A Machine Learning Primer

Antony Ross

Silicon Valley Code Camp
October 8, 2017

Linear Regression

Logistic Regression

Support Vector Machine

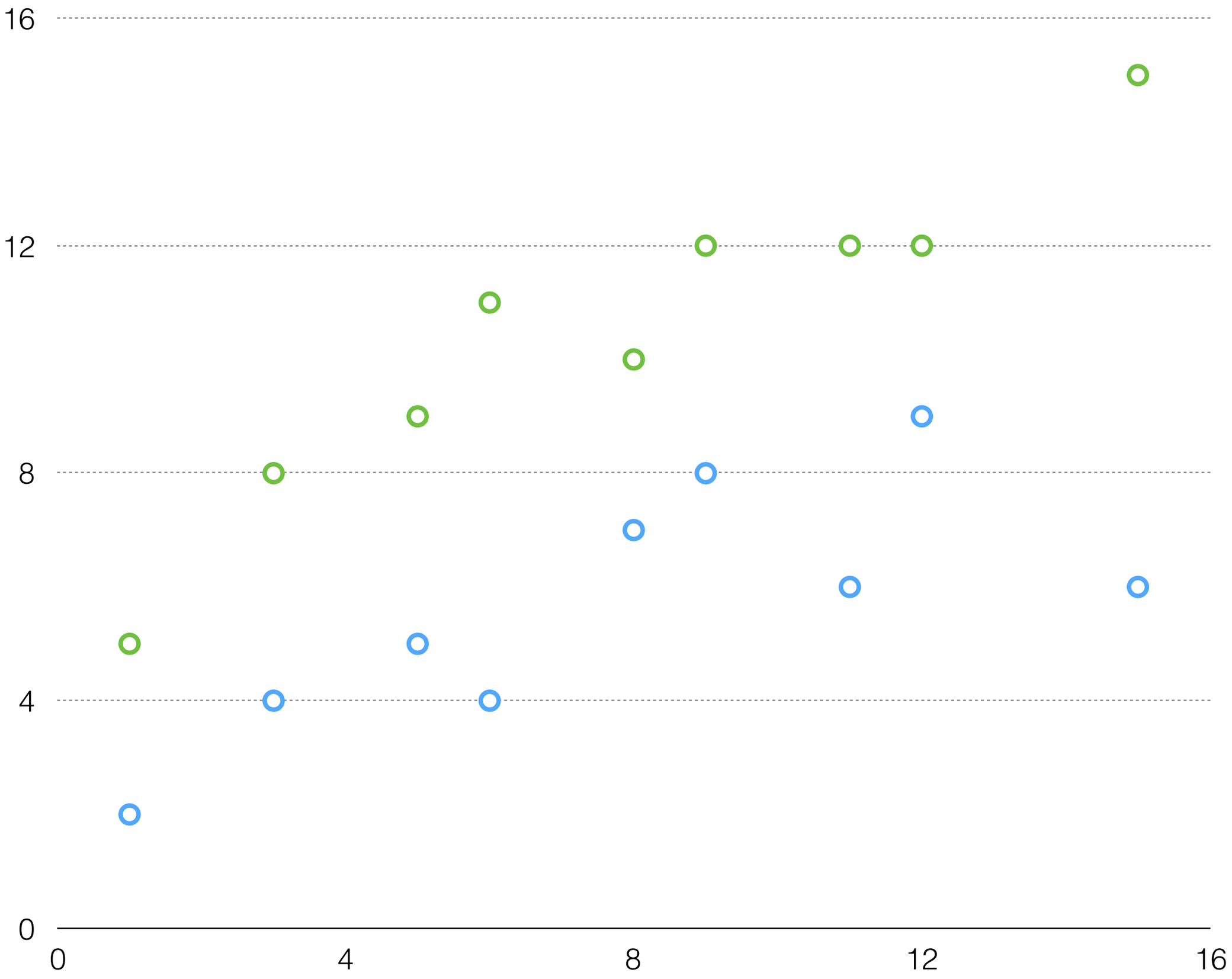
Decision Tree

K-Nearest Neighbor

K-Means

Neural Network

$$\hat{f}(X)$$


16
12
8
4
0
0
4
8
12
16

The Prediction

output

input

$$\hat{y} \equiv \hat{f}(X)$$

Linear Regression

equation of a line

$$y = mx + b$$

Linear Regression

$$y = \beta_0 + \beta_1 x$$

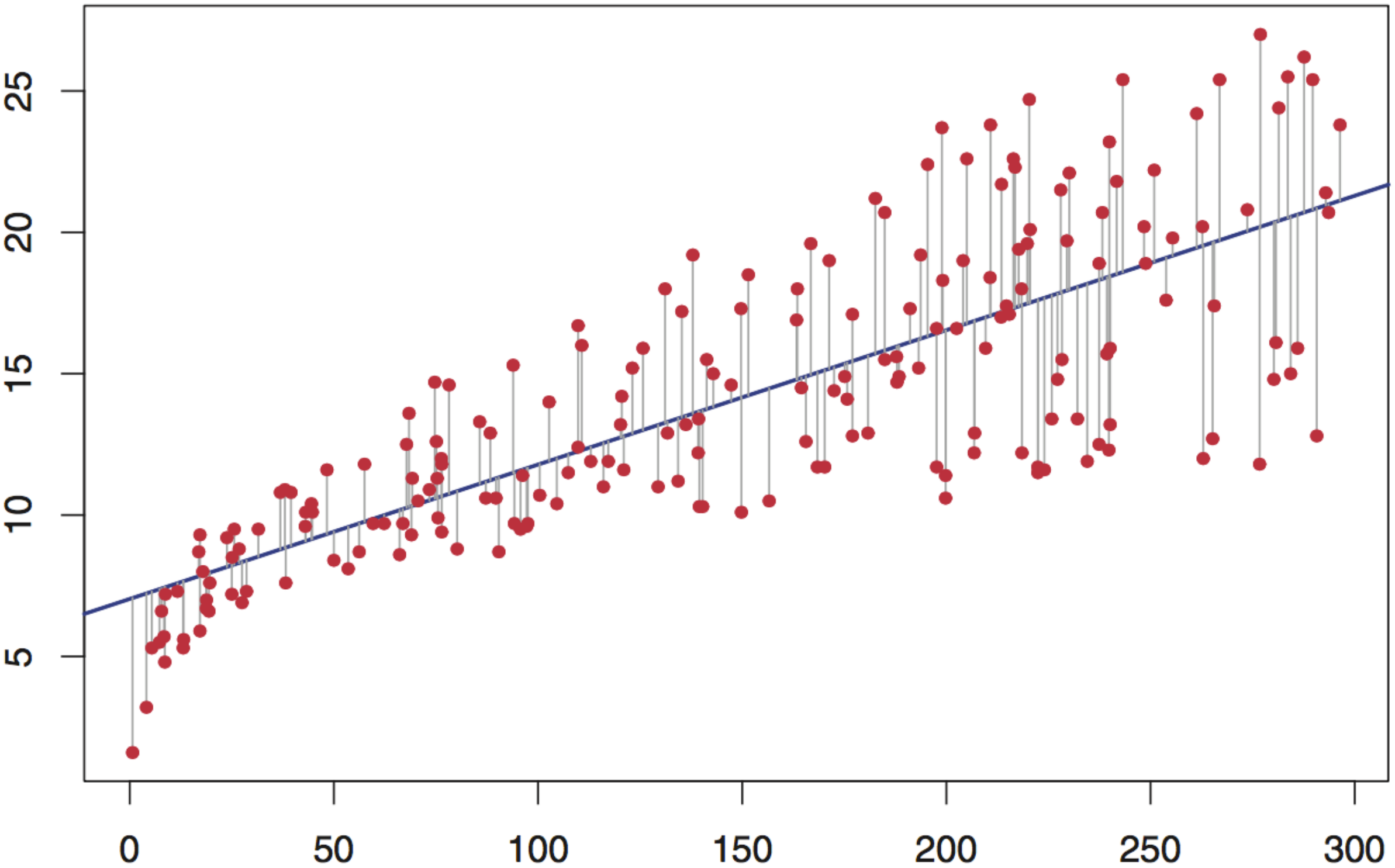
equation of a line

$$y = mx + b$$

Simple Linear Regression

Input	learned coefficients (weights)	Output
x_1	β_0, β_1	y

$$y = \beta_0 + \beta_1 x_1$$


25
20
15
10
5
0
50
100
150
200
250
300

Lines of Code

Gender	Pair Programming	Social Accounts	LOC
Female	3	2	309
Male	2	3	276
Male	2	1	353
Male	4	2	285
Female	4	2	220
Female	2	2	347
Male	3	1	244
Female	2	3	312
Female	4	2	239
Male	4	2	307

Lines of Code

features (X) →

Gender	Pair Programming	Social Accounts	LOC
Female	3	2	309
Male	2	3	276
Male	2	1	353
Male	4	2	285
Female	4	2	220
Female	2	2	347
Male	3	1	244
Female	2	3	312
Female	4	2	239
Male	4	2	307

Lines of Code

Gender	Pair Programming	Social Accounts	LOC
Female	3	2	309
Male	2	3	276
Male	2	1	353
Male	4	2	285
Female	4	2	220
Female	2	2	347
Male	3	1	244
Female	2	3	312
Female	4	2	239
Male	4	2	307

output/target (y)

Lines of Code

sample →

Gender	Pair Programming	Social Accounts	LOC
Female	3	2	309
Male	2	3	276
Male	2	1	353
Male	4	2	285
Female	4	2	220
Female	2	2	347
Male	3	1	244
Female	2	3	312
Female	4	2	239
Male	4	2	307

Lines of Code

Gender	Pair Programming	Social Accounts	LOC
Female	3	2	309
Male	2	3	276
Male	2	1	353
Male	4	2	285
Female	4	2	220
Female	2	2	347
Male	3	1	244
Female	2	3	312
Female	4	2	239
Male	4	2	307

Lines of Code

Gender	Pair Programming	Social Accounts	LOC
1	3	2	309
0	2	3	276
0	2	1	353
0	4	2	285
1	4	2	220
1	2	2	347
0	3	1	244
1	2	3	312
1	4	2	239
0	4	2	307

$$
\mathbf{X} = \begin{bmatrix}
1 & 3 & 2 \\
0 & 2 & 3 \\
0 & 2 & 1 \\
0 & 4 & 2 \\
1 & 4 & 2 \\
1 & 2 & 2 \\
0 & 3 & 1 \\
1 & 2 & 3 \\
1 & 4 & 2 \\
0 & 4 & 2
\end{bmatrix}
\quad
\mathbf{y} = \begin{bmatrix}
309 \\
276 \\
353 \\
285 \\
220 \\
347 \\
244 \\
312 \\
239 \\
307
\end{bmatrix}
$$

Multiple Linear Regression

$$LOC = 227.63 + 9.51x_1 + 2.7x_2 - 3.08x_3$$

x_1 = hour pair programming

x_2 = gender (m = 0; f = 1)

x_3 = number of social accounts

Multiple Linear Regression

$$\text{Apps sold} = 46.55 + 35.03x_1 + 27.11x_2 + 52.48x_3$$

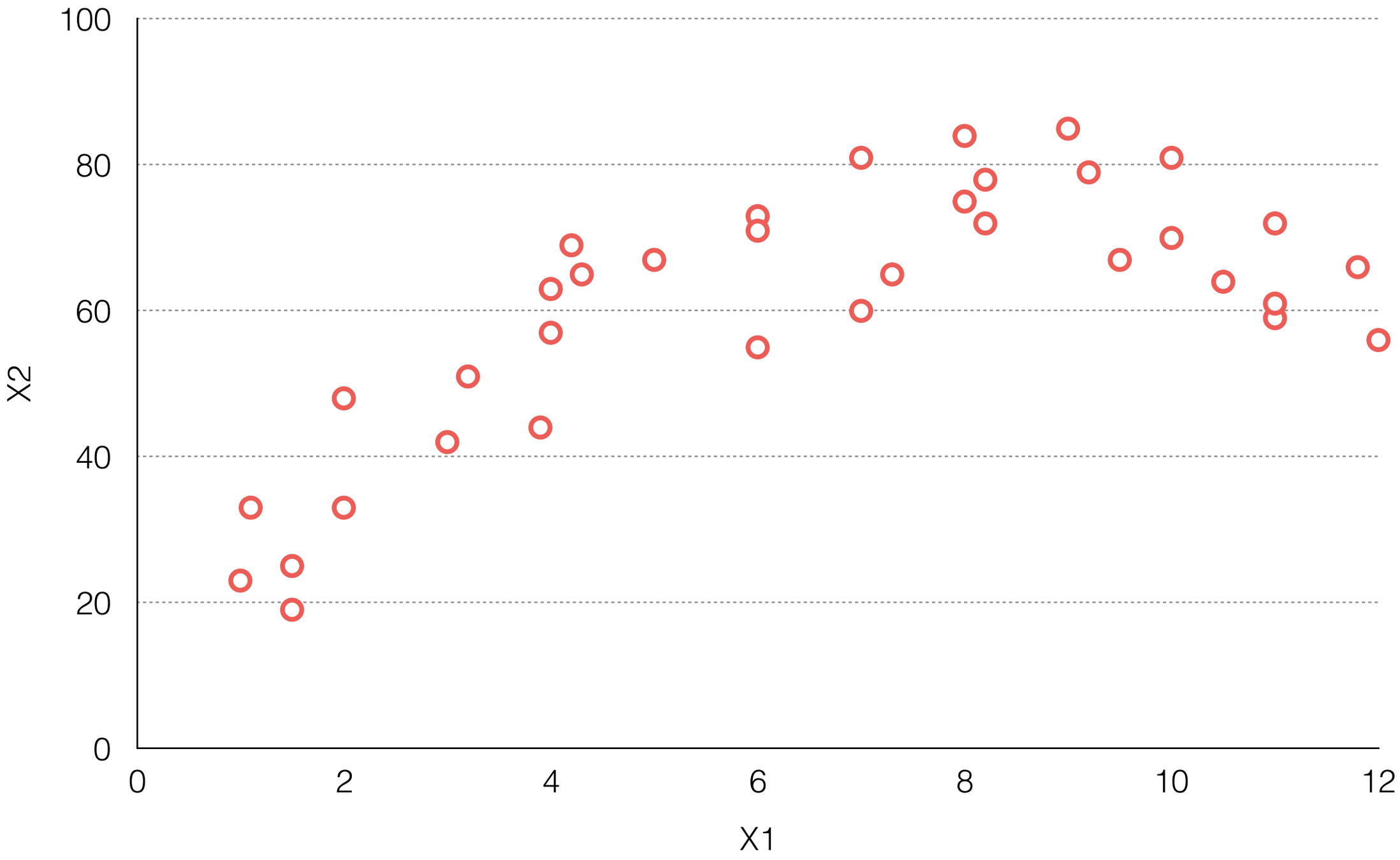
x_1 = per $100 advertising

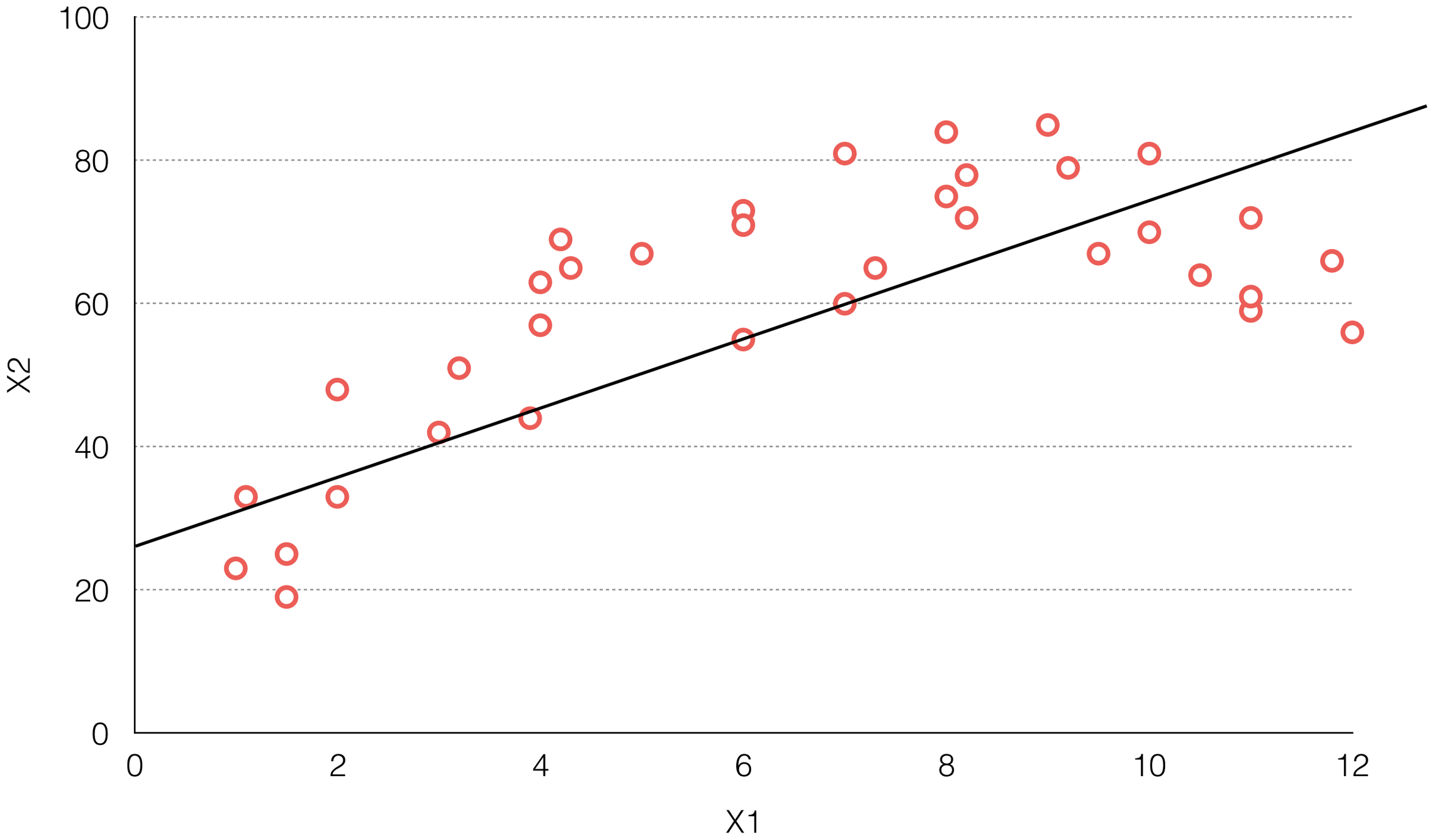
x_2 = per public talk

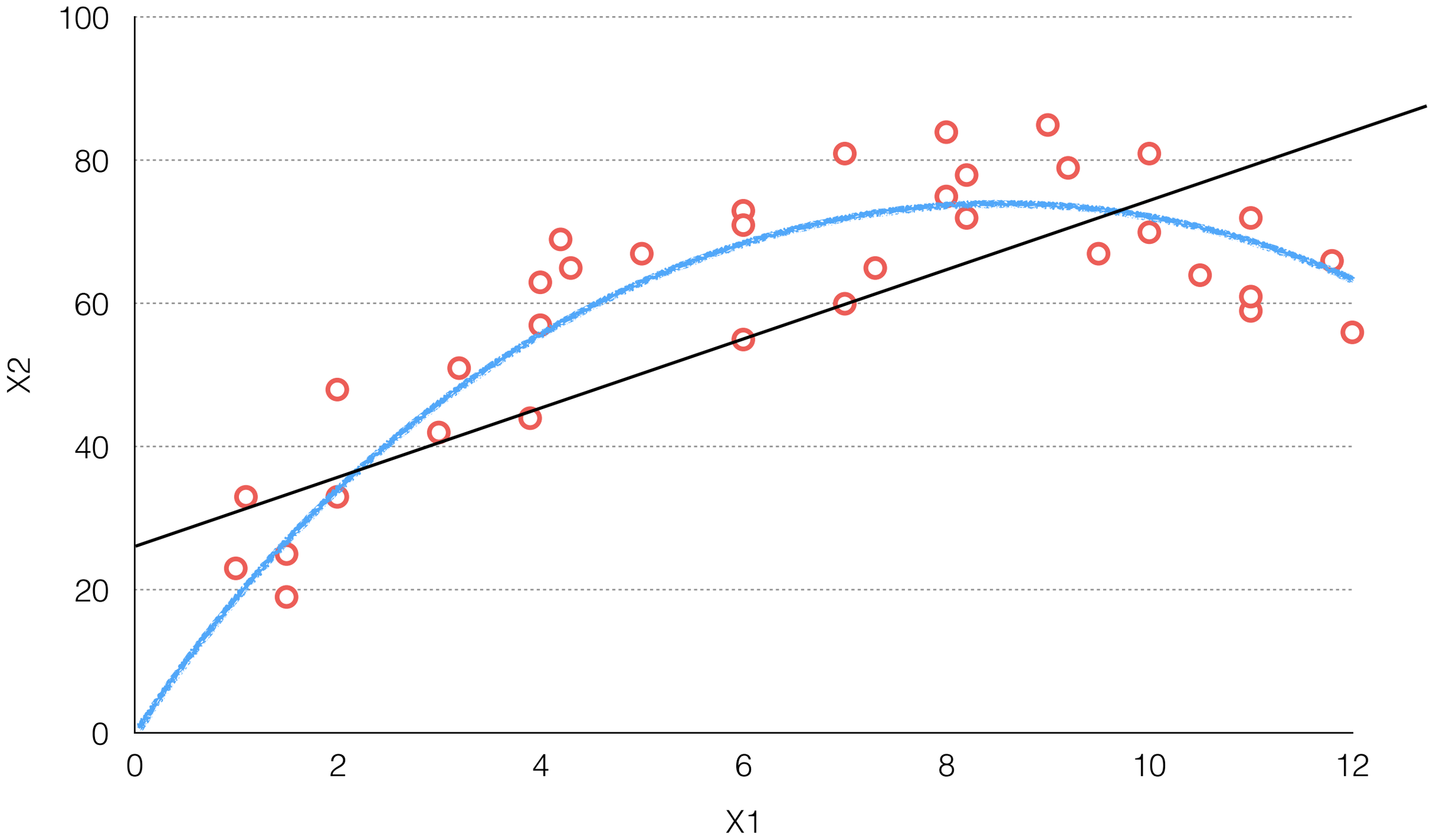
x_3 = per targeted podcast

Polynomial Regression

$$f(X) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \beta_4 x_1^2 + \beta_5 x_2^2$$


