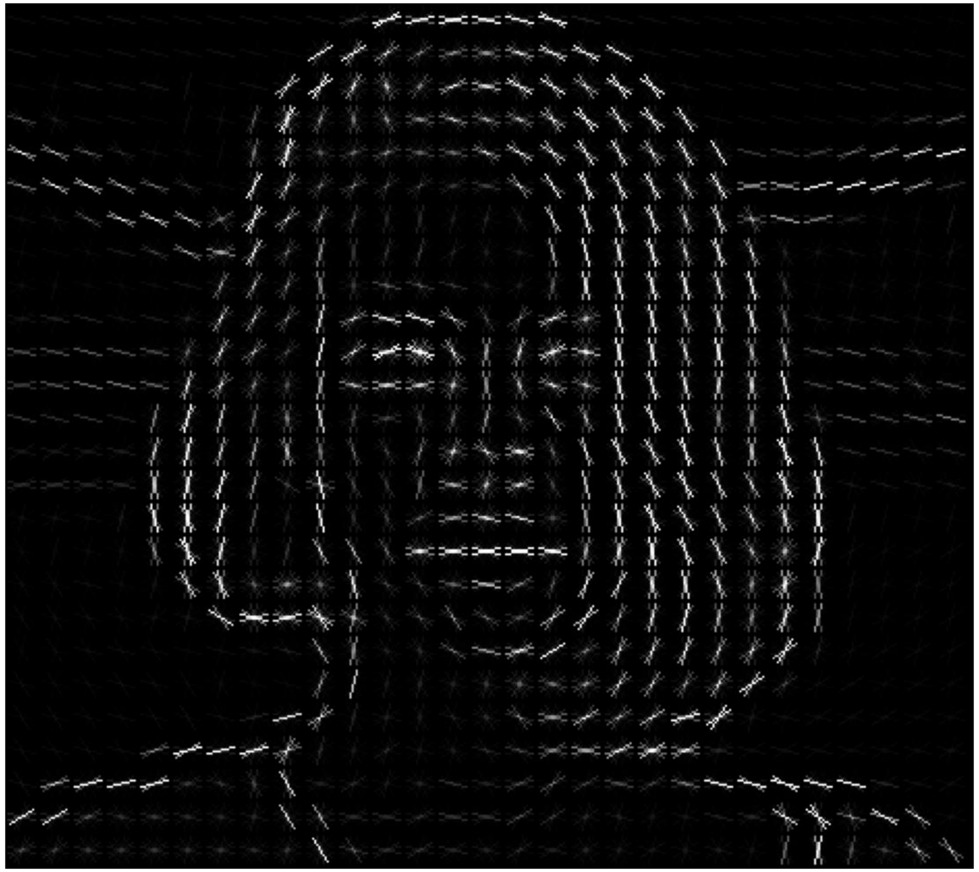
Face Recognition HOG + SVM

Bakytbek uulu Nurzhigit

HSE

06.12.2021

Introduction to the HOG Feature Descriptor

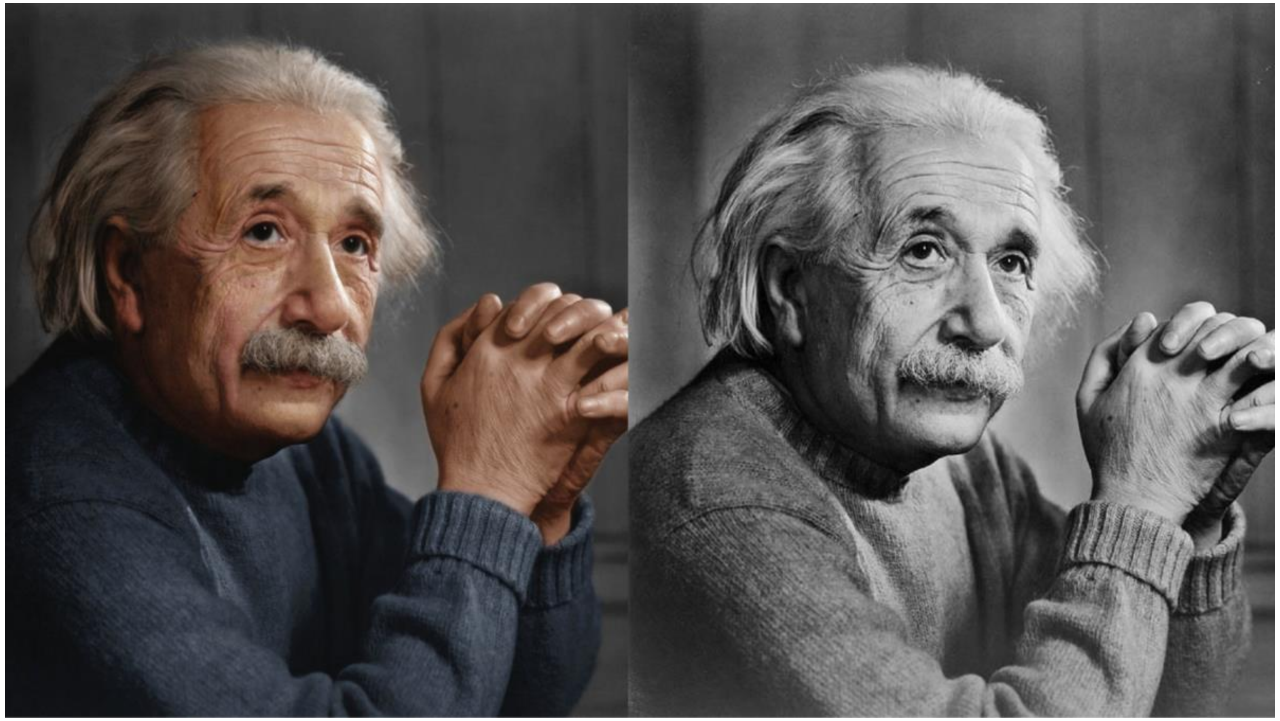


HOG is a feature descriptor that is often used to extract features from image data. It is widely used in computer vision tasks for object detection.

