Statistics and Machine Learning

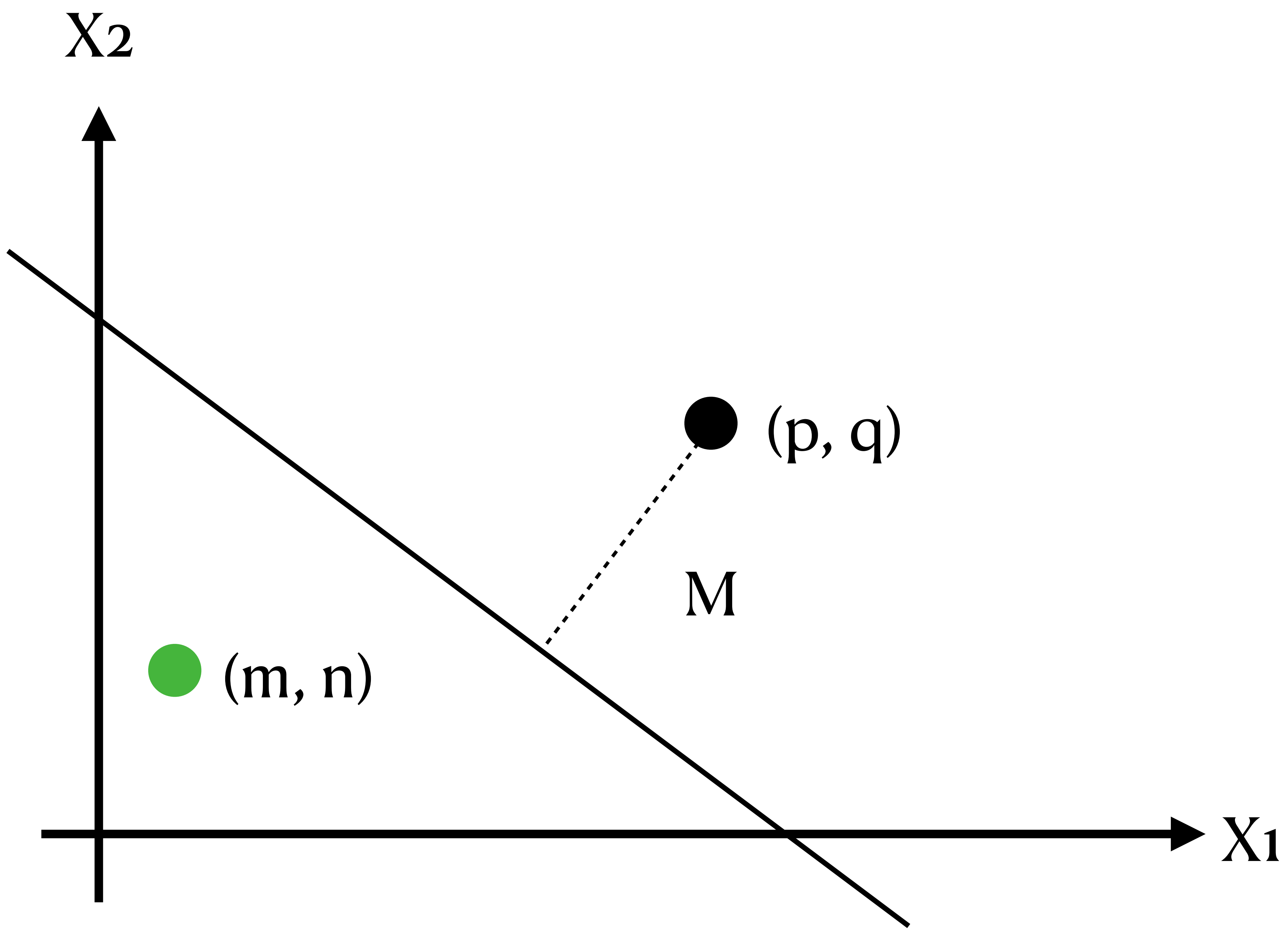
Support Vector Machine (SVM)

Week 13 04/12 — 04/16

Contents

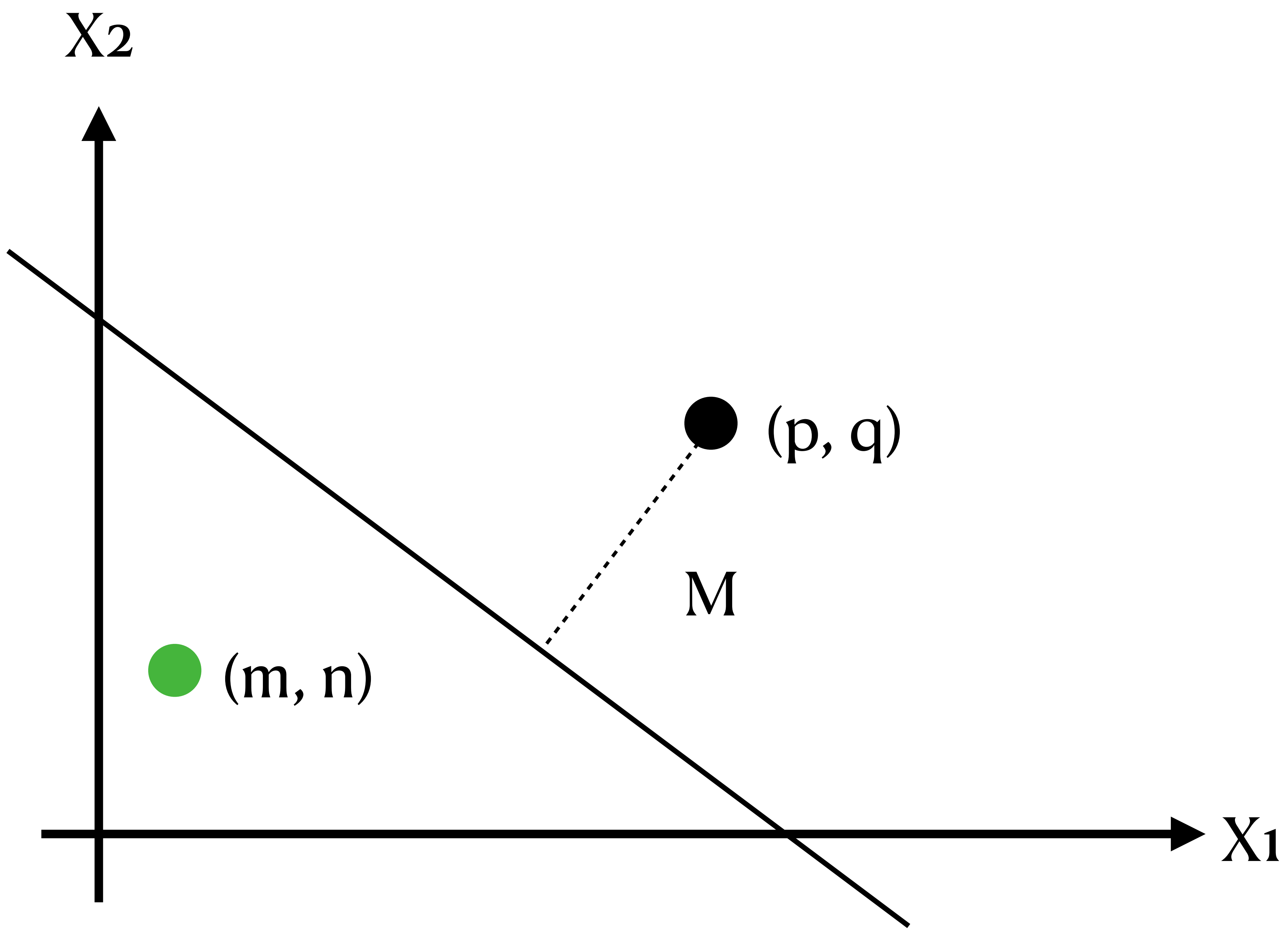
- Math facts
- Support vector classifier
- Support vector machine
- Nonlinear decision boundary

Math fact 1: linear case



$$aX_1 + bX_2 + c = 0$$

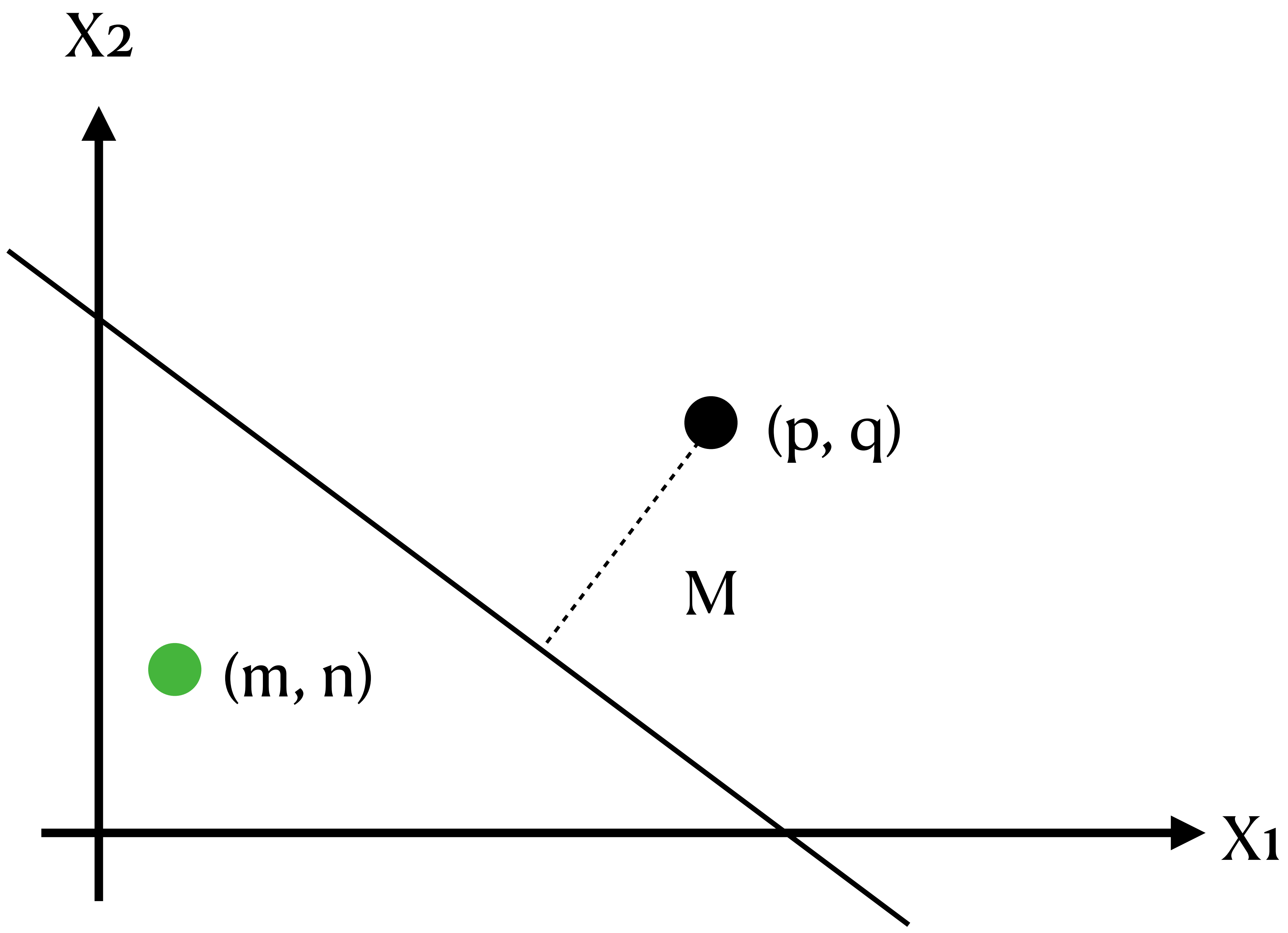
Math fact 1: linear case



$$aX_1 + bX_2 + c = 0$$

Assume a >0 without loss of generality, then:

Math fact 1: linear case

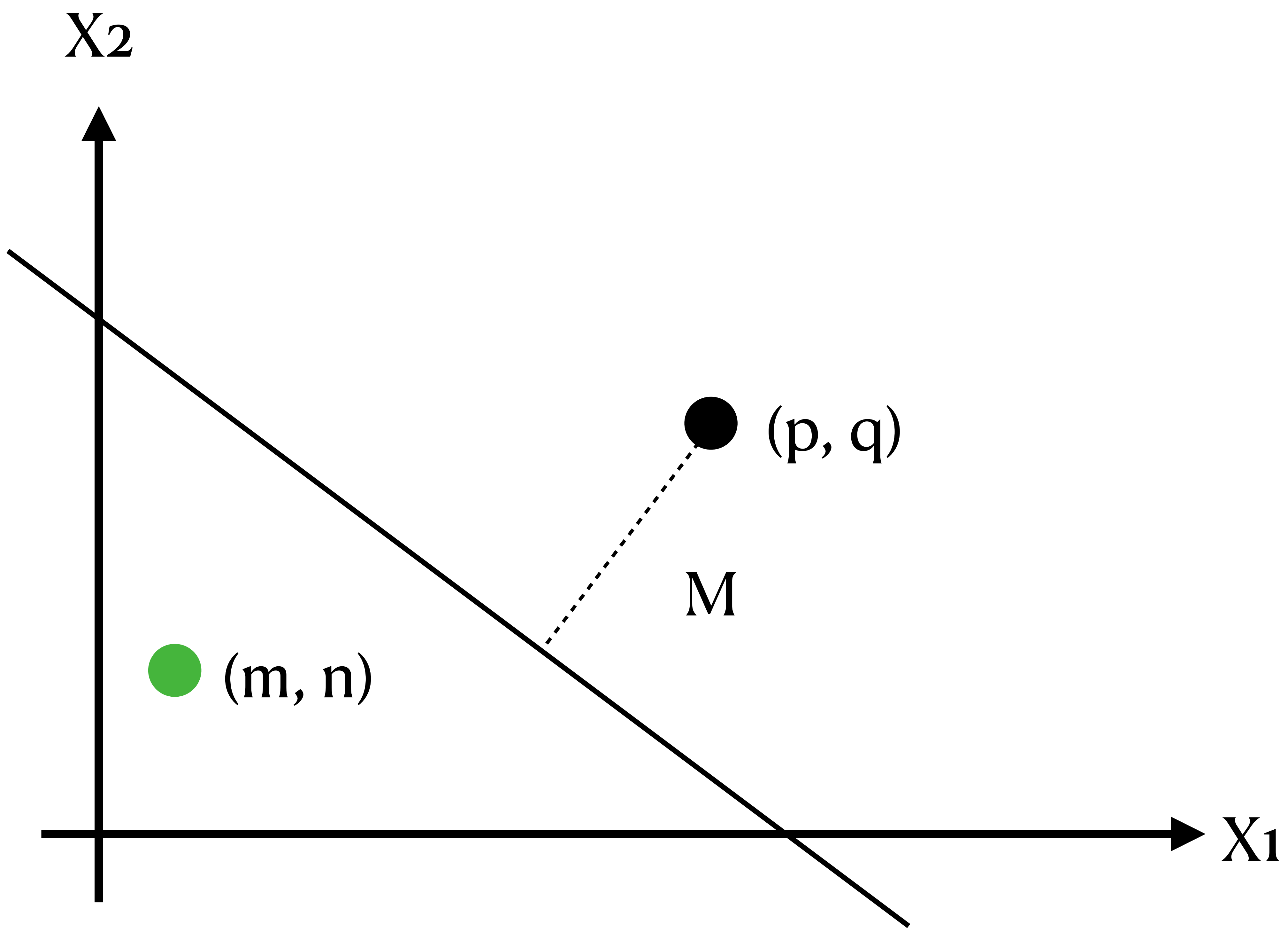


$$aX_1 + bX_2 + c = 0$$

Assume a >0 without loss of generality, then:

1. If the point (p,q) is on the right side of the line, then $ap + bq + c > 0.$

Math fact 1: linear case



$$aX_1 + bX_2 + c = 0$$

Assume a >0 without loss of generality, then:

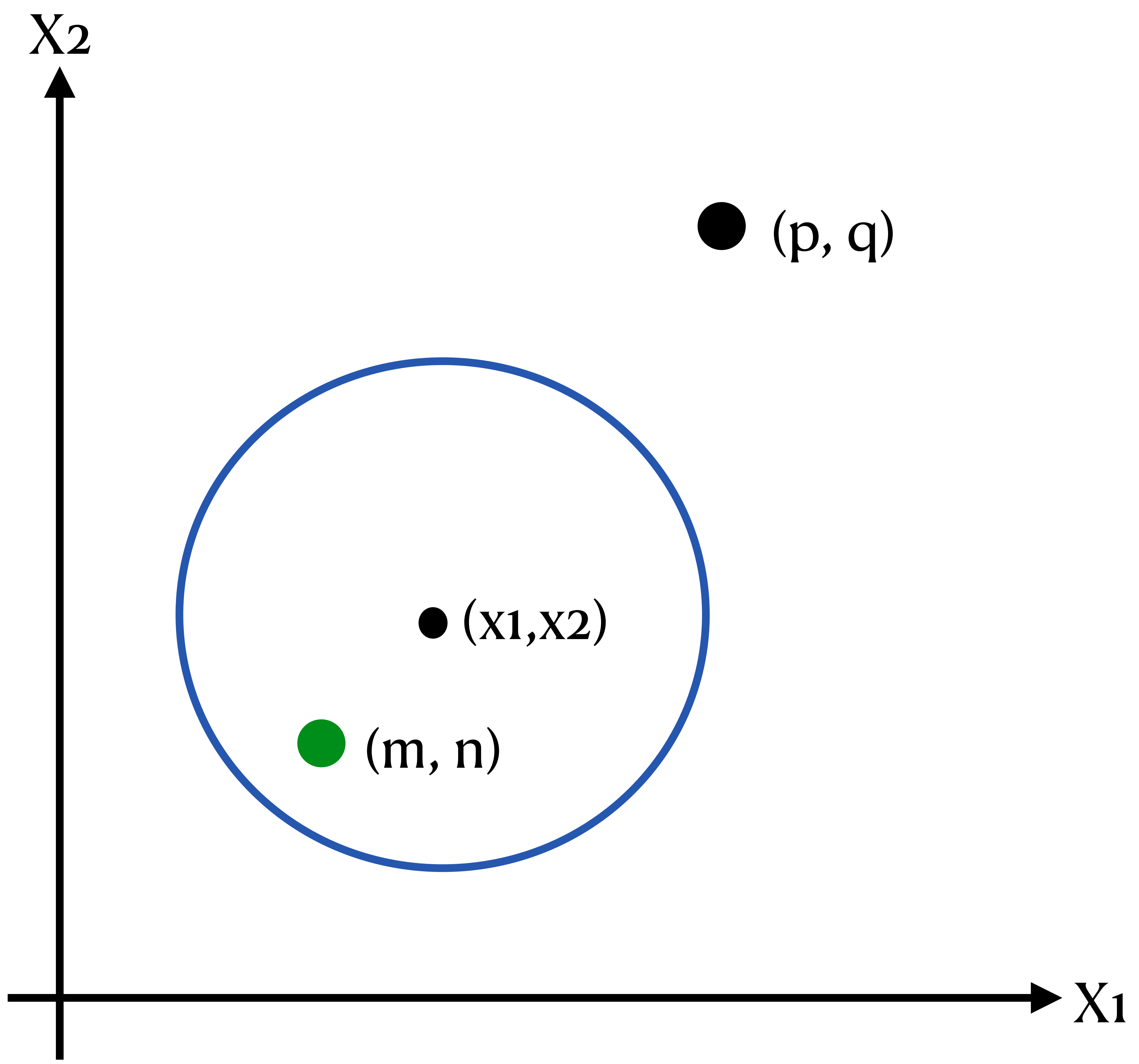
1. If the point (p,q) is on the right side of the line, then

$$ap + bq + c > 0.$$

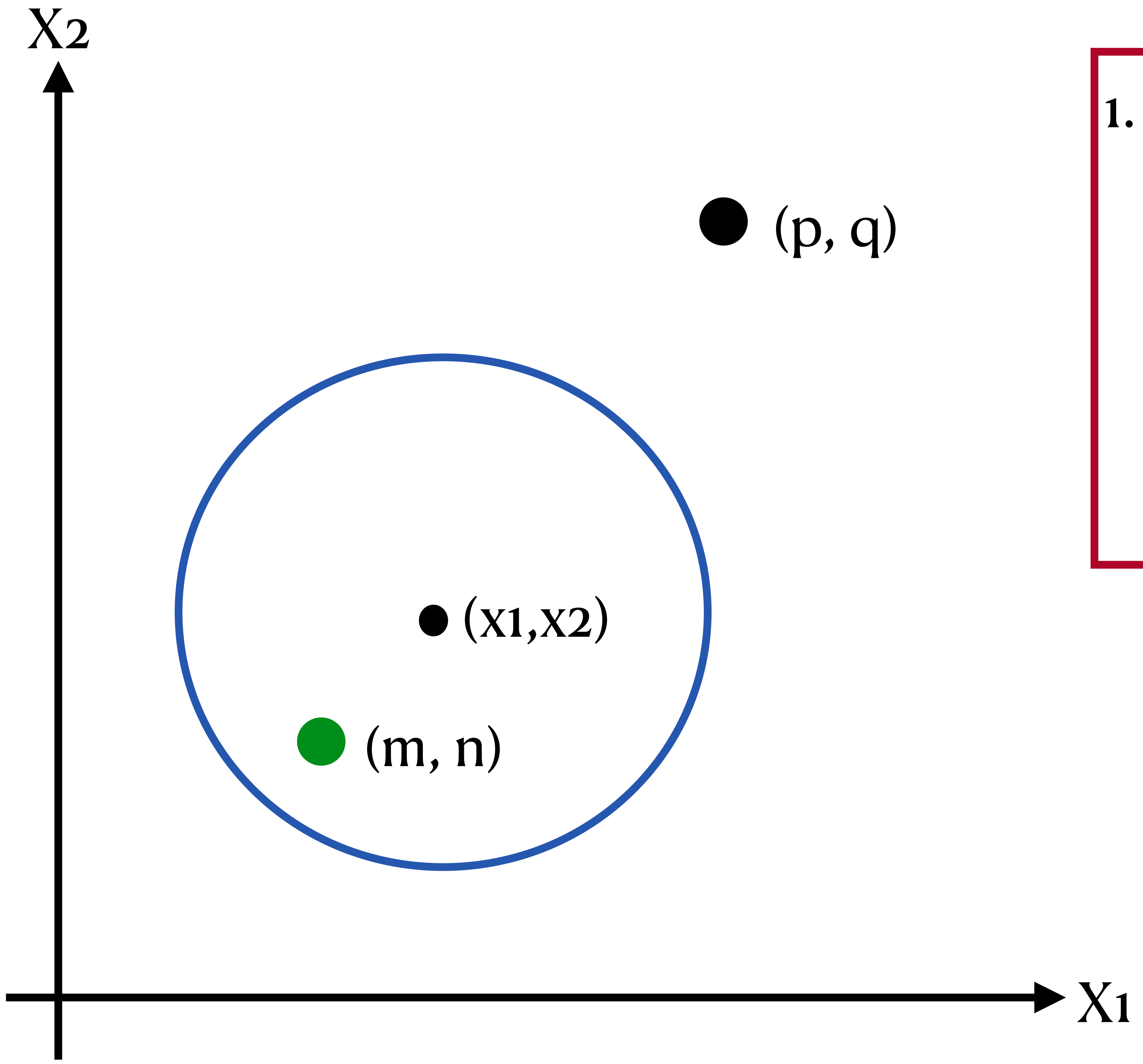
2. The margin between the point and the line is