100
80
60
40
20
0
X2
0
2
4
6
8
10
12
X1


100
80
60
40
20
0
X2
0
2
4
6
8
10
12
X1


100
80
60
40
20
0
X2
0
2
4
6
8
10
12
X1

CLASSIFICATION

Logistic Regression

Logistic Regression

$$\beta_0 + \beta_1 x + \beta_2 x + \beta_3 x$$

positive class = 1

negative class = 0

if probability >= 0.5: predict 1

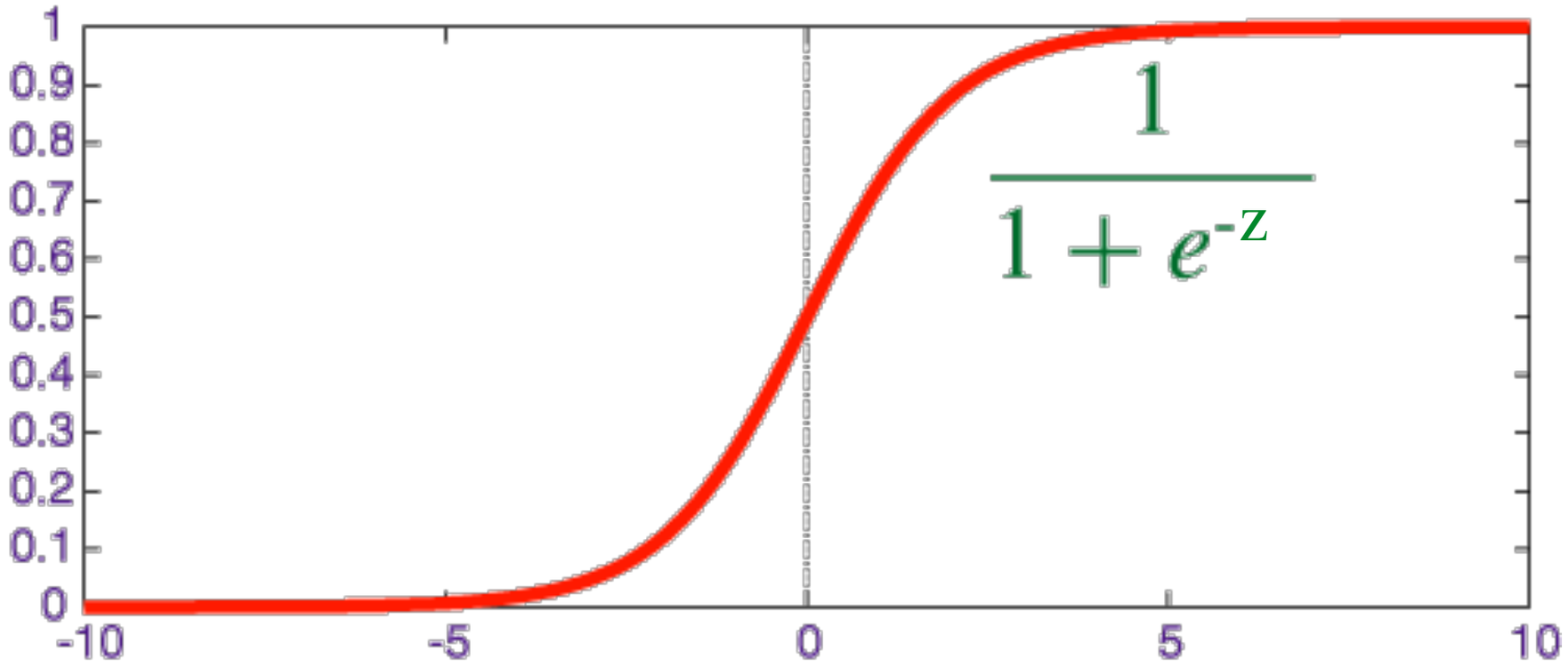
else: predict 0

Logistic Regression

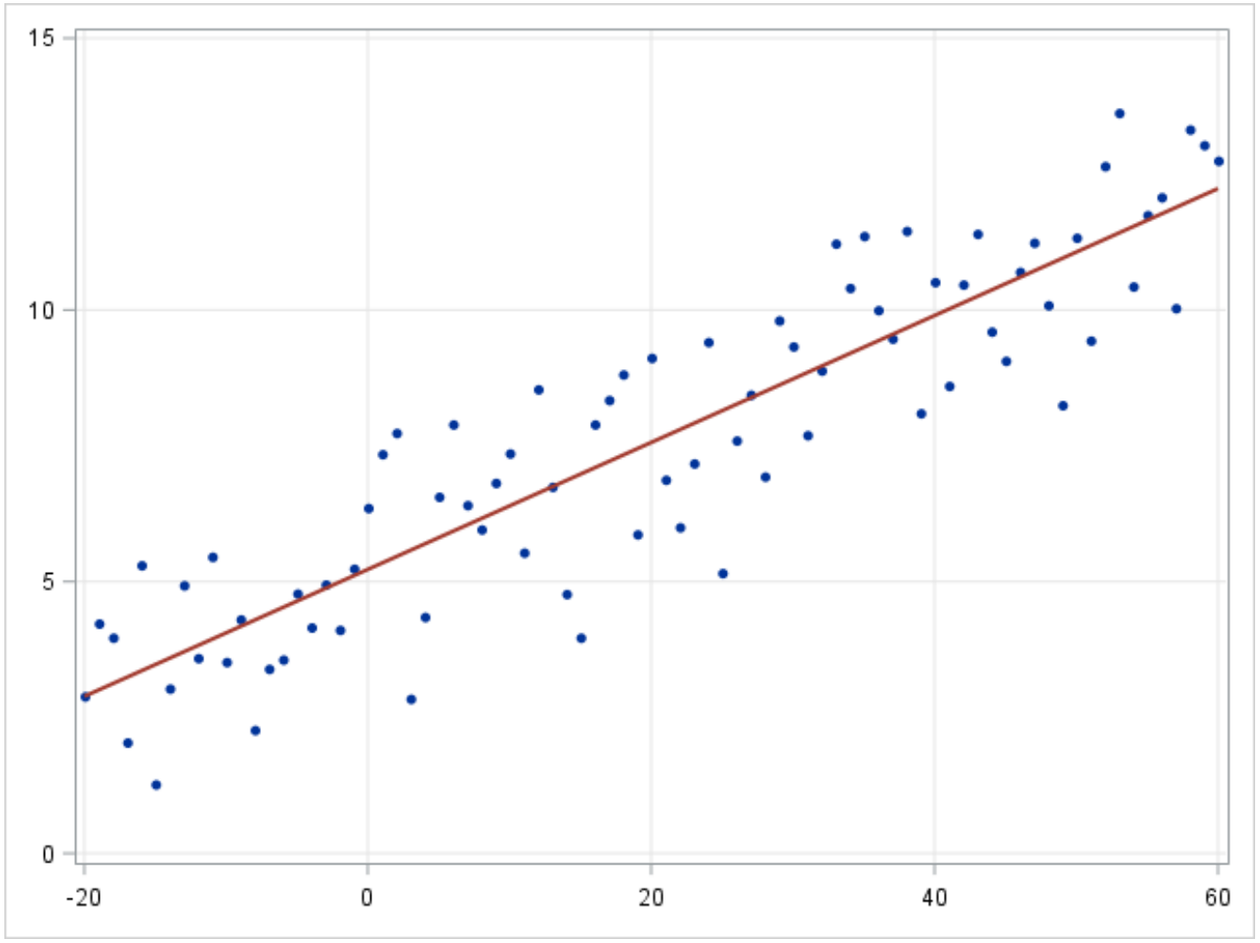
intermediate step

$$z = \beta_0 + \beta_1 x + \beta_2 x + \beta_3 x$$

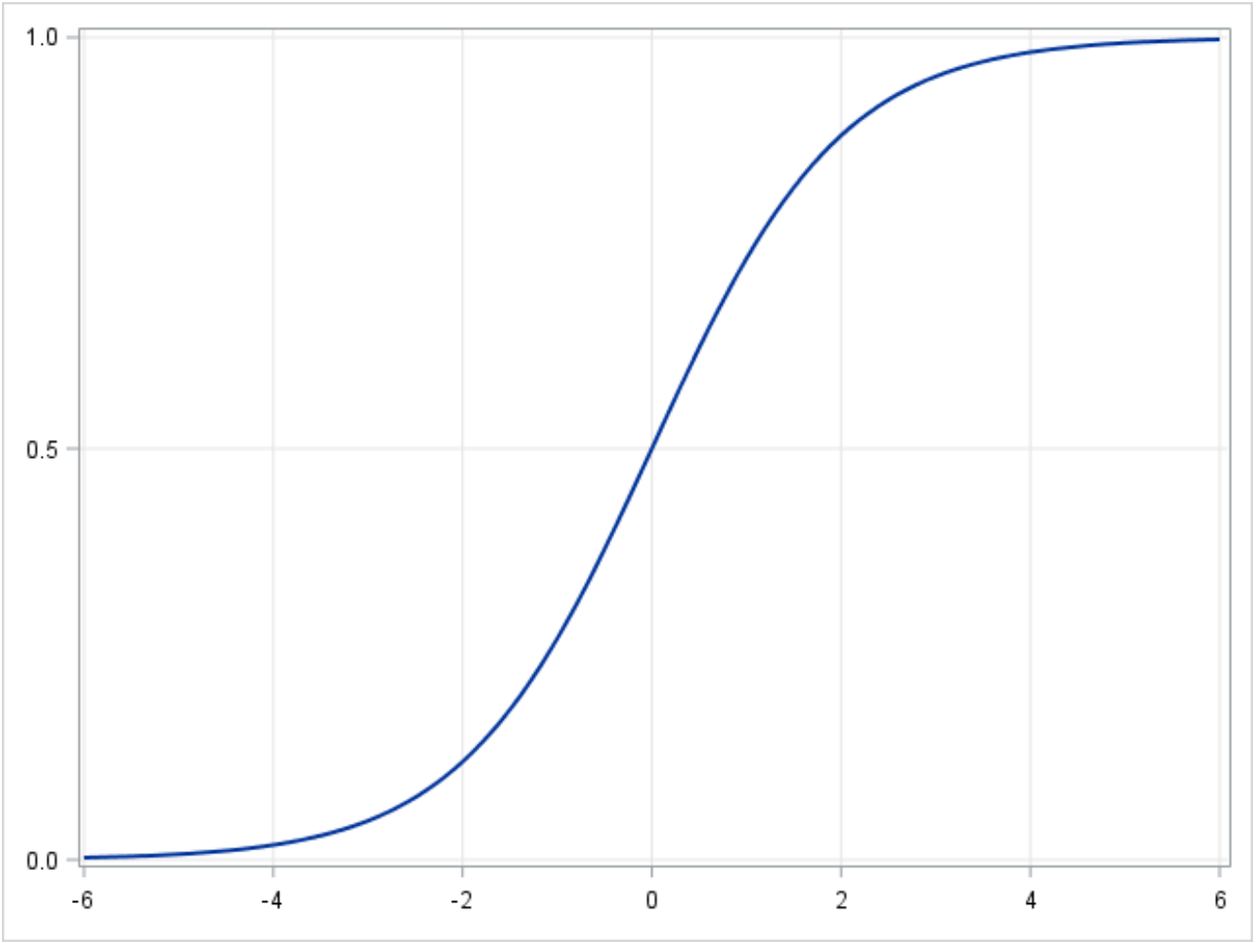
Logistic (Sigmoid) Function



Linear Regression

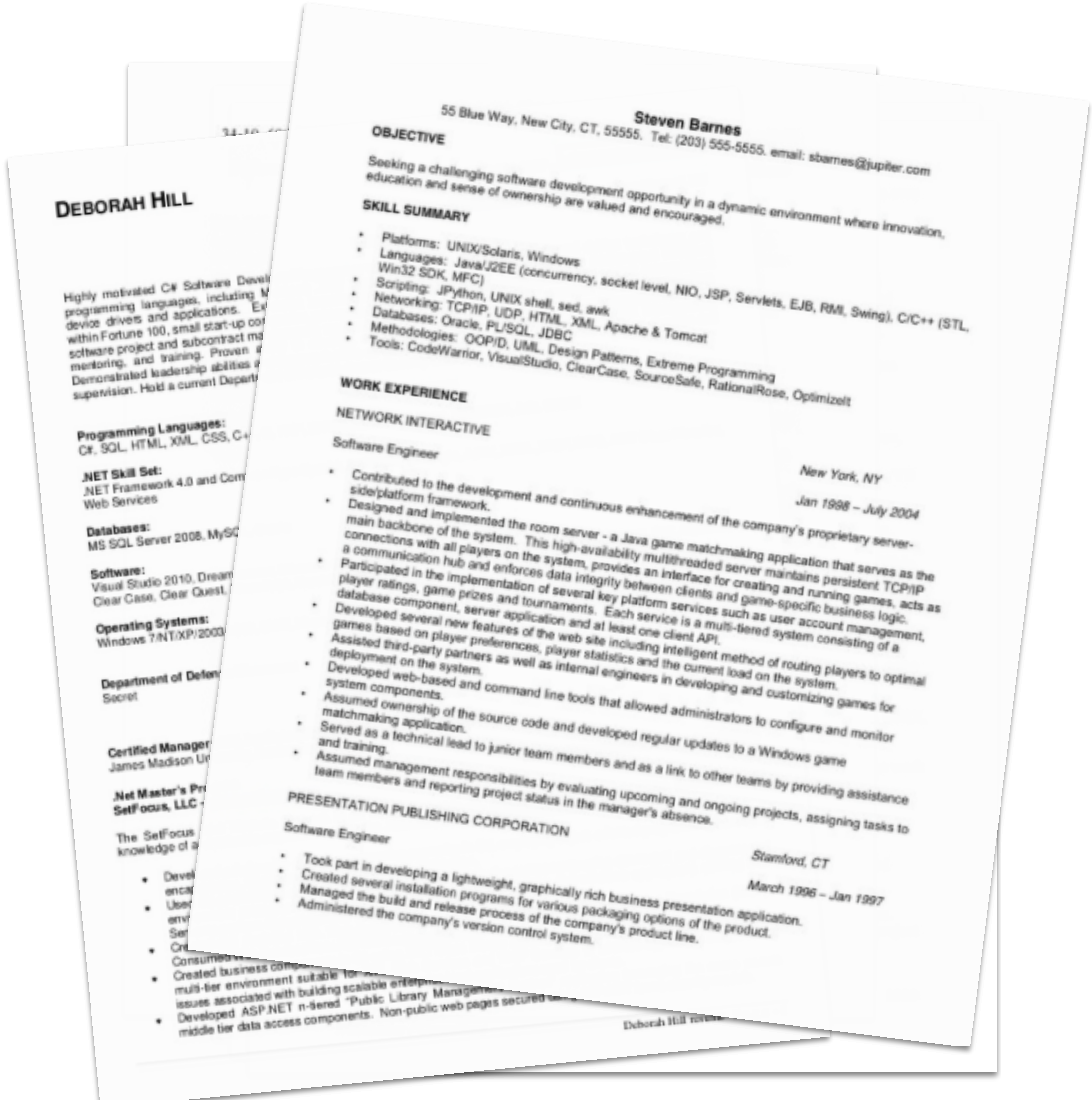


Logistic Regression



Logistic Regression

$$\hat{f}(X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2)}}$$



Steven Barnes

55 Blue Way, New City, CT, 55555. Tel: (203) 555-5555. email: sbarnes@jupiter.com

OBJECTIVE

Seeking a challenging software development opportunity in a dynamic environment where innovation, education and sense of ownership are valued and encouraged.

SKILL SUMMARY

- Platforms: UNIX/Solaris, Windows
- Languages: Java/J2EE (concurrency, socket level, NIO, JSP, Servlets, EJB, RMI, Swing), C/C++ (STL, Win32 SDK, MFC)
- Scripting: JPython, UNIX shell, sed, awk
- Networking: TCP/IP, UDP, HTML, XML, Apache & Tomcat
- Databases: Oracle, PL/SQL, JDBC
- Methodologies: OOP/D, UML, Design Patterns, Extreme Programming
- Tools: CodeWarrior, VisualStudio, ClearCase, SourceSafe, RationalRose, OptimizeIt

WORK EXPERIENCE

NETWORK INTERACTIVE

Software Engineer — *New York, NY* — *Jan 1998 – July 2004*

- Contributed to the development and continuous enhancement of the company's proprietary server-side/platform framework.
- Designed and implemented the room server - a Java game matchmaking application that serves as the main backbone of the system. This high-availability multithreaded server maintains persistent TCP/IP connections with all players on the system, provides an interface for creating and running games, acts as a communication hub and enforces data integrity between clients and game-specific business logic.
- Participated in the implementation of several key platform services such as user account management, player ratings, game prizes and tournaments. Each service is a multi-tiered system consisting of a database component, server application and at least one client API.
- Developed several new features of the web site including intelligent method of routing players to optimal games based on player preferences, player statistics and the current load on the system.
- Assisted third-party partners as well as internal engineers in developing and customizing games for deployment on the system.
- Developed web-based and command line tools that allowed administrators to configure and monitor system components.
- Assumed ownership of the source code and developed regular updates to a Windows game matchmaking application.
- Served as a technical lead to junior team members and as a link to other teams by providing assistance and training.
- Assumed management responsibilities by evaluating upcoming and ongoing projects, assigning tasks to team members and reporting project status in the manager's absence.

PRESENTATION PUBLISHING CORPORATION

Software Engineer — *Stamford, CT* — *March 1996 – Jan 1997*

- Took part in developing a lightweight, graphically rich business presentation application.
- Created several installation programs for various packaging options of the product.
- Managed the build and release process of the company's product line.
- Administered the company's version control system.