Step 1: Preprocess the Data

Choose convenient proportion of image

Colored to grayscale



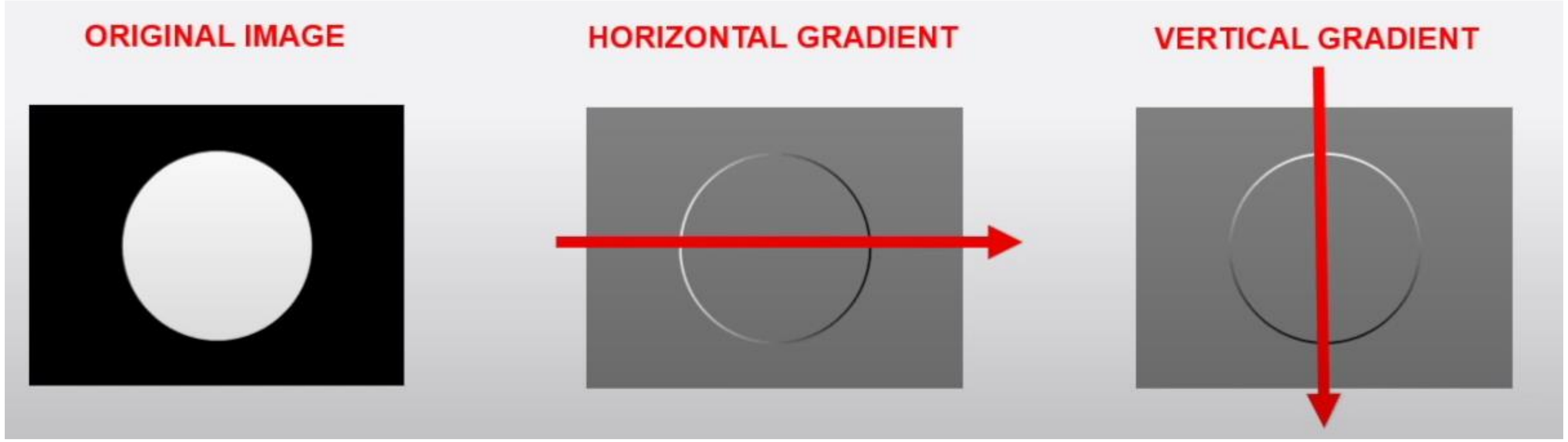
Step 2: Calculating Gradients (direction x and y)

-1	0	1

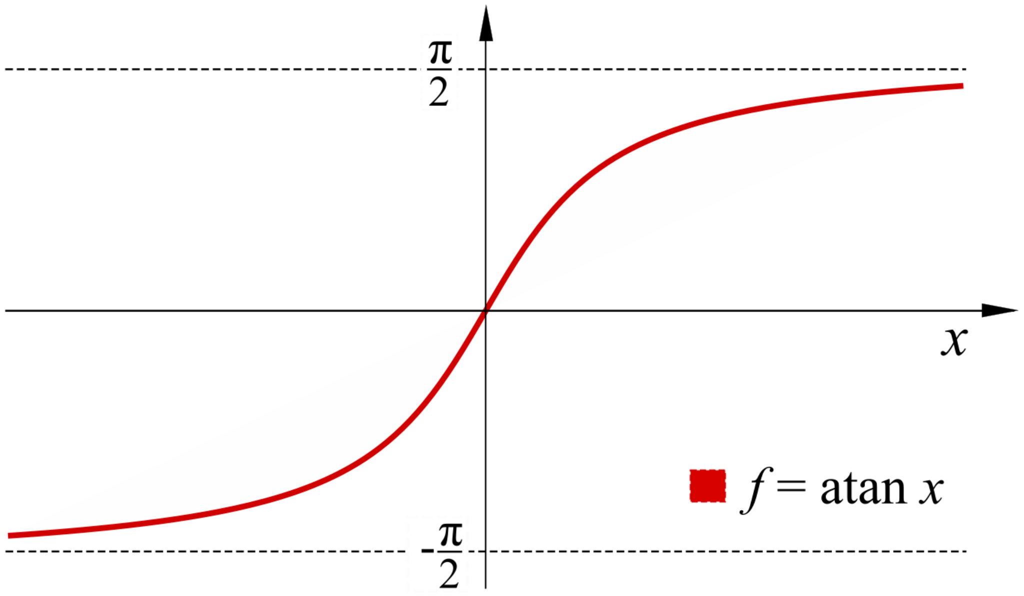
-1
0
1

- Change in X direction(G_x) = 89 – 78 = 11
- Change in Y direction(G_y) = 68 – 56 = 8

121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45



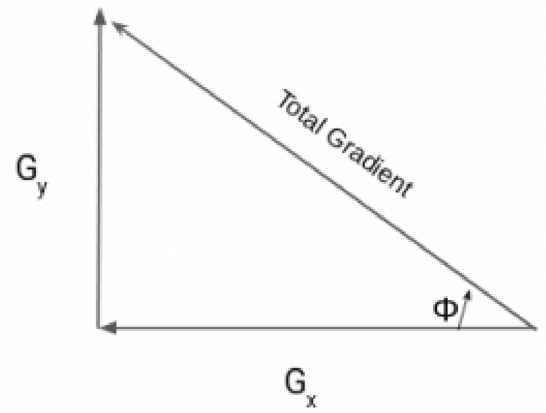
Step 3: Calculate the Magnitude and Orientation



Total Gradient Magnitude = $\sqrt{[(G_x)^2+(G_y)^2]}$

Total Gradient Magnitude = $\sqrt{[(11)^2+(8)^2]} = 13.6$

$\Phi = \text{atan}(Gy / Gx)$

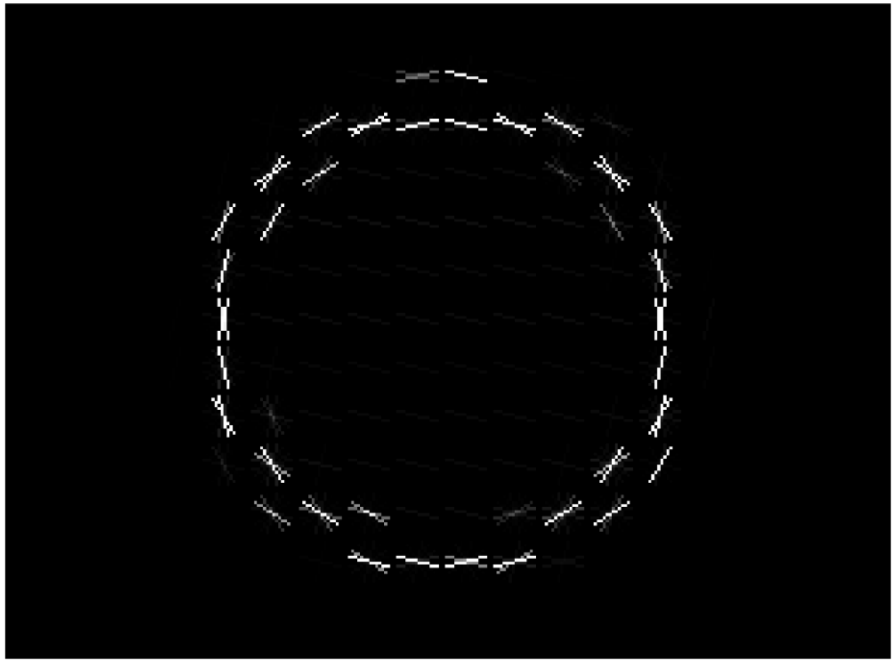


A histogram is a plot that shows the frequency distribution of a set of continuous data. (Another methods)