$$M = \frac{|ap + bq + c|}{\sqrt{a^2 + b^2}}$$

Math fact 2: a nonlinear case



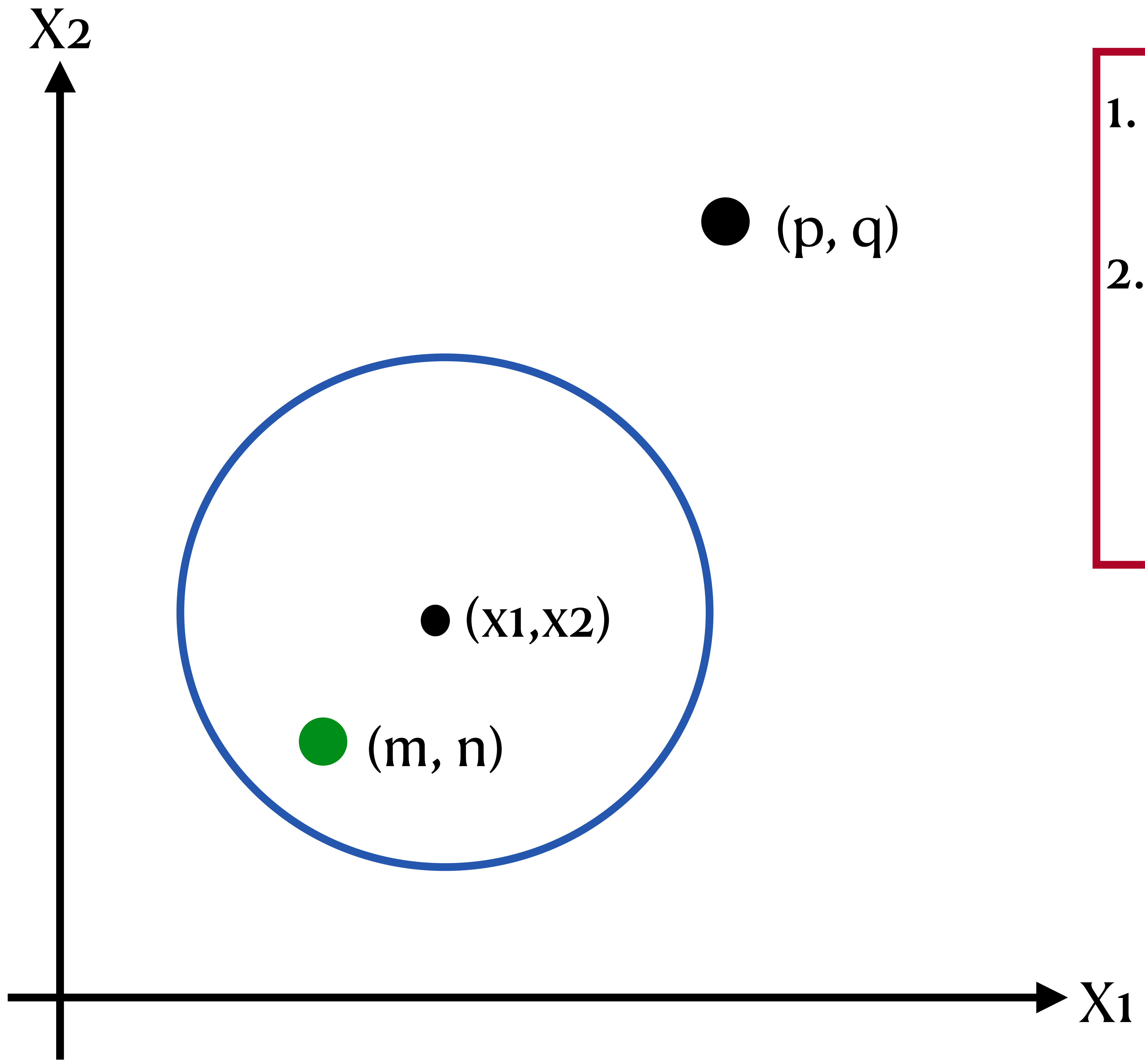
Math fact 2: a nonlinear case



1. The circle correspond to the equation:

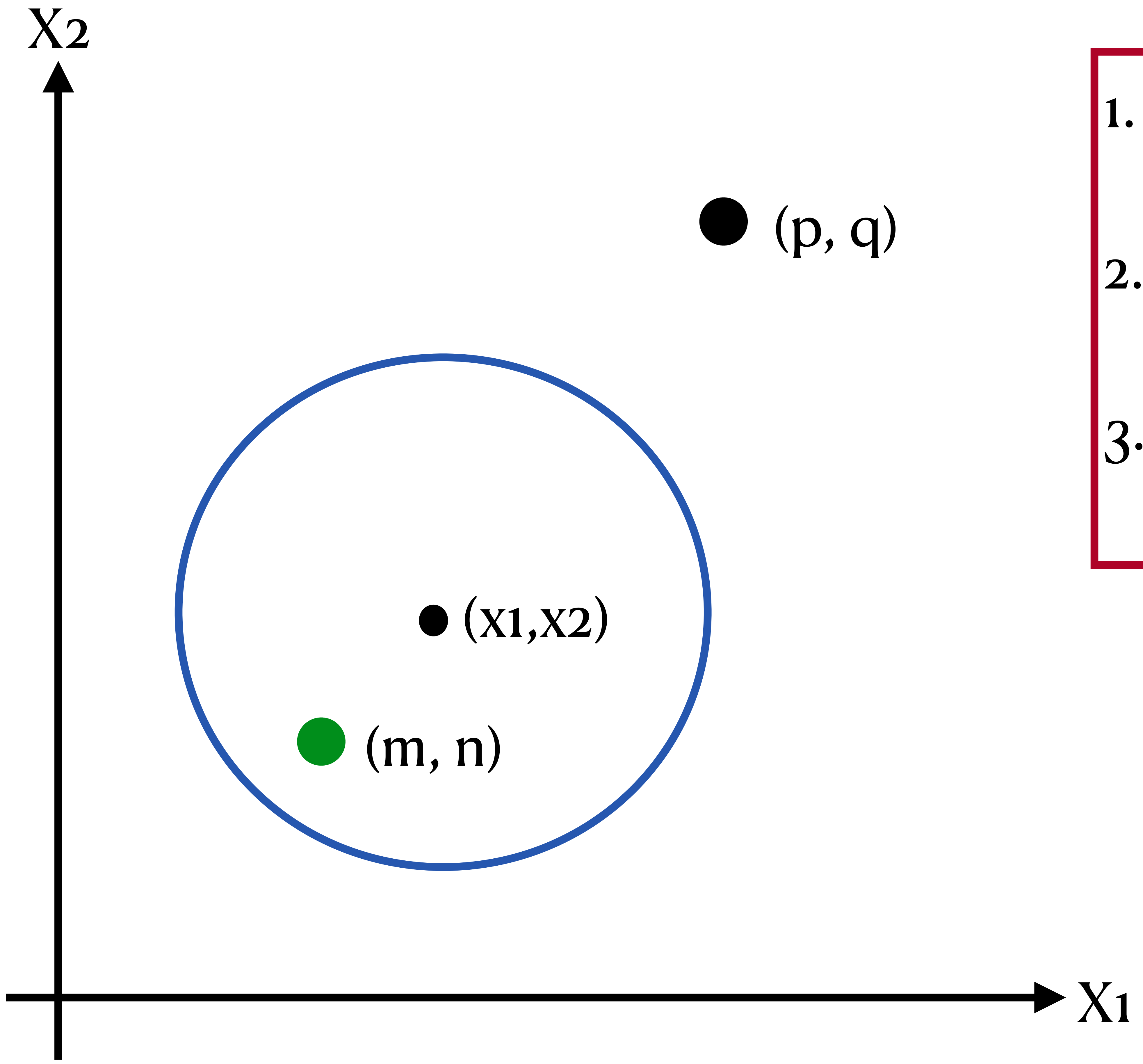
$$(X_1 - x_1)^2 + (X_2 - x_2)^2 - r^2 = 0$$

Math fact 2: a nonlinear case



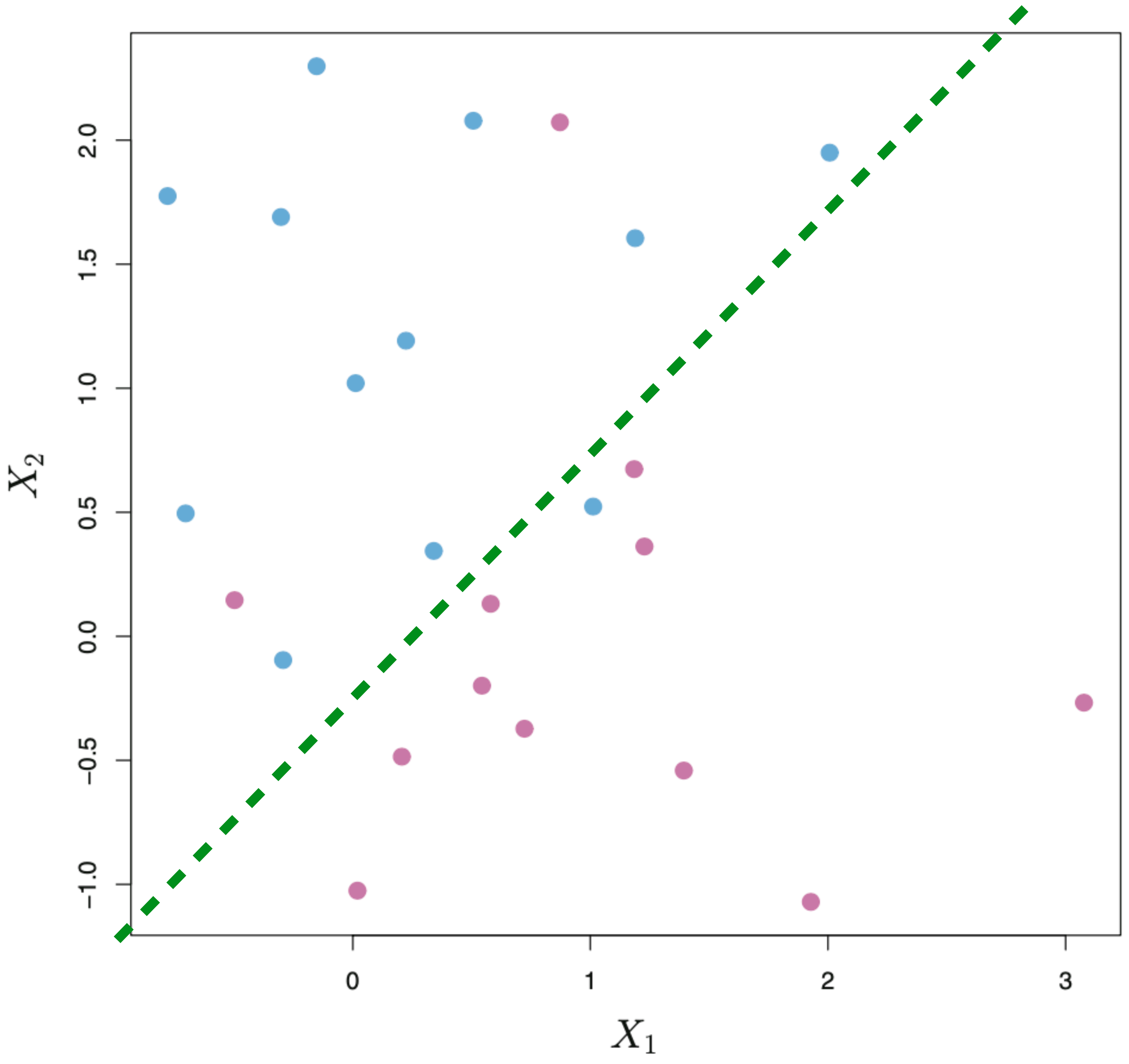
1. The circle correspond to the equation:
$$(X_1 - x_1)^2 + (X_2 - x_2)^2 - r^2 = 0$$
2. The point (p,q) is outside the circle if
$$(p - x_1)^2 + (q - x_2)^2 - r^2 > 0$$

Math fact 2: a nonlinear case



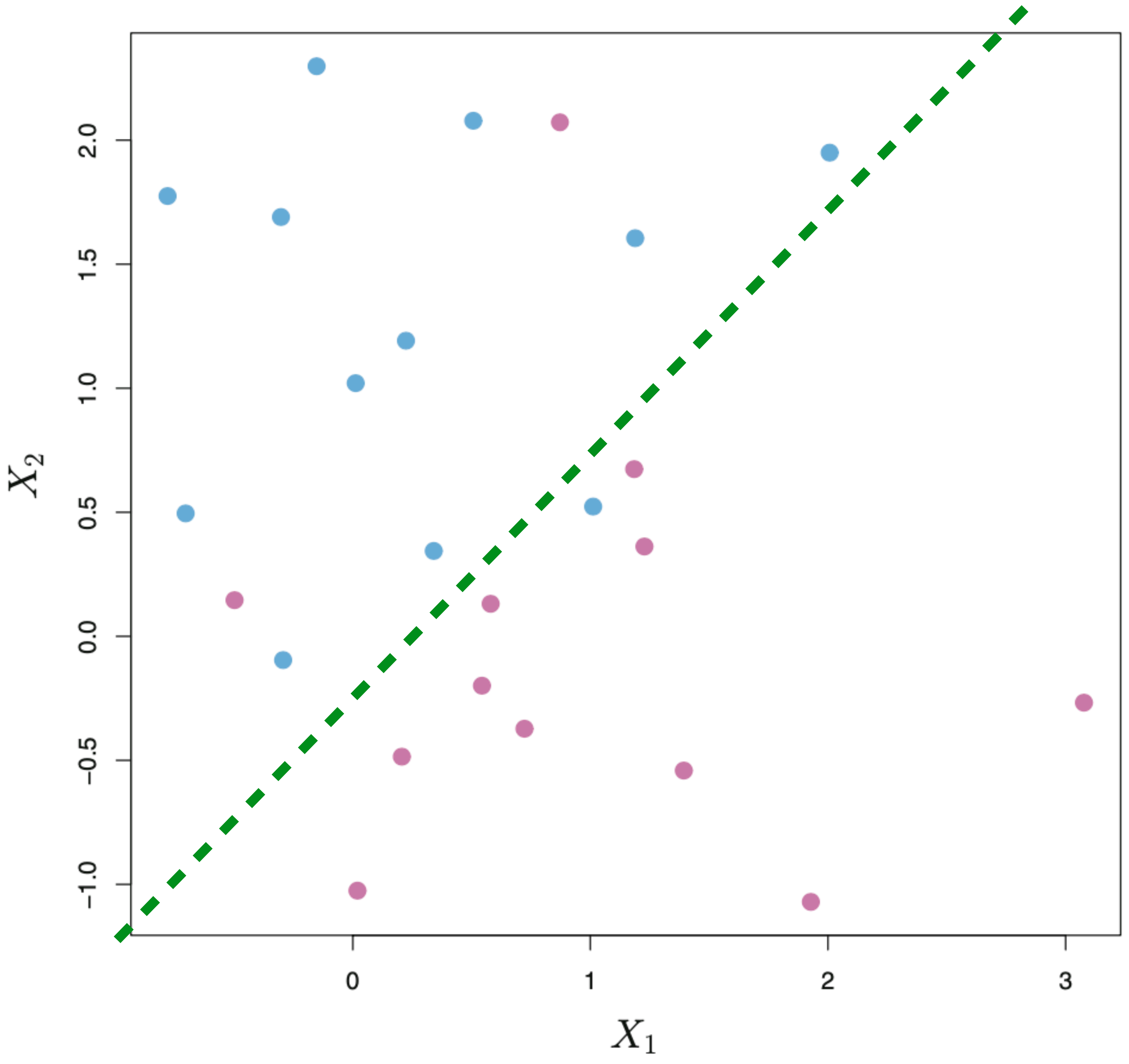
1. The circle correspond to the equation:
$$(X_1 - x_1)^2 + (X_2 - x_2)^2 - r^2 = 0$$
2. The point (p,q) is outside the circle if
$$(p - x_1)^2 + (q - x_2)^2 - r^2 > 0$$
3. The point (m, n) is inside the circle if
$$(m - x_1)^2 + (n - x_2)^2 - r^2 < 0$$

X1	X2	Y
1.5	-0.5	1
0.2	1.2	-1



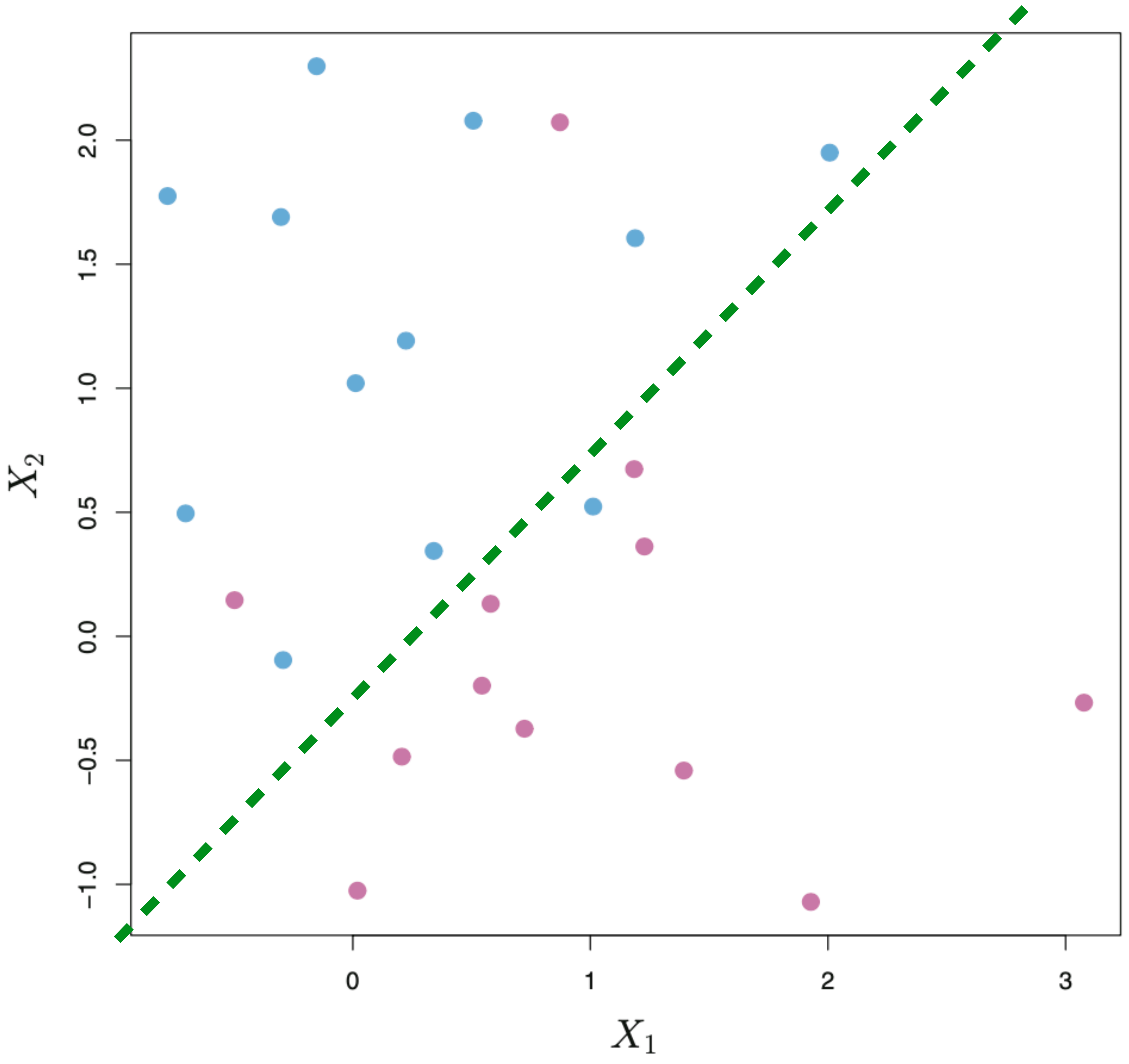
The problem:

X1	X2	Y
1.5	-0.5	1
0.2	1.2	-1



The problem:
Given the data set in which there are two features (X1,X2) and the label (Y=1,-1), we try to find the model f(X1,X2) such that the data with different labels are well separate, i.e.

X1	X2	Y
1.5	-0.5	1
0.2	1.2	-1

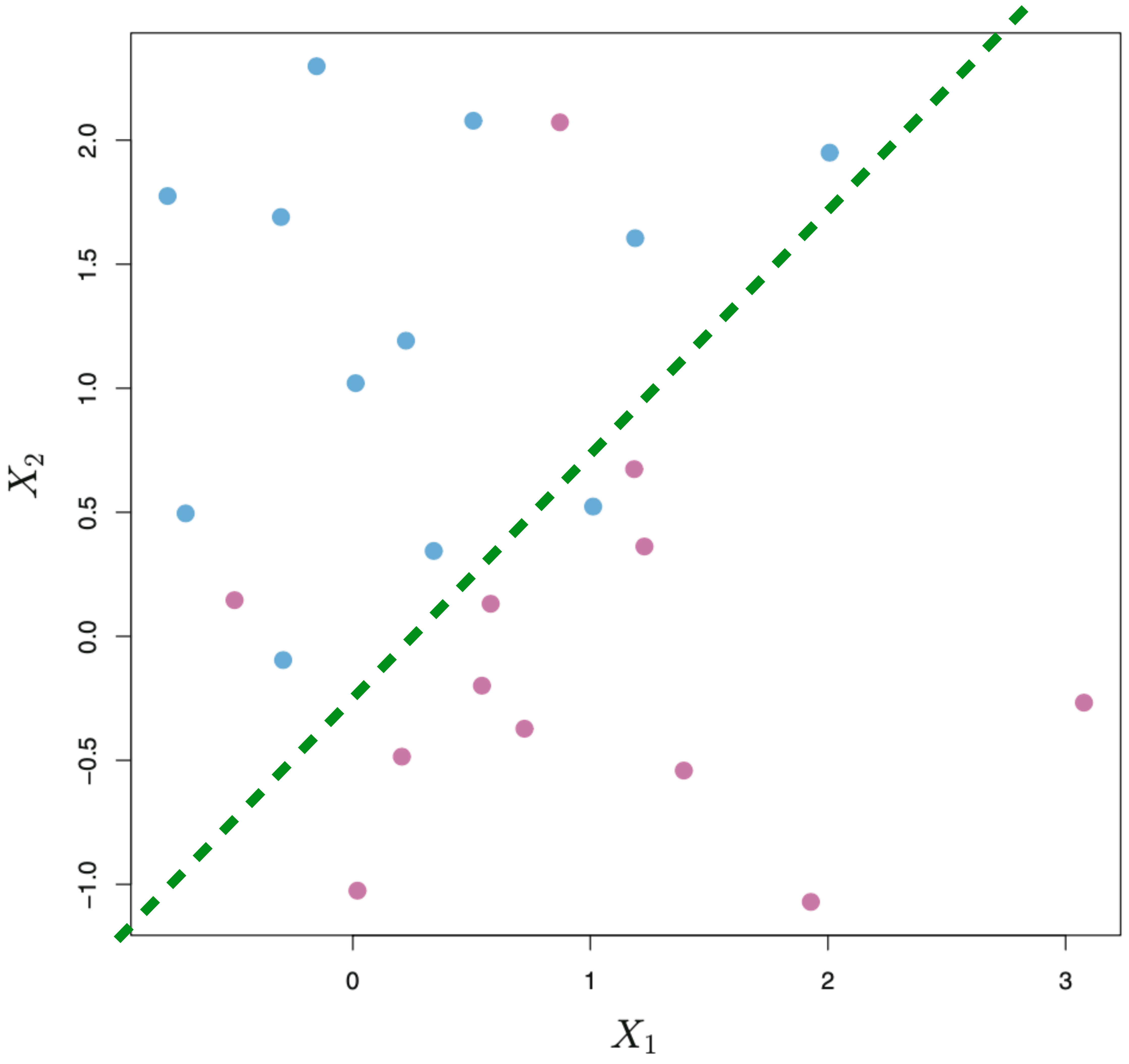


The problem:

Given the data set in which there are two features (X1,X2) and the label (Y=1,-1), we try to find the model f(X1,X2) such that the data with different labels are well separate, i.e.