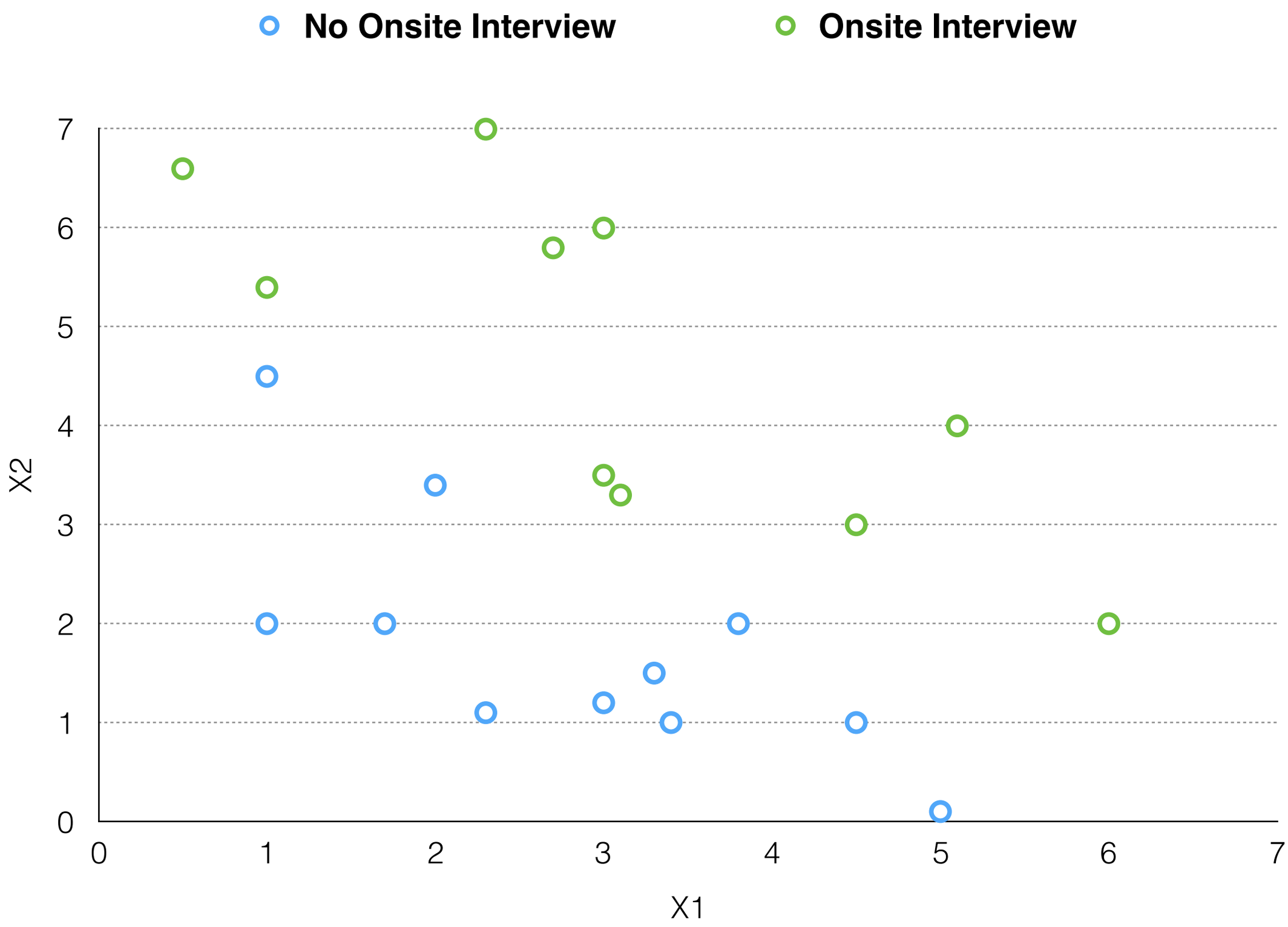
Gender	Years Exp.	Interview Source	Phone Screen	On-site Interview
Female	3	2	9	Yes
Male	2	3	7.5	No
Male	2	1	7	No
Male	4	2	8.5	Yes
Female	4	2	9.5	Yes
Female	2	2	6.5	No
Male	3	1	8	No
Female	2	3	8	No
Female	4	2	9	Yes
Male	4	2	7	Yes

Gender	Years Exp.	Interview Source	Phone Screen	On-site Interview
1	3	2	9	Yes
0	2	3	7.5	No
0	2	1	7	No
0	4	2	8.5	Yes
1	4	2	9.5	Yes
1	2	2	6.5	No
0	3	1	8	No
1	2	3	8	No
1	4	2	9	Yes
0	4	2	7	Yes

One-Hot Encoding

Gender	Years Exp.	Source 1	Source 2	Source 3	Phone Screen	On-site Interview
1	3	0	1	0	9	Yes
0	2	0	0	1	7.5	No
0	2	1	0	0	7	No
0	4	0	1	0	8.5	Yes
1	4	0	1	0	9.5	Yes
1	2	0	1	0	6.5	No
0	3	1	0	0	8	No
1	2	0	0	1	8	No
1	4	0	1	0	9	Yes
0	4	0	1	0	7	Yes

Gender	Years Exp.	Source 1	Source 2	Source 3	Phone Screen	On-site Interview
1	3	0	1	0	9	1
0	2	0	0	1	7.5	0
0	2	1	0	0	7	0
0	4	0	1	0	8.5	1
1	4	0	1	0	9.5	1
1	2	0	1	0	6.5	0
0	3	1	0	0	8	0
1	2	0	0	1	8	0
1	4	0	1	0	9	1
0	4	0	1	0	7	1


No Onsite Interview
Onsite Interview
7
6
5
4
3
2
1
0
X2
0
1
2
3
4
5
6
7
X1

Decision Boundary

$$\text{On-site Interview} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

Decision Boundary

$$\text{On-site Interview} = -6 + 1x_1 + 1x_2$$

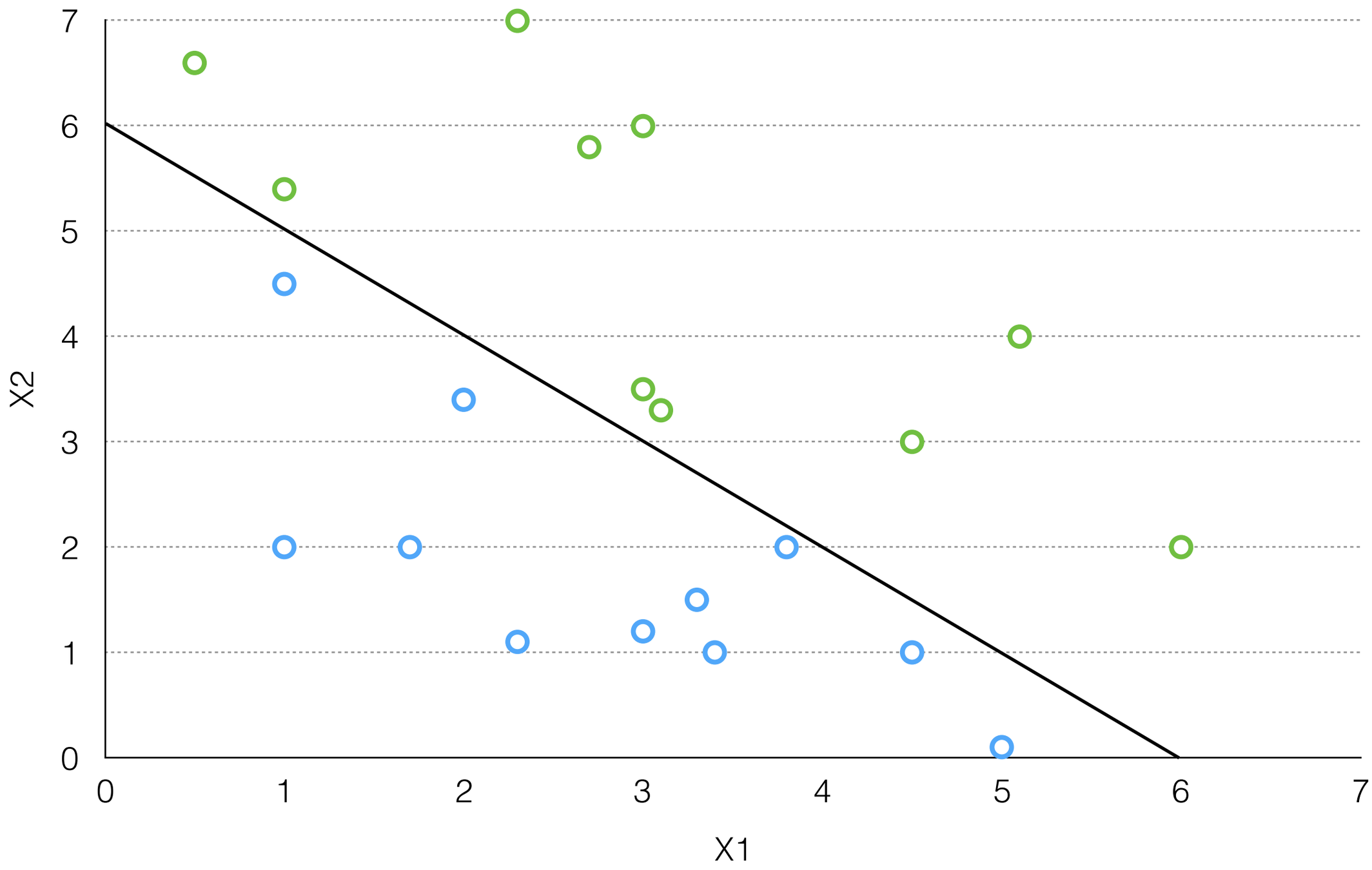
Decision Boundary

On-site Interview = $-6 + 1x_1 + 1x_2$

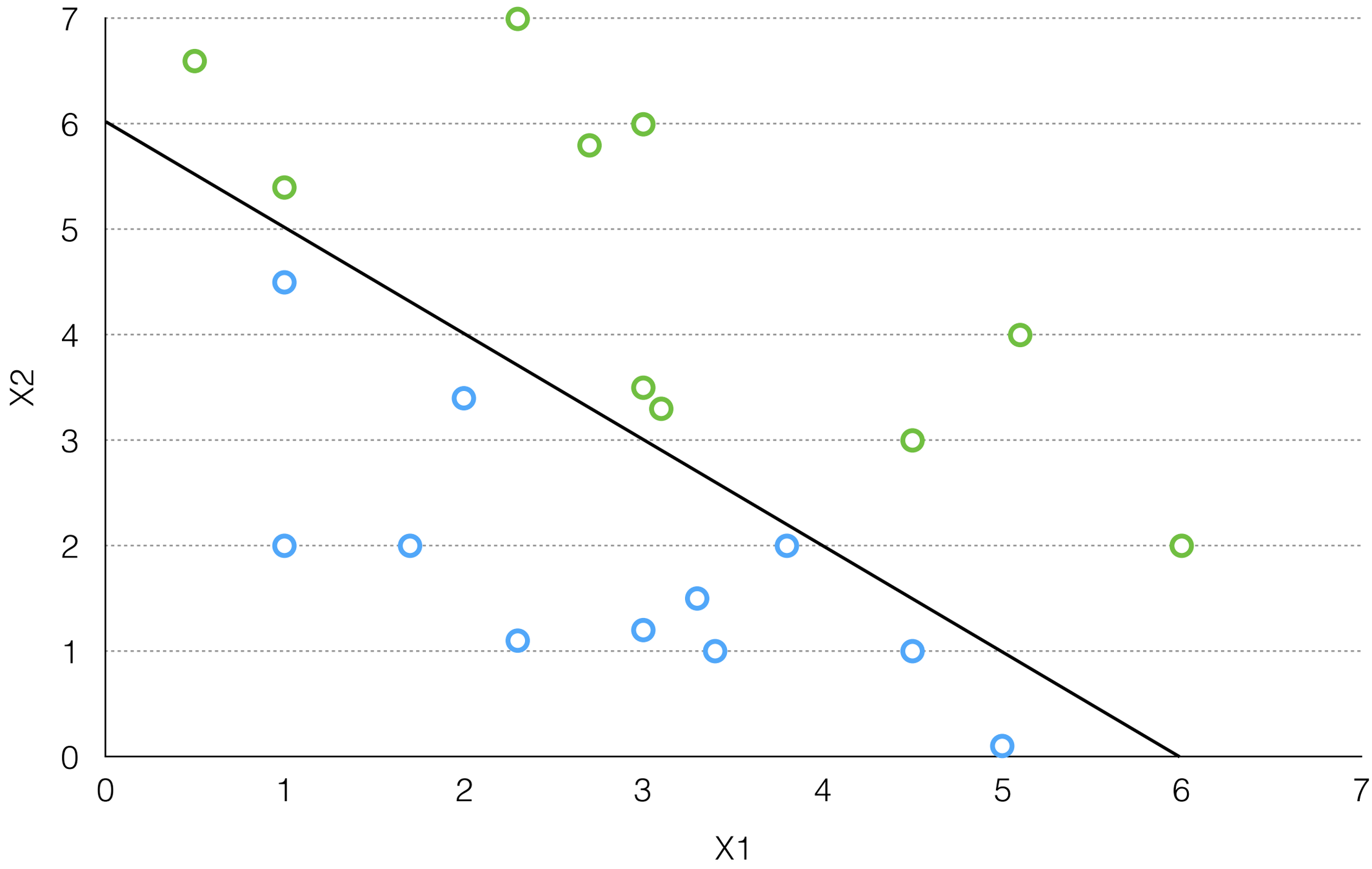
predict 1 when $-6 + 1x_1 + 1x_2 >= 0$

predict 0 when $-6 + 1x_1 + 1x_2 < 0$

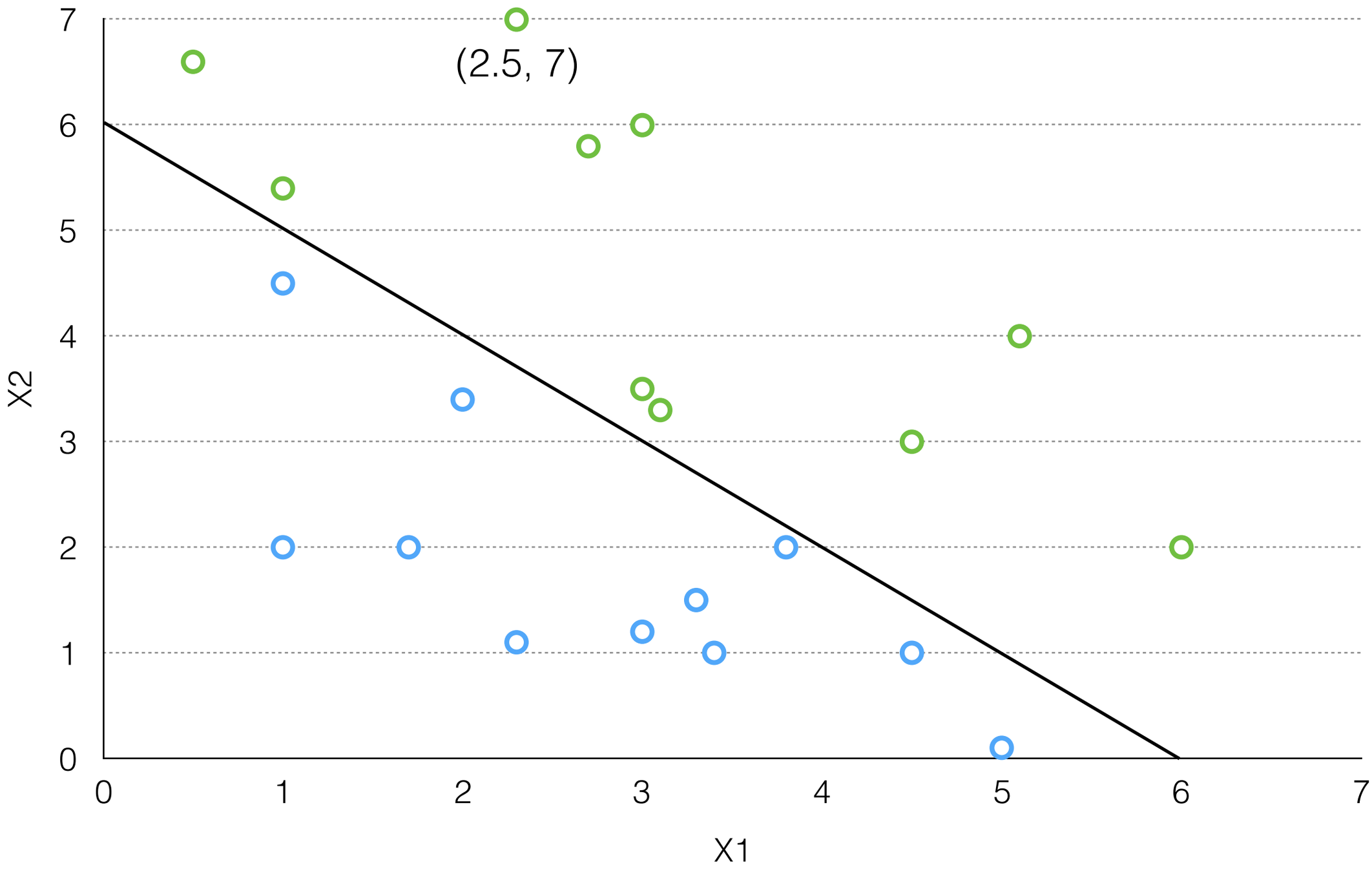
Decision Boundary = -6 + $1x_1$ + $1x_2$



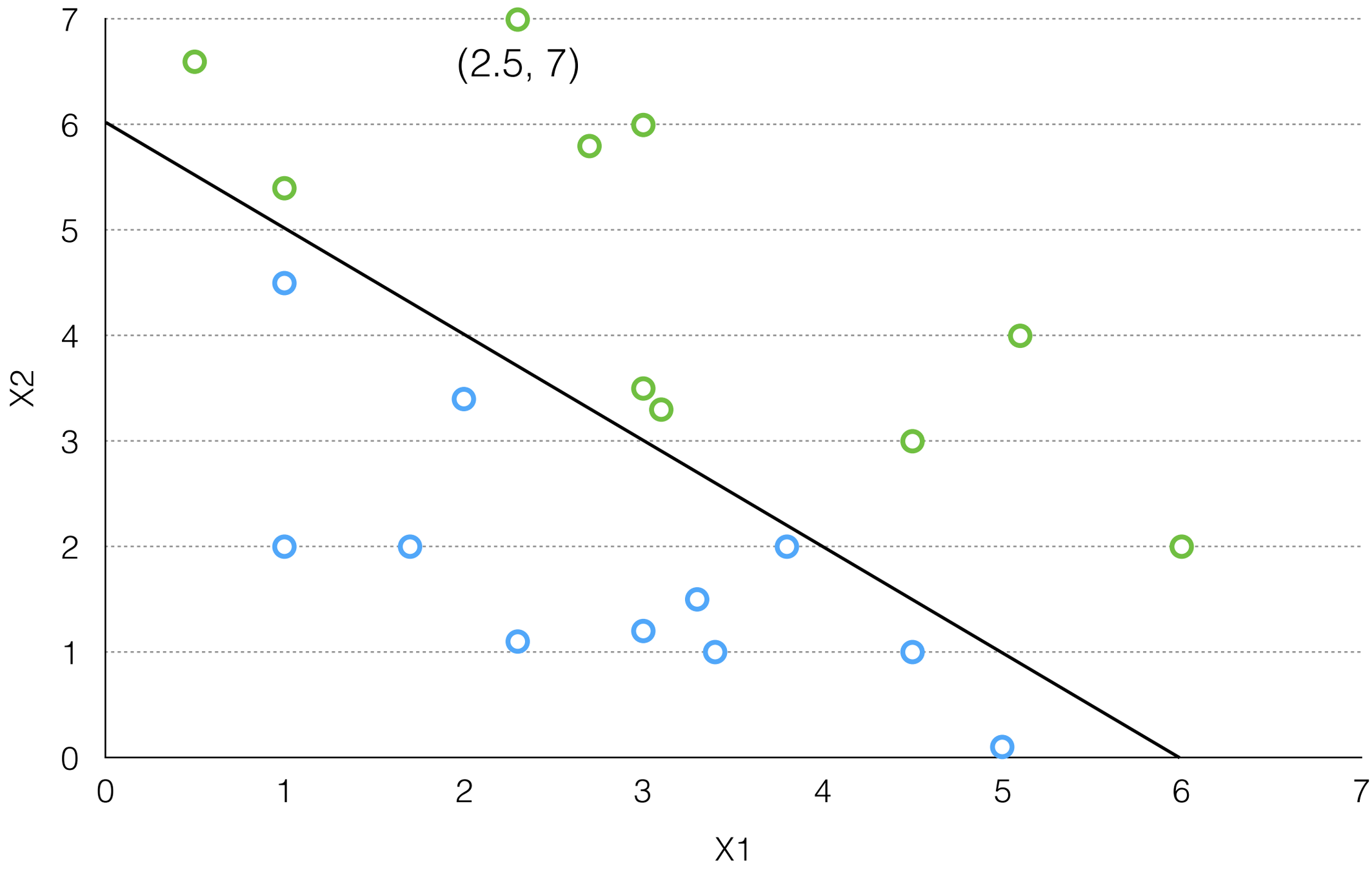
$$\hat{f}(X) = \frac{1}{1+e^{-(-6 + 1x_1 + 1x_2)}}$$