Step 4: Calculate Histogram of Gradients in 8×8 cells (9×1)

121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

Magnitude		1							
Bin	0	20	40	60	80	100	120	140	160



```python
import matplotlib.pyplot as plt

from skimage.feature import hog
from skimage import data, exposure


image = data.astronaut()

fd, hog_image = hog(image, orientations=8, pixels_per_cell=(16, 16),
                    cells_per_block=(1, 1), visualize=True, channel_axis=-1)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 4), sharex=True, sharey=True)

ax1.axis('off')
ax1.imshow(image, cmap=plt.cm.gray)
ax1.set_title('Input image')

# Rescale histogram for better display
hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))

ax2.axis('off')
ax2.imshow(hog_image_rescaled, cmap=plt.cm.gray)
ax2.set_title('Histogram of Oriented Gradients')
plt.show()
```

Input image

Histogram of Oriented Gradients

Classification metrics: confusion matrix

		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR–}}$	F_1 score = $\frac{1}{\frac{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}{2}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR–) $= \frac{\text{FNR}}{\text{TNR}}$		

Accuracy: what a problem?

		Predicted/Classified	
		Negative	Positive
Actual	Negative	998	0
	Positive	1	1

What the positive over here is actually someone who is sick and carrying a virus that can spread very quickly?

Recall

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative.

Precision

		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$= \frac{True\ Positive}{Total\ Predicted\ Positive}$$

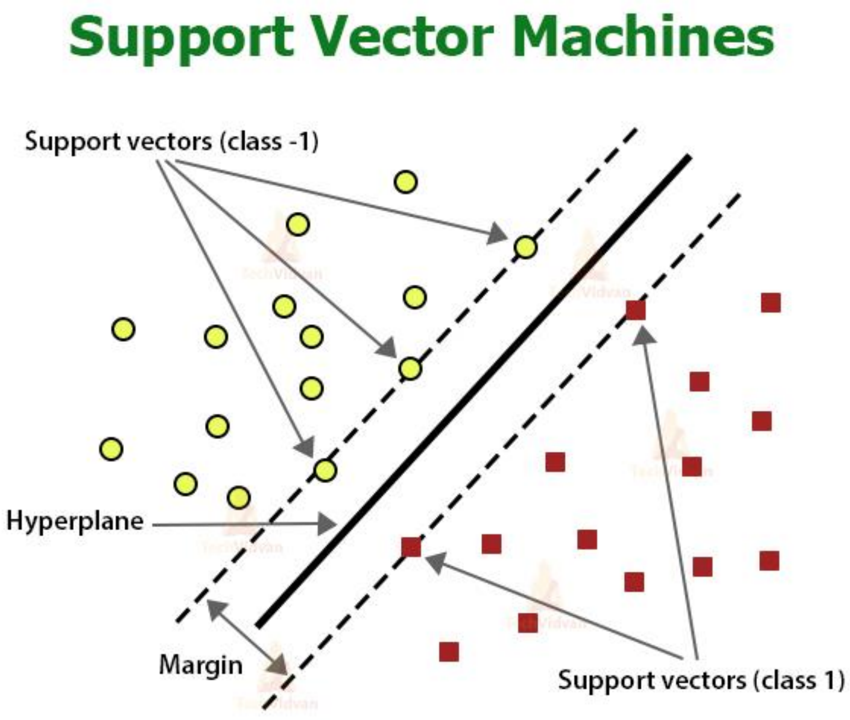
Precision is a good measure to determine, when the costs of False Positive is high. For instance, email spam detection.

