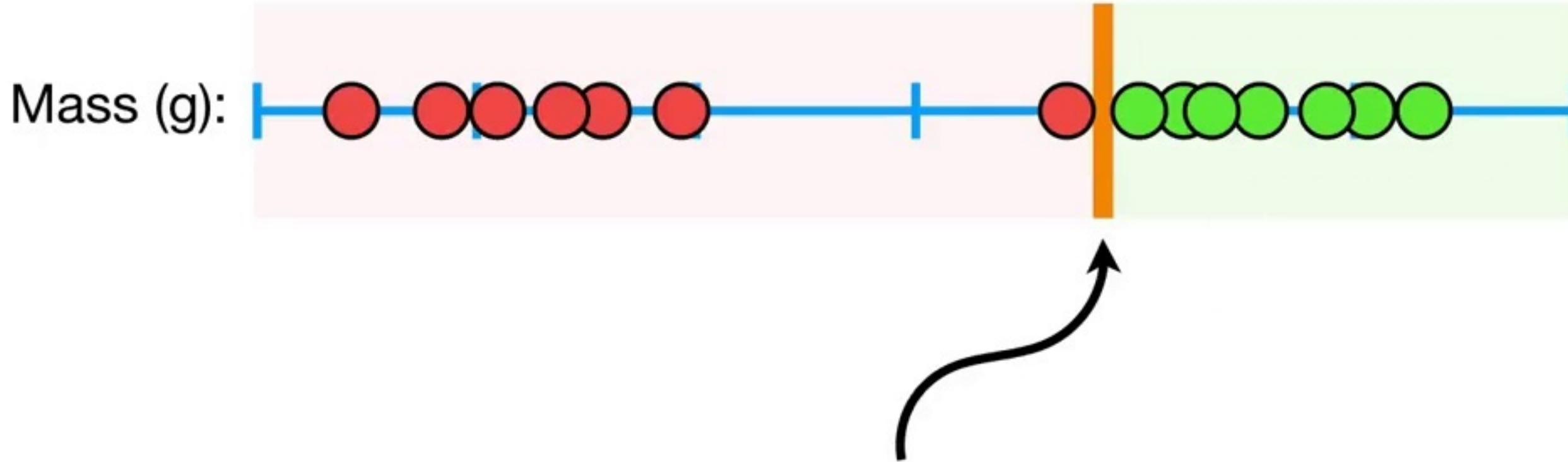
...we would classify it as ***not obese***, even though most of the ***not obese*** observations are much further away than the ***obese*** observations.



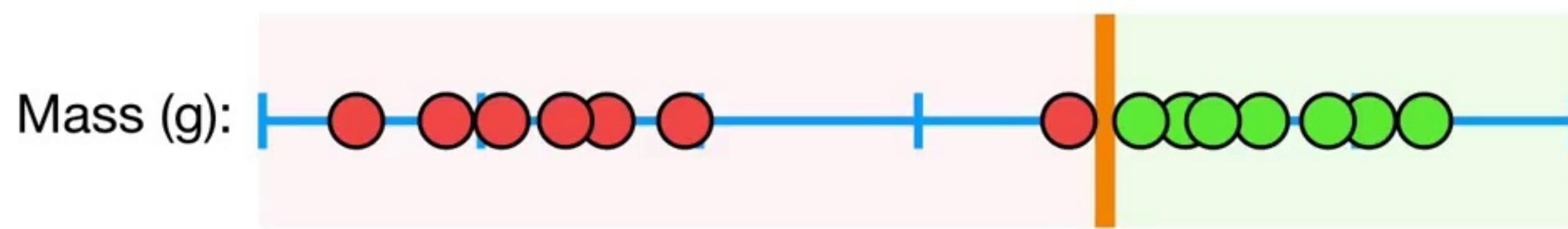
...we would classify it as ***not obese***, even though most of the ***not obese*** observations are much further away than the ***obese*** observations.



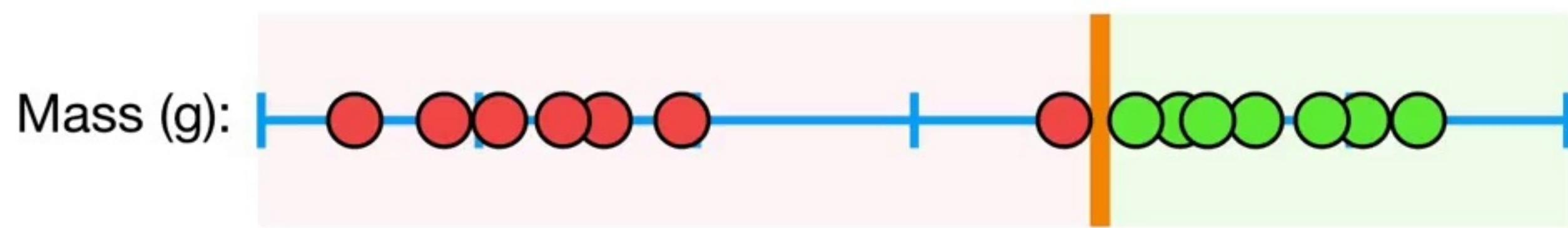
...we would classify it as **not obese**, even though most of the **not obese** observations are much further away than the **obese** observations.



So Maximal Margin Classifiers
are *super sensitive to outliers* in the
training data and that makes them
pretty lame.



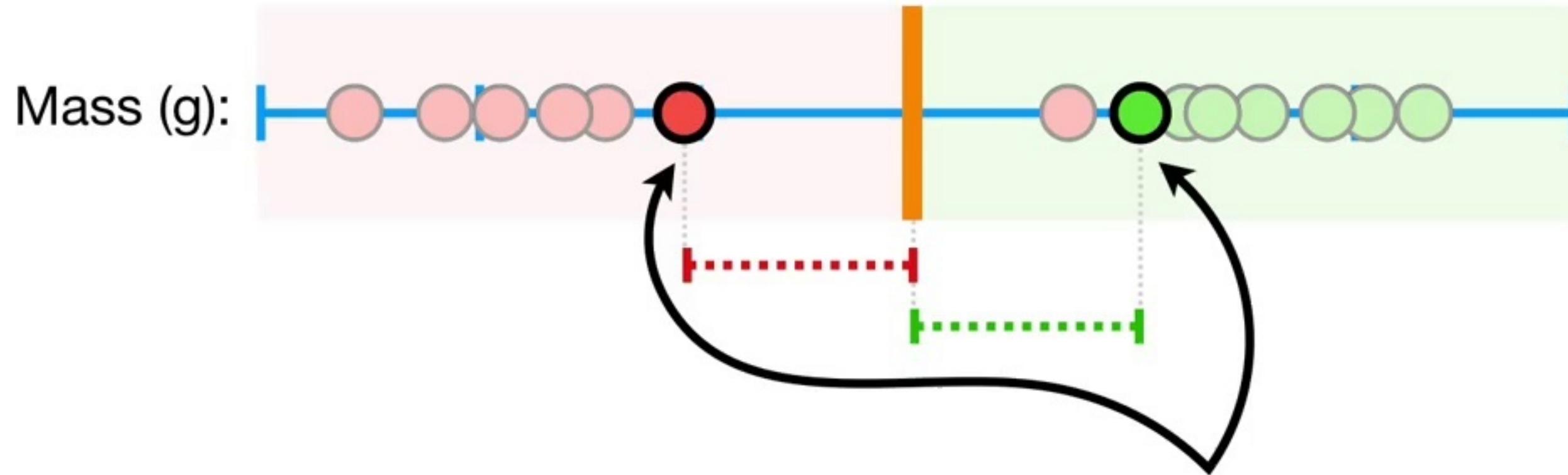
Can we do better?



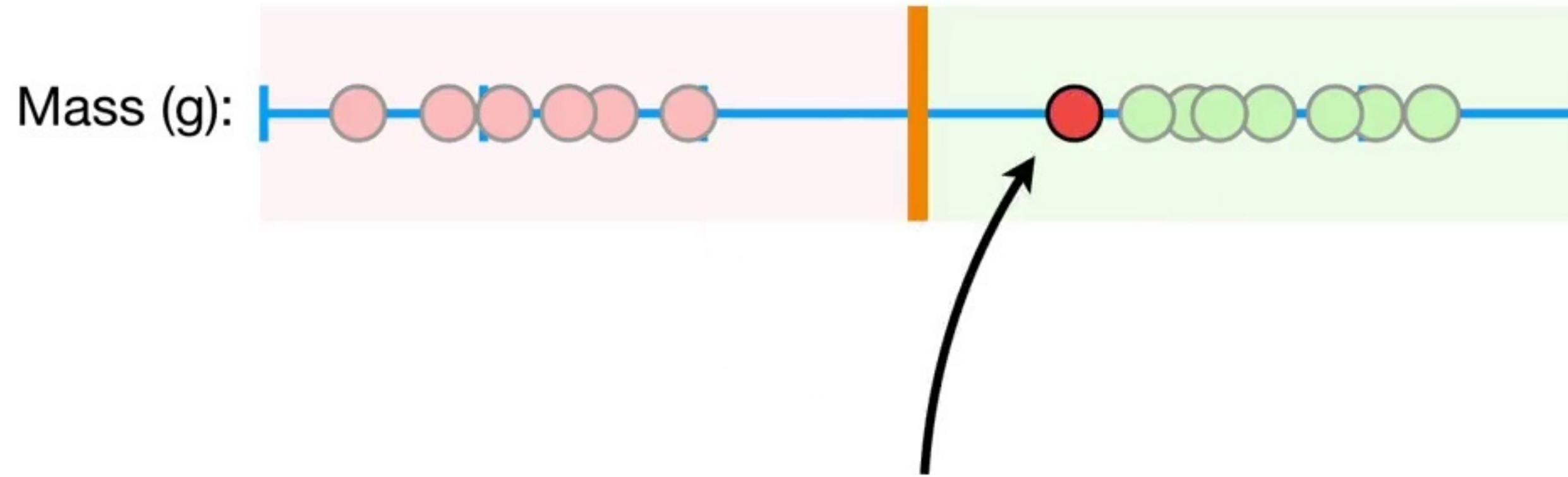
YES!!!



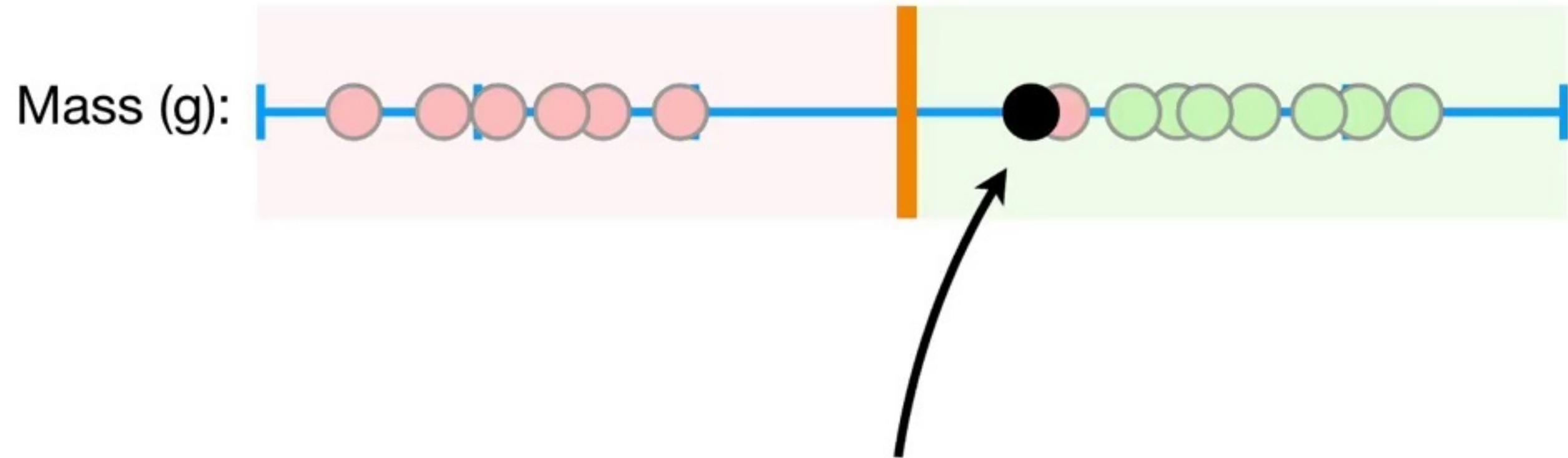
To make a threshold that is not so sensitive to outliers we must **allow misclassifications**.



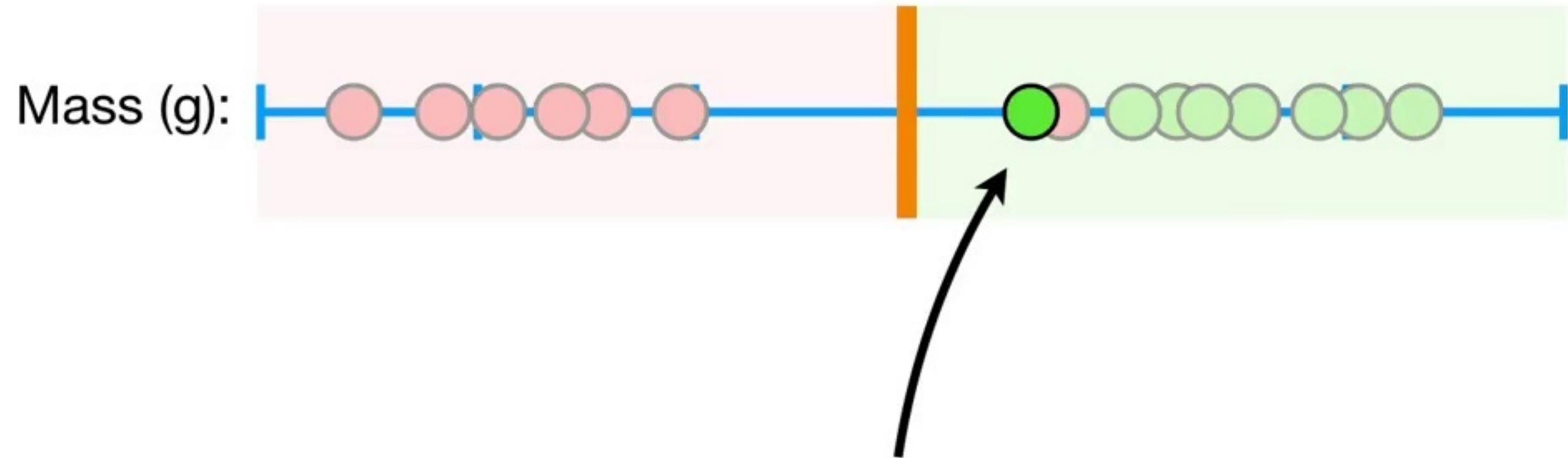
For example, if we put the threshold
halfway between these two
observations...



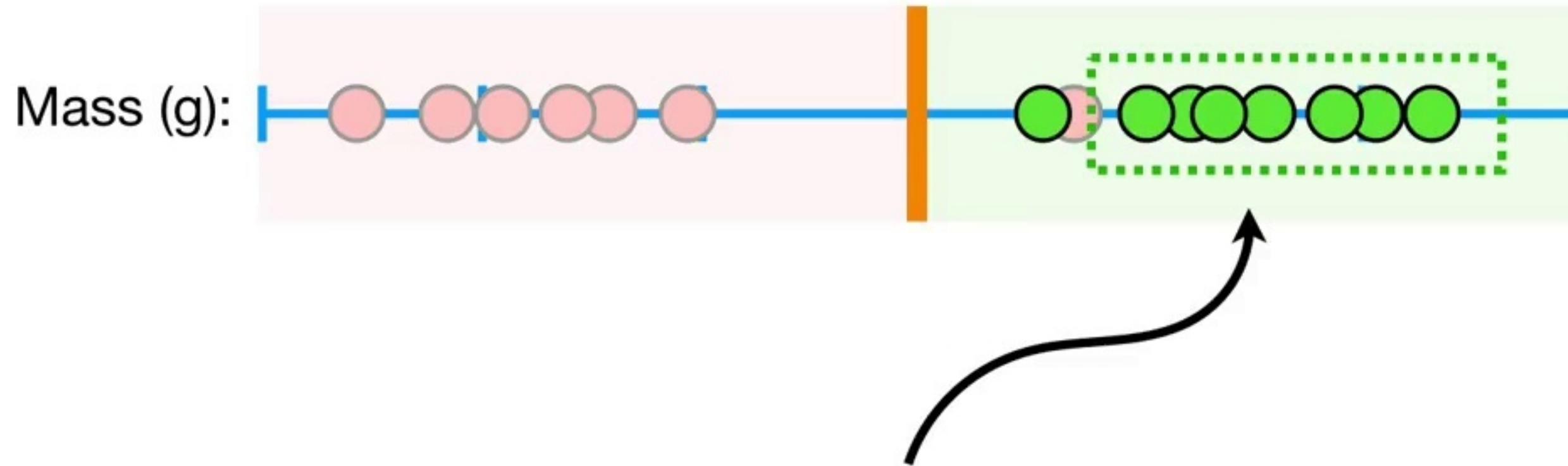
...then we will misclassify this
observation.



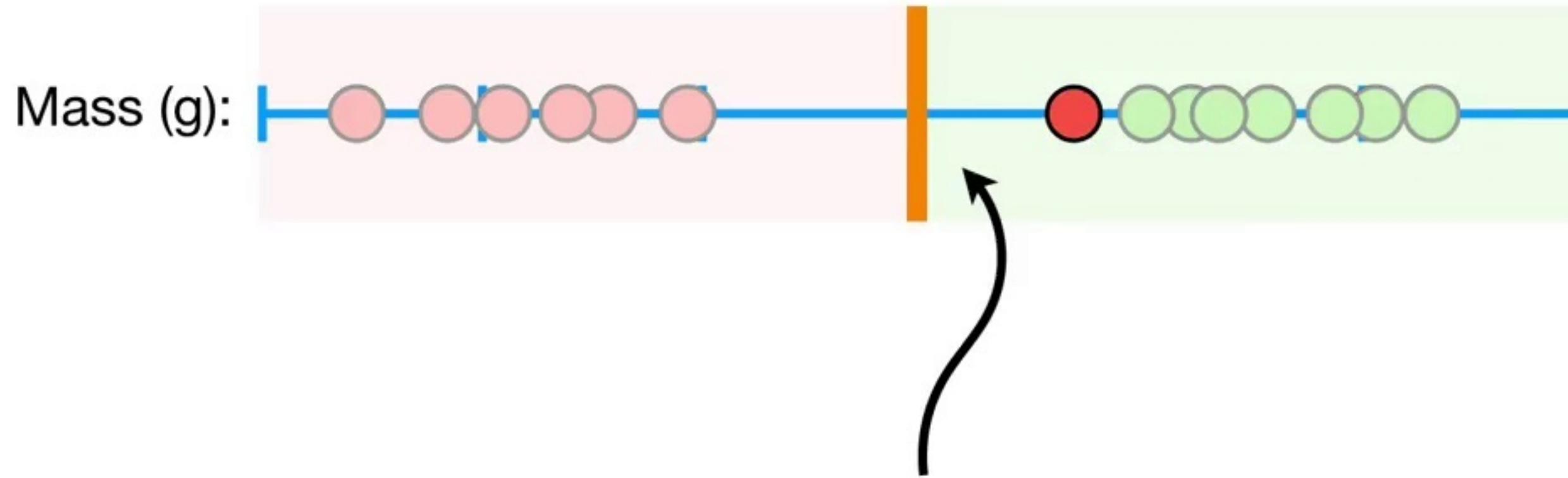
However, now when we get a
new observation here...



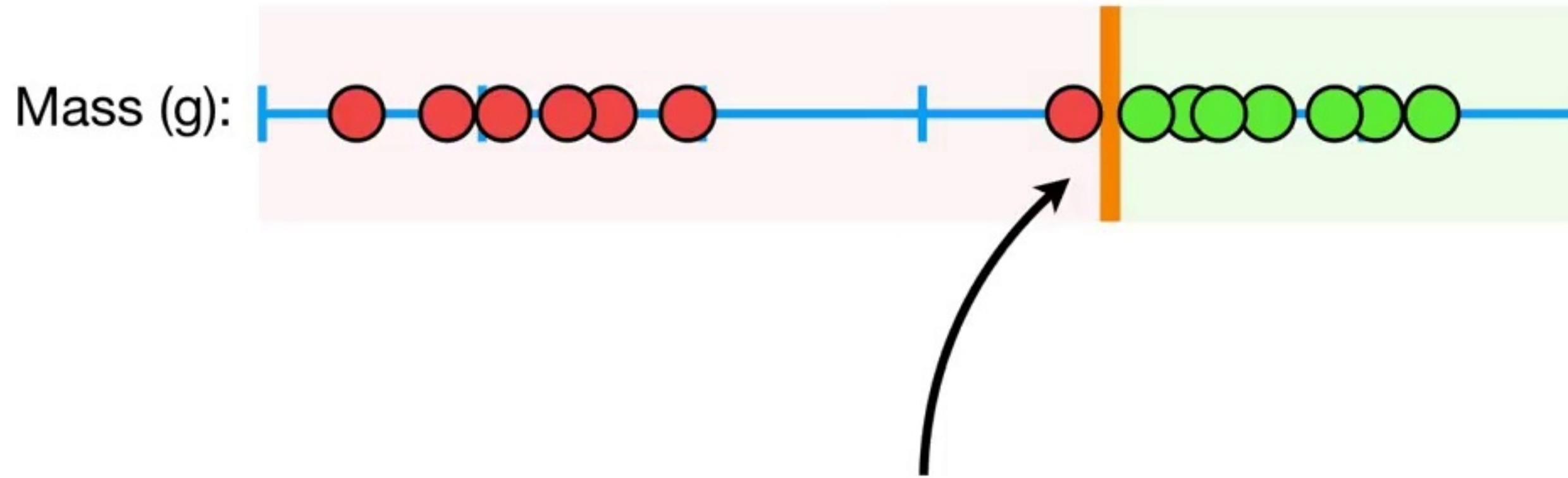
...we will classify it as **obese**...



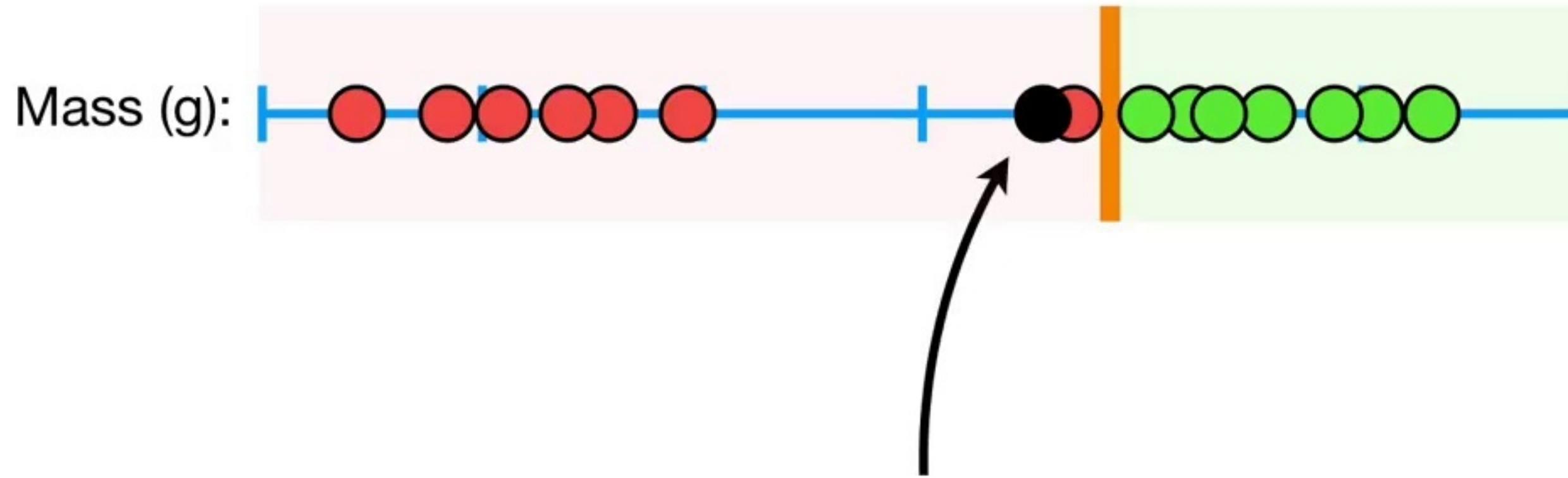
...and that makes sense
because it is closer to most of
the **obese** observations.



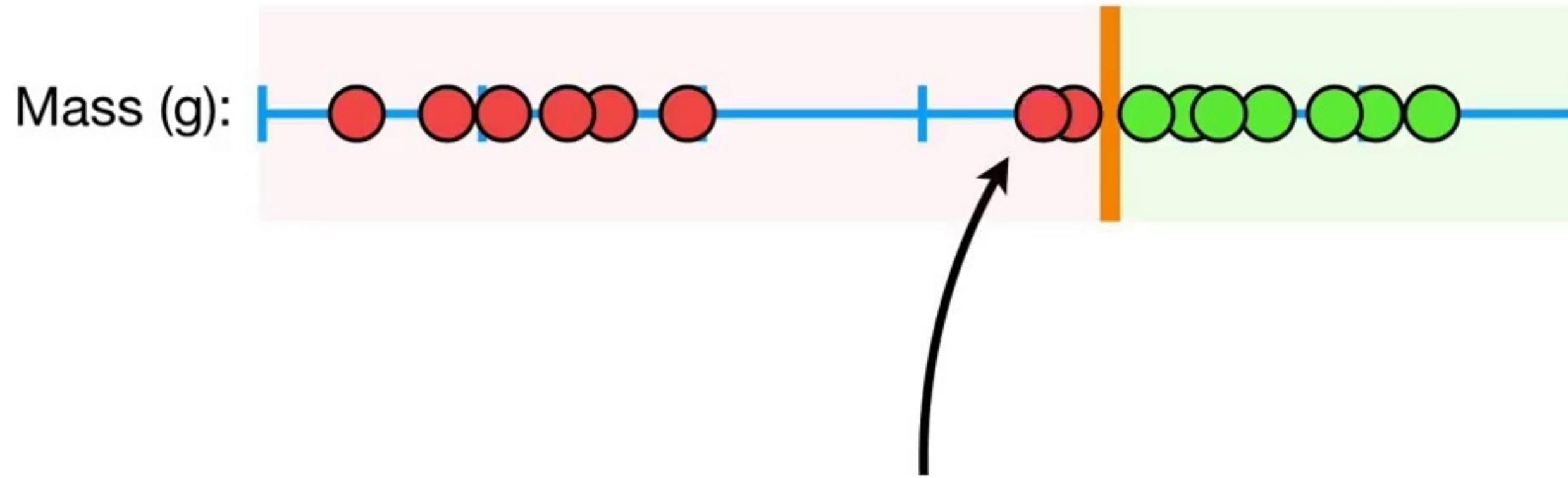
Choosing a threshold that allows misclassifications is an example of the **Bias/Variance Tradeoff** that plagues all of machine learning.



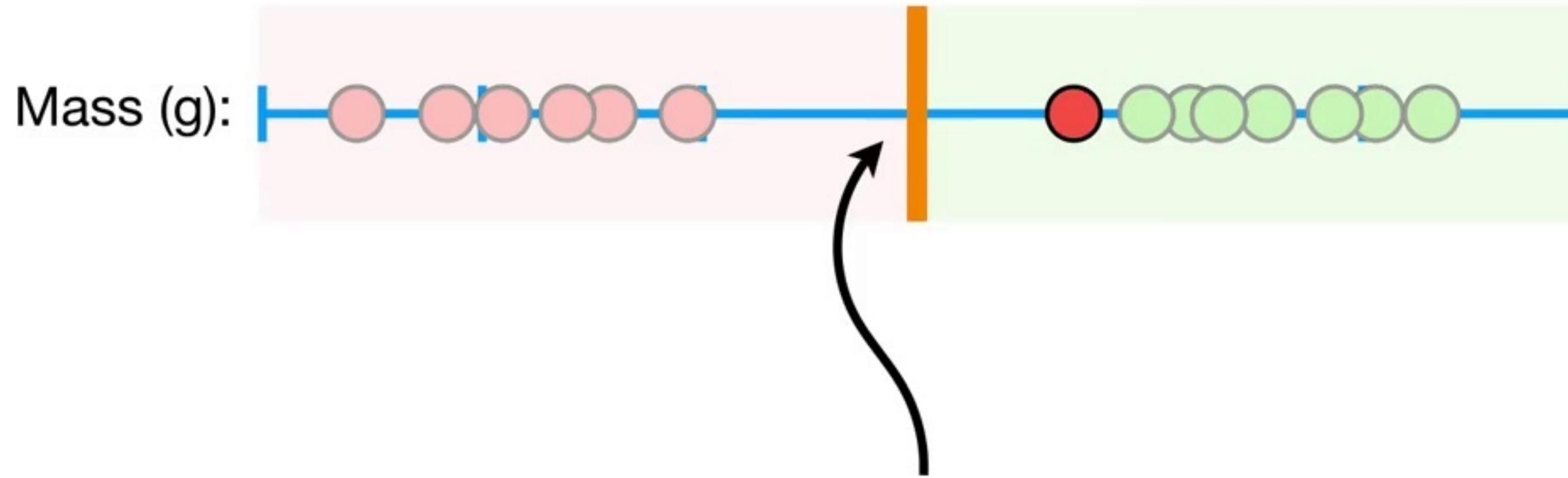
In other words, before we allowed misclassifications, we picked a threshold that was very sensitive to the training data (low bias)...



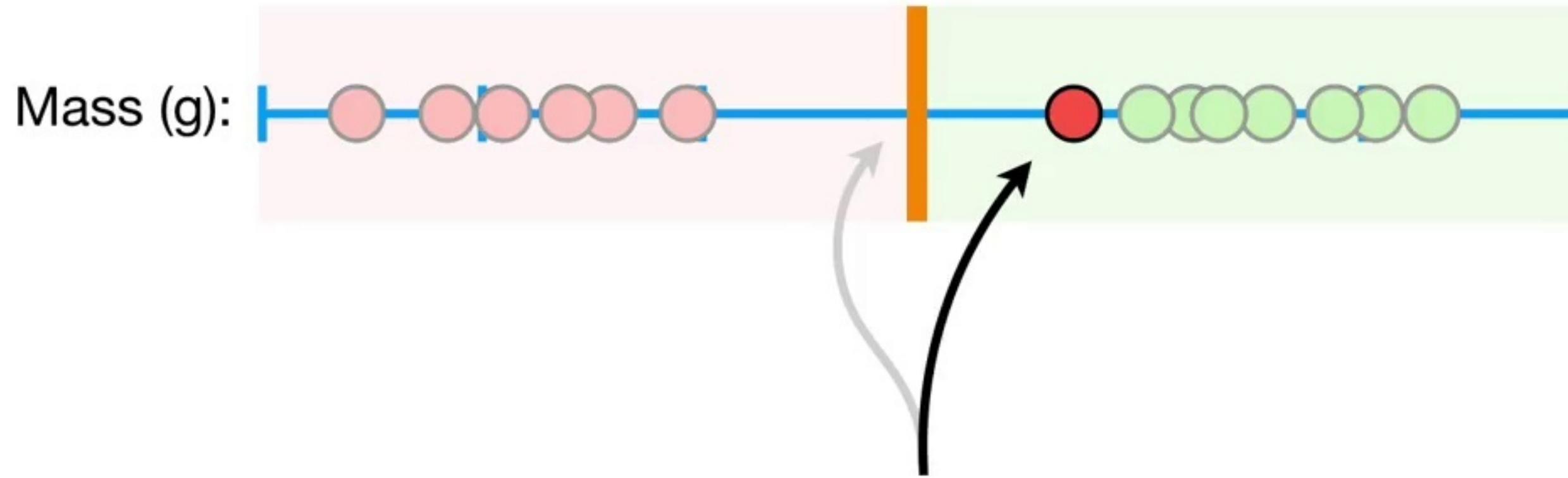
...and it performed poorly when
we got new data (high variance).



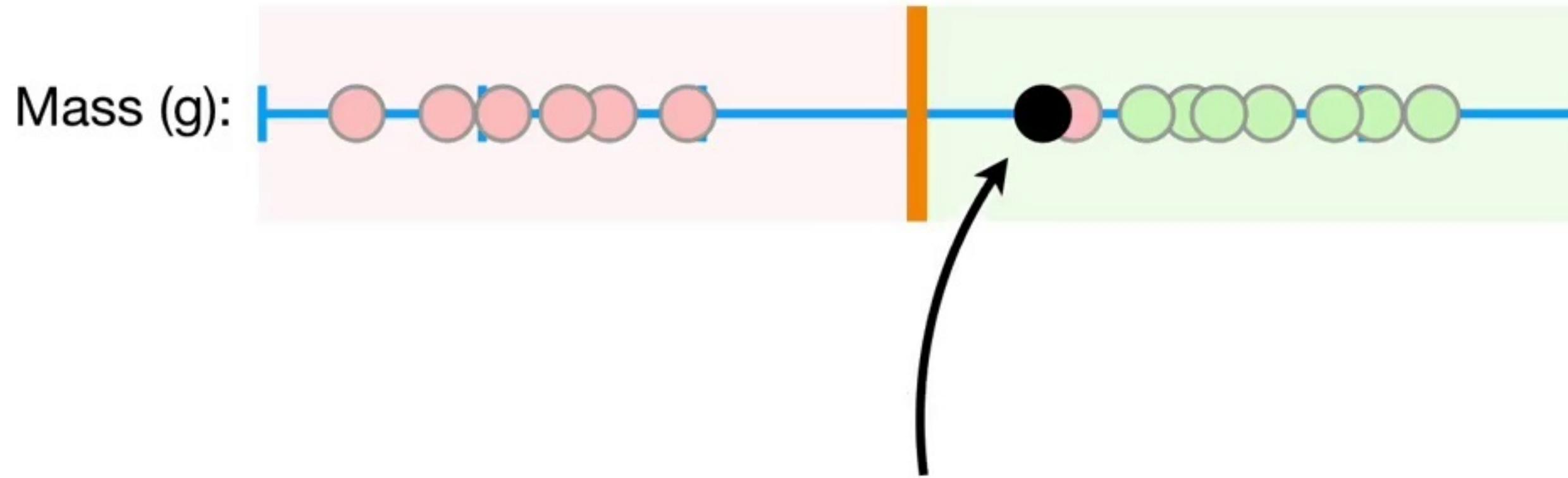
...and it performed poorly when
we got new data (high variance).



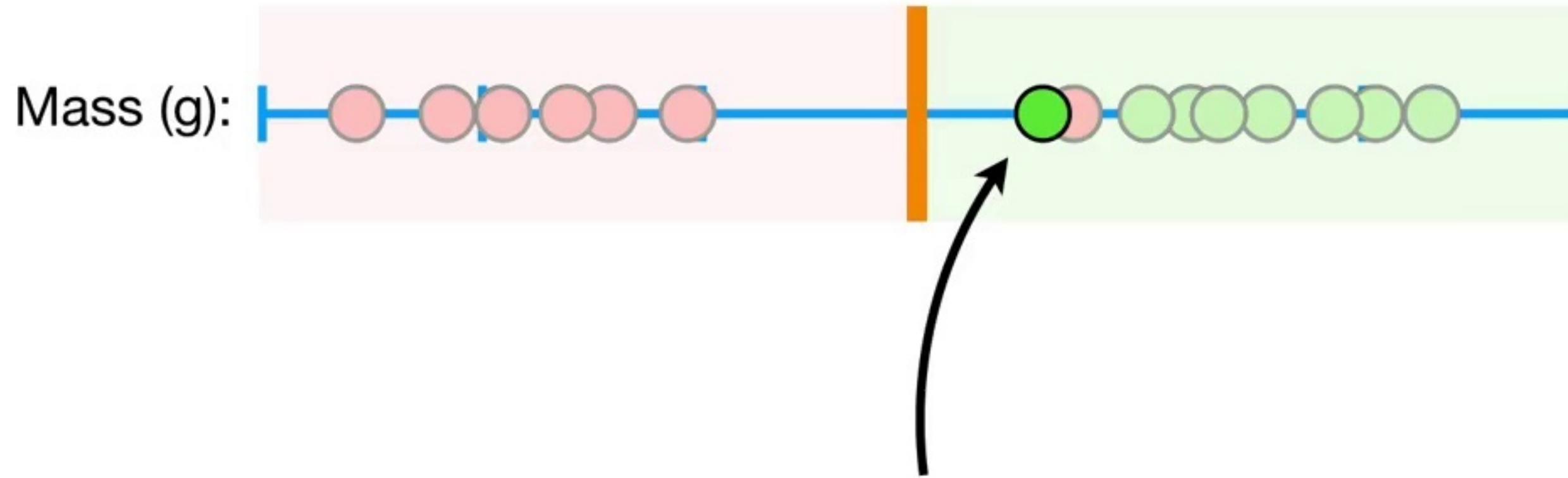
In contrast, when we picked a threshold that was less sensitive to the training data and allowed misclassifications (higher bias)...



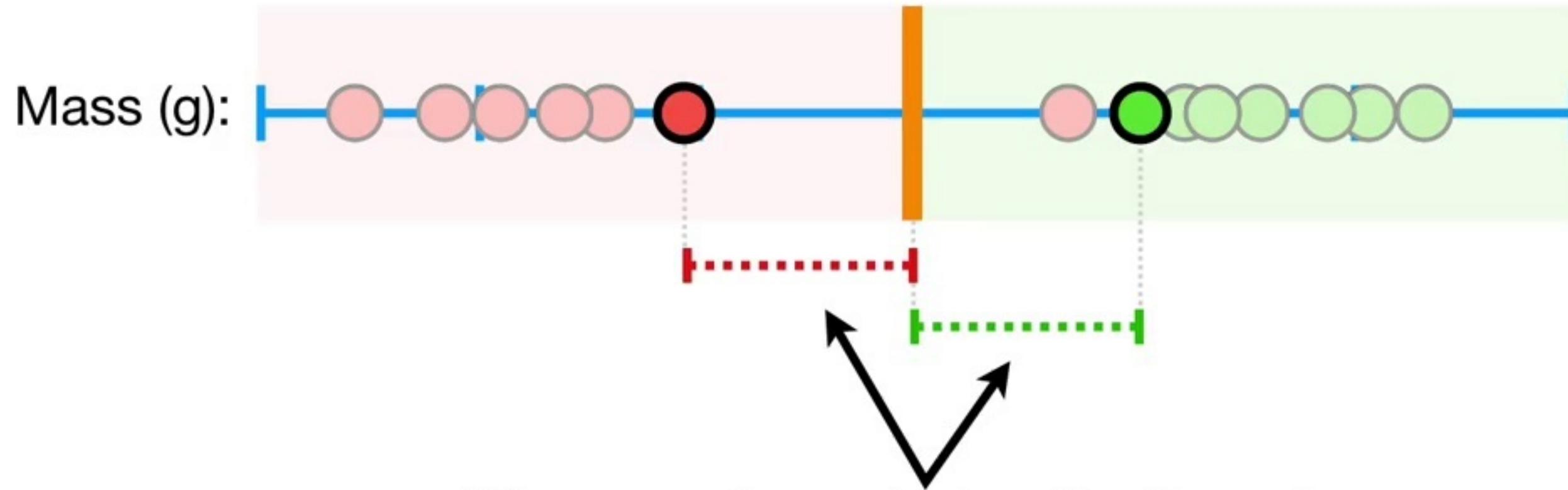
In contrast, when we picked a threshold that was less sensitive to the training data and allowed misclassifications (higher bias)...



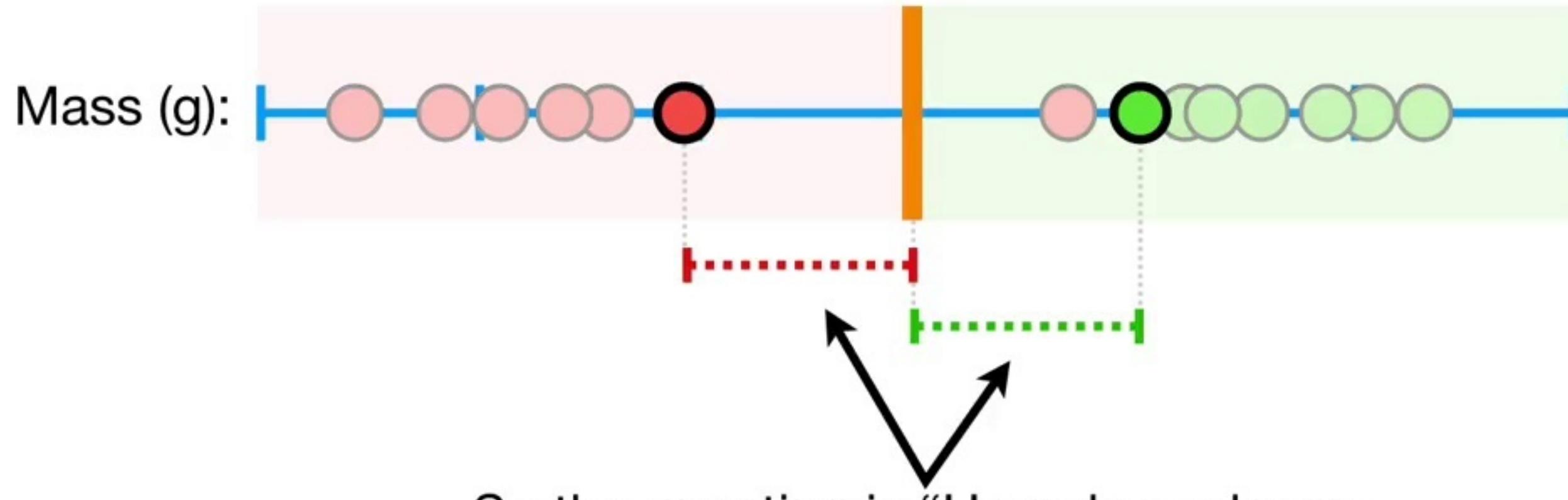
...it performed better when we got
new data (low variance).



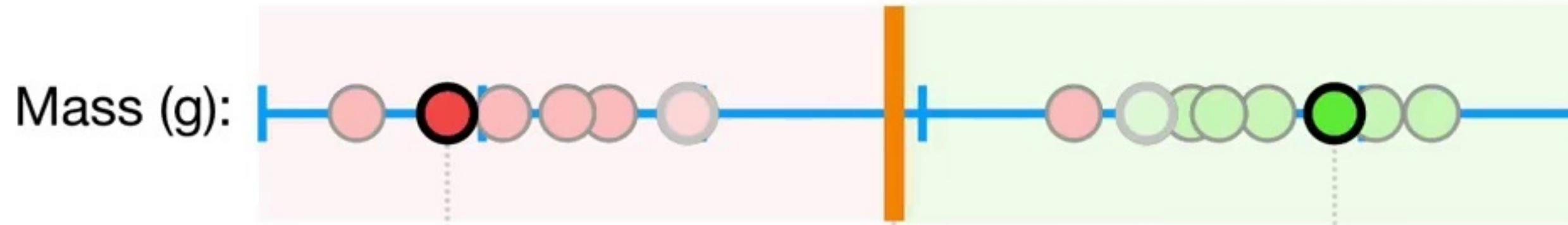
...it performed better when we got
new data (low variance).



When we allow misclassifications, the distance between the observations and the threshold is called a **Soft Margin**.

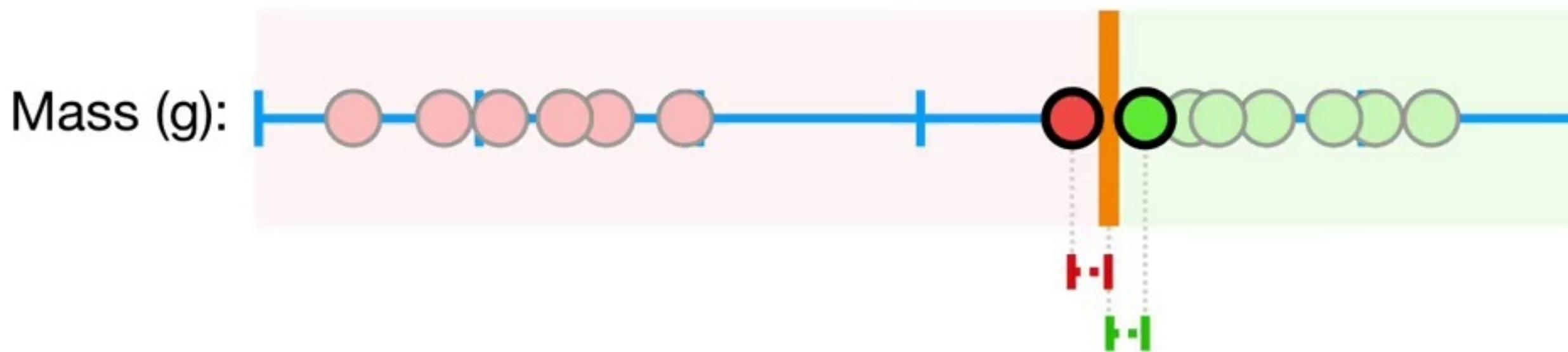


So the question is “How do we know
that this **soft margin**...

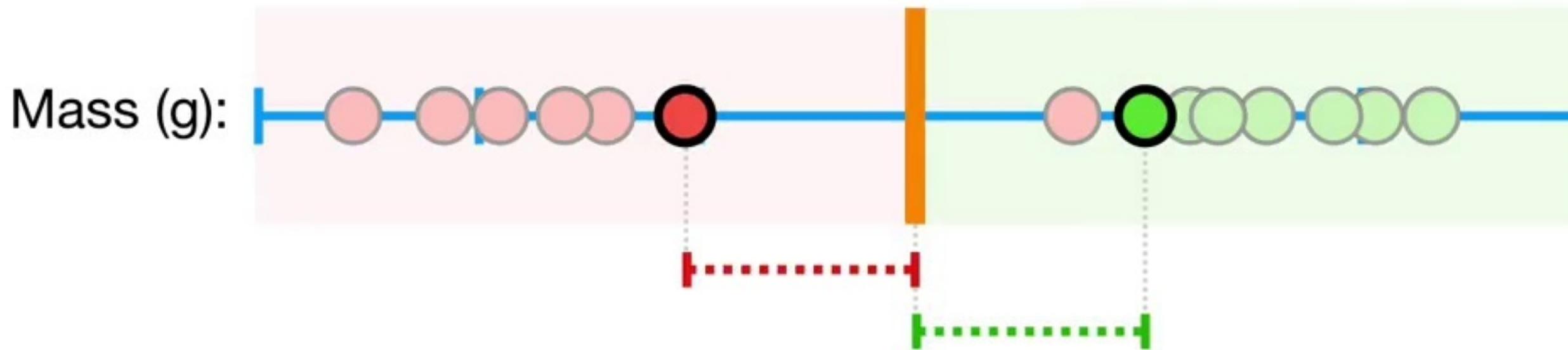


So the question is “How do we know
that this **soft margin**...

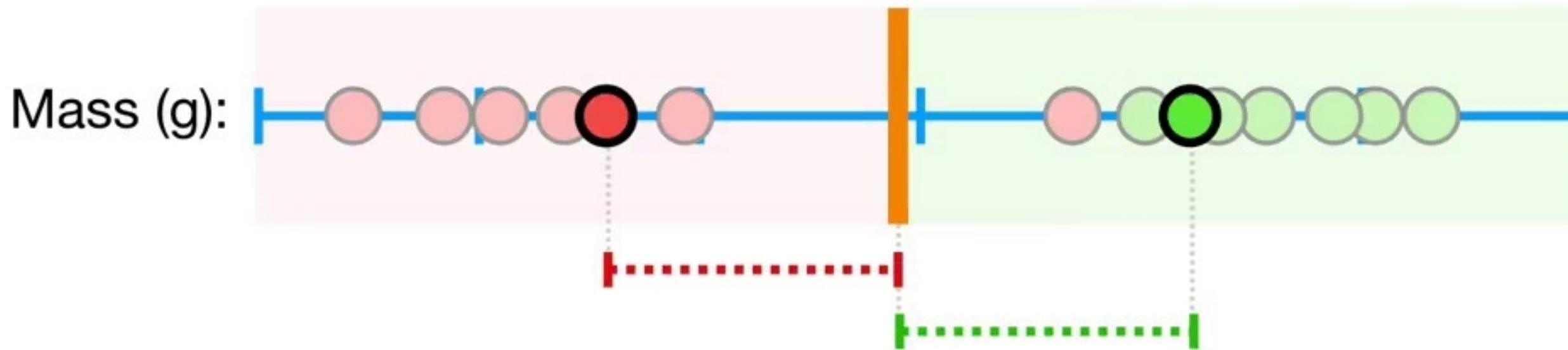
...is better than this **Soft Margin?**”



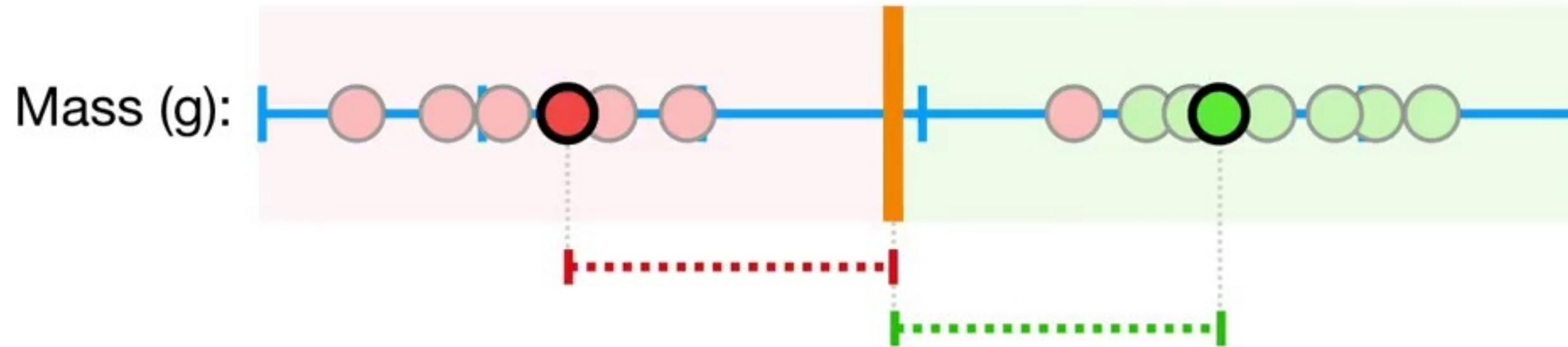
The answer is simple: We use **Cross Validation** to determine how many misclassifications and observations to allow inside of the **Soft Margin** to get the best classification.



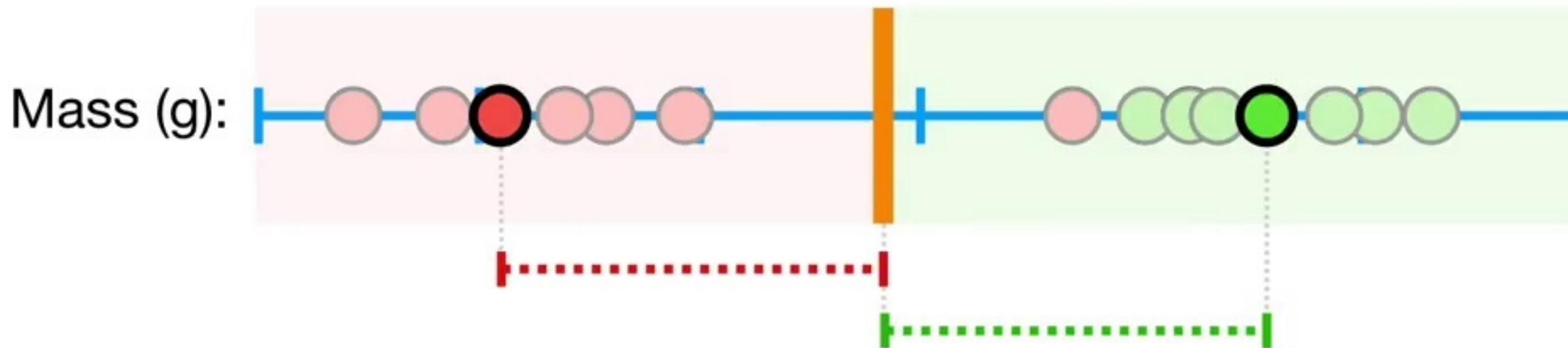
The answer is simple: We use **Cross Validation** to determine how many misclassifications and observations to allow inside of the **Soft Margin** to get the best classification.



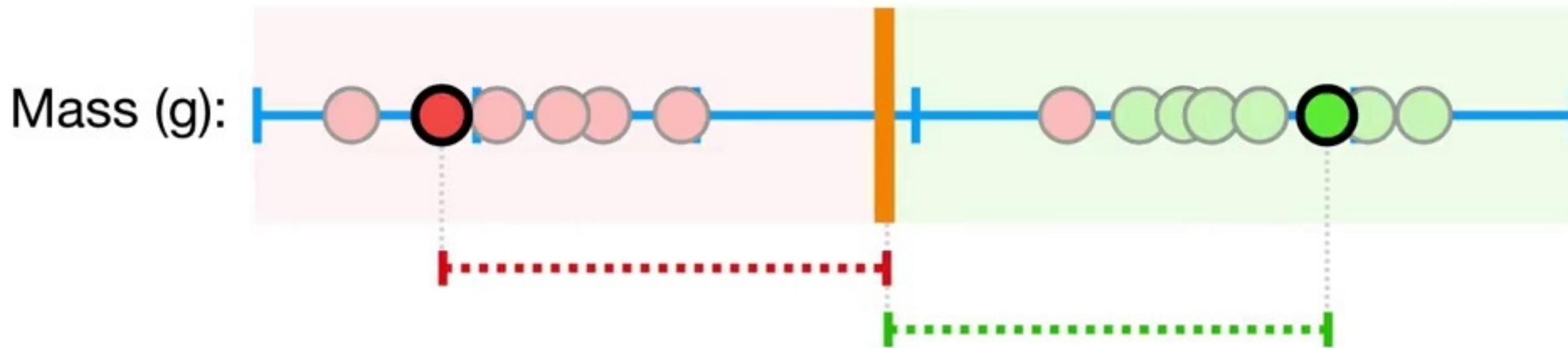
The answer is simple: We use **Cross Validation** to determine how many misclassifications and observations to allow inside of the **Soft Margin** to get the best classification.



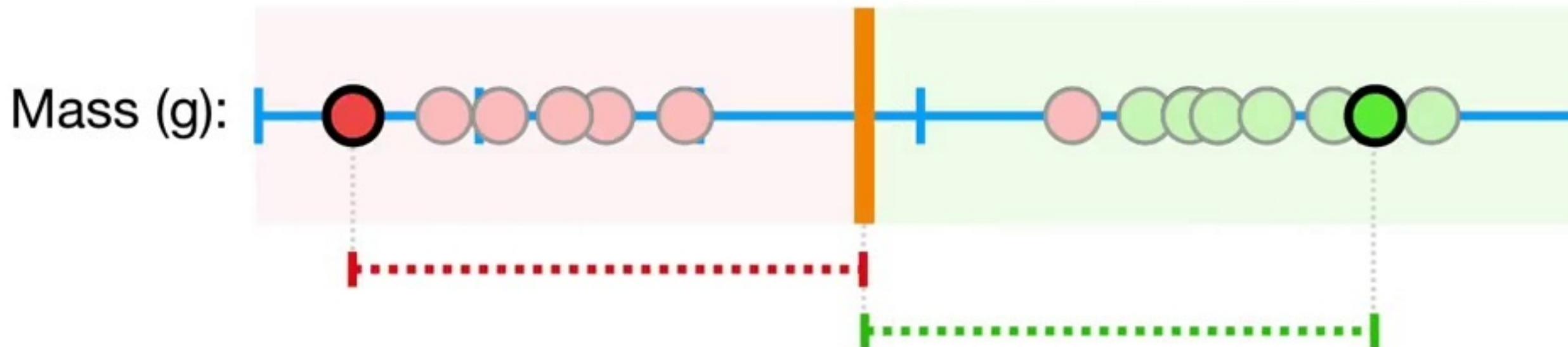
The answer is simple: We use **Cross Validation** to determine how many misclassifications and observations to allow inside of the **Soft Margin** to get the best classification.



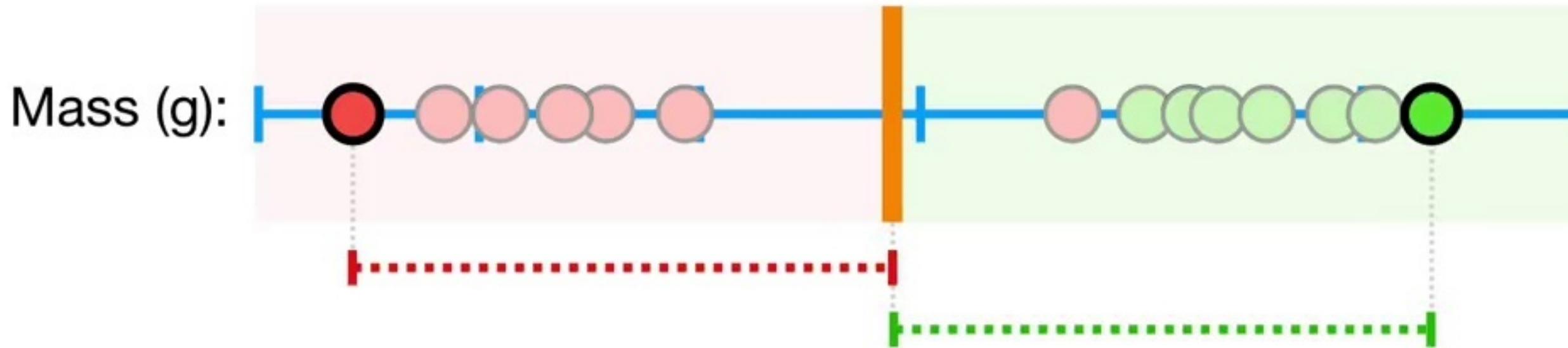
The answer is simple: We use **Cross Validation** to determine how many misclassifications and observations to allow inside of the **Soft Margin** to get the best classification.



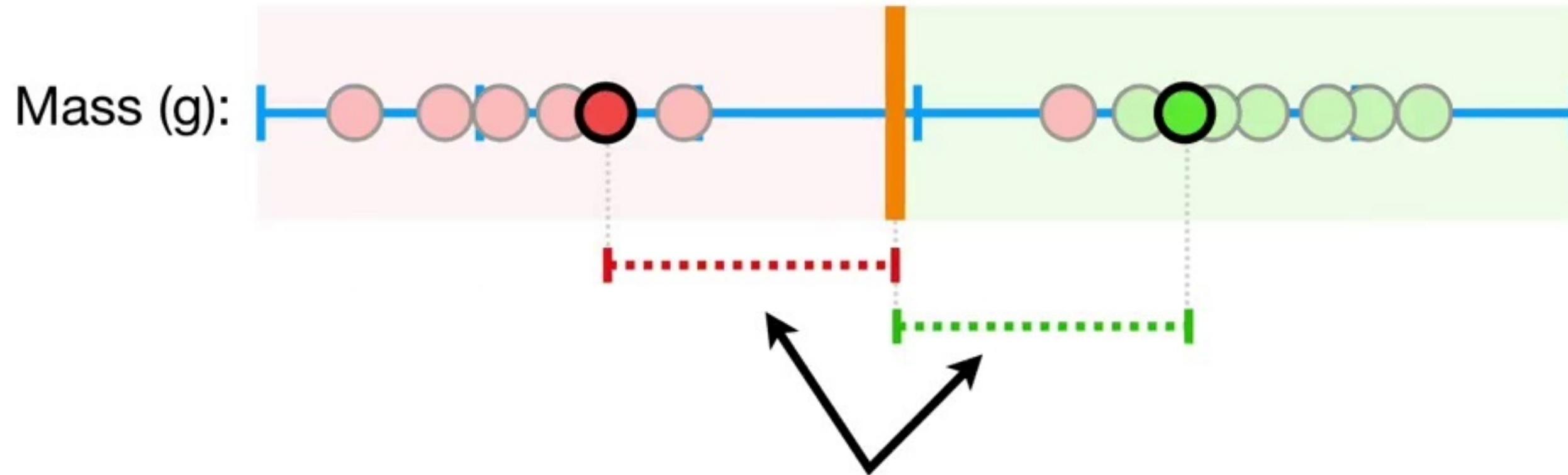
The answer is simple: We use **Cross Validation** to determine how many misclassifications and observations to allow inside of the **Soft Margin** to get the best classification.