1. For data with Y = 1, the function f(X1,X2) > 0

X1	X2	Y
1.5	-0.5	1
0.2	1.2	-1

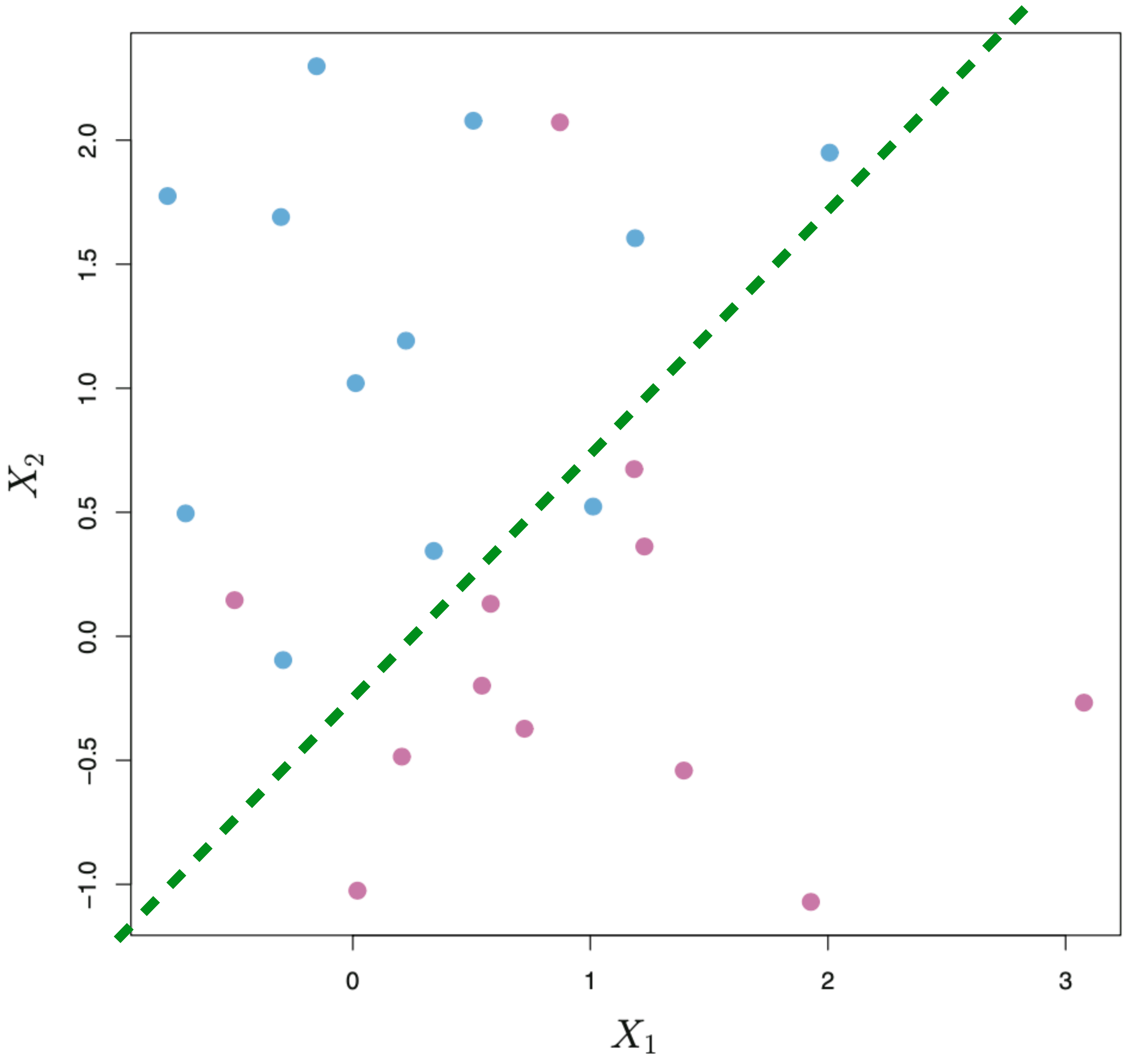


The problem:

Given the data set in which there are two features (X1,X2) and the label (Y=1,-1), we try to find the model f(X1,X2) such that the data with different labels are well separate, i.e.

1. For data with Y = 1, the function f(X1,X2) > 0
2. For data with Y = -1, the function f(X1,X2) < 0

X1	X2	Y
1.5	-0.5	1
0.2	1.2	-1

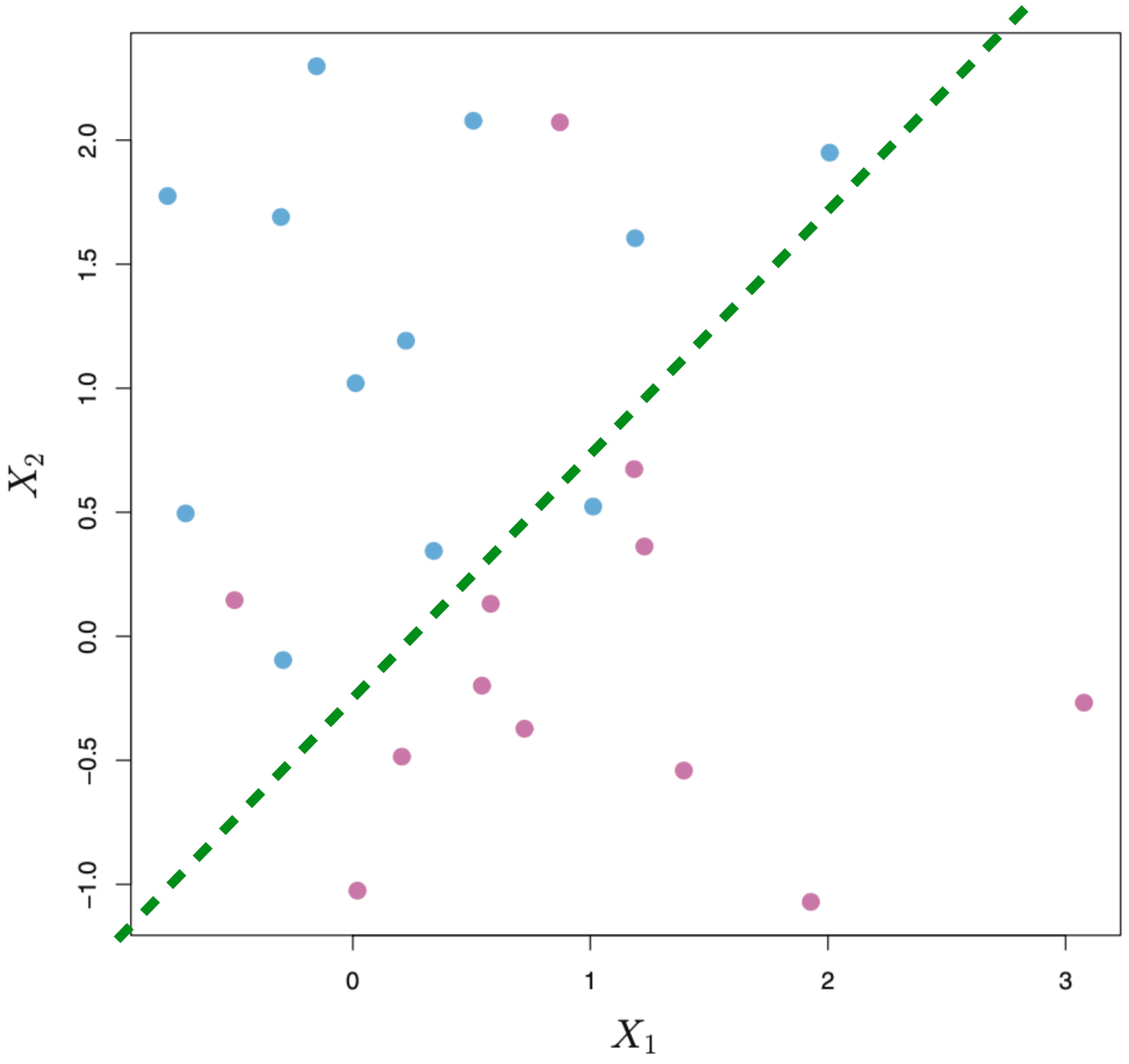


The problem:

Given the data set in which there are two features (X1,X2) and the label (Y=1,-1), we try to find the model f(X1,X2) such that the data with different labels are well separate, i.e.

1. For data with Y = 1, the function f(X1,X2) > 0
2. For data with Y = -1, the function f(X1,X2) < 0
3. Or we can summarize for all data Y*f(X1,X2) > 0

X1	X2	Y
1.5	-0.5	1
0.2	1.2	-1

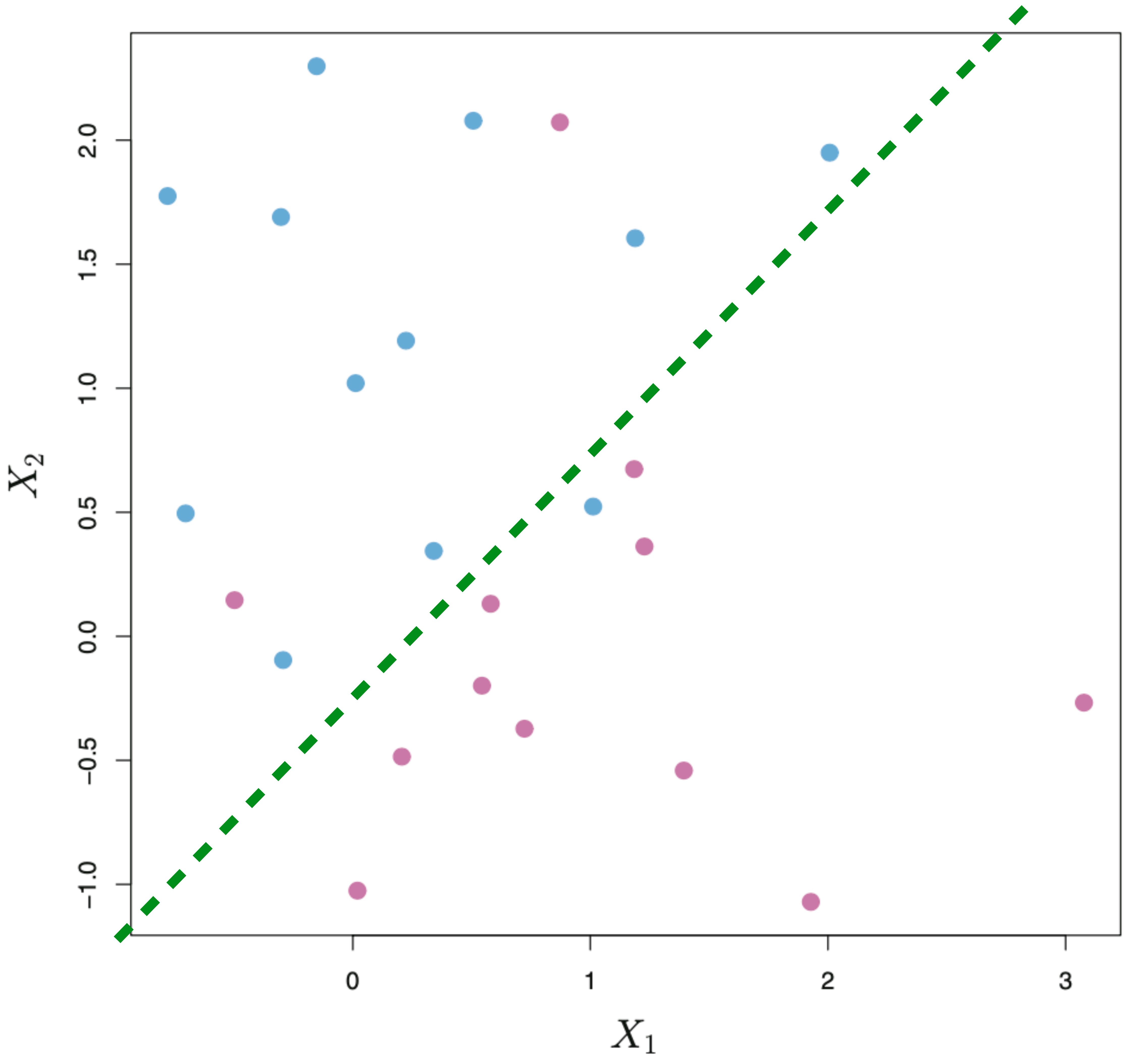


The problem:

Given the data set in which there are two features (X1,X2) and the label (Y=1,-1), we try to find the model f(X1,X2) such that the data with different labels are well separate, i.e.

1. For data with Y = 1, the function f(X1,X2) > 0
2. For data with Y = -1, the function f(X1,X2) < 0
3. Or we can summarize for all data Y*f(X1,X2) > 0

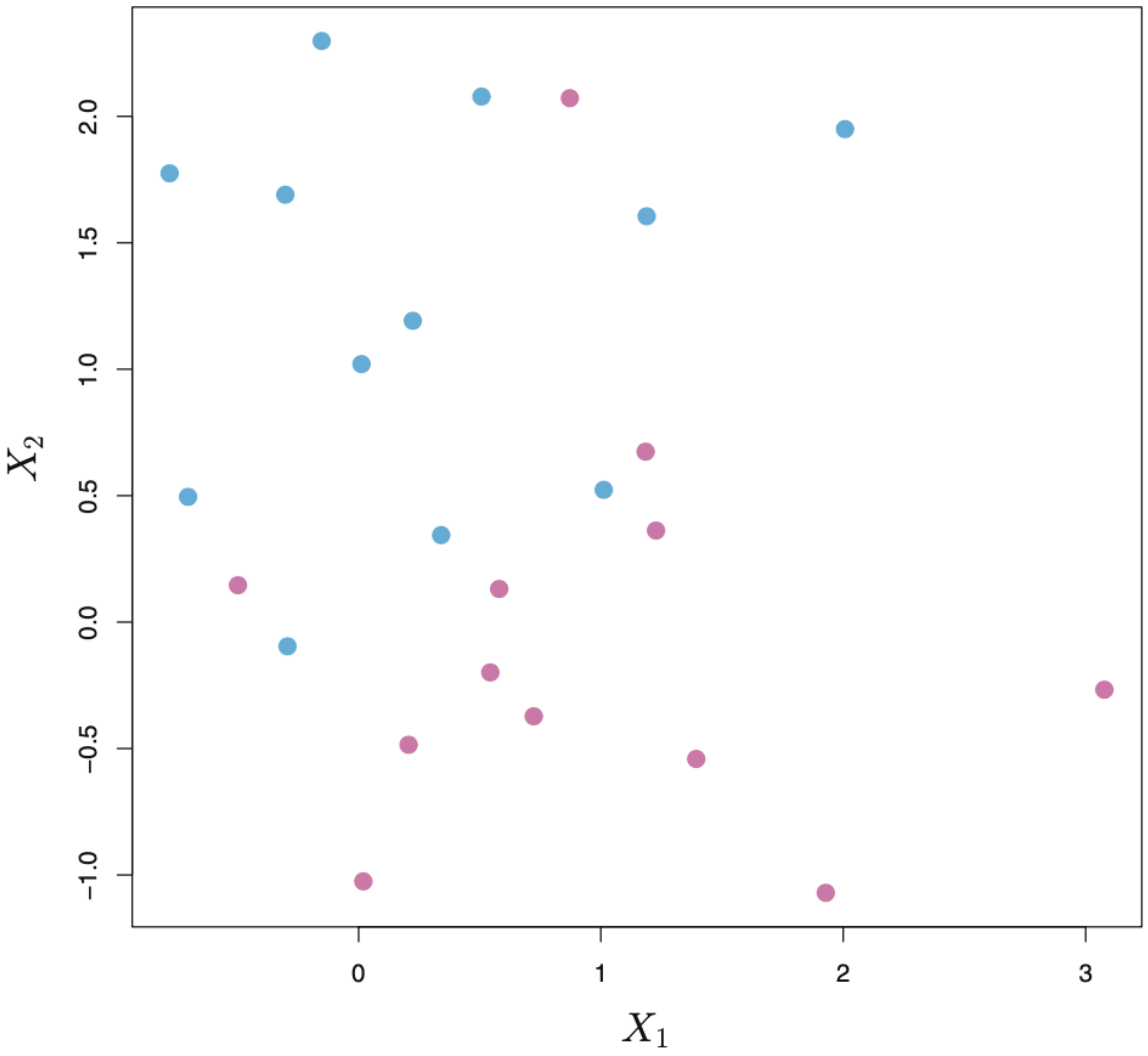
X1	X2	Y
1.5	-0.5	1
0.2	1.2	-1



What is the model f(X1, X2):

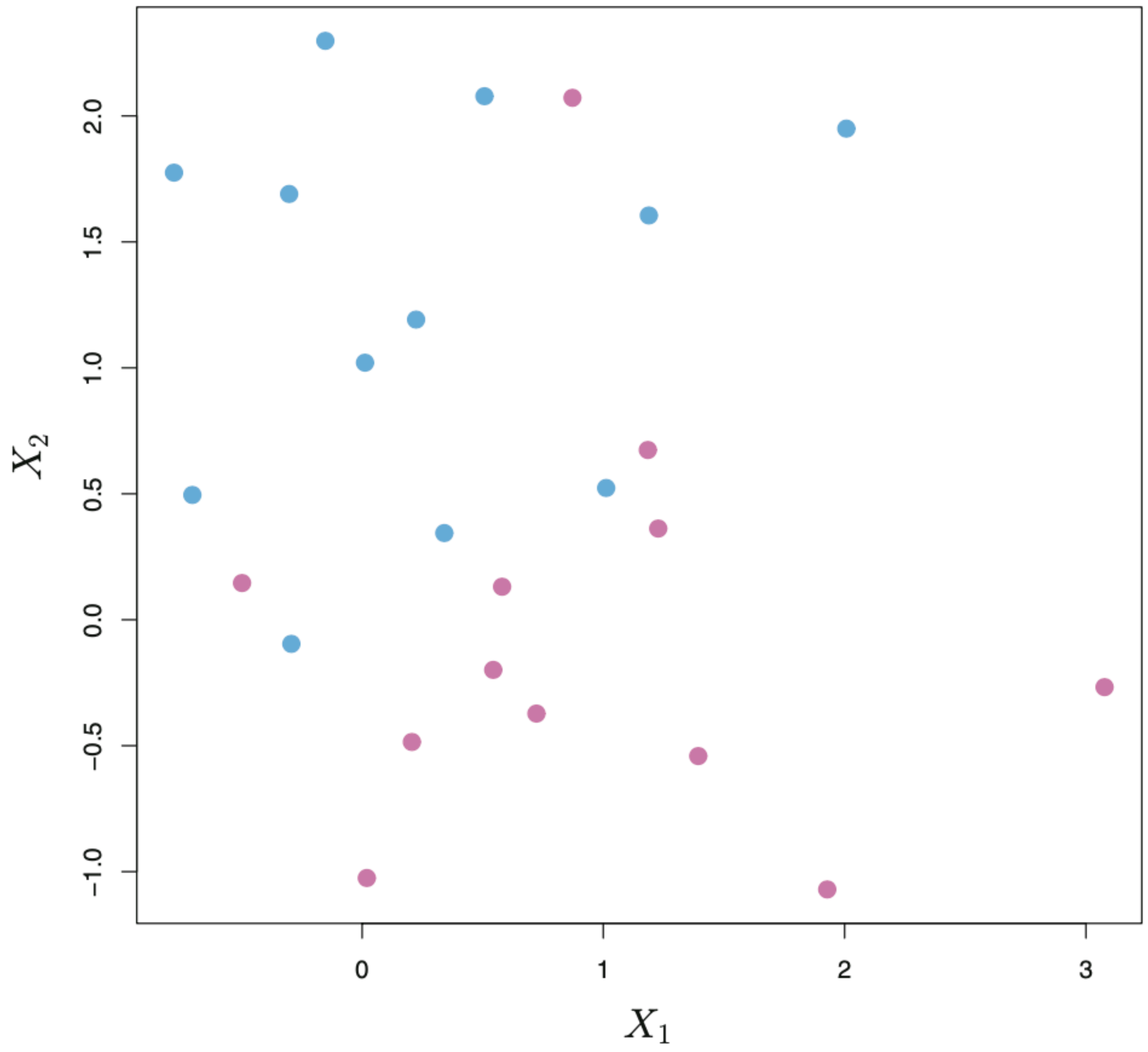
In the linear separable case, $f(X_1, X_2) = aX_1 + bX_2 + c$. And we use machine learning algorithm to find out the parameters a, b, c so that the 'loss' is minimized.

Some intuitions:



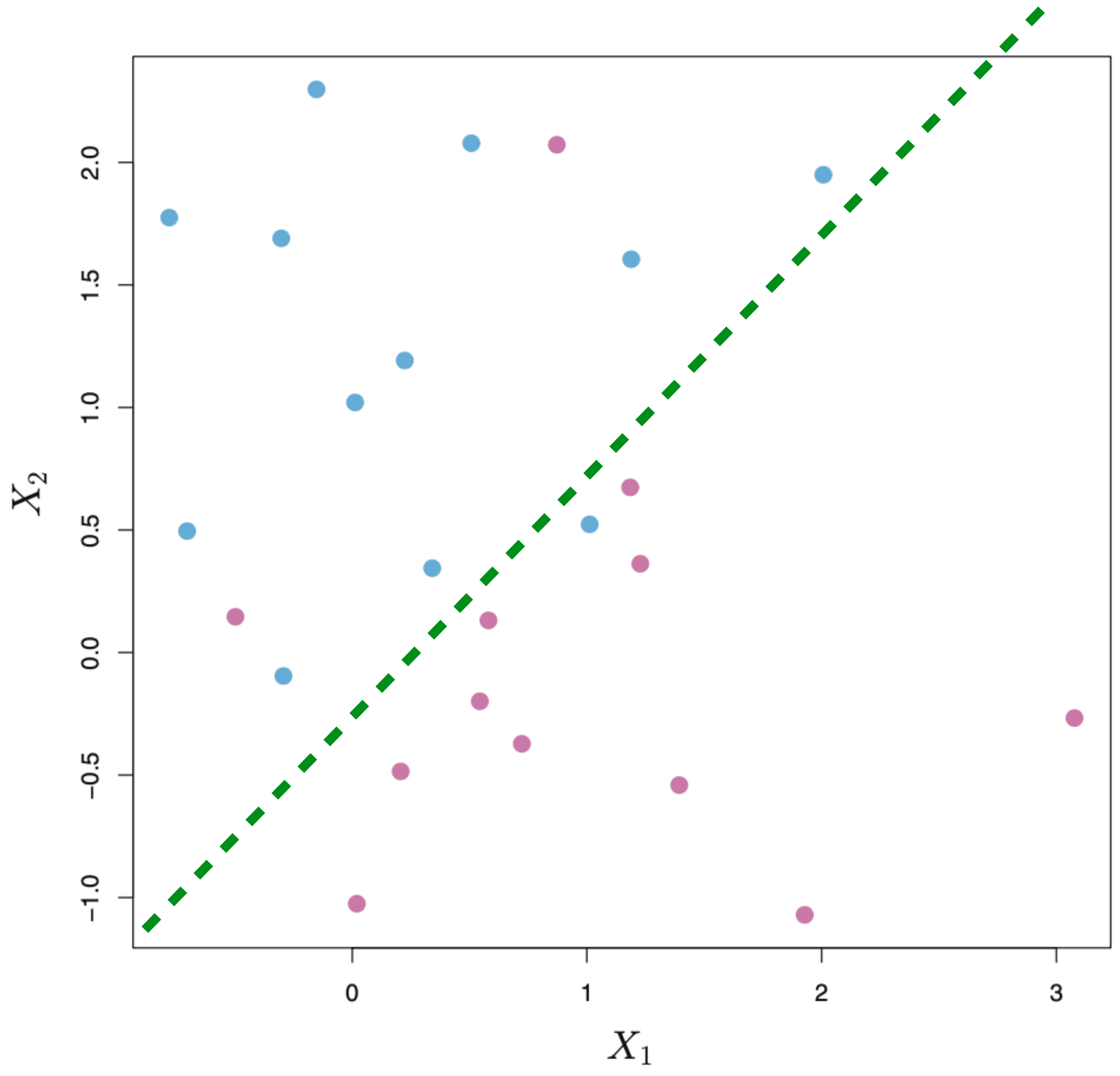
Some intuitions:

- It's easy to draw a line separating the two classes of data



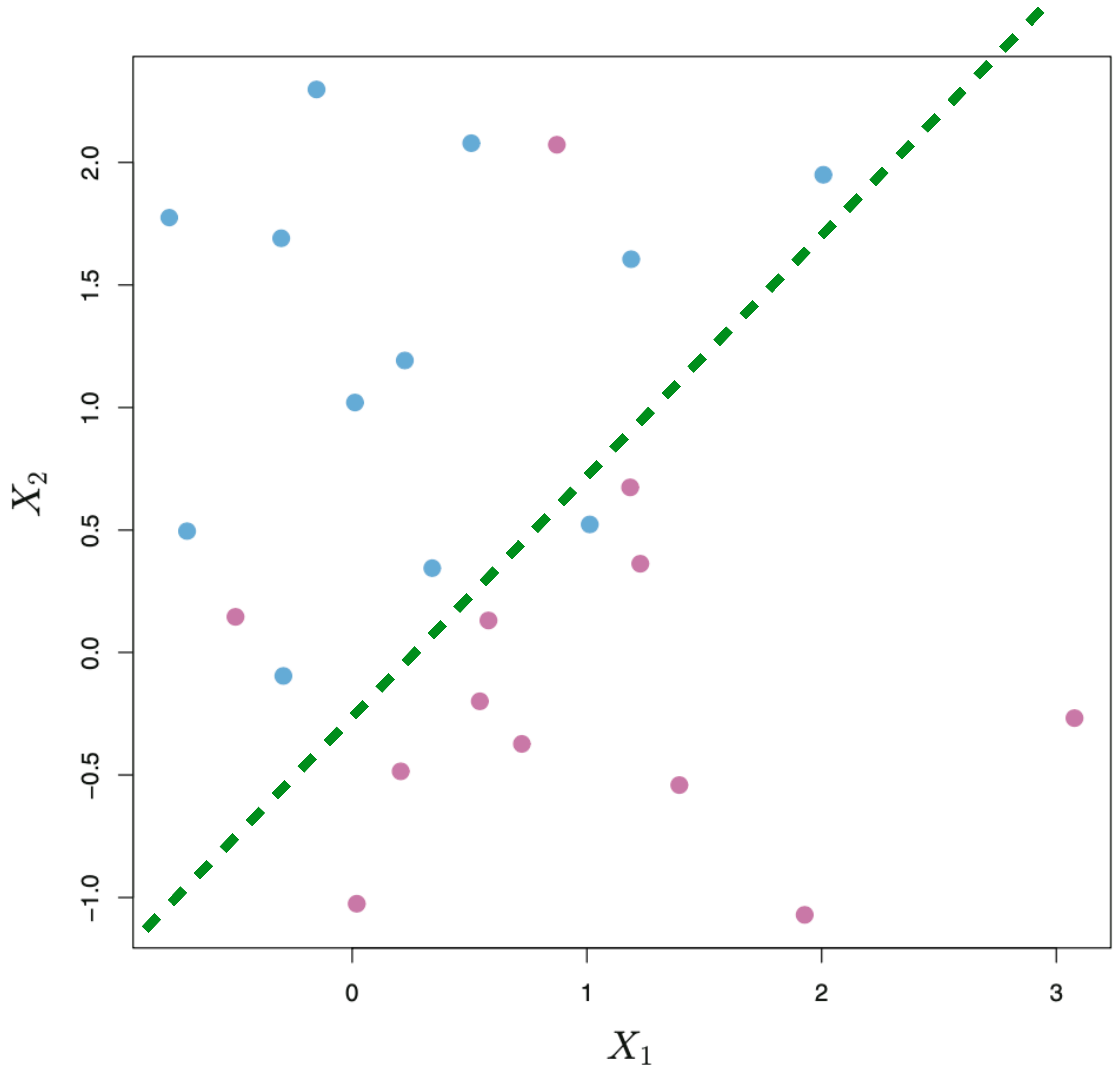
Some intuitions:

- It's easy to draw a line separating the two classes of data



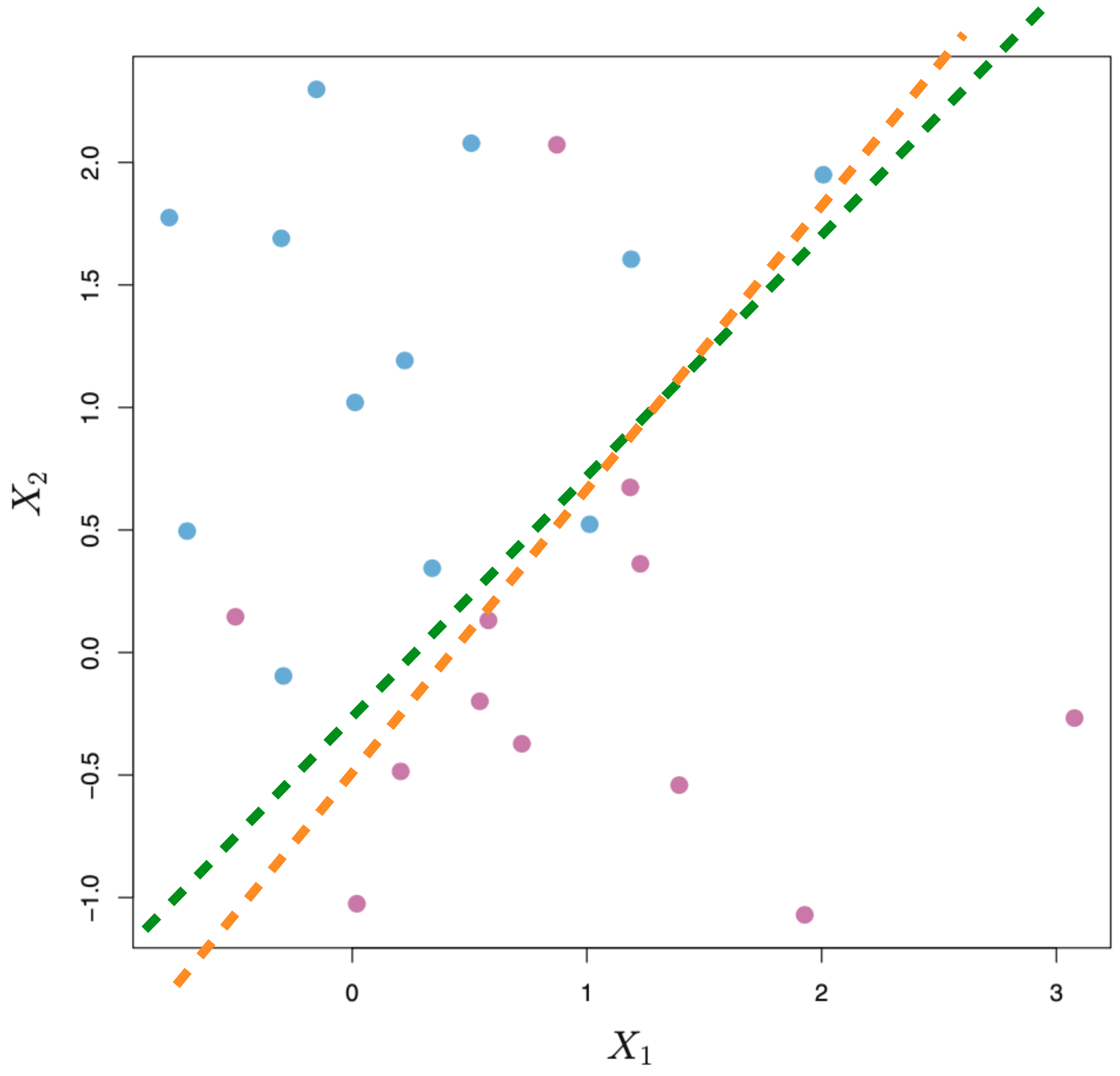
Some intuitions:

- It's easy to draw a line separating the two classes of data
- But there are many lines that can do the job



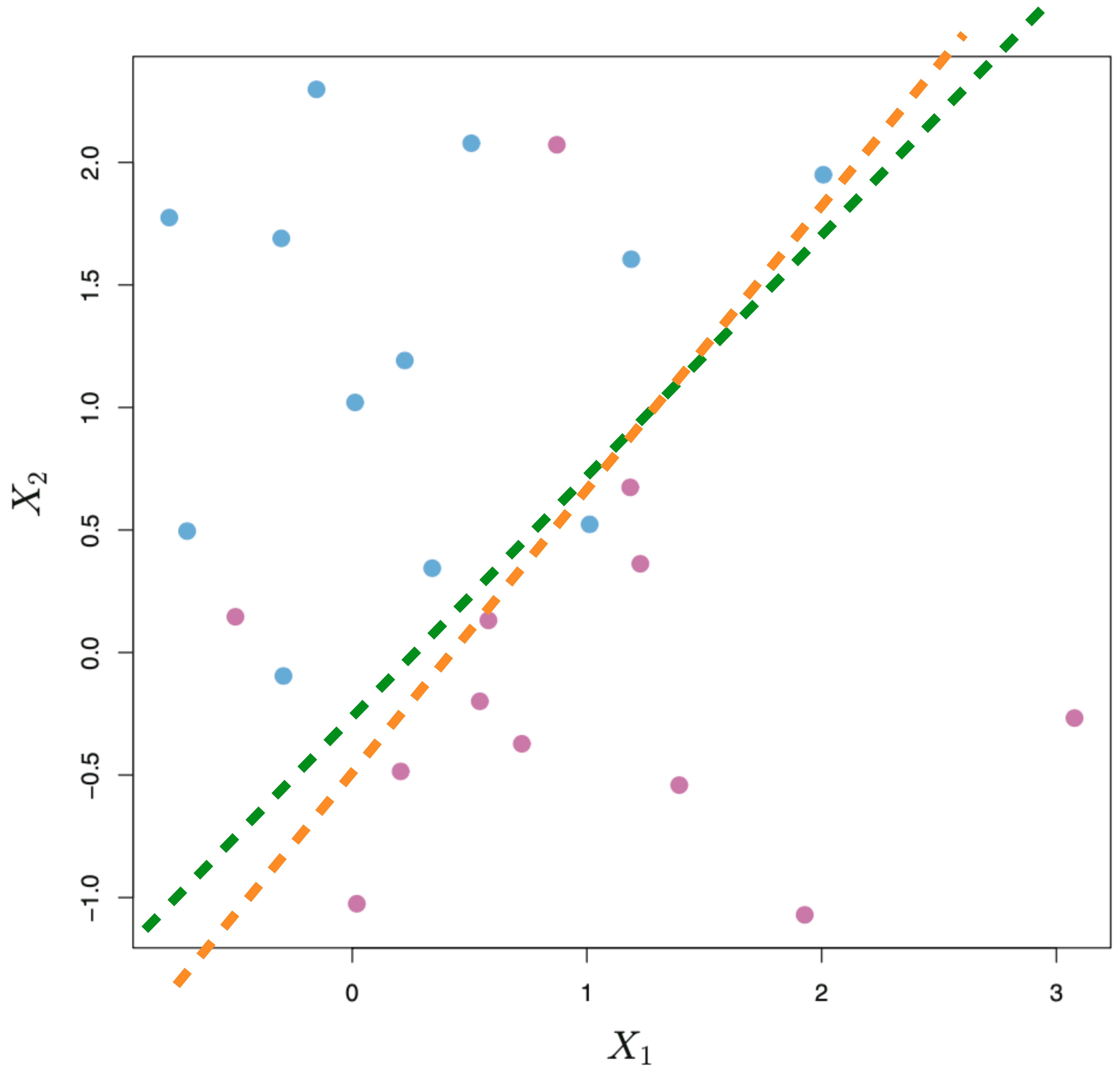
Some intuitions:

- It's easy to draw a line separating the two classes of data
- But there are many lines that can do the job



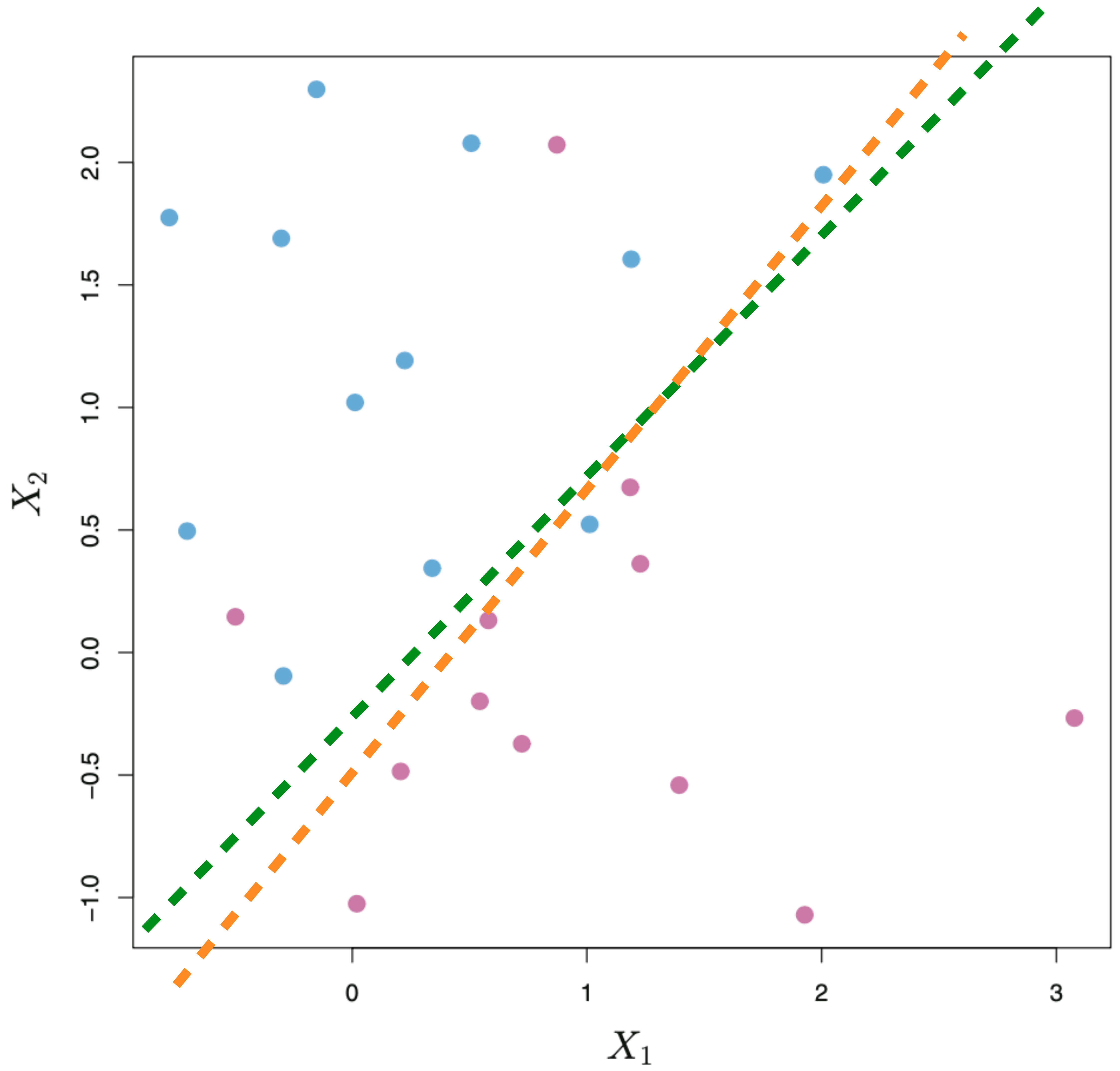
Some intuitions:

- It's easy to draw a line separating the two classes of data
- But there are many lines that can do the job
- How do we select the best line?

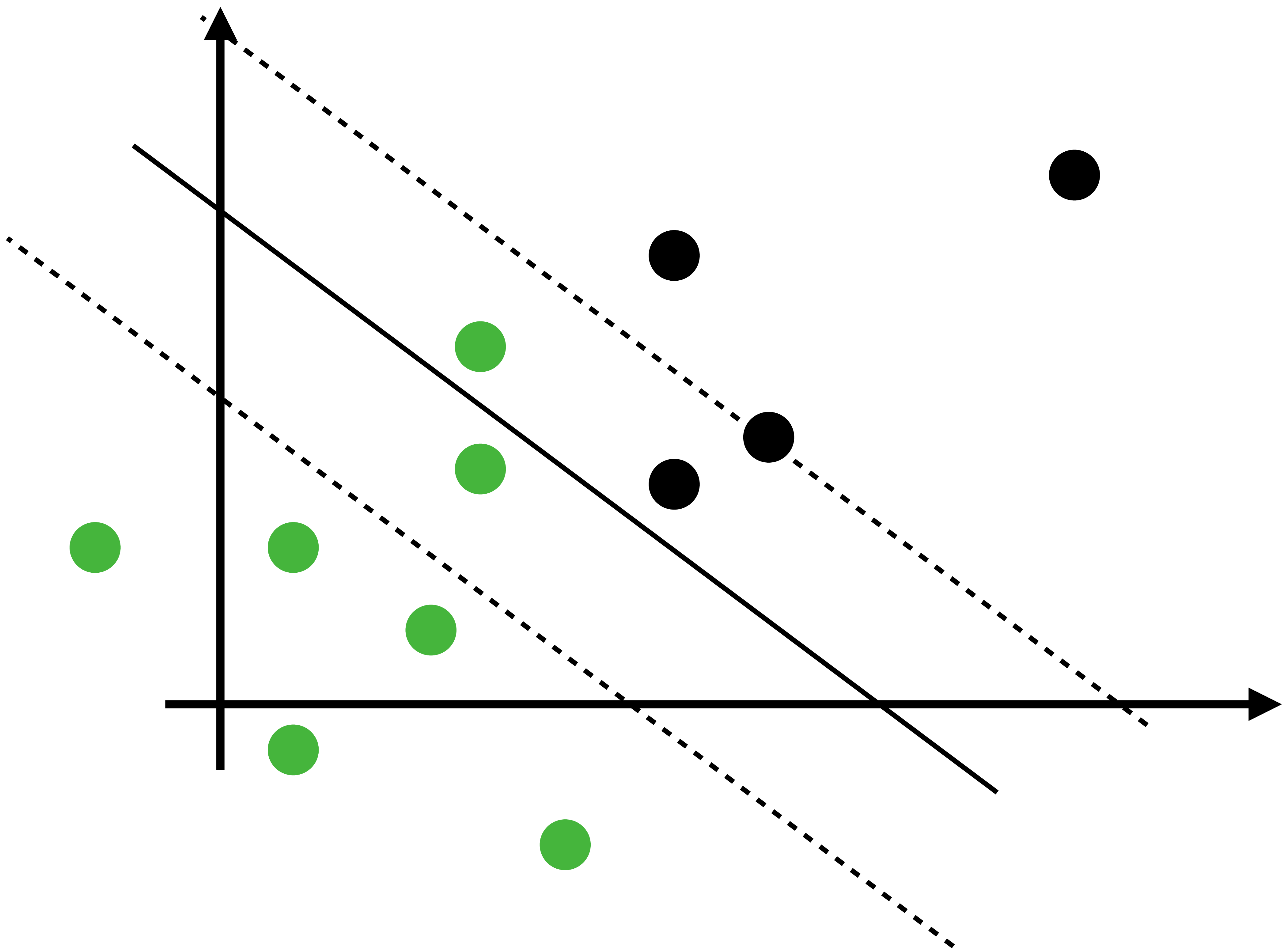


Some intuitions:

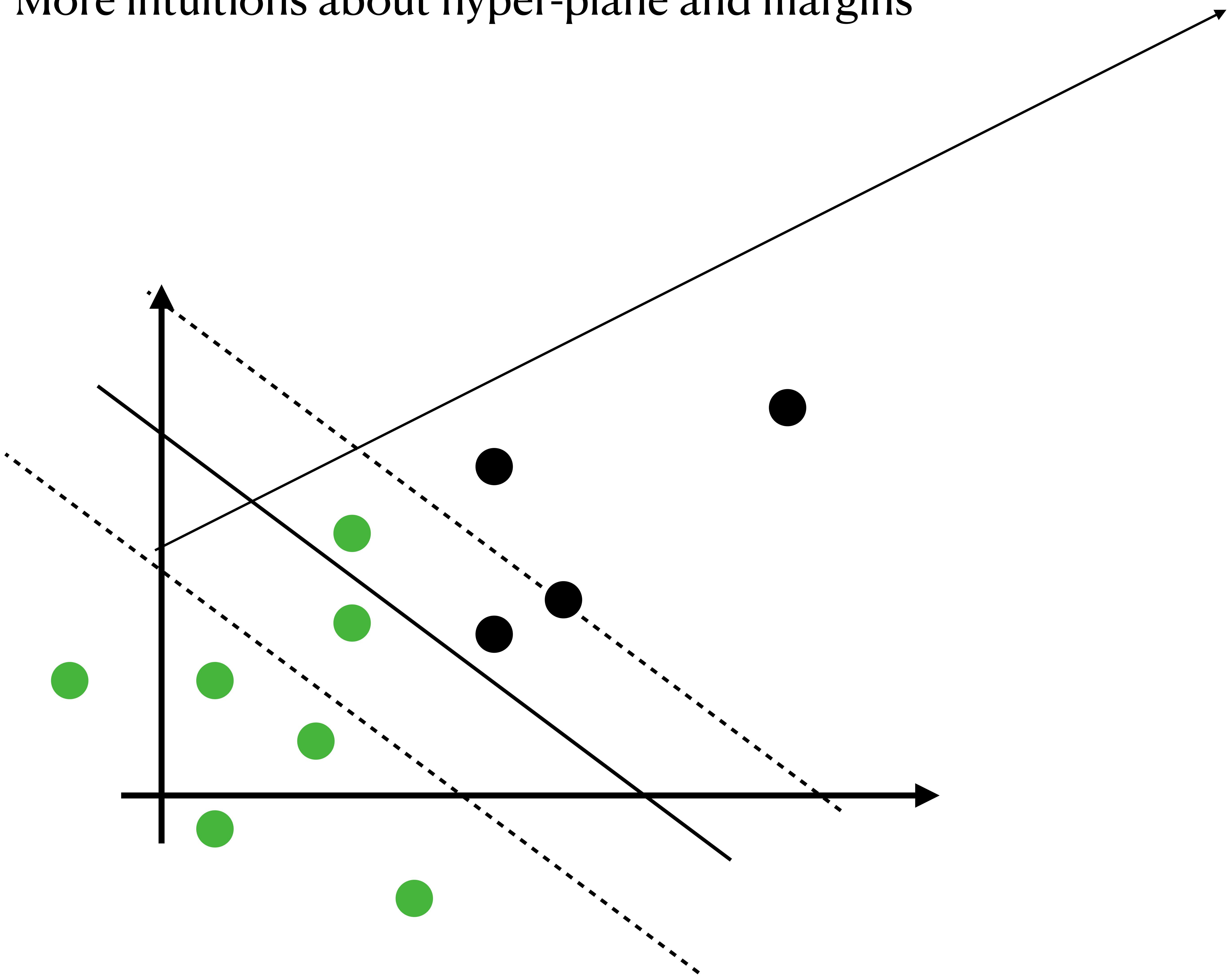
- It’s easy to draw a line separating the two classes of data
- But there are many lines that can do the job
- How do we select the best line?
- We want robustness: perturbation to the training points does not change the line significantly



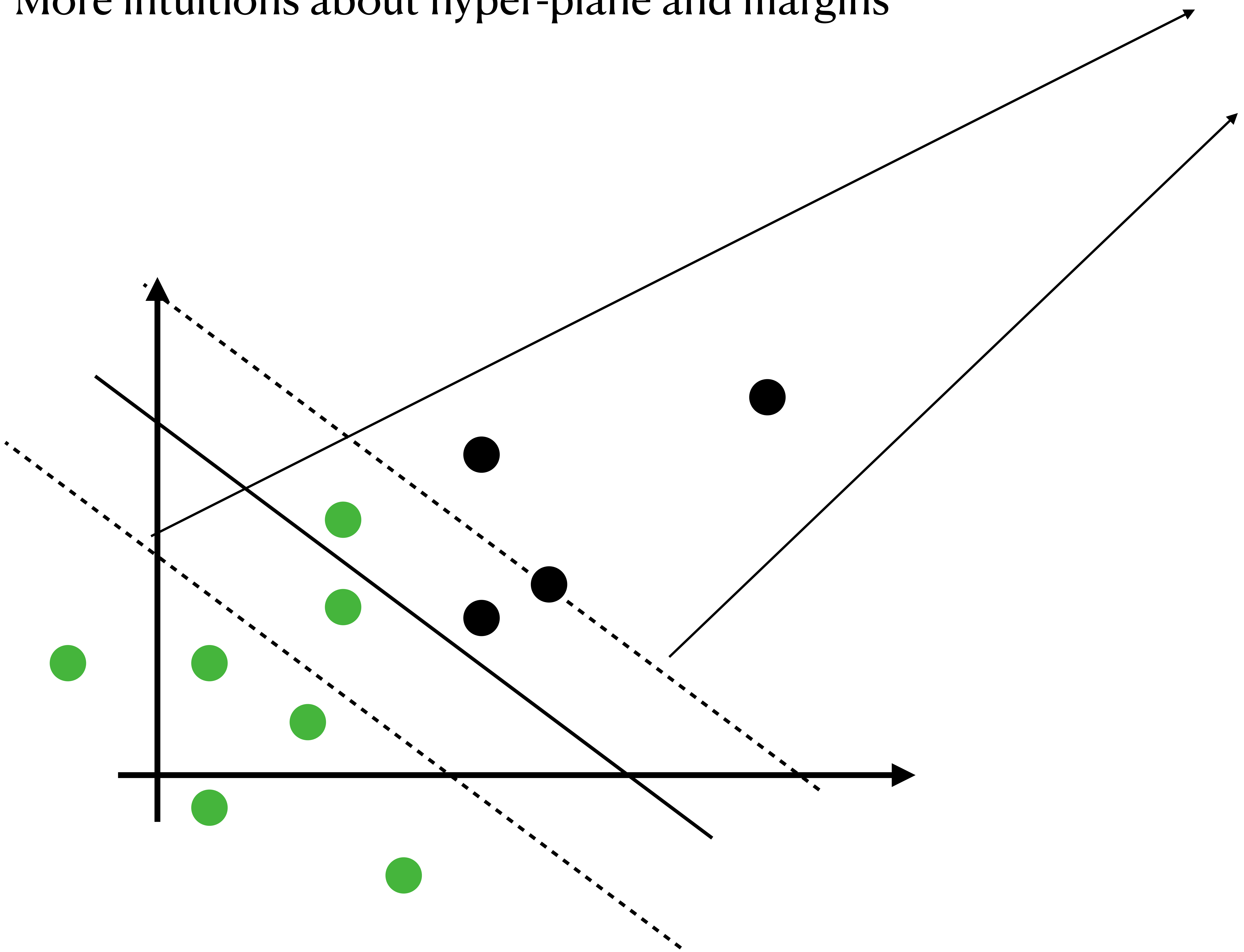
More intuitions about hyper-plane and margins