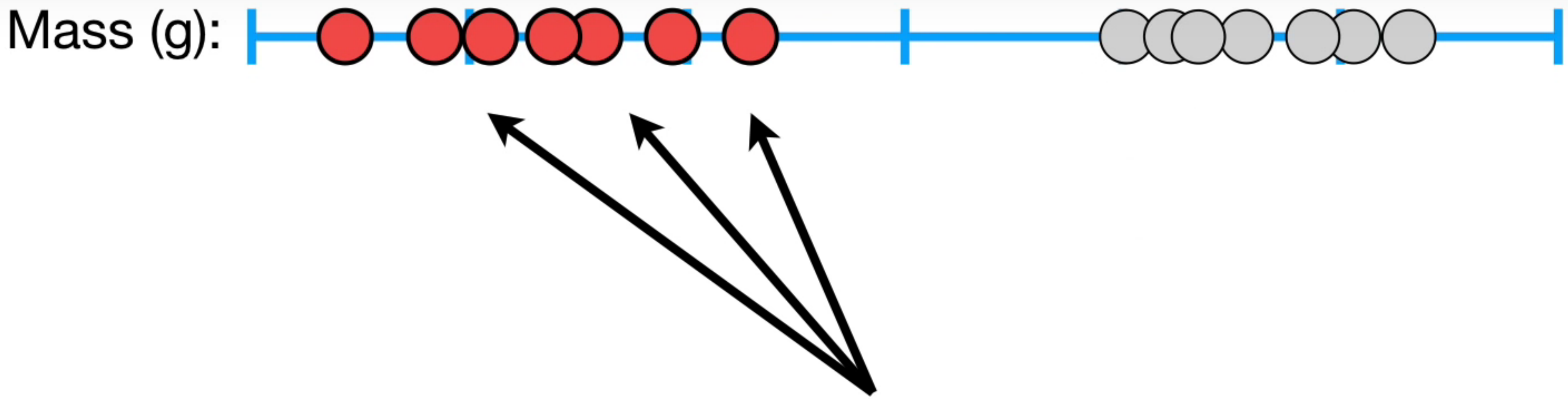
F1 Score

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

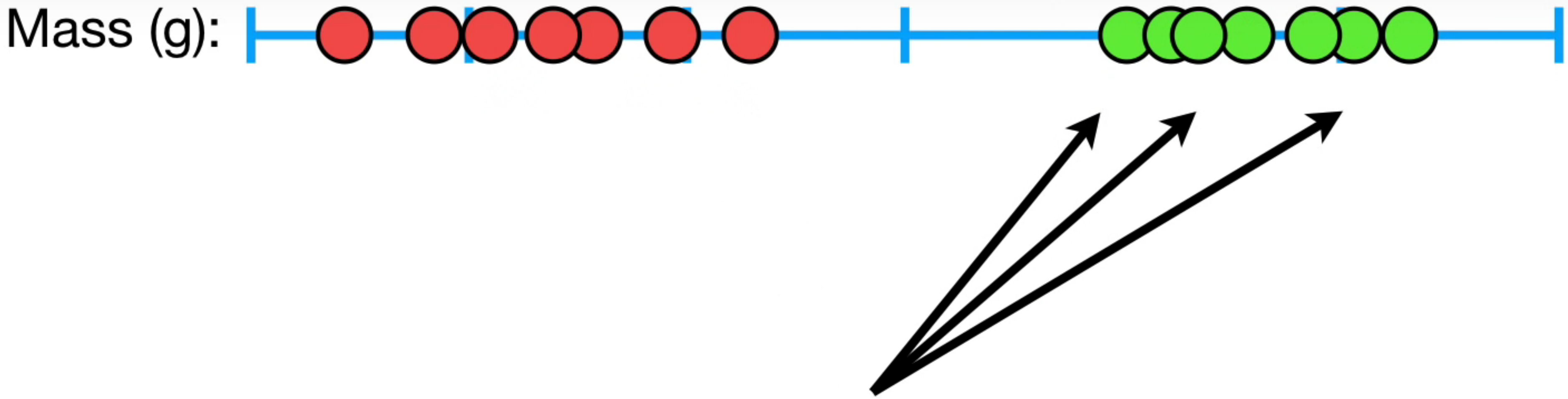
F1 Score is needed when you want to seek a balance between Precision and Recall.

Introduction to Support Vector Machine

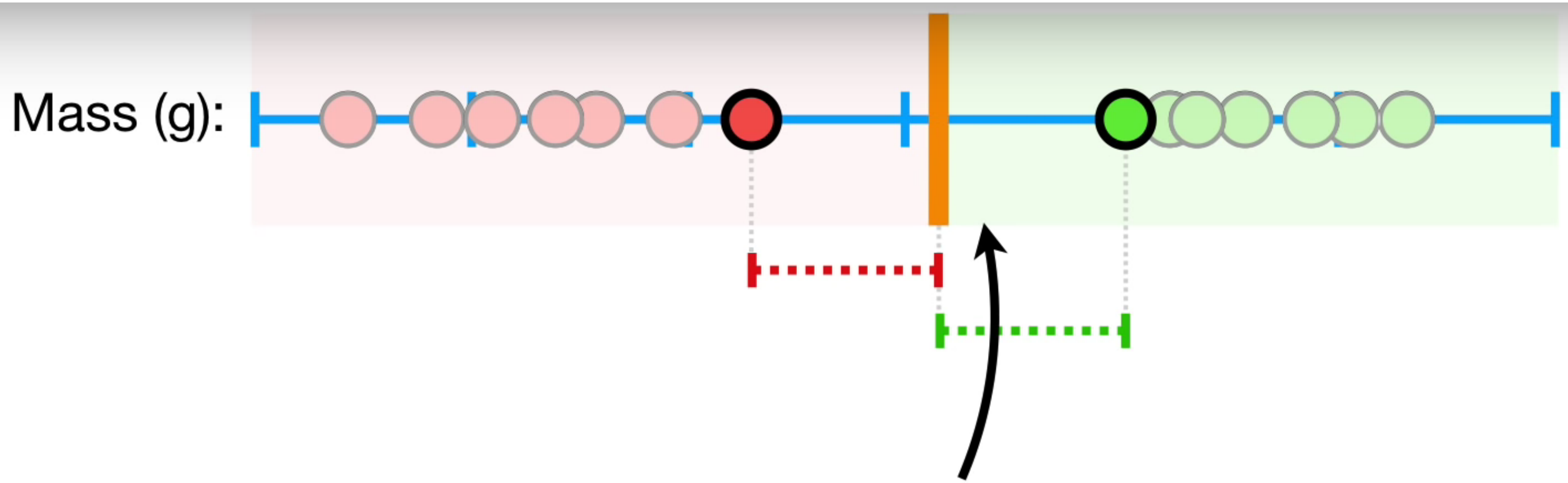




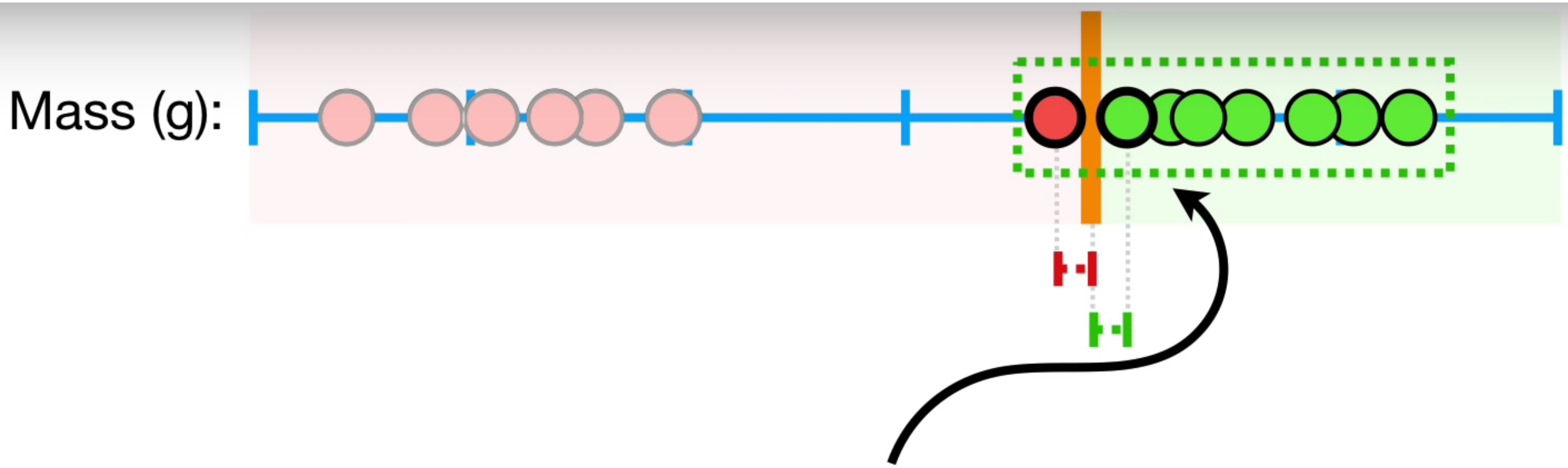
The **red dots** represent mice are ***not obese***…