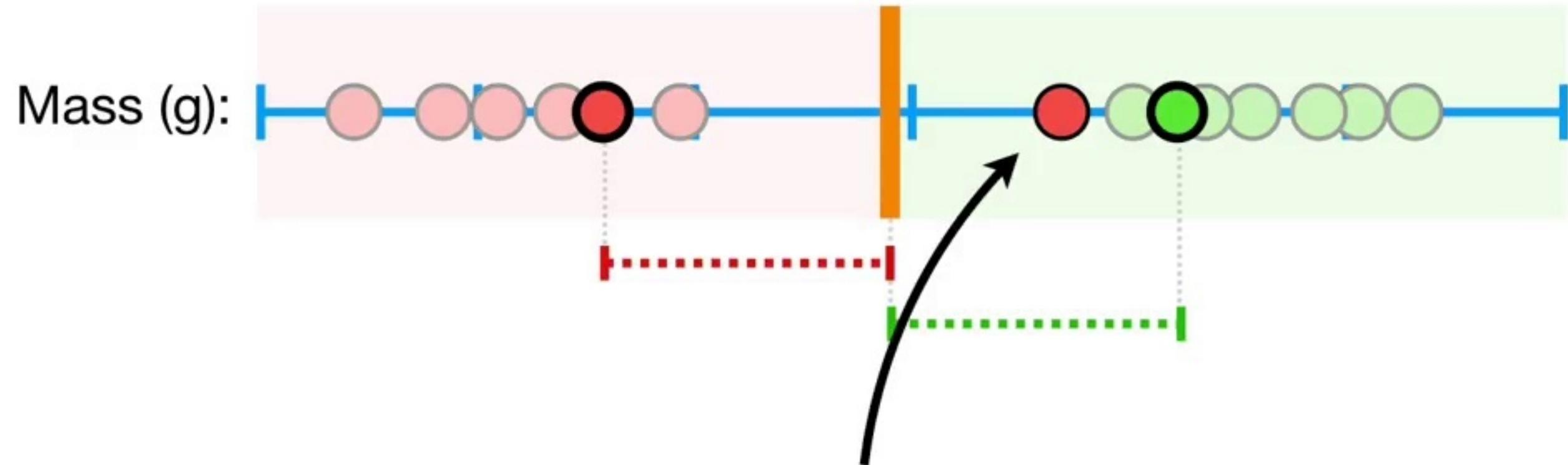
The answer is simple: We use **Cross Validation** to determine how many misclassifications and observations to allow inside of the **Soft Margin** to get the best classification.



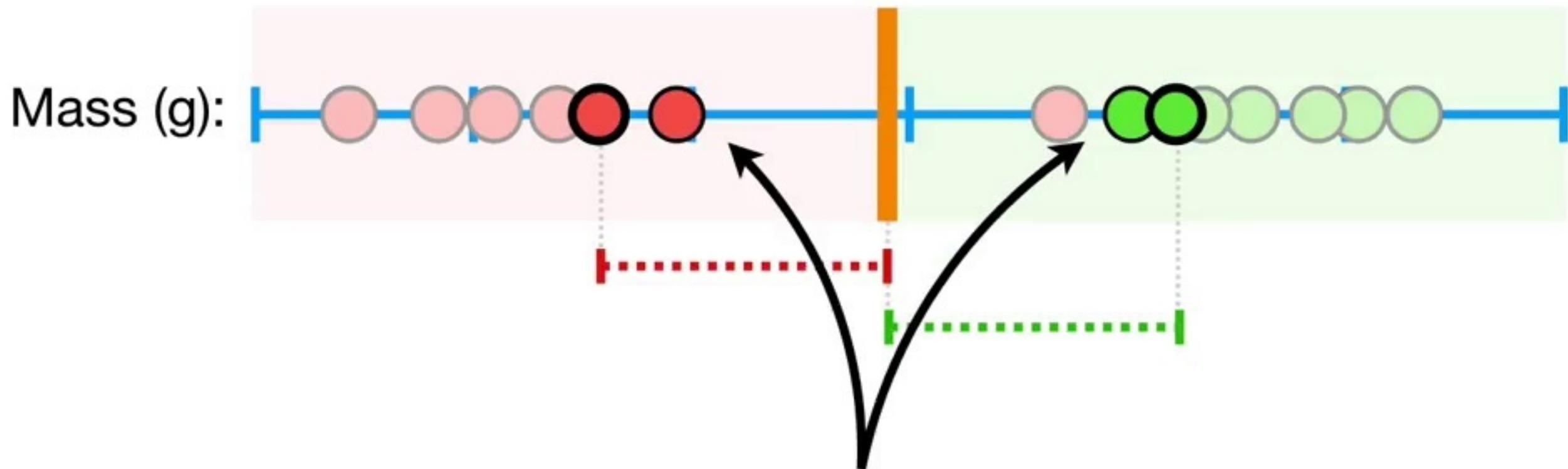
The answer is simple: We use **Cross Validation** to determine how many misclassifications and observations to allow inside of the **Soft Margin** to get the best classification.



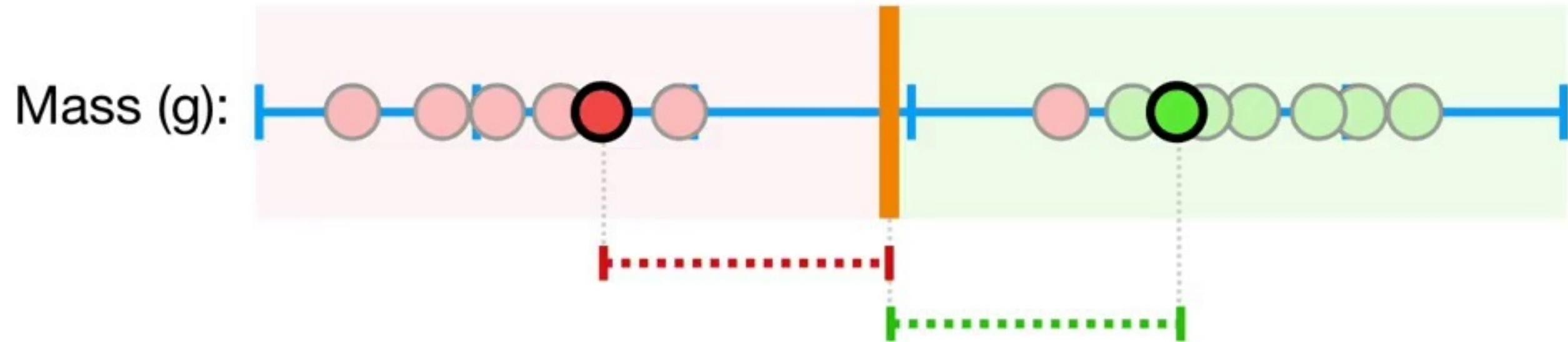
For example, if **Cross Validation** determined that this was the best **Soft Margin**...



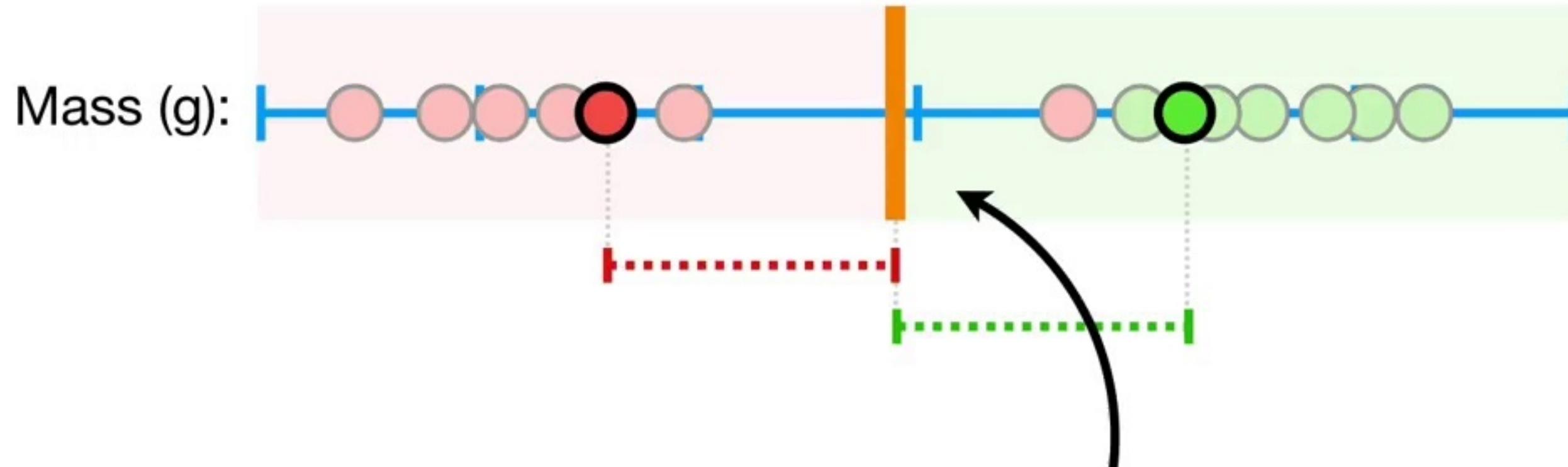
...then we would allow one
misclassification...



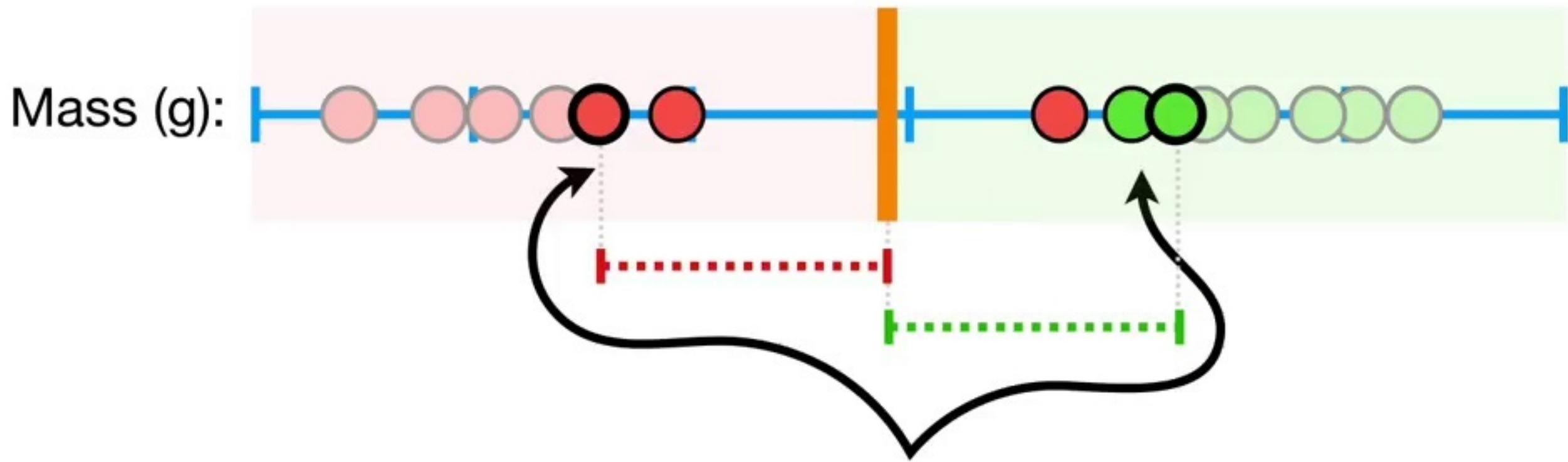
...and two observations, that are
correctly classified, to be within
the **Soft Margin**.



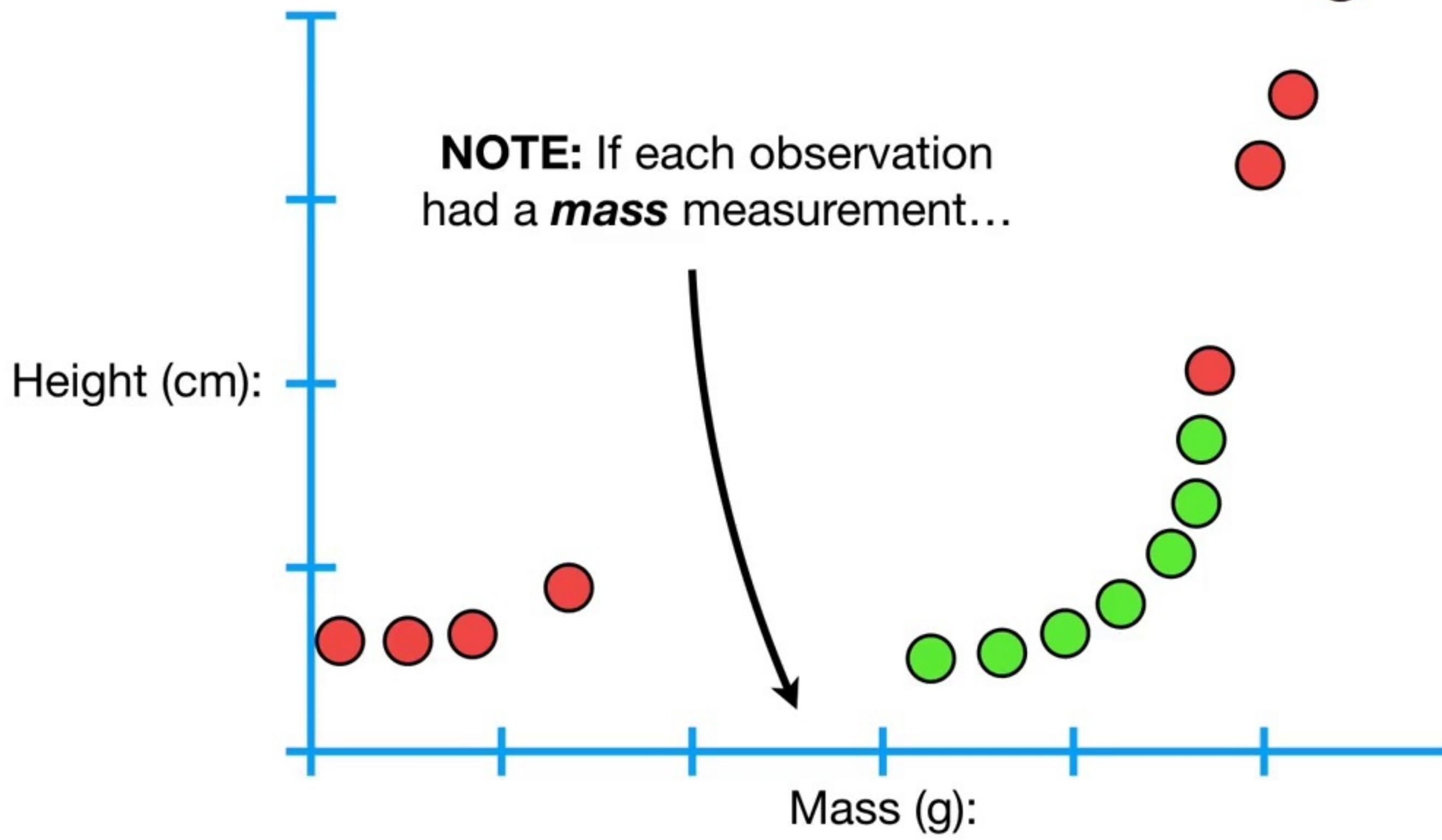
When we use a **Soft Margin** to determine the location of a threshold...

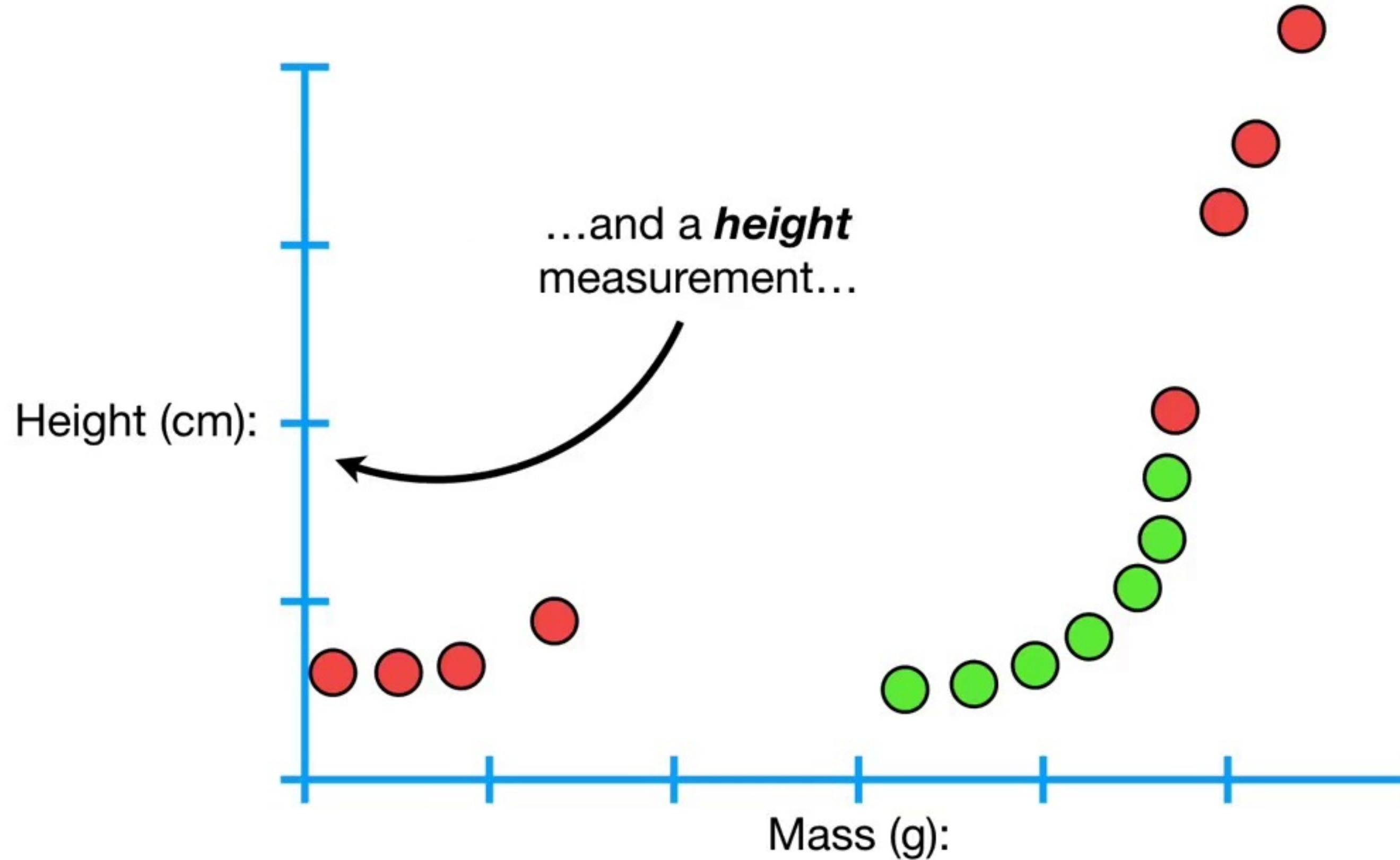


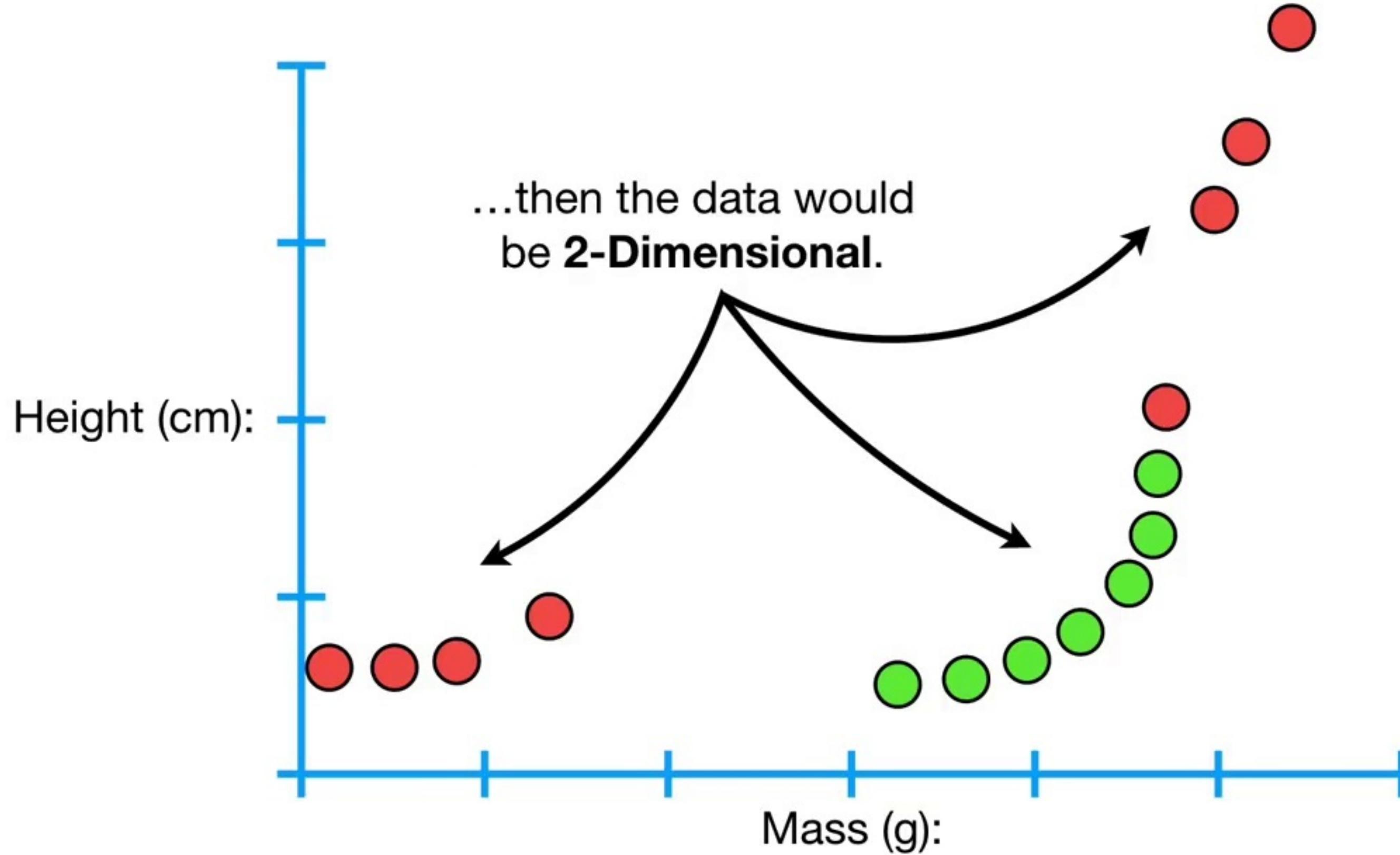
...then we are using a **Soft Margin Classifier** aka
a **Support Vector Classifier** to classify
observations.



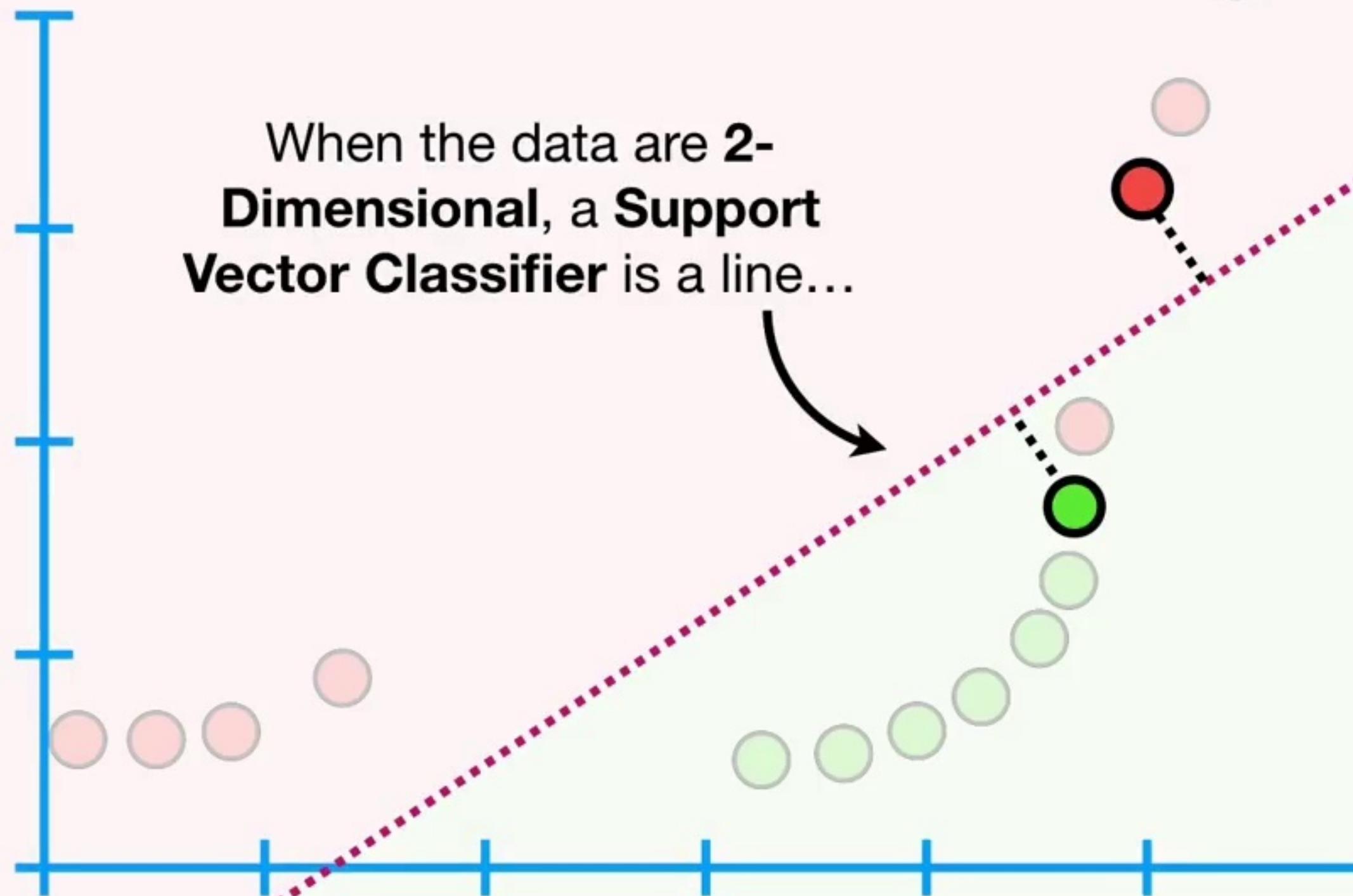
The name **Support Vector Classifier** comes from the fact that the observations on the edge *and within* the **Soft Margin** are called **Support Vectors**.

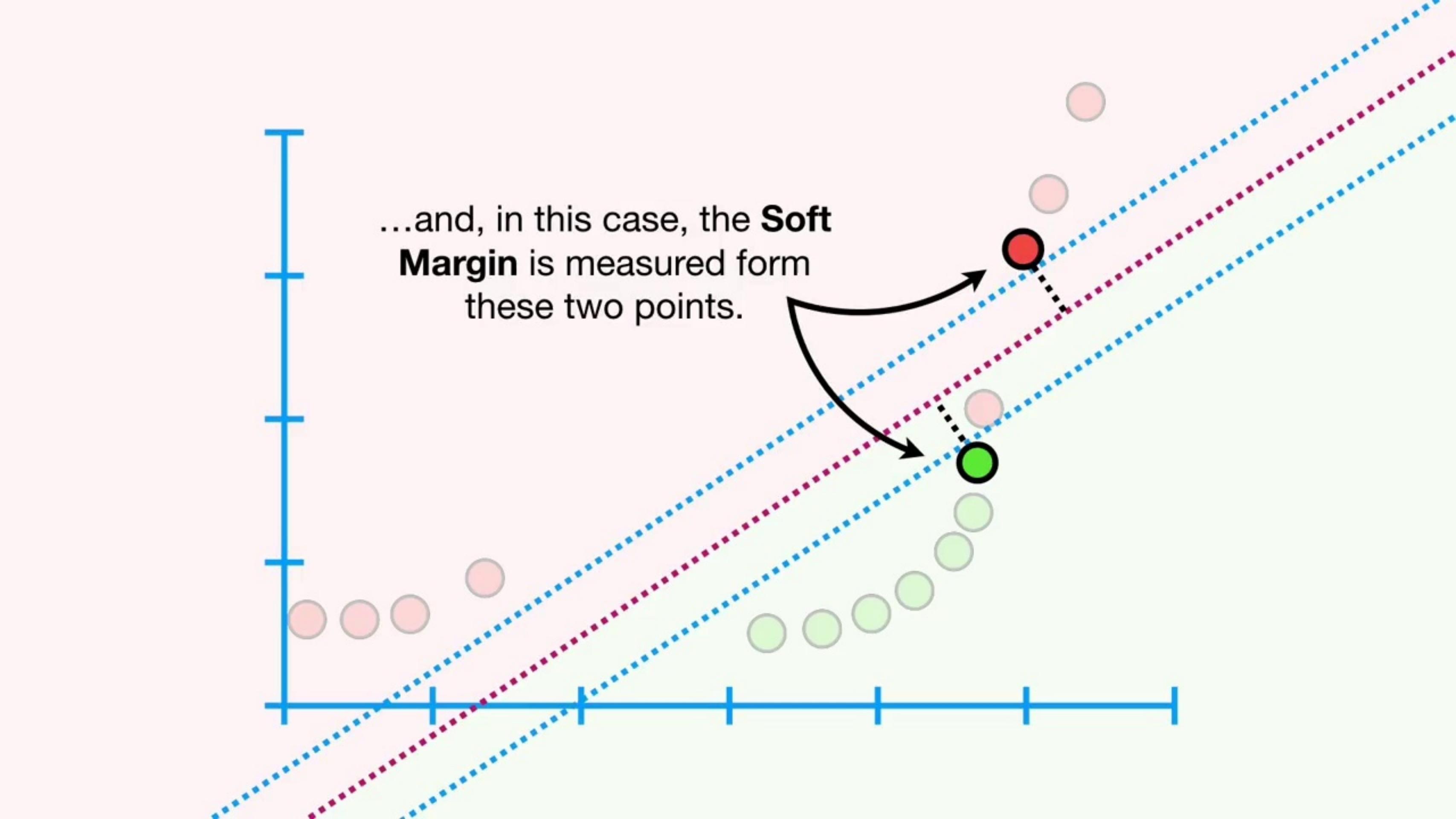




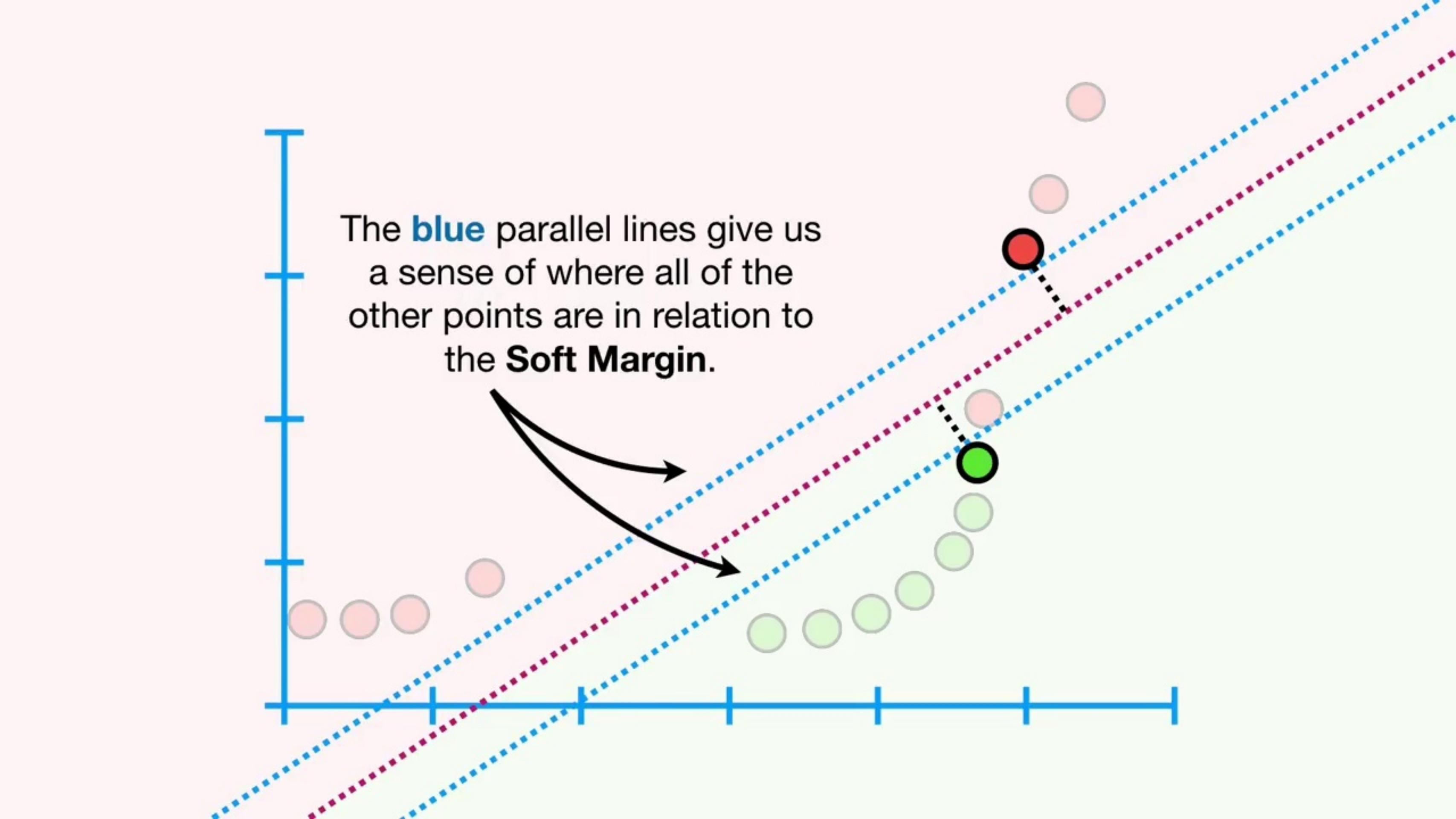


When the data are **2-Dimensional**, a **Support Vector Classifier** is a line...

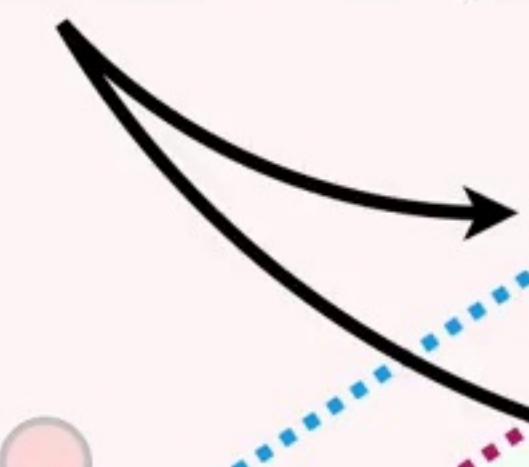




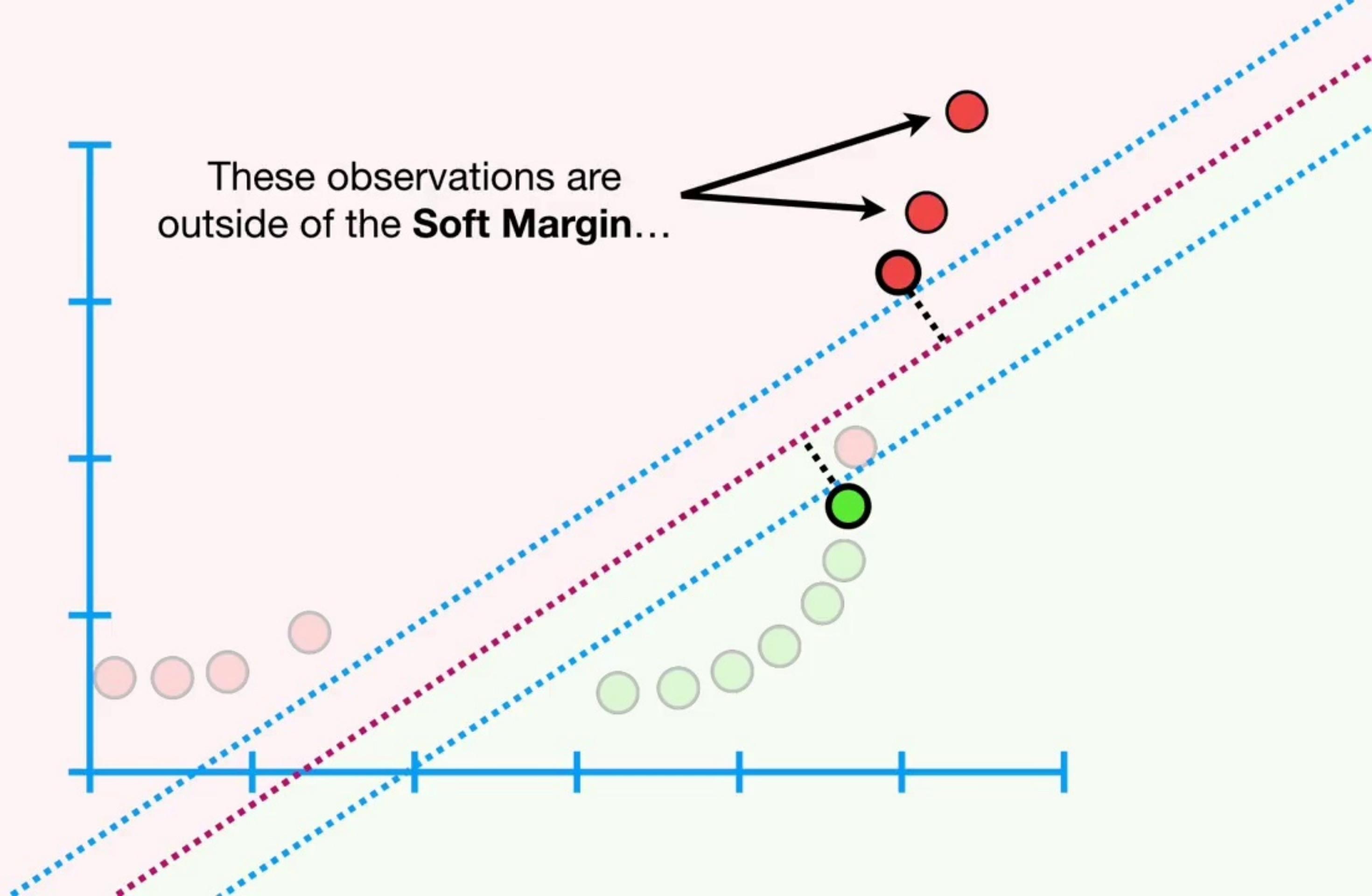
...and, in this case, the **Soft Margin** is measured from these two points.



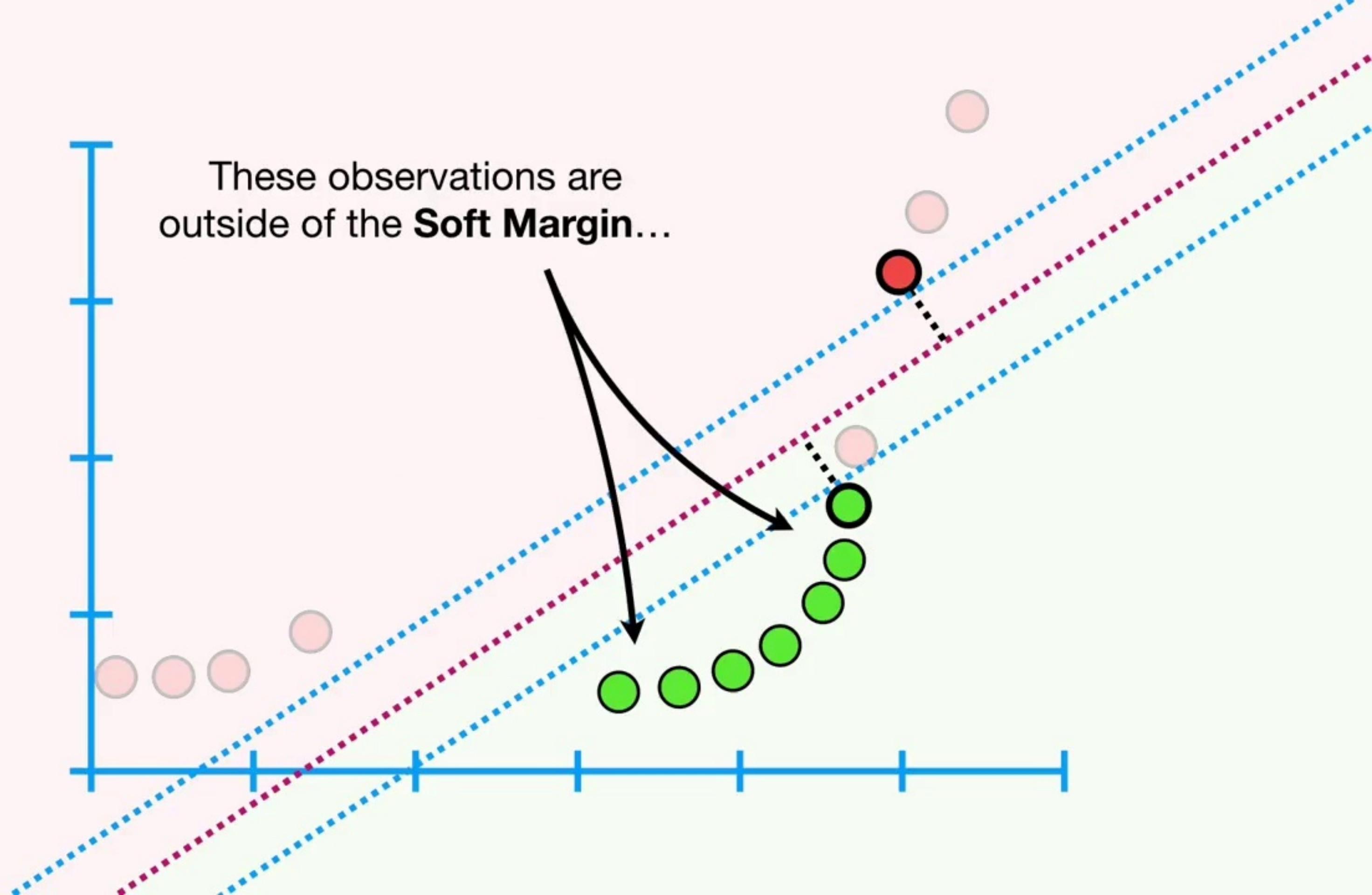
The **blue** parallel lines give us
a sense of where all of the
other points are in relation to
the Soft Margin.



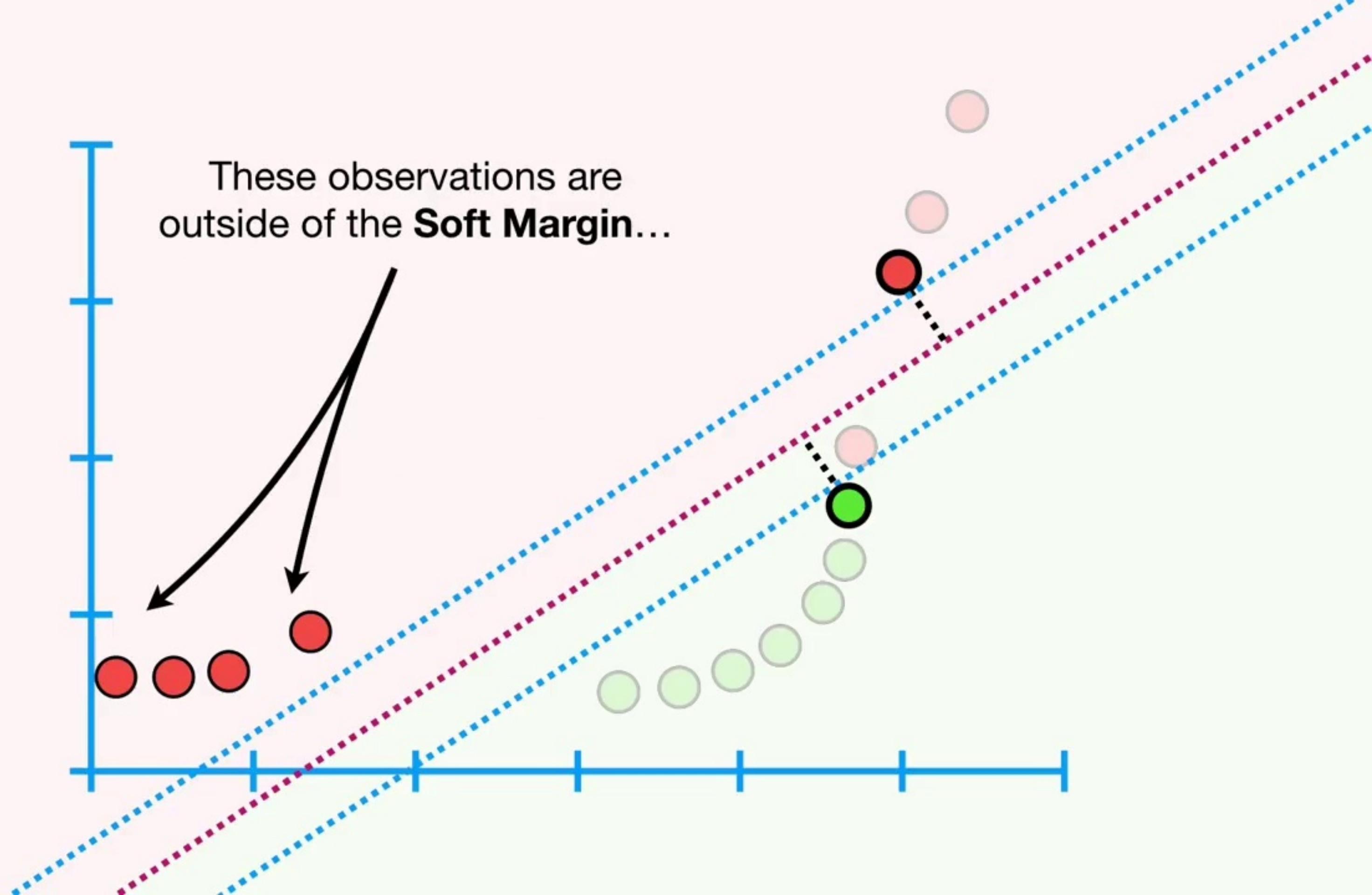
These observations are
outside of the **Soft Margin**...



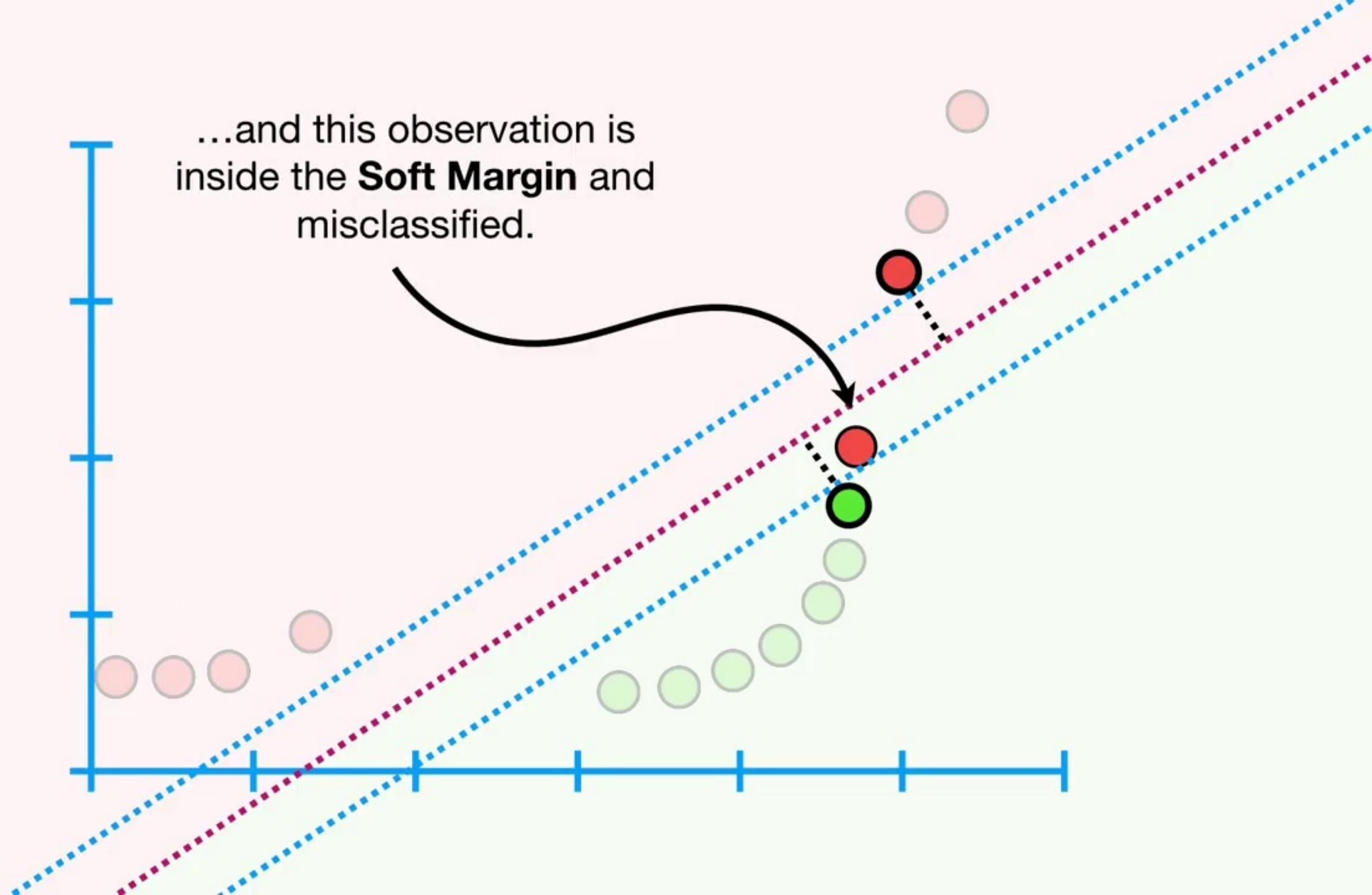
These observations are
outside of the **Soft Margin**...



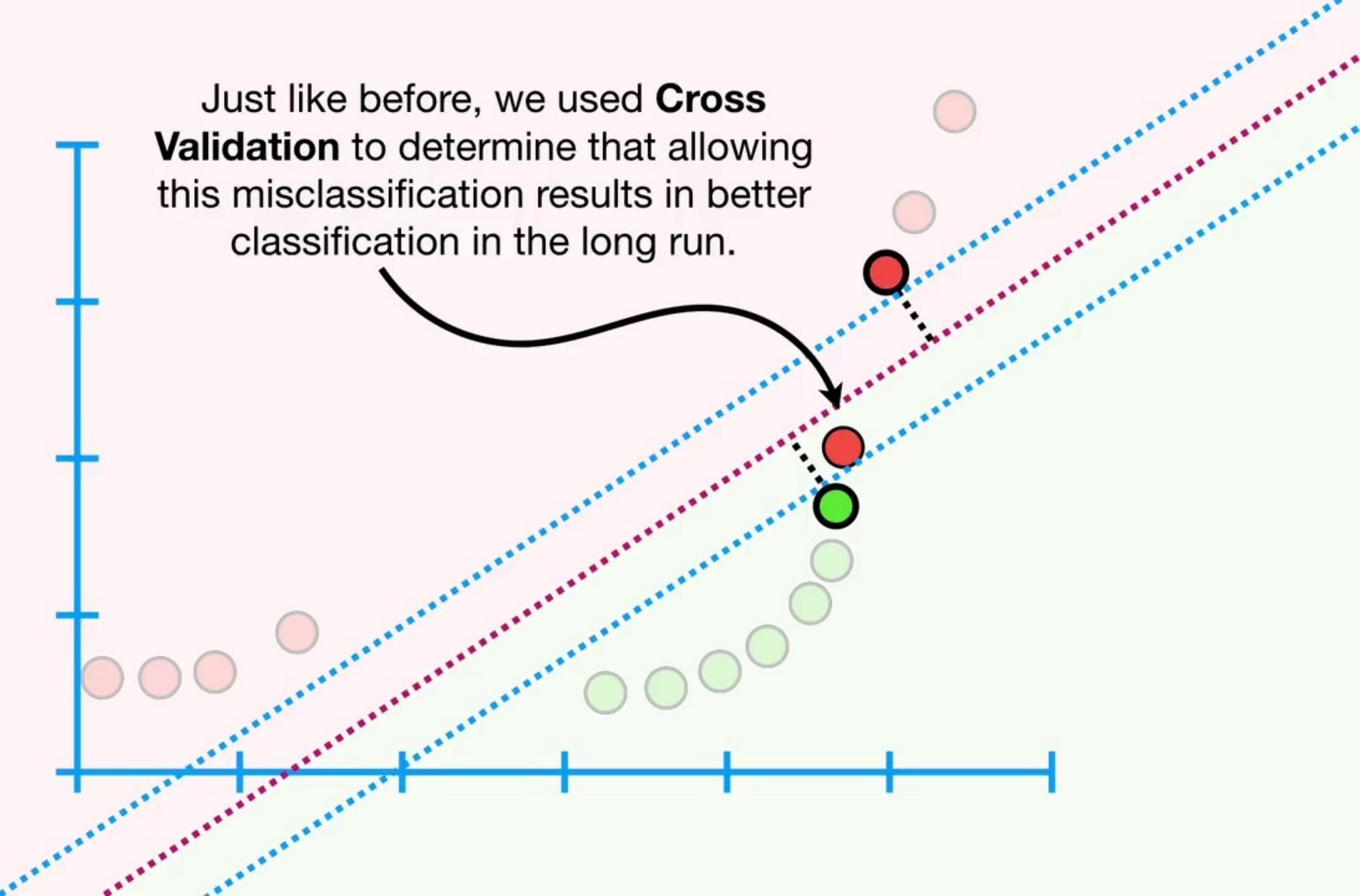
These observations are outside of the **Soft Margin..**

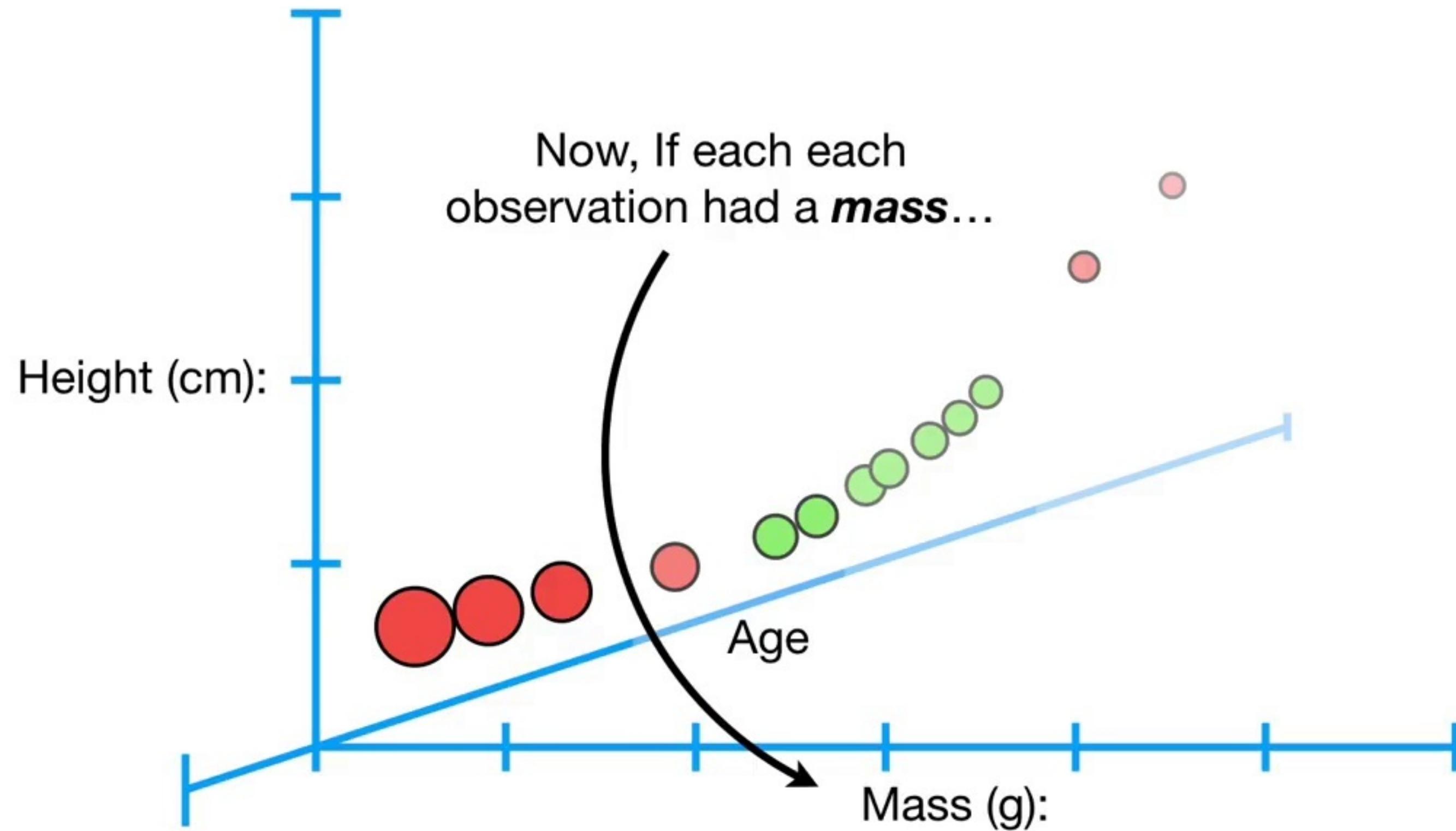


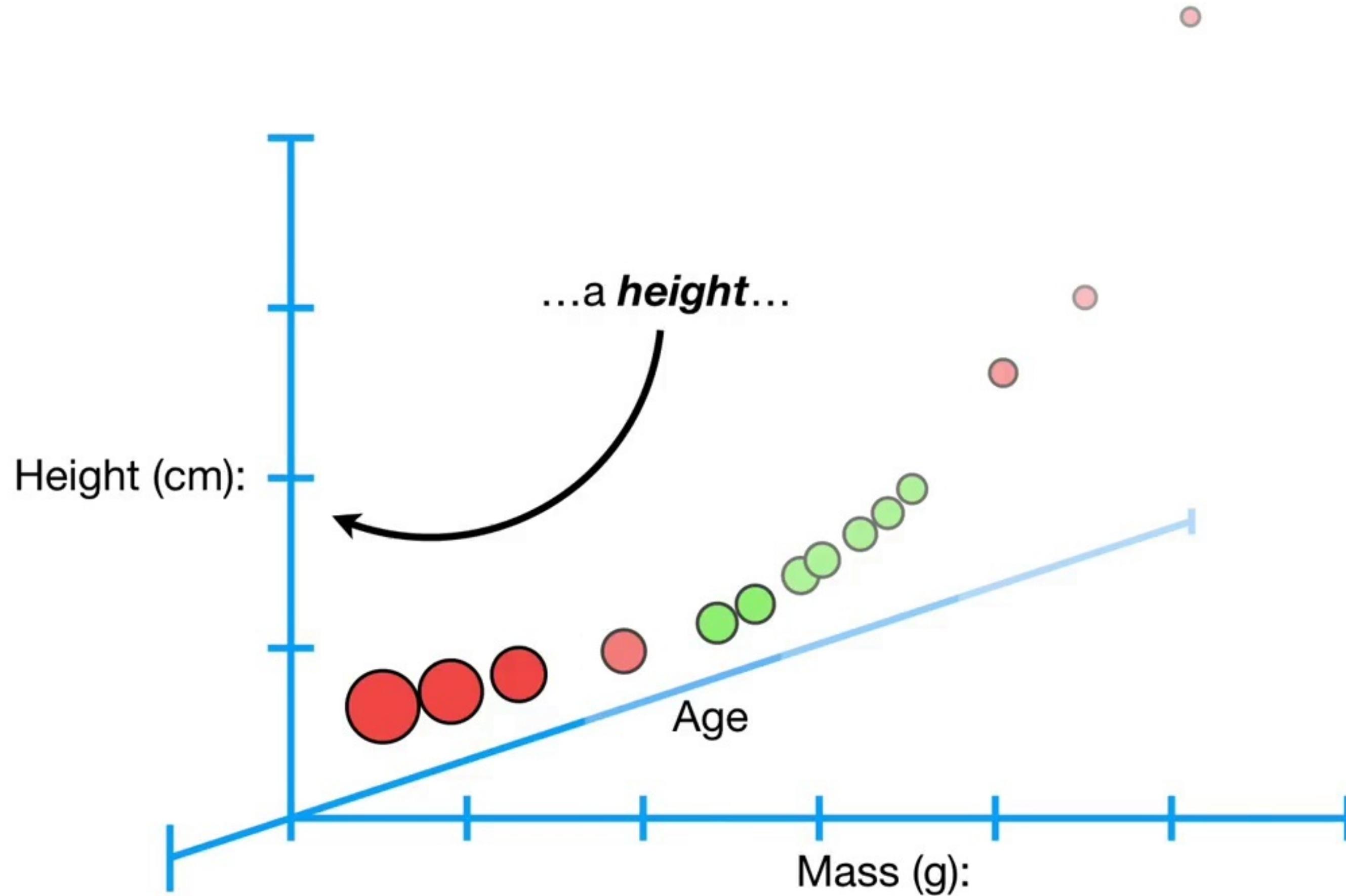
...and this observation is
inside the **Soft Margin** and
misclassified.