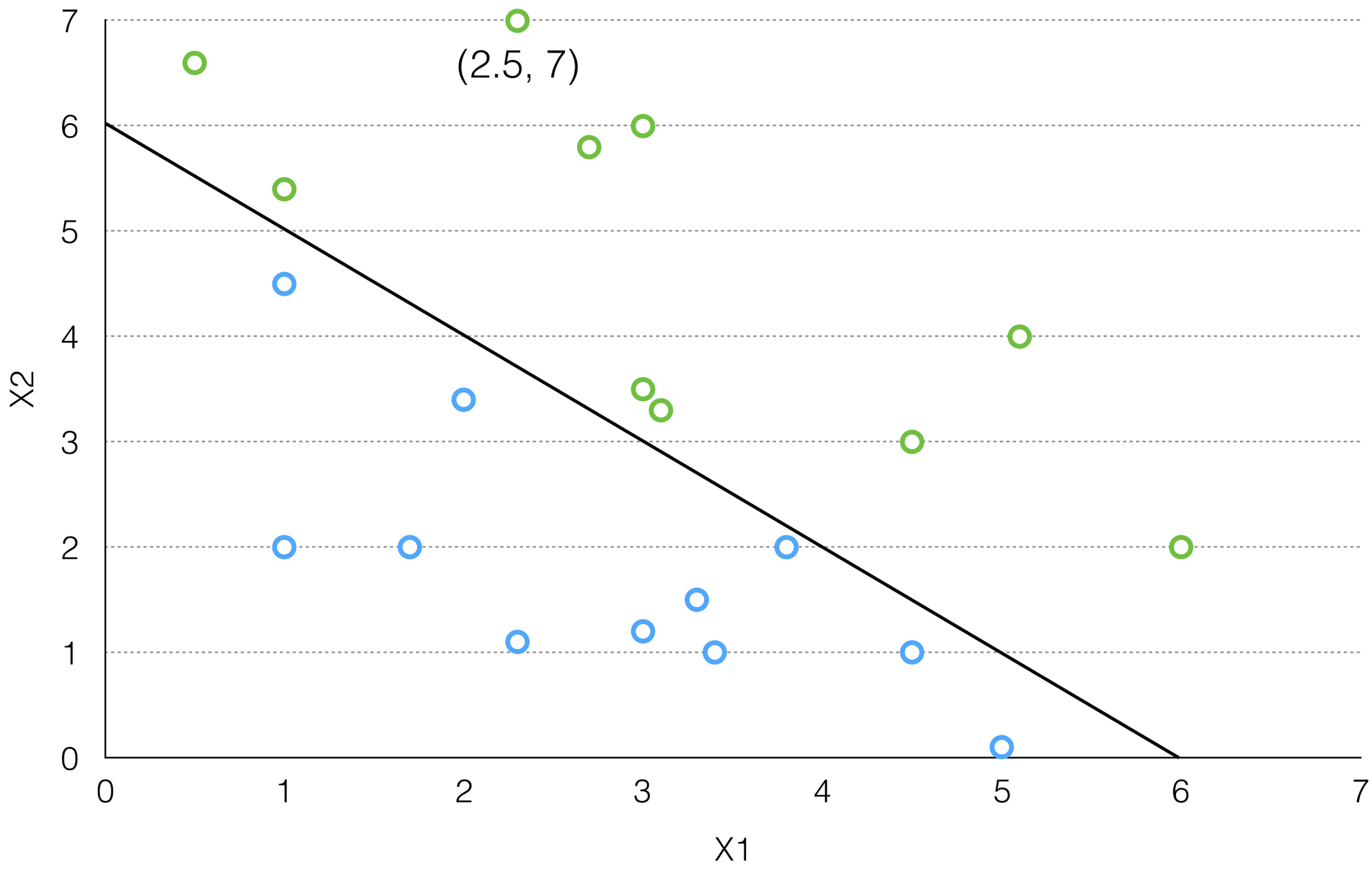
$$\frac{1}{1+e^{-(-6 + 1x_1 + 1x_2)}}$$



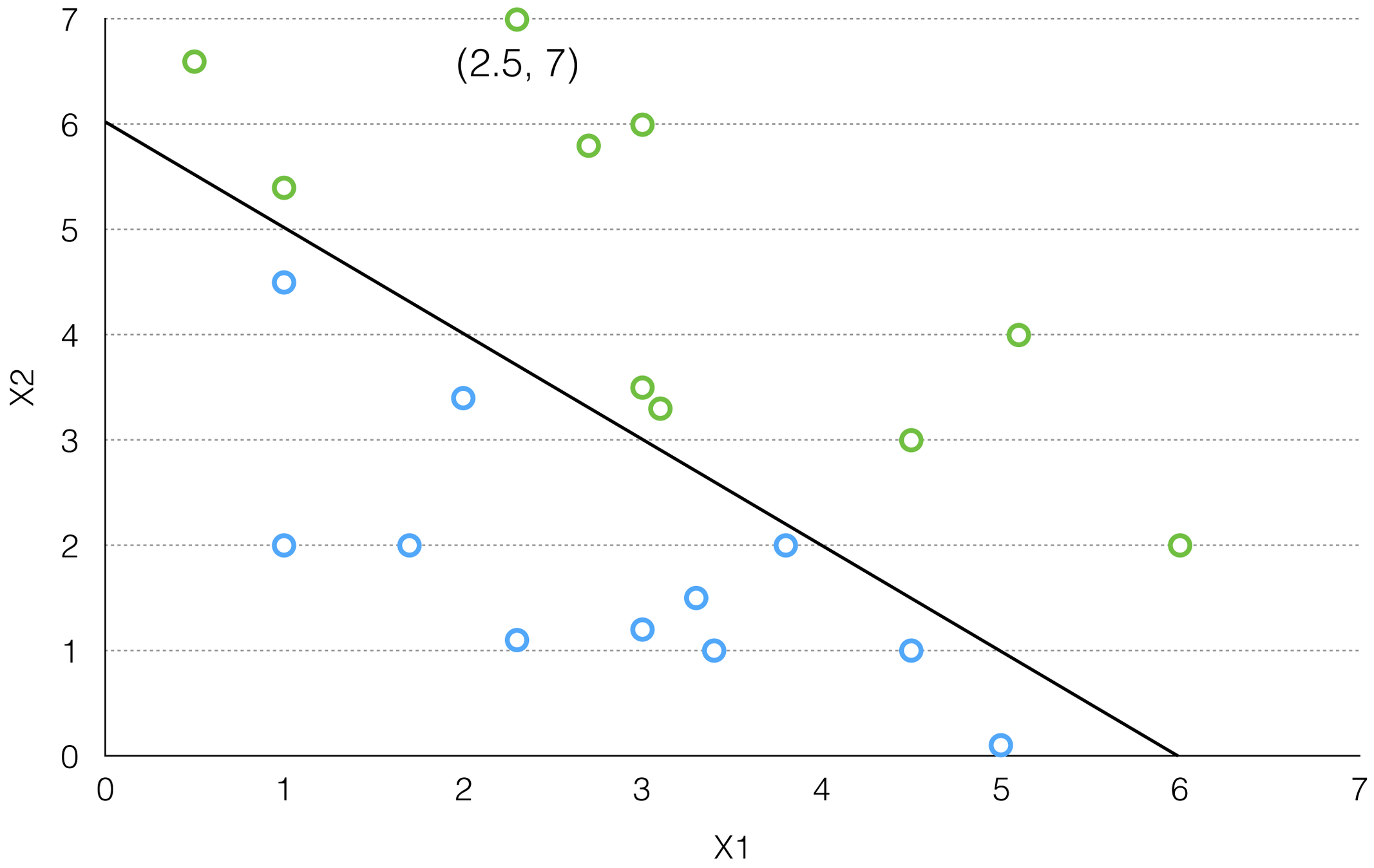
$$\frac{1}{1+e^{-(-6+2.5+7)}}$$



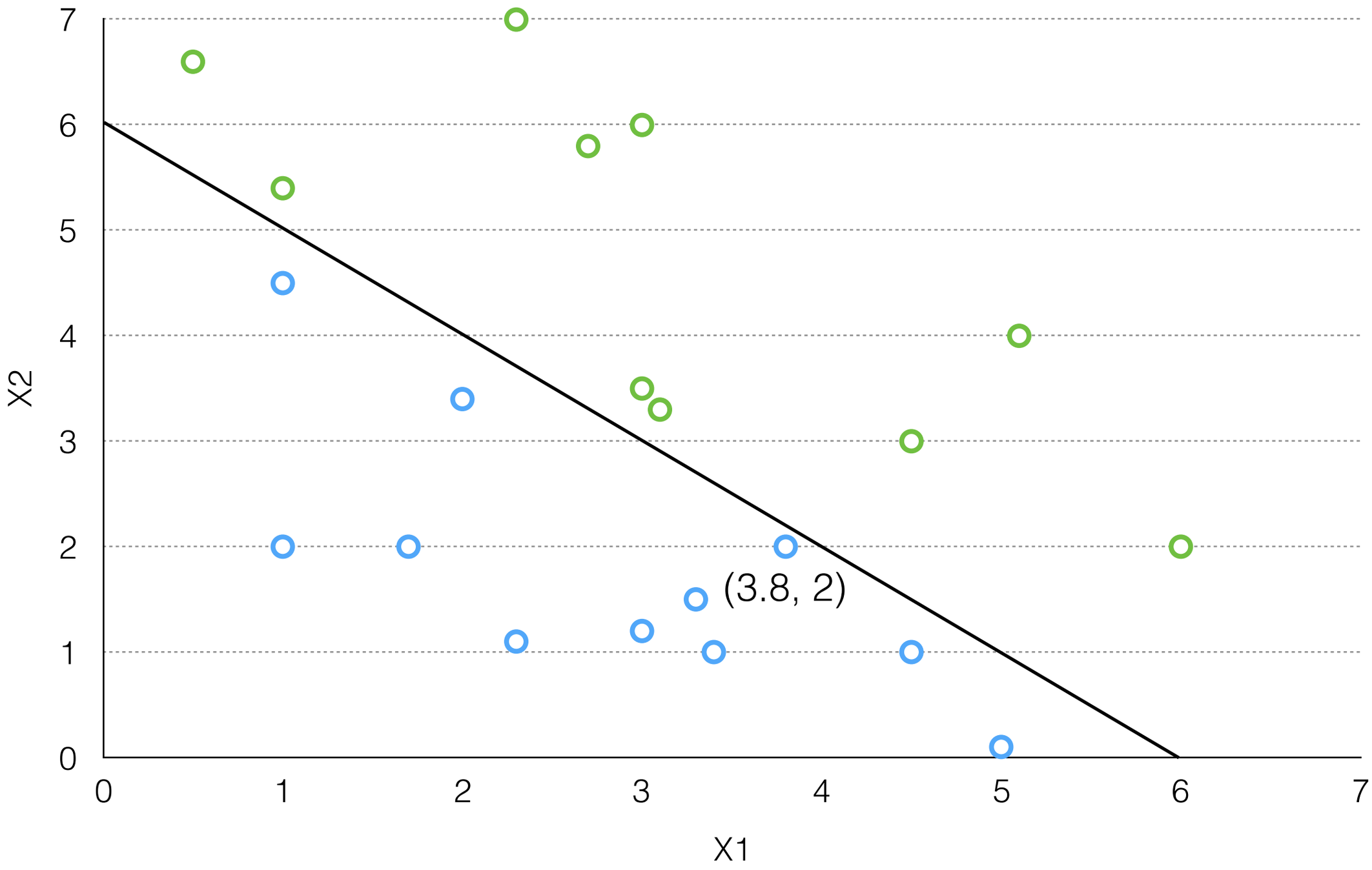
$$\frac{1}{1+e^{-(-6+9.5)}}$$



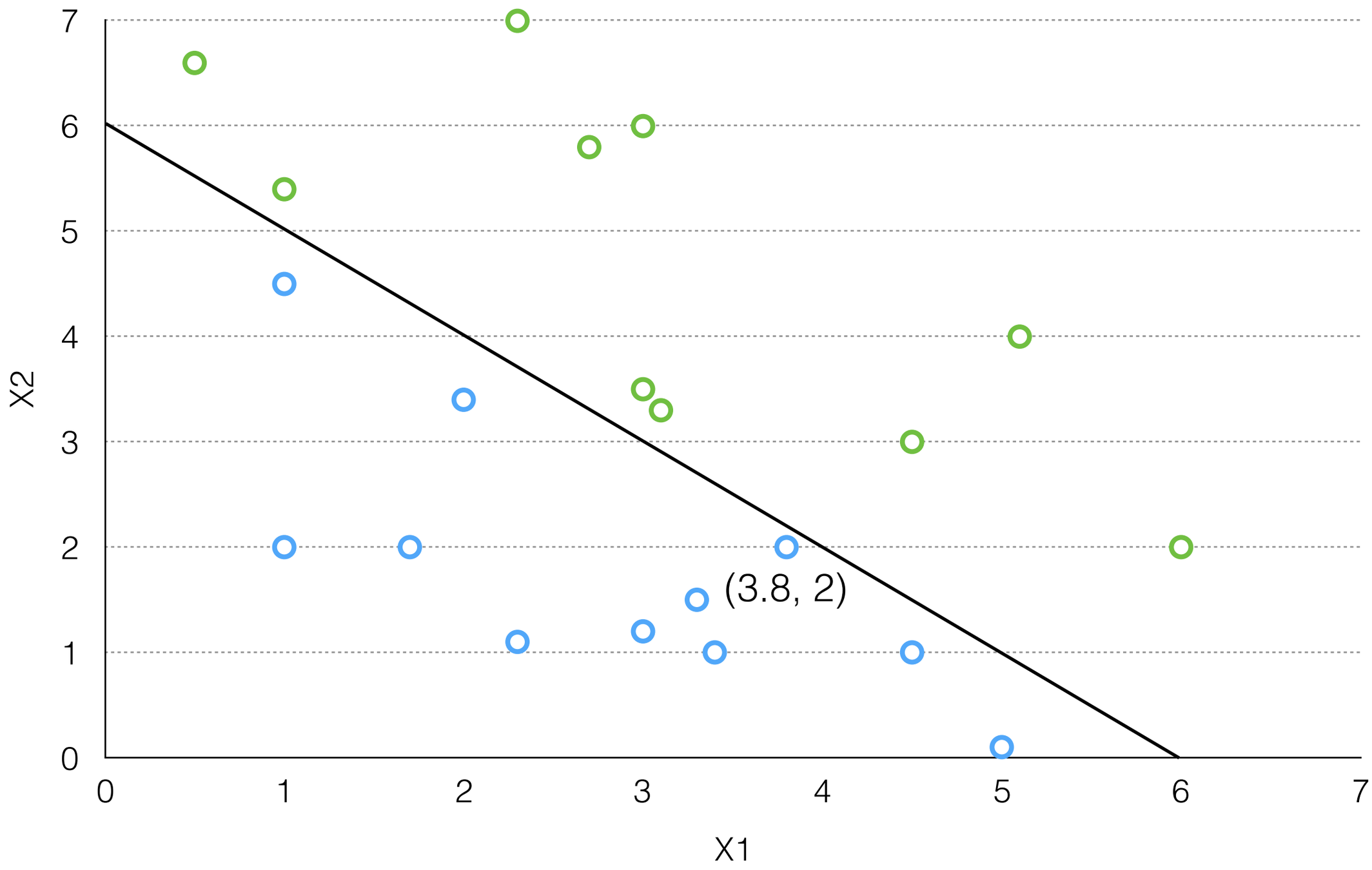
$$\frac{1}{1+e^{-(3.5)}} = .97 \text{ (probability)}$$