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

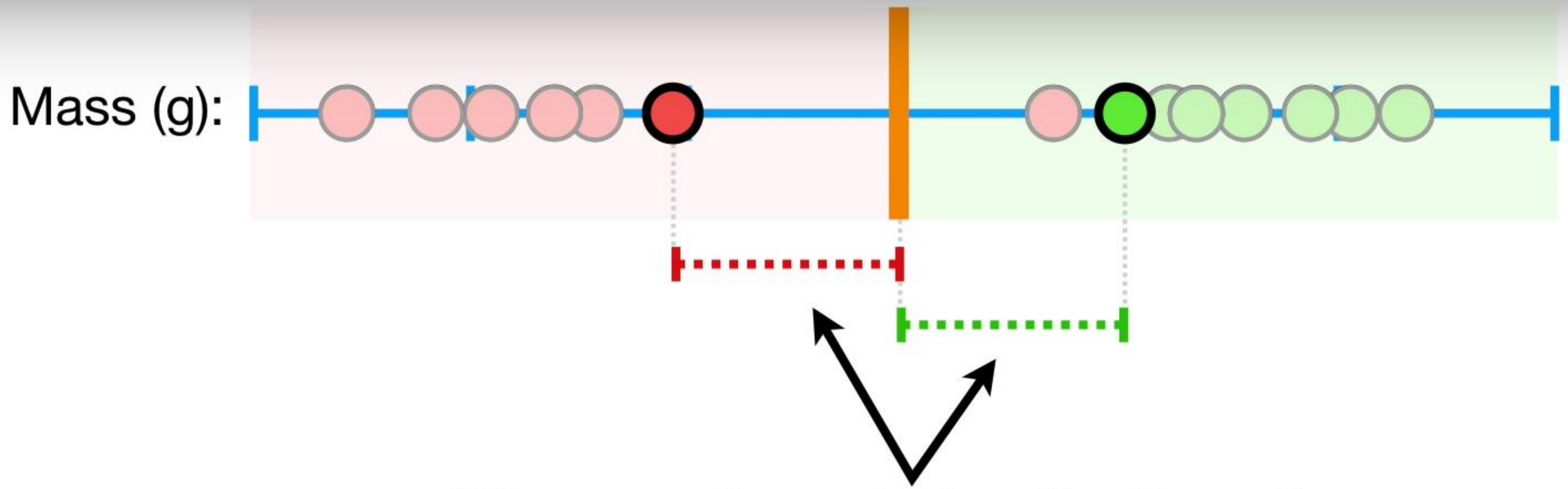


Maximal Margin Classifiers
seem pretty cool…

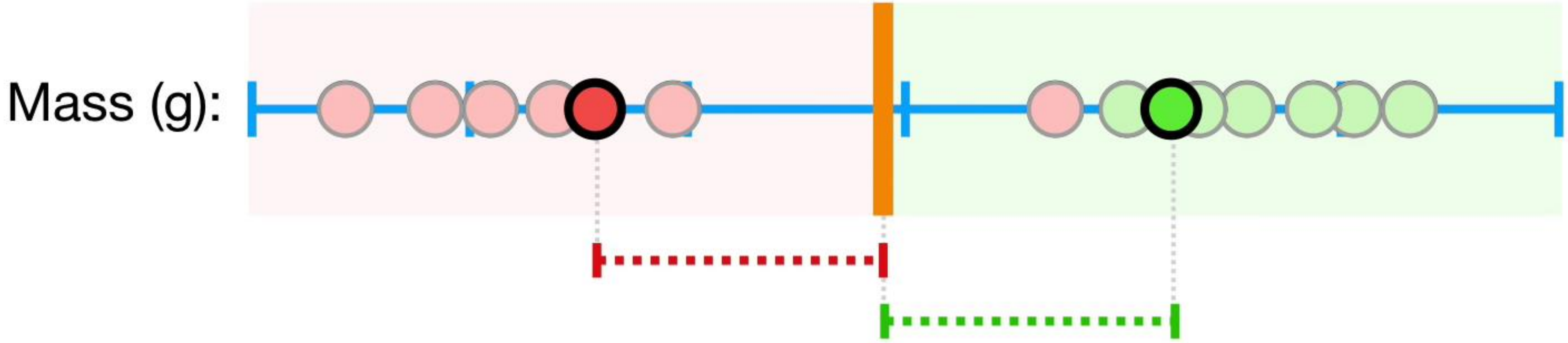


In this case, the **Maximum Margin Classifier** would be super close to the ***obese*** observations…