More intuitions about hyper-plane and margins

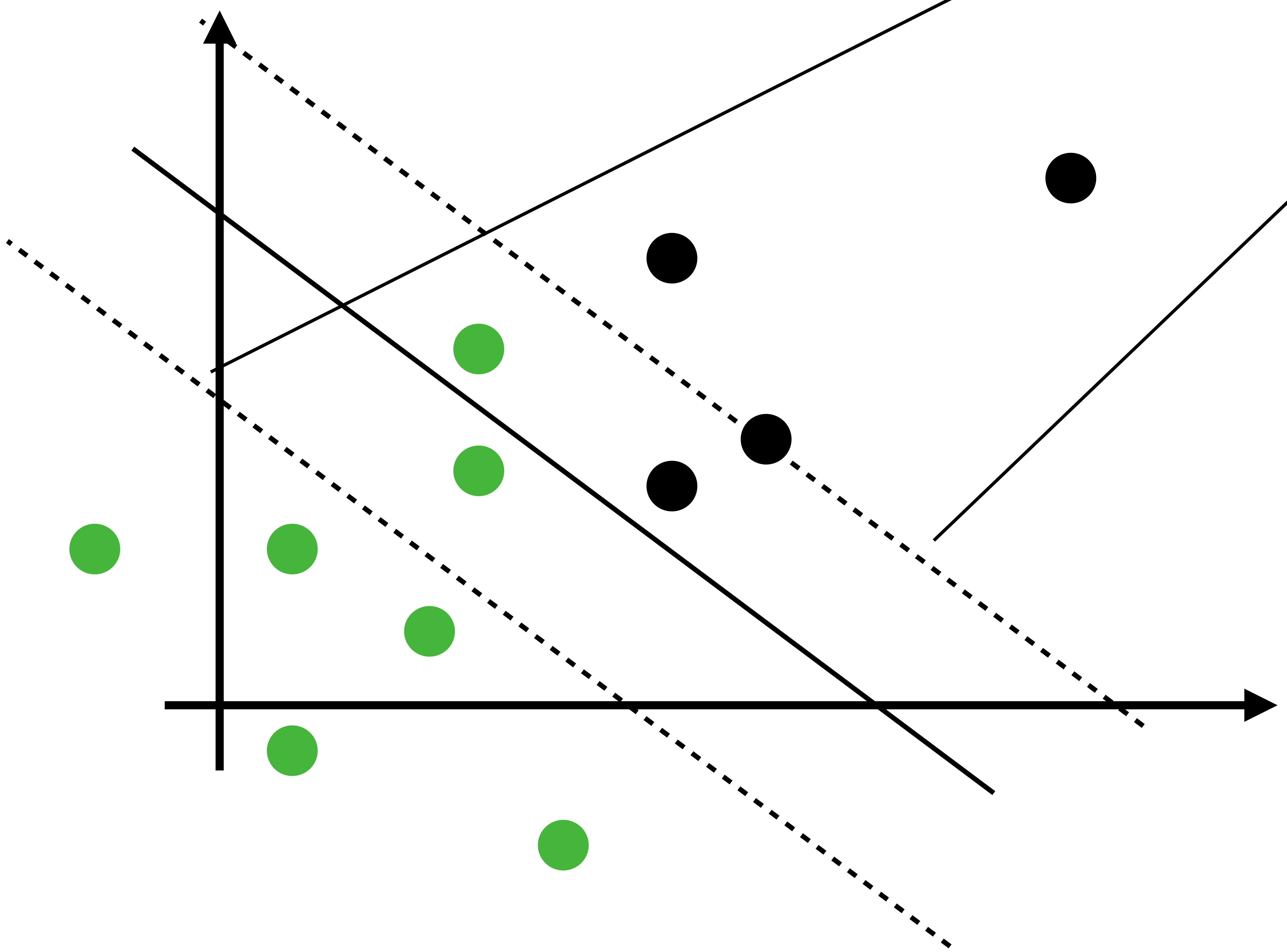


More intuitions about hyper-plane and margins



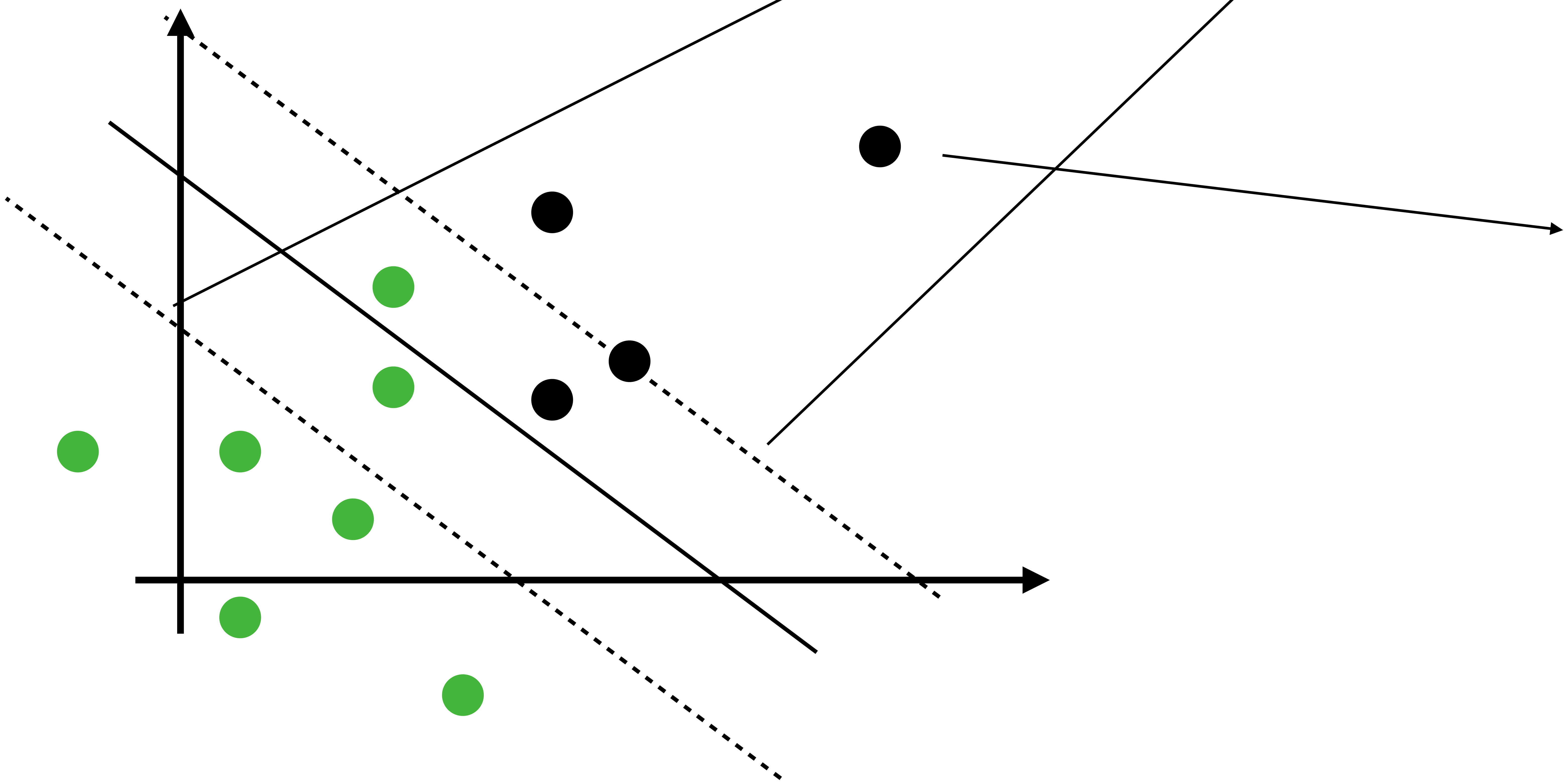
More intuitions about hyper-plane and margins

Margin: protection bands for the hyper-plane



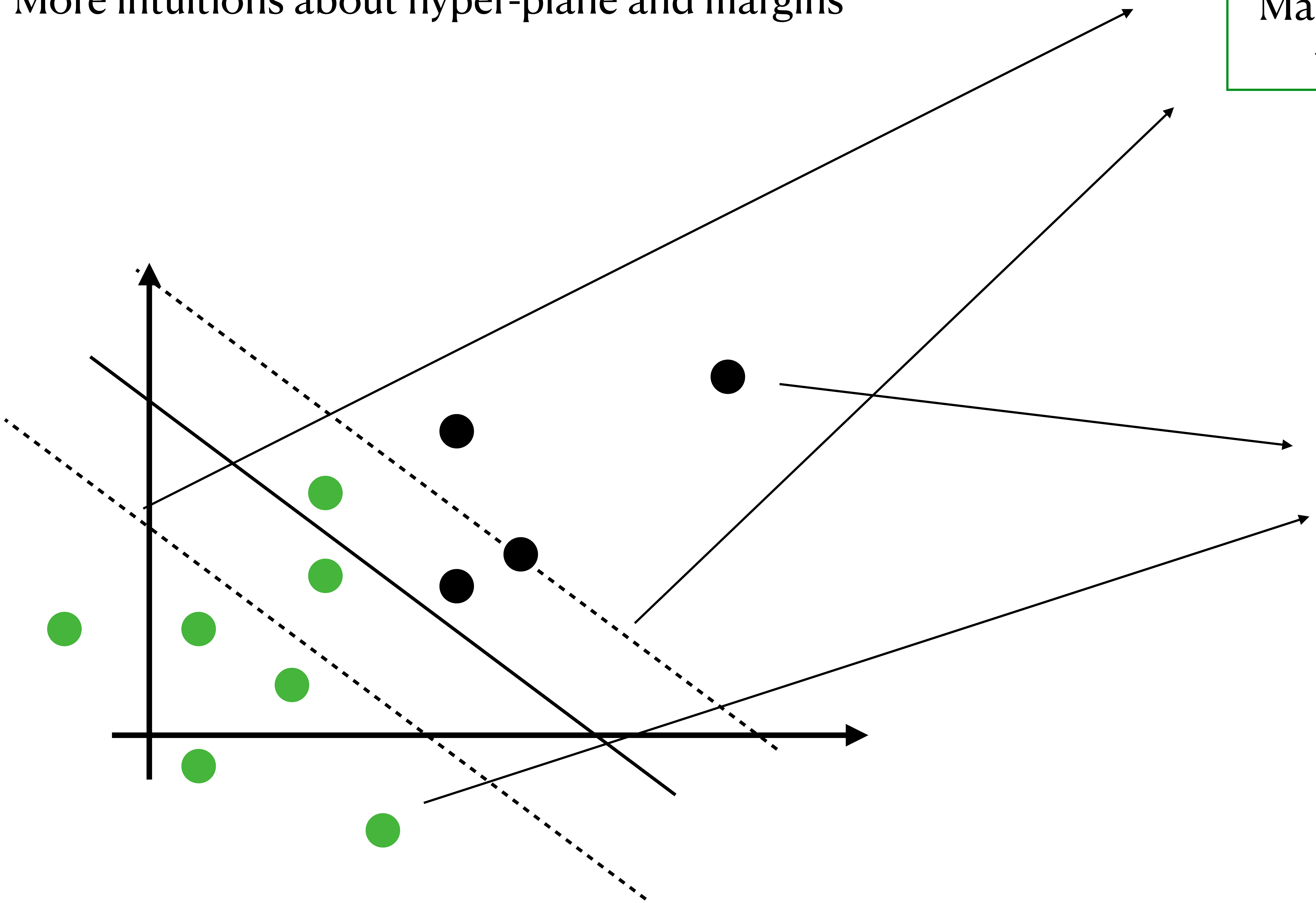
More intuitions about hyper-plane and margins

Margin: protection bands for the hyper-plane

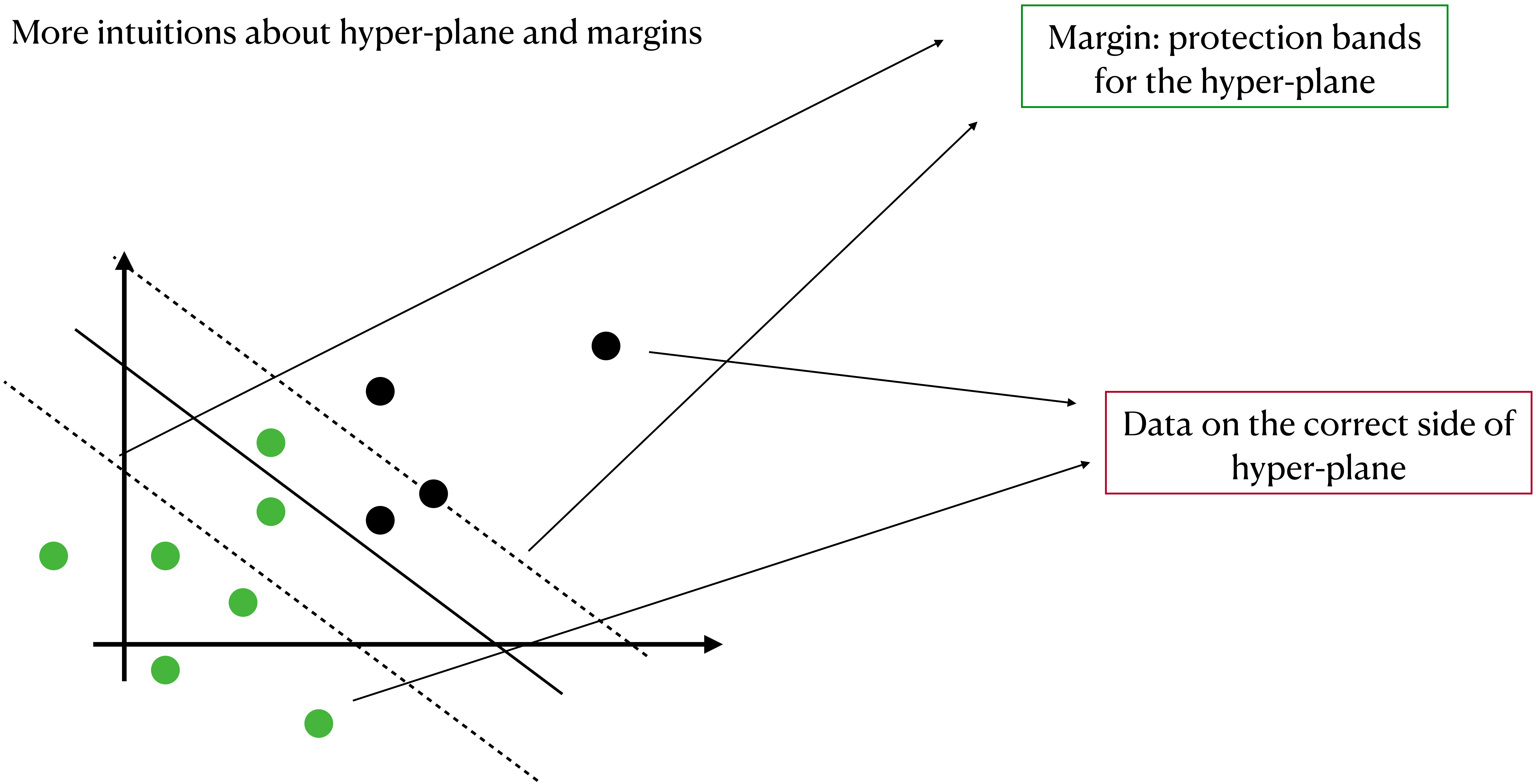


More intuitions about hyper-plane and margins

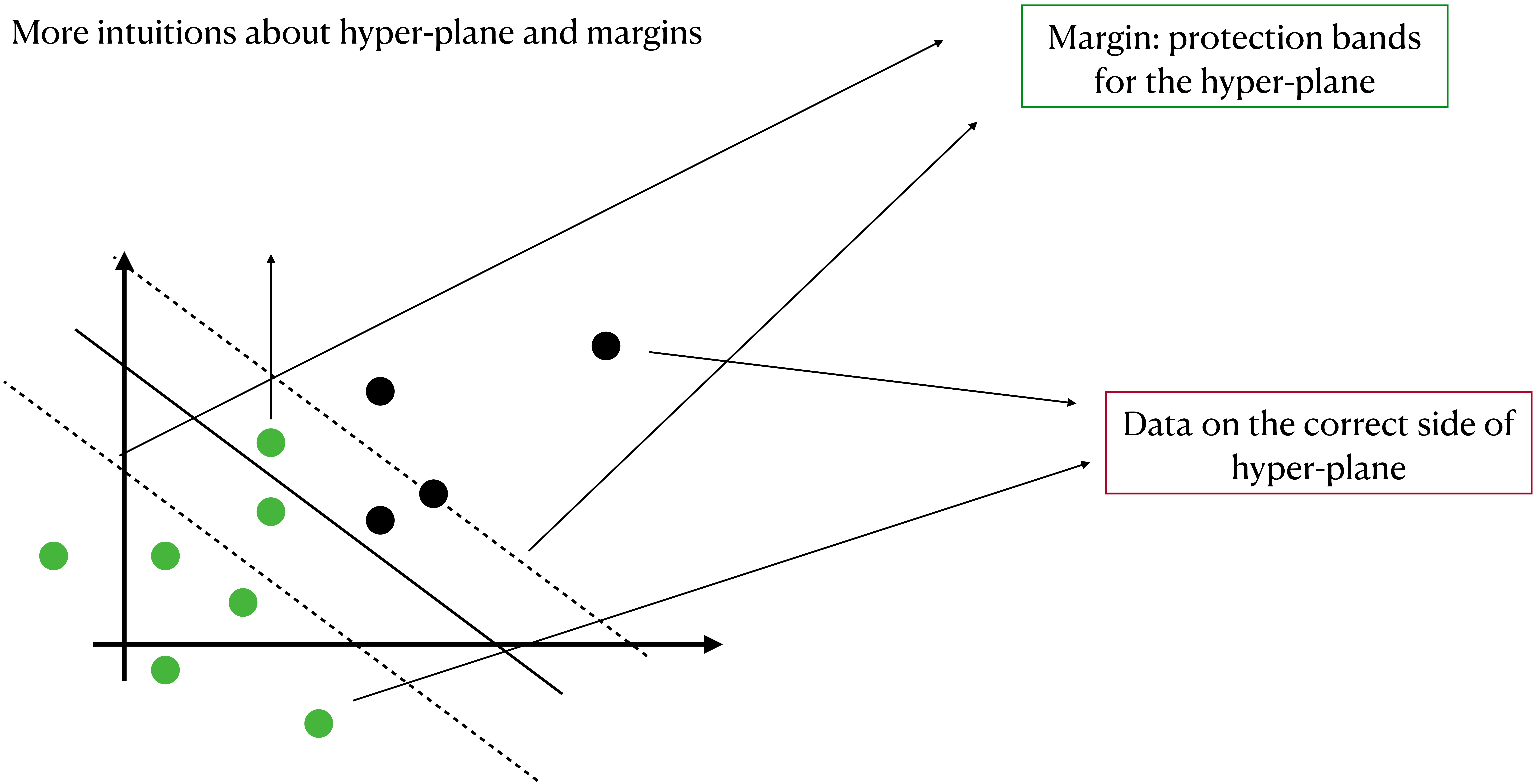
Margin: protection bands for the hyper-plane



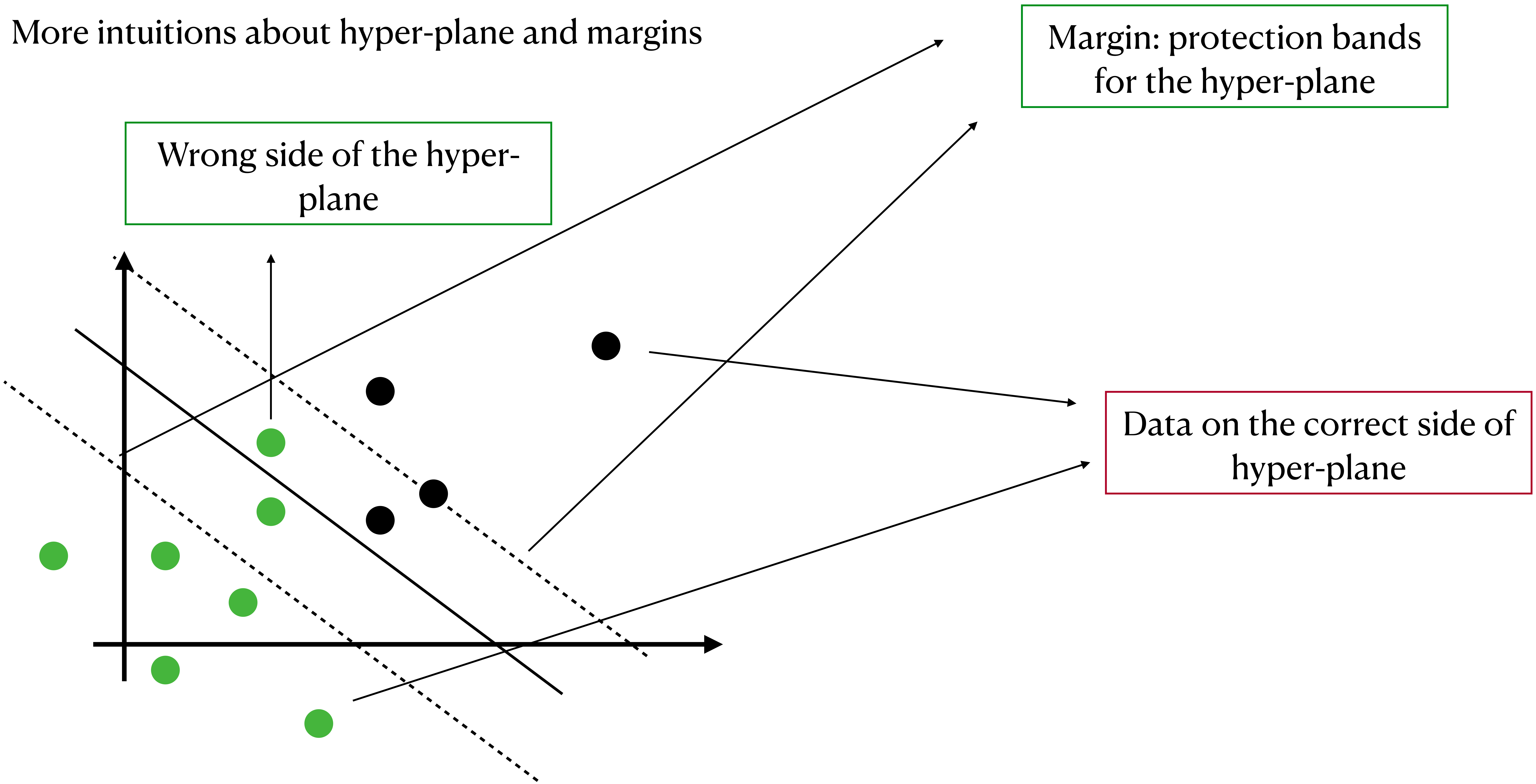
More intuitions about hyper-plane and margins