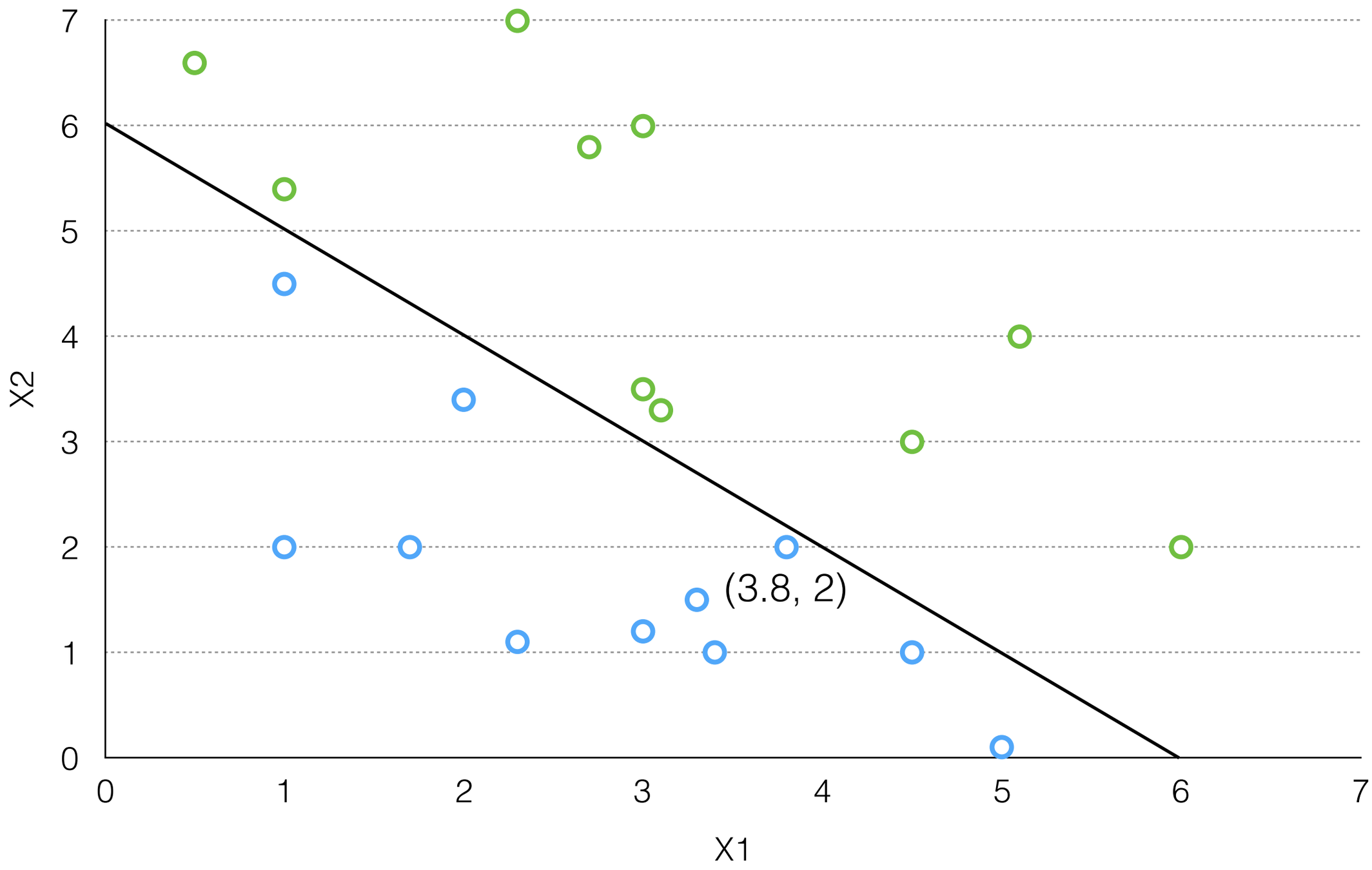
$$\frac{1}{1+e^{-(-6 + 1x_1 + 1x_2)}}$$



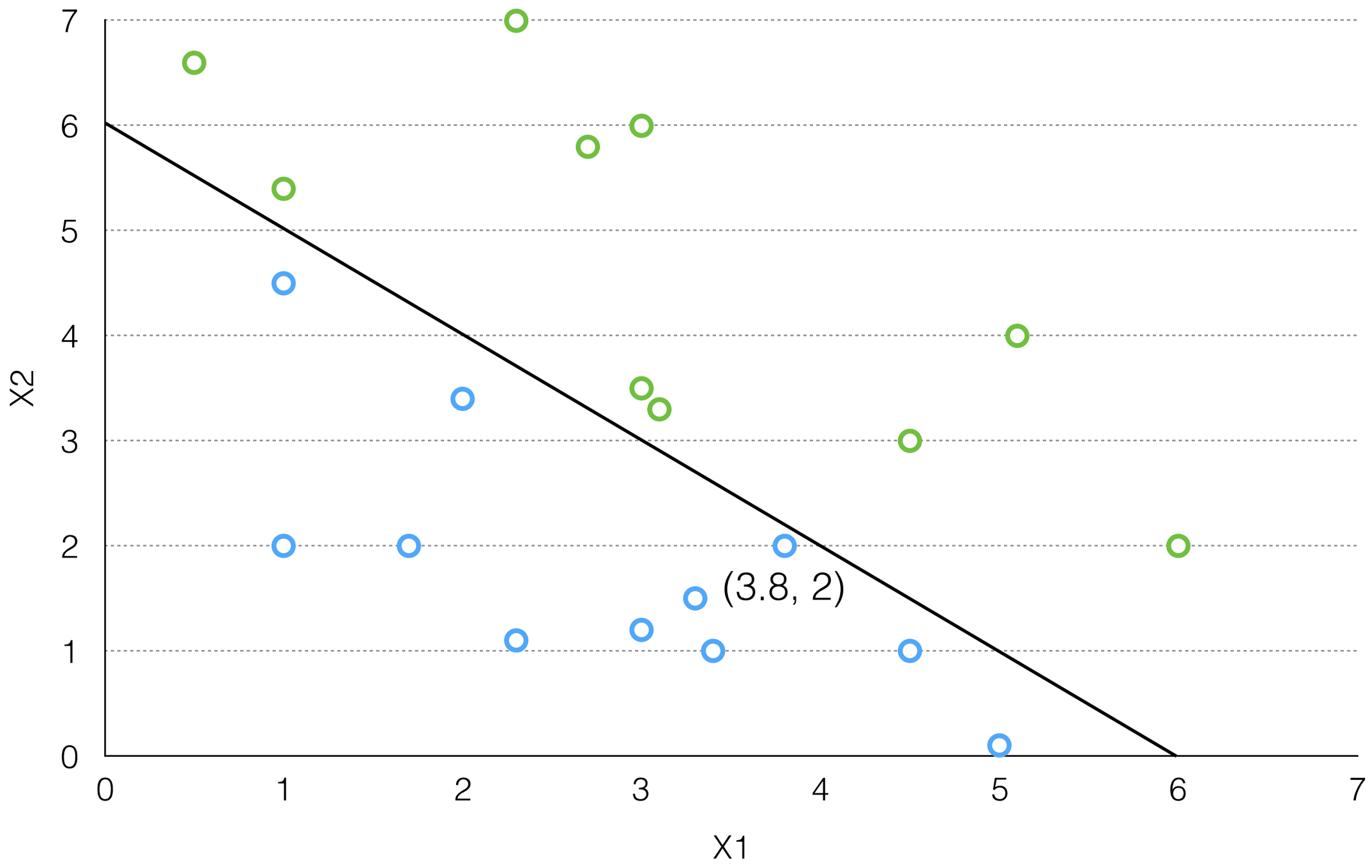
$$\frac{1}{1+e^{-(-6+3.8+2)}}$$



$$\frac{1}{1+e^{-(-6+5.8)}}$$



$$\frac{1}{1+e^{-(-0.2)}} = .45 \text{ probability}$$



Support Vector Machine

Support Vector Machine

$$\beta_0 + \beta_1 x + \beta_2 x + \beta_3 x$$

Support Vector Machine

predict 1 when $\beta_0 + \beta_1 x_1 + \beta_2 x_2 >= 0$

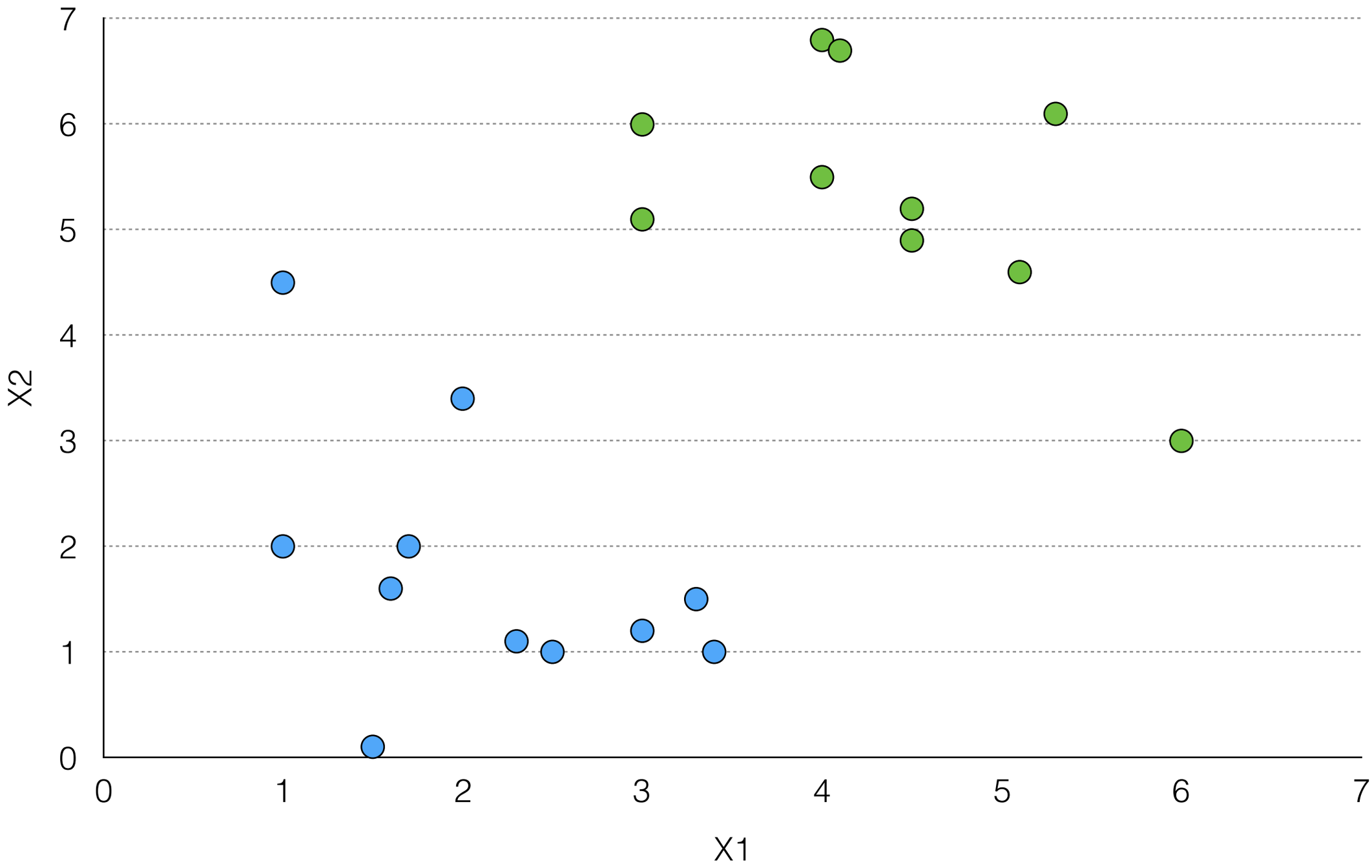
predict 0 when $\beta_0 + \beta_1 x_1 + \beta_2 x_2 < 0$

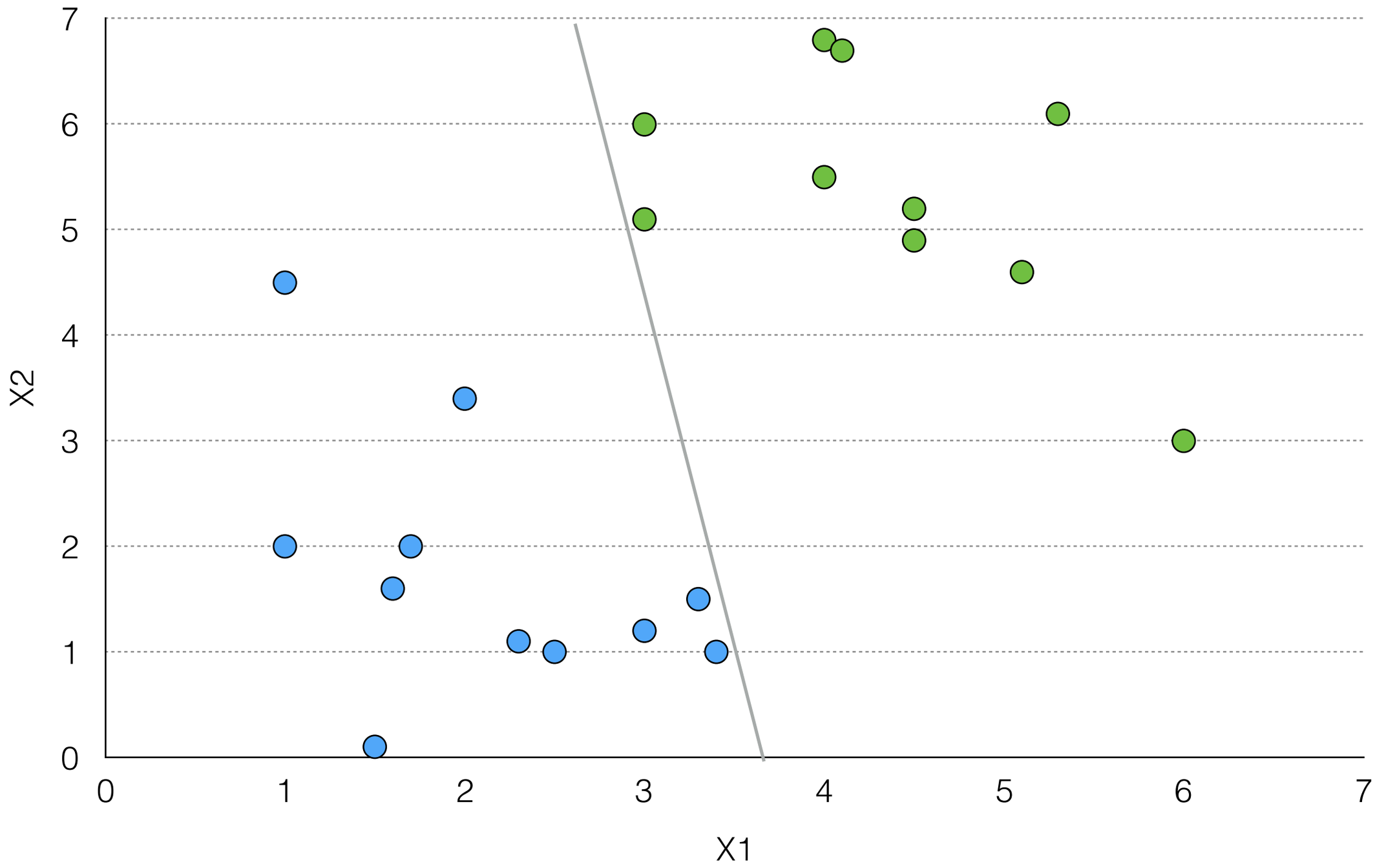
Support Vector Machine

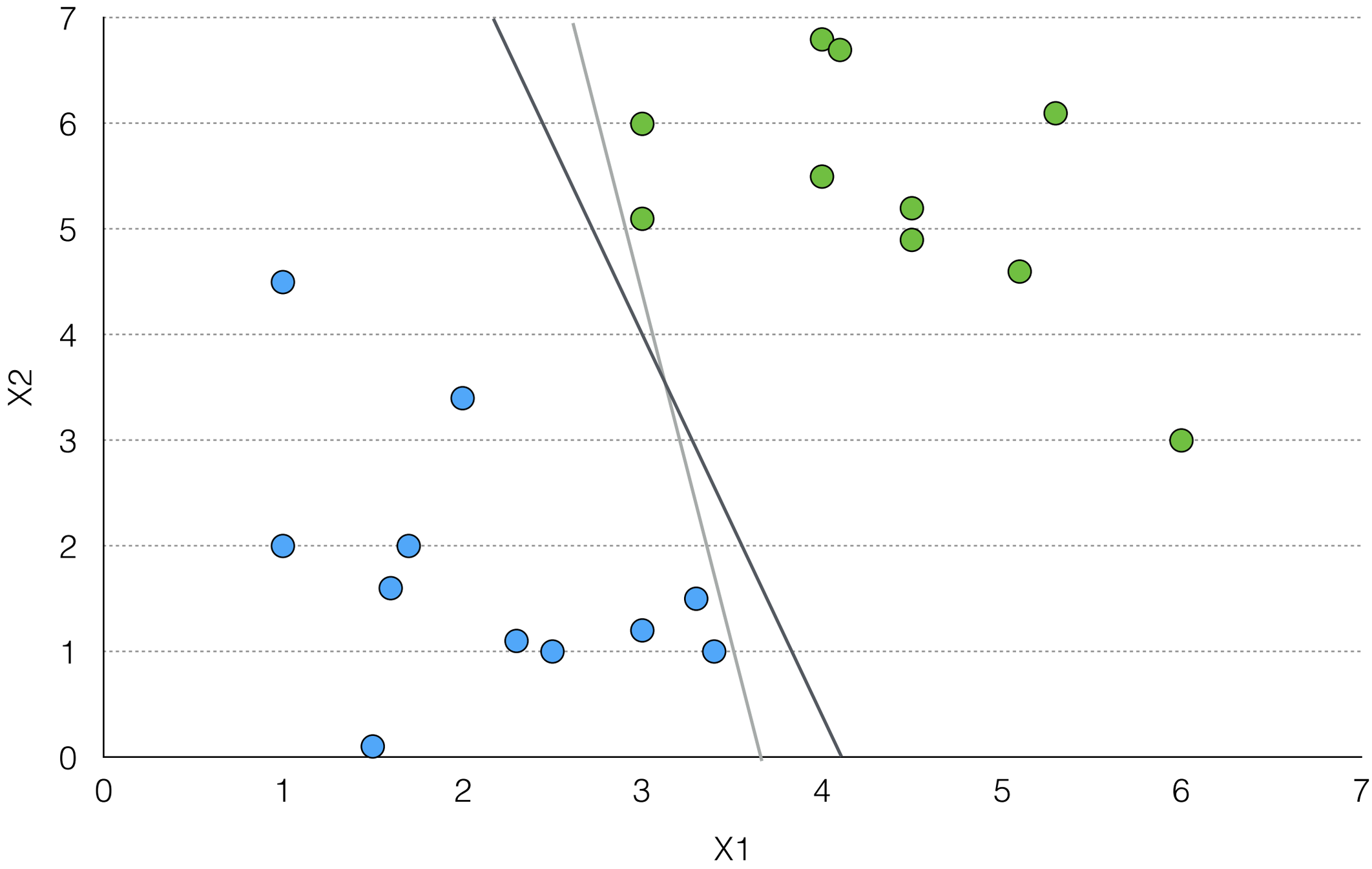
Large Margin Classifier

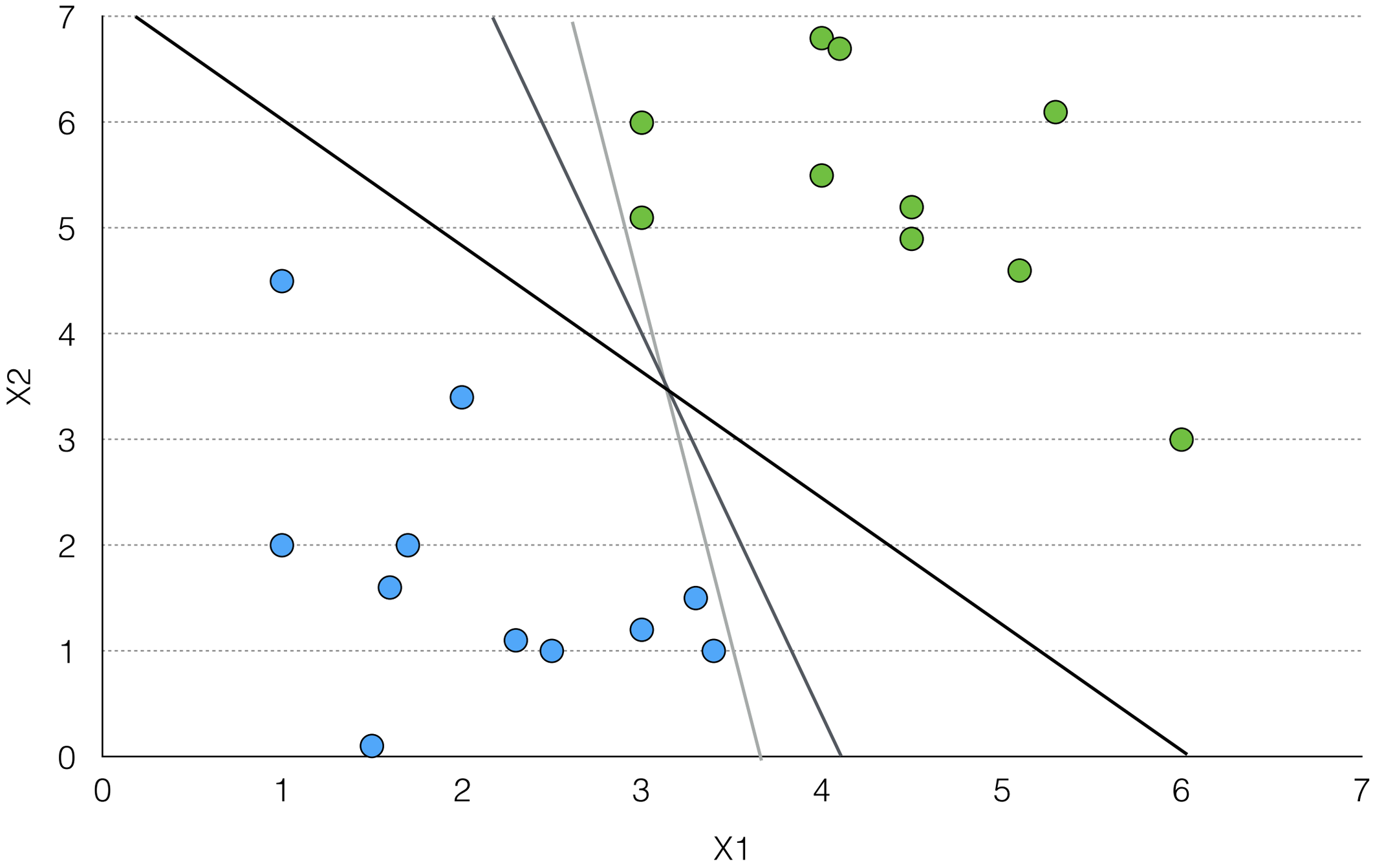
predict 1 when $\beta_0 + \beta_1 x_1 + \beta_2 x_2 >= 1$

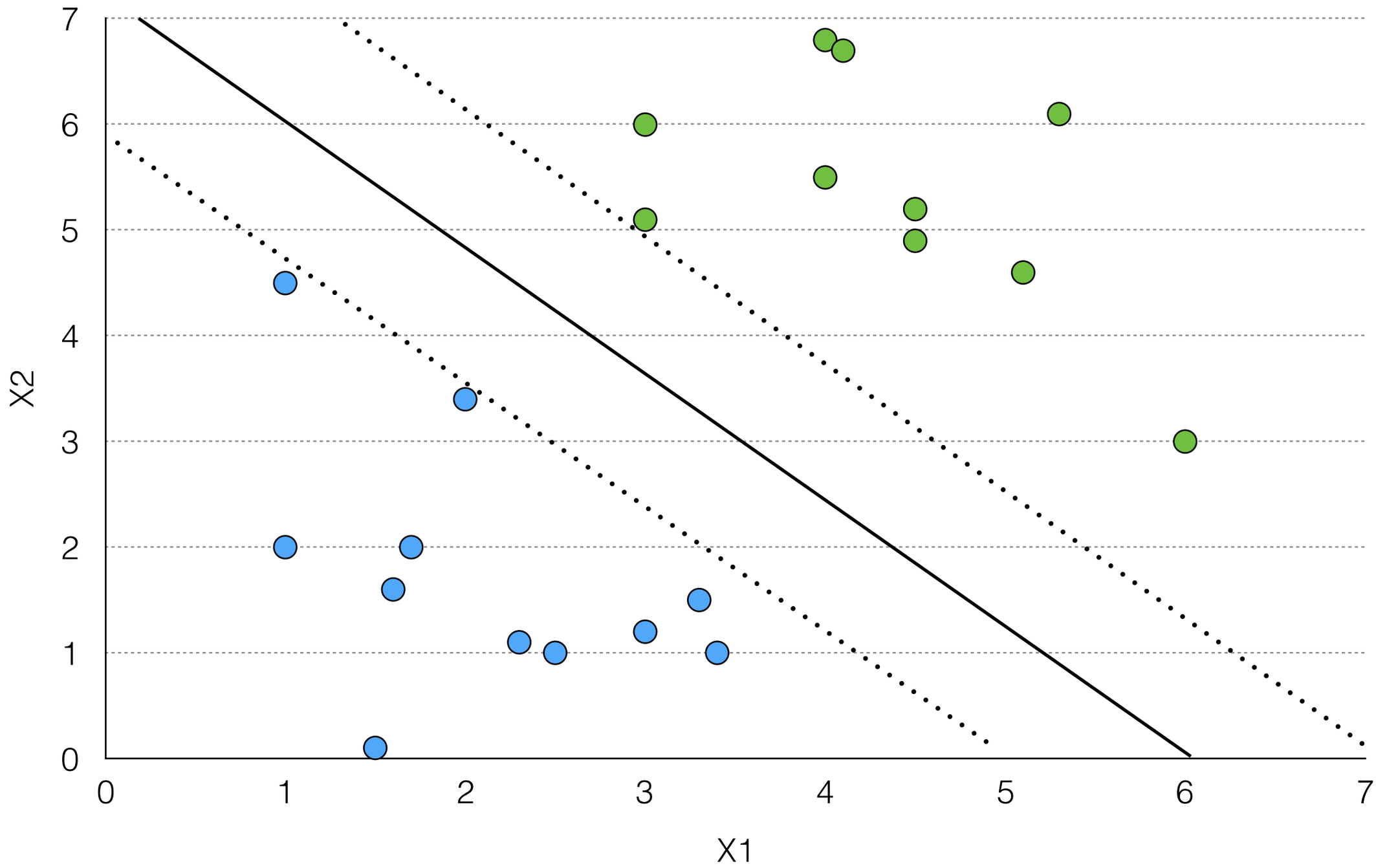
predict 0 when $\beta_0 + \beta_1 x_1 + \beta_2 x_2 < -1$