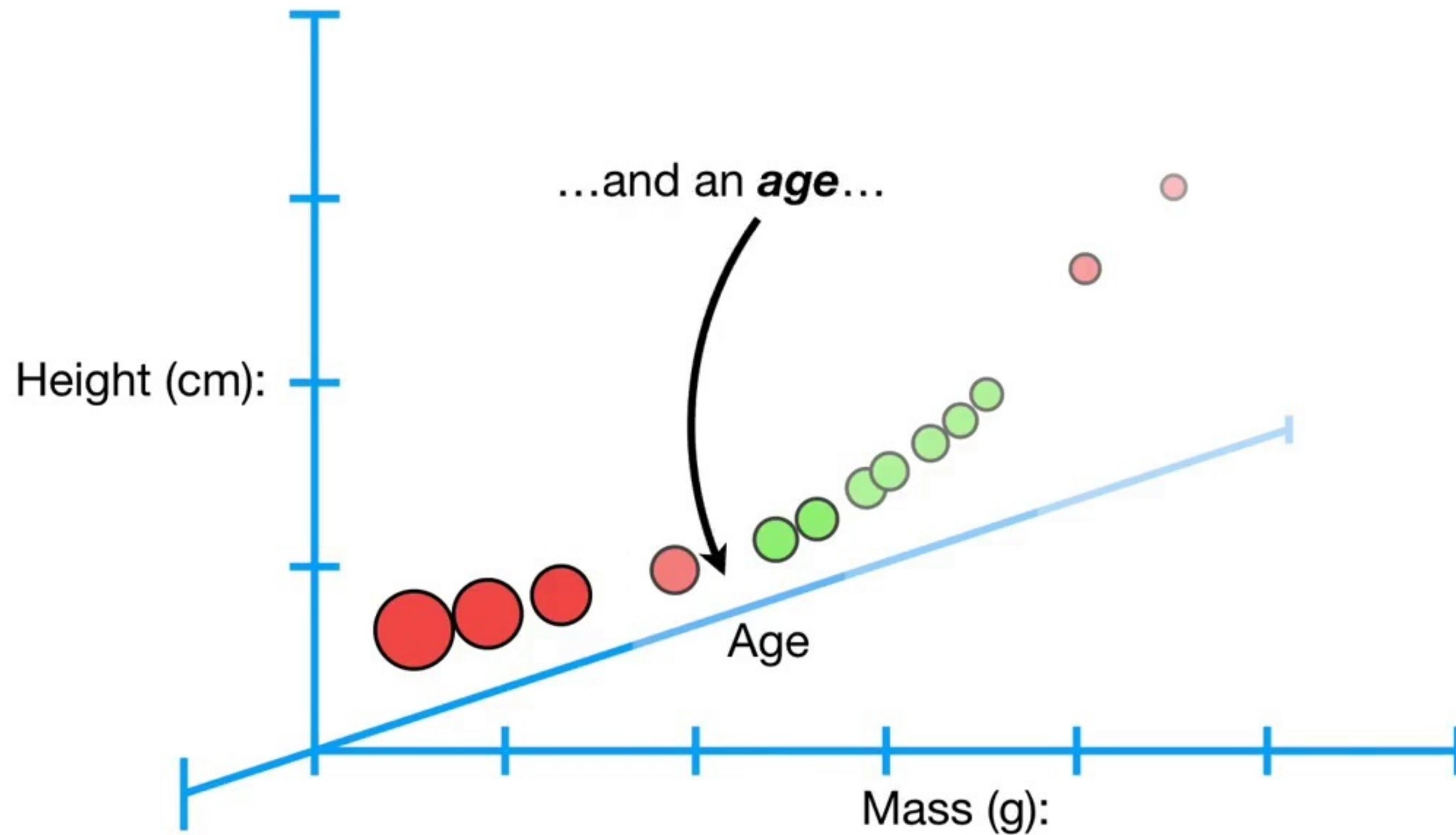


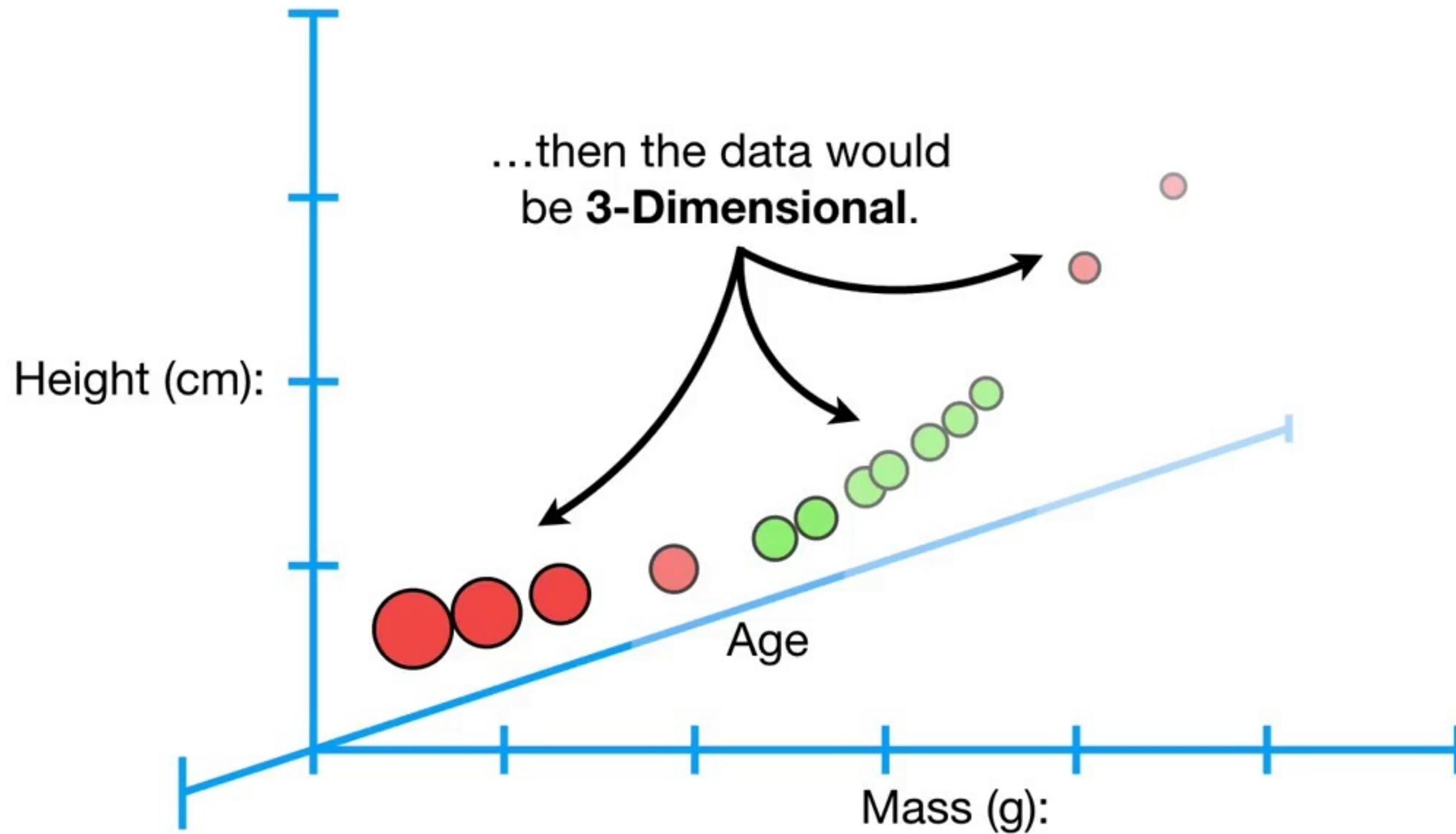
Just like before, we used **Cross Validation** to determine that allowing this misclassification results in better classification in the long run.

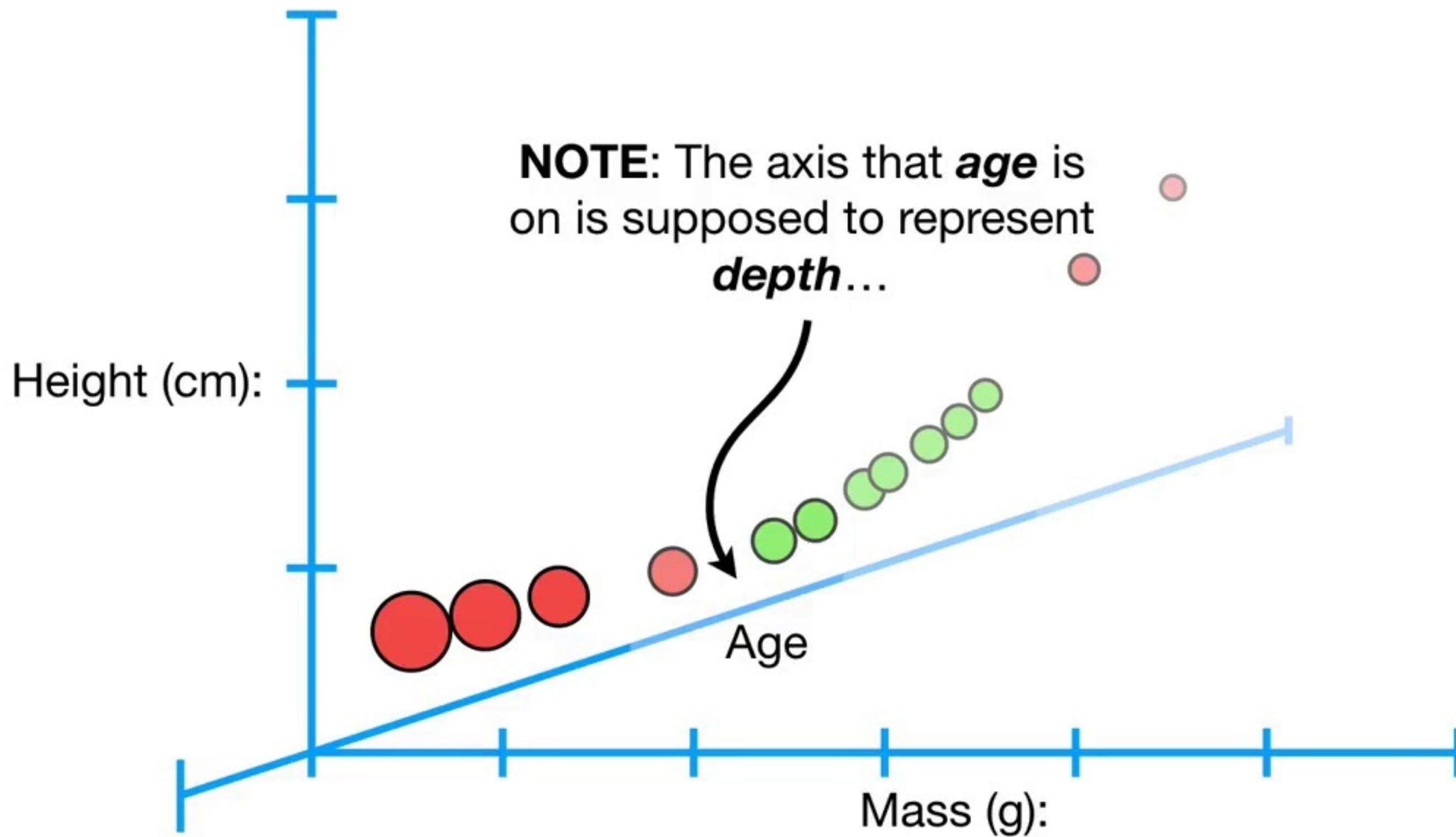


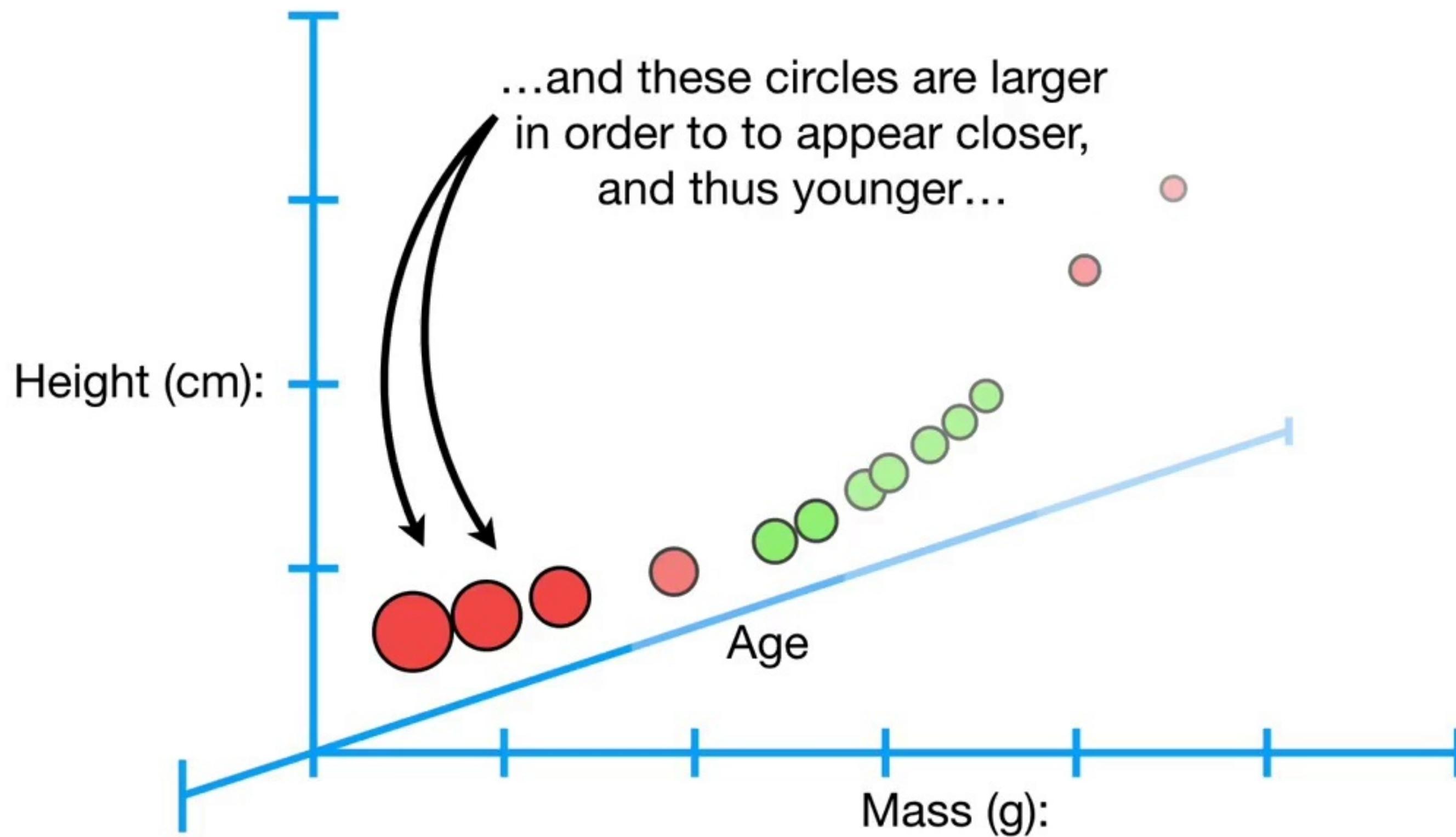


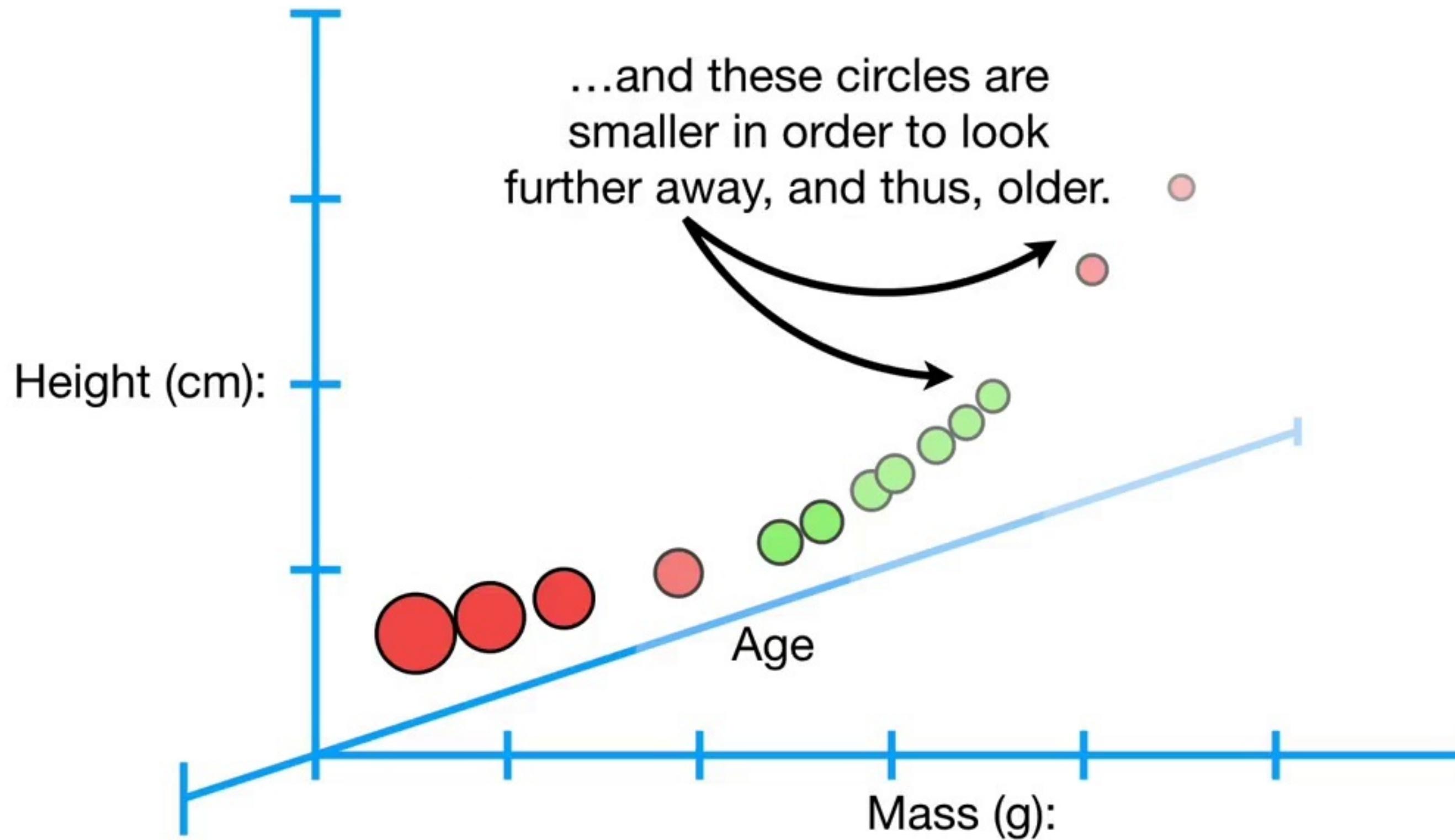




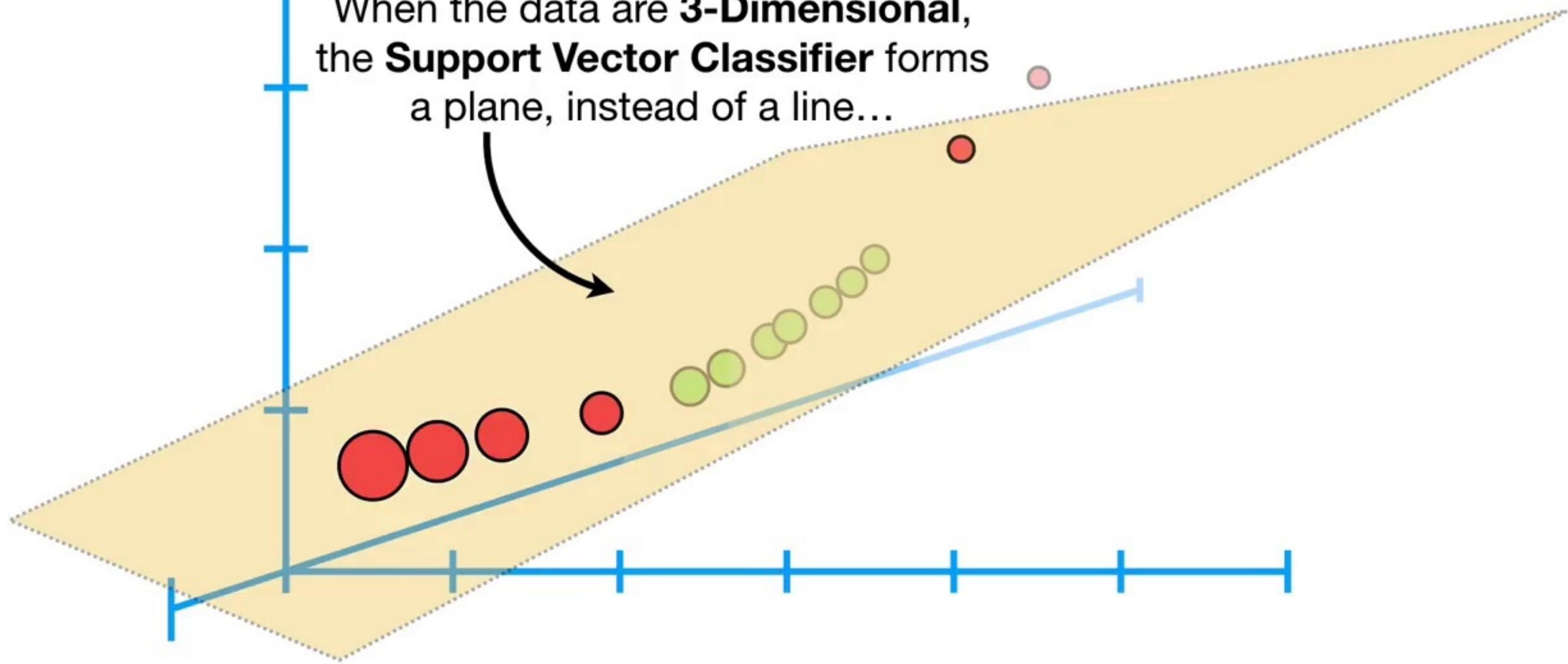


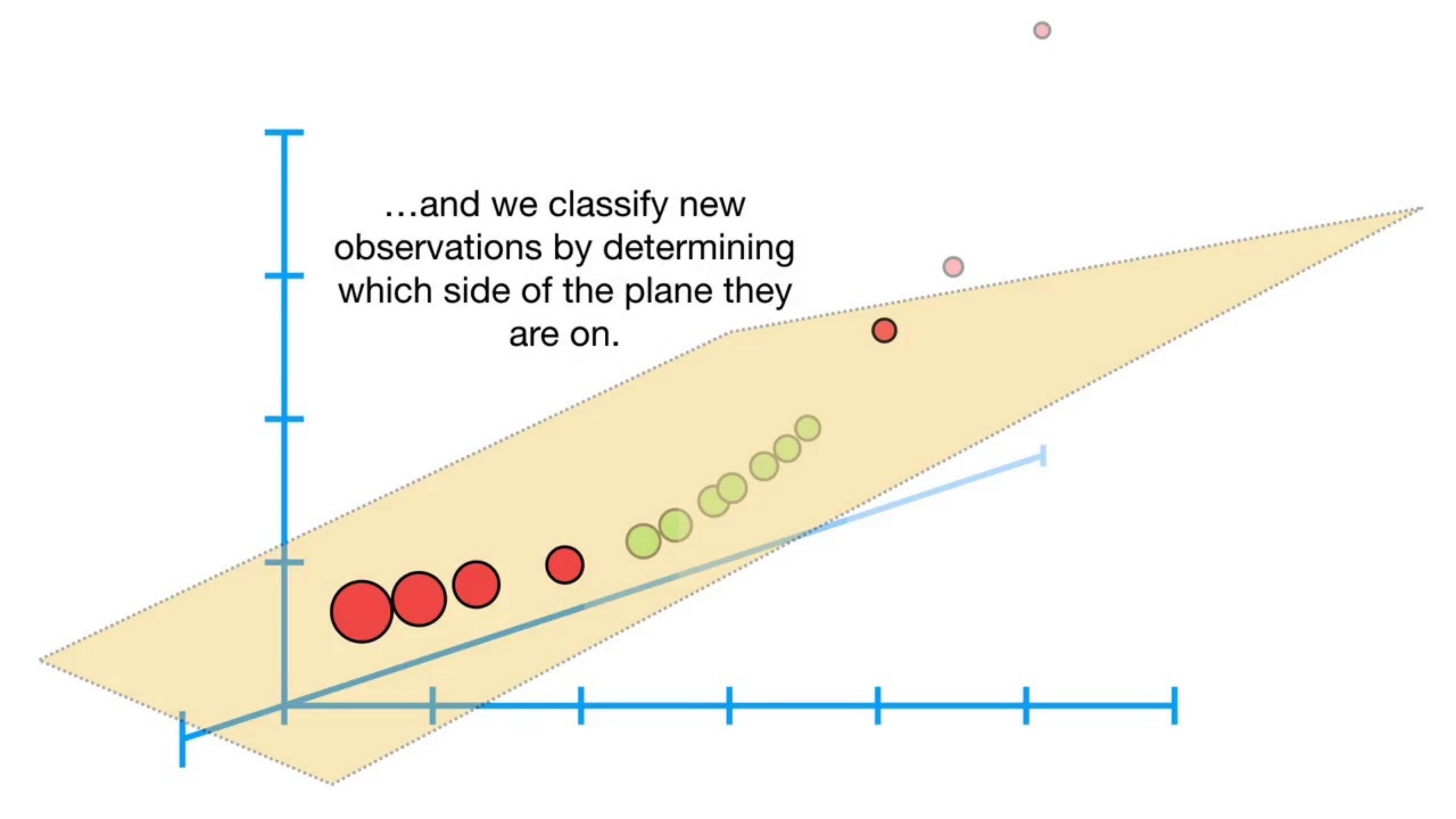






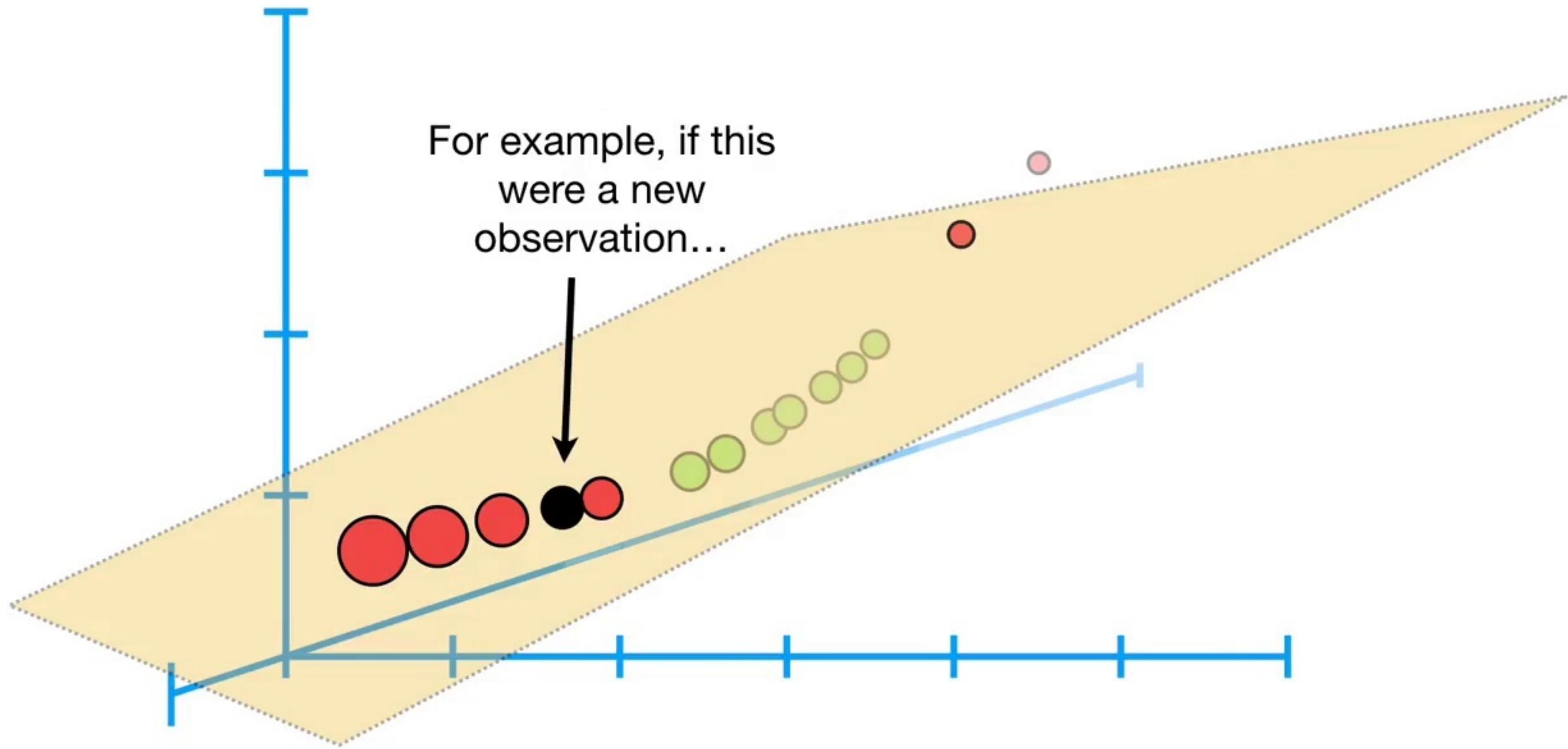
When the data are **3-Dimensional**,
the **Support Vector Classifier** forms
a plane, instead of a line...



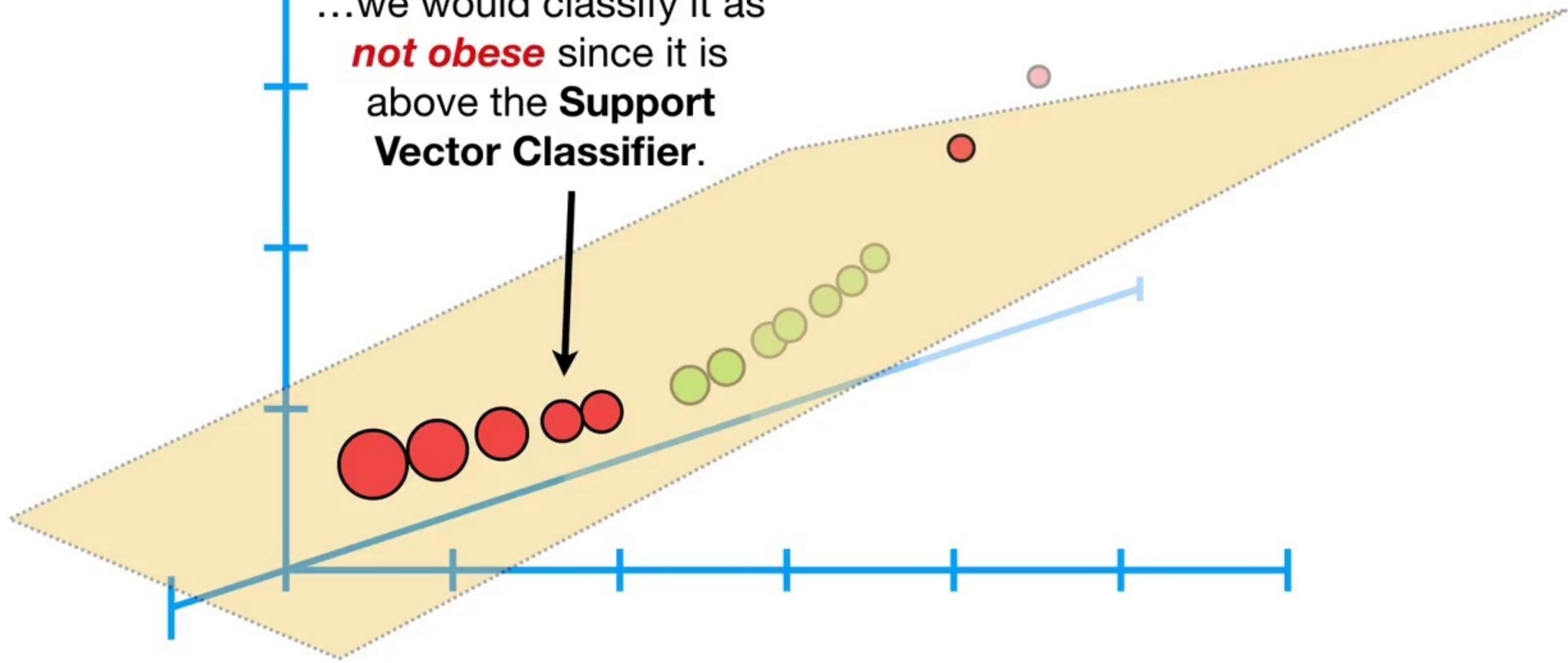


...and we classify new observations by determining which side of the plane they are on.

For example, if this
were a new
observation...

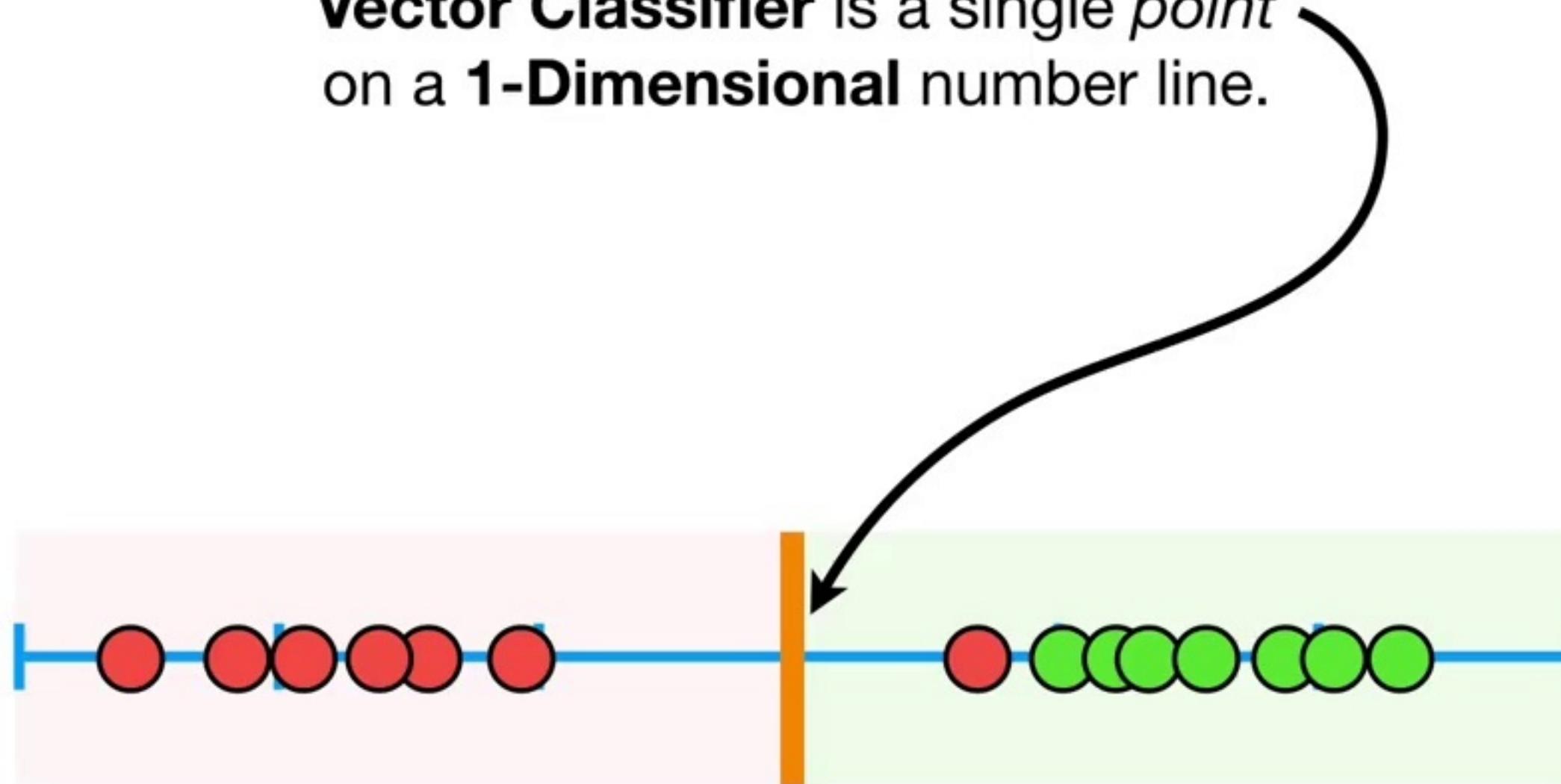


...we would classify it as
not obese since it is
above the **Support
Vector Classifier**.



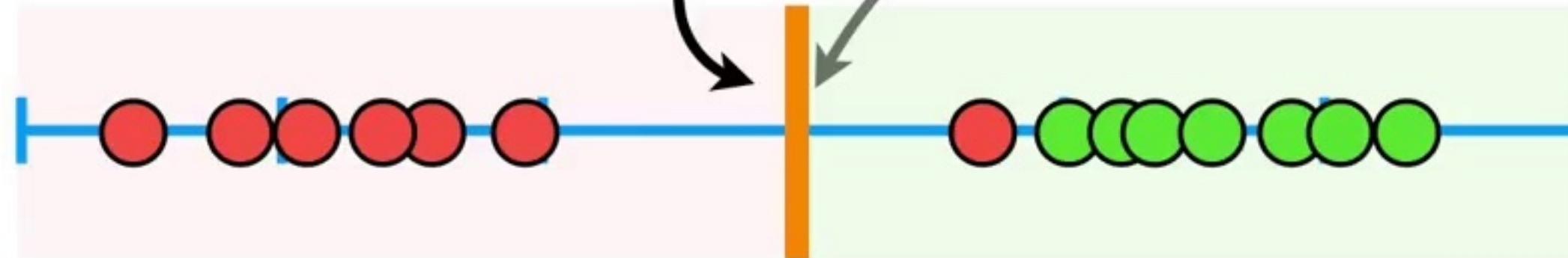
NOTE: If we measured ***mass***,
height, ***age*** and ***blood pressure***,
then the data would be in **4
Dimensions...**

But we know that when the data are **1-Dimensional**, the **Support Vector Classifier** is a single *point* on a **1-Dimensional** number line.

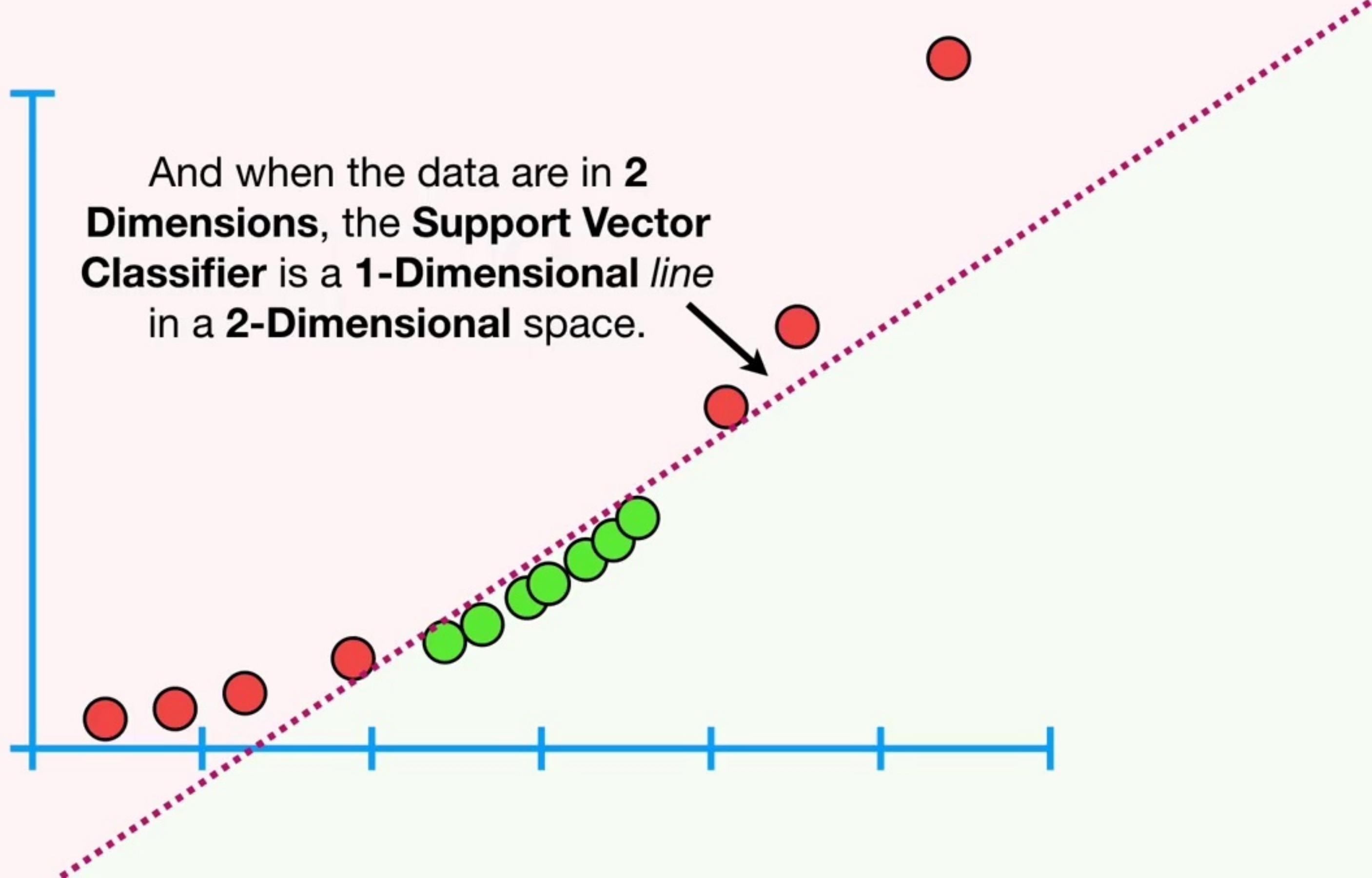


But we know that when the data are **1-Dimensional**, the **Support Vector Classifier** is a single *point* on a **1-Dimensional** number line.

psst! In mathematical jargon, a point is a “flat affine **0-Dimensional** subspace”.

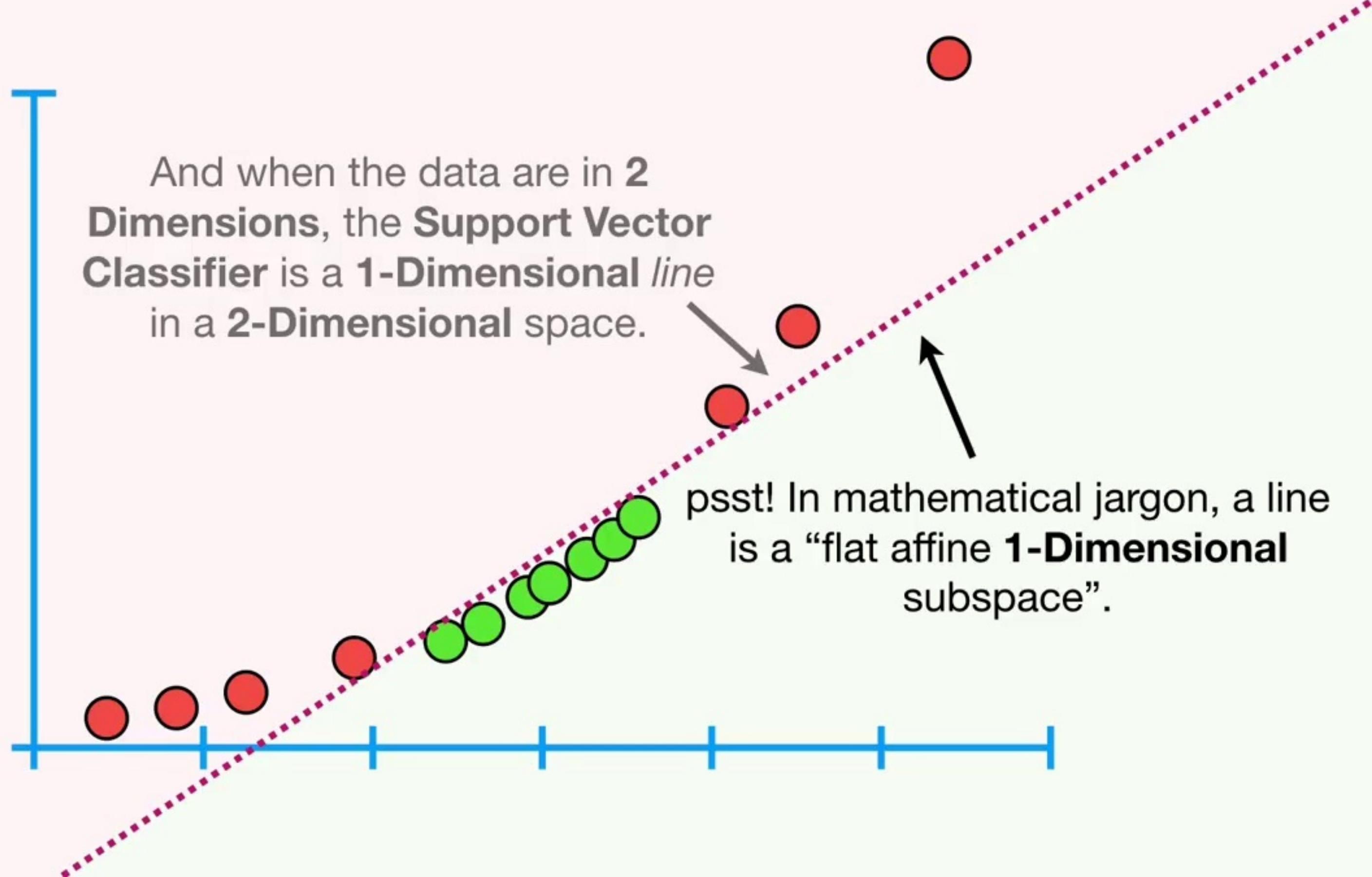


And when the data are in **2 Dimensions**, the **Support Vector Classifier** is a **1-Dimensional line** in a **2-Dimensional space**.

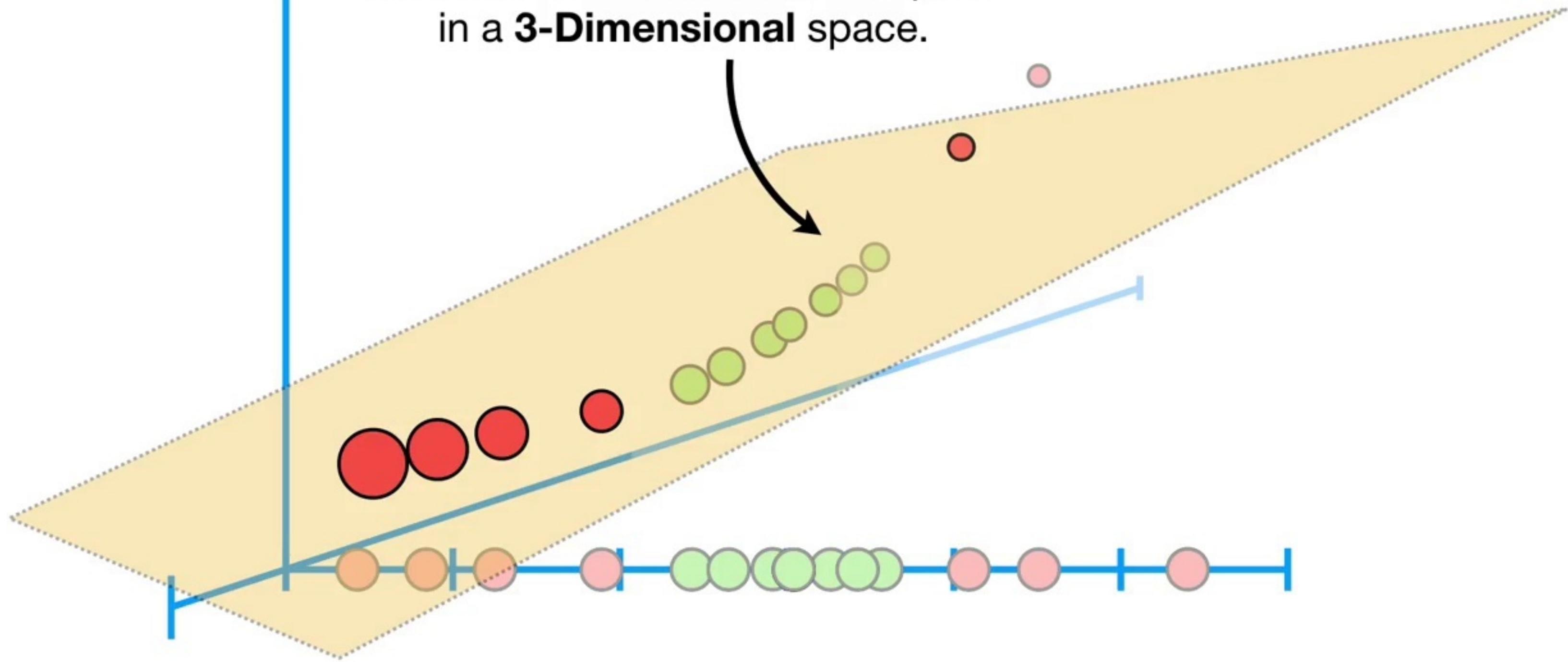


And when the data are in **2 Dimensions**, the **Support Vector Classifier** is a **1-Dimensional line** in a **2-Dimensional space**.

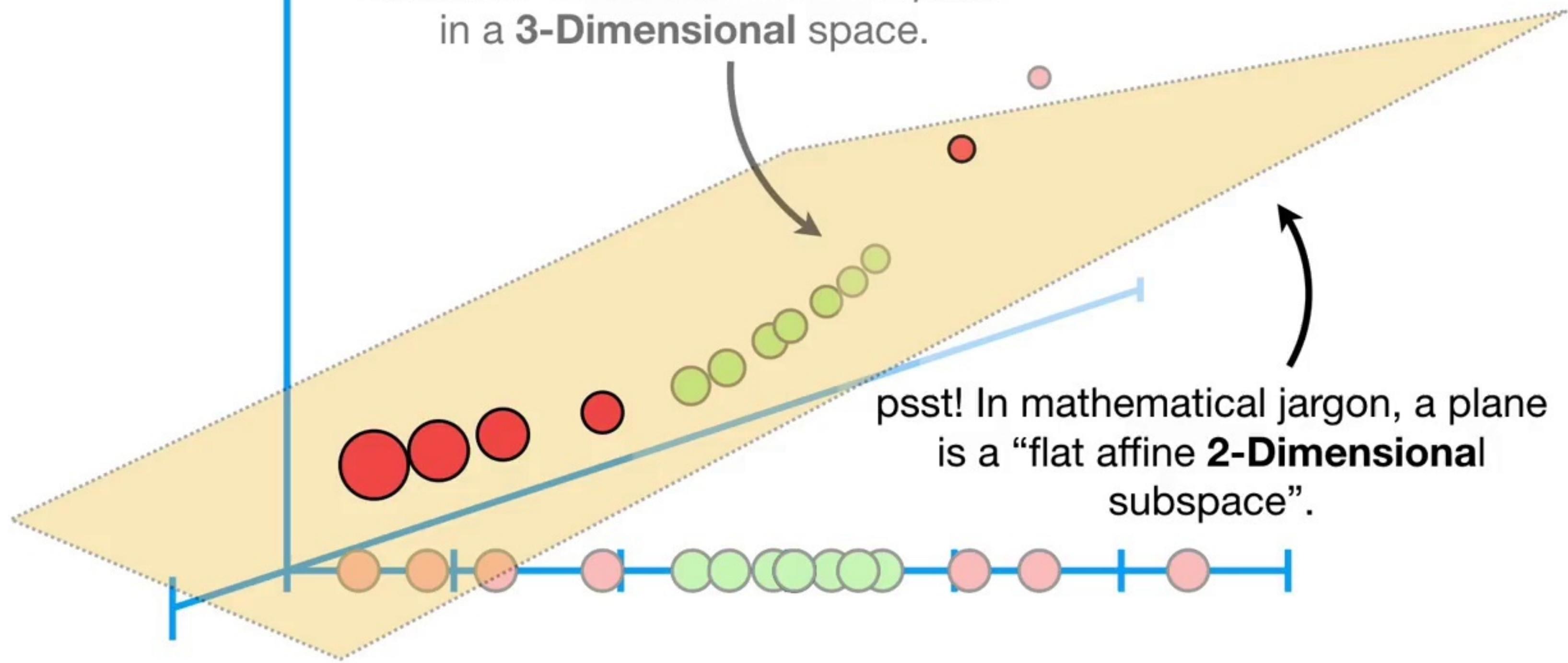
psst! In mathematical jargon, a line is a “flat affine **1-Dimensional subspace**”.



And when the data are 3-Dimensional, the **Support Vector Classifier** is a 2-Dimensional plane in a 3-Dimensional space.



And when the data are 3-Dimensional, the **Support Vector Classifier** is a 2-Dimensional *plane* in a 3-Dimensional space.



And when the data are in **4 or more Dimensions**, the **Support Vector Classifier** is a *hyperplane*.

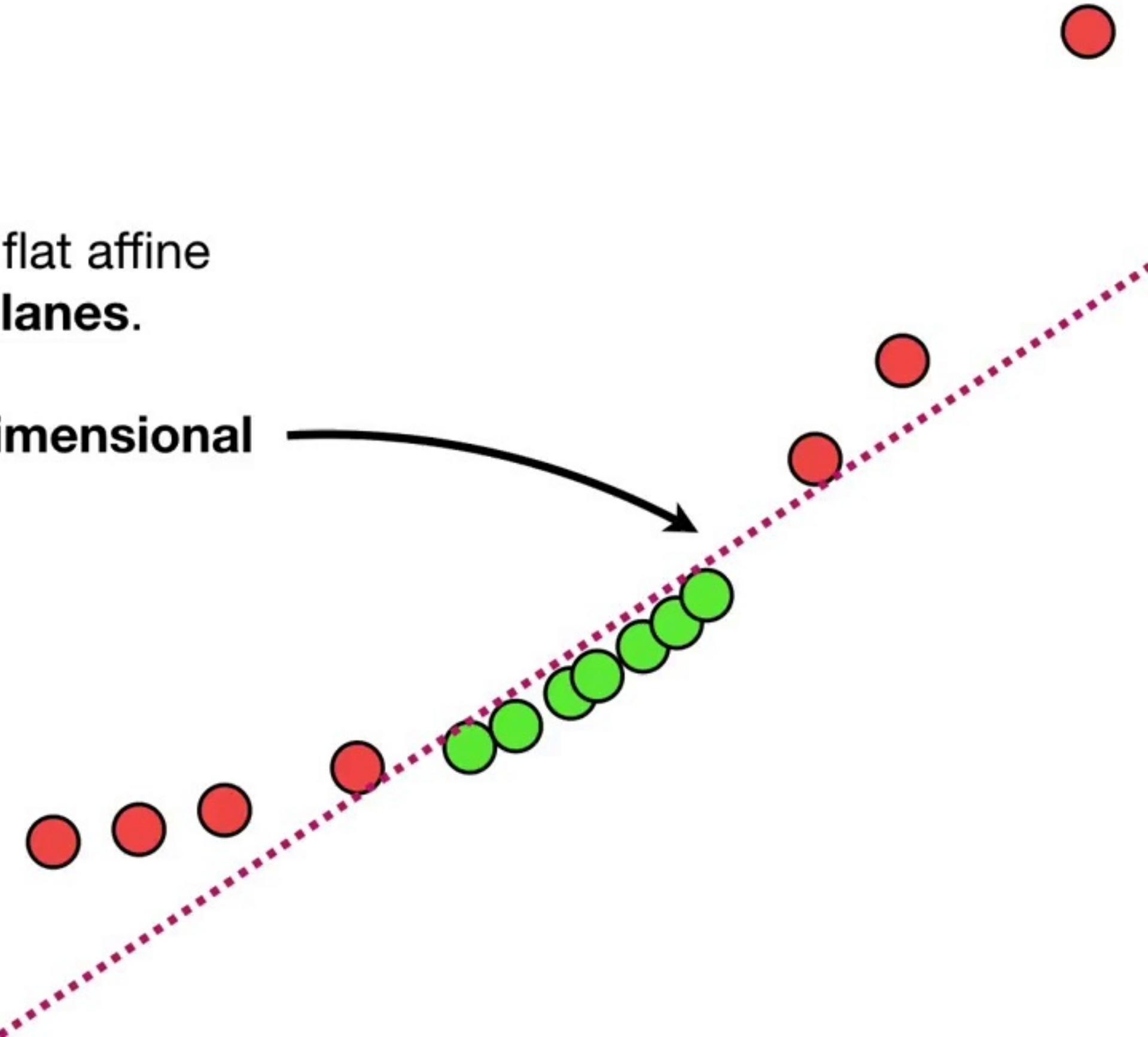
And when the data are in **4 or more Dimensions**, the **Support Vector Classifier** is a *hyperplane*.

psst! In mathematical jargon, a hyperplane is a “flat affine subspace”.

NOTE: Technically speaking, all flat affine subspaces are called **hyperplanes**.

NOTE: Technically speaking, all flat affine subspaces are called **hyperplanes**.