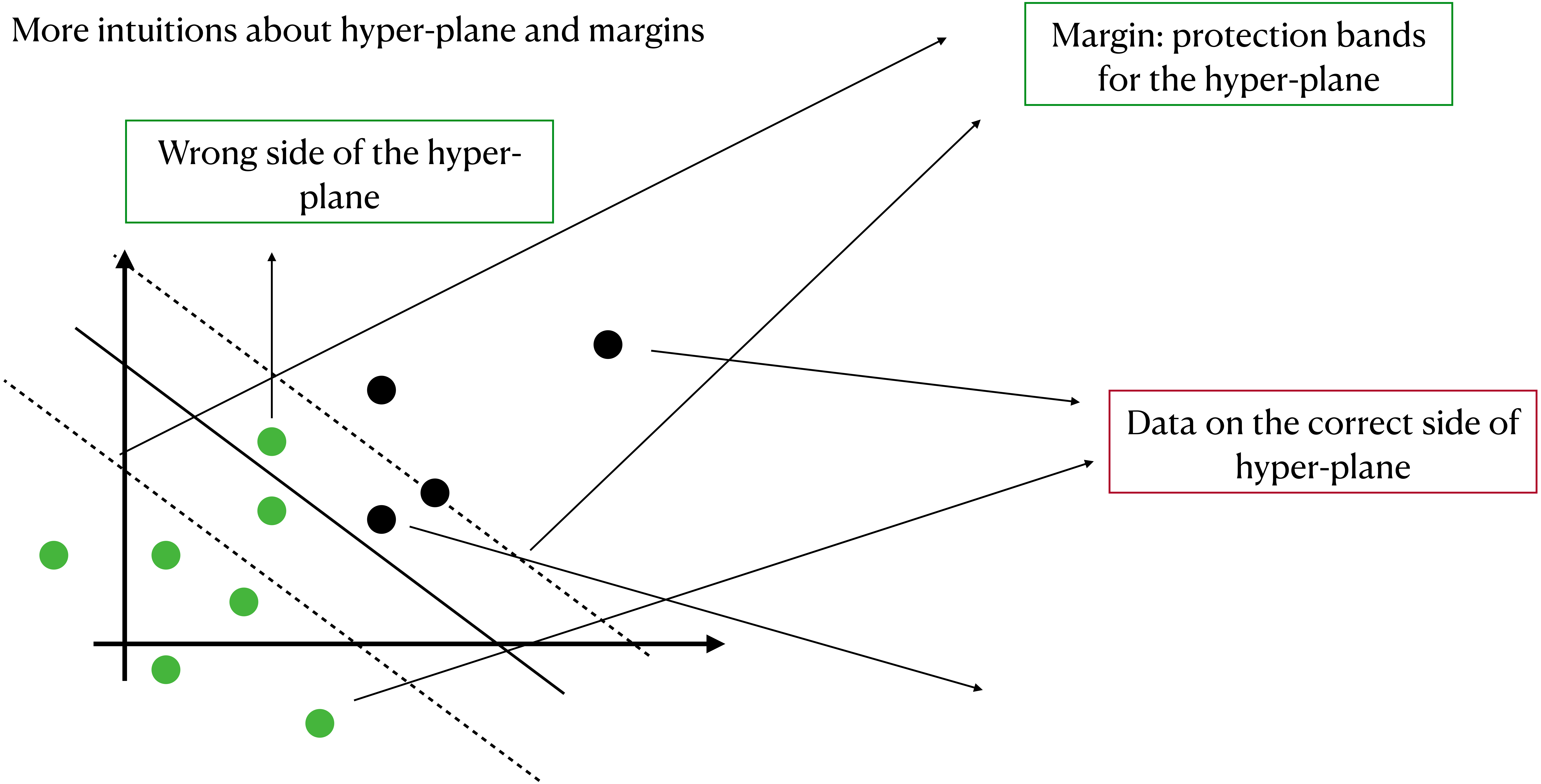
More intuitions about hyper-plane and margins



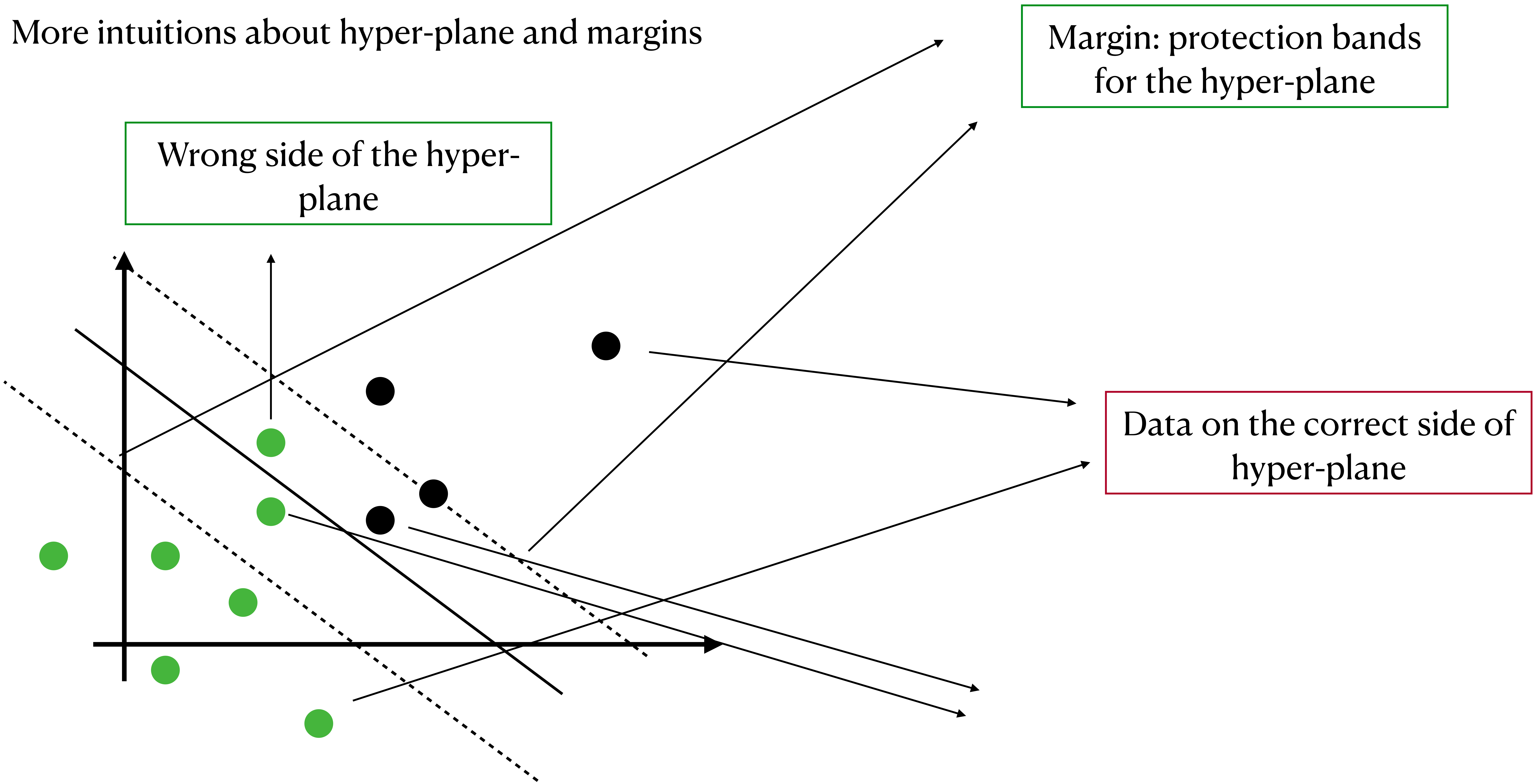
More intuitions about hyper-plane and margins



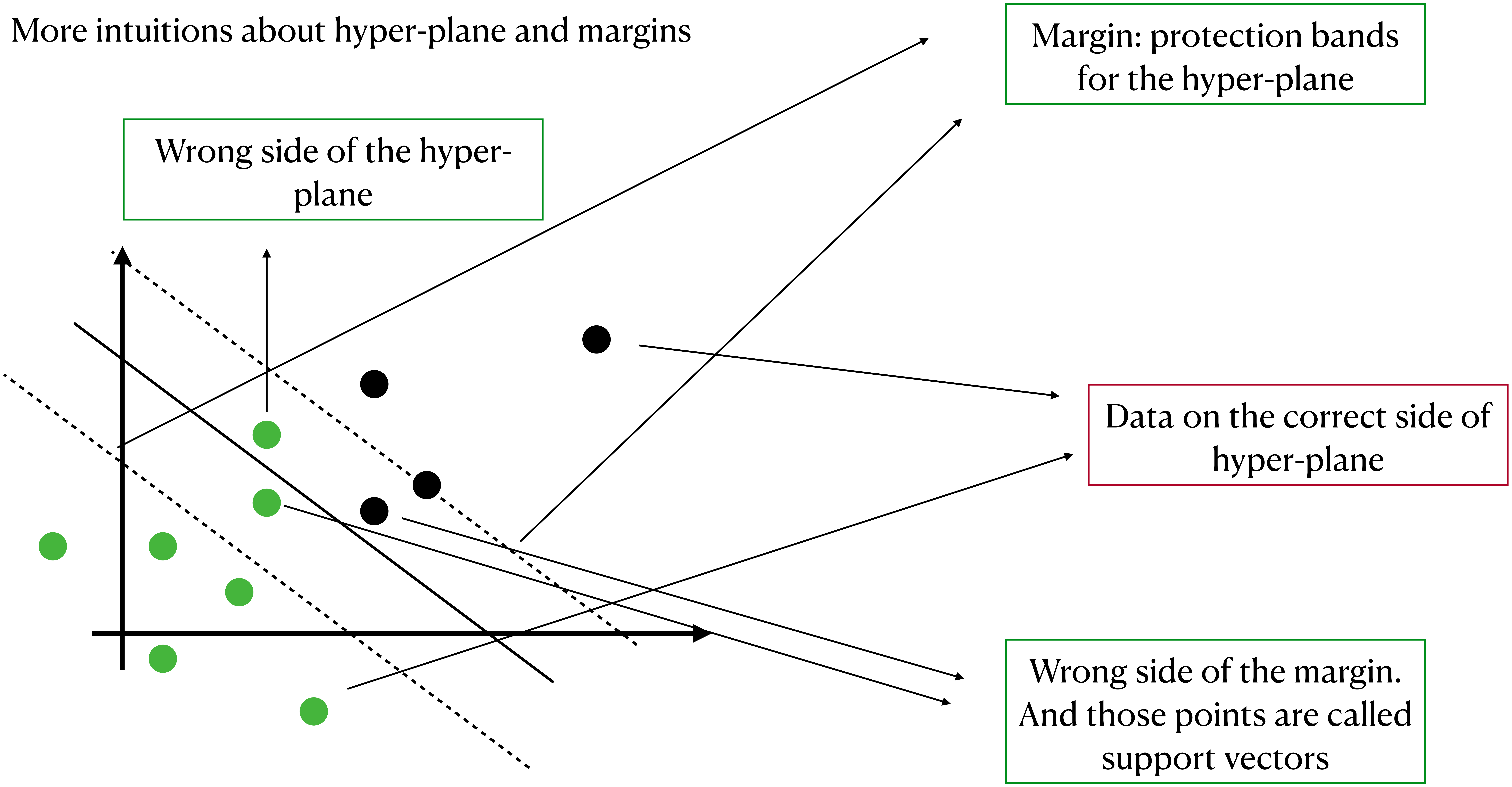
More intuitions about hyper-plane and margins



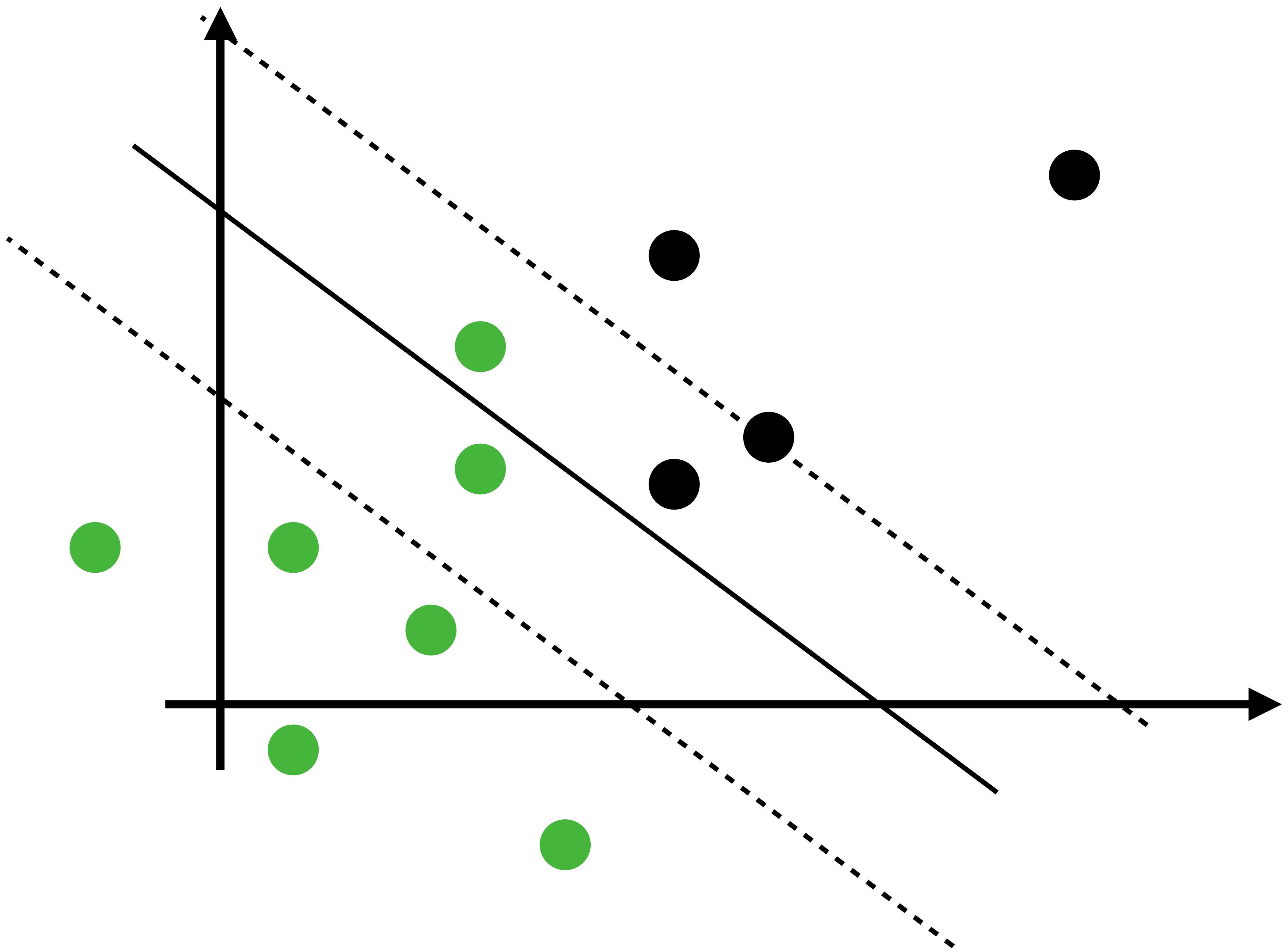
More intuitions about hyper-plane and margins



More intuitions about hyper-plane and margins



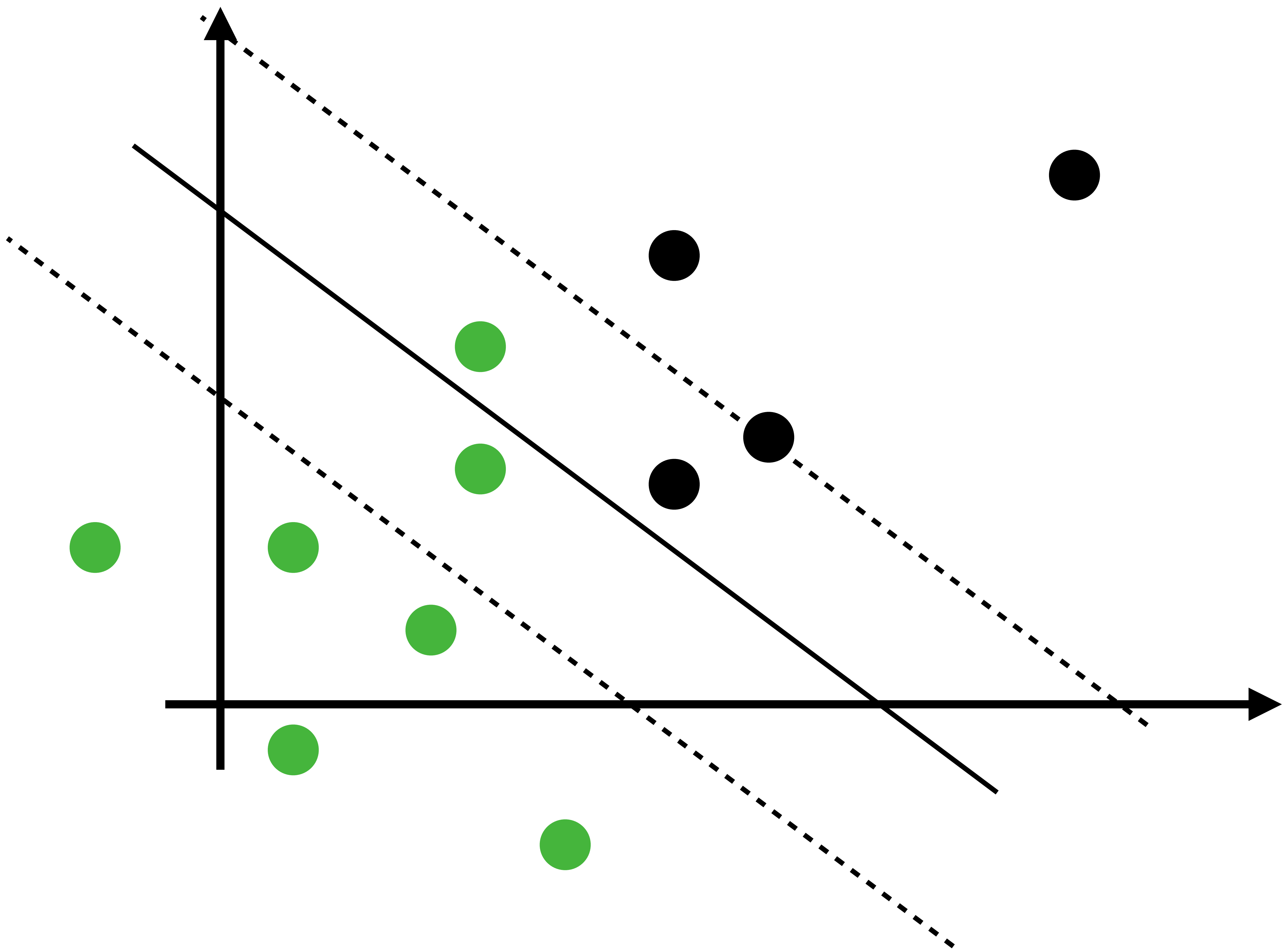
More intuitions about hyper-plane and margins



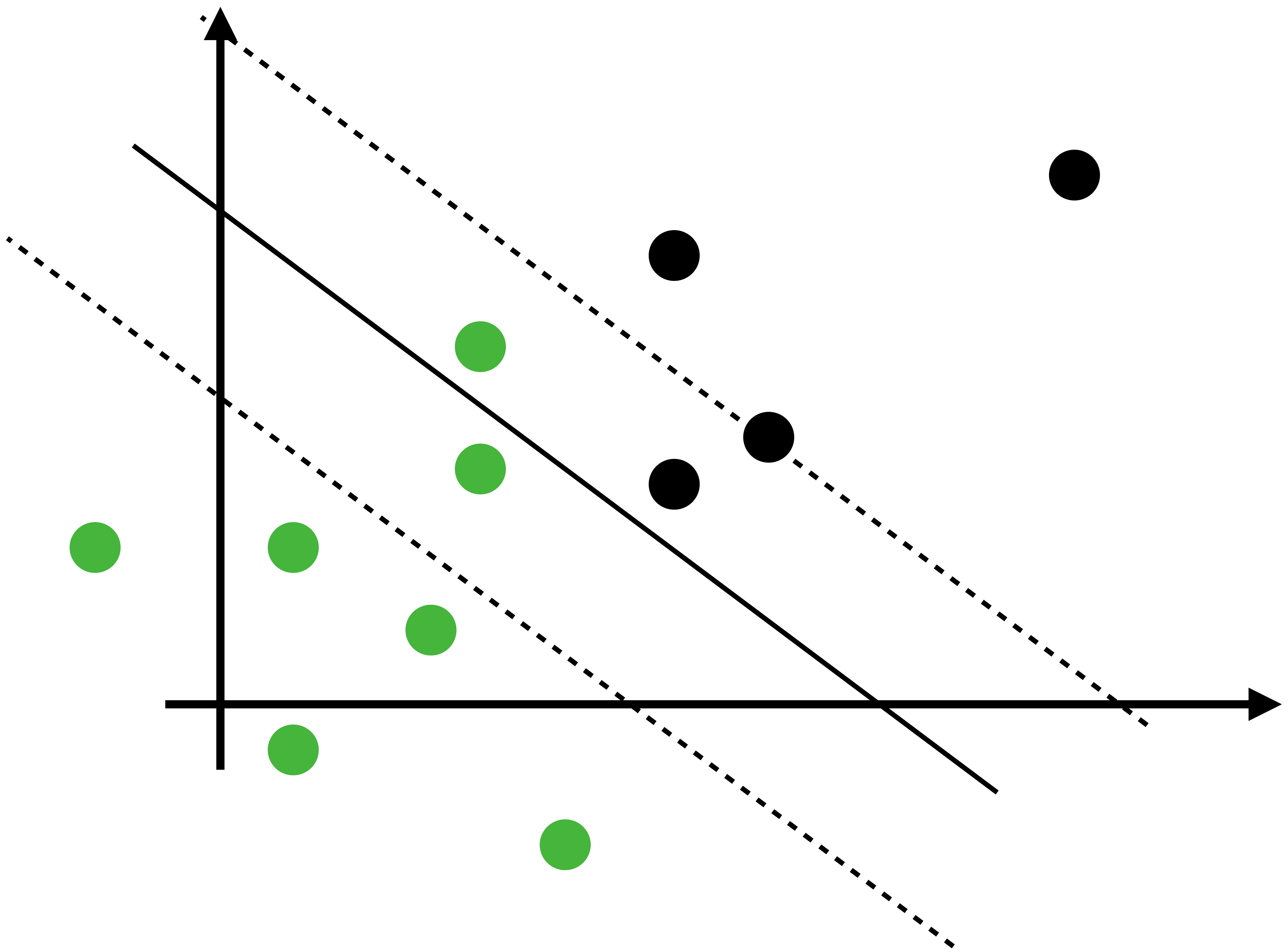
More intuitions about hyper-plane and margins

For data (X,Y) on the correct side of hyper-plane:

$$y(aX_1 + bX_2 + c) > M > 0$$



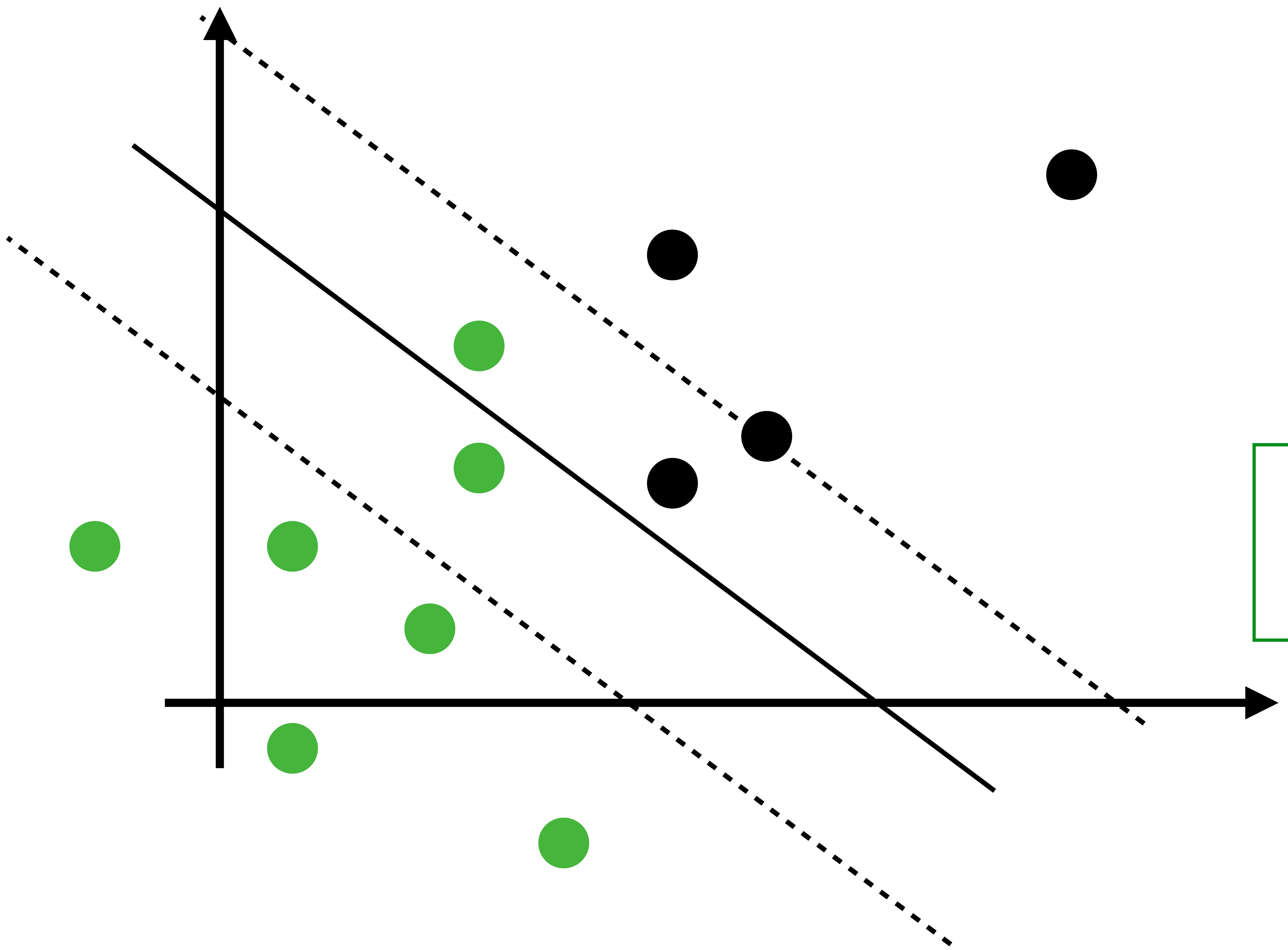
More intuitions about hyper-plane and margins