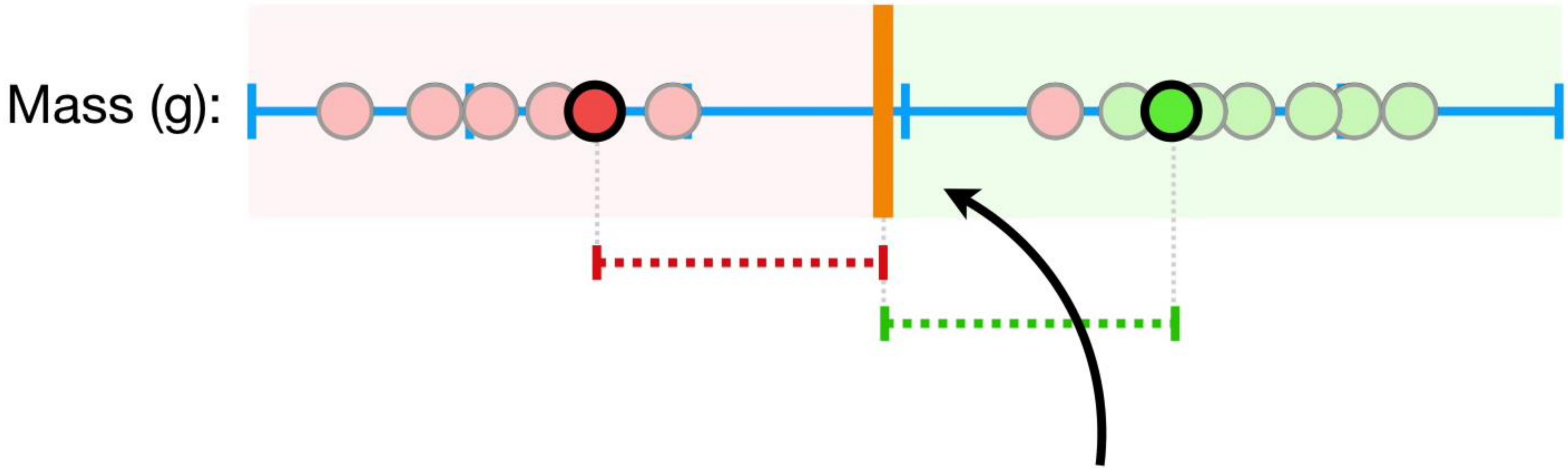
To make a threshold that is not so sensitive to outliers we must **allow misclassifications**.



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

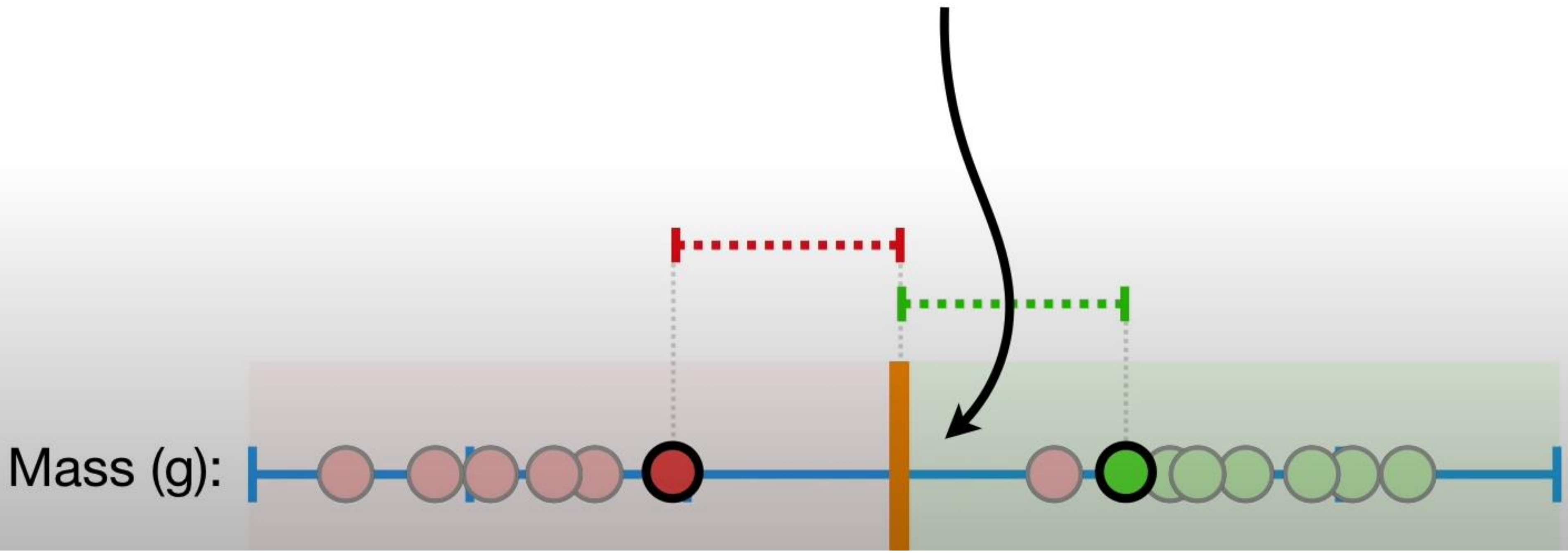


When we use a **Soft Margin** to determine the location of a threshold…

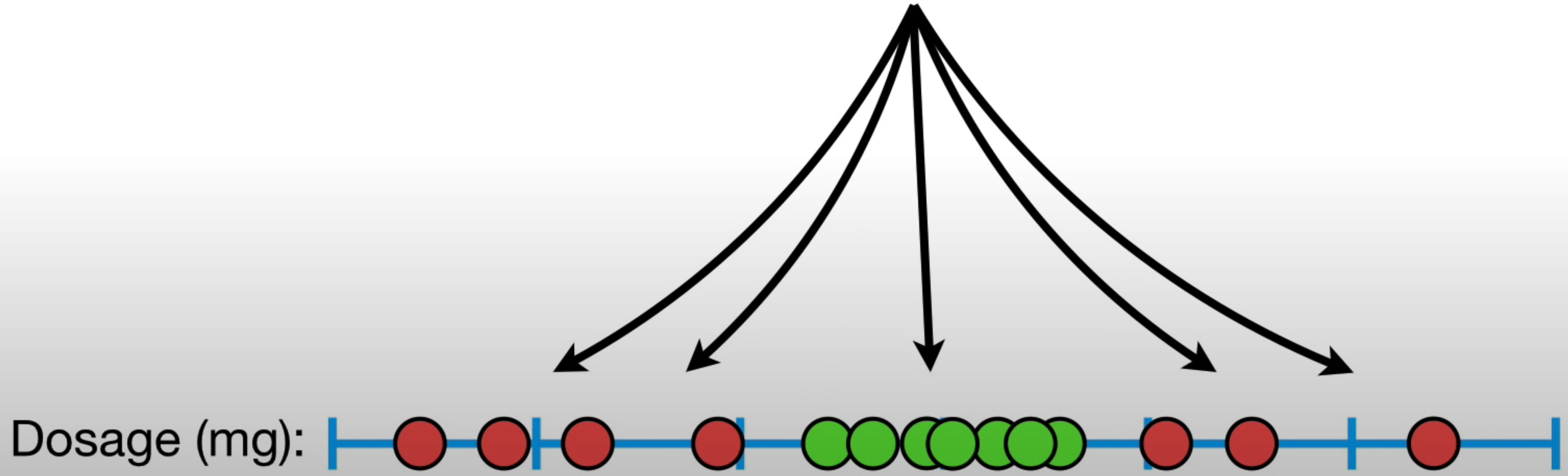


…then we are using a **Soft Margin Classifier** aka a **Support Vector Classifier** to classify observations.