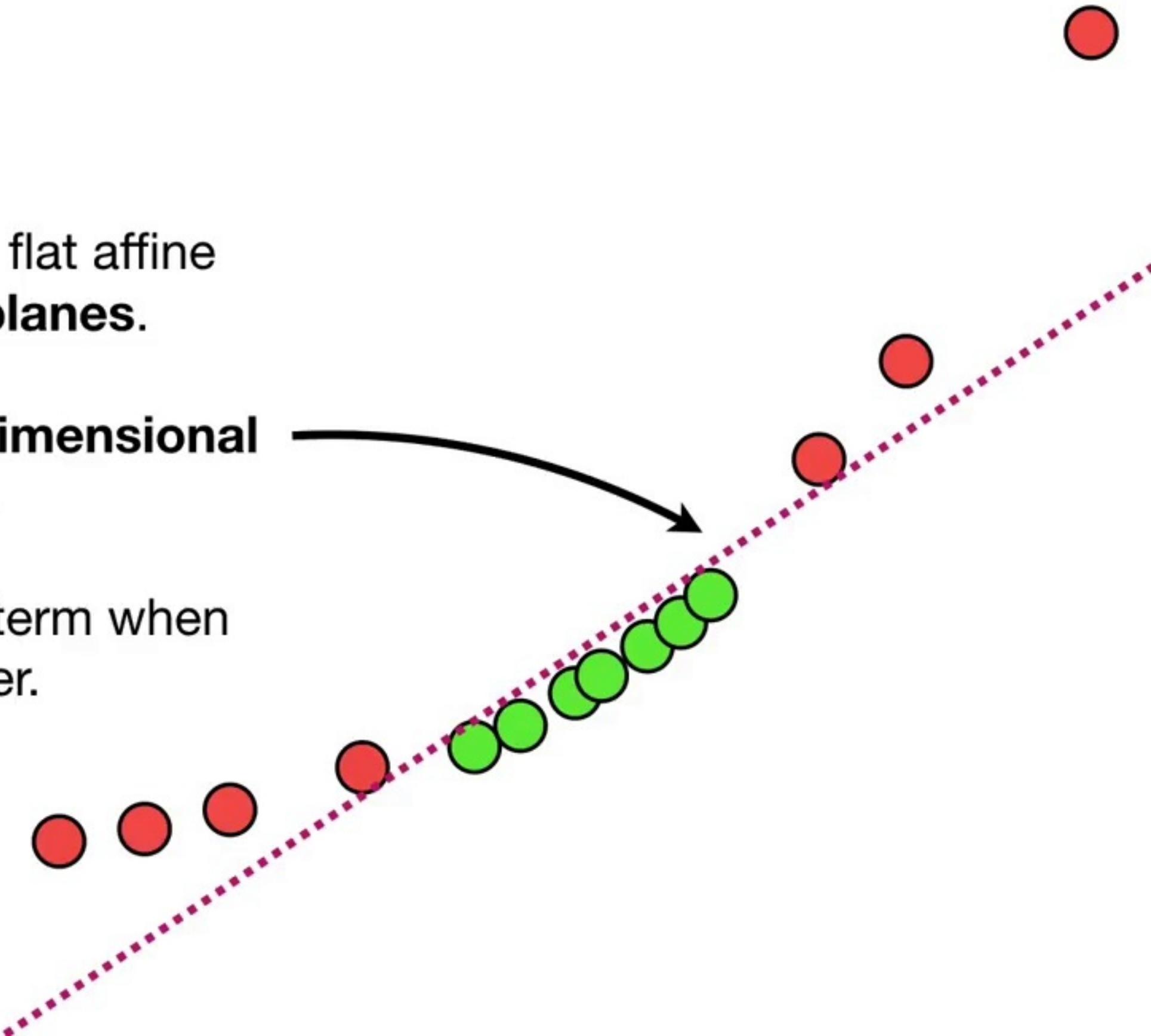
So, technically speaking, this **1-Dimensional line** is a **hyperplane**...



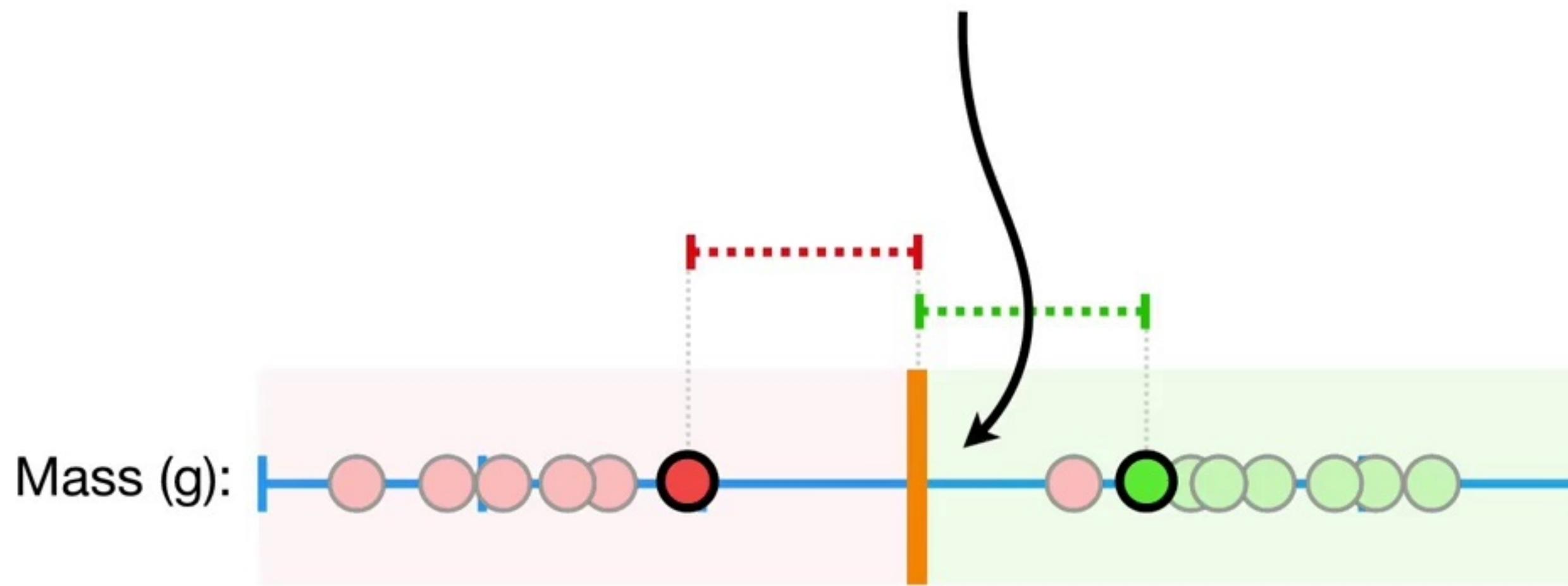
NOTE: Technically speaking, all flat affine subspaces are called **hyperplanes**.

So, technically speaking, this **1-Dimensional line** is a **hyperplane**...

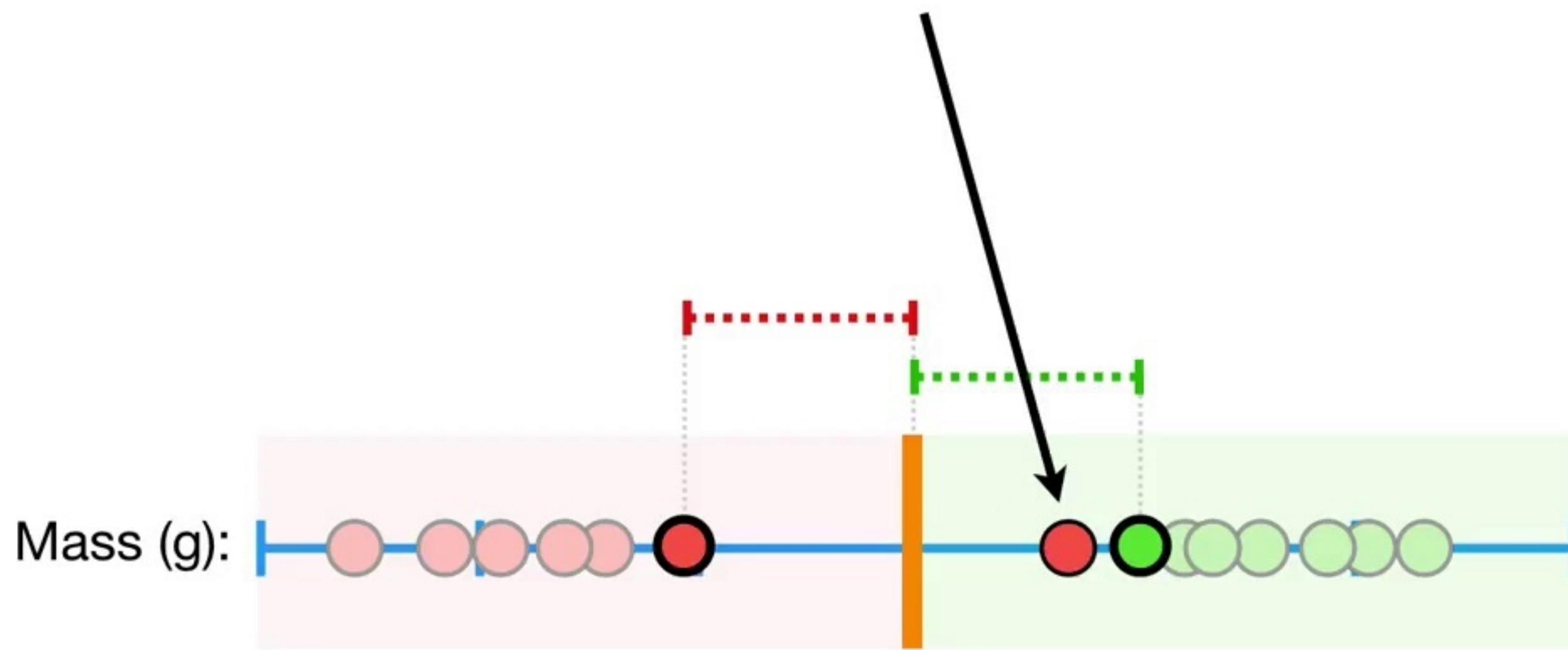
...but we generally only use the term when we can't draw it on paper.



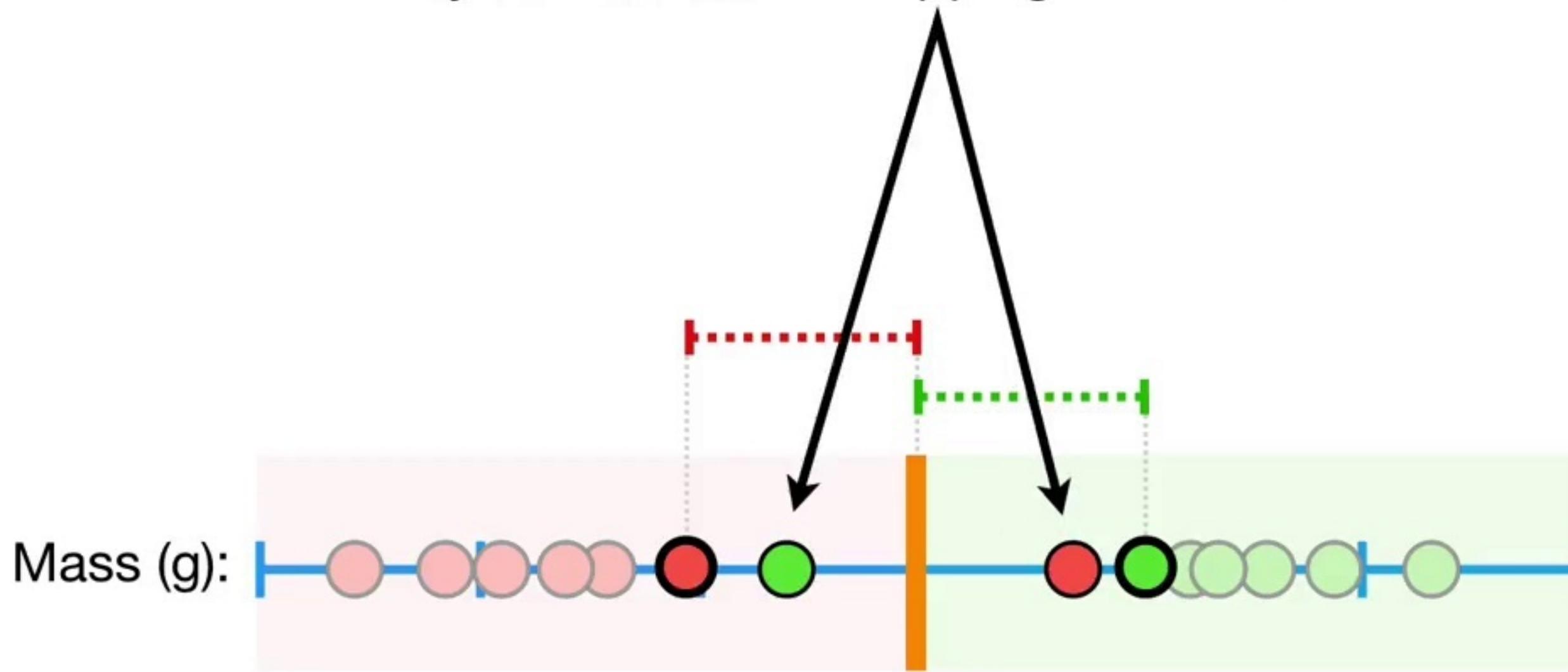
Support Vector Classifiers seem pretty cool
because they can handle...



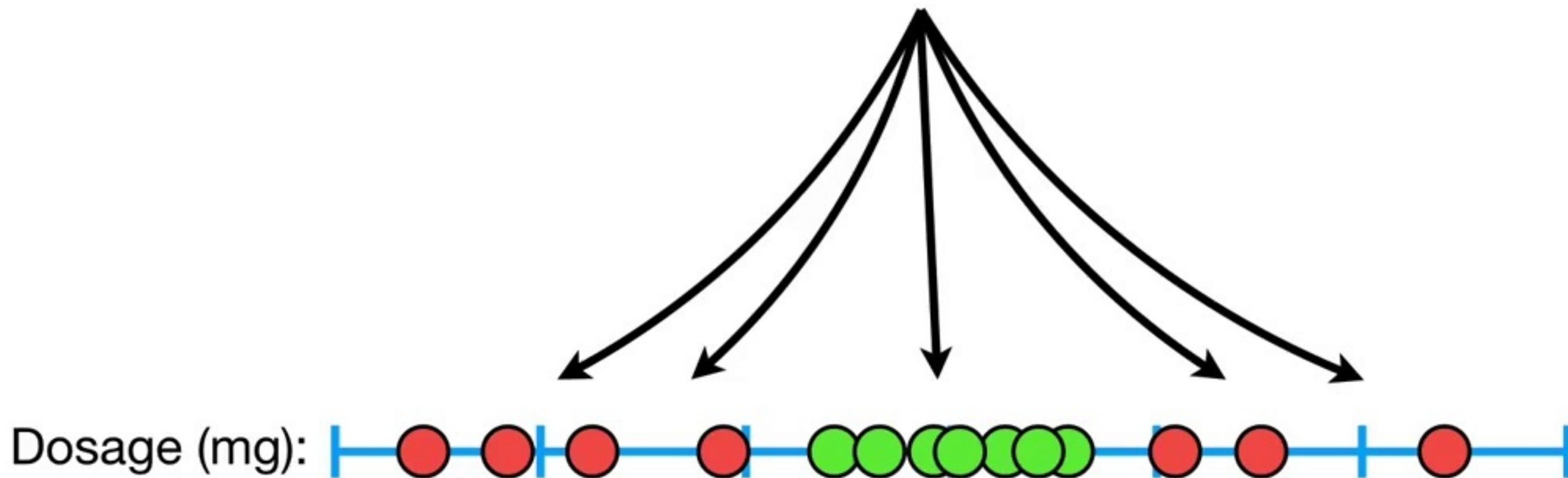
...outliers...



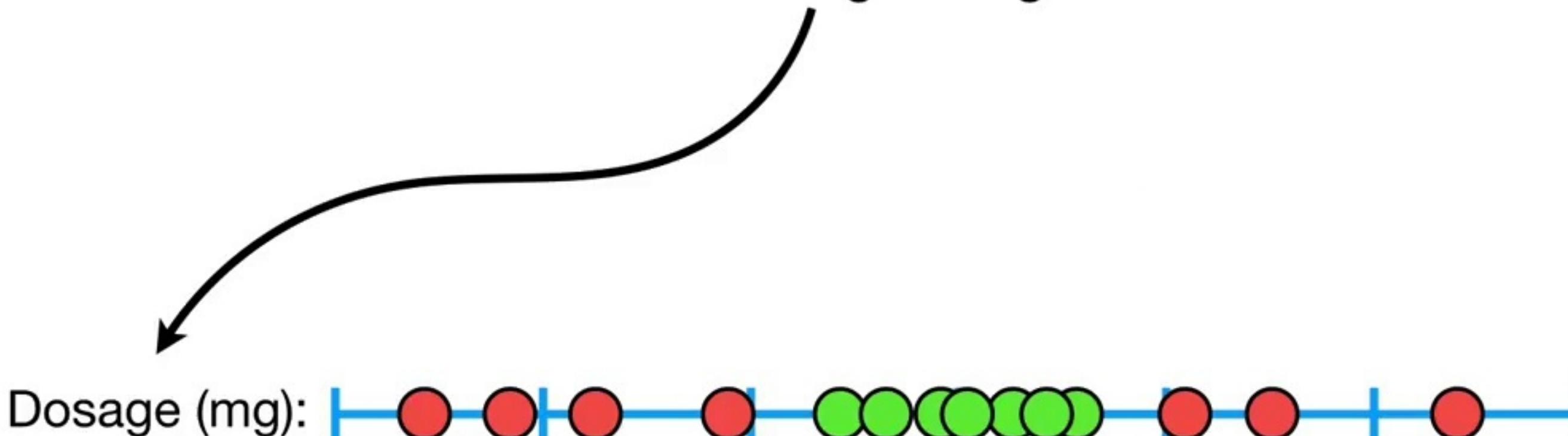
...and, because they allow misclassifications,
they can handle overlapping classifications...



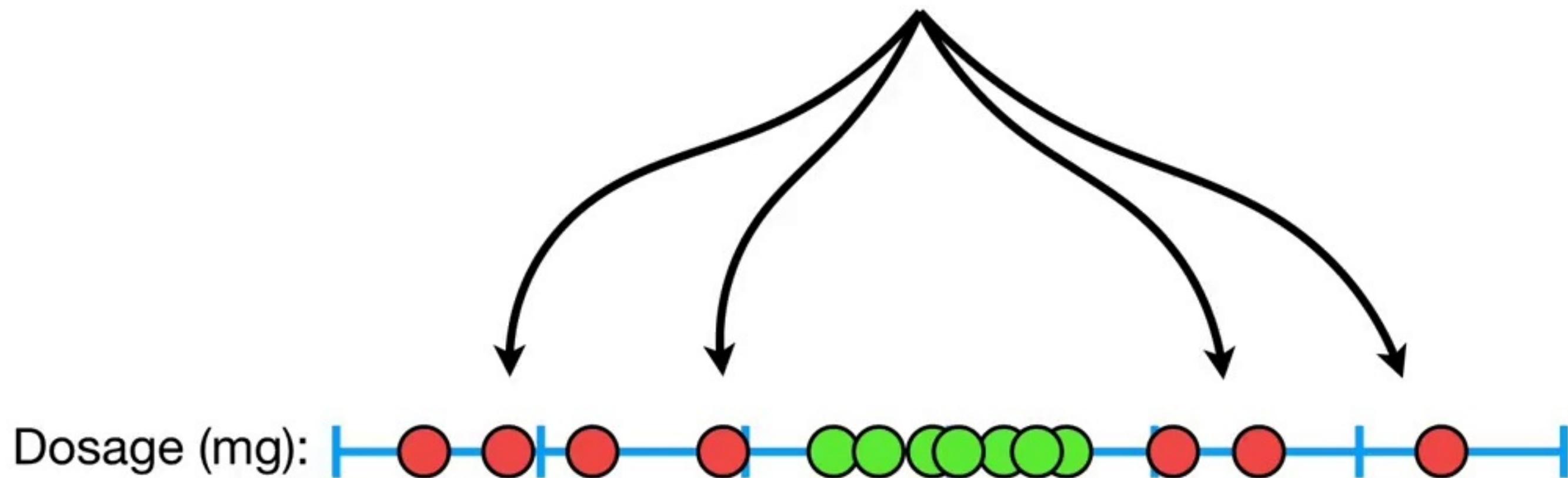
...but what if this was our training
data and we had tons of overlap?



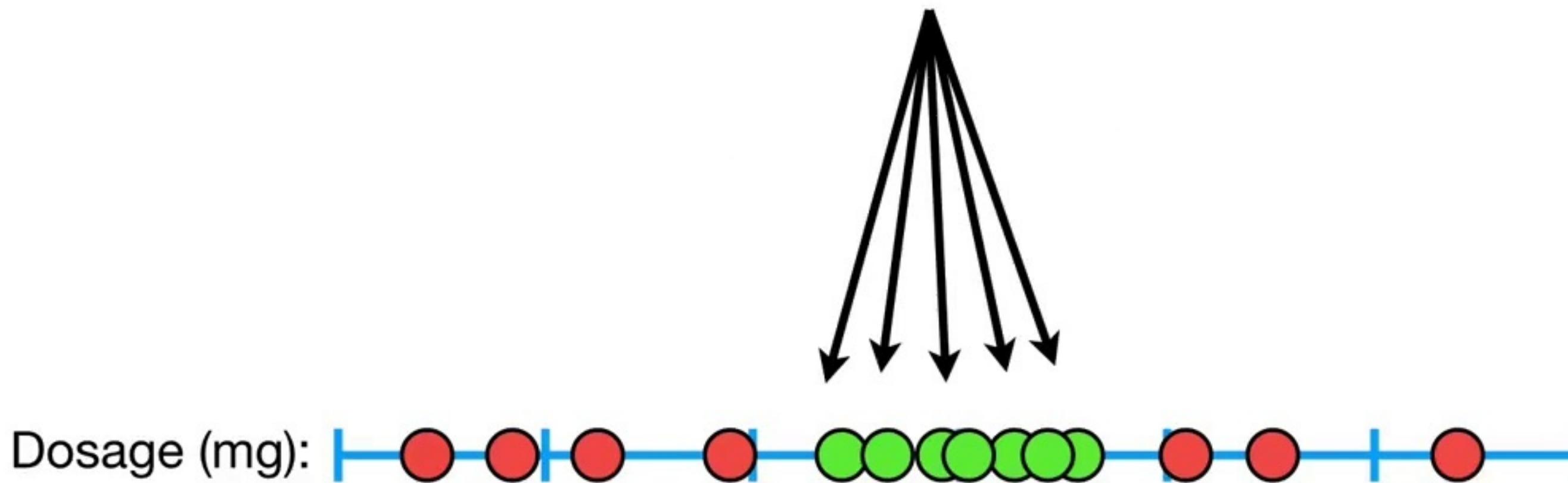
In this new example, with tons of overlap, we are now looking at
Drug Dosages...



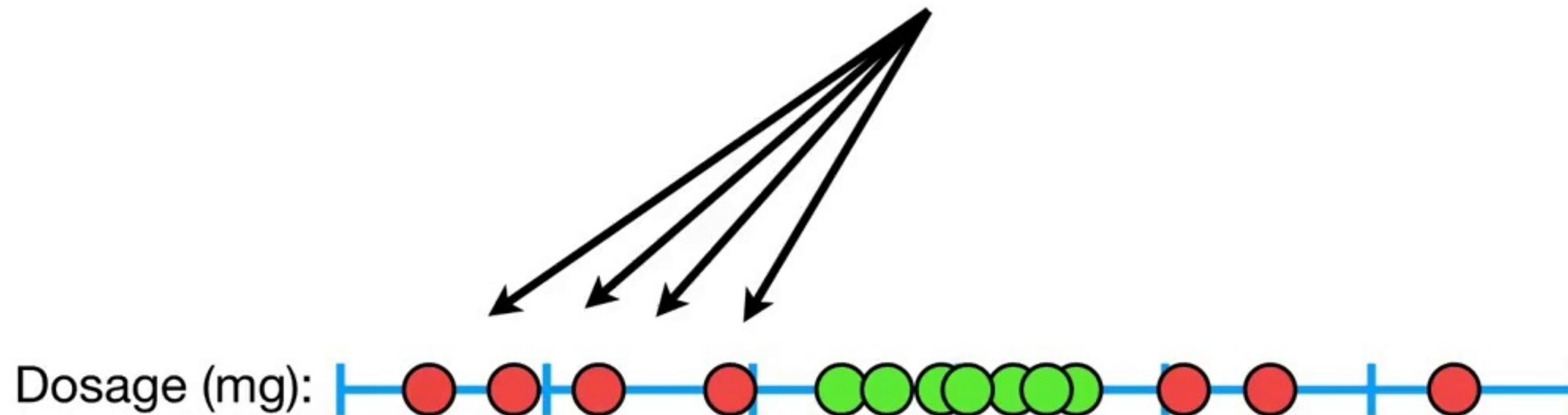
...and the **red dots** represent patients that were ***not cured***...



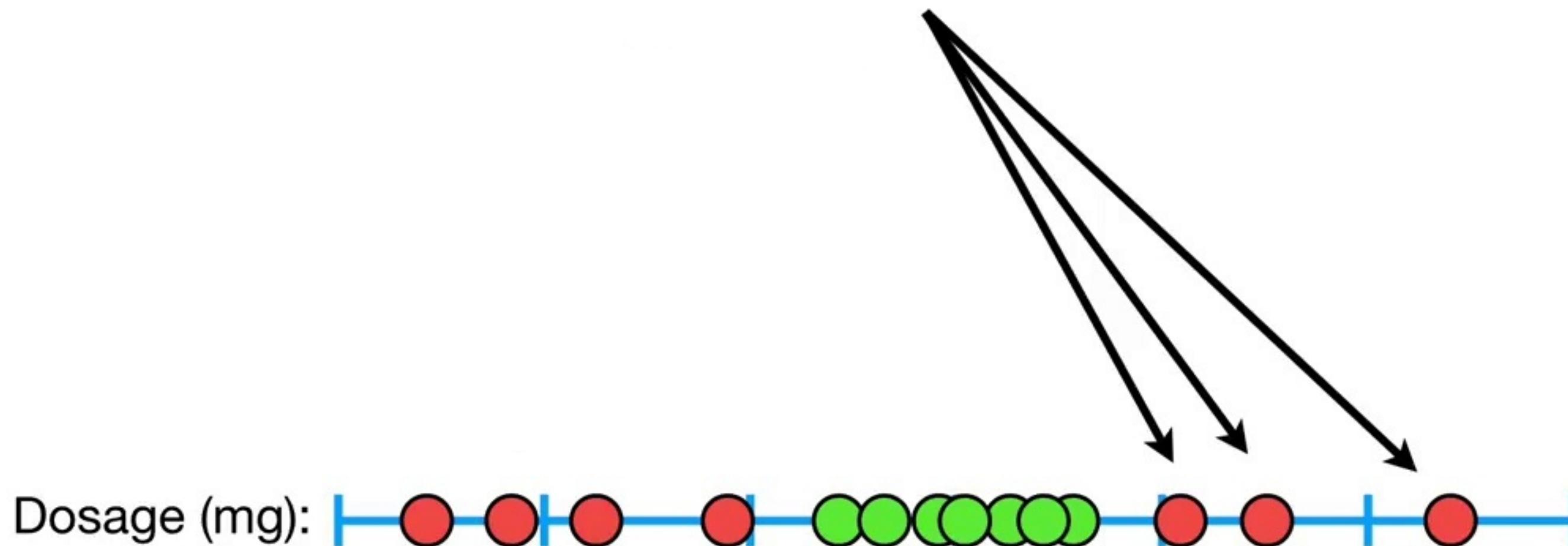
...and the **green dots** represent patients that were **cured**.



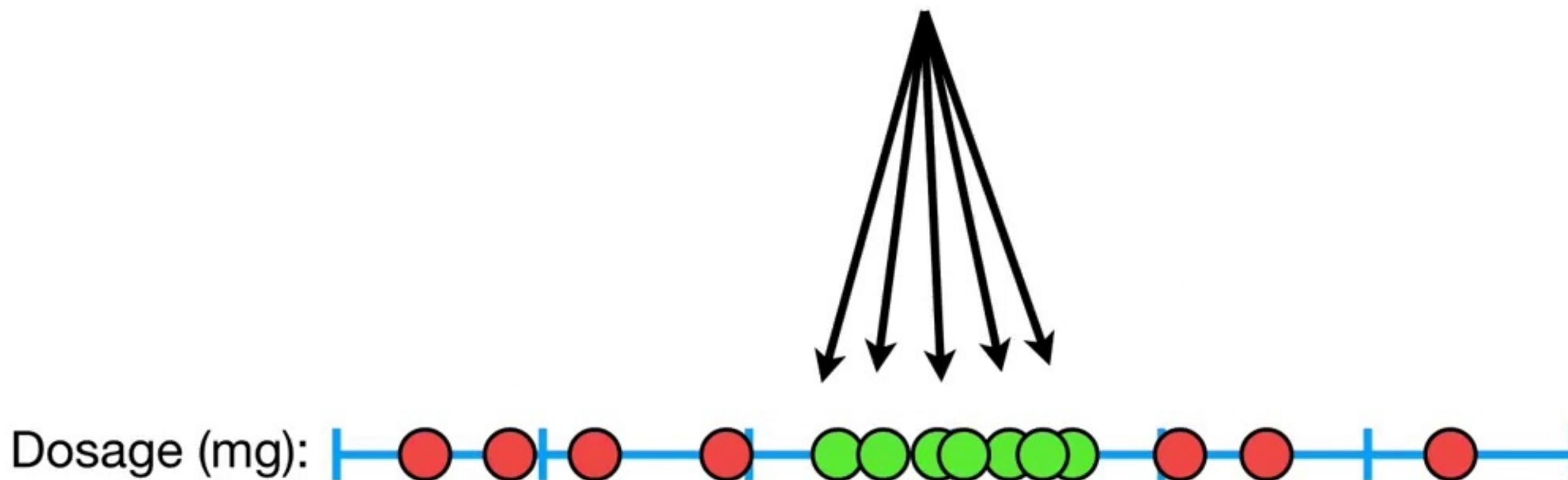
In other words, the drug
doesn't work if the dosage is
too small...



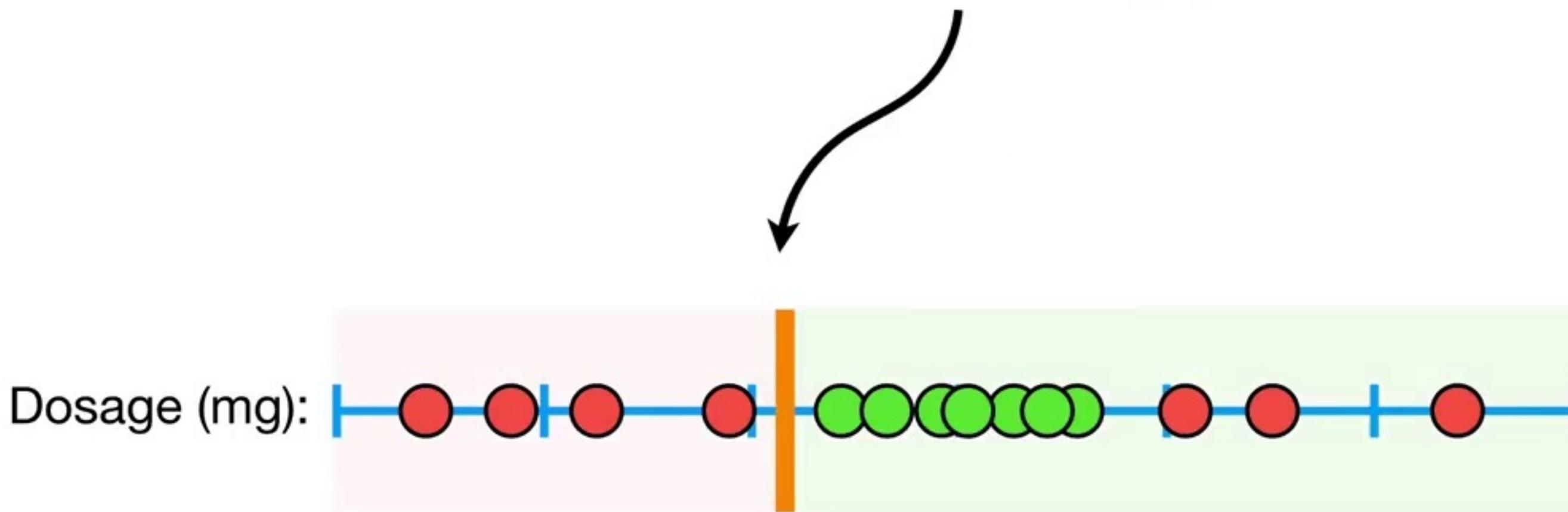
...or too large.



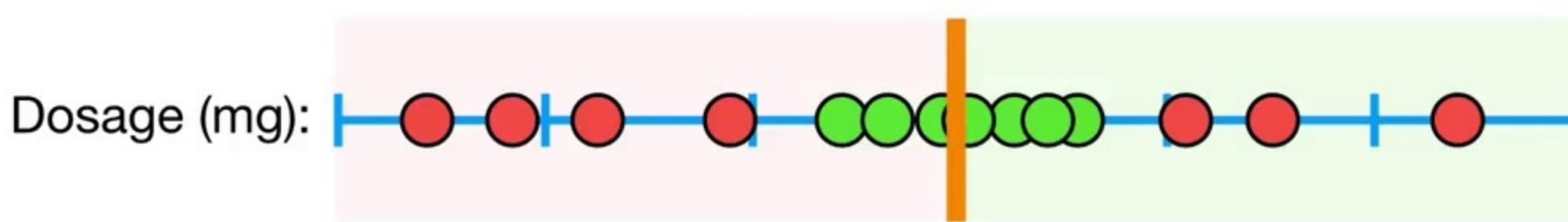
It only works when the dosage is just right.



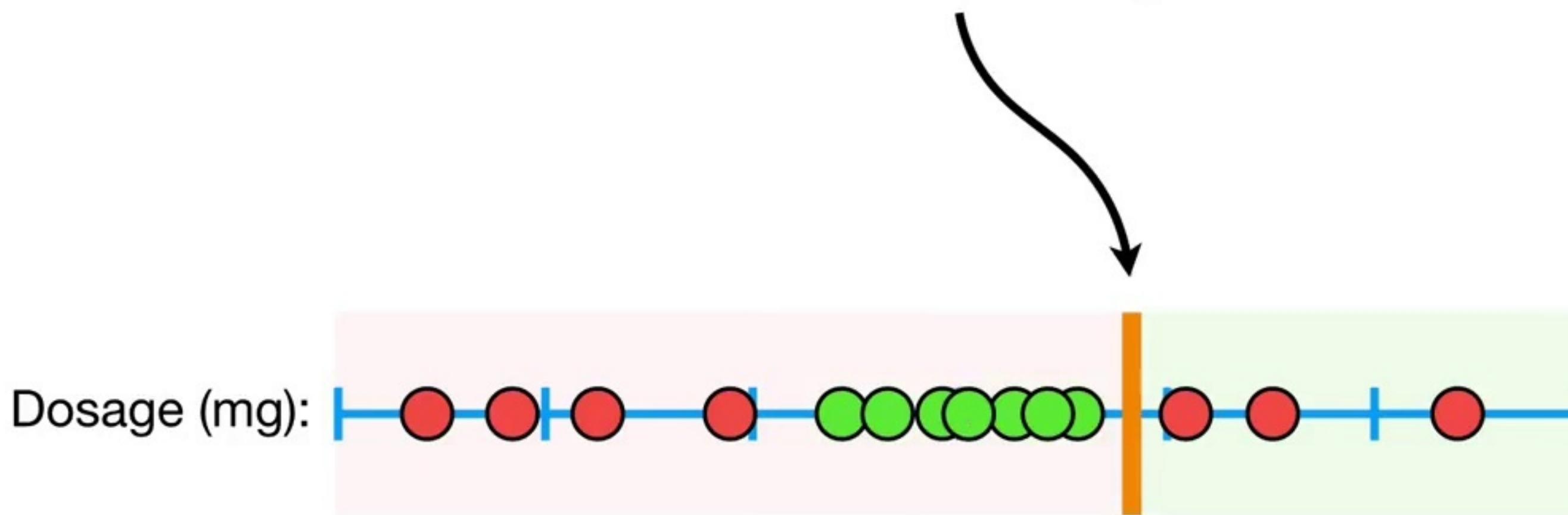
Now, no matter where we put
the classifier, we will make a
lot of misclassifications.



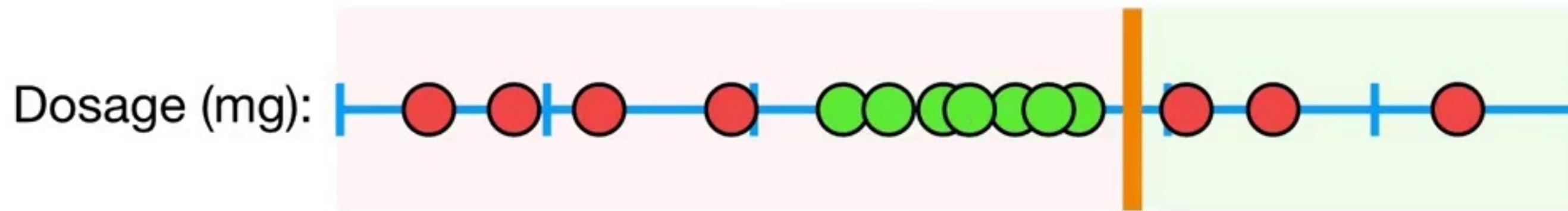
Now, no matter where we put
the classifier, we will make a
lot of misclassifications.



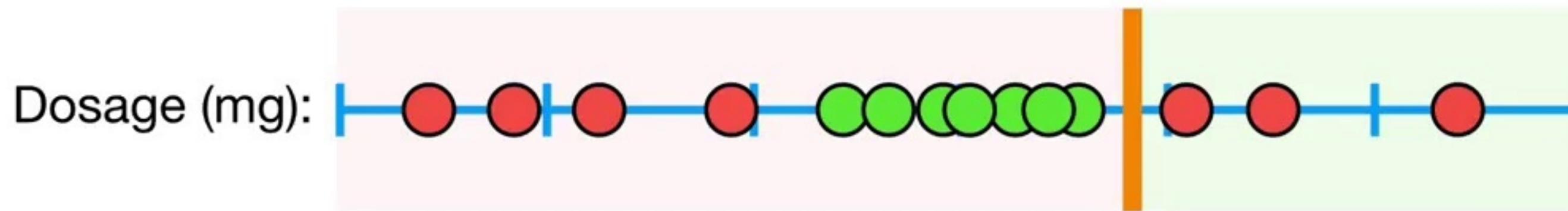
Now, no matter where we put
the classifier, we will make a
lot of misclassifications.



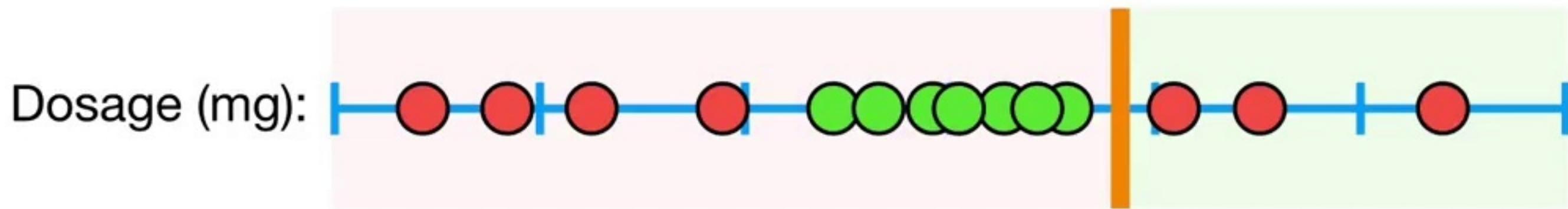
So **Support Vector Classifiers** are only semi-cool, since they don't perform well with this type of data.



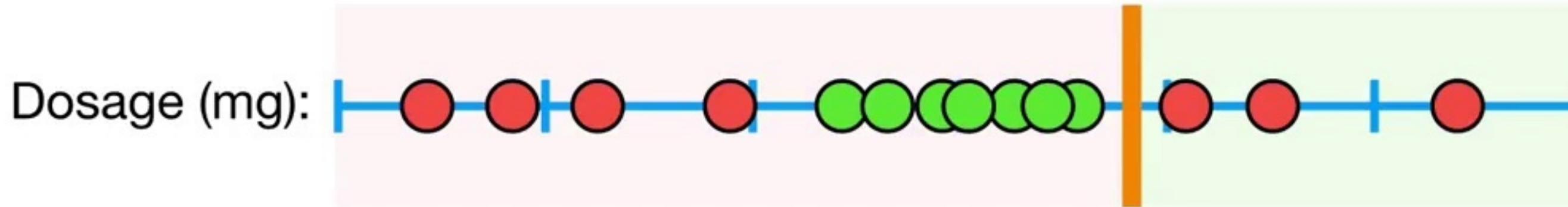
Can we do better than **Maximal Margin Classifiers** and **Support Vector Classifiers**?



YES!!!



Since **Maximal Margin Classifiers** and
Support Vector Classifiers can't
handle this data, it's high time we talked
about...

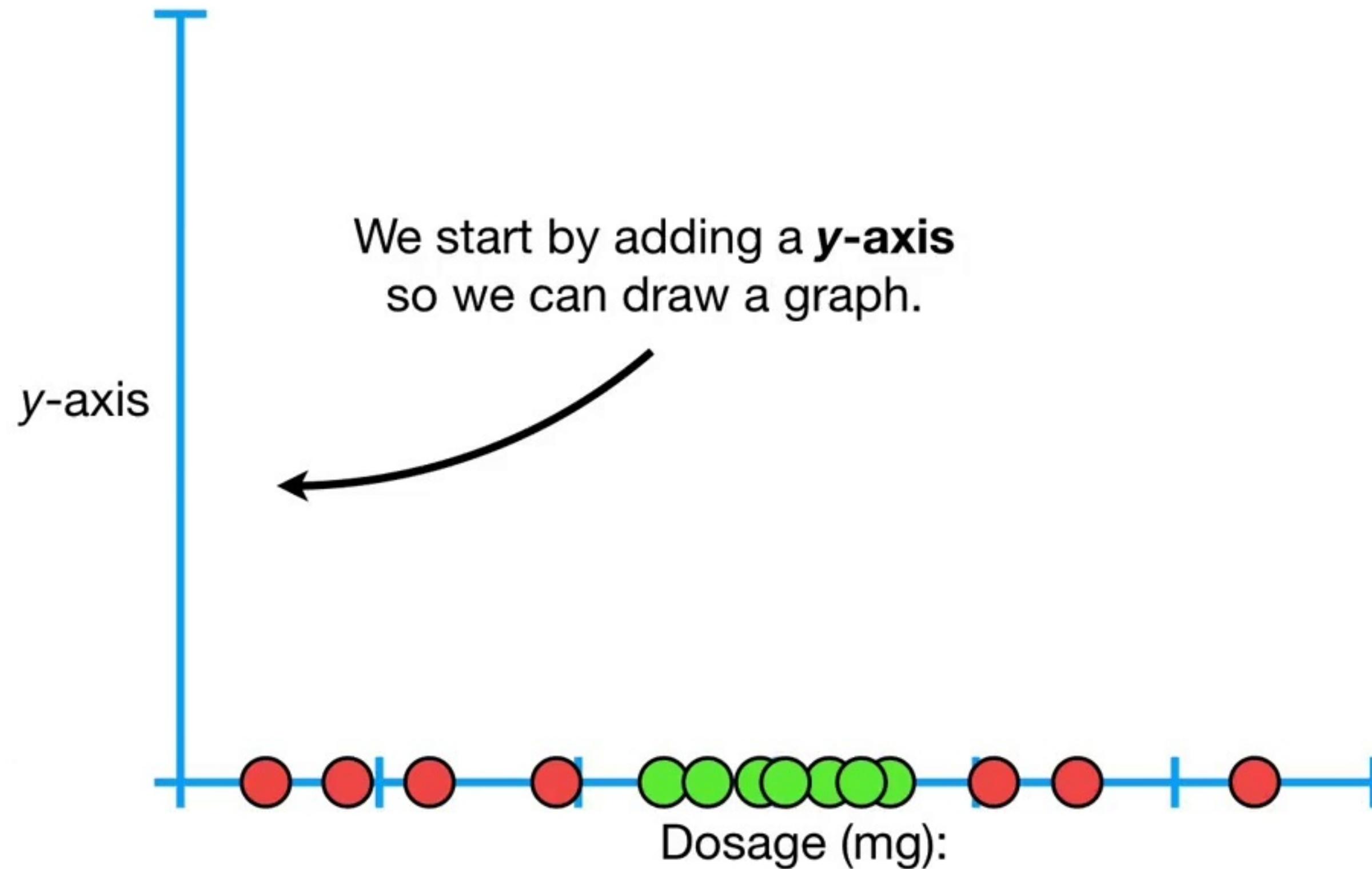


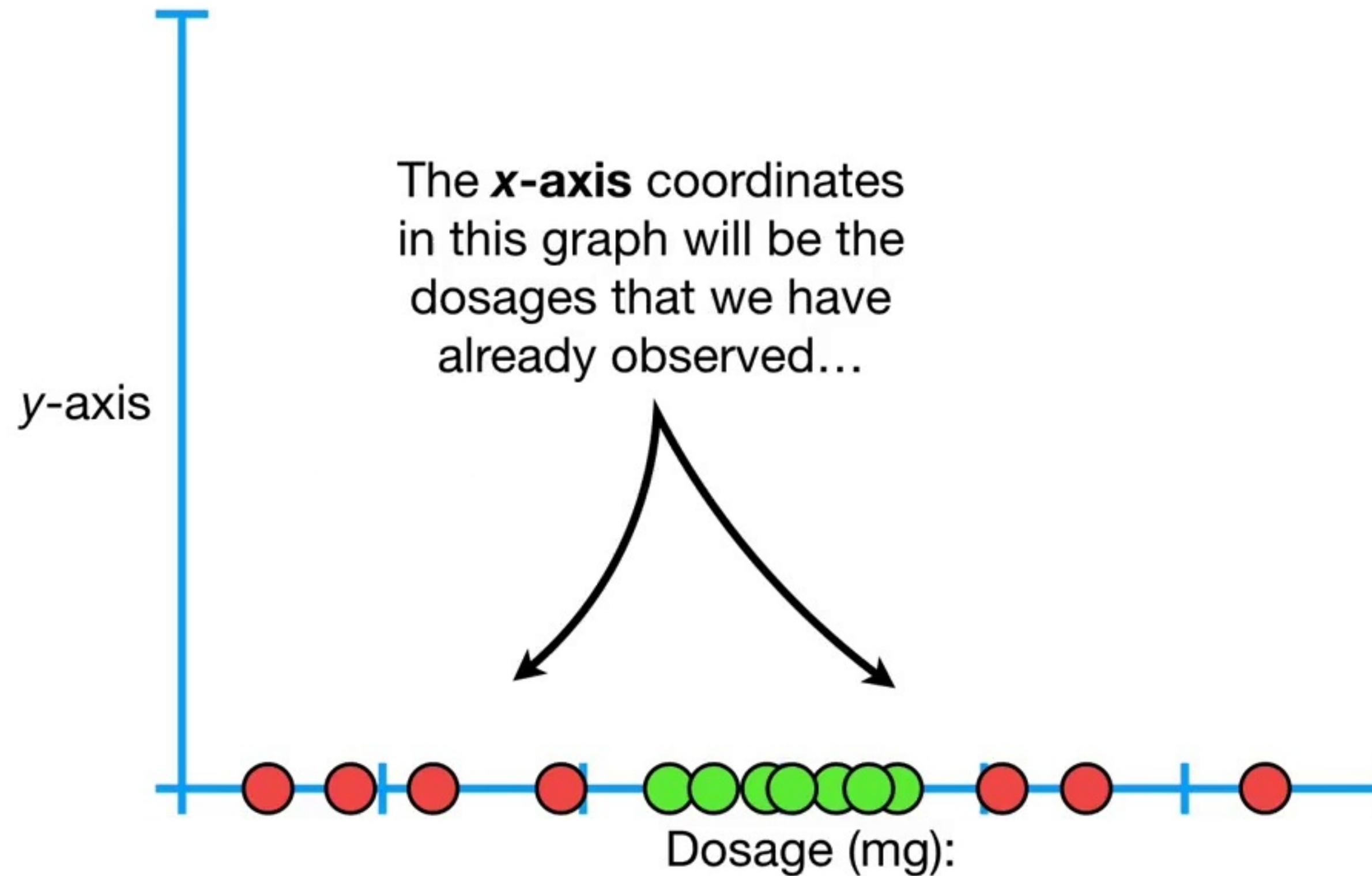
Support Vector Machines!!!

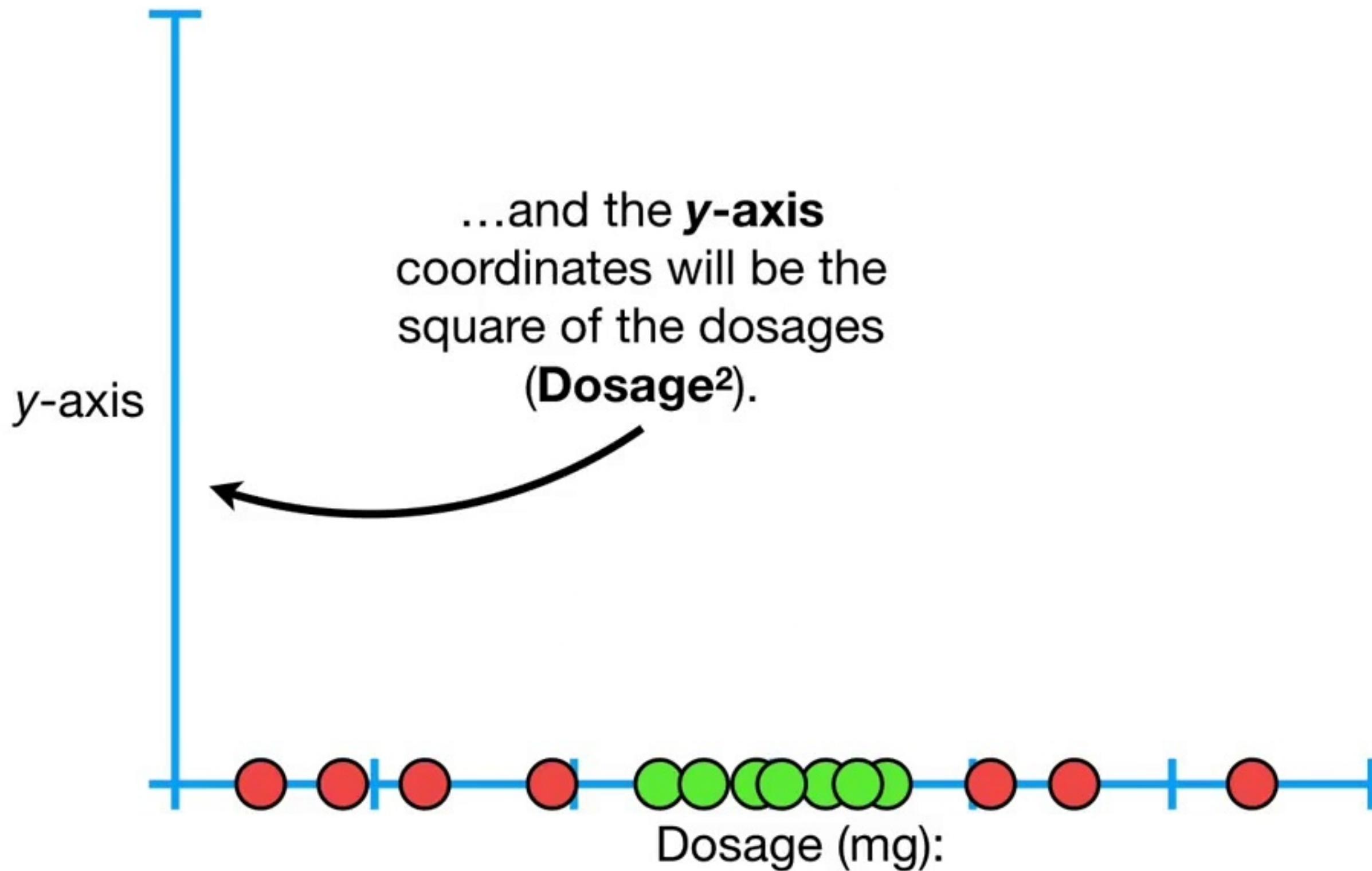
Dosage (mg): 

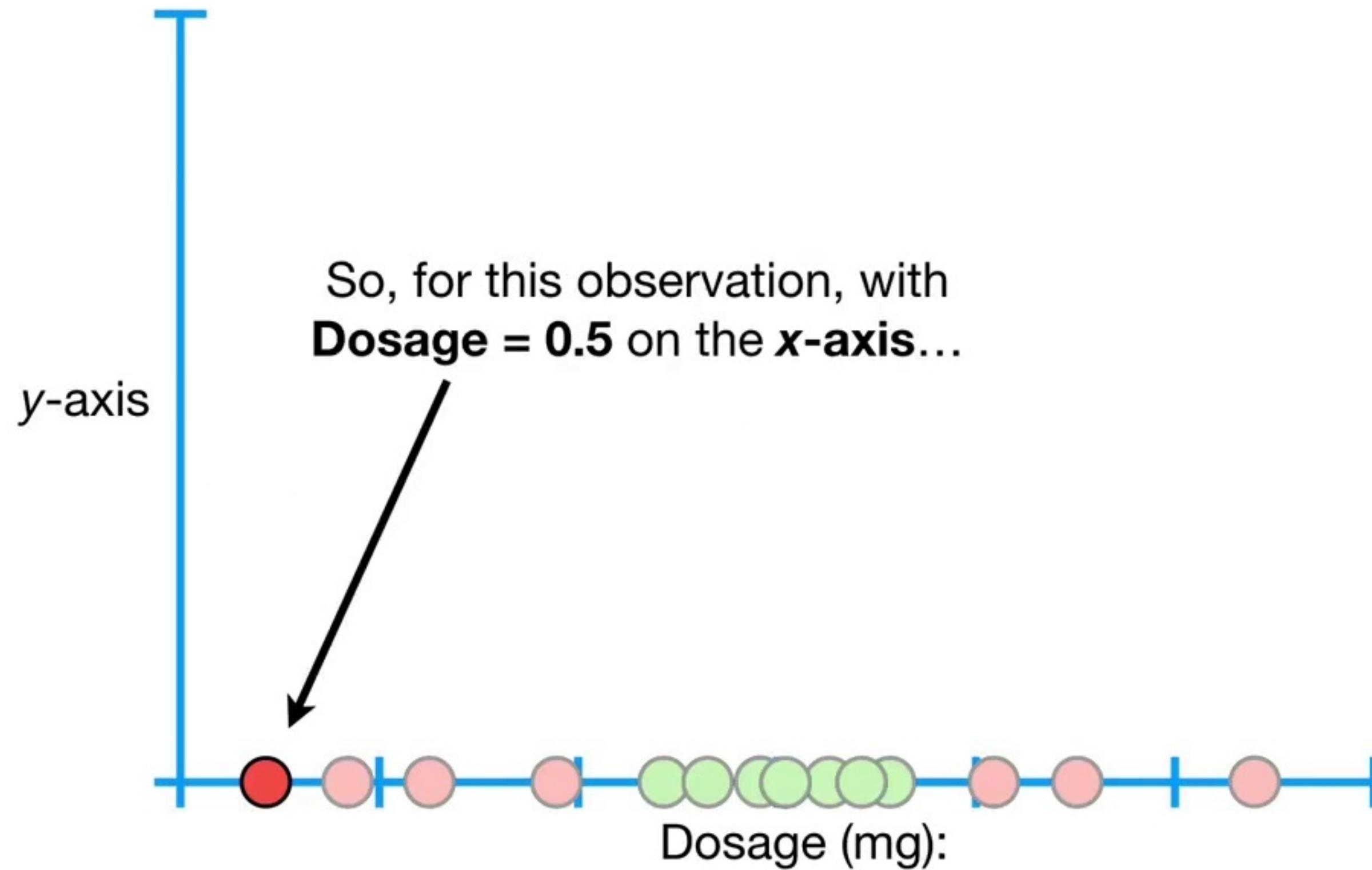
So let's start by getting an intuitive
sense of the main ideas behind
Support Vector Machines.