For data (X,Y) on the correct side of hyper-plane:
$$y(aX_1 + bX_2 + c) > M > 0$$

For data (X,Y) on the margin:
$$y(aX_1 + bX_2 + c) = M > 0$$

More intuitions about hyper-plane and margins



For data (X,Y) on the correct side of hyper-plane:

$$y(aX_1 + bX_2 + c) > M > 0$$

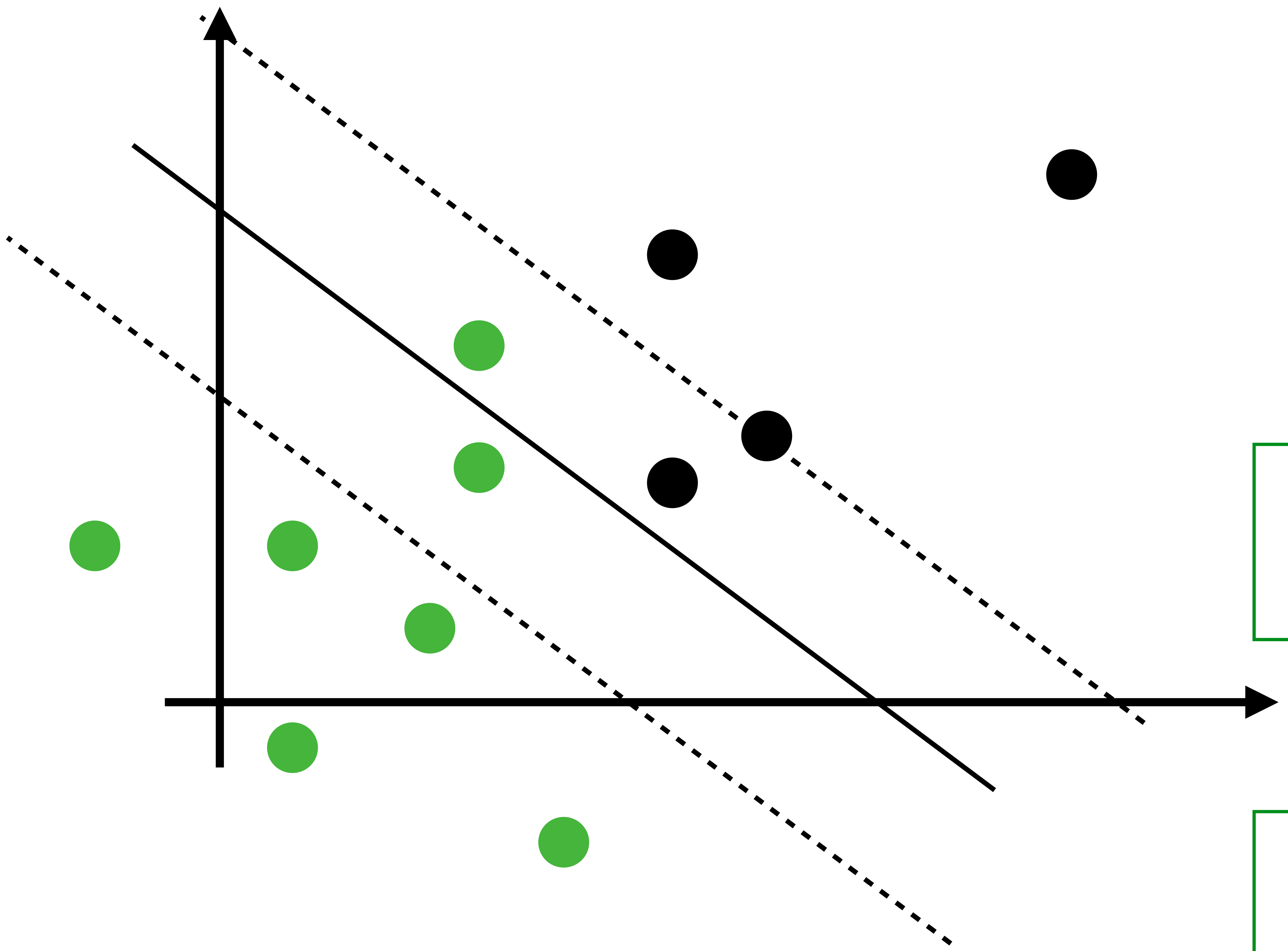
For data (X,Y) on the margin:

$$y(aX_1 + bX_2 + c) = M > 0$$

For data (X,Y) on the wrong side of margin:

$$y(aX_1 + bX_2 + c) = M(1 - \epsilon) > 0$$

More intuitions about hyper-plane and margins



For data (X,Y) on the correct side of hyper-plane:
$$y(aX_1 + bX_2 + c) > M > 0$$

For data (X,Y) on the margin:
$$y(aX_1 + bX_2 + c) = M > 0$$

For data (X,Y) on the wrong side of margin:
$$y(aX_1 + bX_2 + c) = M(1 - \epsilon) > 0$$

For data (X,Y) on the wrong side of hyper-plane:
$$y(aX_1 + bX_2 + c) = M(1 - \epsilon) < 0$$

Support vector classifier

Training algorithm

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M \tag{9.12}$$

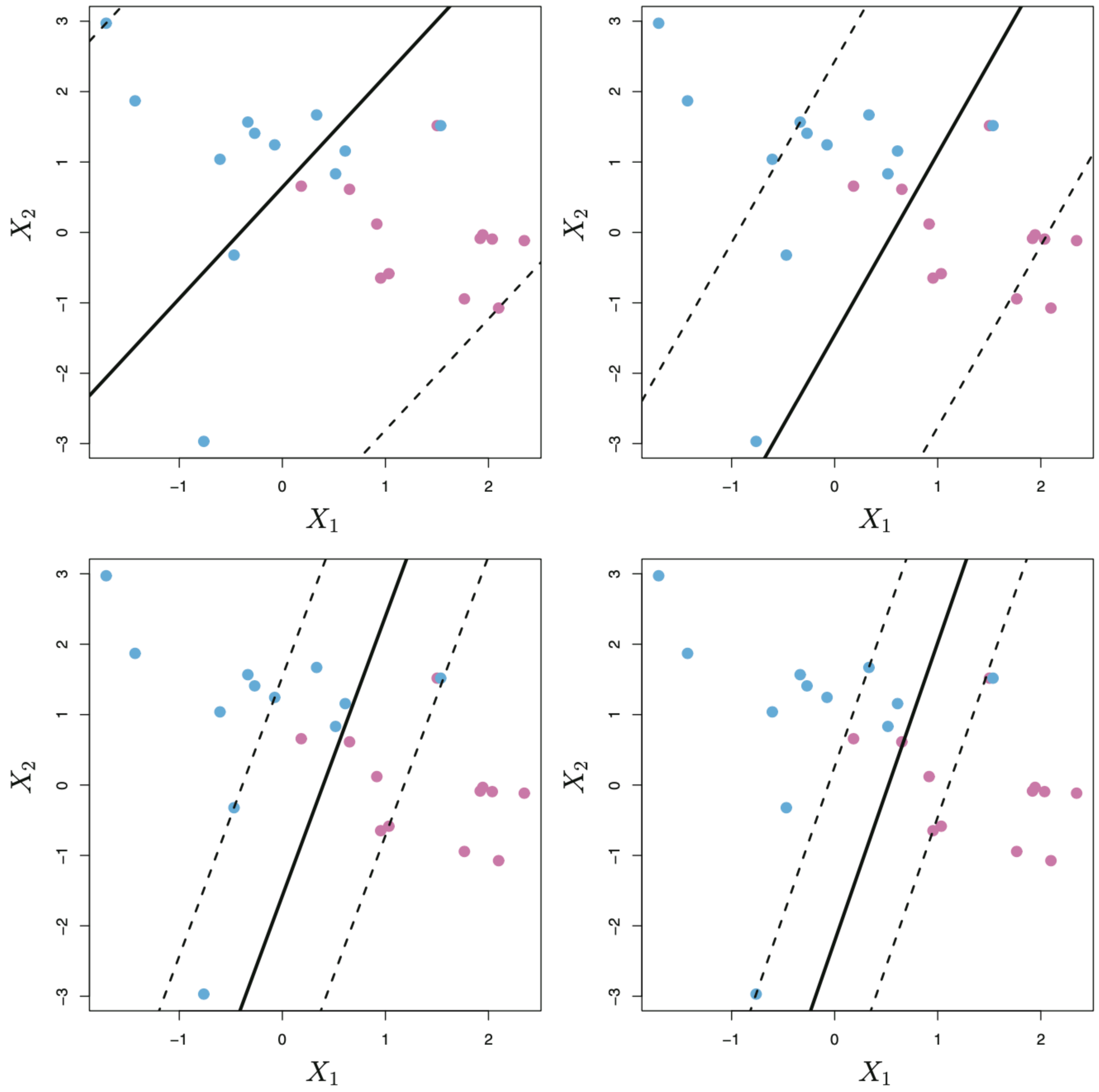
$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1, \tag{9.13}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i), \tag{9.14}$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C, \tag{9.15}$$

To know more, google the key words
linear programming https://en.wikipedia.org/wiki/Linear_programming,
constrained optimization https://en.wikipedia.org/wiki/Constrained_optimization,
Lagrangian multipier https://en.wikipedia.org/wiki/Lagrange_multiplier.

Larger C results in larger margin



Question worth thinking about: SVMs with larger margin (having more support vectors) leads to low-bias and high-variance, or the other way around?

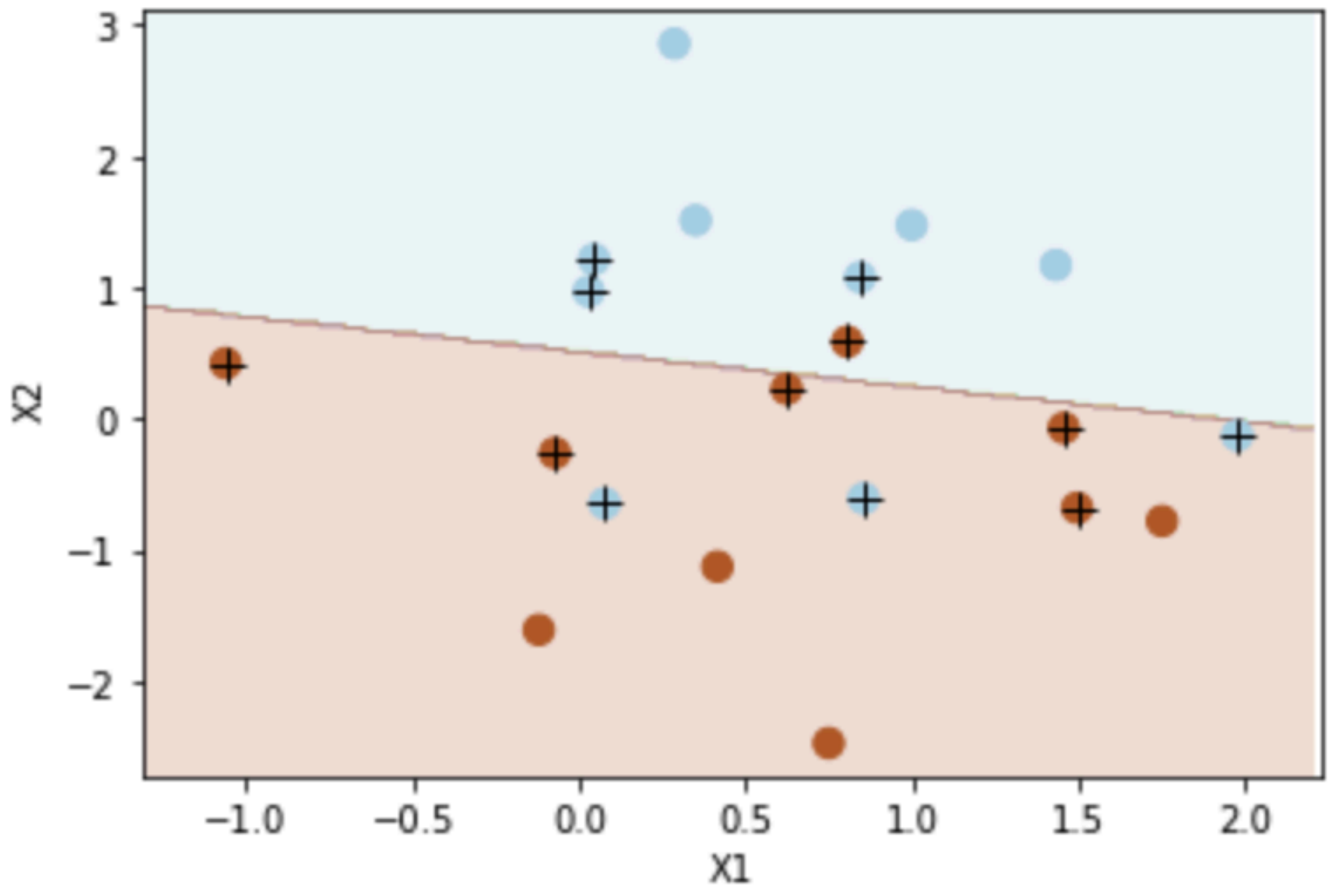
Read p.347 of ISLR

The parameter C in sklearn is inverse!!

X1

In [61]:
```python
# Support Vector Classifier with linear kernel.
svc = SVC(C= 1.0, kernel='linear')
svc.fit(X, y)

plot_svc(svc, X, y)
```



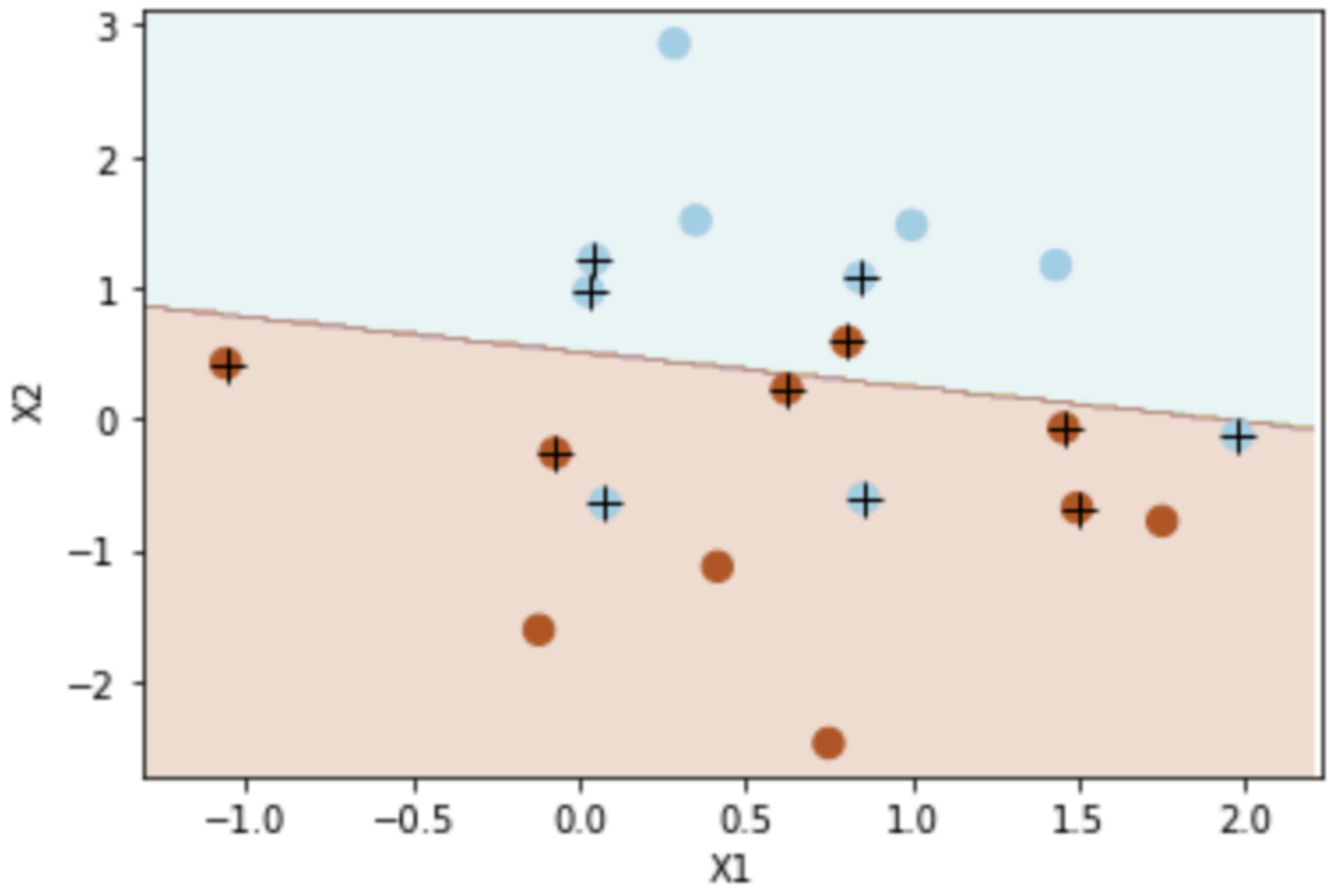

```
Number of support vectors:  12
```

The parameter C in sklearn is inverse!!

X1

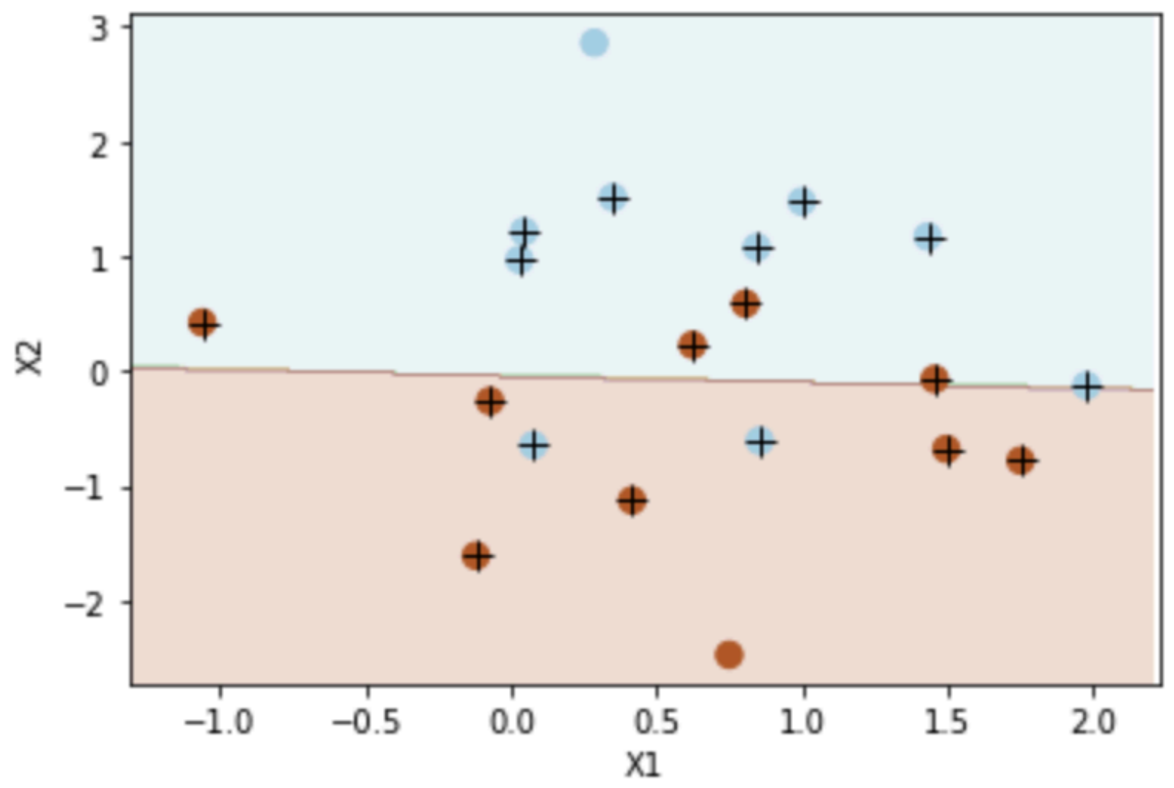
In [61]:
```python
# Support Vector Classifier with linear kernel.
svc = SVC(C= 1.0, kernel='linear')
svc.fit(X, y)

plot_svc(svc, X, y)
```



Number of support vectors: 12

In [62]:
```python
# When using a smaller cost parameter (C=0.1) the margin is wider, resulting in more support vectors.
svc2 = SVC(C=0.1, kernel='linear')
svc2.fit(X, y)
plot_svc(svc2, X, y)
```

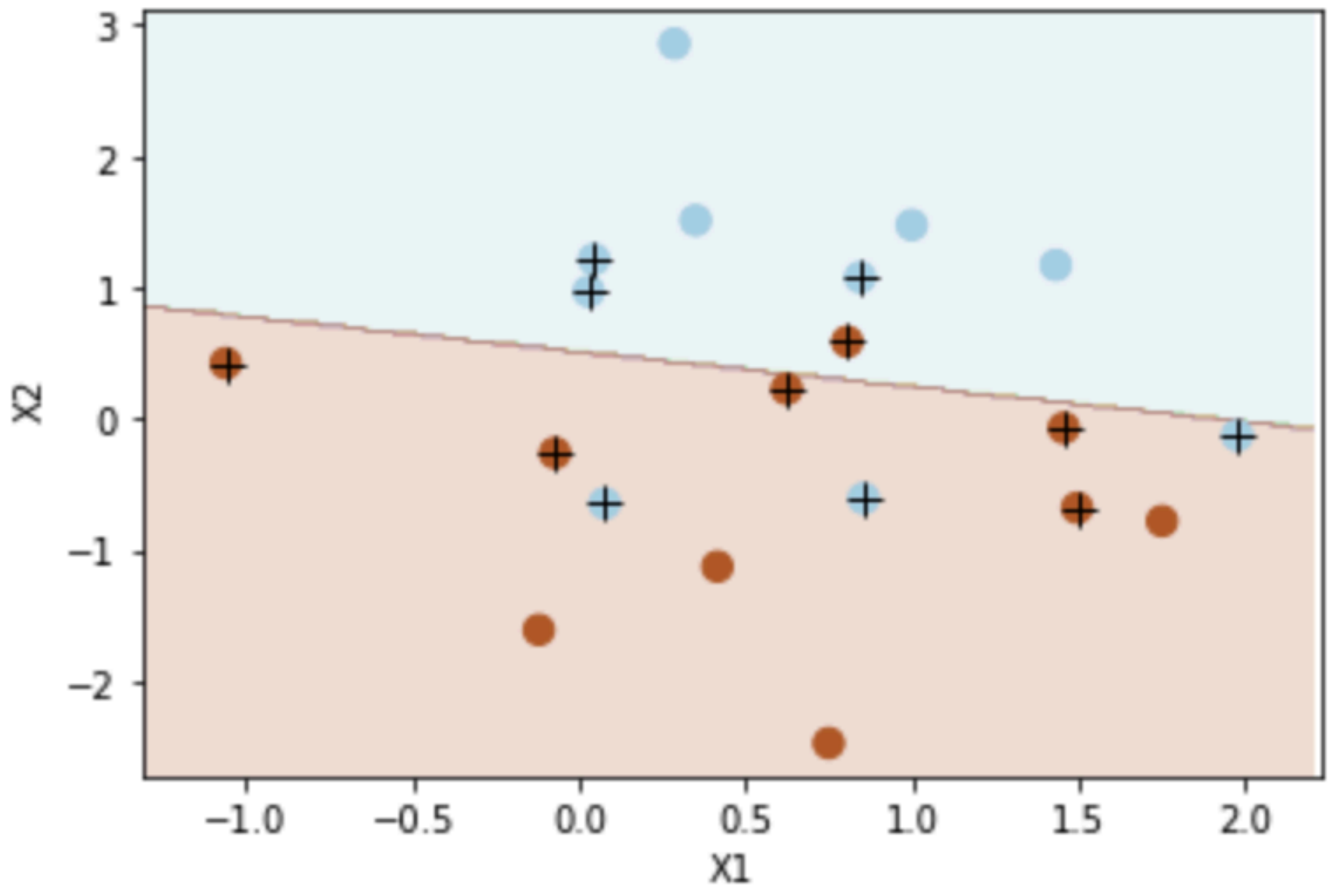


Number of support vectors: 18

The parameter C in sklearn is inverse!!