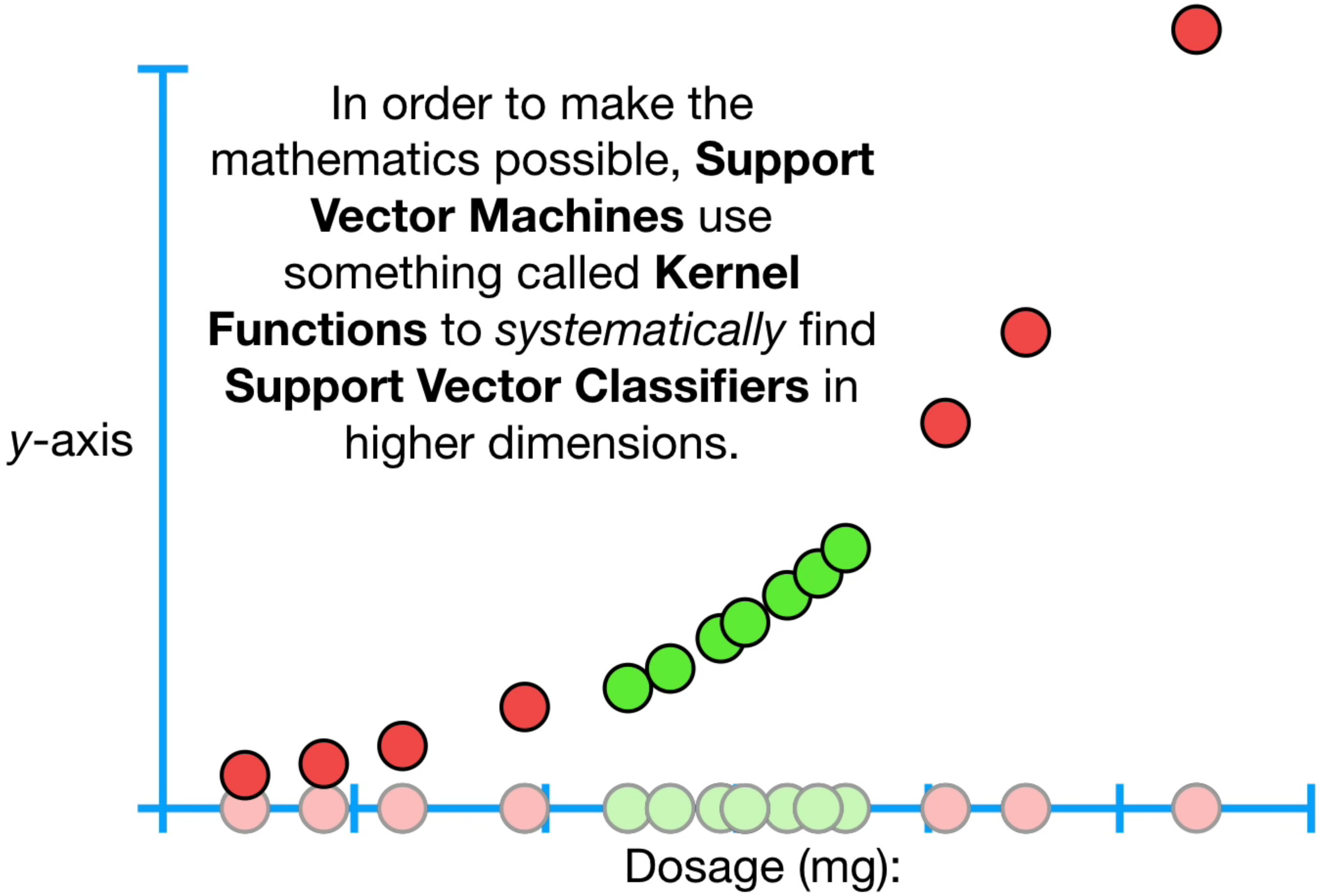
Support Vector Classifiers seem pretty cool because they can handle…