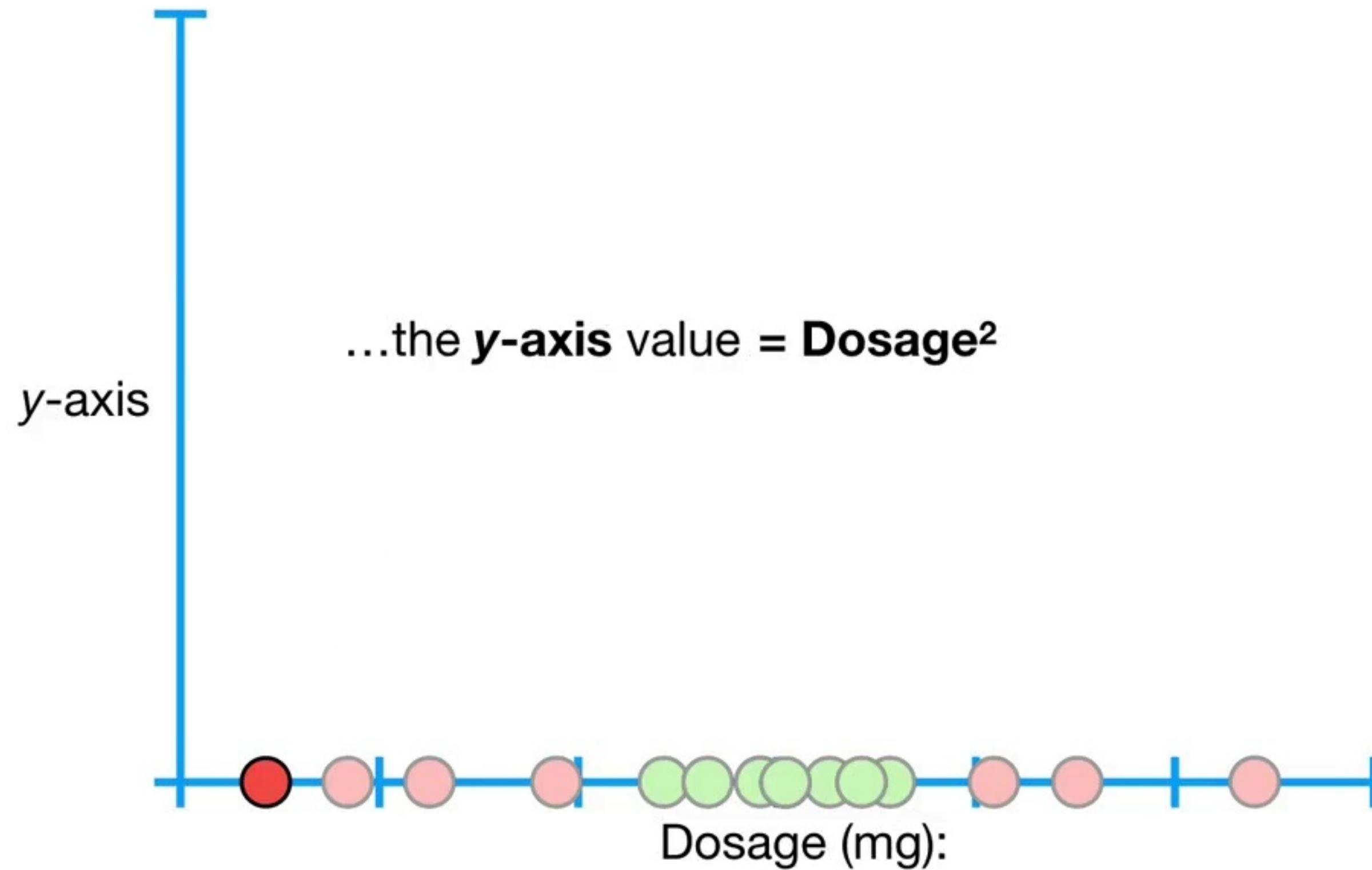


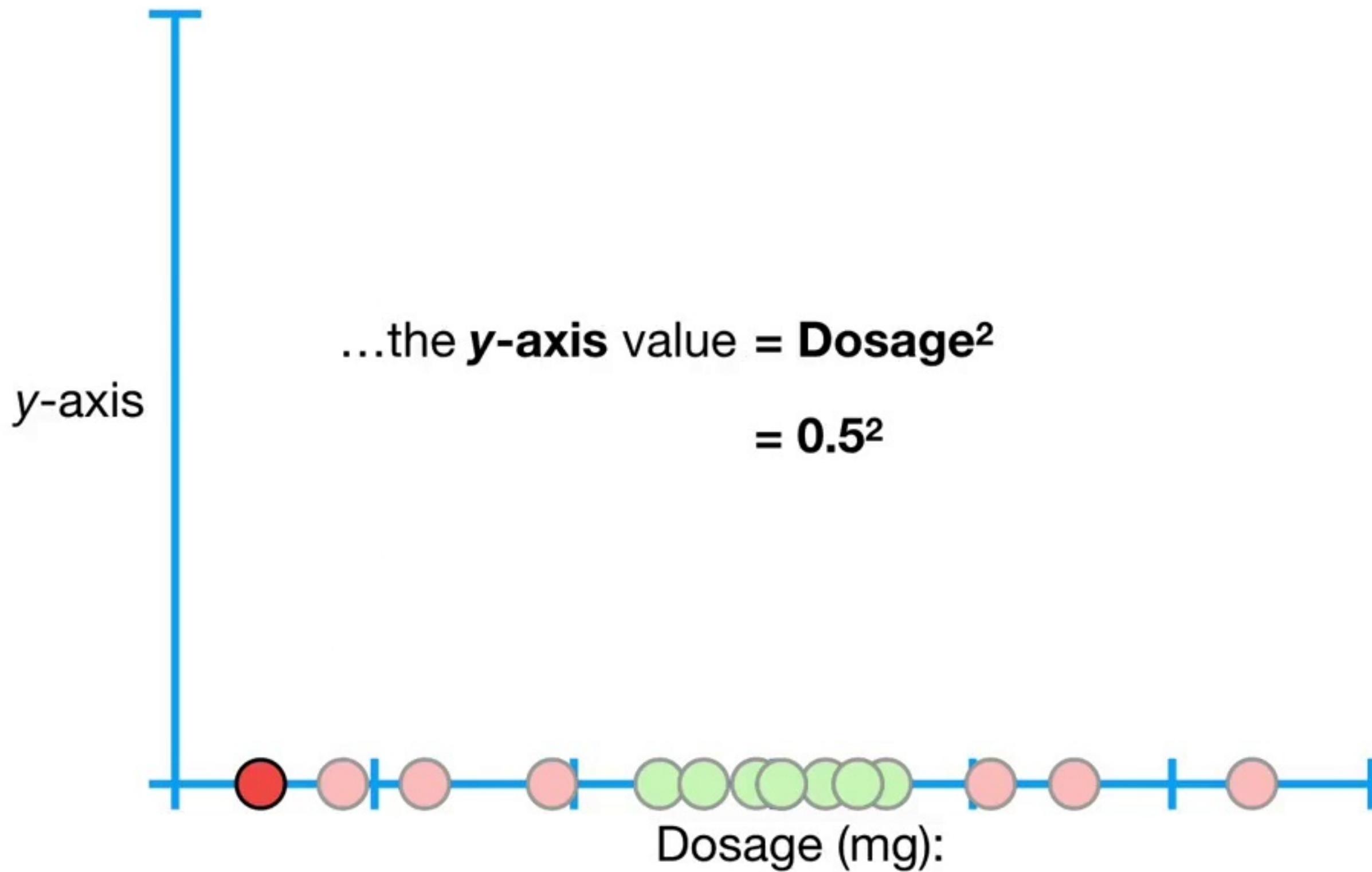


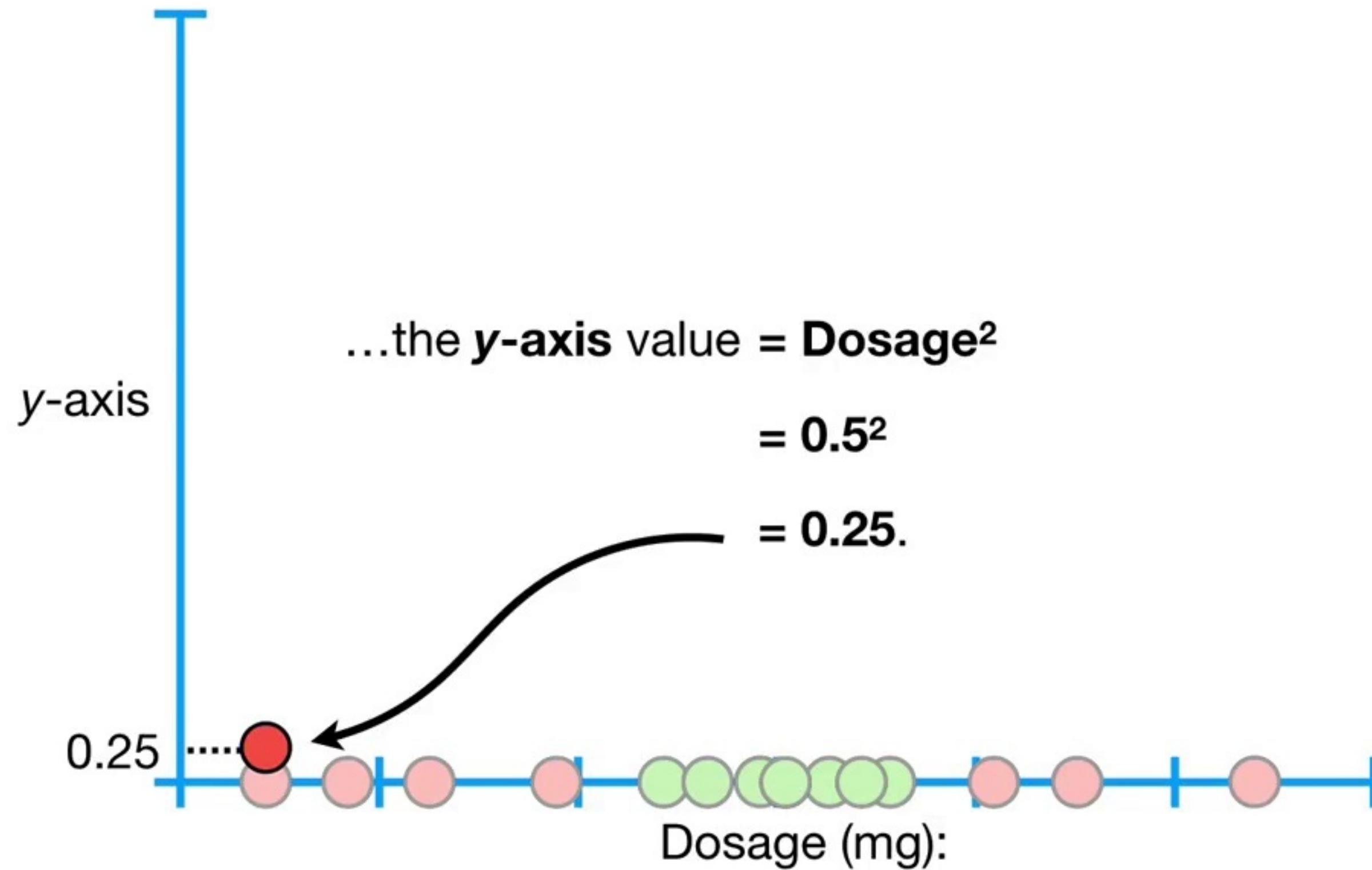


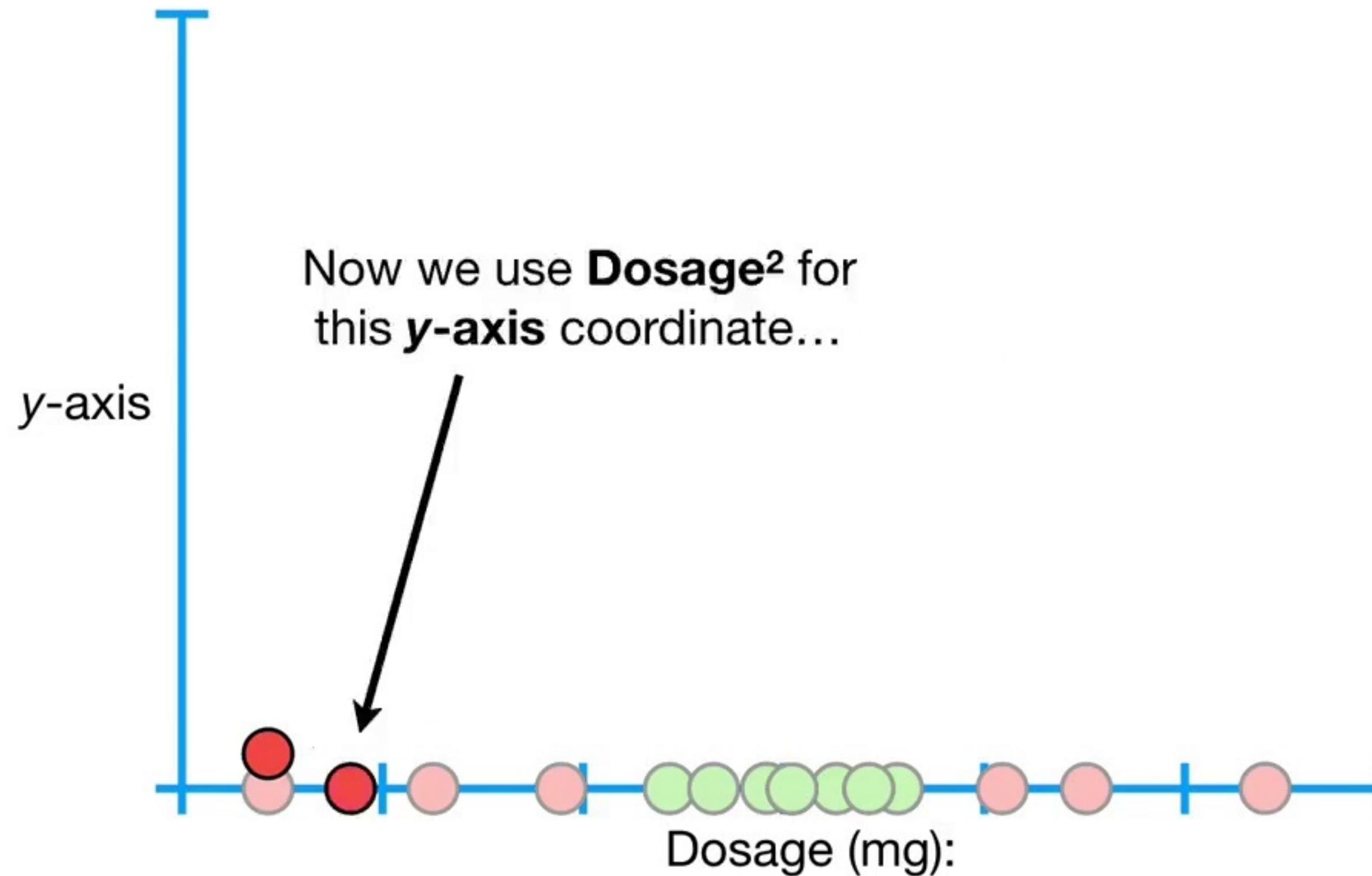


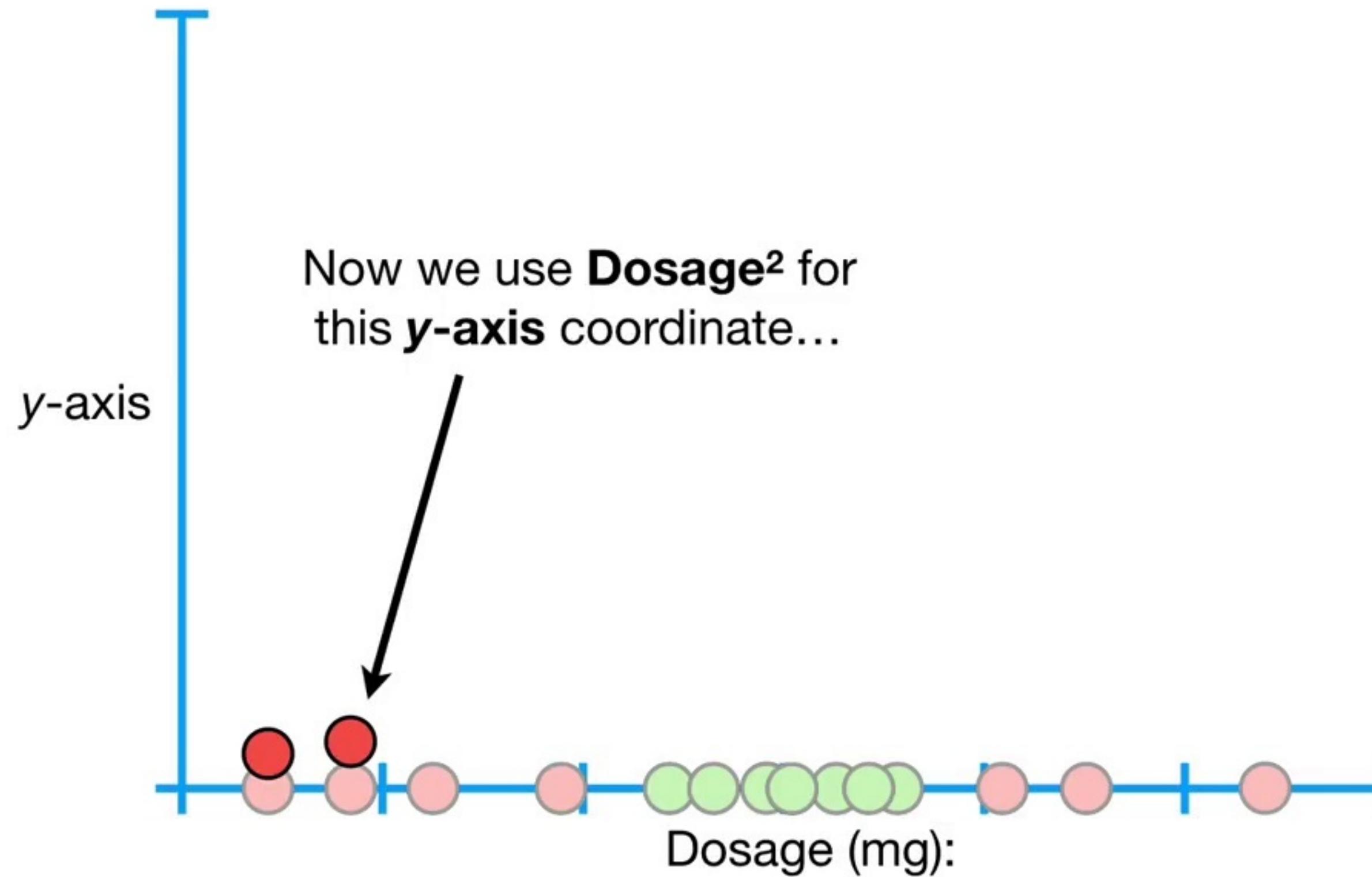




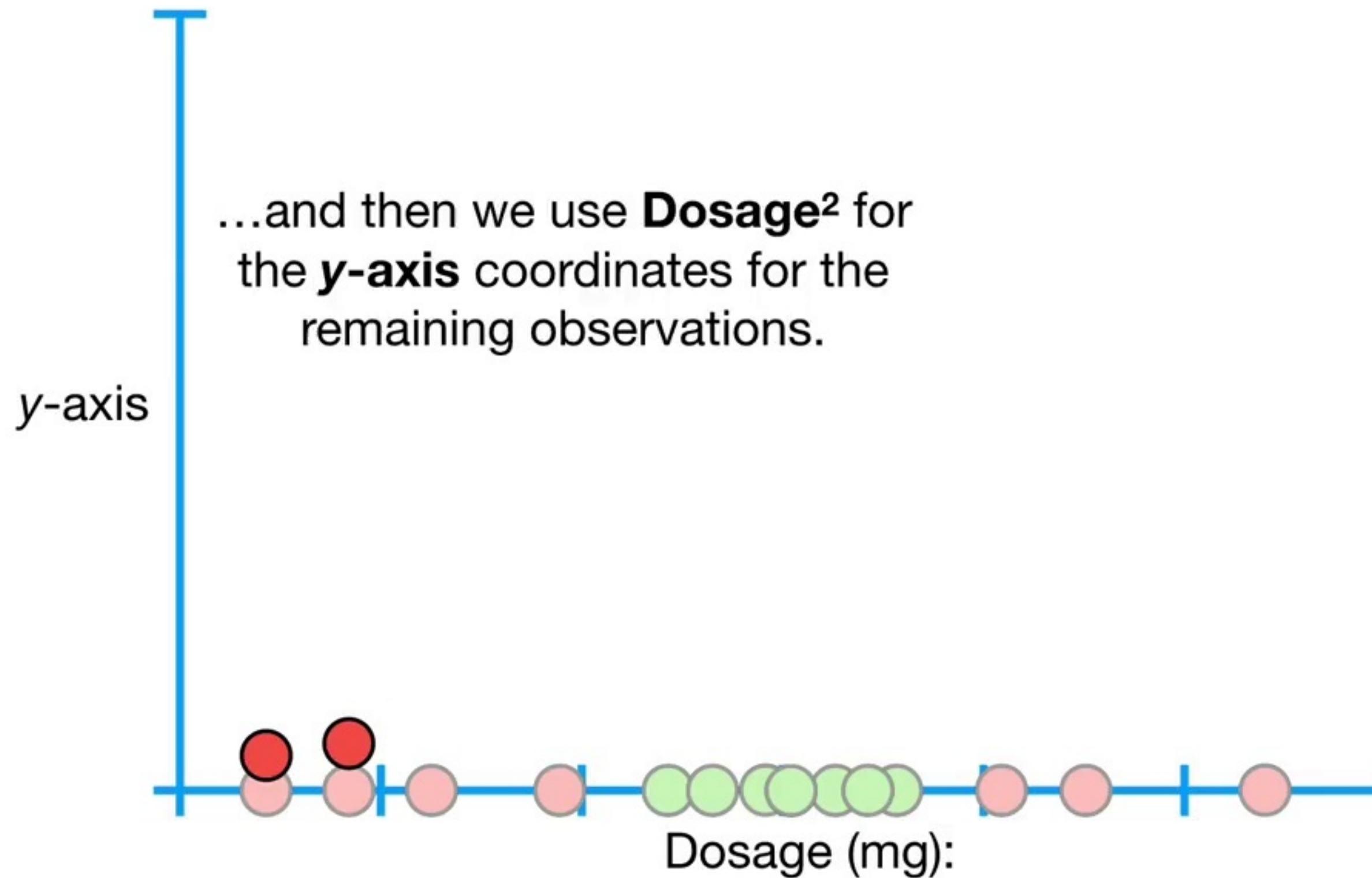


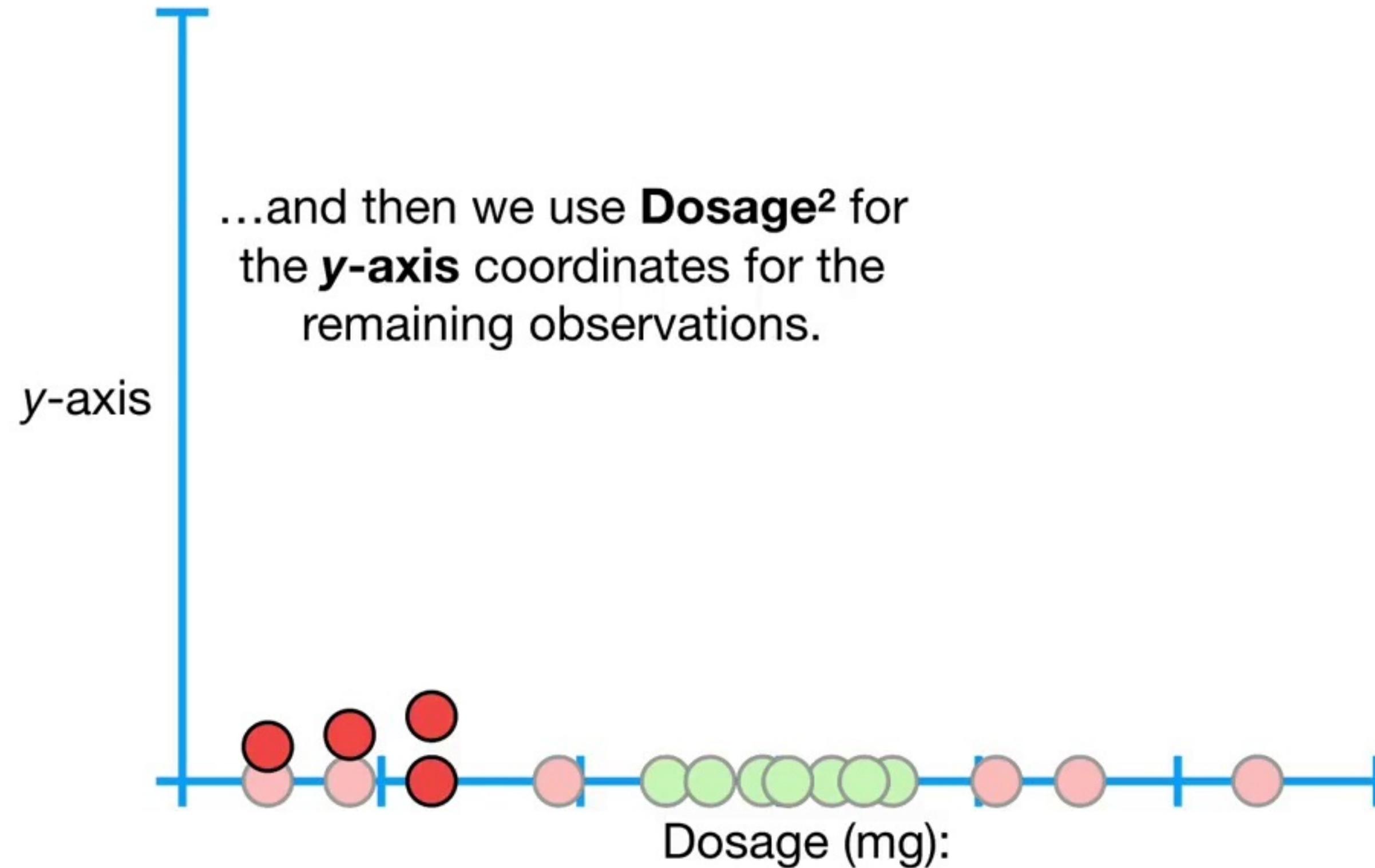




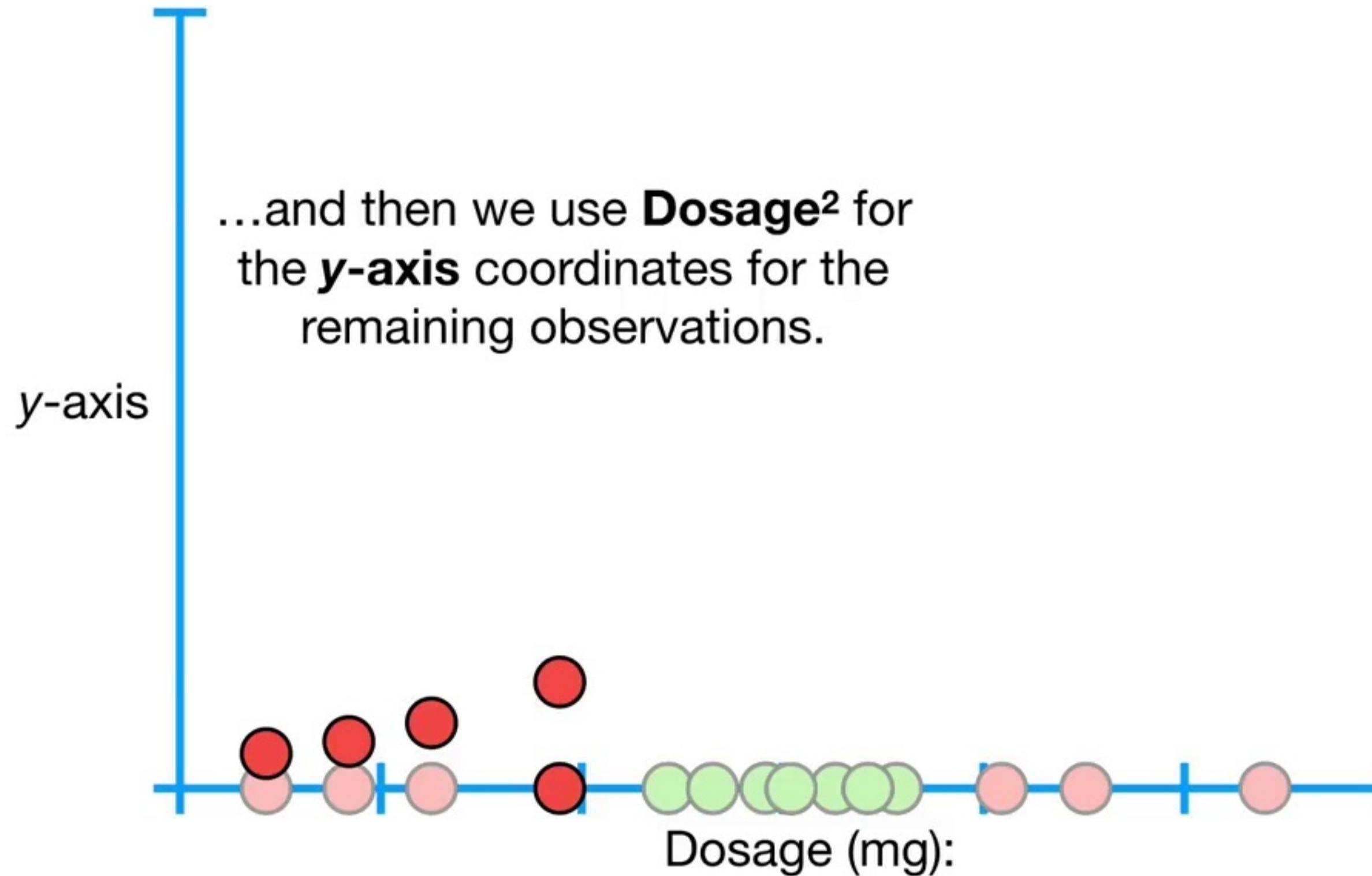


...and then we use **Dosage²** for
the **y-axis** coordinates for the
remaining observations.

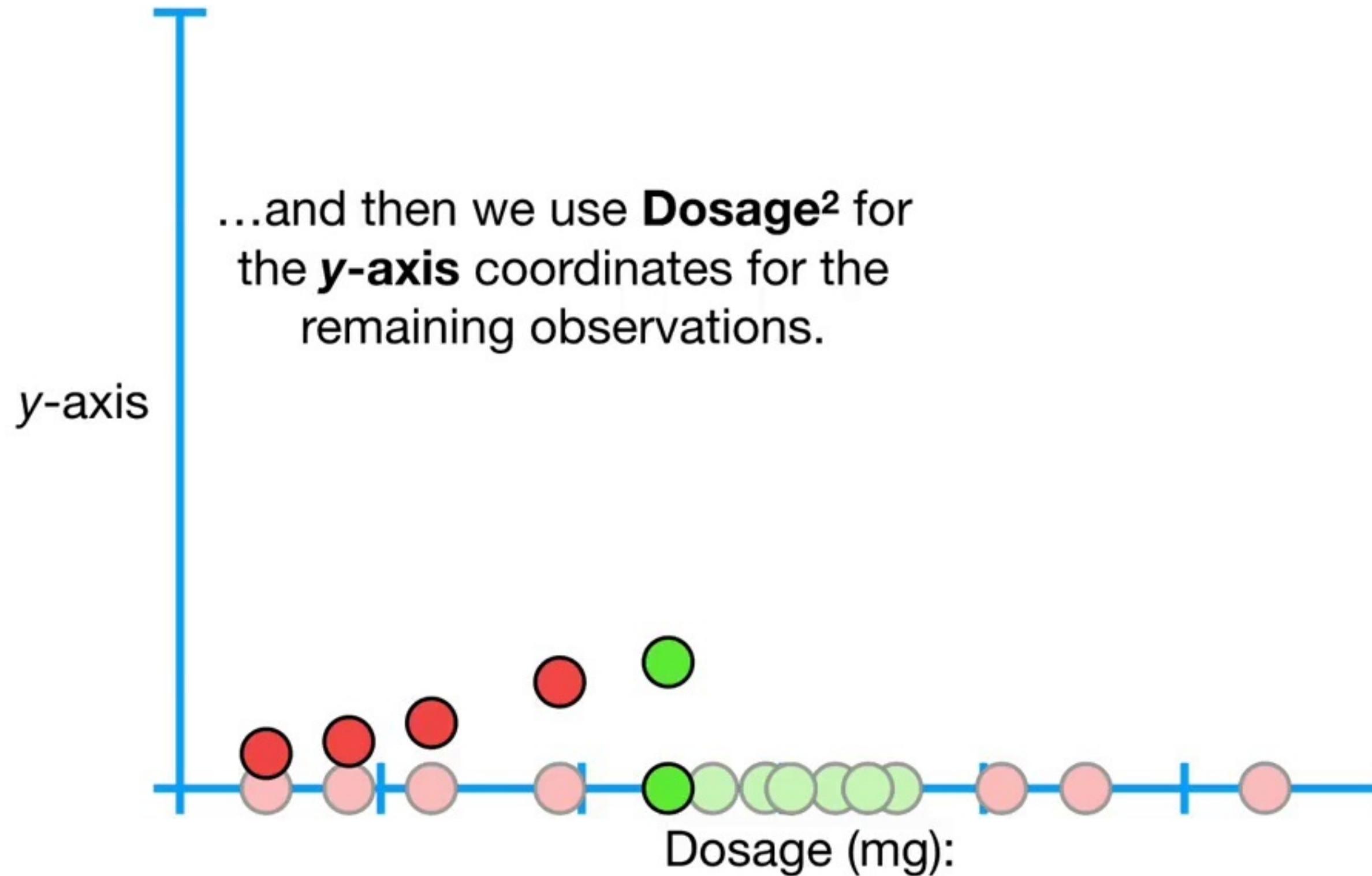




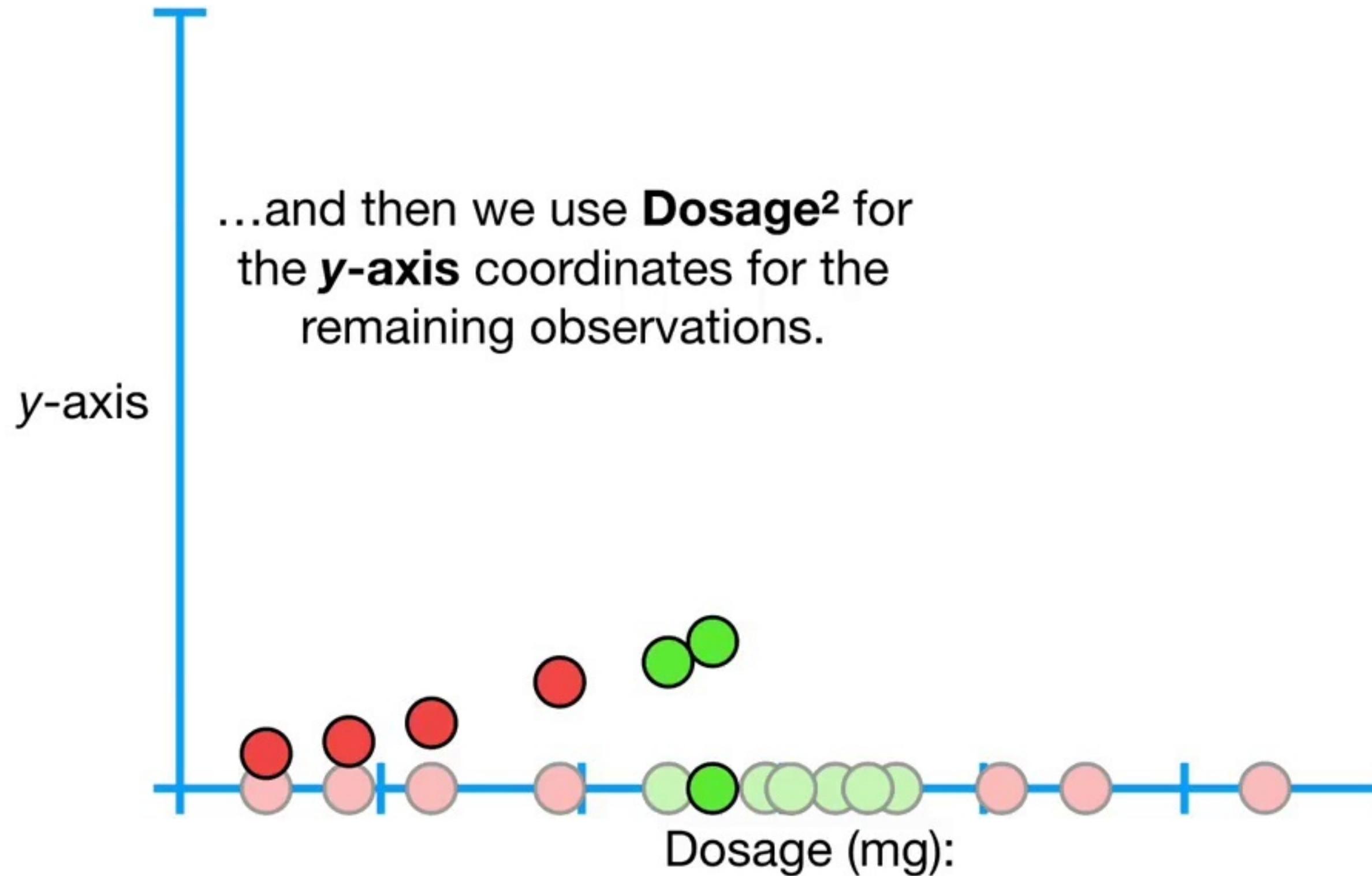
...and then we use **Dosage²** for
the **y-axis** coordinates for the
remaining observations.



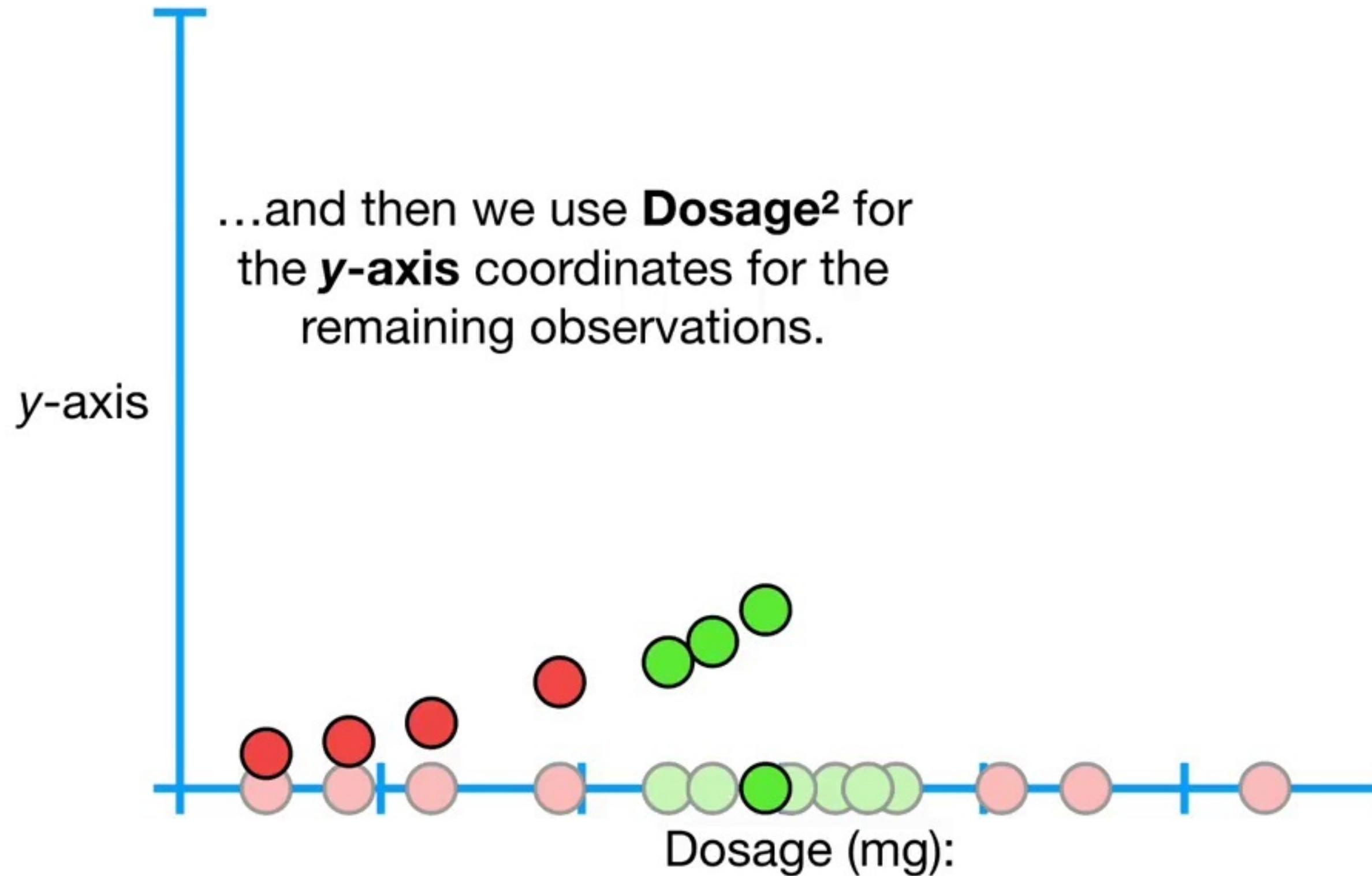
...and then we use **Dosage²** for
the **y-axis** coordinates for the
remaining observations.



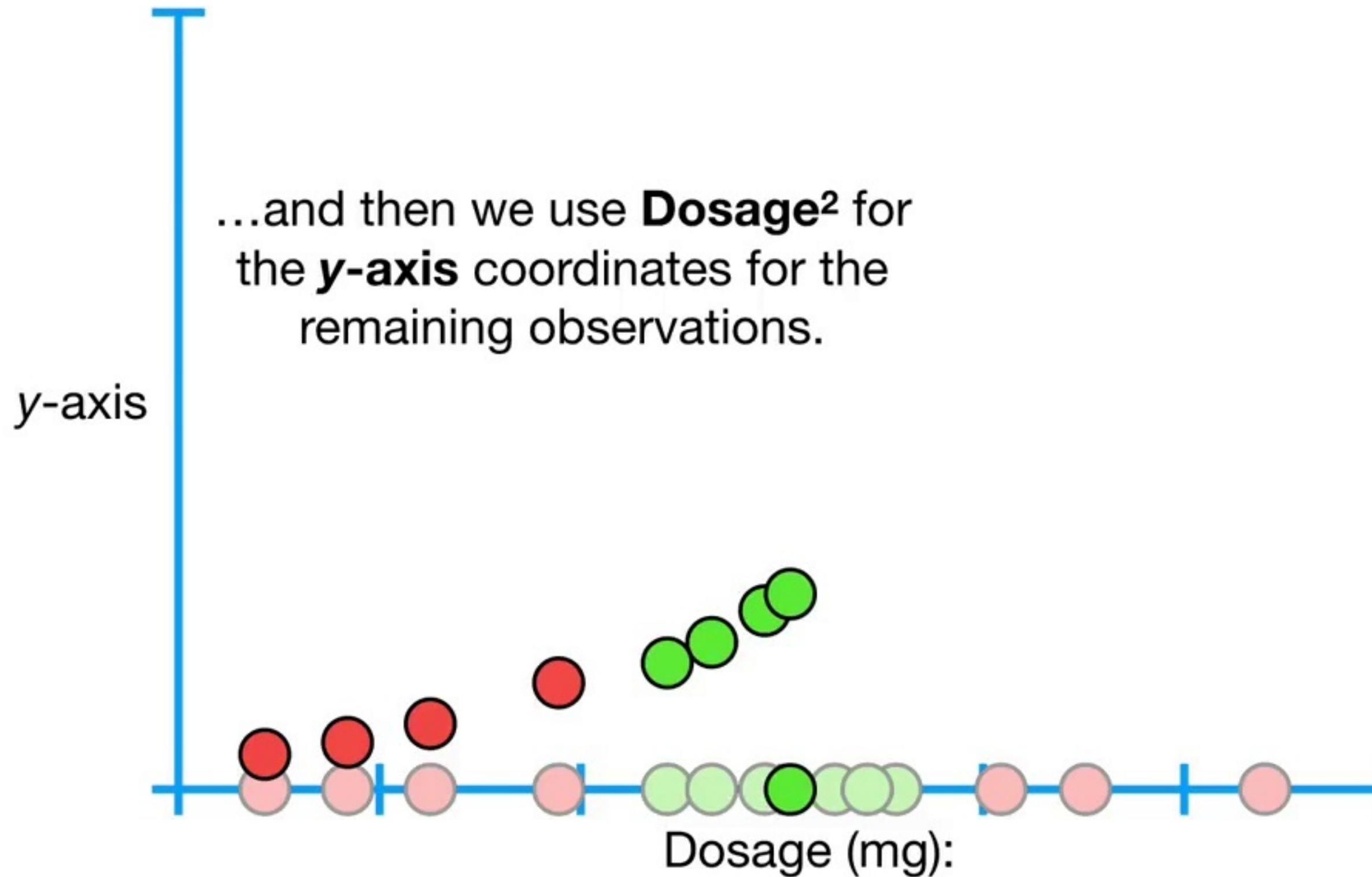
...and then we use **Dosage²** for
the **y-axis** coordinates for the
remaining observations.



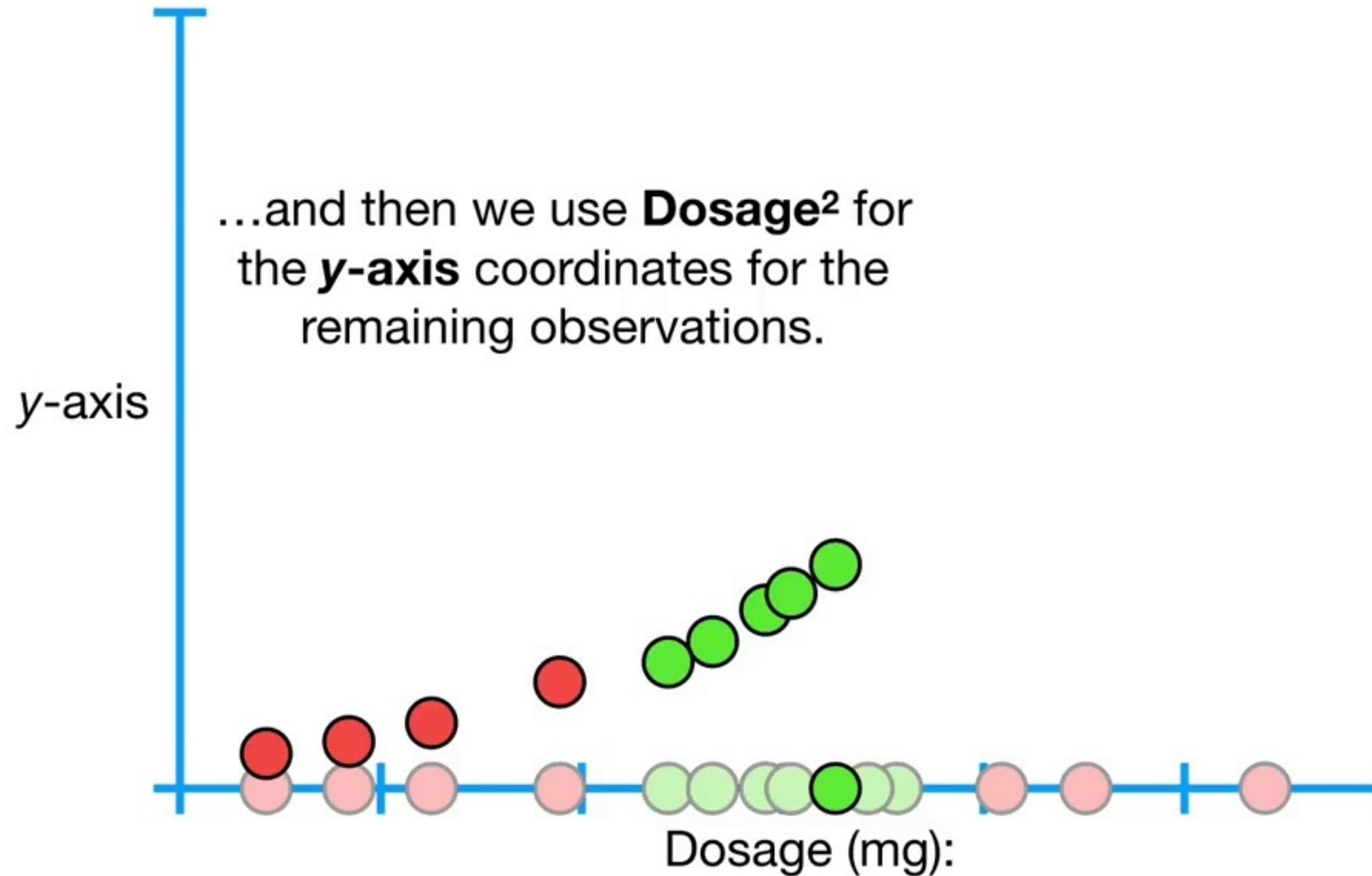
...and then we use **Dosage²** for
the **y-axis** coordinates for the
remaining observations.



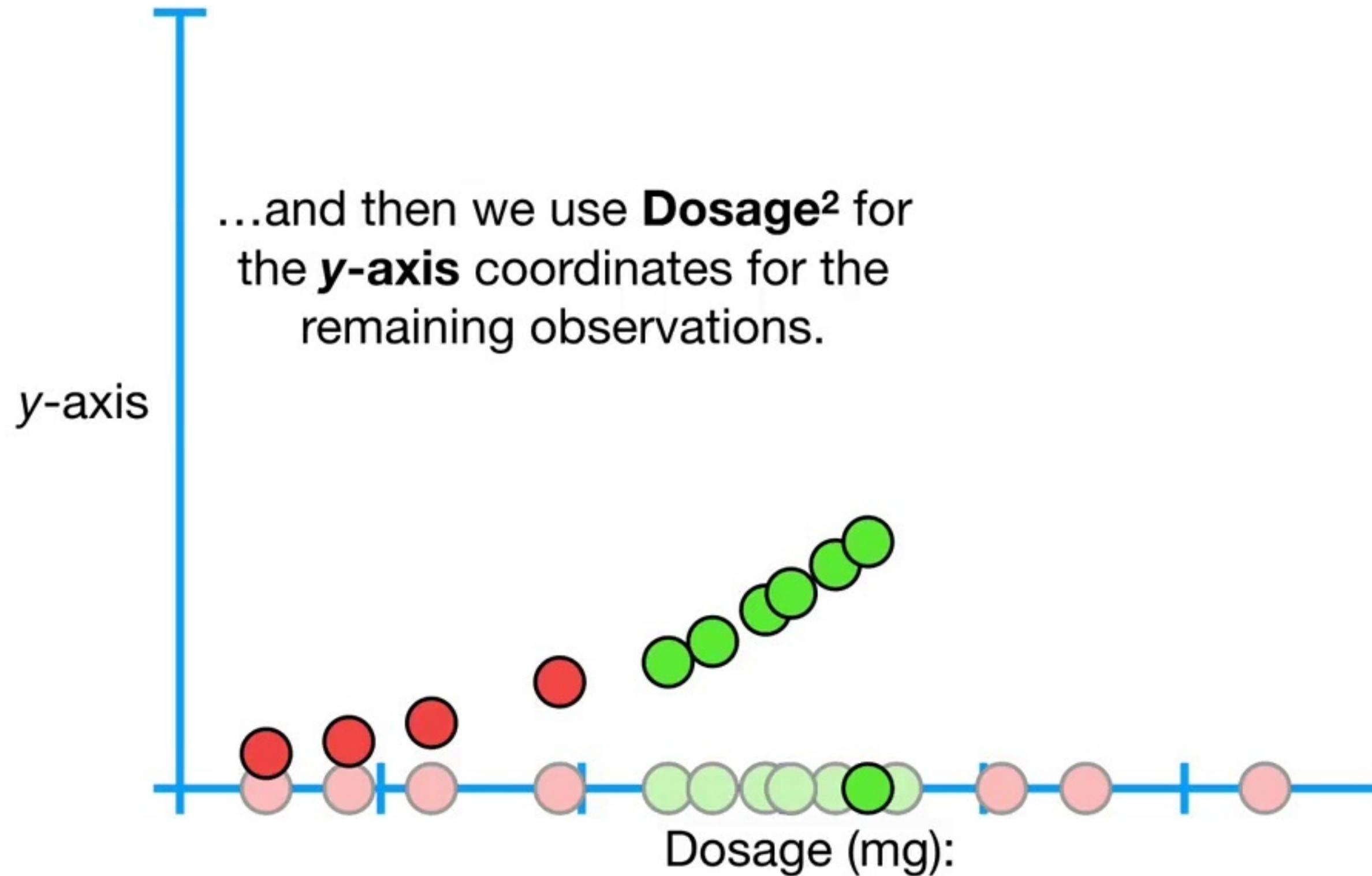
...and then we use **Dosage²** for
the **y-axis** coordinates for the
remaining observations.



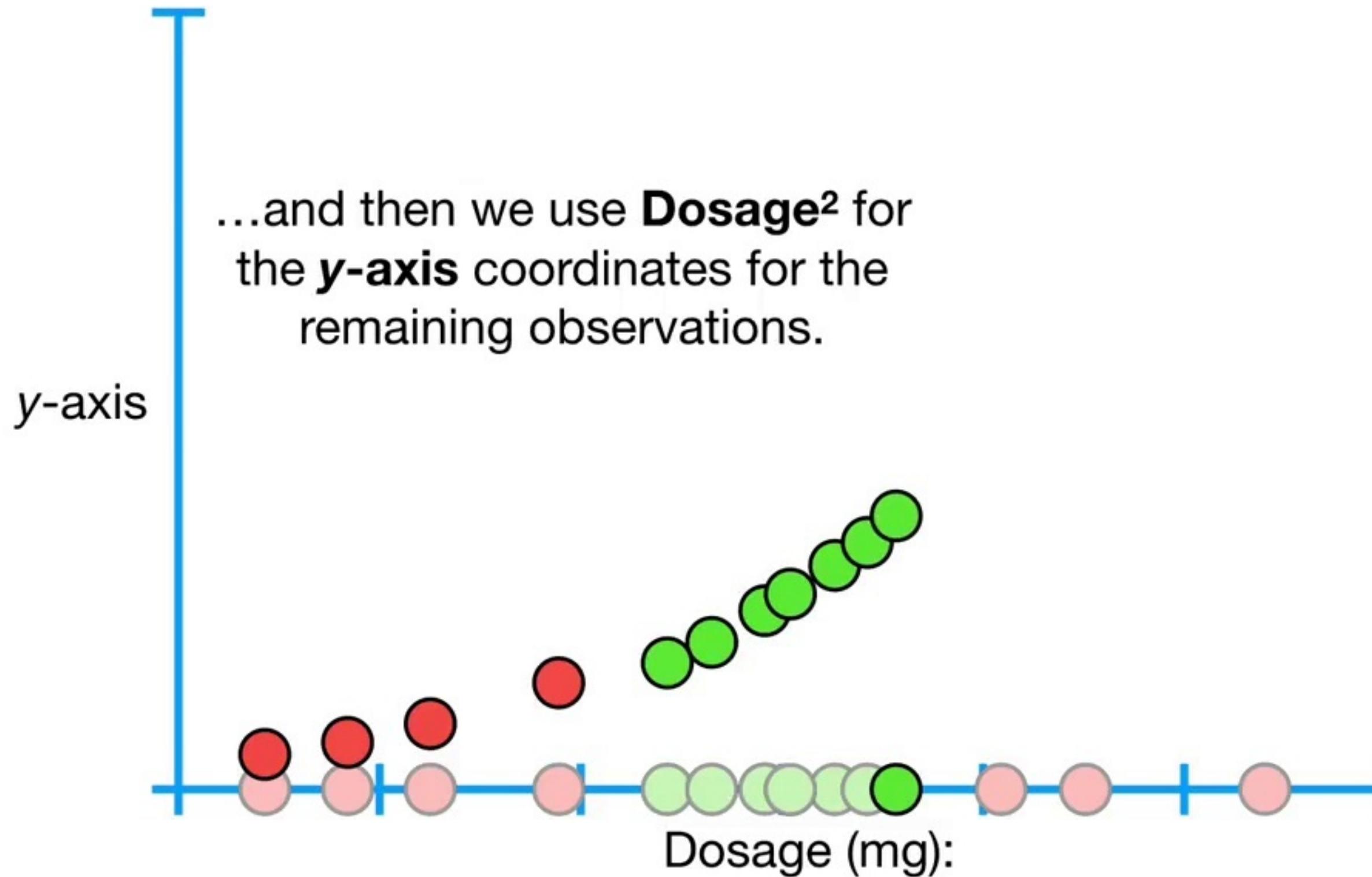
...and then we use **Dosage²** for
the **y-axis** coordinates for the
remaining observations.

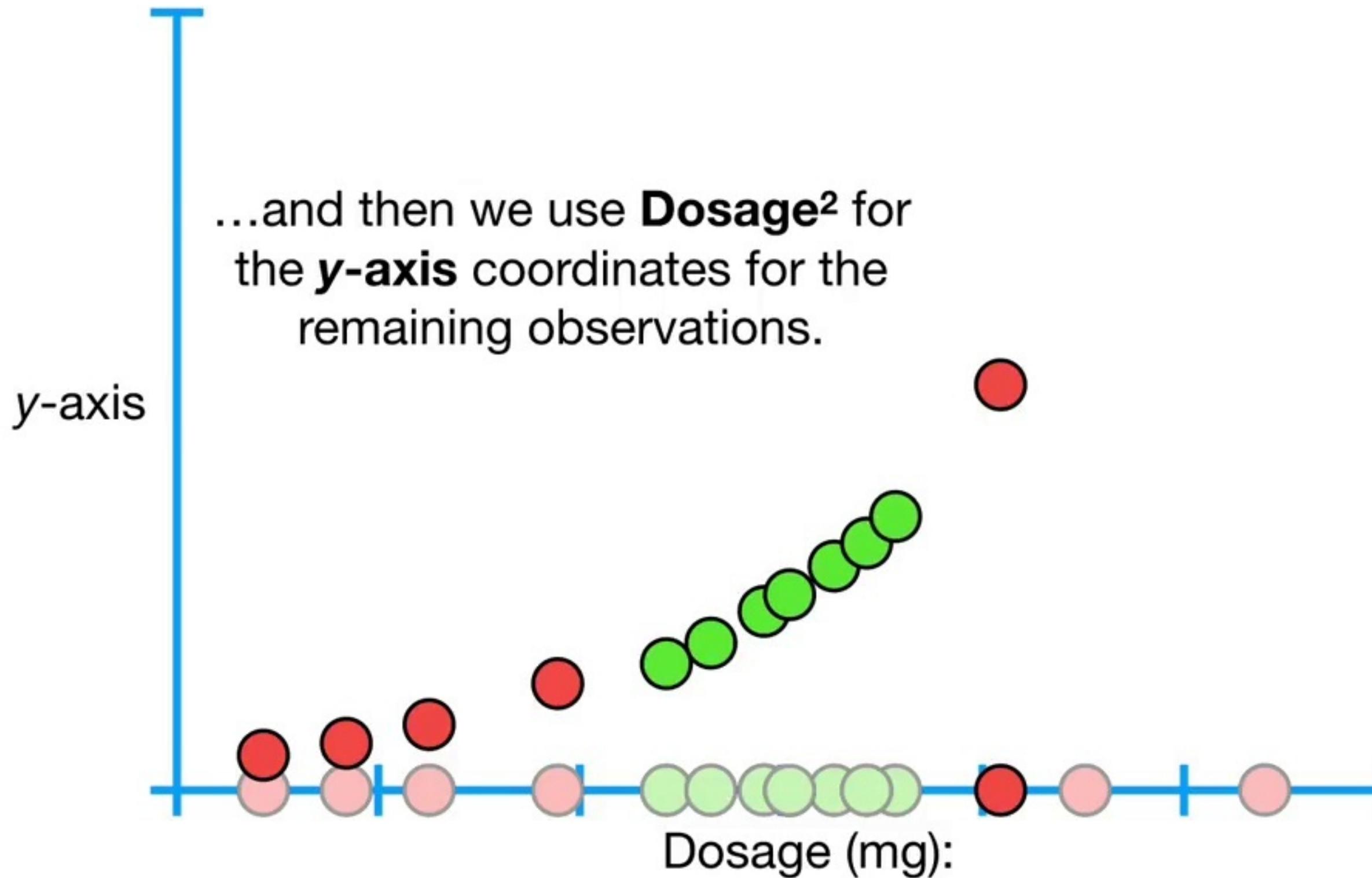


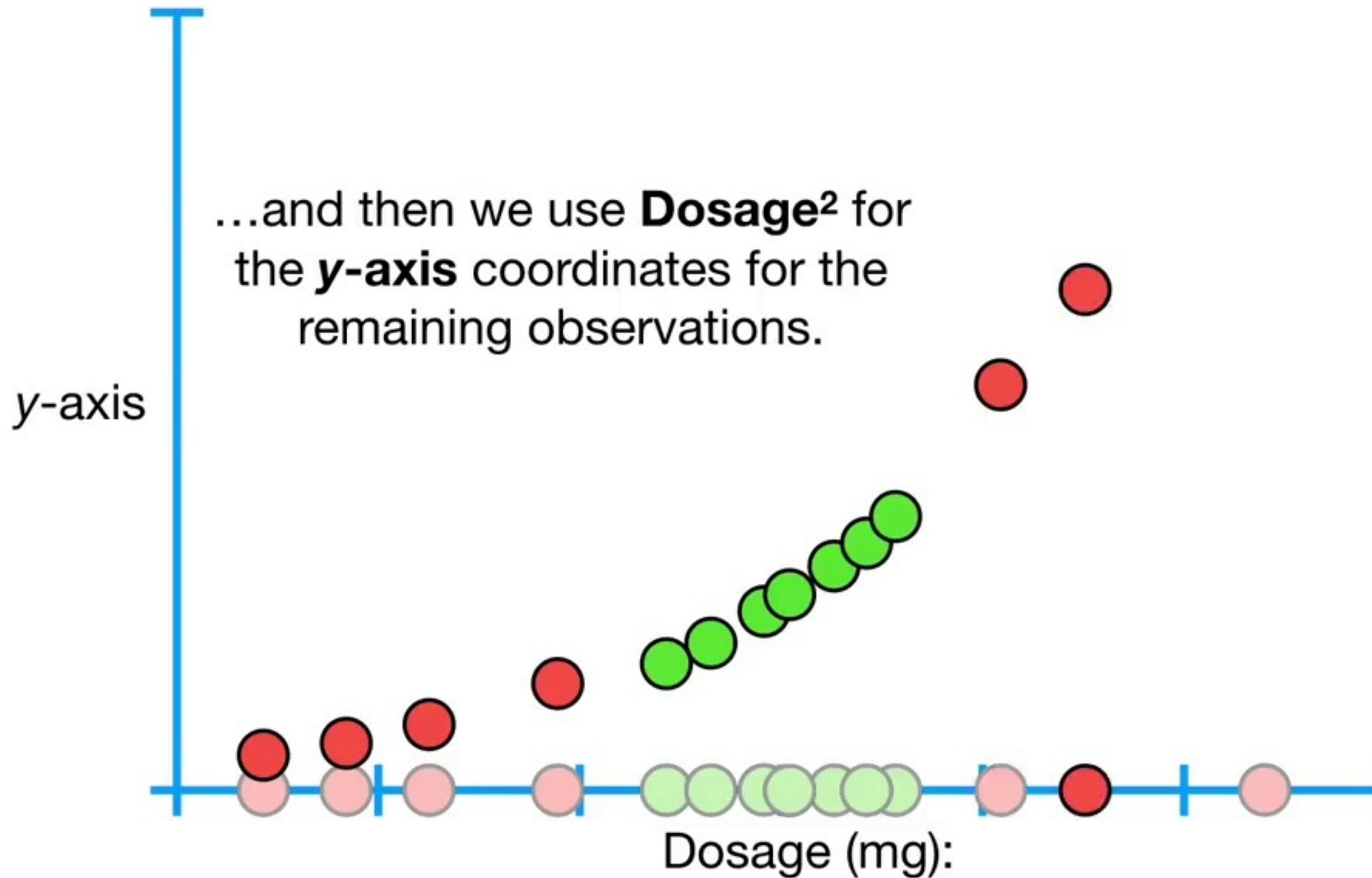
...and then we use **Dosage²** for
the **y-axis** coordinates for the
remaining observations.