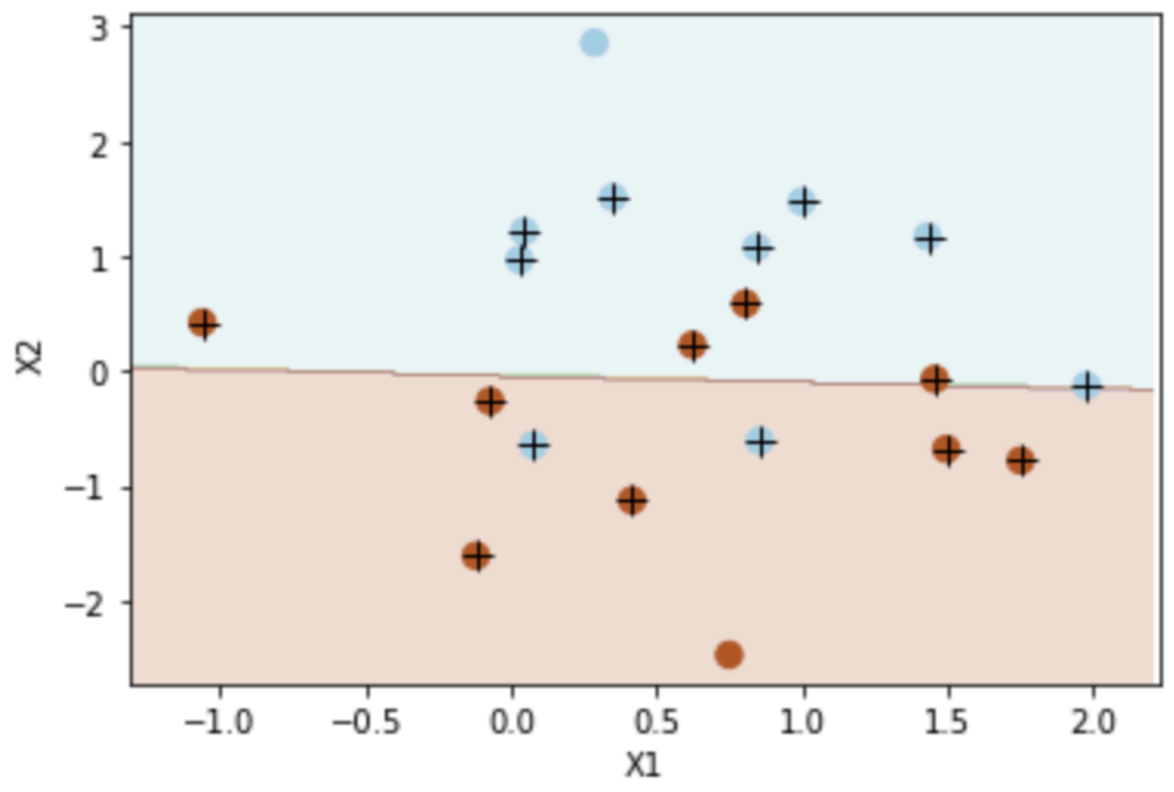
In [61]:
```python
# Support Vector Classifier with linear kernel.
svc = SVC(C= 1.0, kernel='linear')
svc.fit(X, y)

plot_svc(svc, X, y)
```



Number of support vectors: 12

In [62]:
```python
# When using a smaller cost parameter (C=0.1) the margin is wider, resulting in more support vectors.
svc2 = SVC(C=0.1, kernel='linear')
svc2.fit(X, y)
plot_svc(svc2, X, y)
```

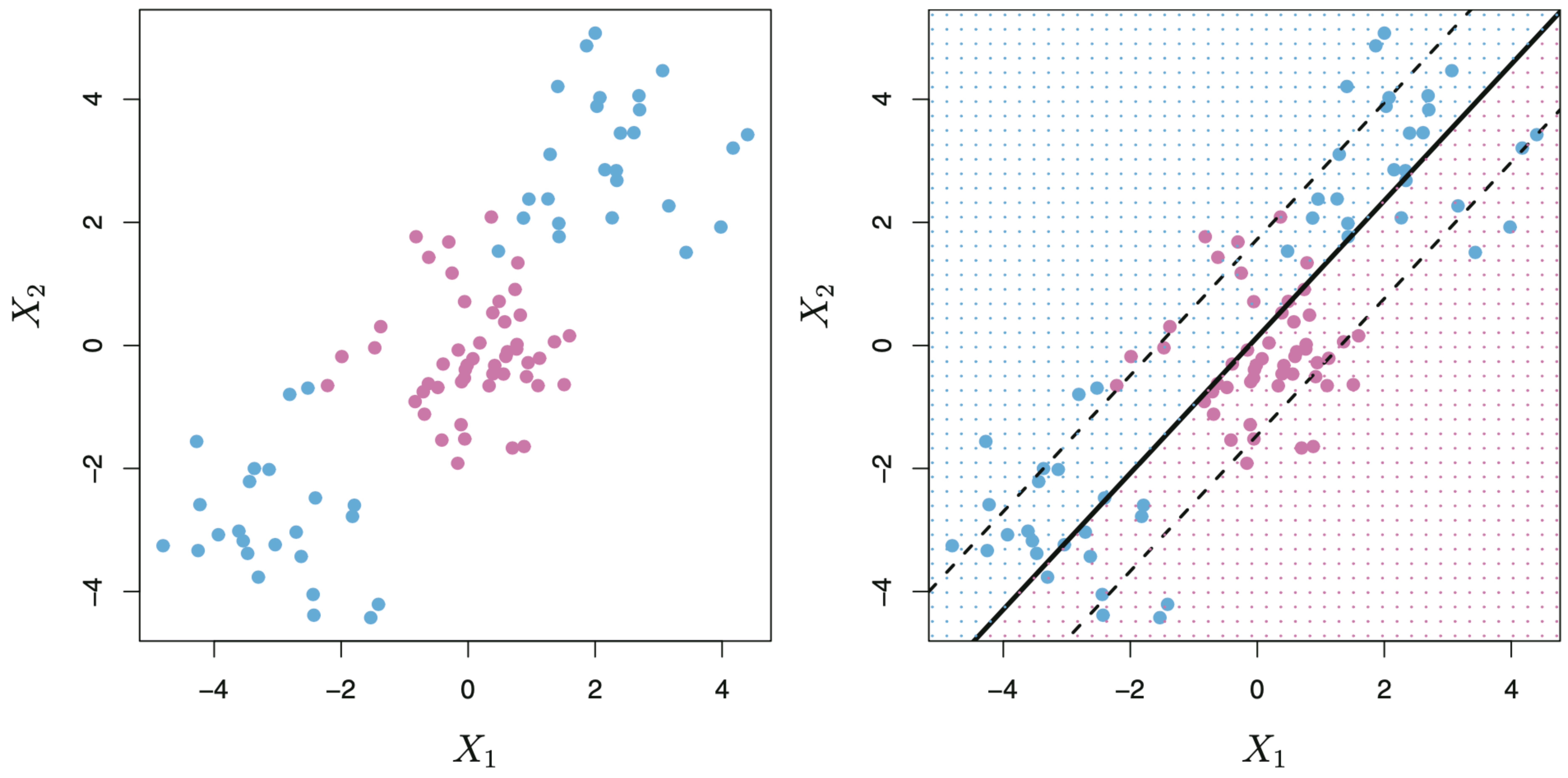


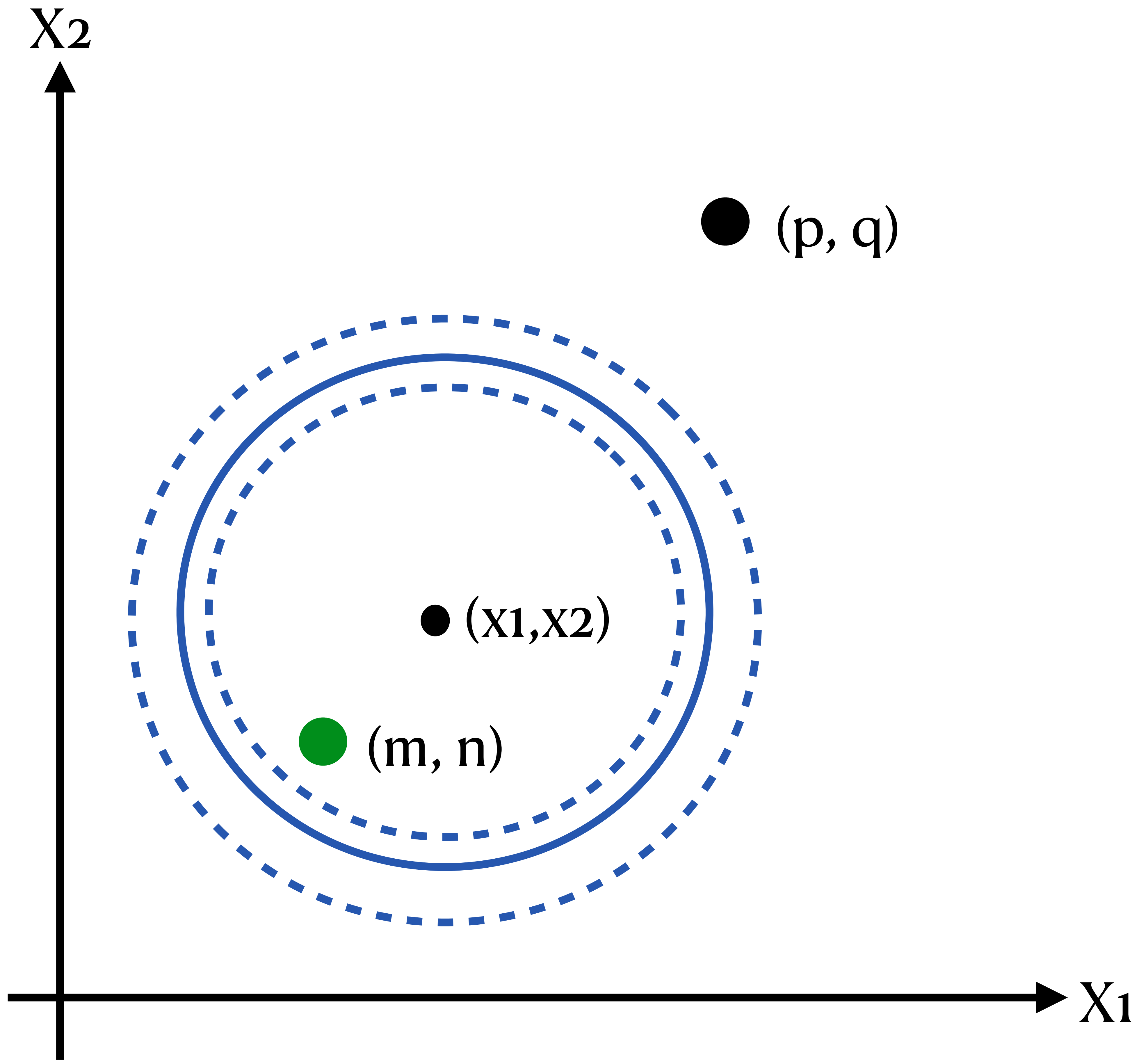
Number of support vectors: 18

Parameters:	**C** : ***float, default=1.0*** Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

SVM — support vector machine

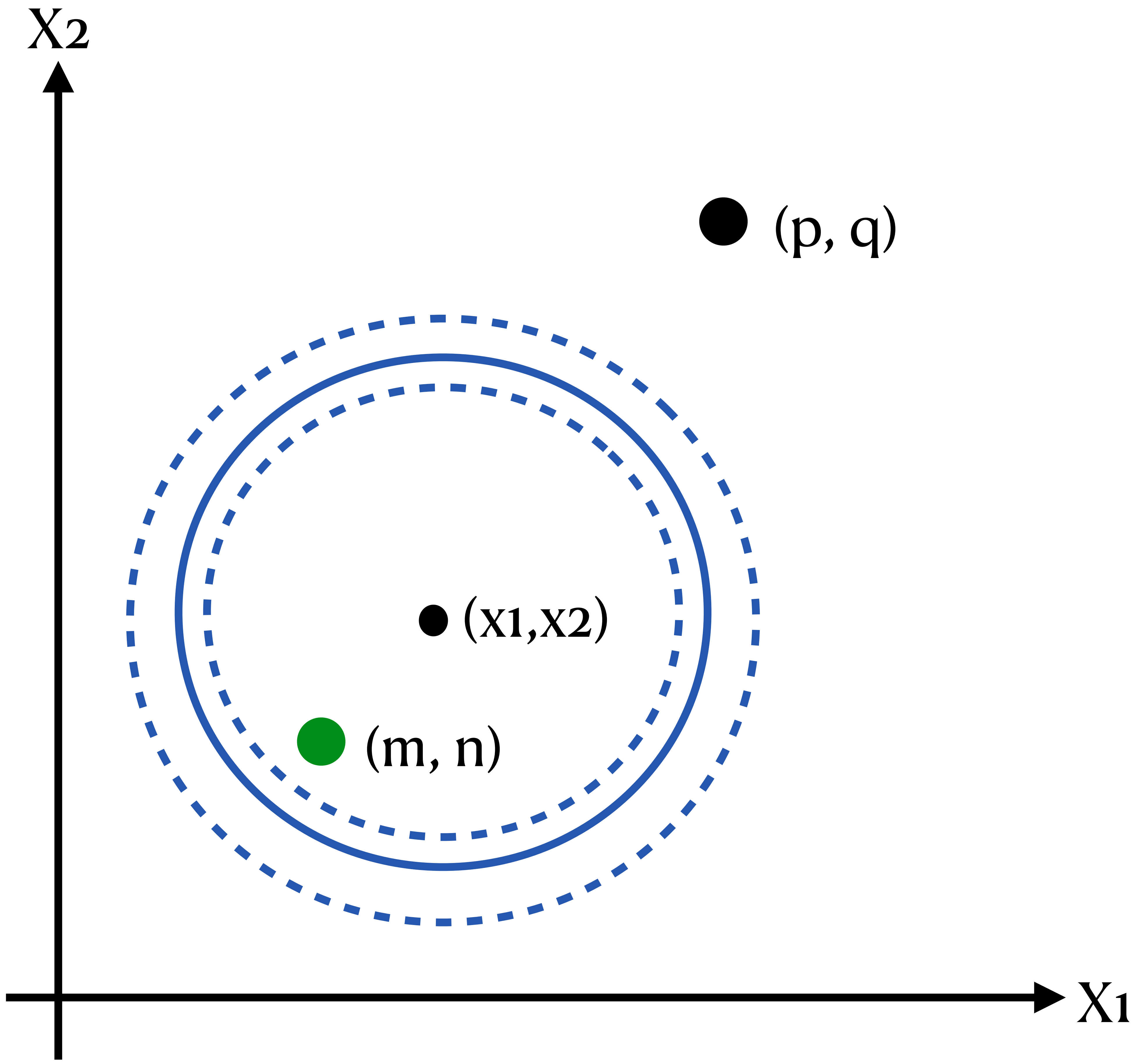
From linear case to nonlinear case





We still try to find the model f(X) so that the data (X,Y) have

$$Yf(X) > 0$$



We still try to find the model f(X) so that the data (X,Y) have

$$Yf(X) > 0$$

In fact, the model $f(X) = (X_1 - x_1)^2 + (X_2 - x_2)^2 - r^2$ can be rewritten as

$$f(X) = r^2 - x_1^2 - x_2^2 + 2x_1X_1 - X_1^2 + 2x_2X_2 - X_2^2$$

Add new features from old:

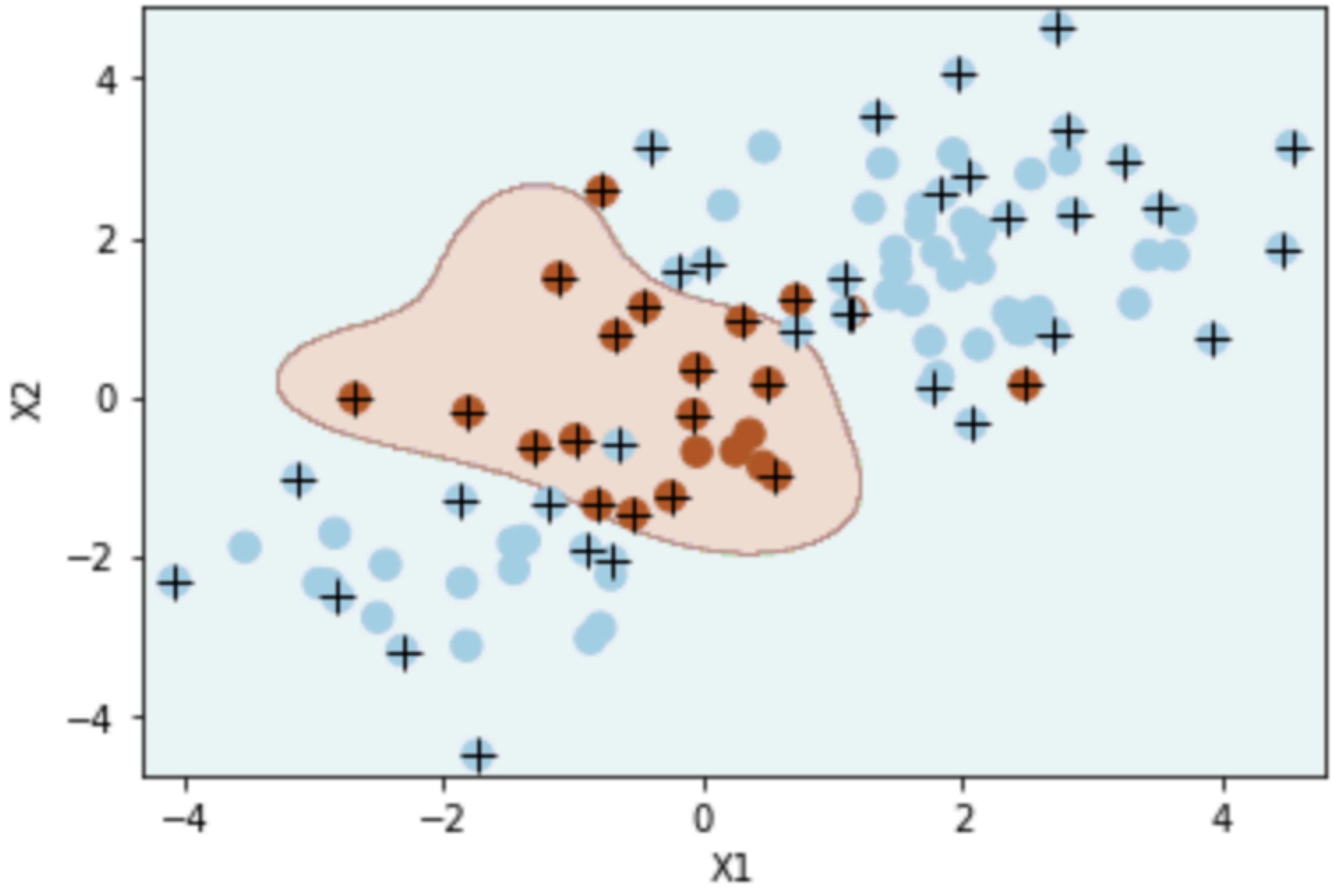
$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2.$$

SVM does similar things:

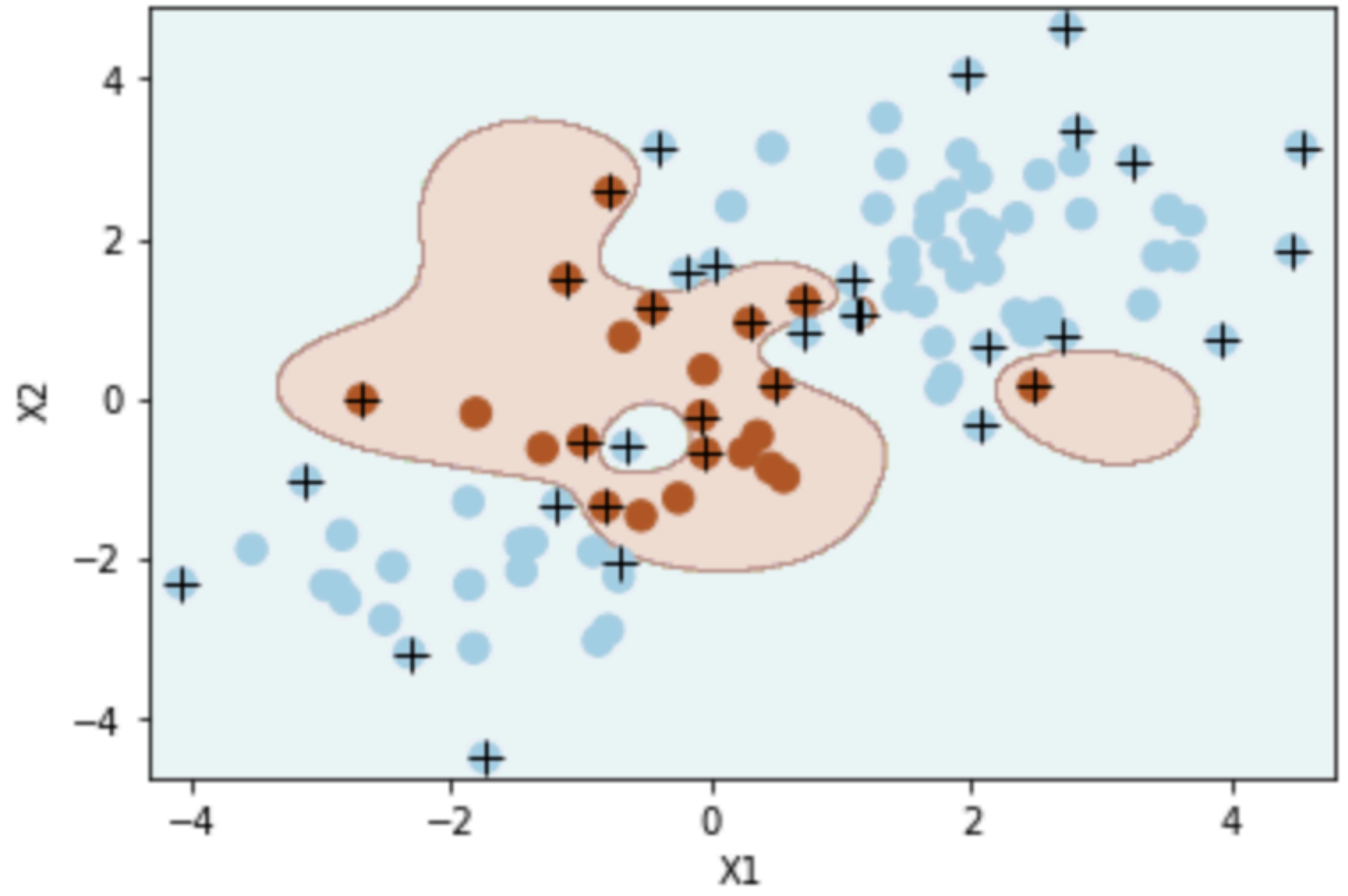
$$\underset{\beta_0,\beta_{11},\beta_{12}\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M \tag{9.16}$$

$$\text{subject to } y_i\left(\beta_0 + \sum_{j=1}^{p}\beta_{j1}x_{ij} + \sum_{j=1}^{p}\beta_{j2}x_{ij}^2\right) \geq M(1-\epsilon_i),$$

$$\sum_{i=1}^{n}\epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^{p}\sum_{k=1}^{2}\beta_{jk}^2 = 1.$$

In [49]:
```
svm = SVC(C=1.0, kernel='rbf', gamma=1)
svm.fit(X_train, y_train)
```
Out[49]:
```
SVC(gamma=1)
```
In [50]:
```
plot_svc(svm, X_train, y_train)
```



```
Number of support vectors:  51
```

In [51]:
```
# Increasing C parameter, allowing more flexibility
svm2 = SVC(C=100, kernel='rbf', gamma=1.0)
svm2.fit(X_train, y_train)
```
Out[51]:
```
SVC(C=100, gamma=1.0)
```
In [52]:
```
plot_svc(svm2, X_train, y_train)
```



```
Number of support vectors:  36
```