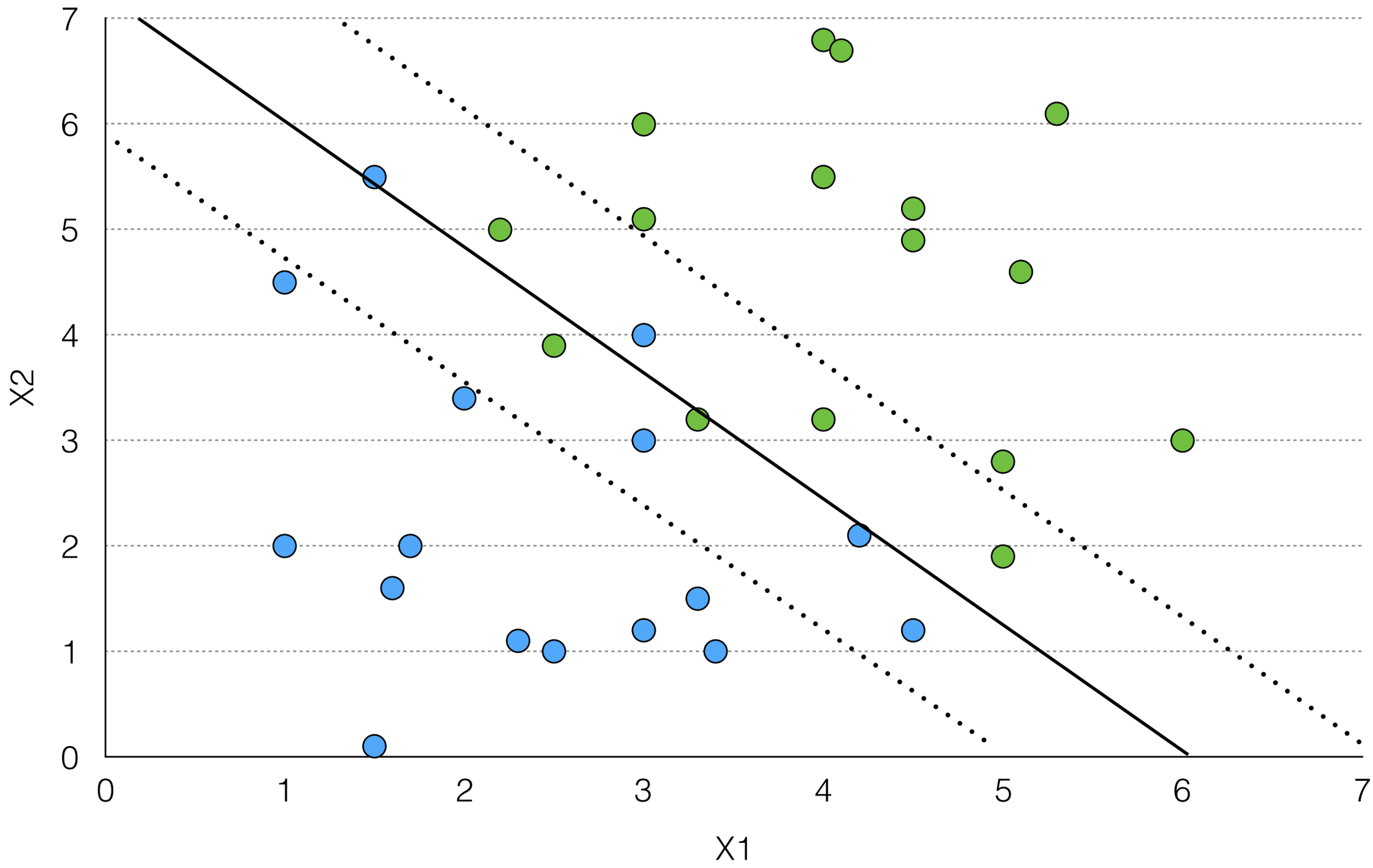

7
6
5
4
3
2
1
0
X2
0
1
2
3
4
5
6
7
X1


7
6
5
4
3
2
1
0
X2
0
1
2
3
4
5
6
7
X1


7
6
5
4
3
2
1
0
X2
0
1
2
3
4
5
6
7
X1

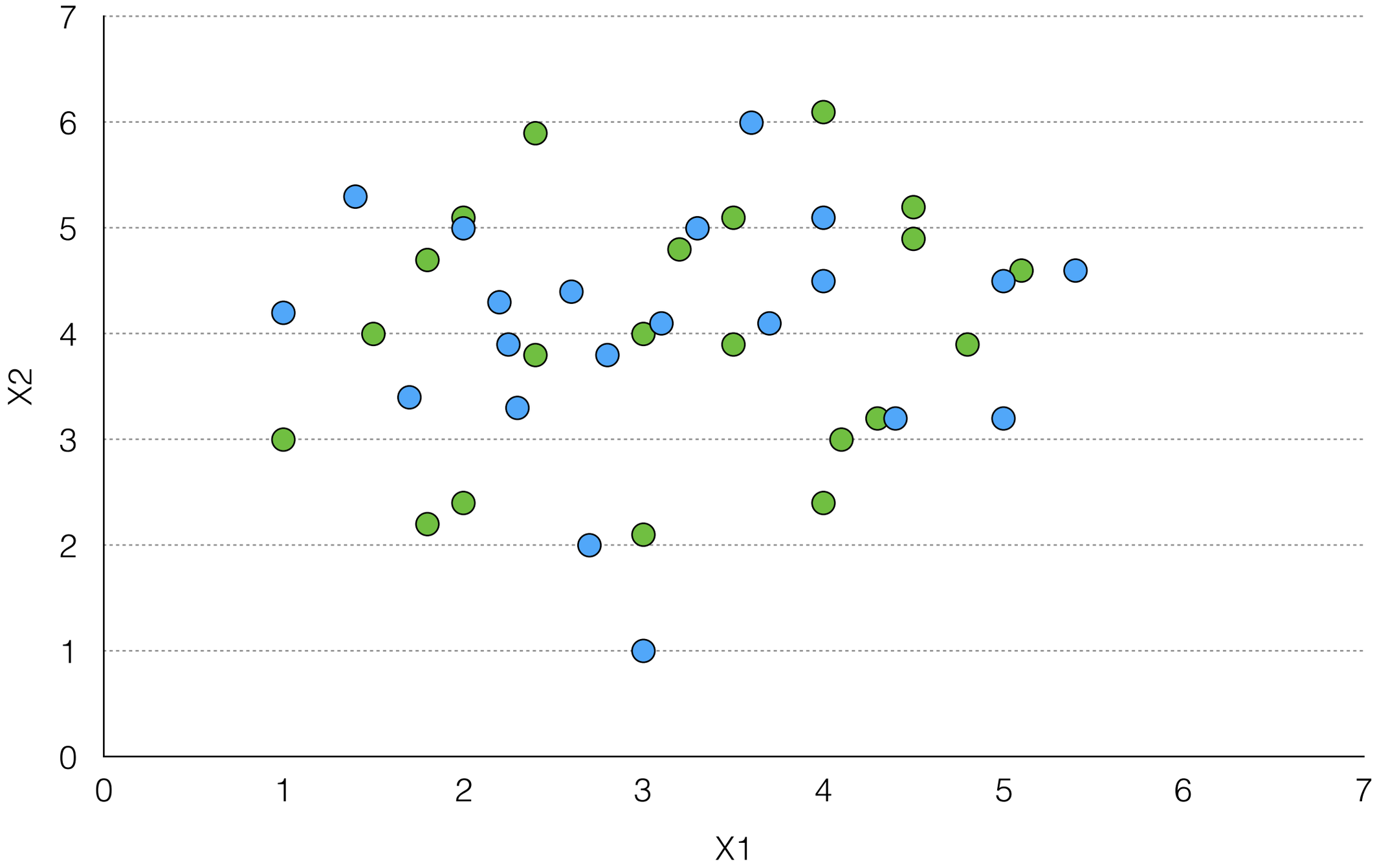

7
6
5
4
3
2
1
0
X2
0
1
2
3
4
5
6
7
X1


7
6
5
4
3
2
1
0
X2
0
1
2
3
4
5
6
7
X1


7
6
5
4
3
2
1
0
X2
0
1
2
3
4
5
6
7
X1


NFL
SCOUTING
COMBINE
2016

40-yard dash	Weight	Height	Drafted
5.10	290	74	1
4.92	275	75.5	1
4.43	178	69	0
4.62	221	74.5	1
4.91	248	75	0
5.53	303	77	0
4.47	189	71	1
4.56	205	71	1
4.75	267	73	0
4.84	261	74	1


7
6
5
4
3
2
1
0
X2
0
1
2
3
4
5
6
7
X1

40-yard dash	Weight	Height	Drafted
5.10	290	74	1
4.92	275	75.5	1
4.43	178	69	0
4.62	221	74.5	1
4.91	248	75	0
5.53	303	77	0
4.47	189	71	1
4.56	205	71	1
4.75	267	73	0
4.84	261	74	1

Feature Engineering

40-yard dash	BMI (wt/ht^2)	Drafted
5.10	37.2	1
4.92	33.9	1
4.43	26.3	0
4.62	28	1
4.91	31	0
5.53	35.9	0
4.47	26.4	1
4.56	28.6	1
4.75	35.2	0
4.84	33.5	1

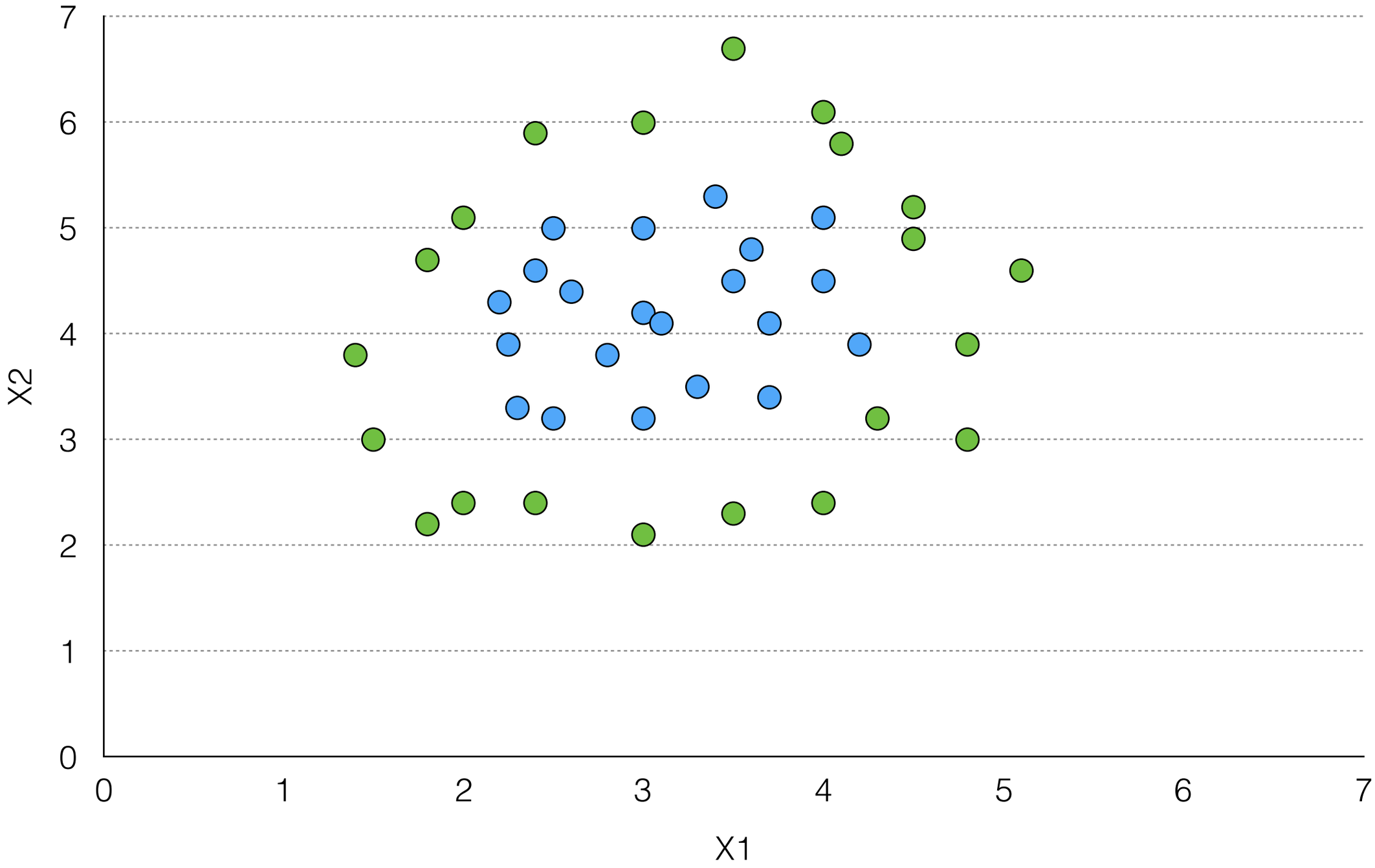

44
OL

Feature Engineering

40-yard dash	BMI (wt/ht^2)	Drafted
5.10	37.2	1
4.92	33.9	1
4.43	26.3	0
4.62	28	1
4.91	31	0
5.53	35.9	0
4.47	26.4	1
4.56	28.6	1
4.75	35.2	0
4.84	33.5	1

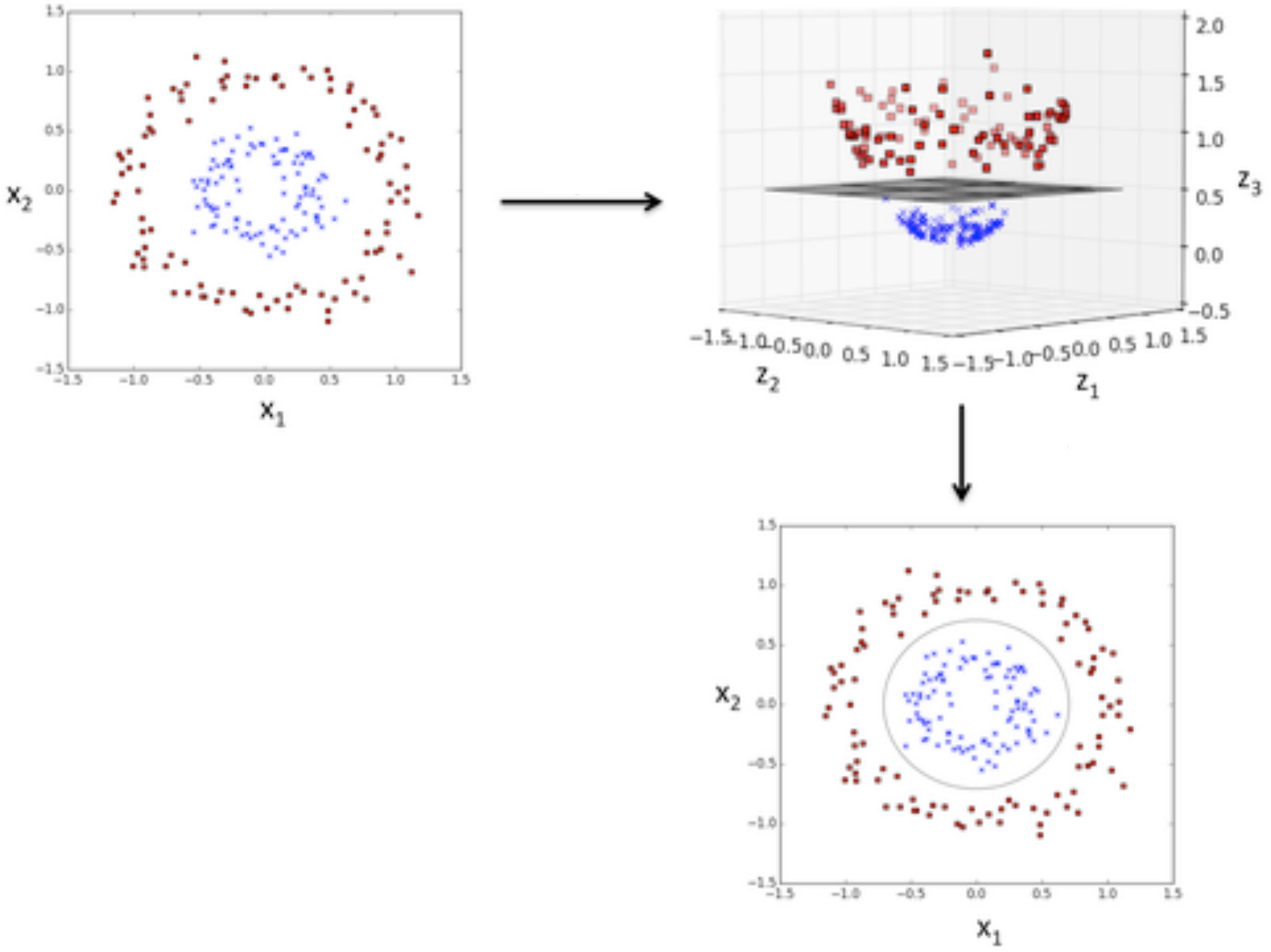
Feature Engineering

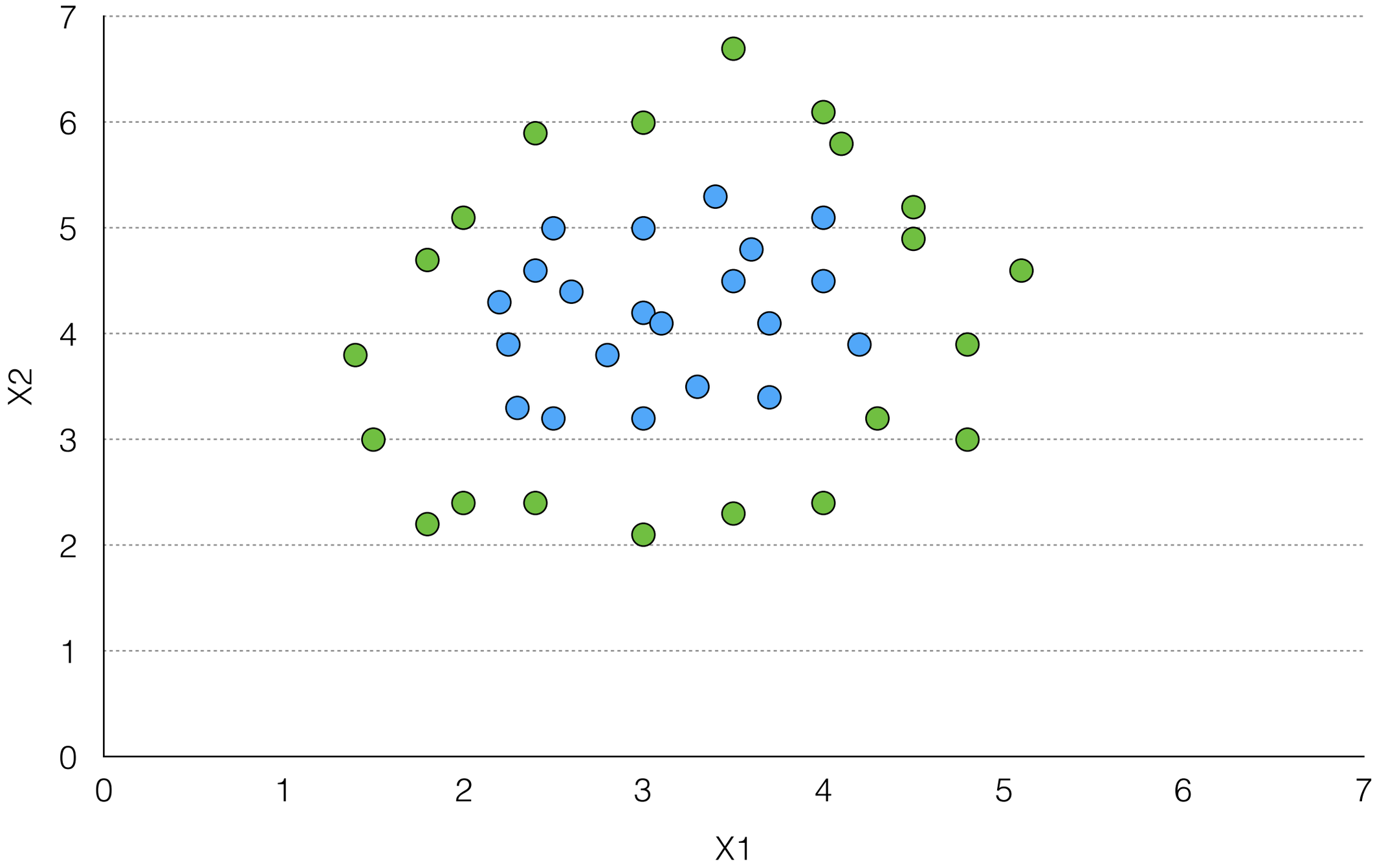
Speed-to-Size (40-yd/bsa)	BMI (wt/ht^2)	Drafted
2.16	37.2	1
2.06	33.9	1
2.02	26.3	0
1.97	28	1
2.23	31	0
2.00	35.9	0
2.03	26.4	1
1.99	28.6	1
1.85	35.2	0
2.03	33.5	1