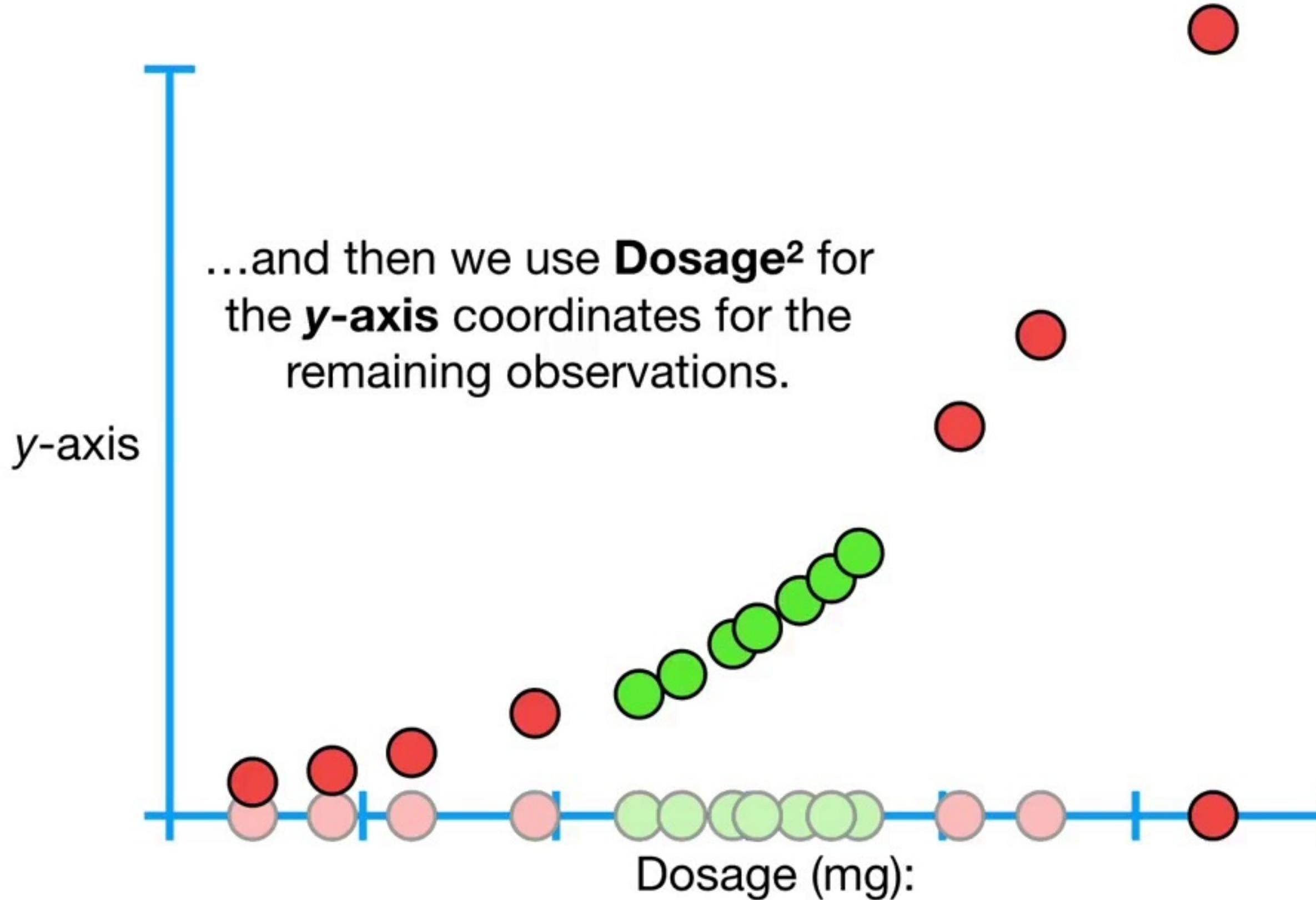


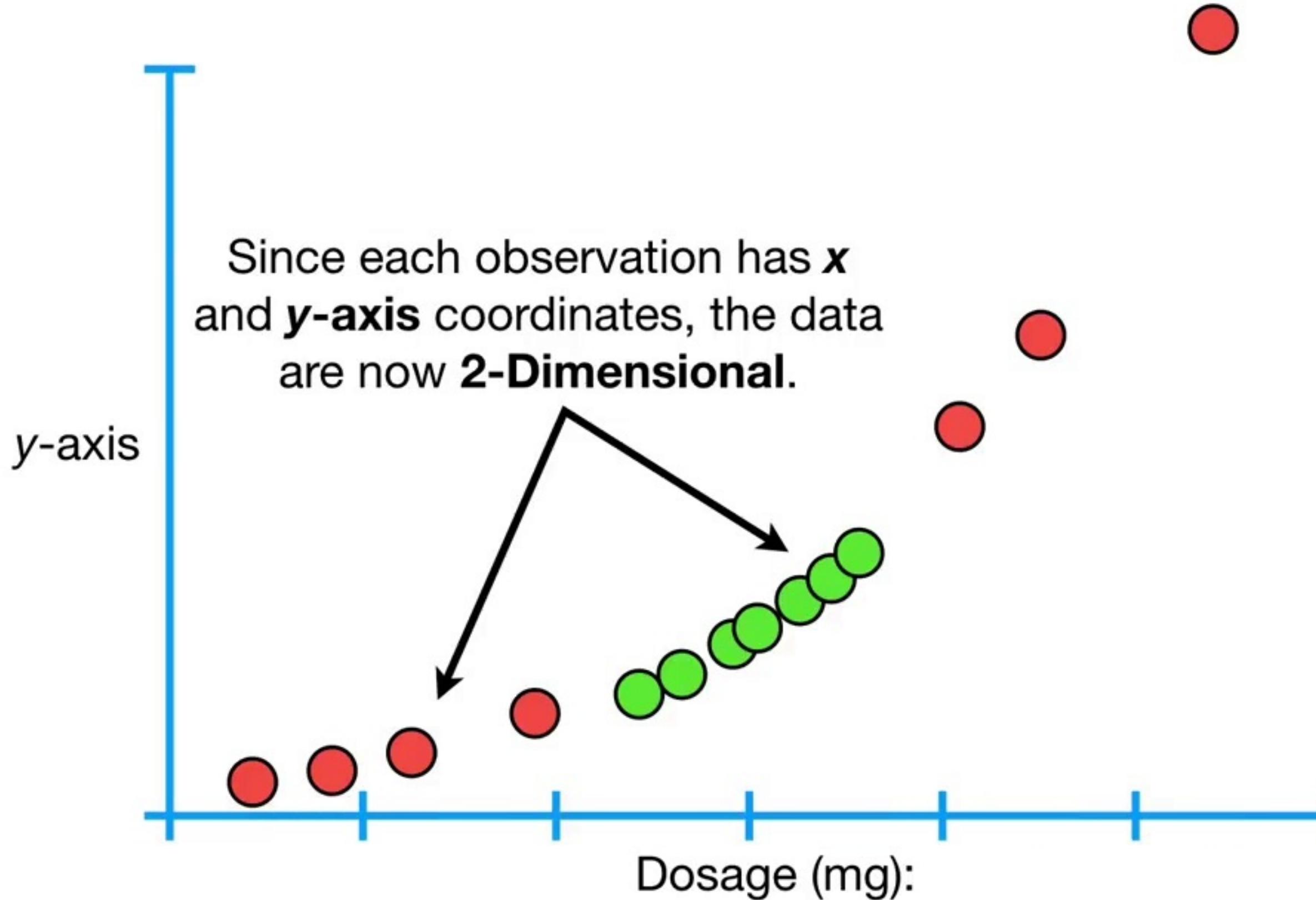
...and then we use **Dosage²** for
the **y-axis** coordinates for the
remaining observations.







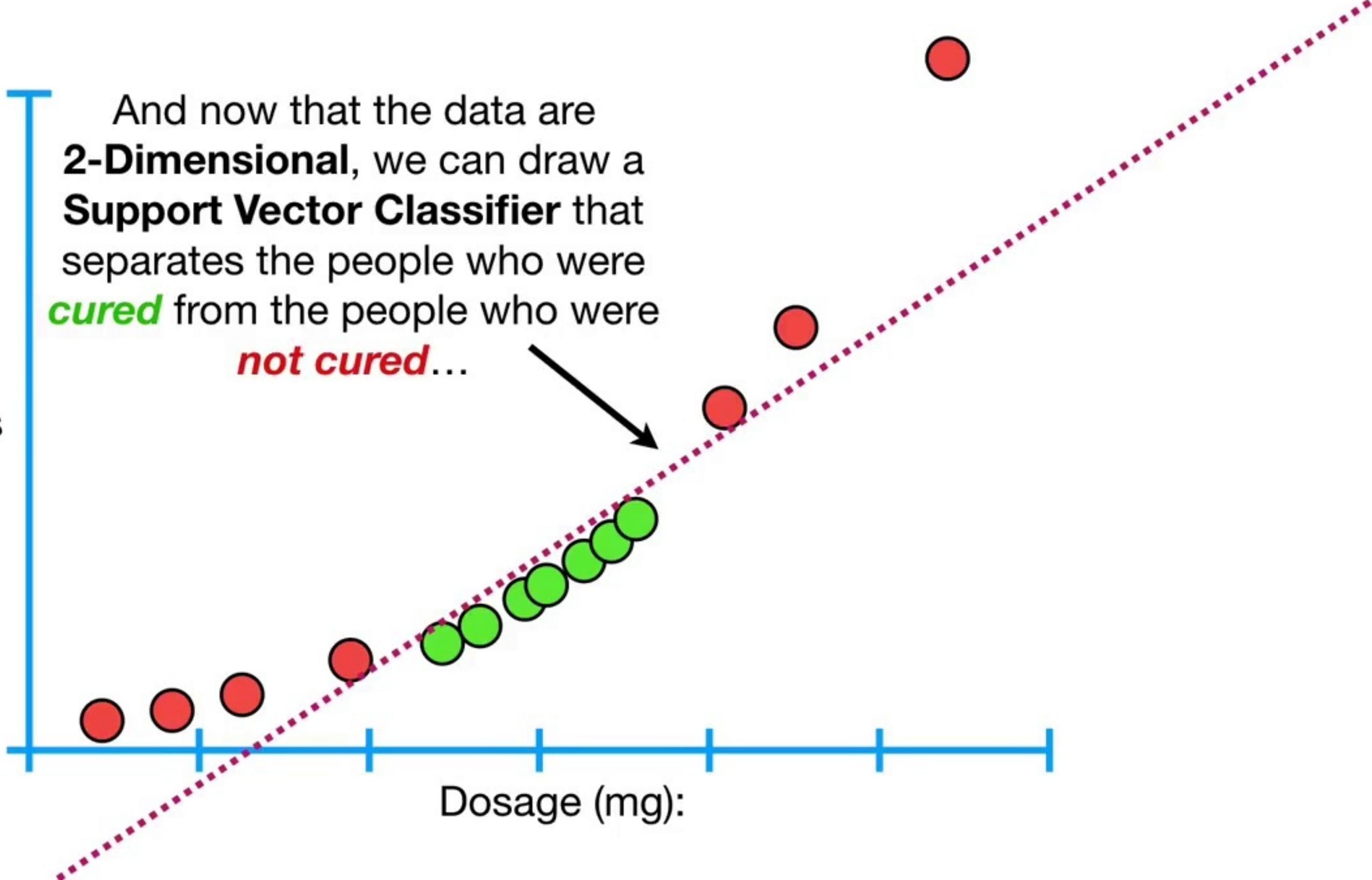


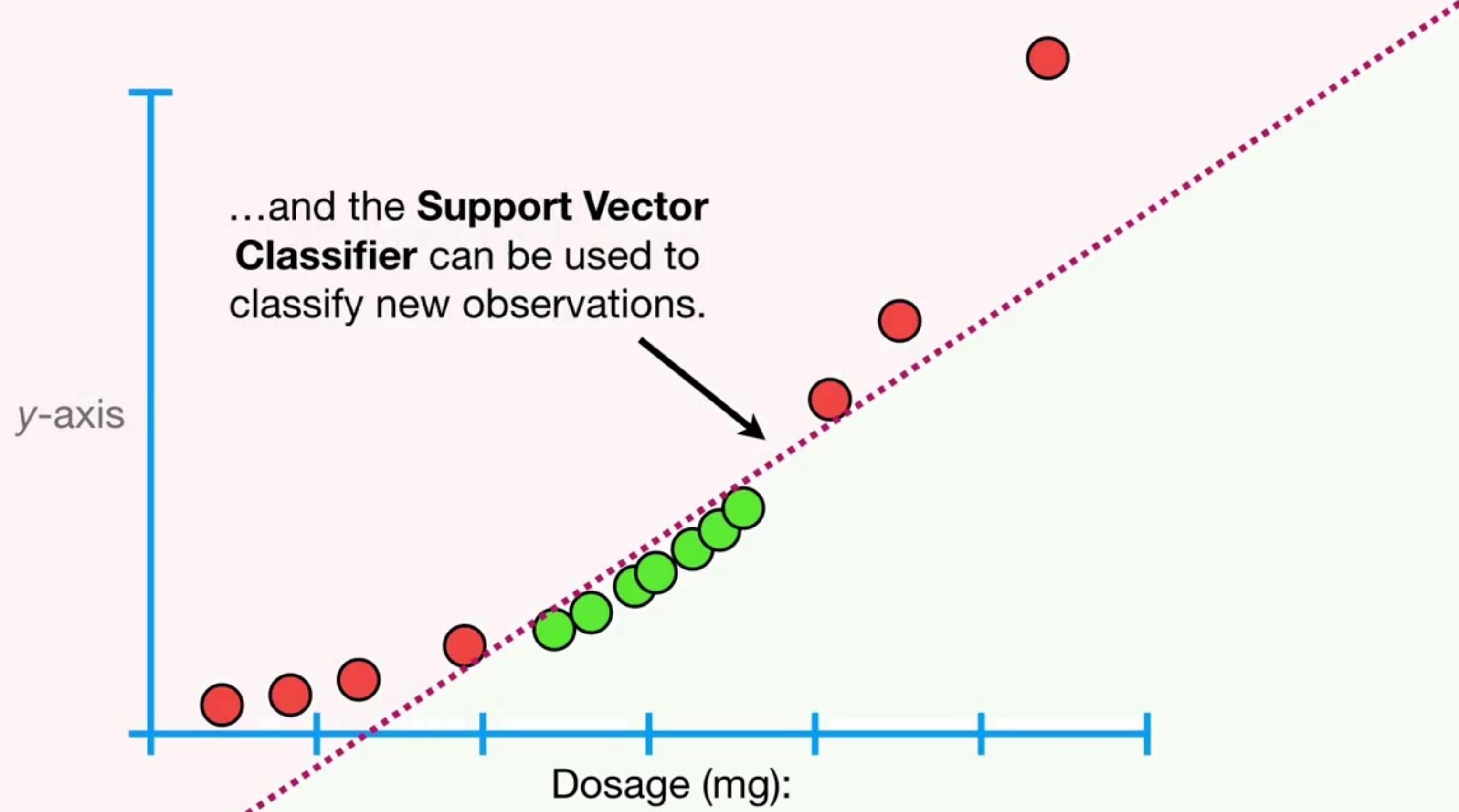


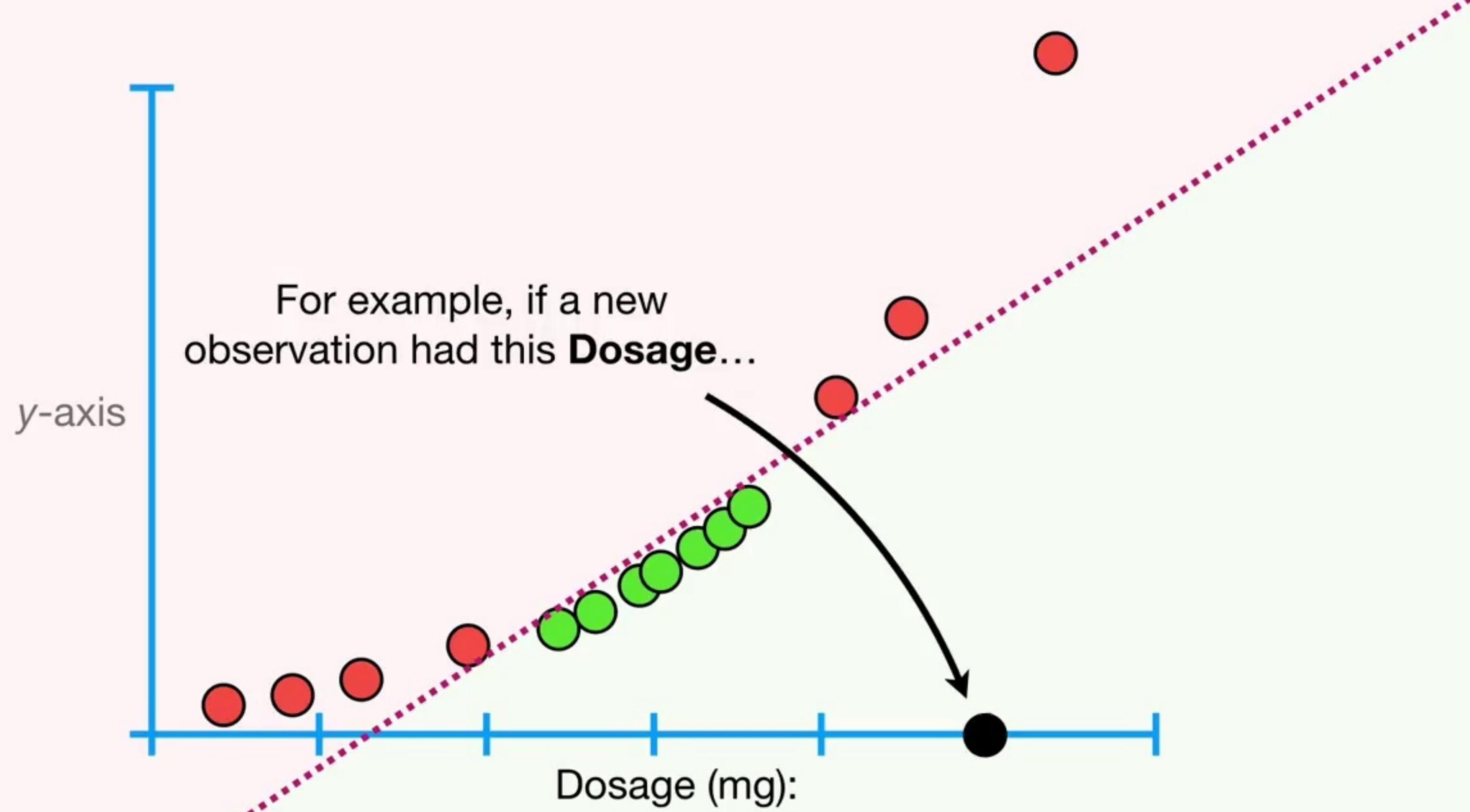
And now that the data are **2-Dimensional**, we can draw a **Support Vector Classifier** that separates the people who were **cured** from the people who were ***not cured***...

y-axis

Dosage (mg):



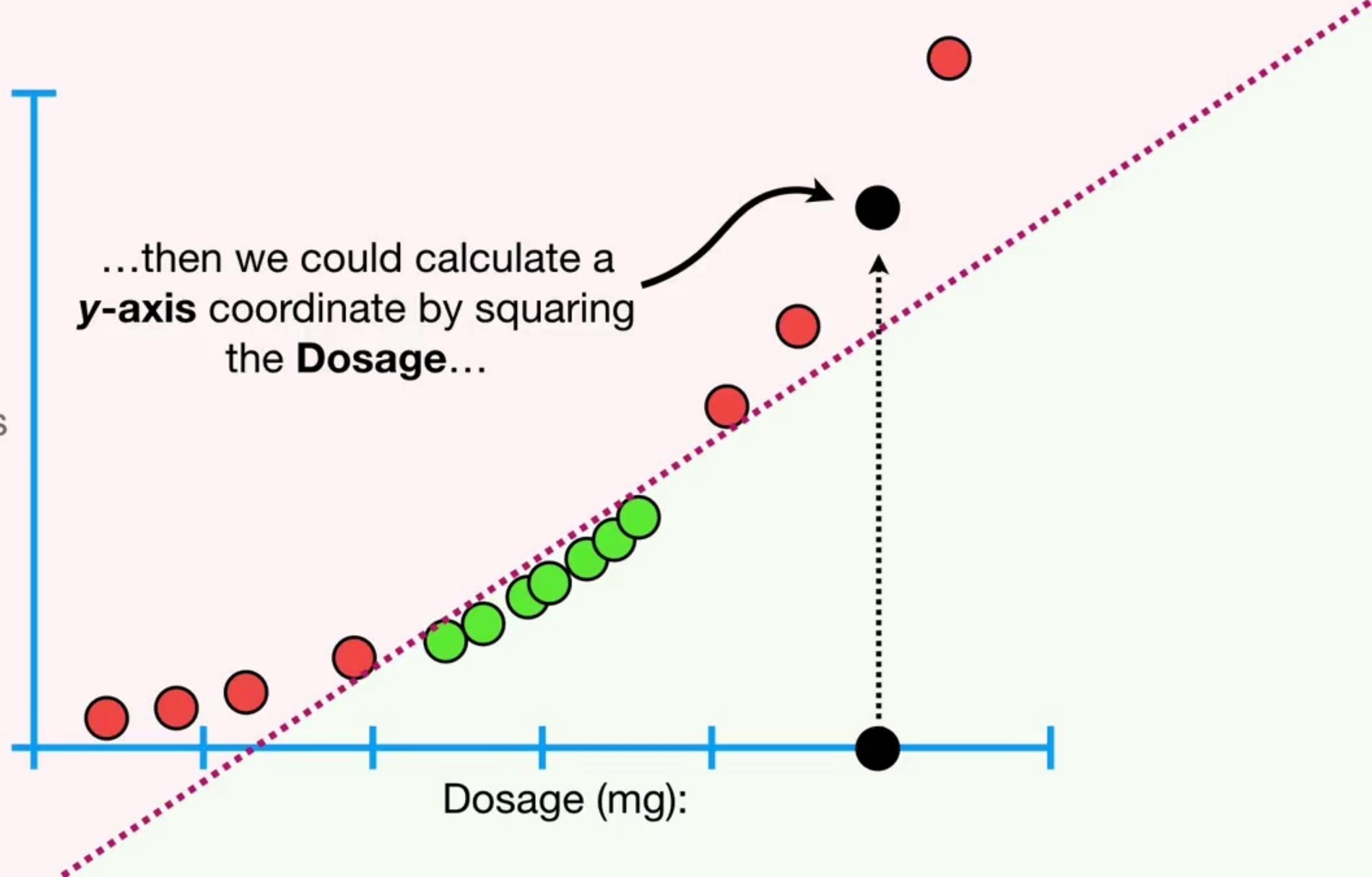




y-axis

...then we could calculate a
y-axis coordinate by squaring
the **Dosage**...

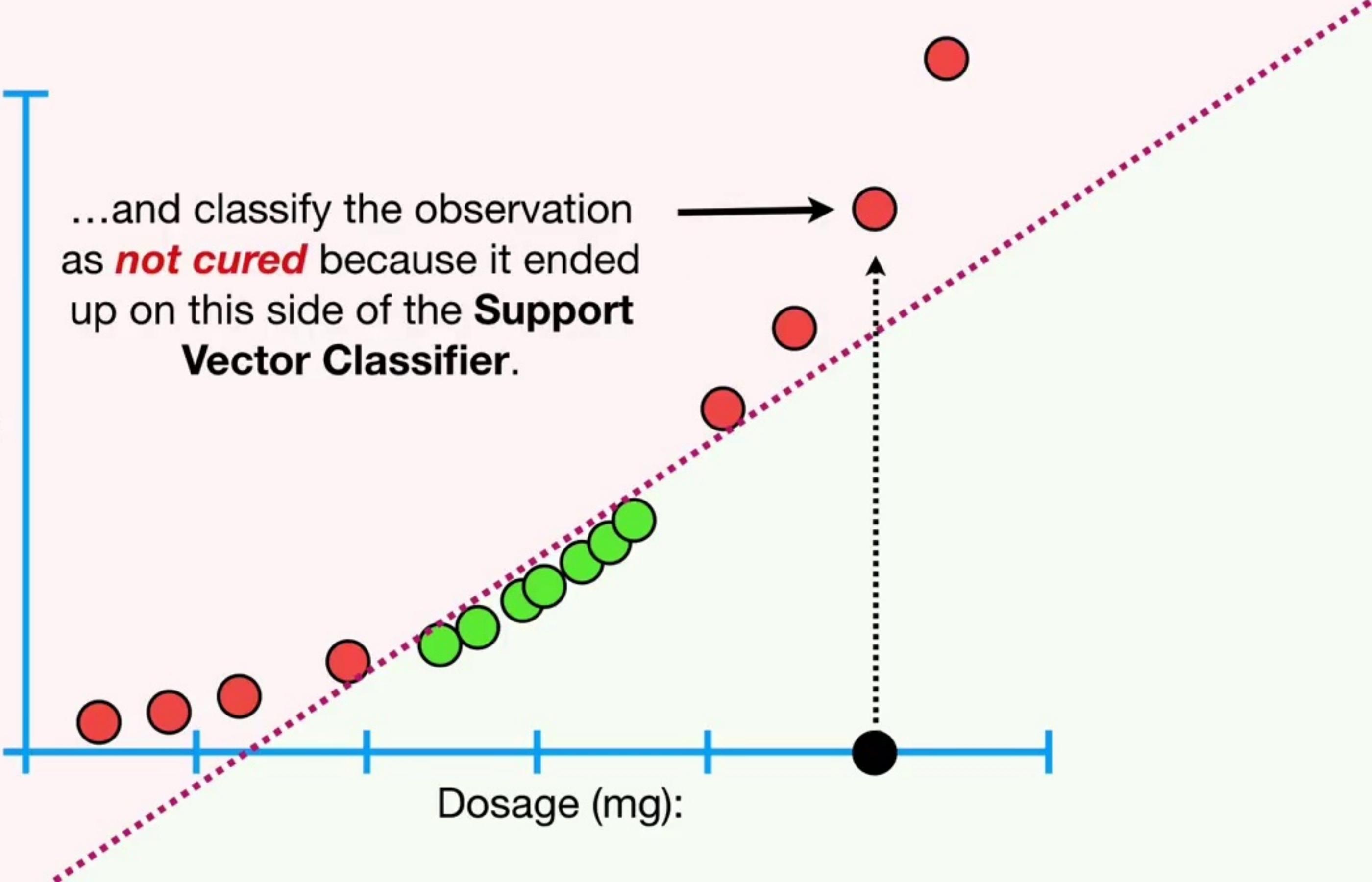
Dosage (mg):

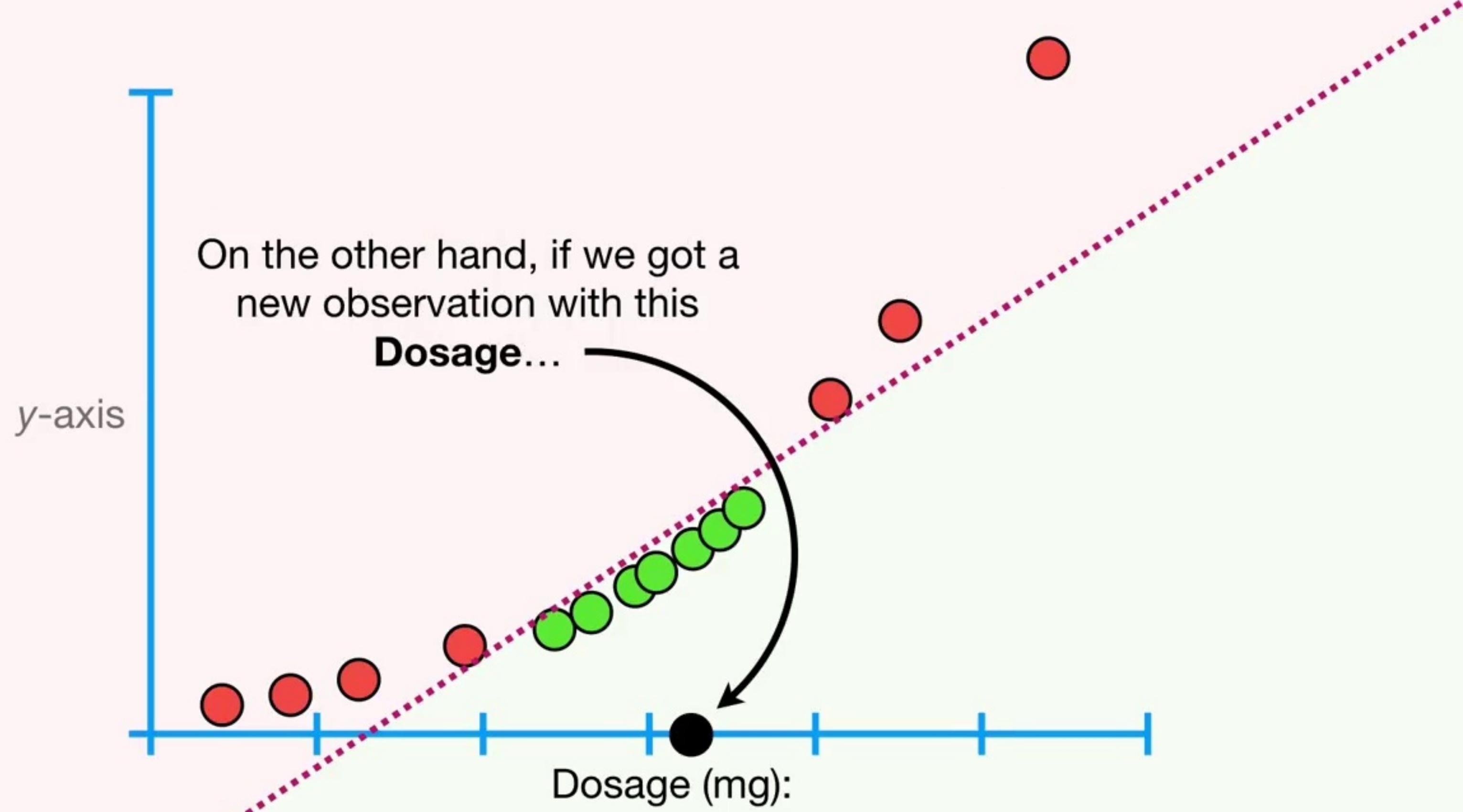


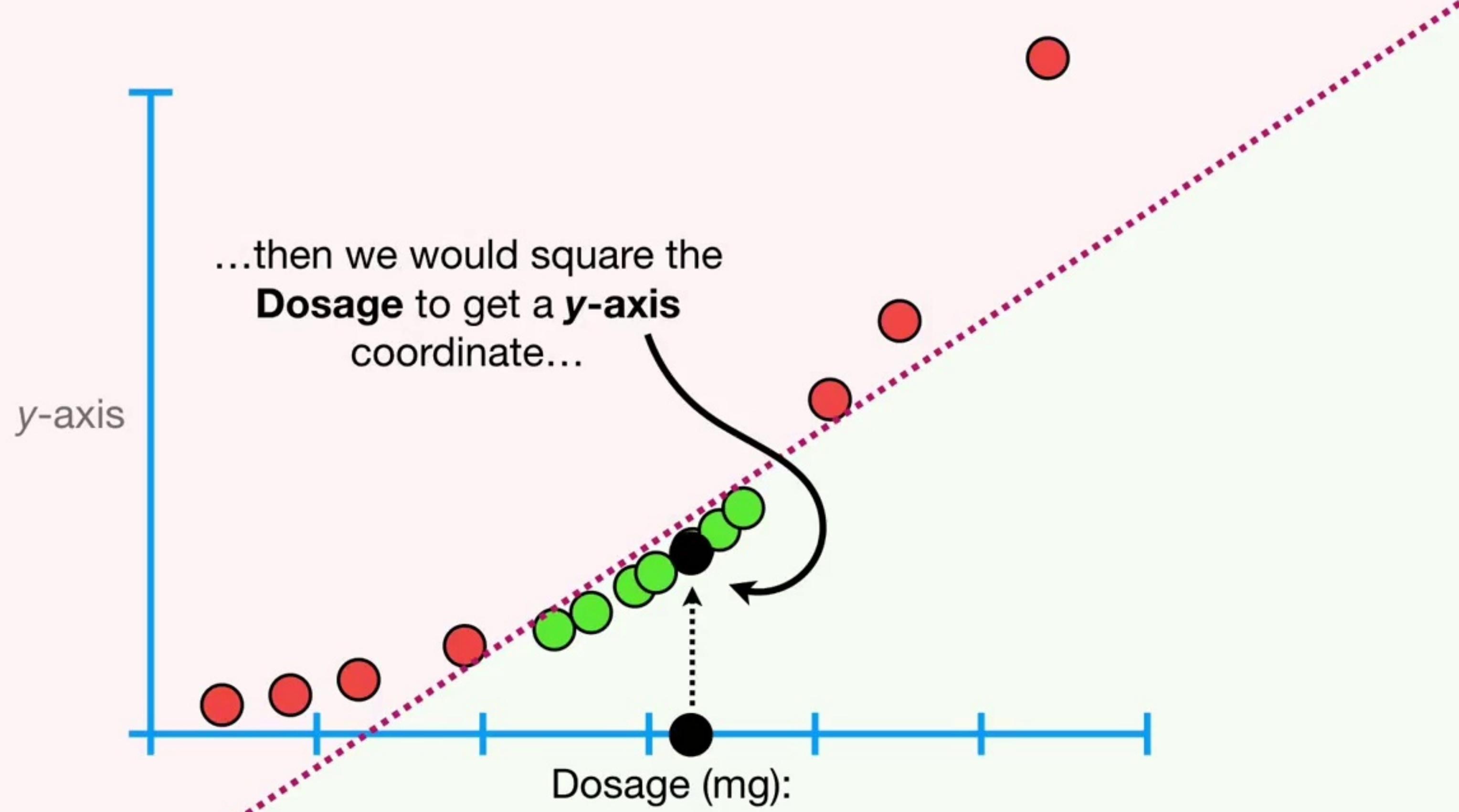
y-axis

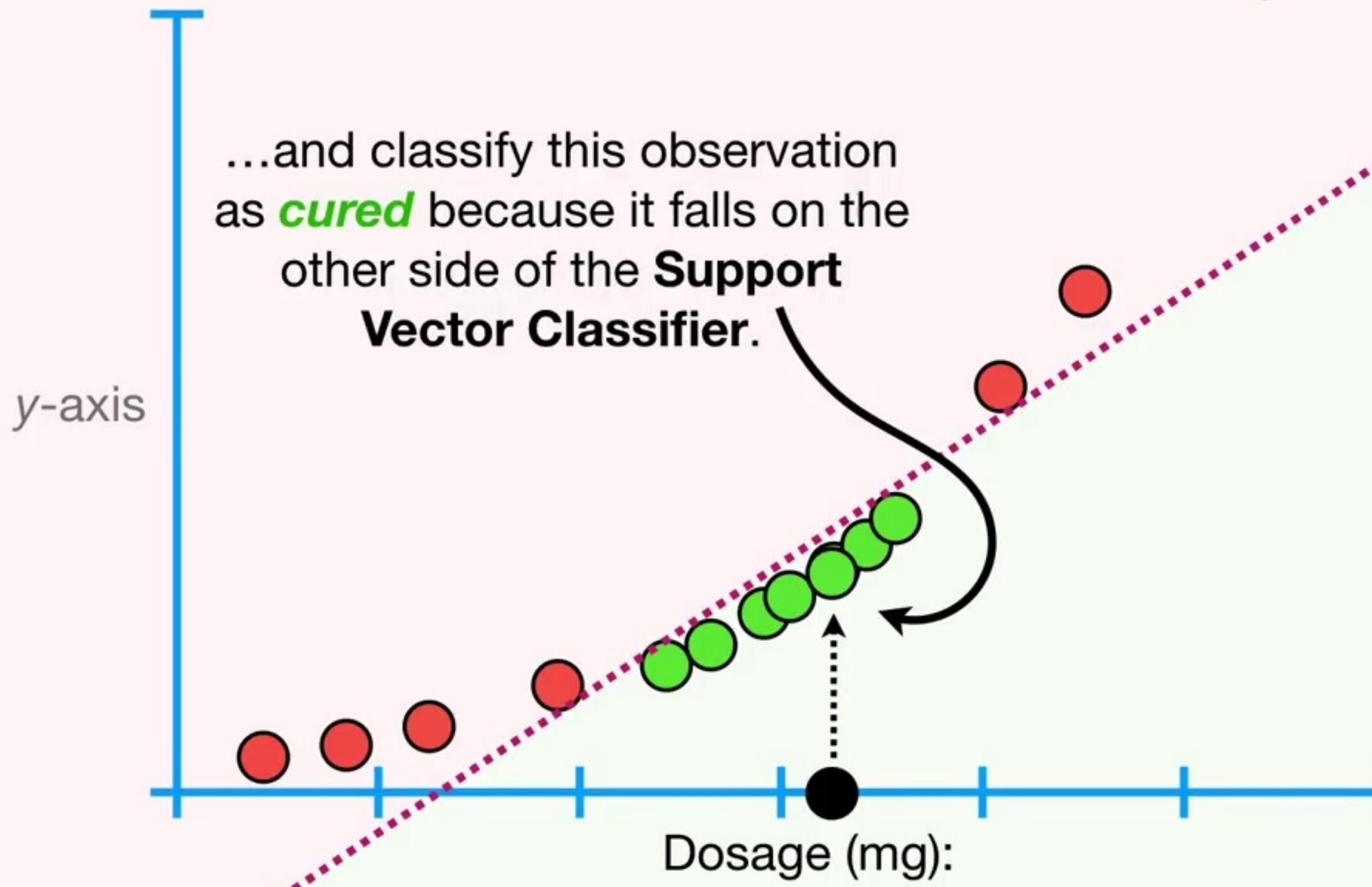
...and classify the observation
as ***not cured*** because it ended
up on this side of the **Support
Vector Classifier**.

Dosage (mg):



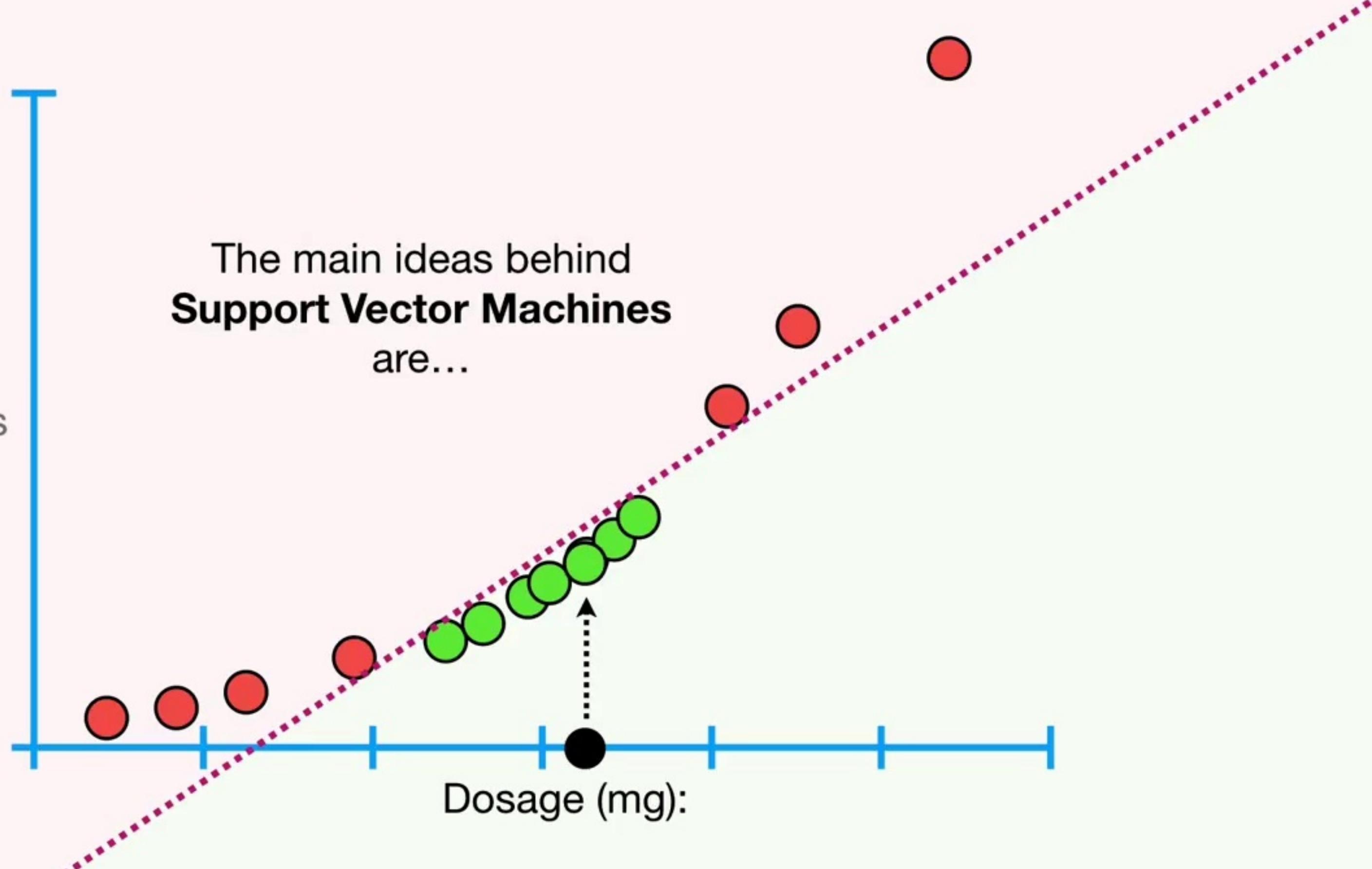


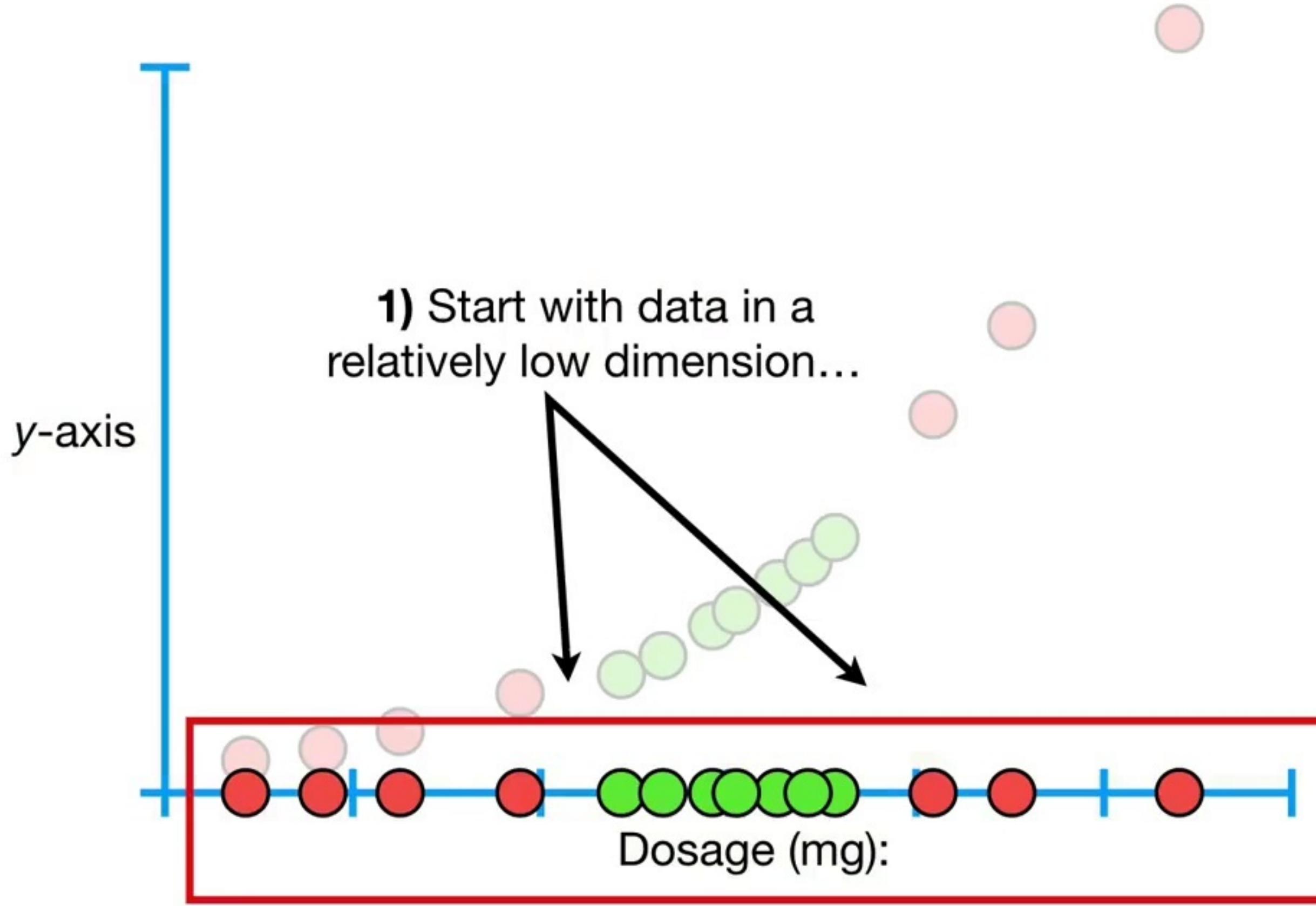


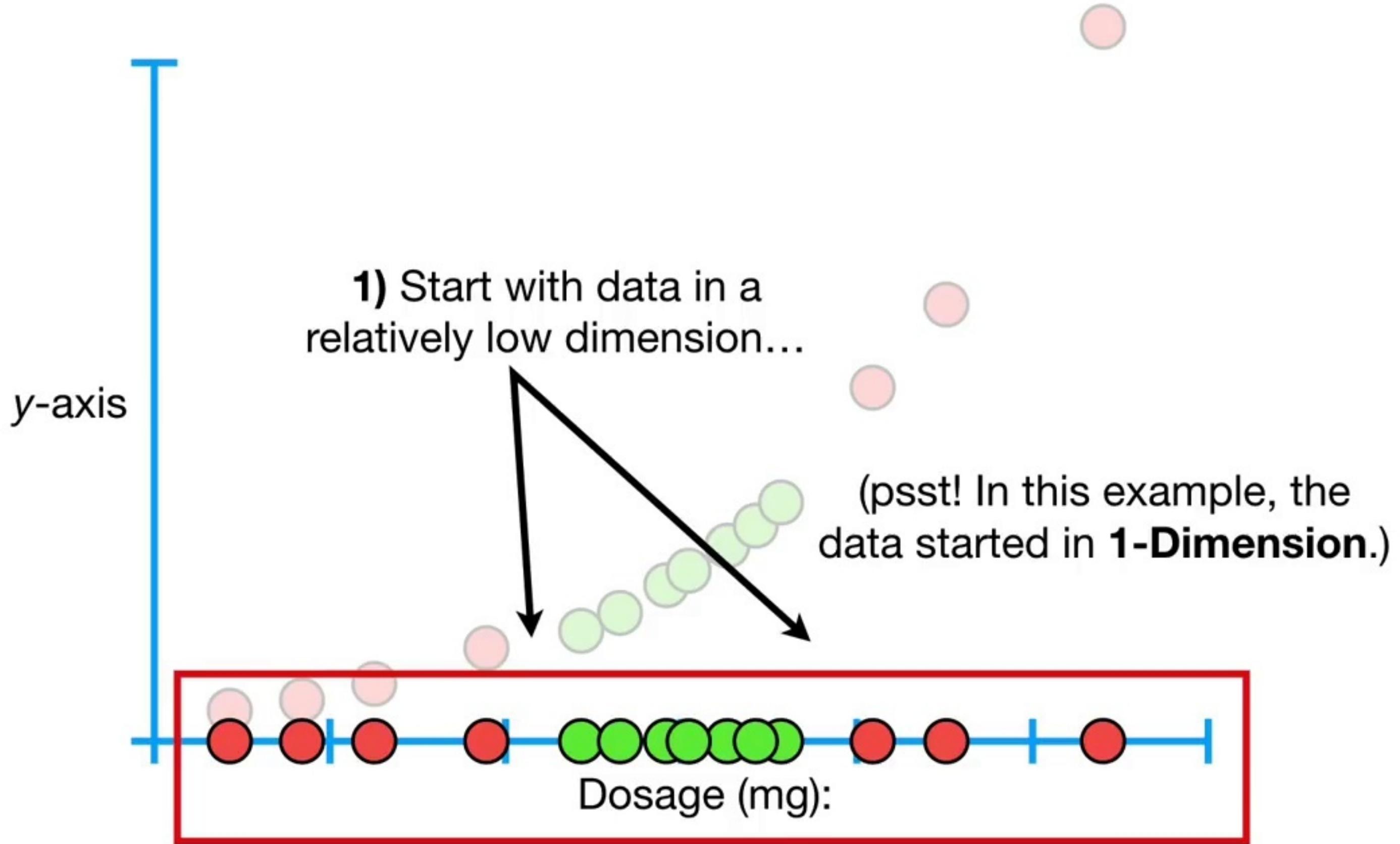


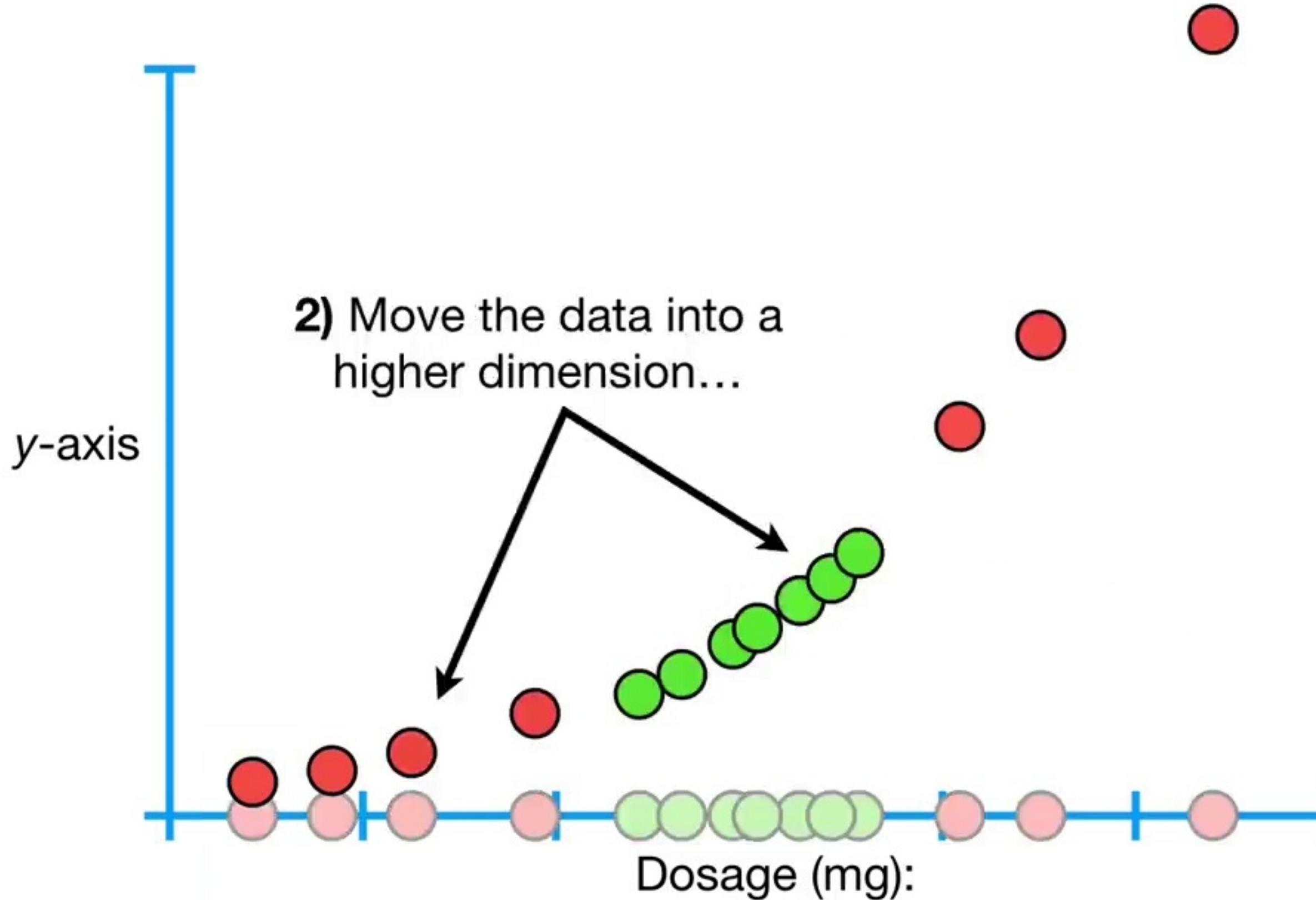
y-axis

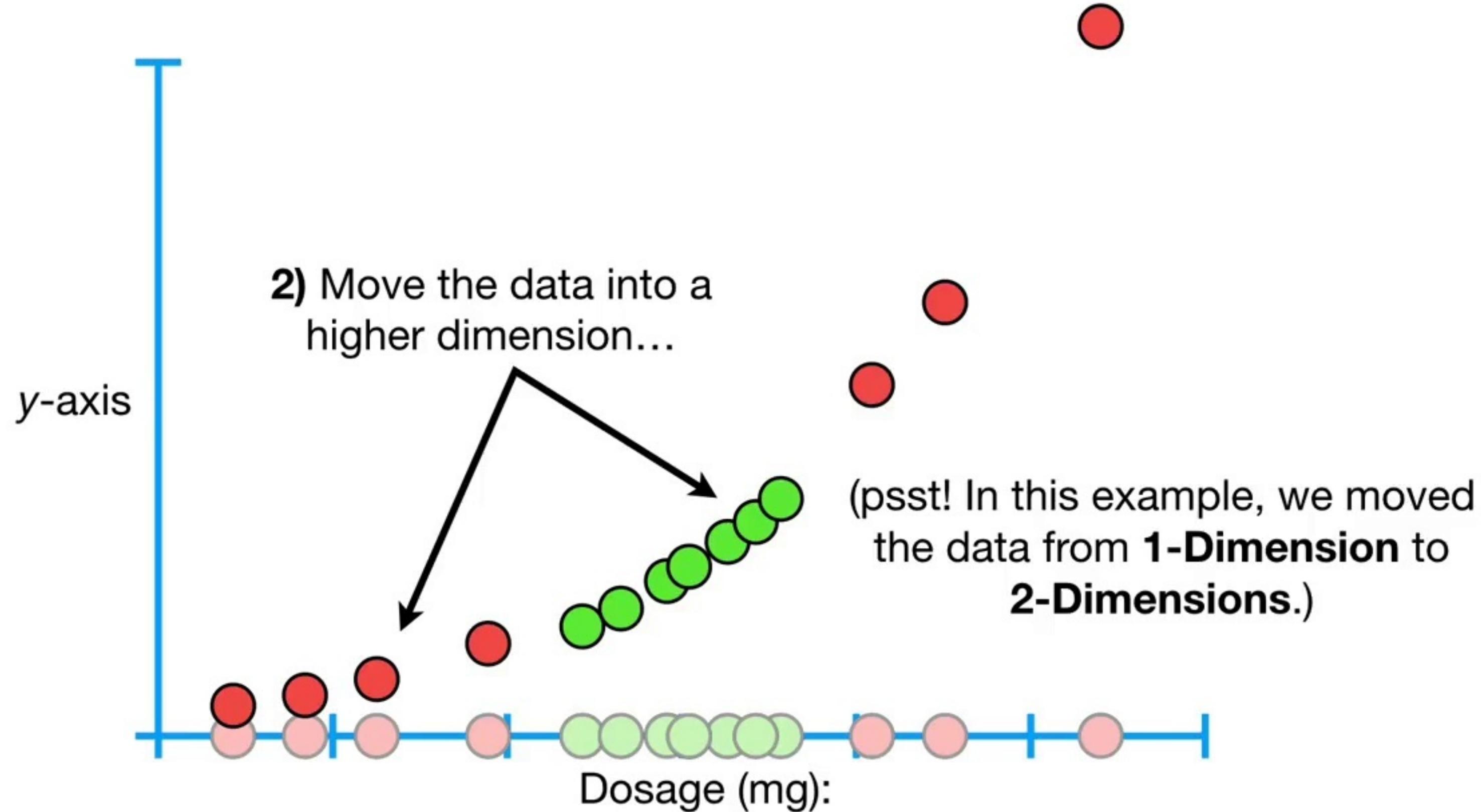
The main ideas behind
Support Vector Machines
are...