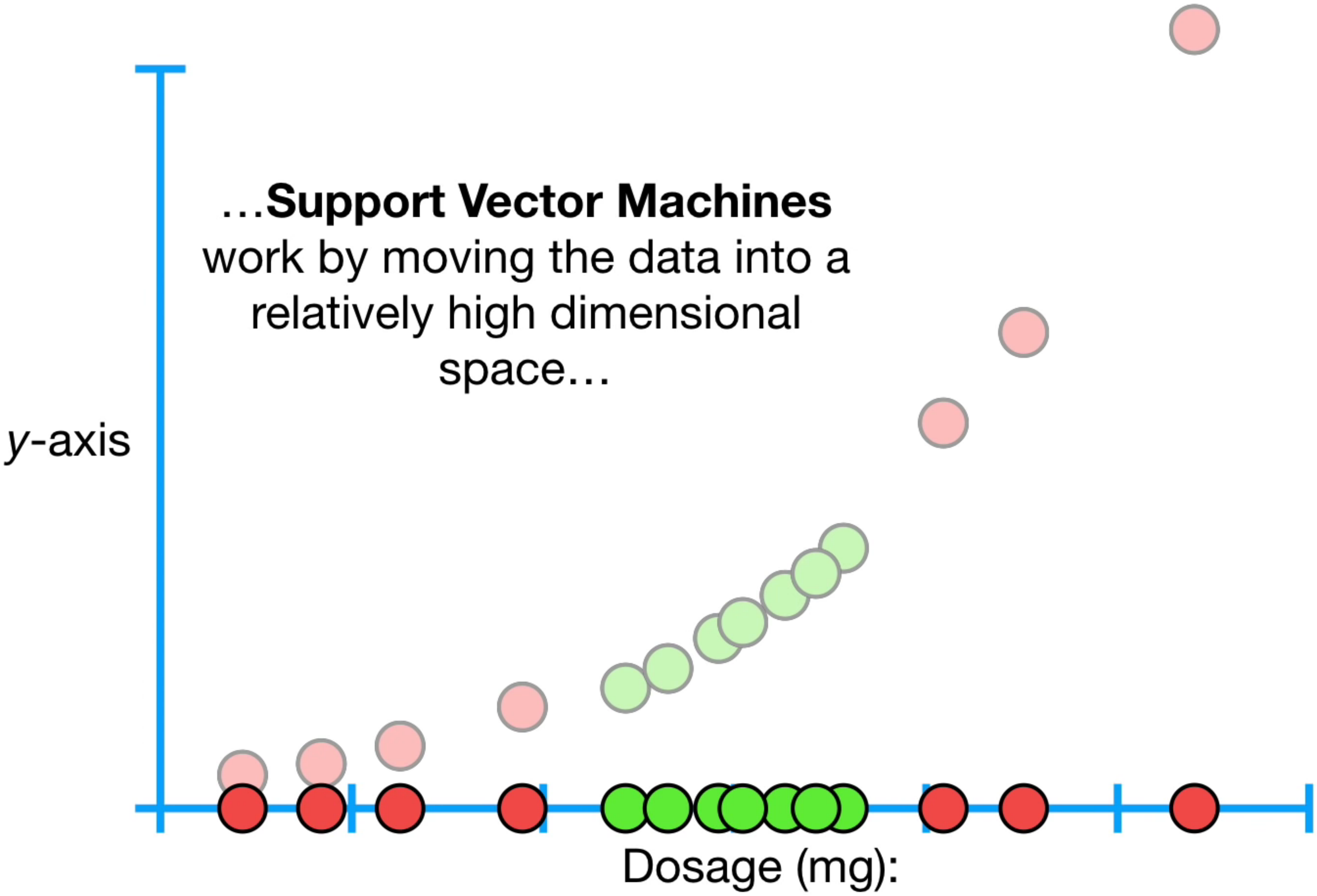


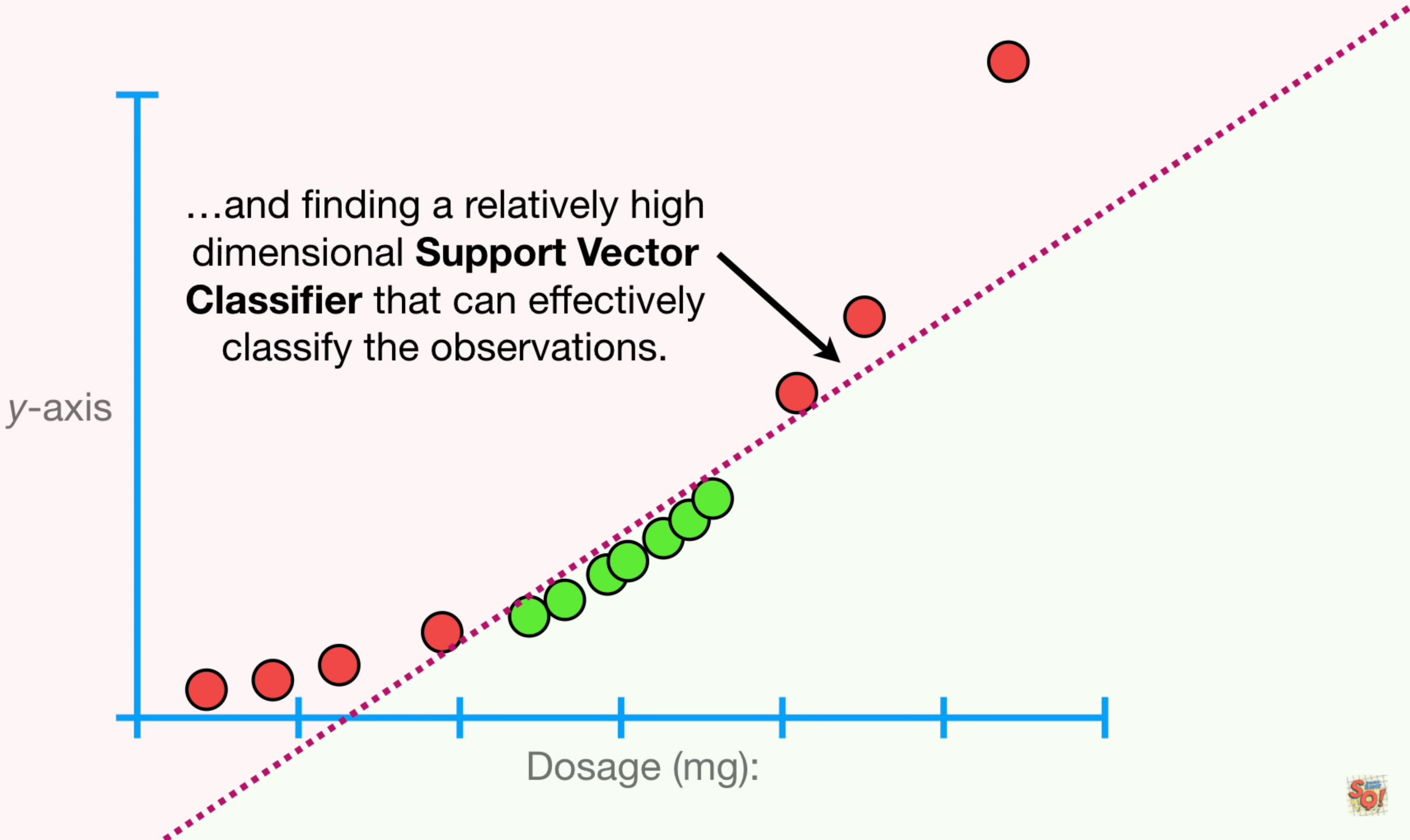
...but what if this was our training data and we had tons of overlap?



In order to make the mathematics possible, **Support Vector Machines** use something called **Kernel Functions** to *systematically* find **Support Vector Classifiers** in higher dimensions.




…**Support Vector Machines**
work by moving the data into a
relatively high dimensional
space…
y-axis
Dosage (mg):


…and finding a relatively high dimensional **Support Vector Classifier** that can effectively classify the observations.
y-axis
Dosage (mg):

Resources

1. https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/
2. https://habr.com/ru/post/306568/
3. https://www.youtube.com/watch?v=XmO0CSsKg88
4. https://scikit-learn.org/0.19/datasets/labeled_faces.html
5. https://towardsdatascience.com/building-a-face-recognition-system-using-scikit-learn-in-python-163fd423513b
6. https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html
7. https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/
8. https://scikit-learn.org/0.19/auto_examples/applications/plot_face_recognition.html#sphx-glr-auto-examples-applications-plot-face-recognition-py
9. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9
10. https://www.youtube.com/watch?v=efR1C6CvhmE
11. https://colab.research.google.com/drive/1PRjzNOGKCANGM-TYMKG6W7ohd7-07pZD?usp=sharing