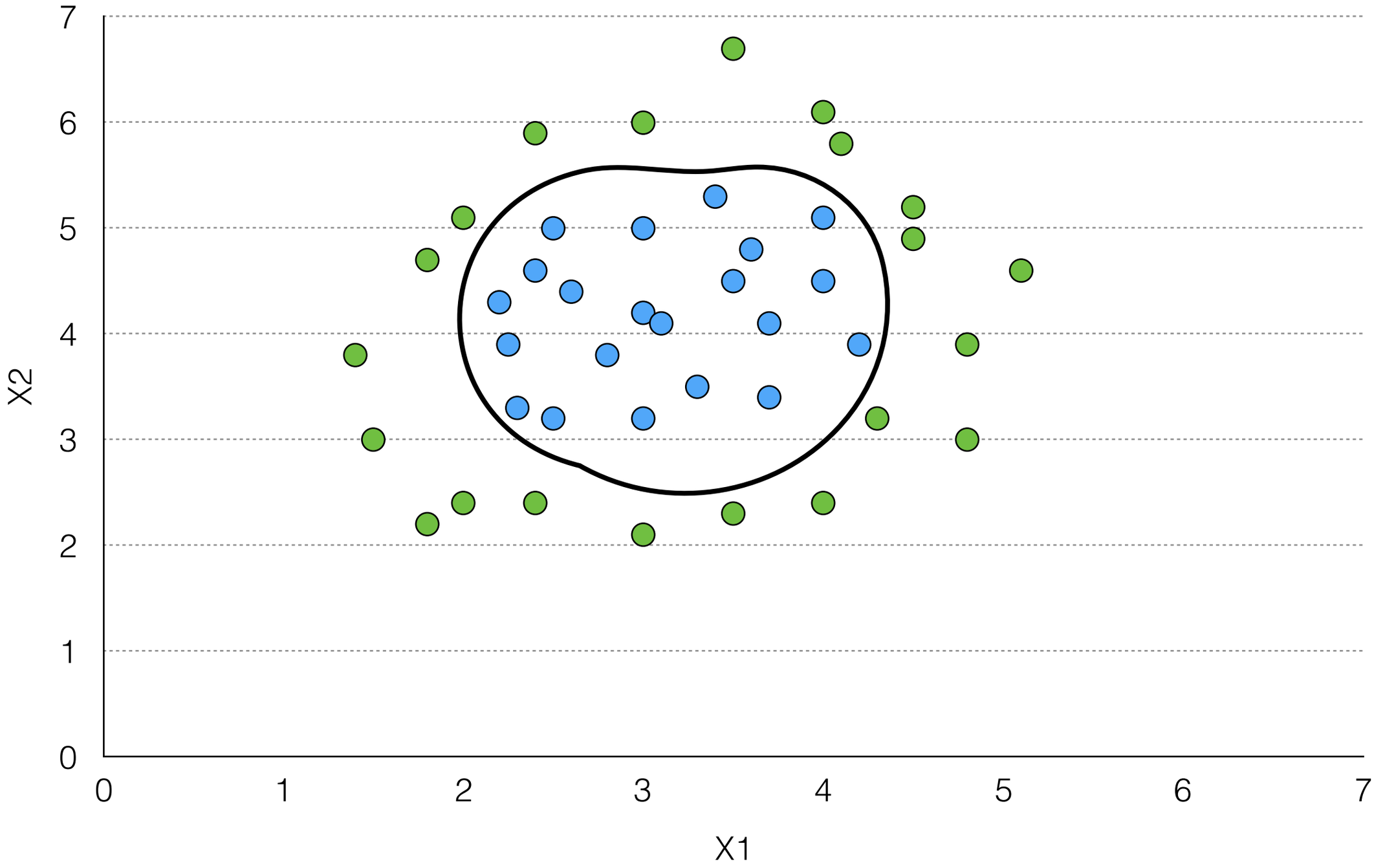

7
6
5
4
3
2
1
0
X2
0
1
2
3
4
5
6
7
X1

Kernel

non-linear classification

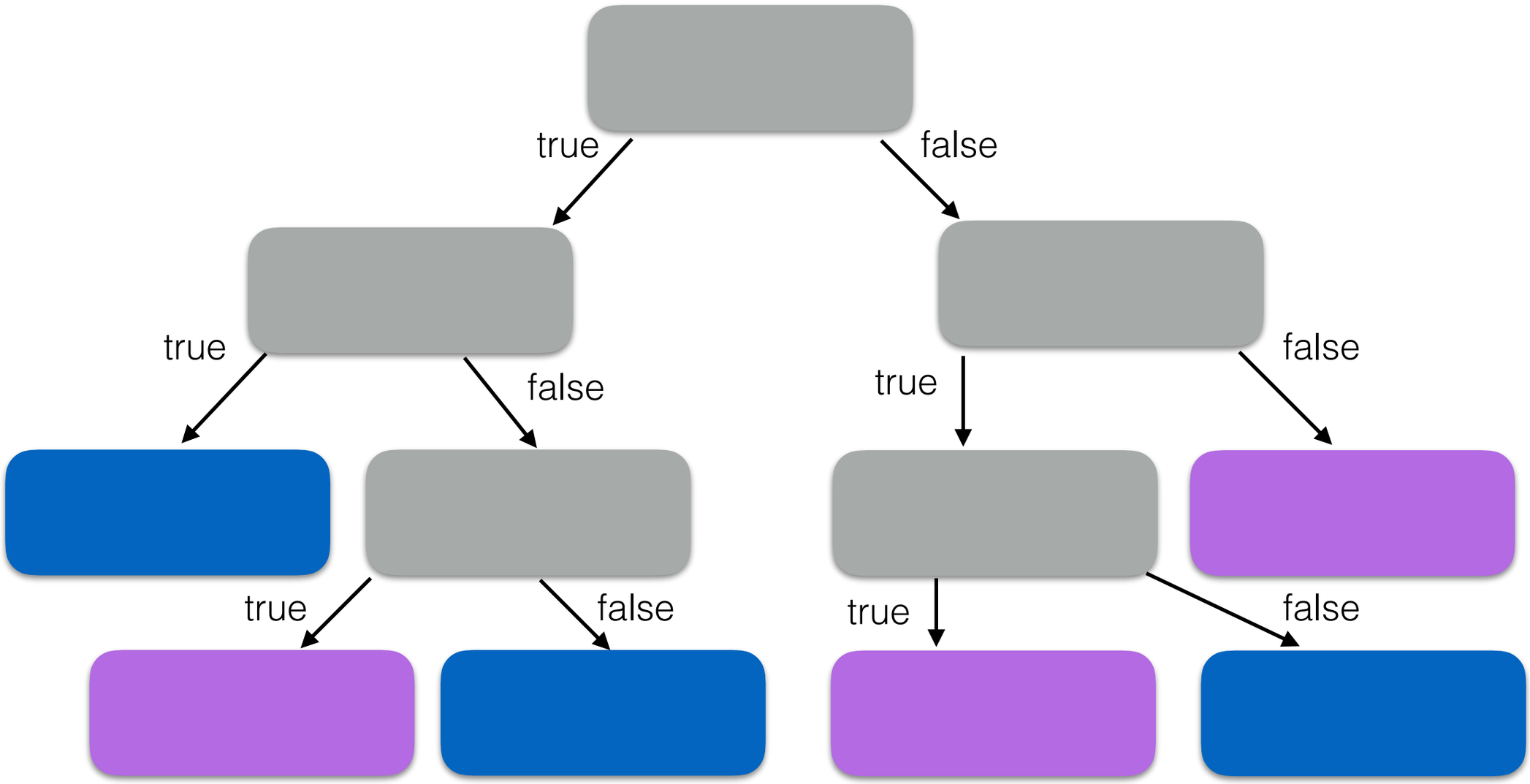

x_2
x_1
z_3
z_2
z_1
x_2
x_1


7
6
5
4
3
2
1
0
X2
0
1
2
3
4
5
6
7
X1


7
6
5
4
3
2
1
0
X2
0
1
2
3
4
5
6
7
X1

Decision Tree

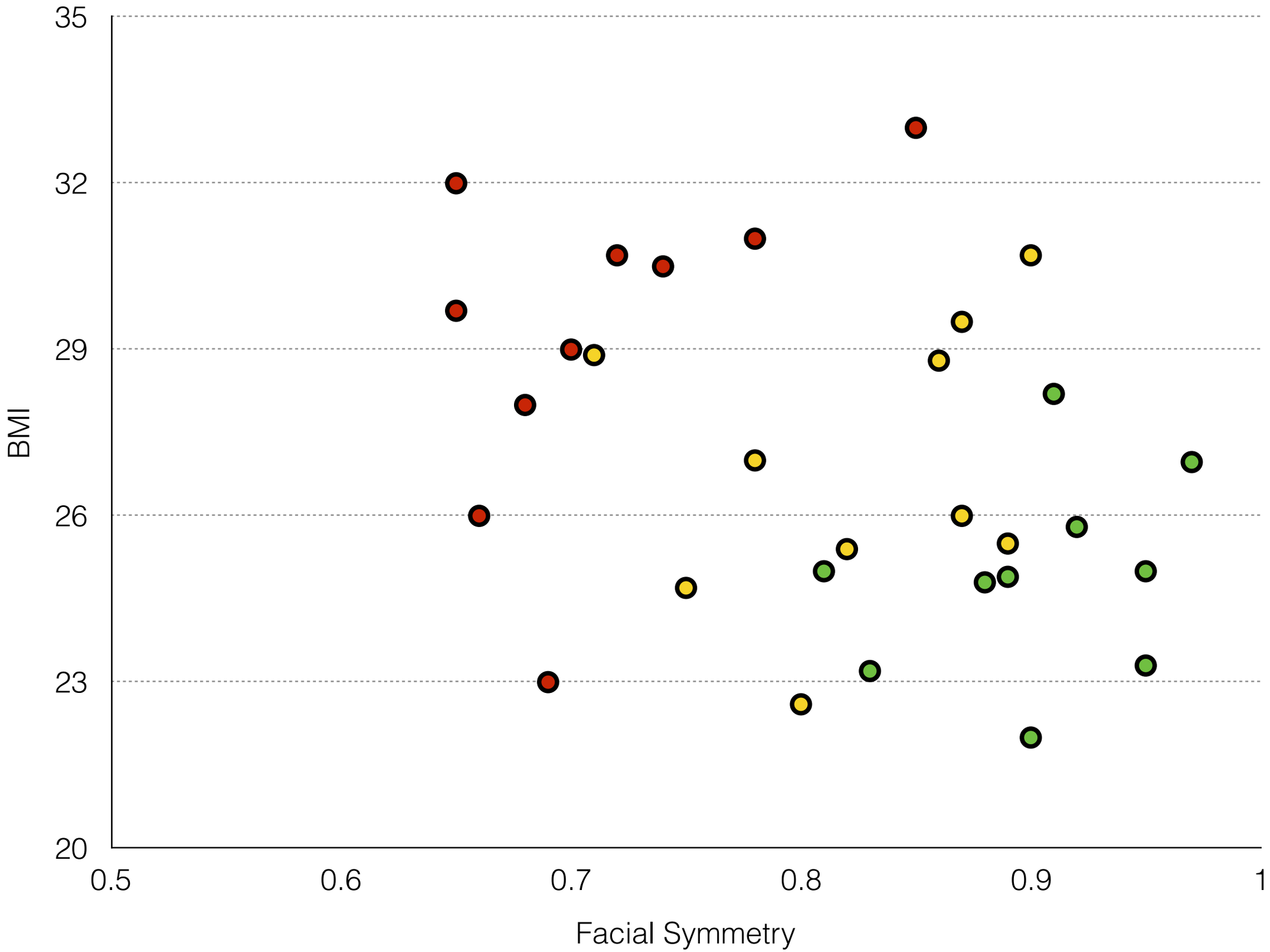
Decision Tree



Short-term Attractiveness

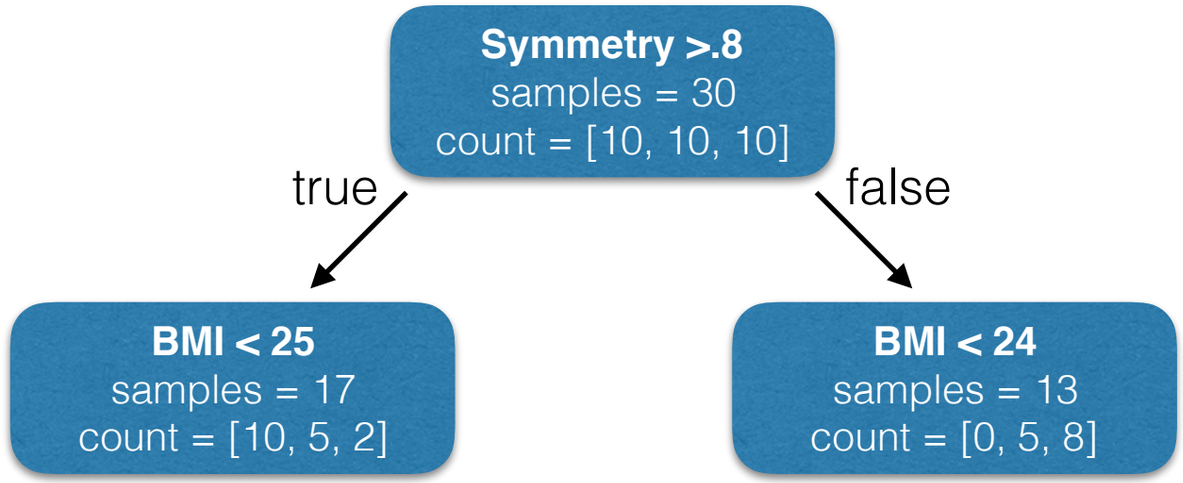
Short-term Attractiveness

Facial Symmetry	BMI	Waist-to-Hip	Well-Groomed
0.9	23.4	0.93	1
0.85	27.9	0.87	0
0.65	27.1	0.79	1
0.85	22.6	0.91	1
0.9	30.3	0.82	0
0.75	29.0	0.82	0
0.85	22.3	0.89	1
0.7	37.6	0.73	0
0.85	24.2	0.85	0

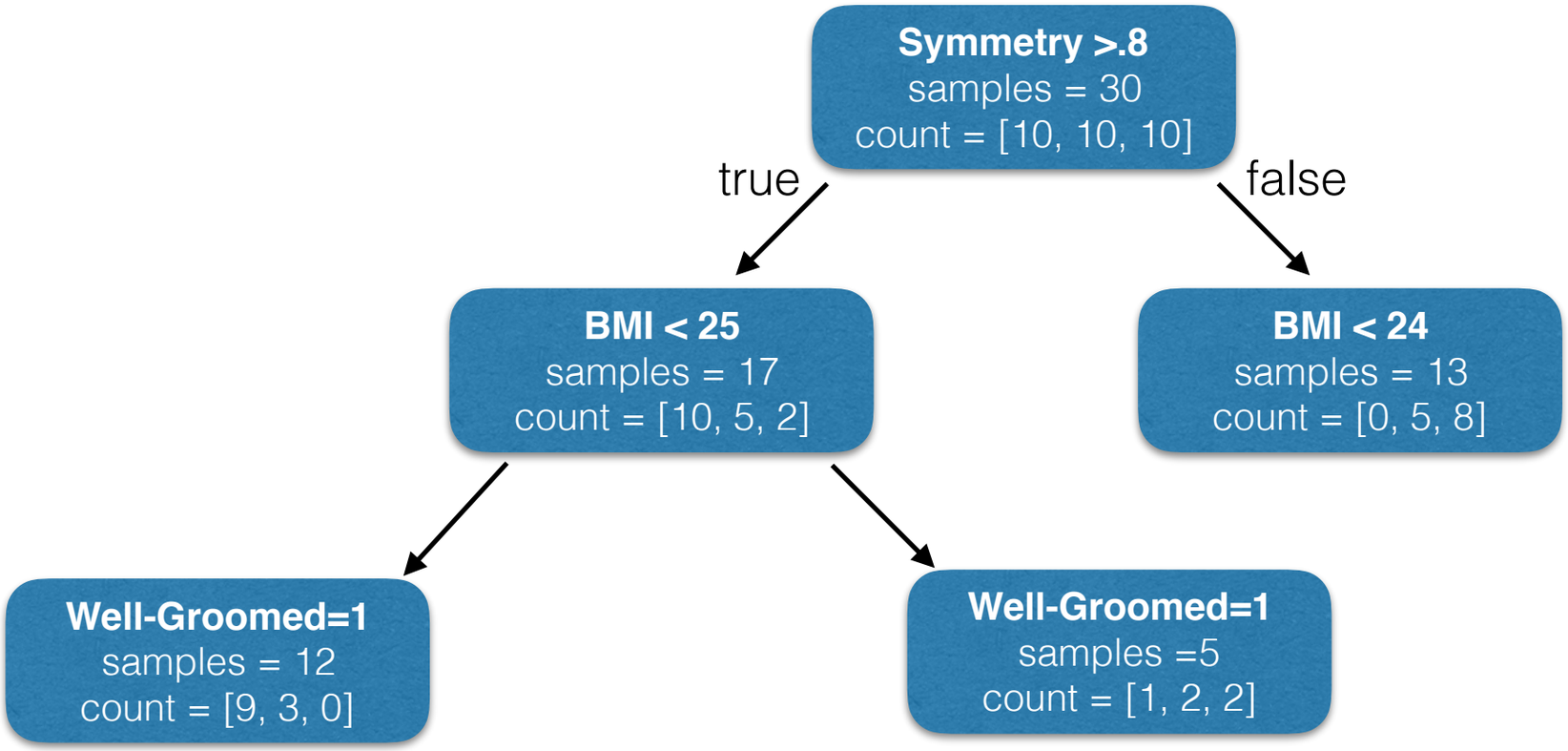

35
32
29
26
23
20
BMI
0.5
0.6
0.7
0.8
0.9
1
Facial Symmetry

Symmetry >.8
samples = 30
count = [10, 10, 10]

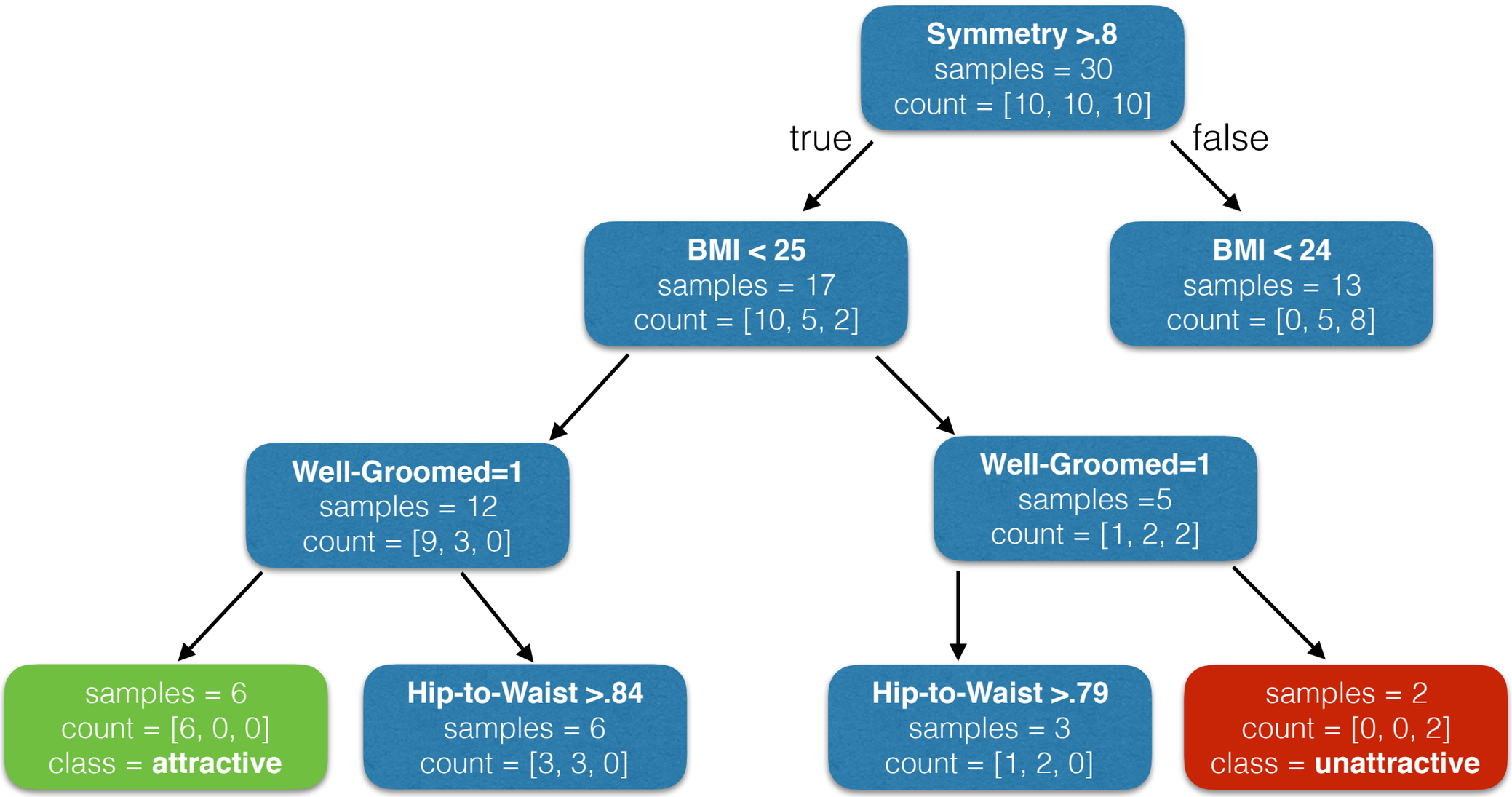
[att, ave, un]


Symmetry >.8
samples = 30
count = [10, 10, 10]
true
false
BMI < 25
samples = 17
count = [10, 5, 2]
BMI < 24
samples = 13
count = [0, 5, 8]