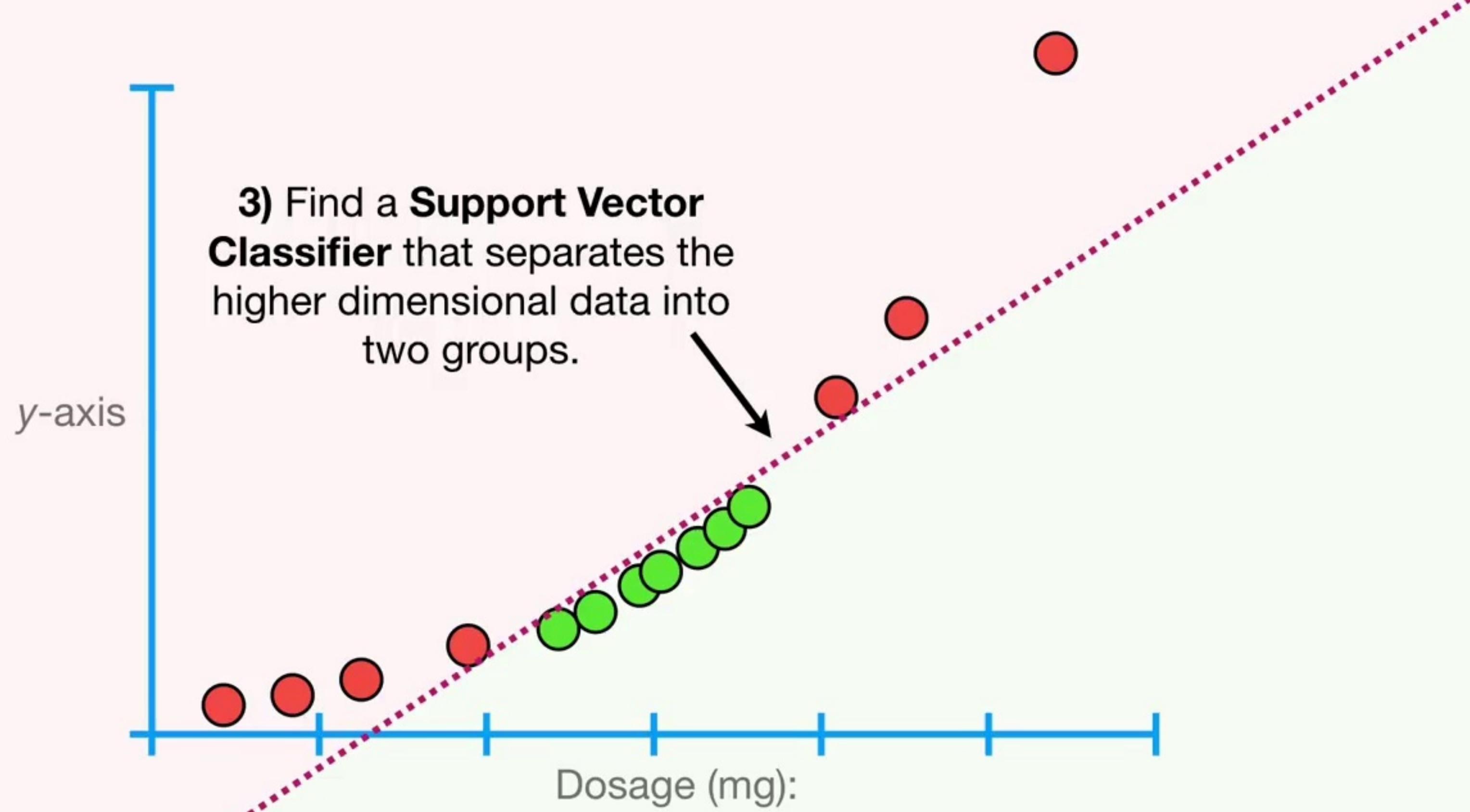












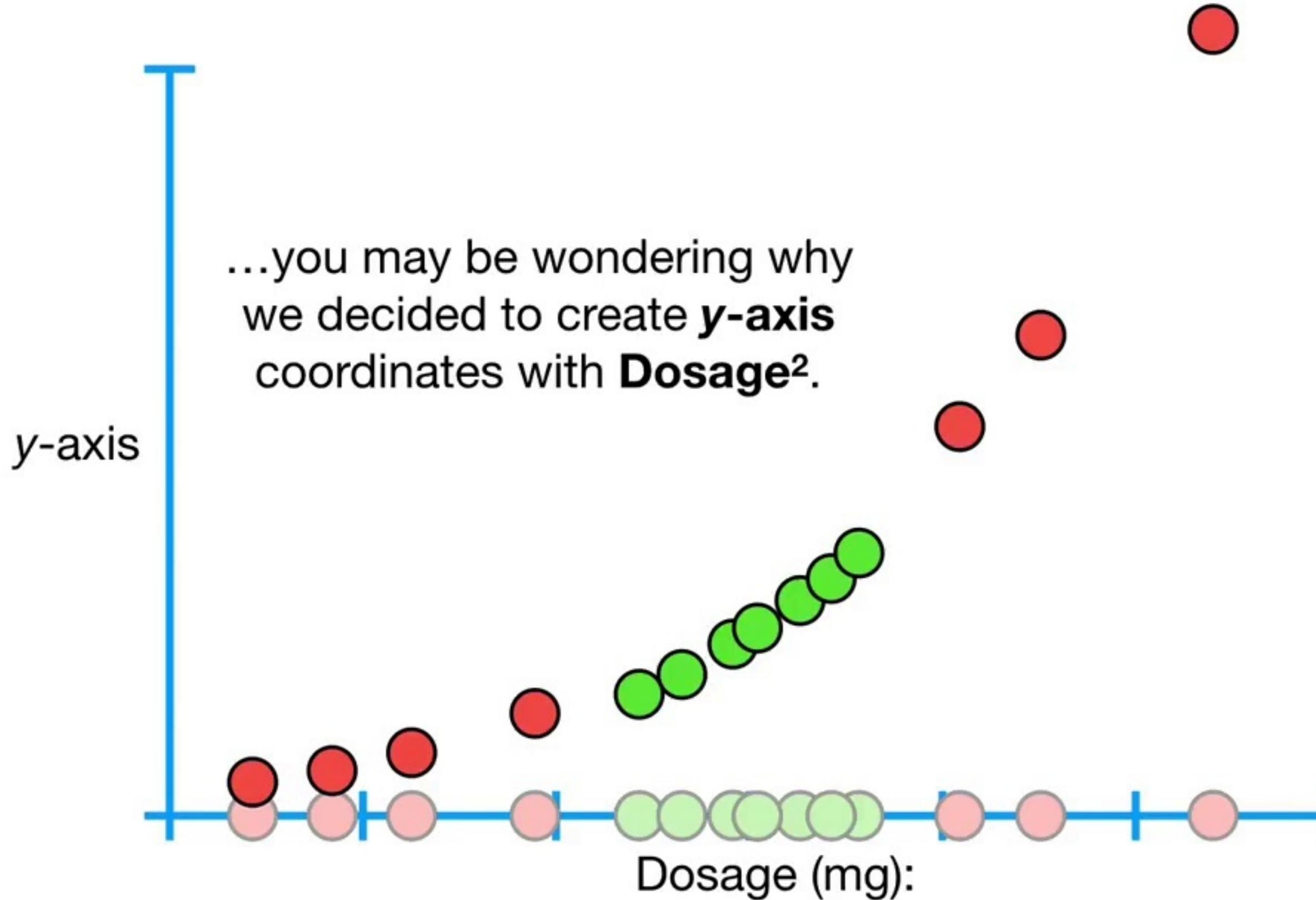
y-axis

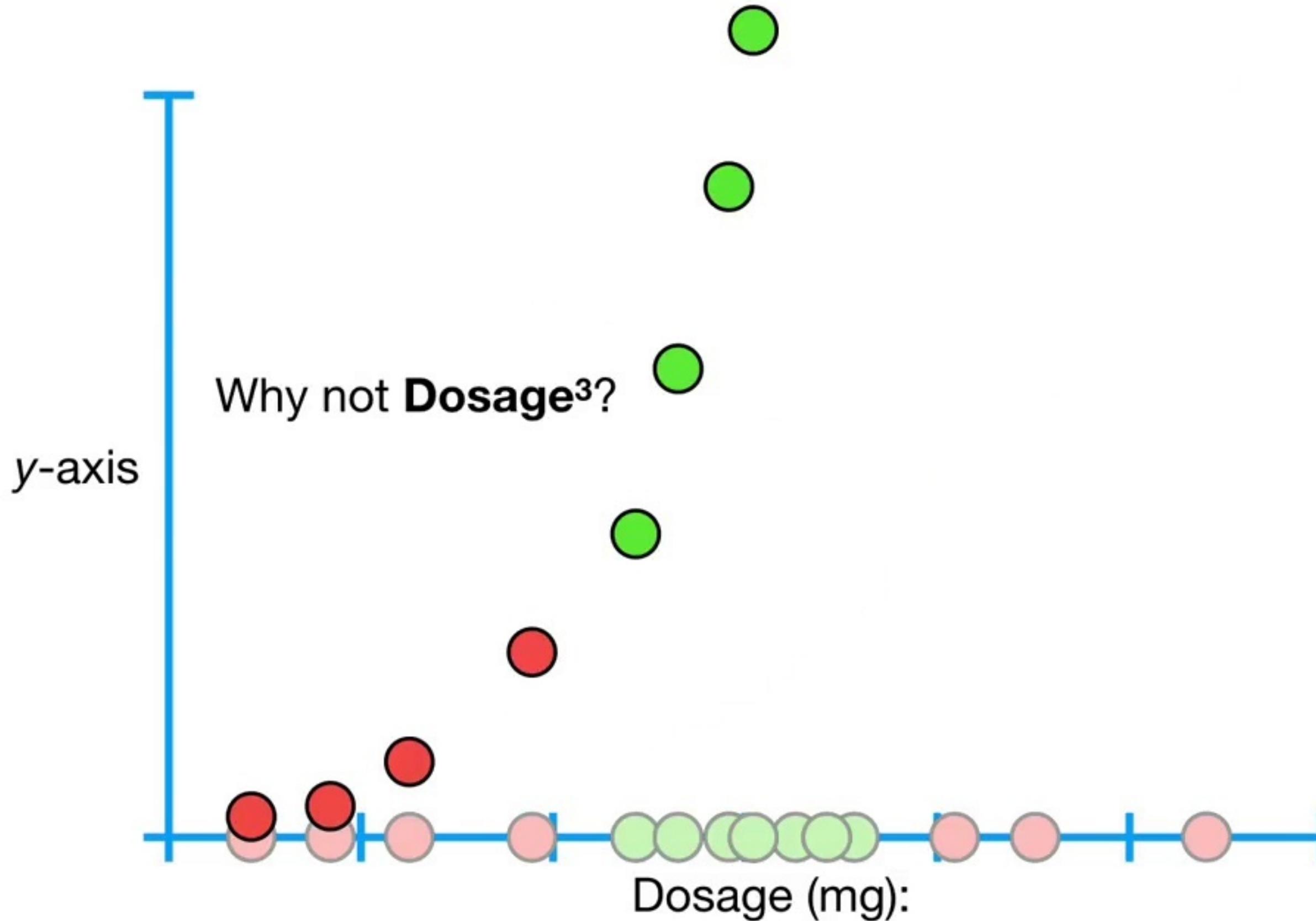
That's all there is to it.

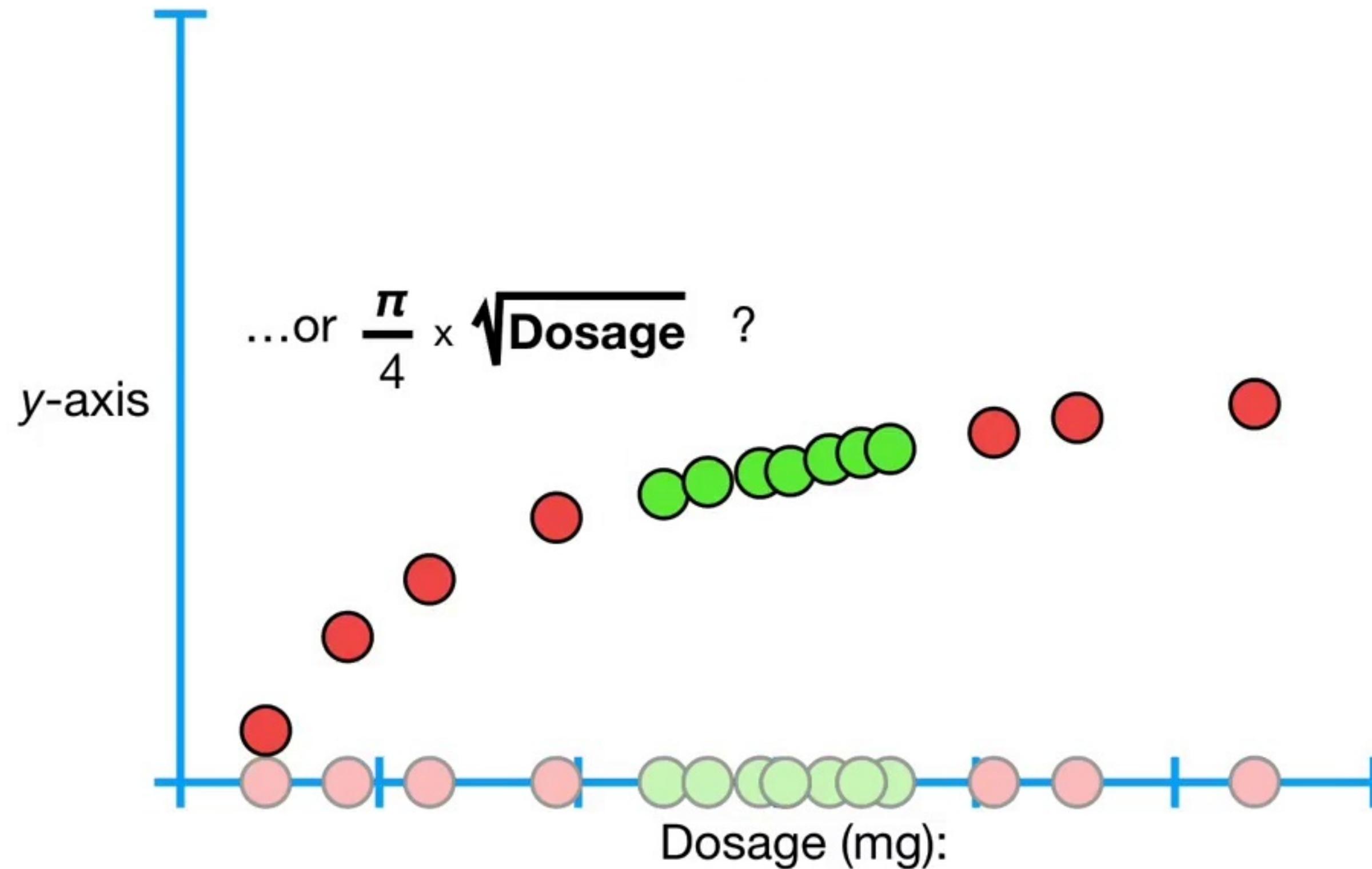
Dosage (mg):

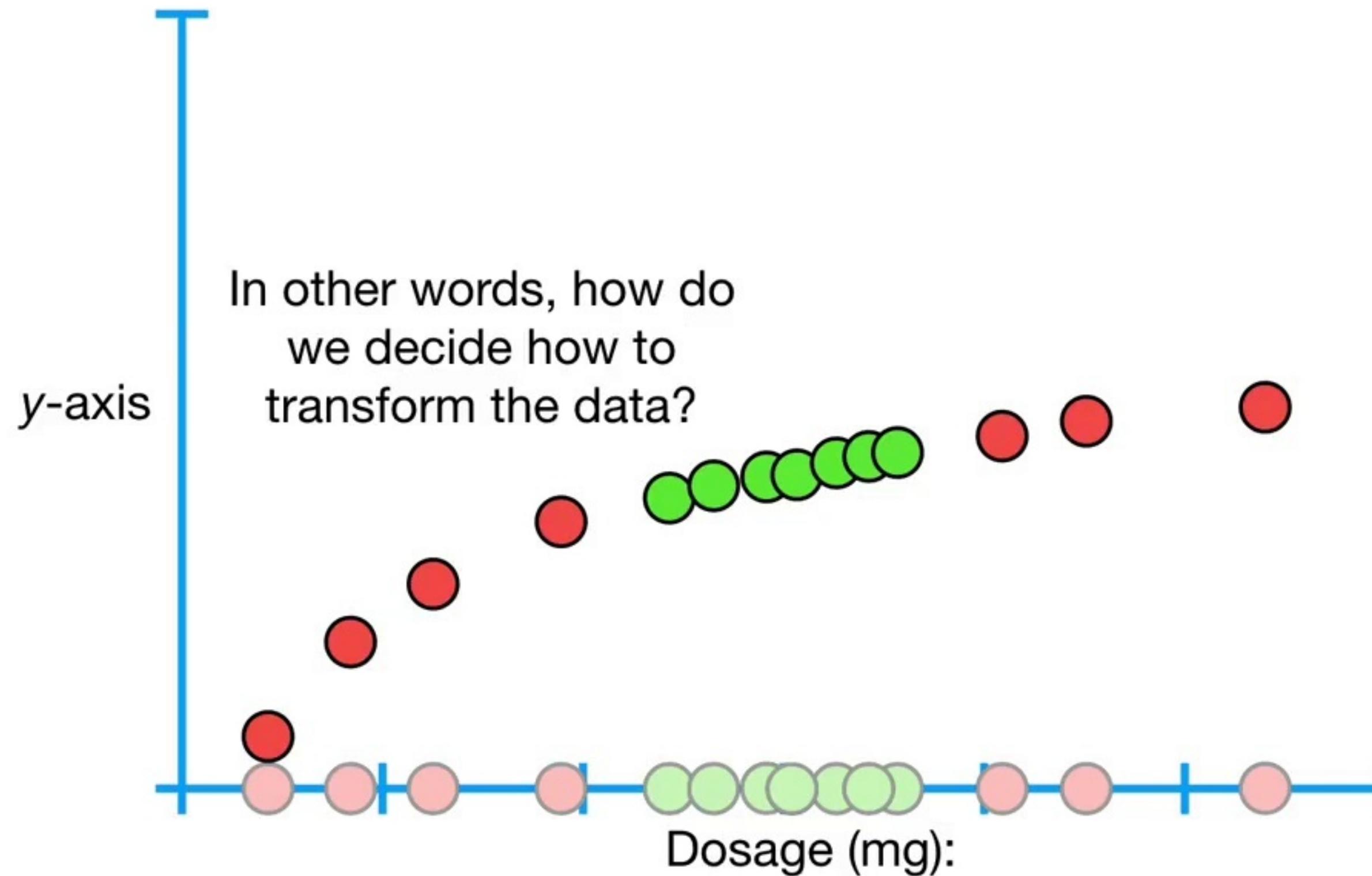
Going back to the original
1-Dimensional data...

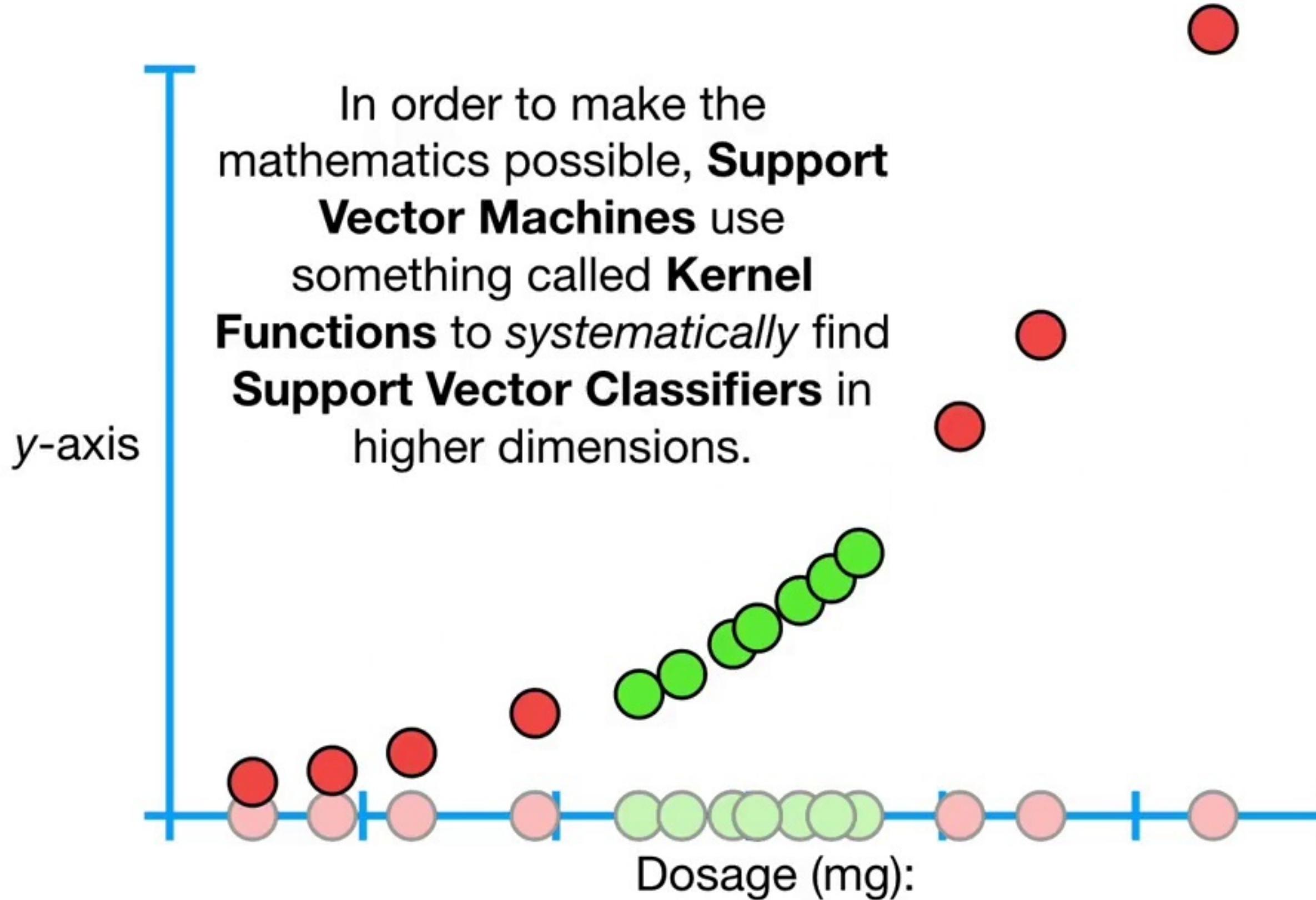






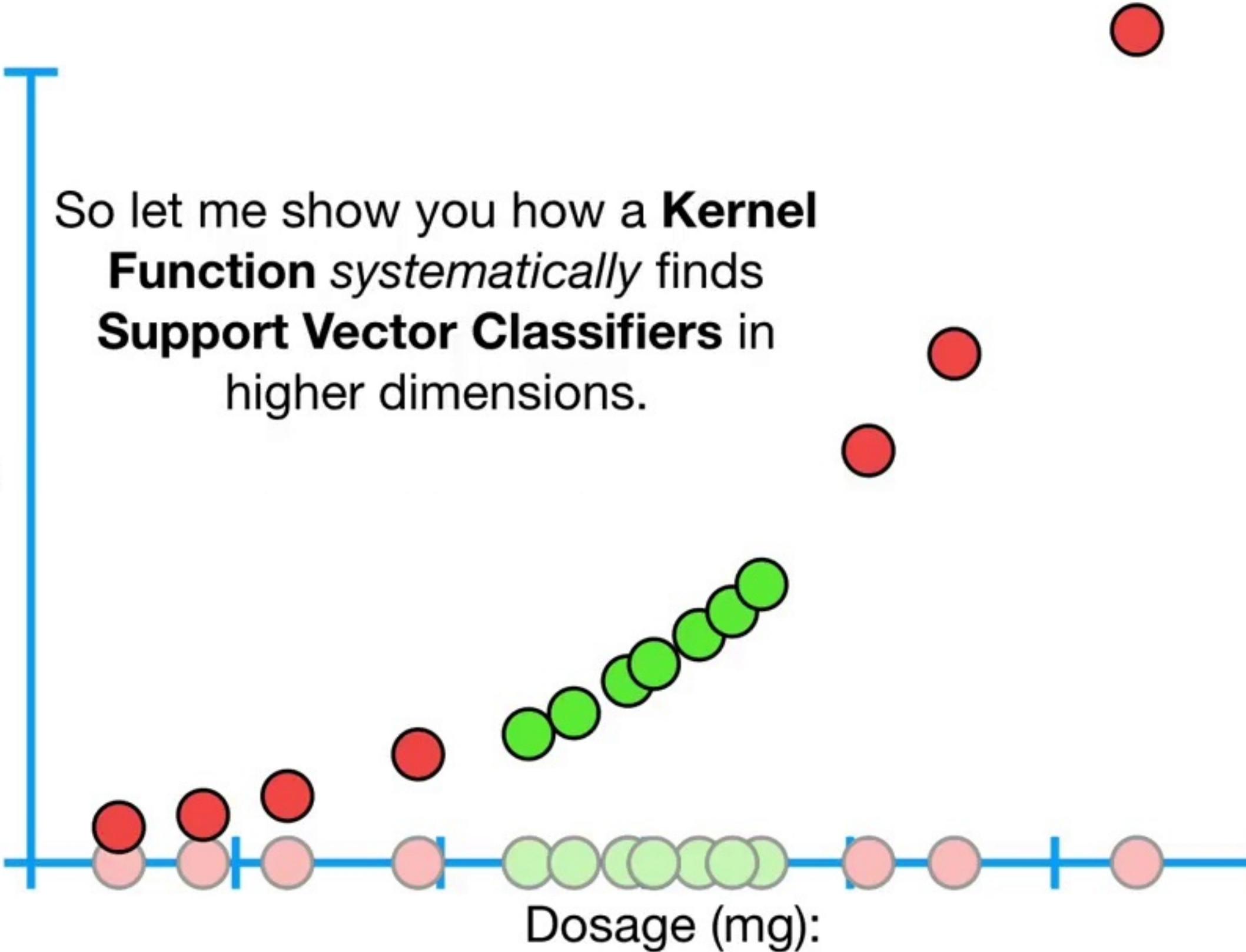


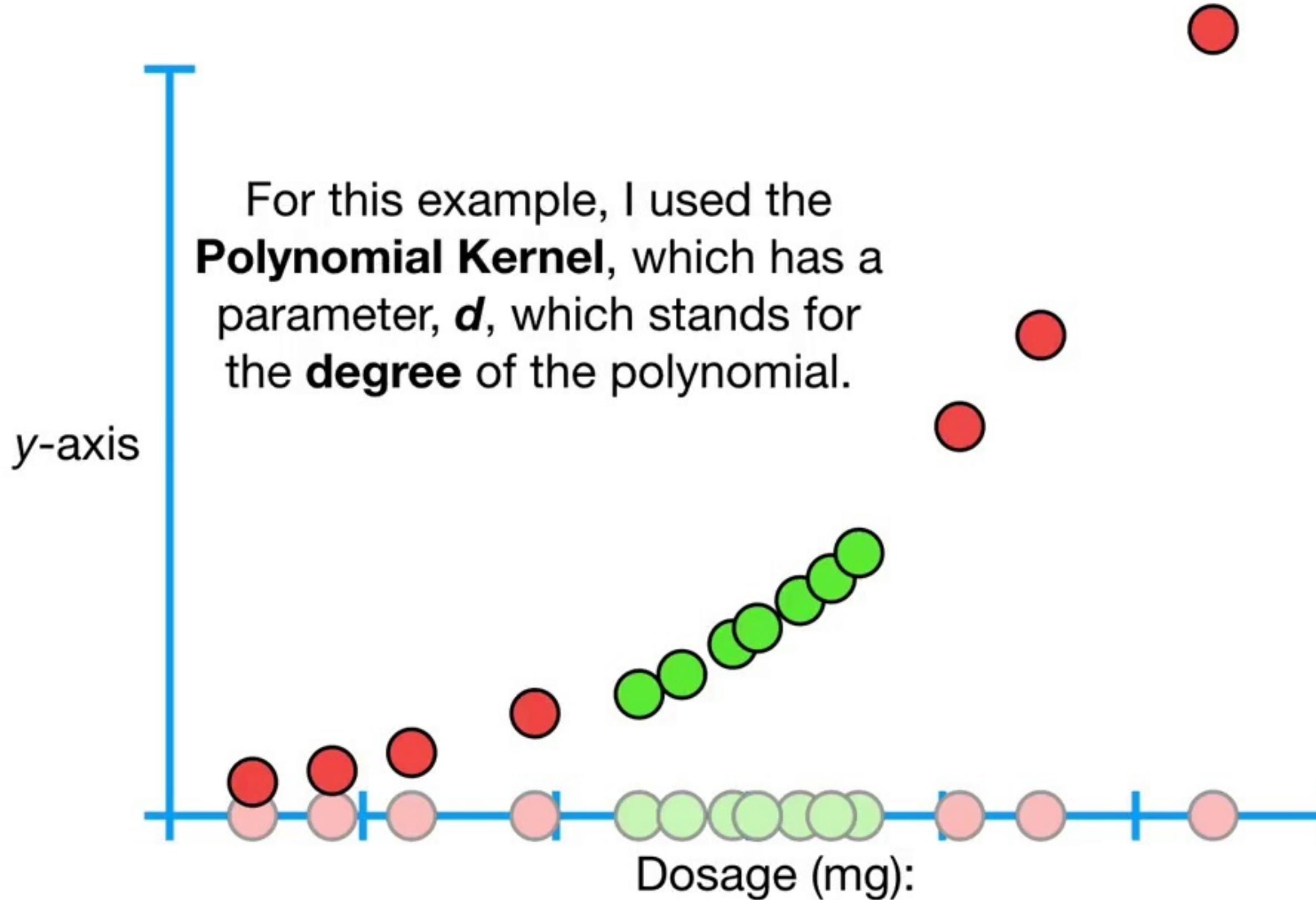




So let me show you how a **Kernel Function** systematically finds **Support Vector Classifiers** in higher dimensions.

y-axis

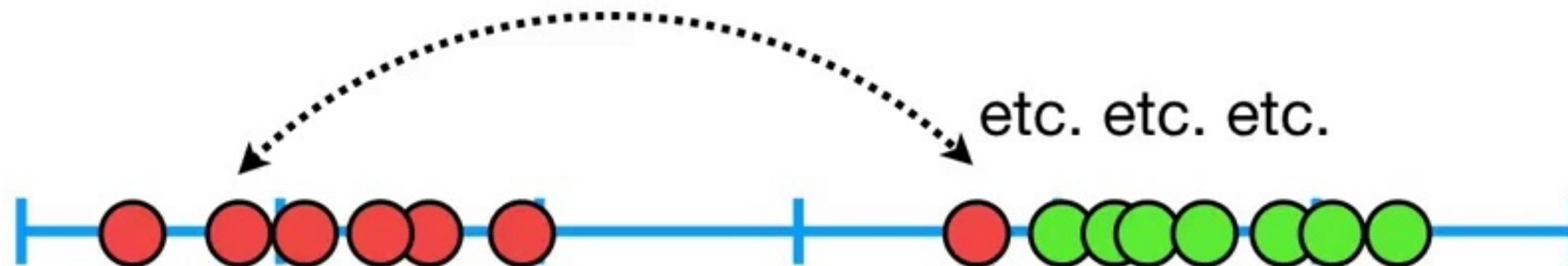




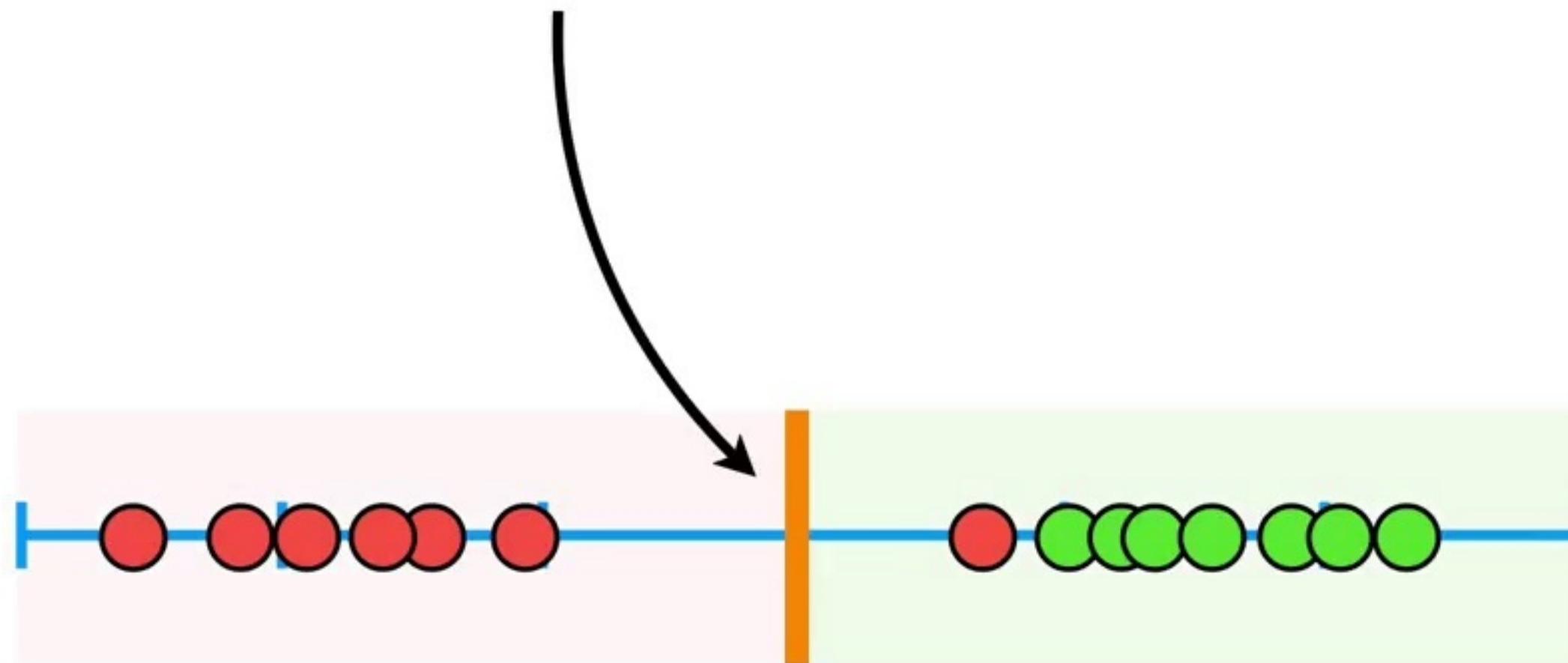
When $d = 1$, the **Polynomial Kernel** computes the relationships between each pair of observations in **1-Dimension**...

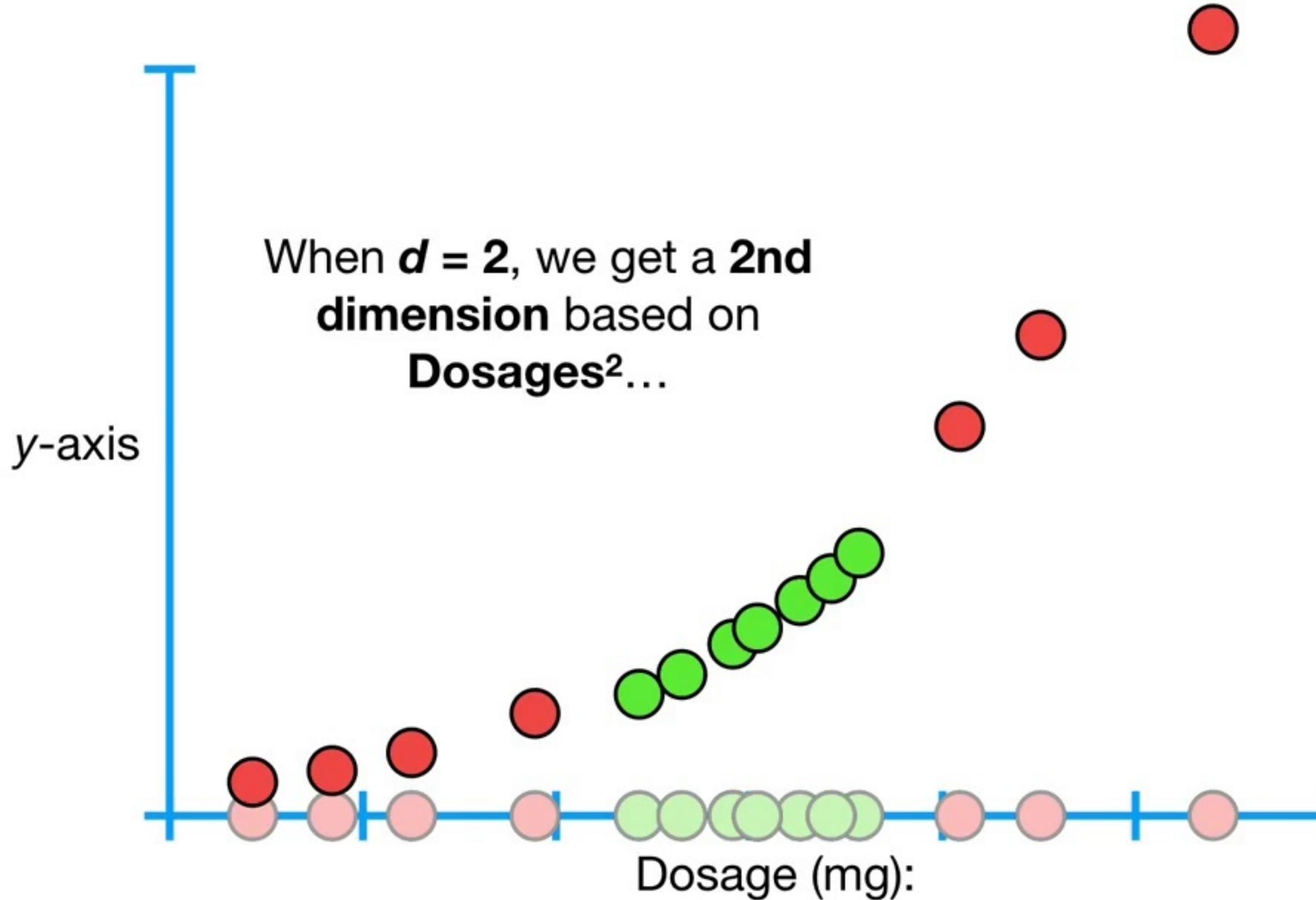


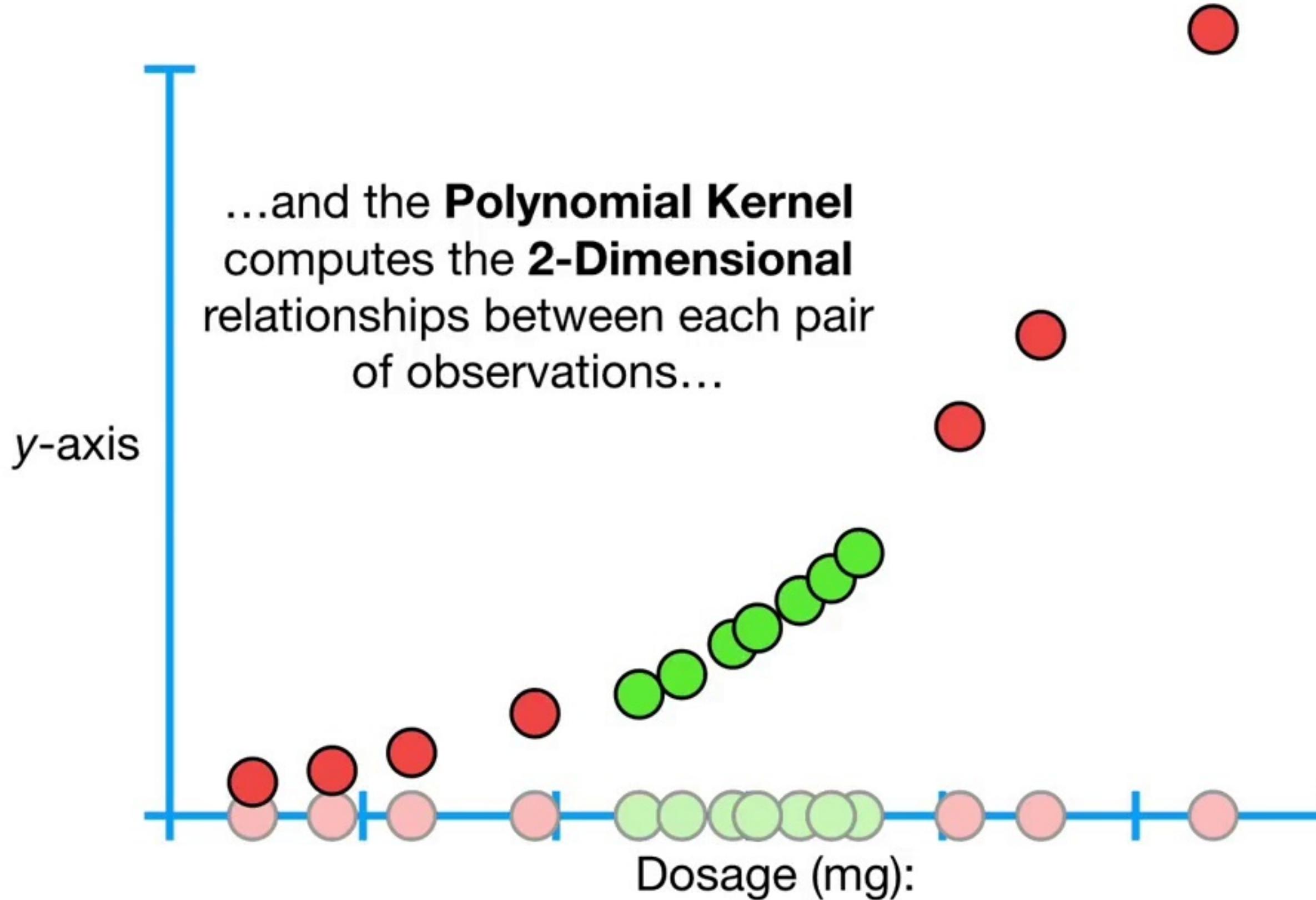
When $d = 1$, the **Polynomial Kernel** computes the relationships between each pair of observations in **1-Dimension**...

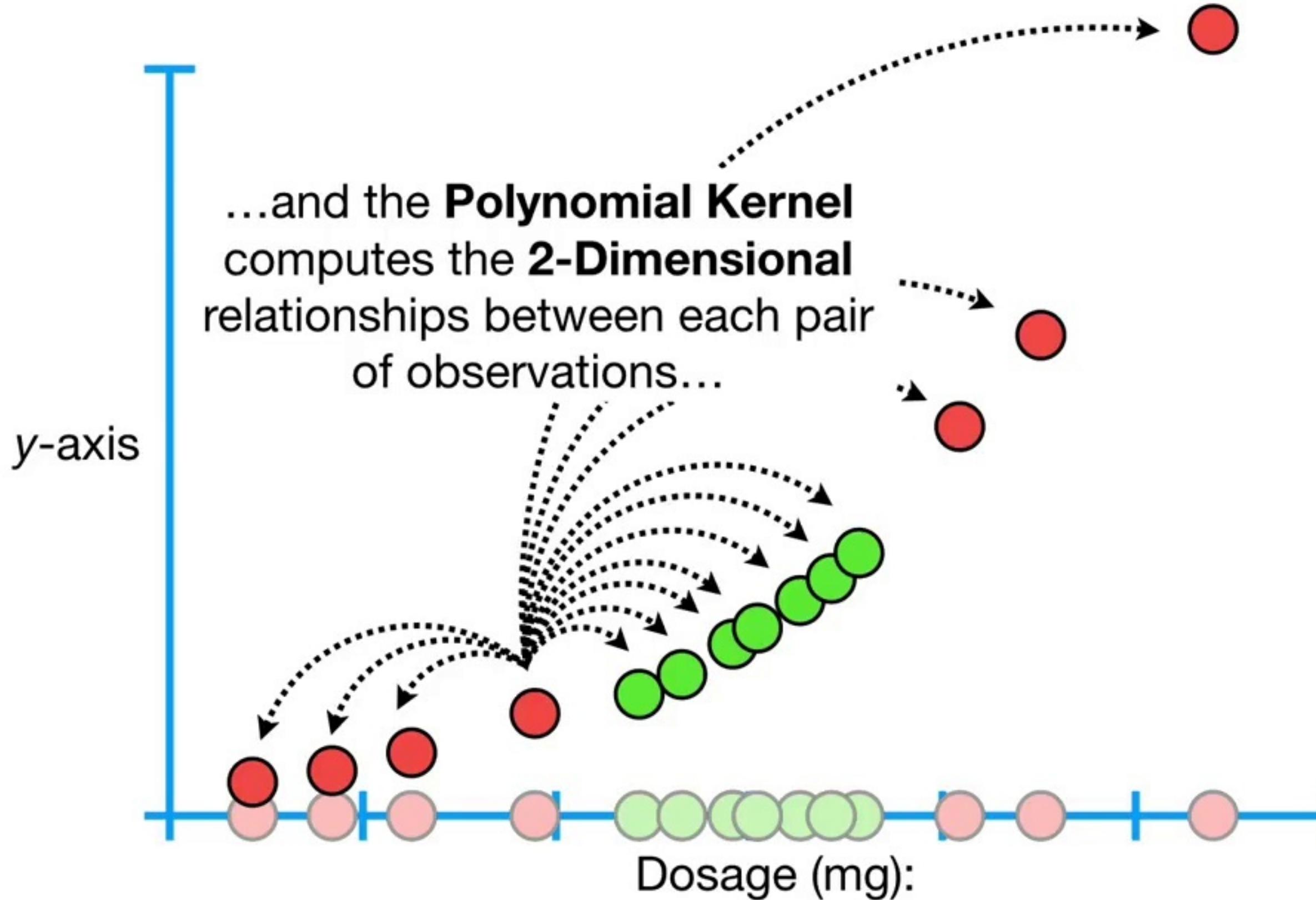


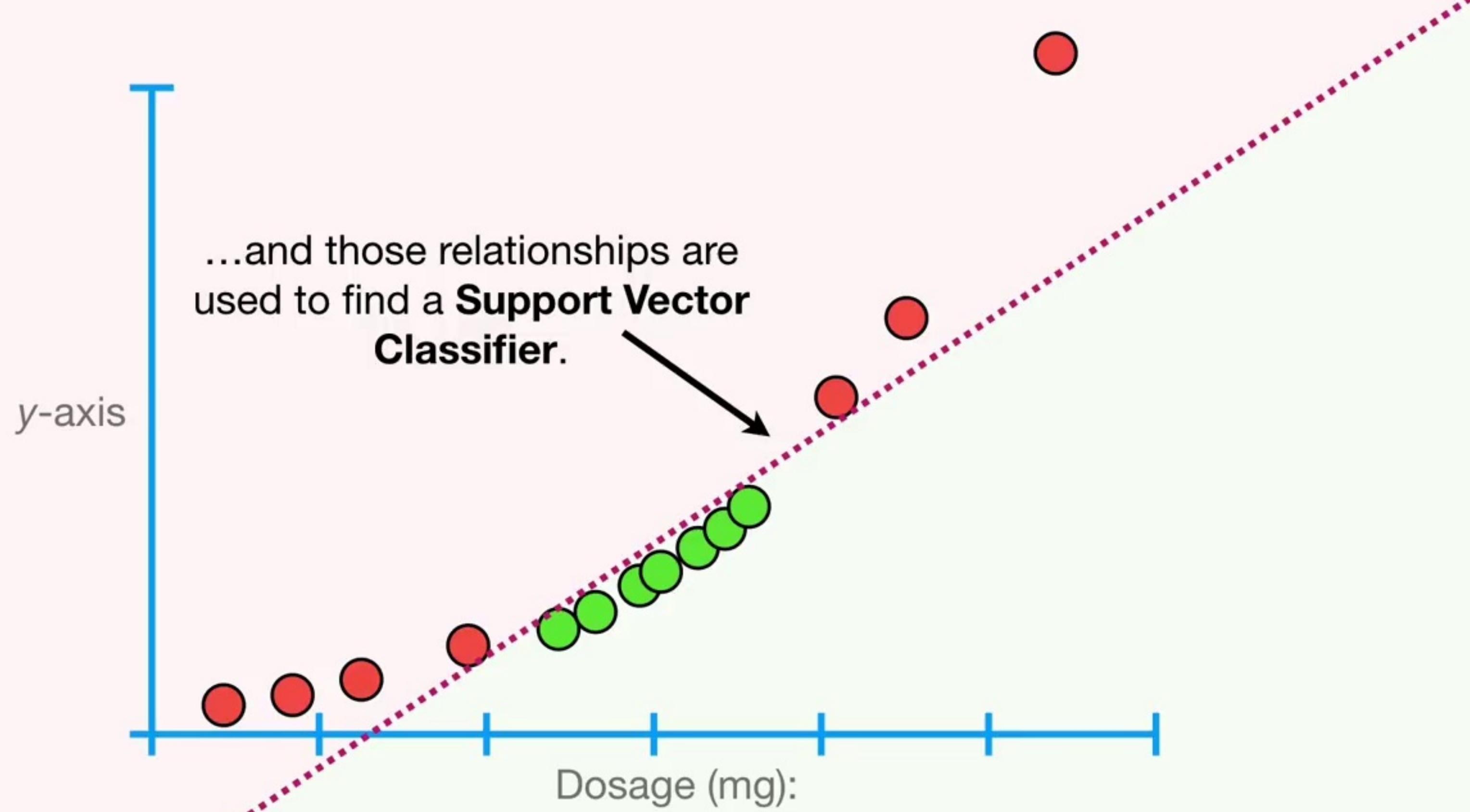
...and these relationships are used to find a **Support Vector Classifier**.



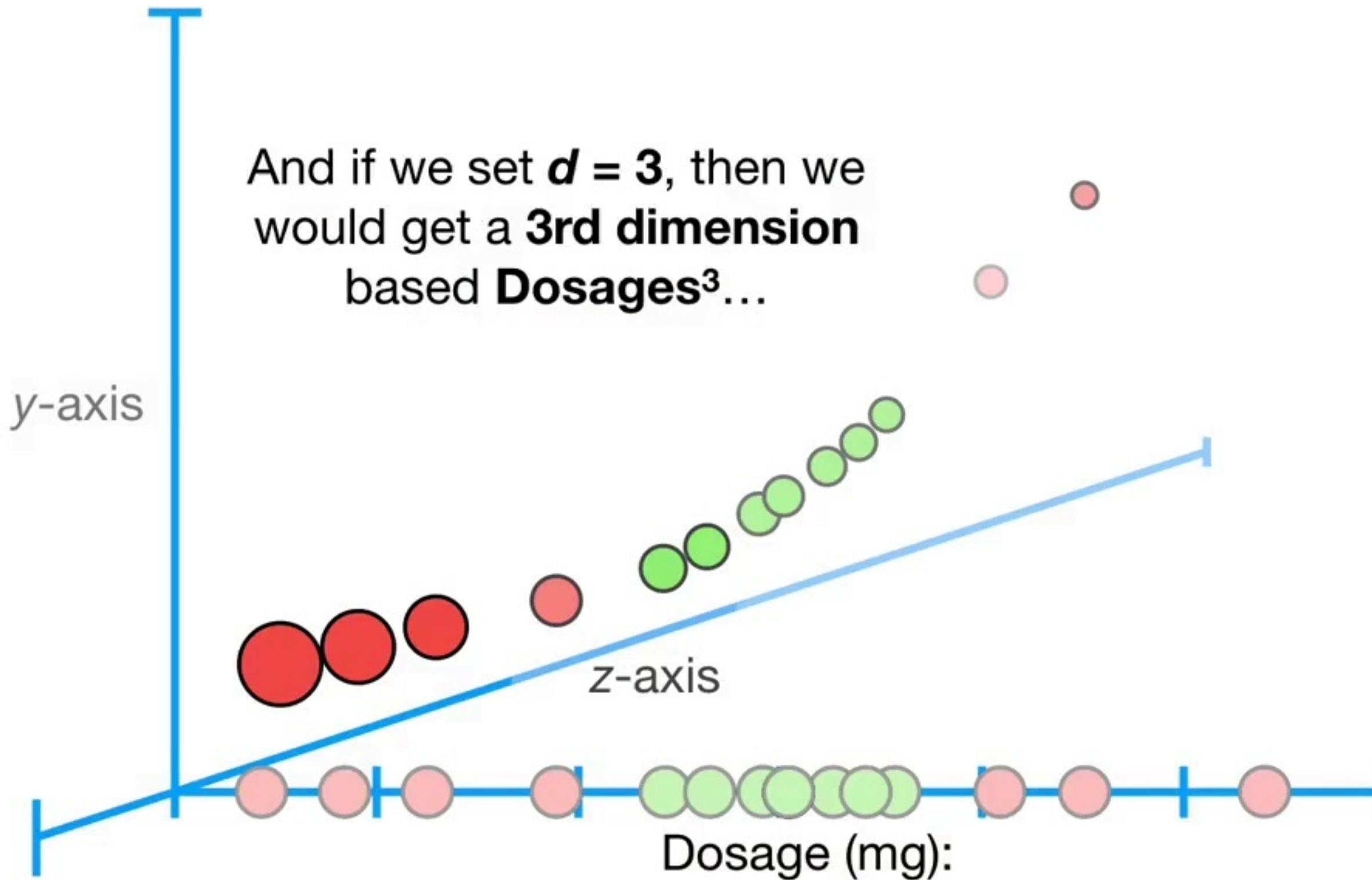




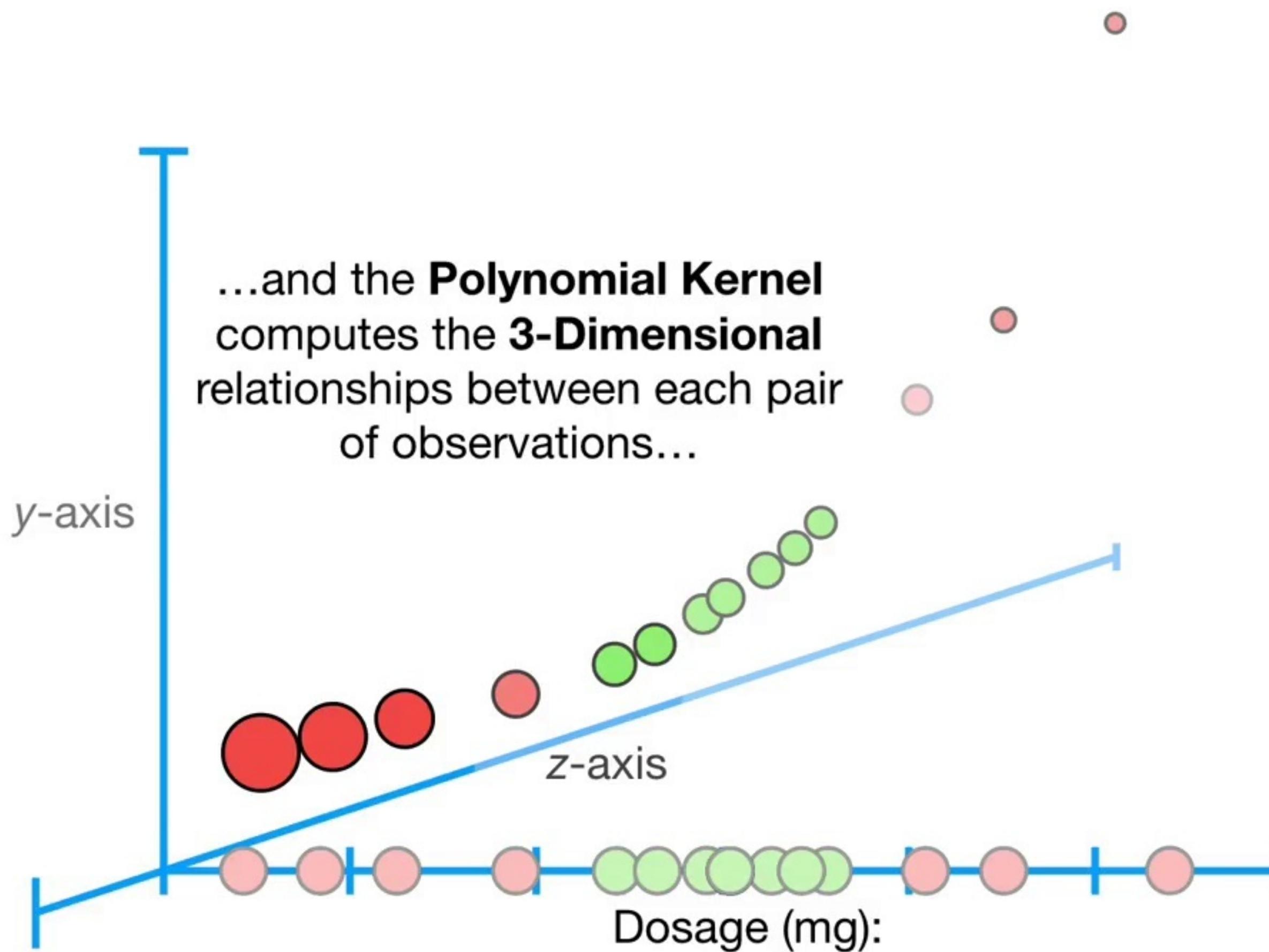




And if we set $d = 3$, then we would get a **3rd dimension** based **Dosages³**...



...and the **Polynomial Kernel**
computes the **3-Dimensional**
relationships between each pair
of observations...

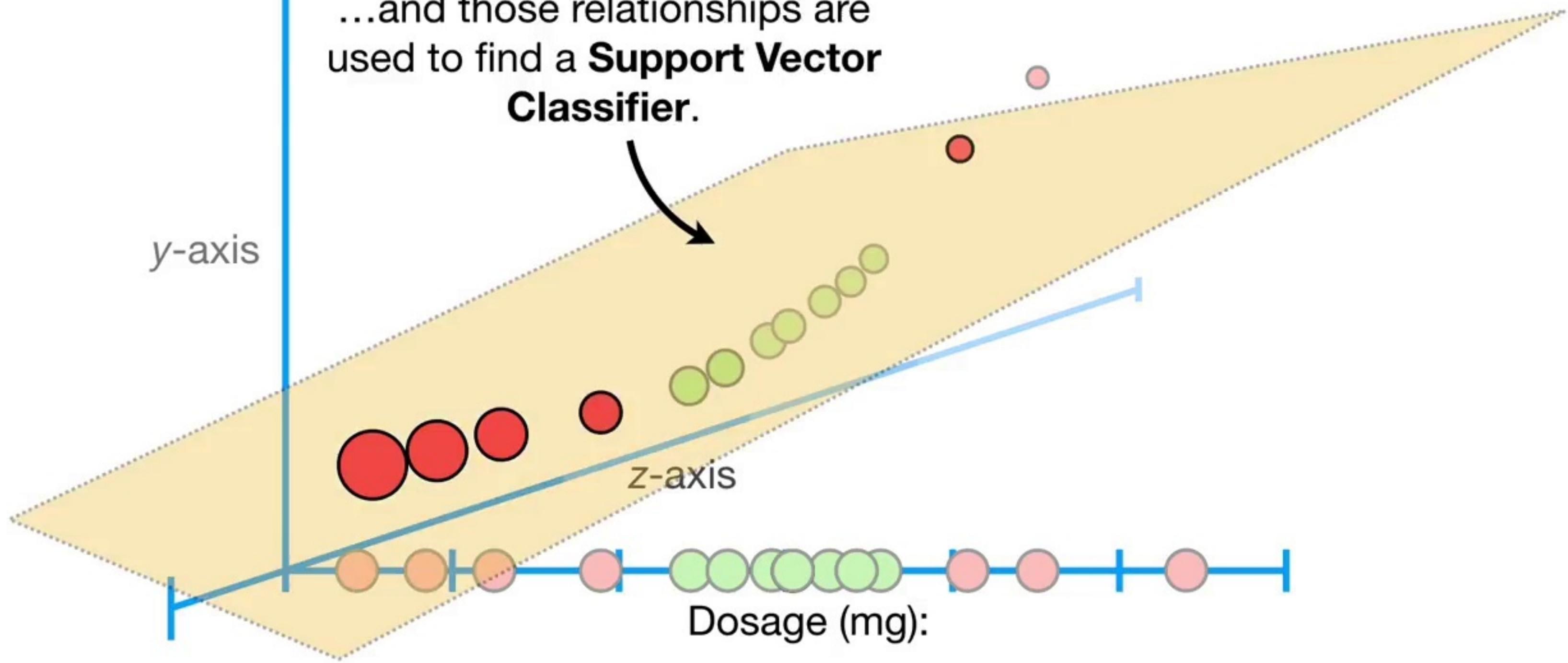


...and those relationships are used to find a **Support Vector Classifier**.

y-axis

z-axis

Dosage (mg):



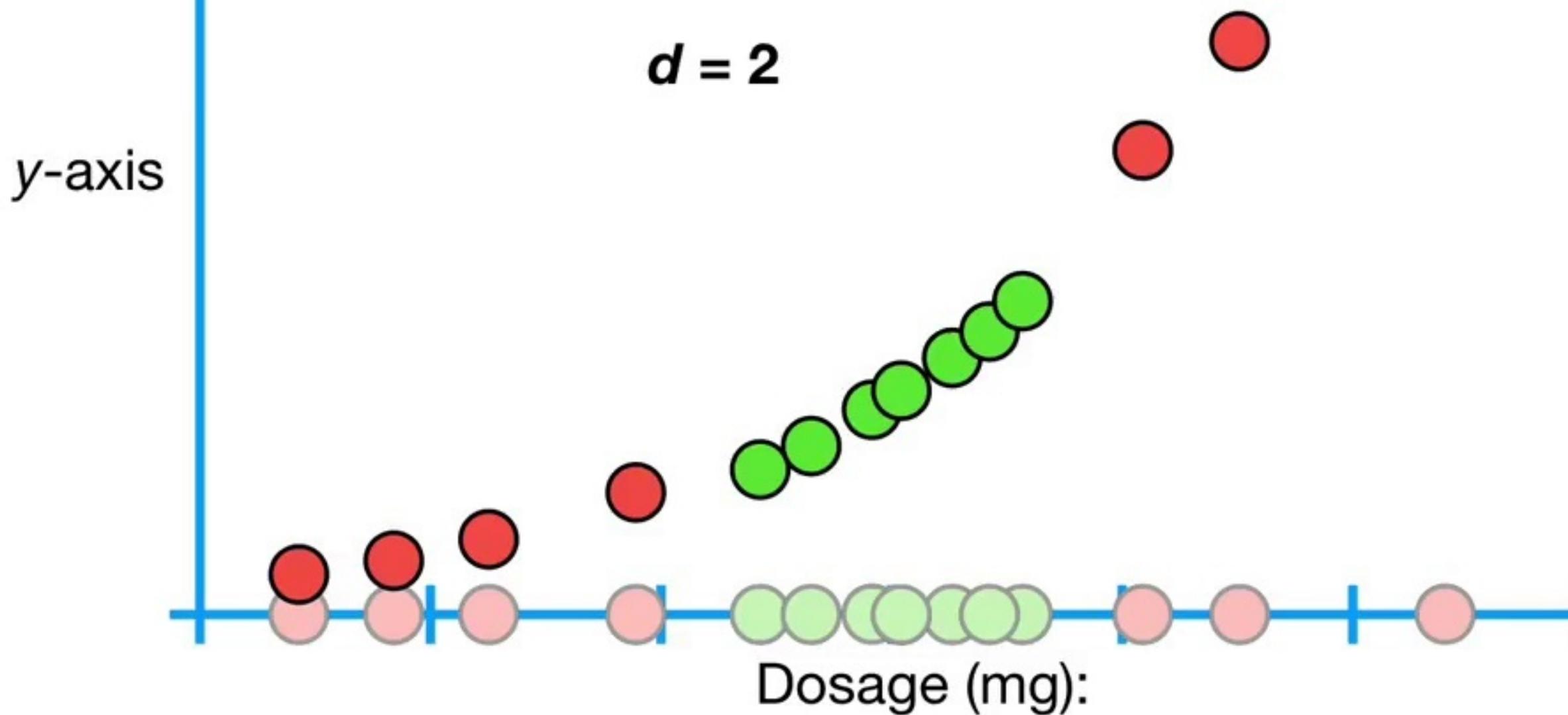
And when **$d = 4$ or more**, then we get even more dimensions to find a **Support Vector Classifier**.

In summary, the **Polynomial Kernel** systematically increases dimensions by setting d , the degree of the polynomial...

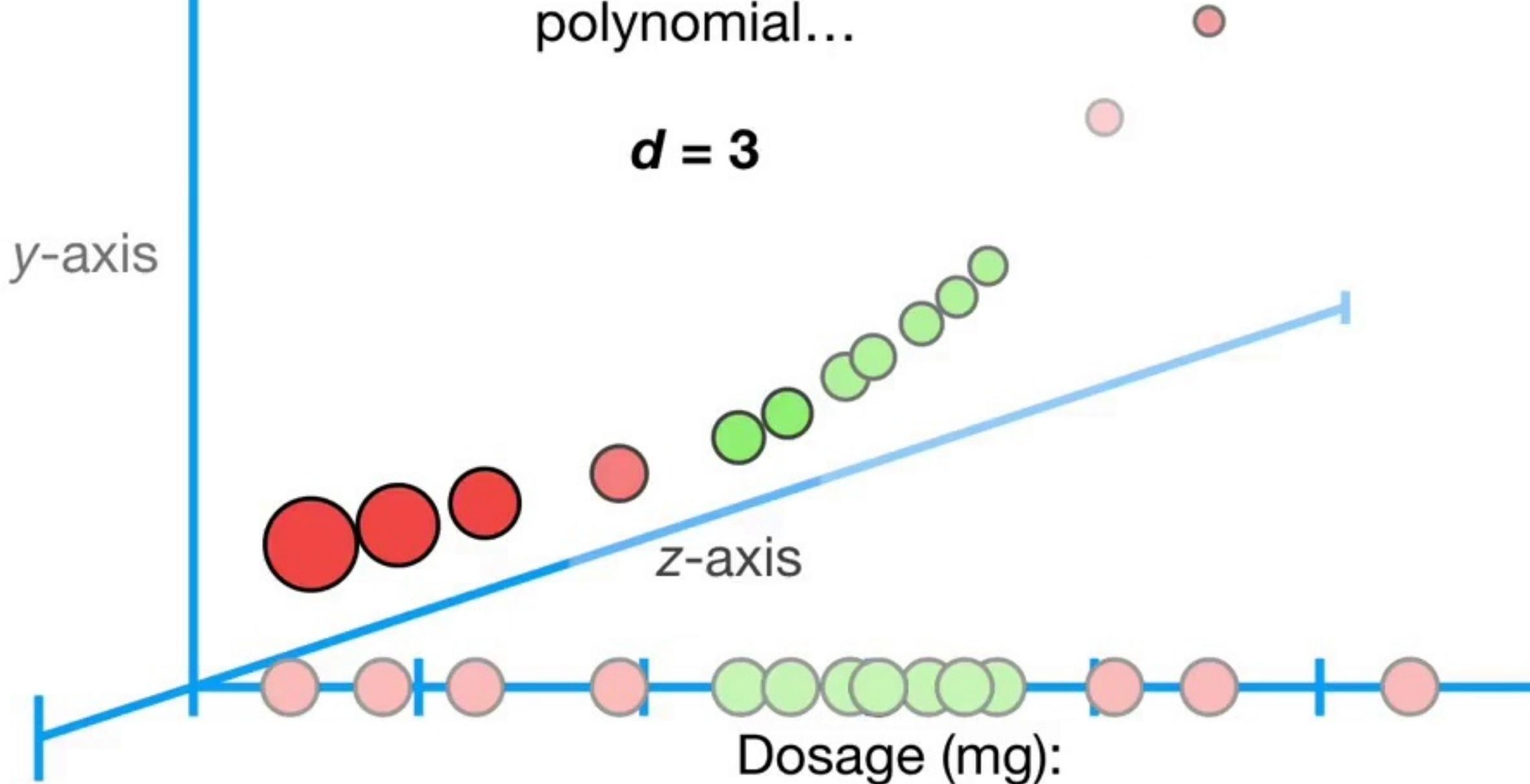
$$d = 1$$



In summary, the **Polynomial Kernel** systematically increases dimensions by setting d , the degree of the polynomial...



In summary, the **Polynomial Kernel** systematically increases dimensions by setting d , the degree of the polynomial...

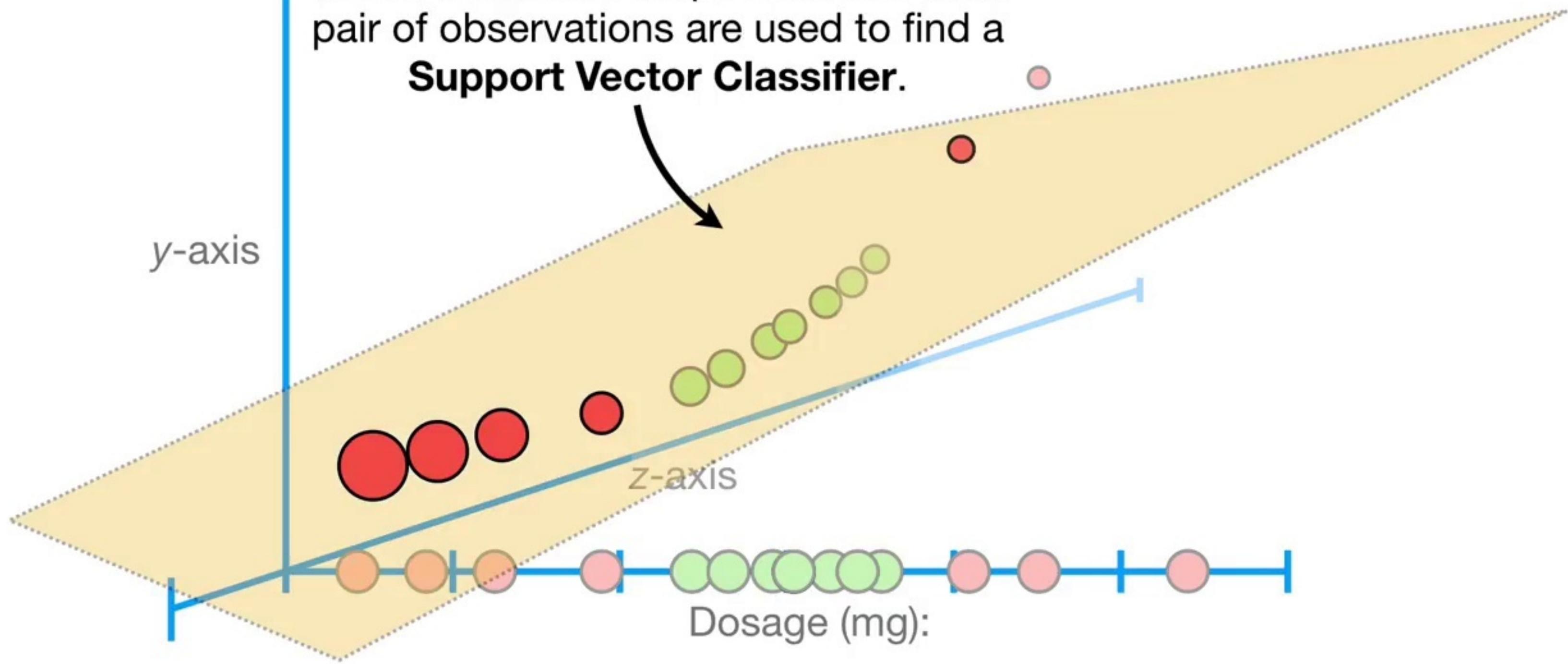


...and the relationships between each pair of observations are used to find a **Support Vector Classifier**.

y-axis

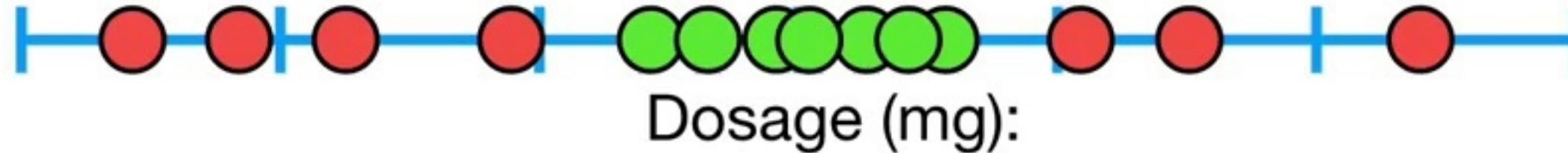
z-axis

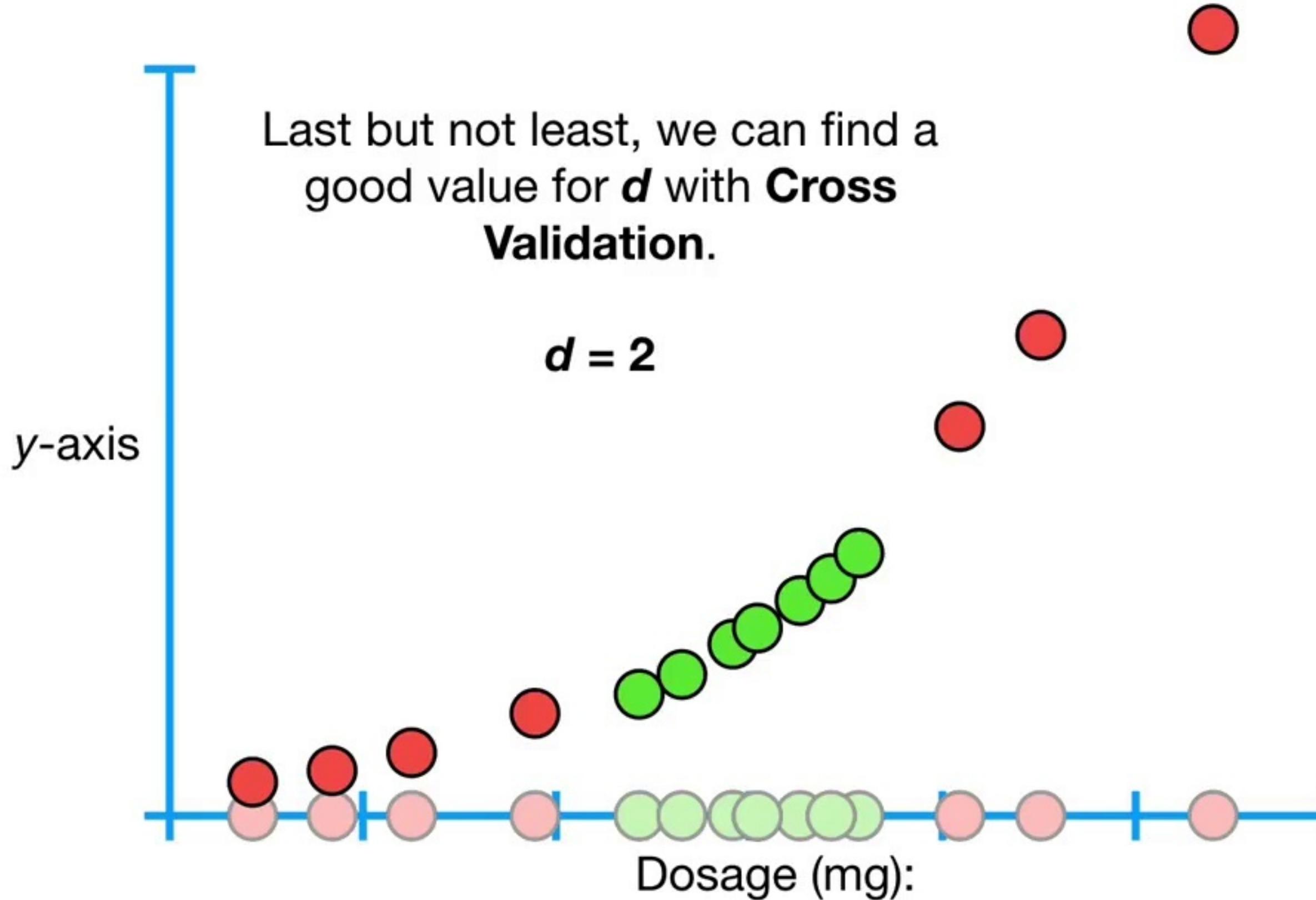
Dosage (mg):



Last but not least, we can find a good value for d with **Cross Validation.**

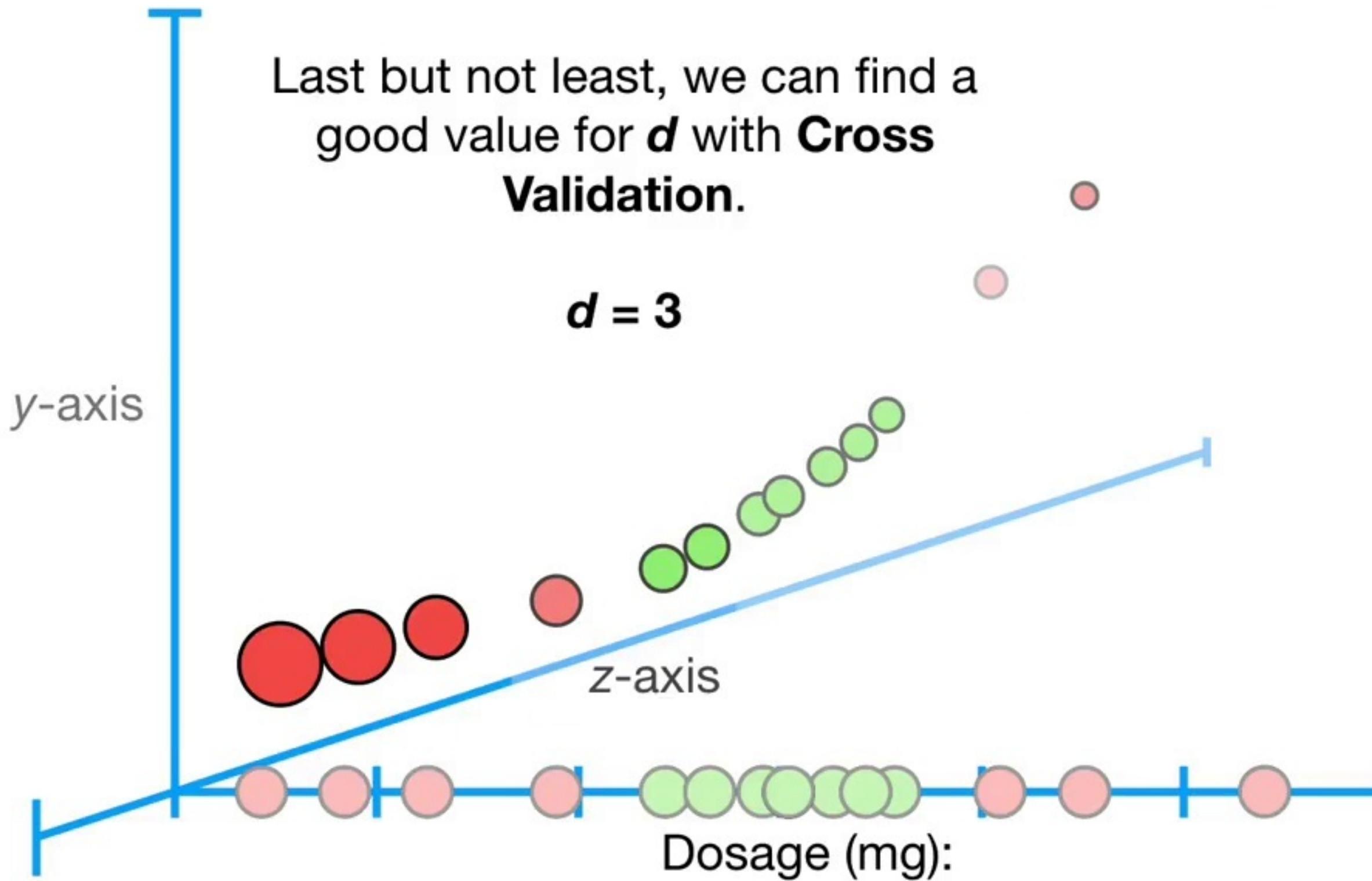
$$d = 1$$





Last but not least, we can find a good value for d with **Cross Validation.**

$$d = 3$$



Last but not least, we can find a good value for d with **Cross Validation.**

y-axis

z-axis

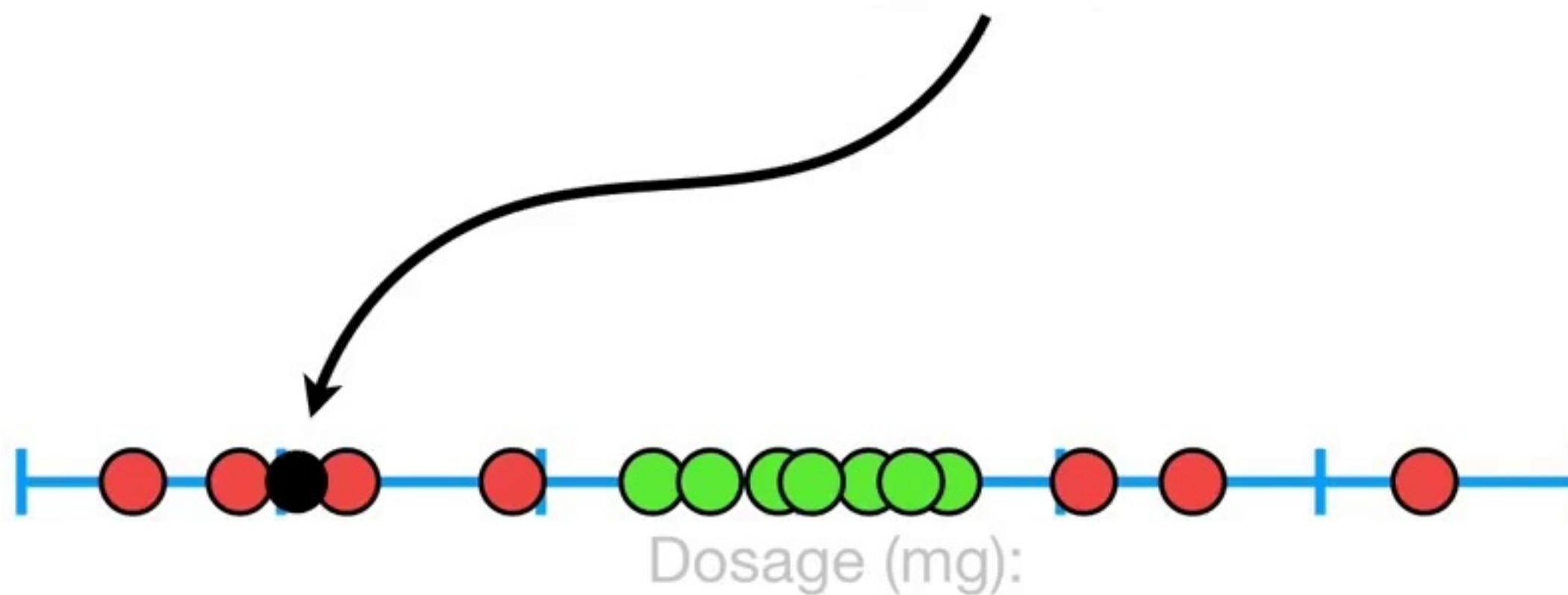
Dosage (mg):

Another very commonly used **Kernel** is the **Radial Kernel**, also known as the **Radial Basis Function (RBF) Kernel**.

Unfortunately, the **Radial Kernel** finds **Support Vector Classifiers** in *infinite dimensions*, so I can't give you an example of what it does exactly.



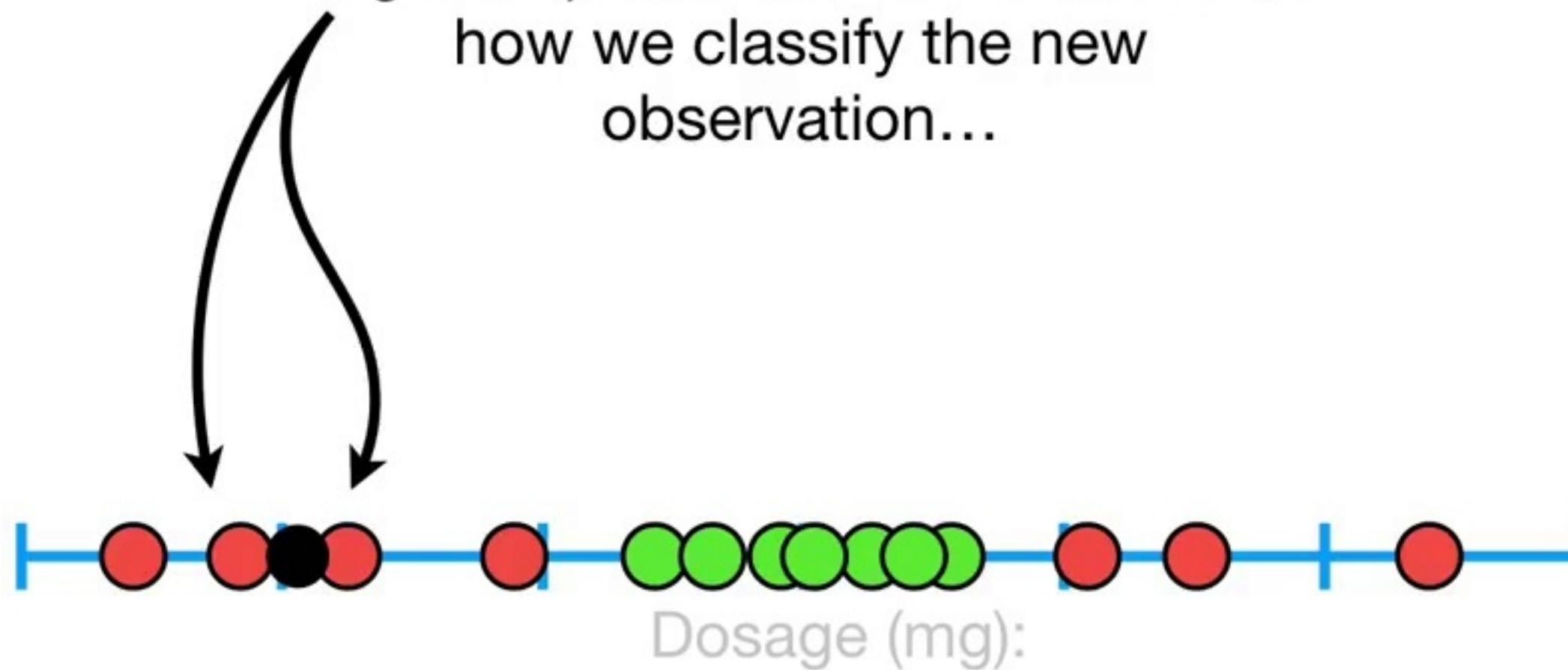
However, when using it on a new
observation like this...



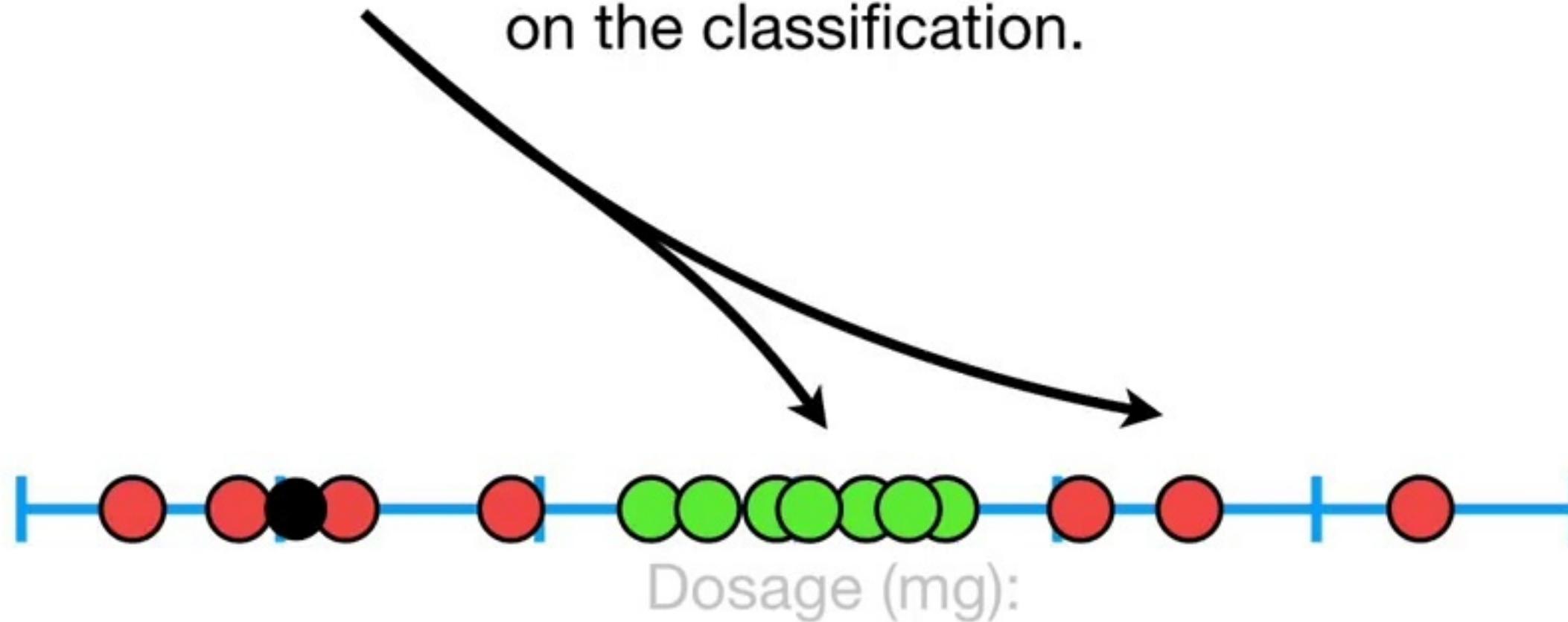
...the **Radial Kernel** behaves like a
Weighted Nearest Neighbor model.



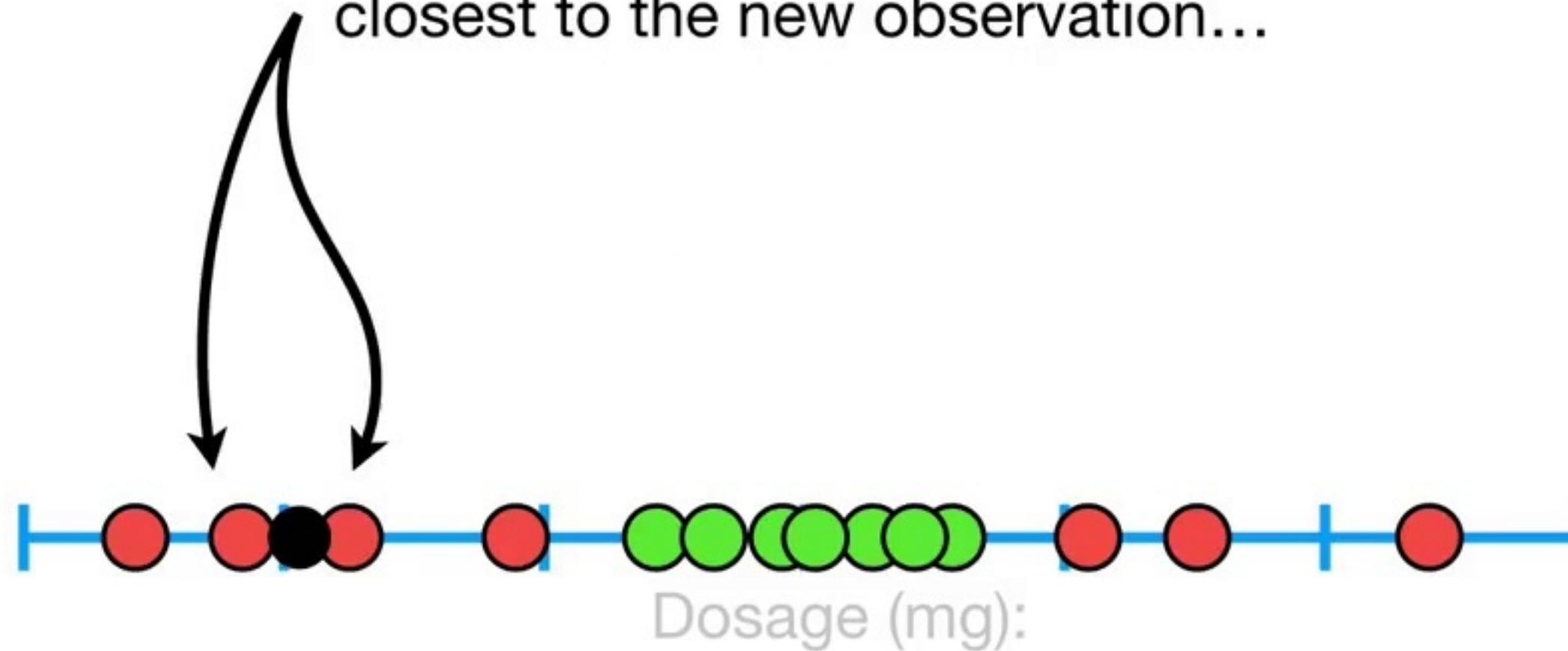
In other words, the closest observations (aka the nearest neighbors) have a lot of influence on how we classify the new observation...



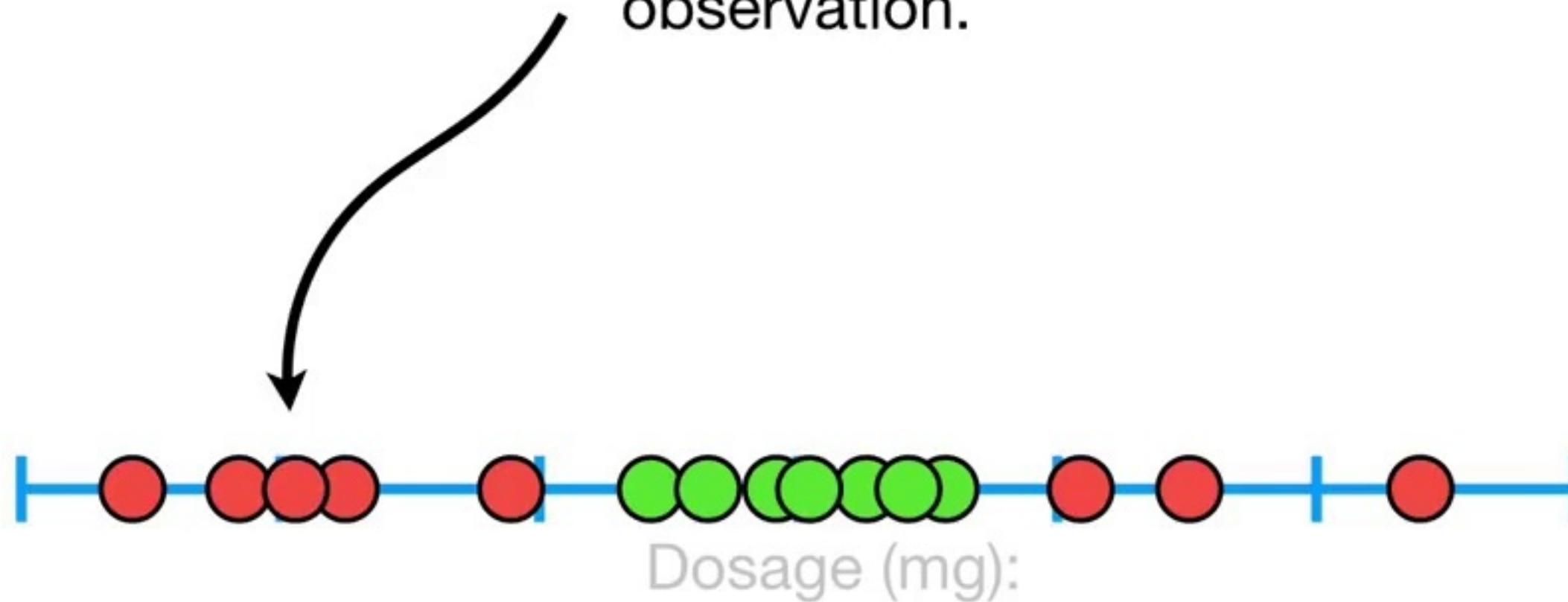
...and observations that are further away have relatively little influence on the classification.



So, since these observations are the closest to the new observation...



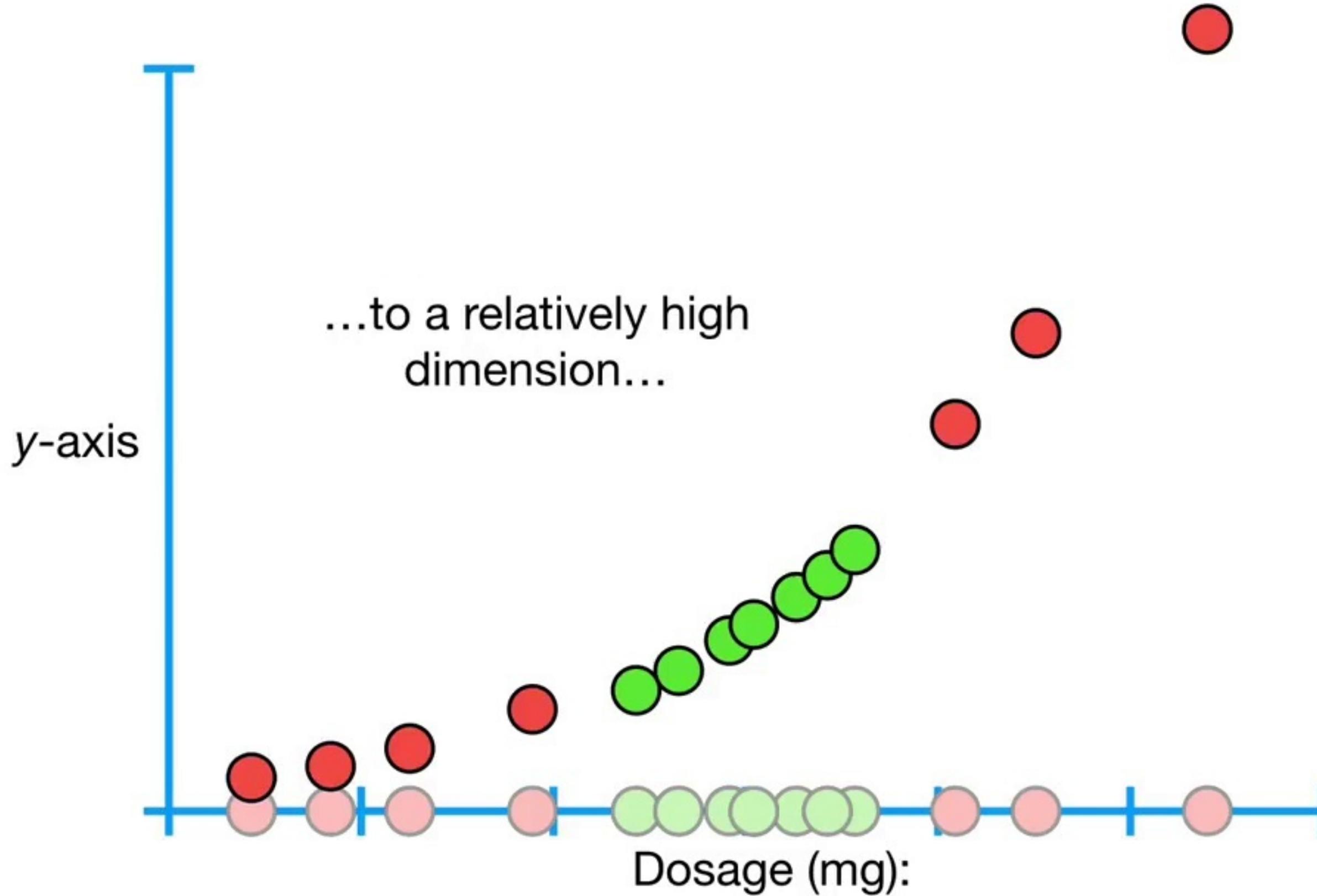
...the **Radial Kernel** uses their classification for the new observation.

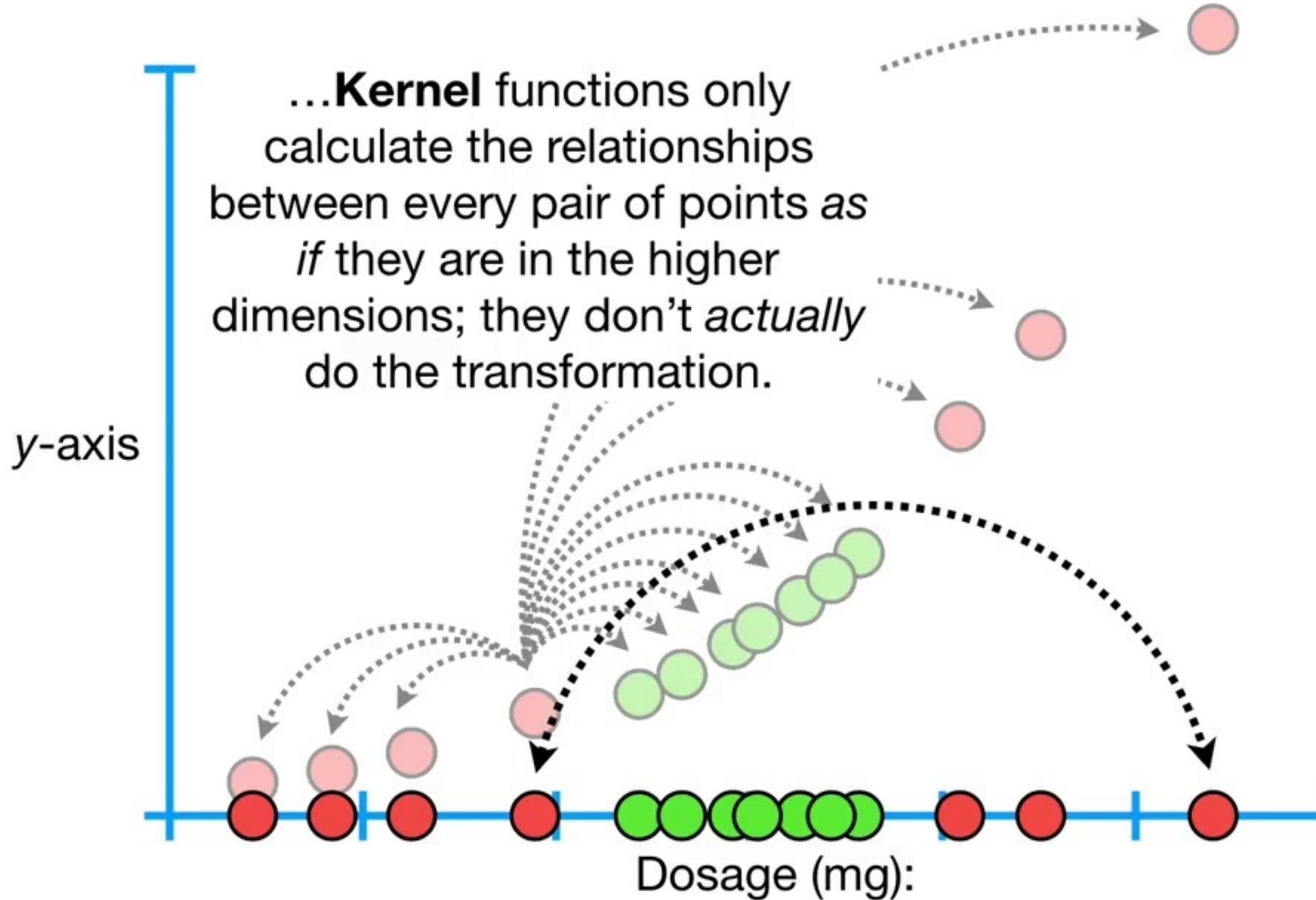


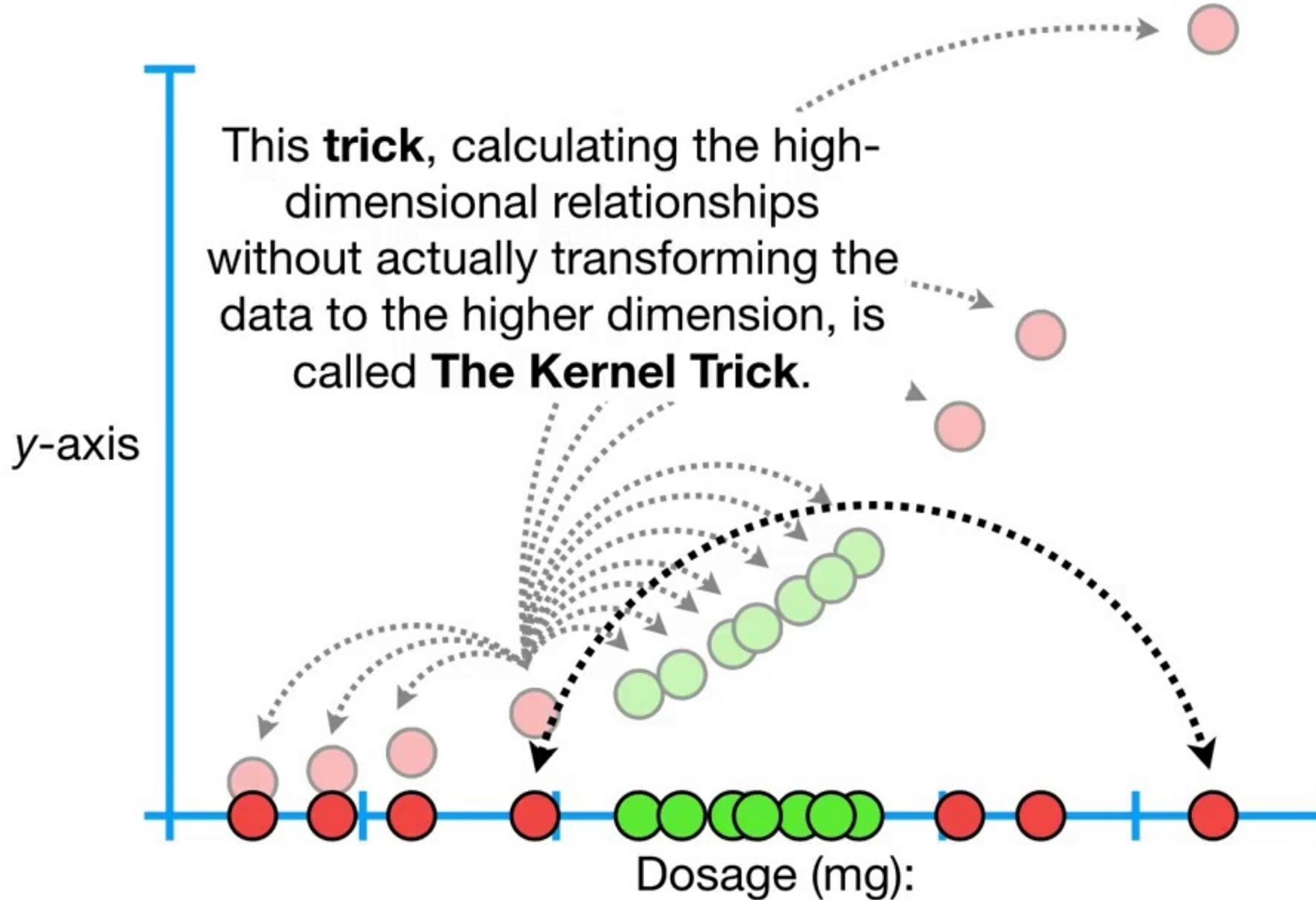
Now, for the sake of completeness, let
me mention one last detail about
Kernels.

Although the examples I have given show the data being transformed from a relatively low dimension...



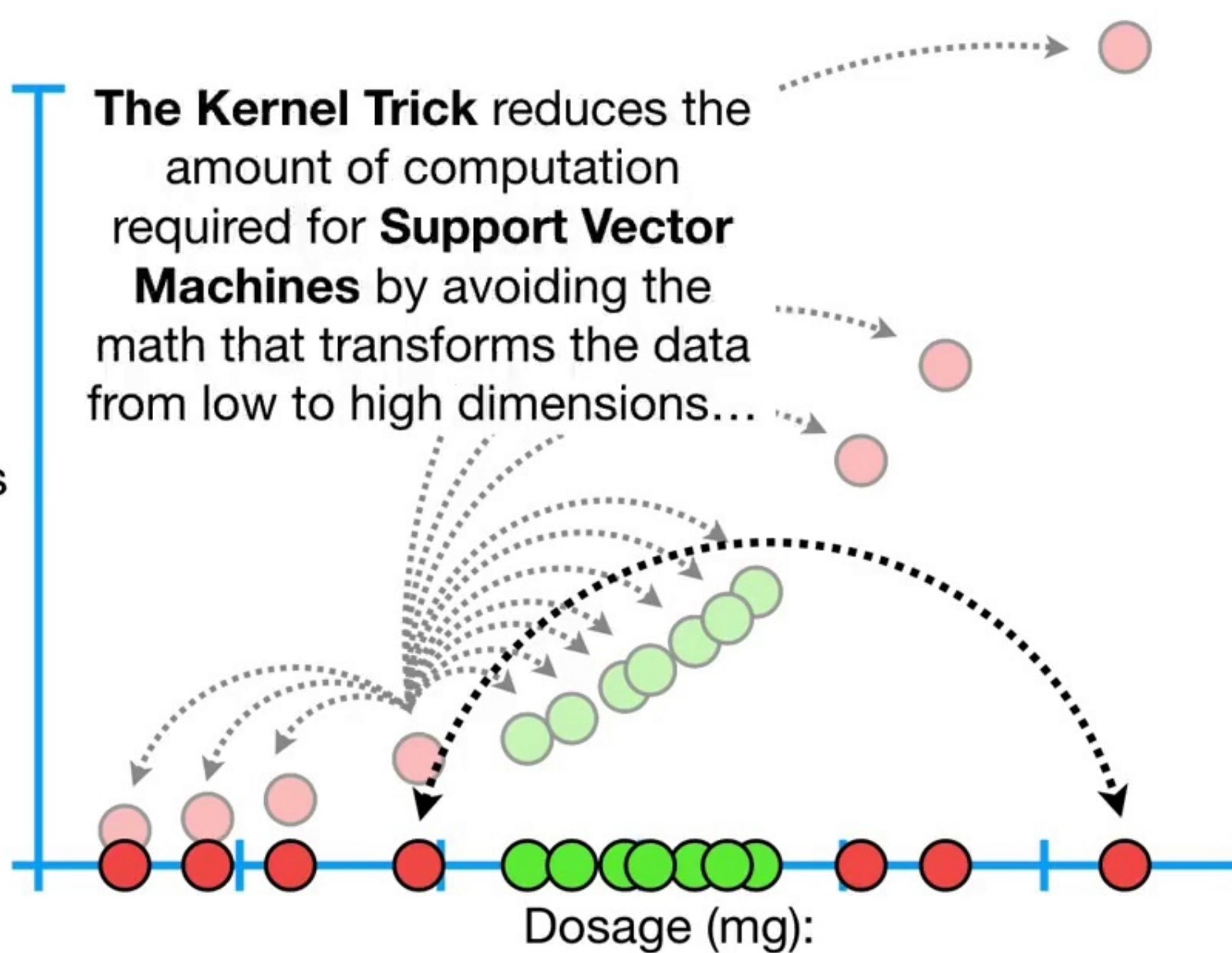


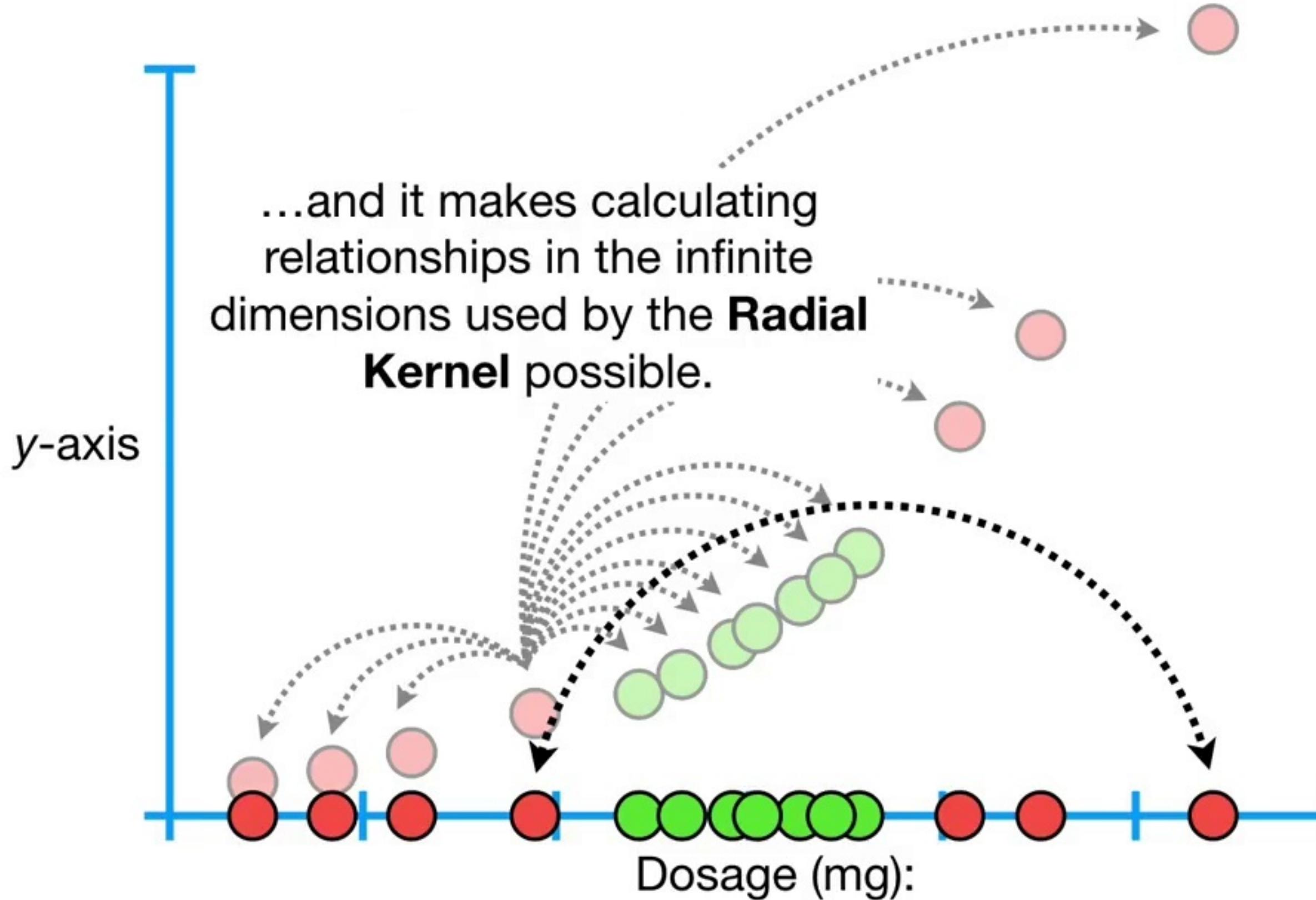




The Kernel Trick reduces the amount of computation required for **Support Vector Machines** by avoiding the math that transforms the data from low to high dimensions...

y-axis

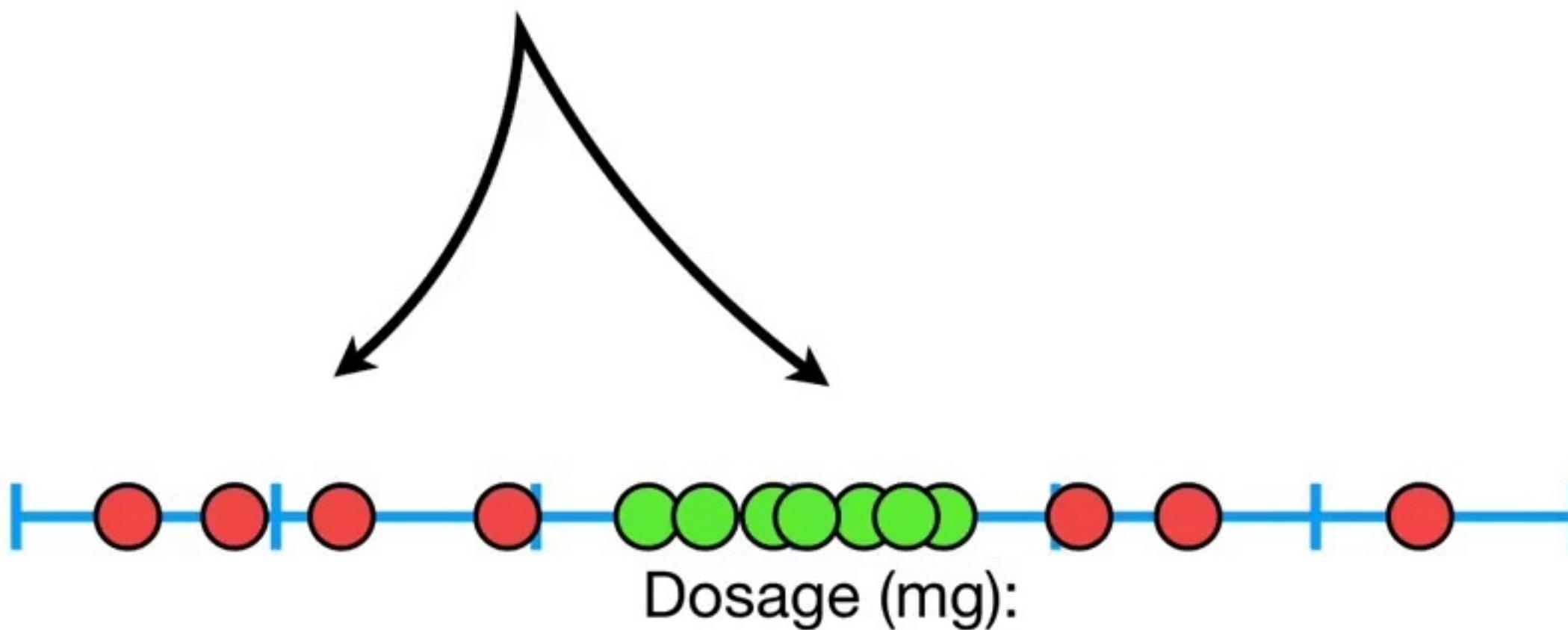


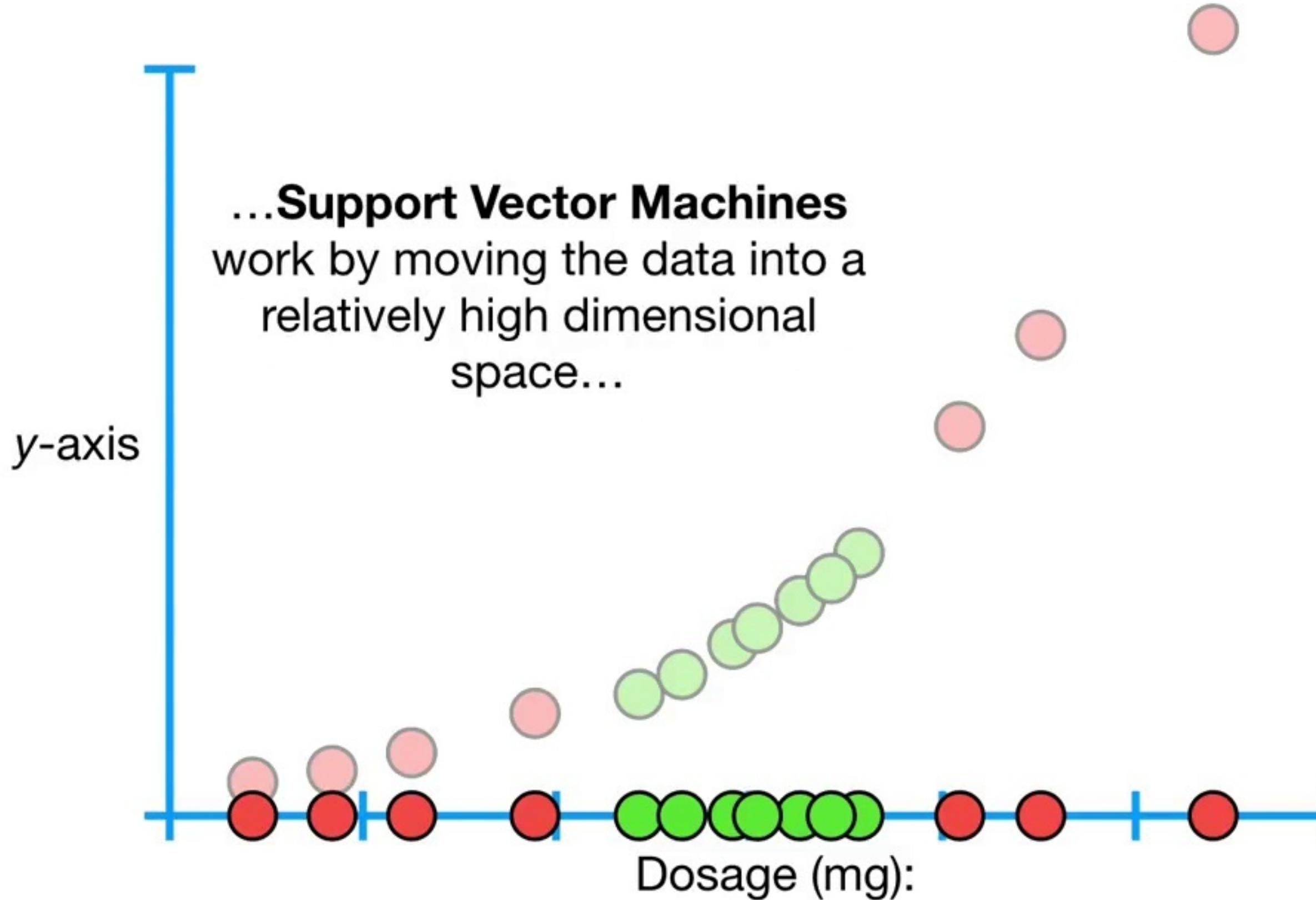


However, regardless of how the relationships are calculated, the concepts are the same.



When we have **2** categories, but
no obvious linear classifier that
separates them in a nice way...







...and finding a relatively high dimensional **Support Vector Classifier** that can effectively classify the observations.