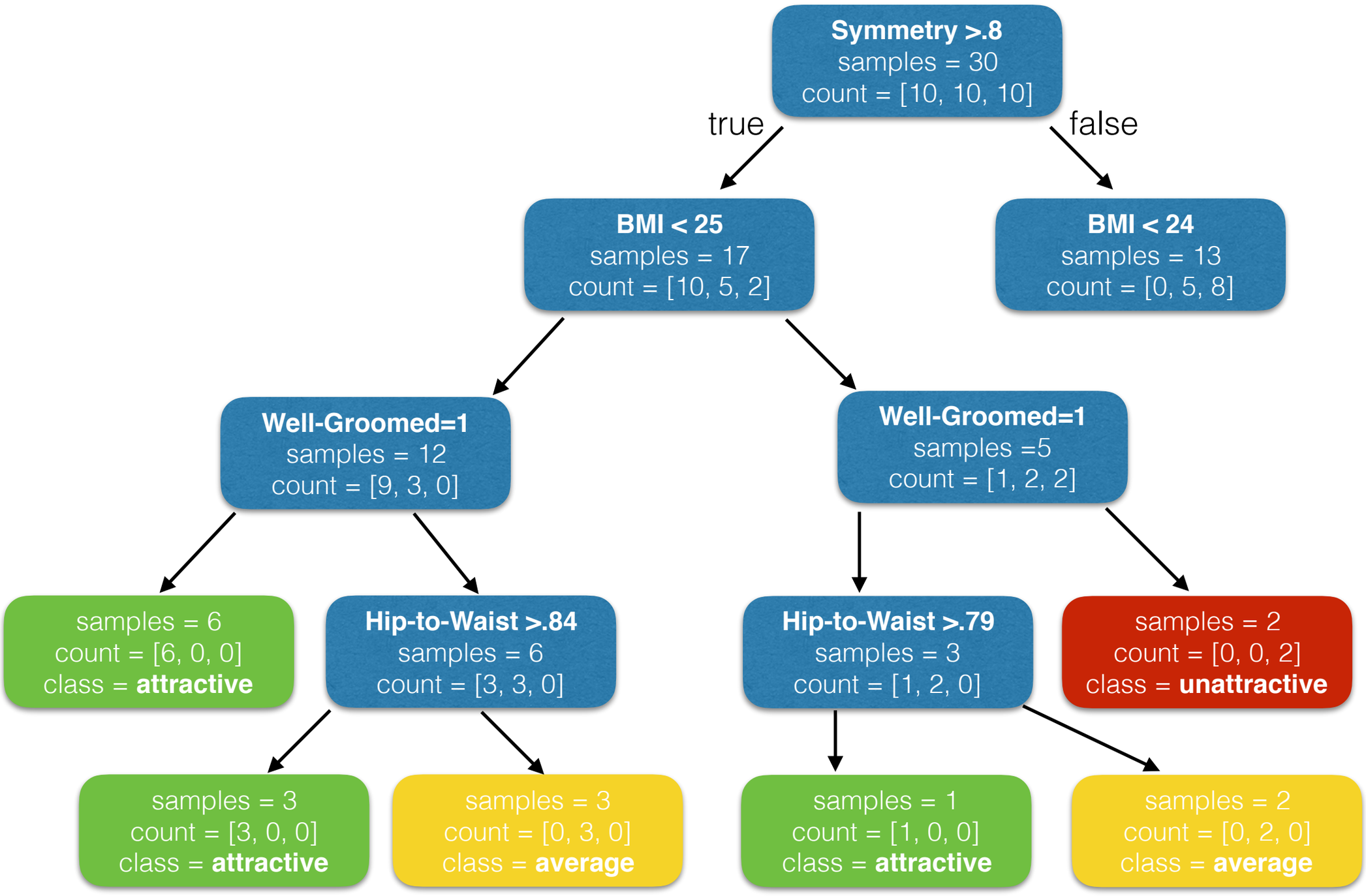
[att, ave, un]


Symmetry >.8
samples = 30
count = [10, 10, 10]
true
false
BMI < 25
samples = 17
count = [10, 5, 2]
BMI < 24
samples = 13
count = [0, 5, 8]
Well-Groomed=1
samples = 12
count = [9, 3, 0]
Well-Groomed=1
samples =5
count = [1, 2, 2]

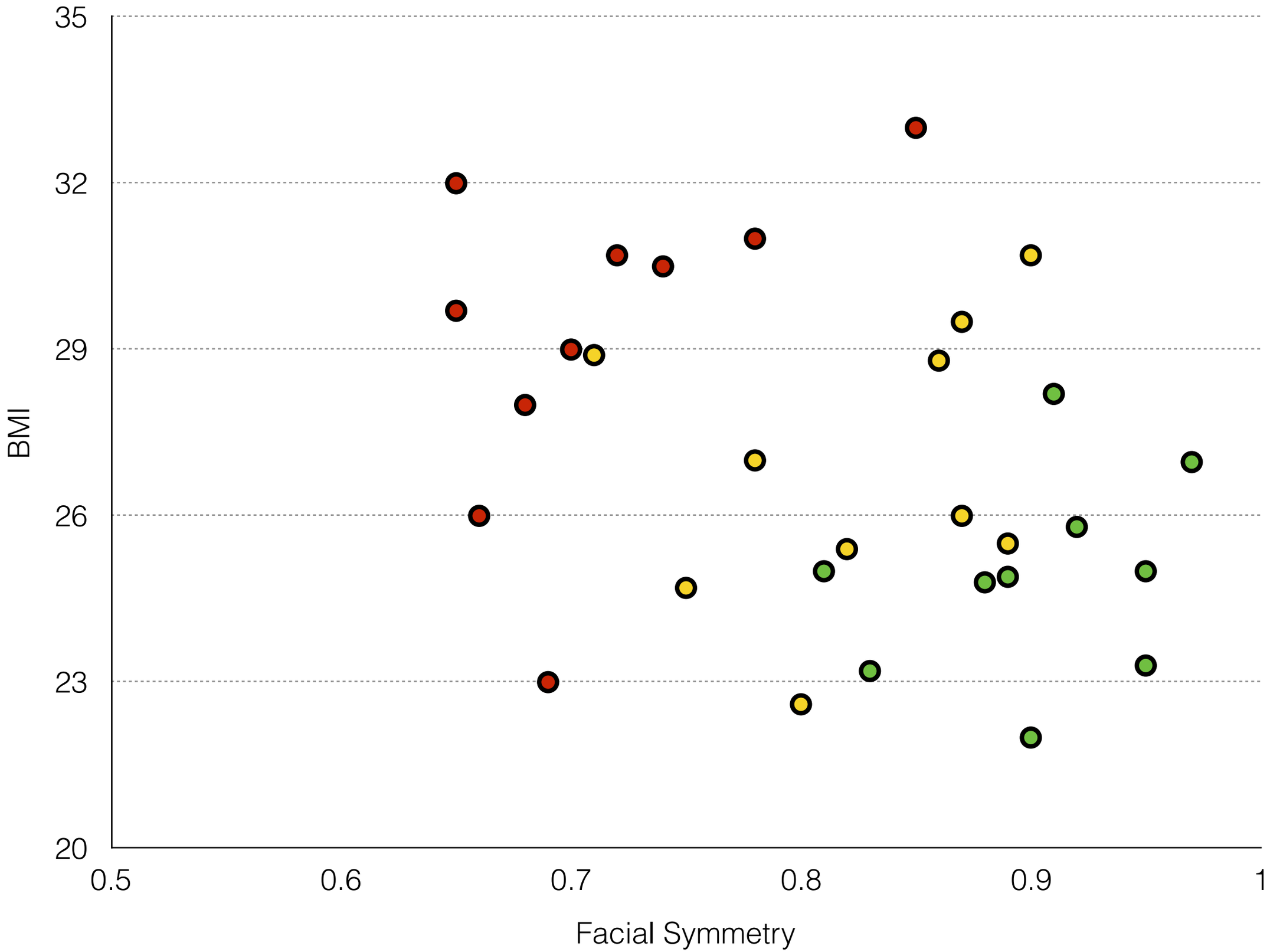
[att, ave, un]

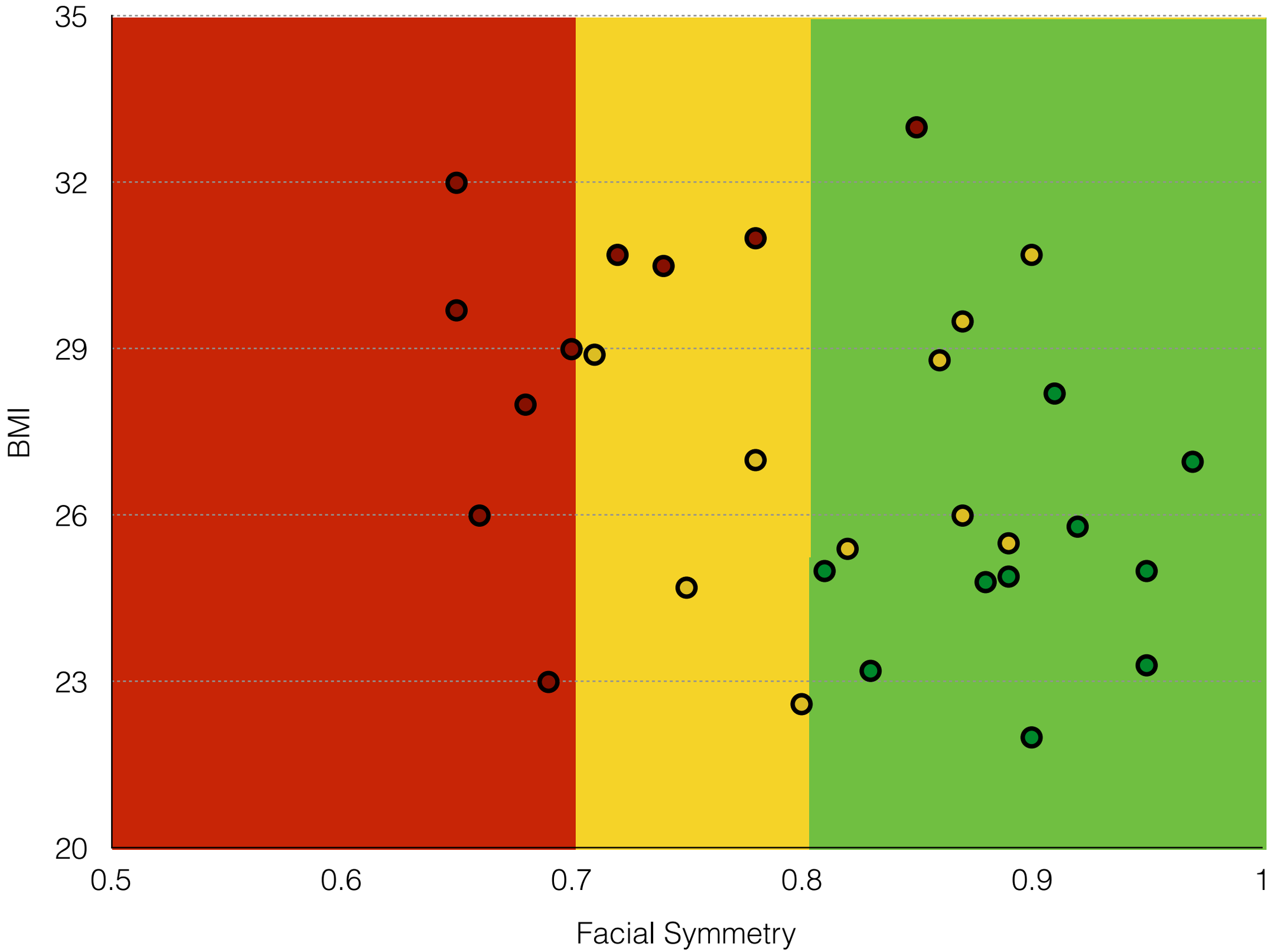

Symmetry >.8
samples = 30
count = [10, 10, 10]
true
false
BMI < 25
samples = 17
count = [10, 5, 2]
BMI < 24
samples = 13
count = [0, 5, 8]
Well-Groomed=1
samples = 12
count = [9, 3, 0]
Well-Groomed=1
samples =5
count = [1, 2, 2]
samples = 6
count = [6, 0, 0]
class = attractive
Hip-to-Waist >.84
samples = 6
count = [3, 3, 0]
Hip-to-Waist >.79
samples = 3
count = [1, 2, 0]
samples = 2
count = [0, 0, 2]
class = unattractive

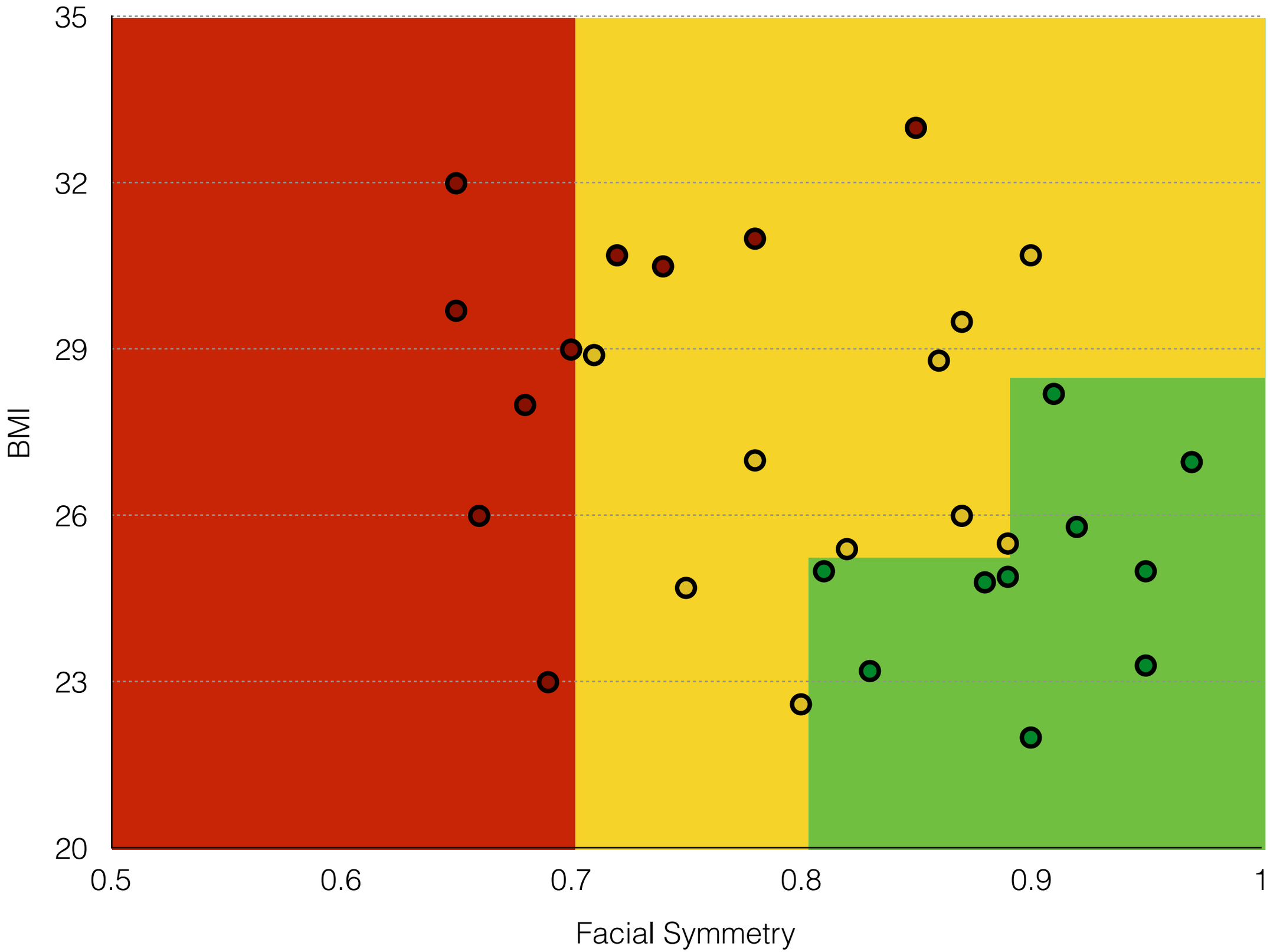
[att, ave, un]

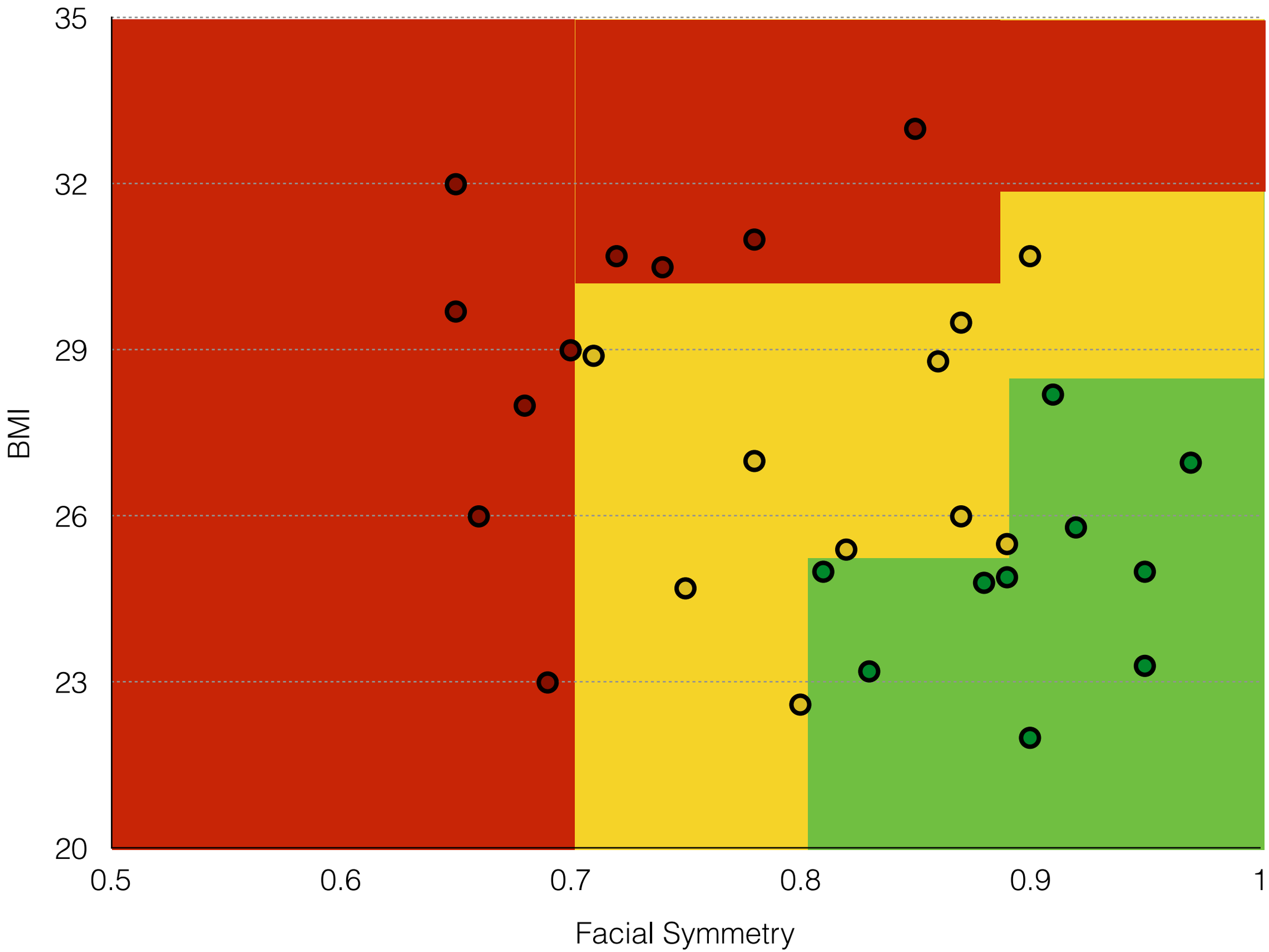

Symmetry >.8
samples = 30
count = [10, 10, 10]
true
false
BMI < 25
samples = 17
count = [10, 5, 2]
BMI < 24
samples = 13
count = [0, 5, 8]
Well-Groomed=1
samples = 12
count = [9, 3, 0]
Well-Groomed=1
samples =5
count = [1, 2, 2]
samples = 6
count = [6, 0, 0]
class = attractive
Hip-to-Waist >.84
samples = 6
count = [3, 3, 0]
Hip-to-Waist >.79
samples = 3
count = [1, 2, 0]
samples = 2
count = [0, 0, 2]
class = unattractive
samples = 3
count = [3, 0, 0]
class = attractive
samples = 3
count = [0, 3, 0]
class = average
samples = 1
count = [1, 0, 0]
class = attractive
samples = 2
count = [0, 2, 0]
class = average

[att, ave, un]


35
32
29
26
23
20
BMI
0.5
0.6
0.7
0.8
0.9
1
Facial Symmetry


35
32
29
26
23
20
BMI
0.5
0.6
0.7
0.8
0.9
1
Facial Symmetry


35
32
29
26
23
20
BMI
0.5
0.6
0.7
0.8
0.9
1
Facial Symmetry


35
32
29
26
23
20
BMI
0.5
0.6
0.7
0.8
0.9
1
Facial Symmetry

K-nearest Neighbor

Euclidean Distance

point a = [a1, a2]

point b = [b1, b2]

Euclidean Distance

point a = [a1, a2]

point b = [b1, b2]

Two dimensions (features)

$$\sqrt{(a_1-b_1)^2+(a_2-b_2)^2}$$

	Number of Relationships	Grade of First Relationship
sample a	3	7
sample b	6	11
sample c	3	9
sample d	5	10

Euclidean Distance

Feature Vector

a = [3, 7]

b = [6, 11]

Euclidean Distance

$$\sqrt{(3-6)^2+(7-11)^2}$$

Feature Vector

a = [3, 7]

b = [6, 11]

Euclidean Distance

$$\sqrt{(3-6)^2+(7-11)^2}$$

$$\sqrt{(-3)^2+(-4)^2}$$

Feature Vector

a = [3, 7]

b = [6, 11]

Euclidean Distance

Feature Vector

a = [3, 7]

b = [6, 11]

$$\sqrt{(3-6)^2+(7-11)^2}$$

$$\sqrt{(-3)^2+(-4)^2}$$

$$\sqrt{9+16}$$

Euclidean Distance

Feature Vector

a = [3, 7]

b = [6, 11]

$$\sqrt{(3-6)^2+(7-11)^2}$$

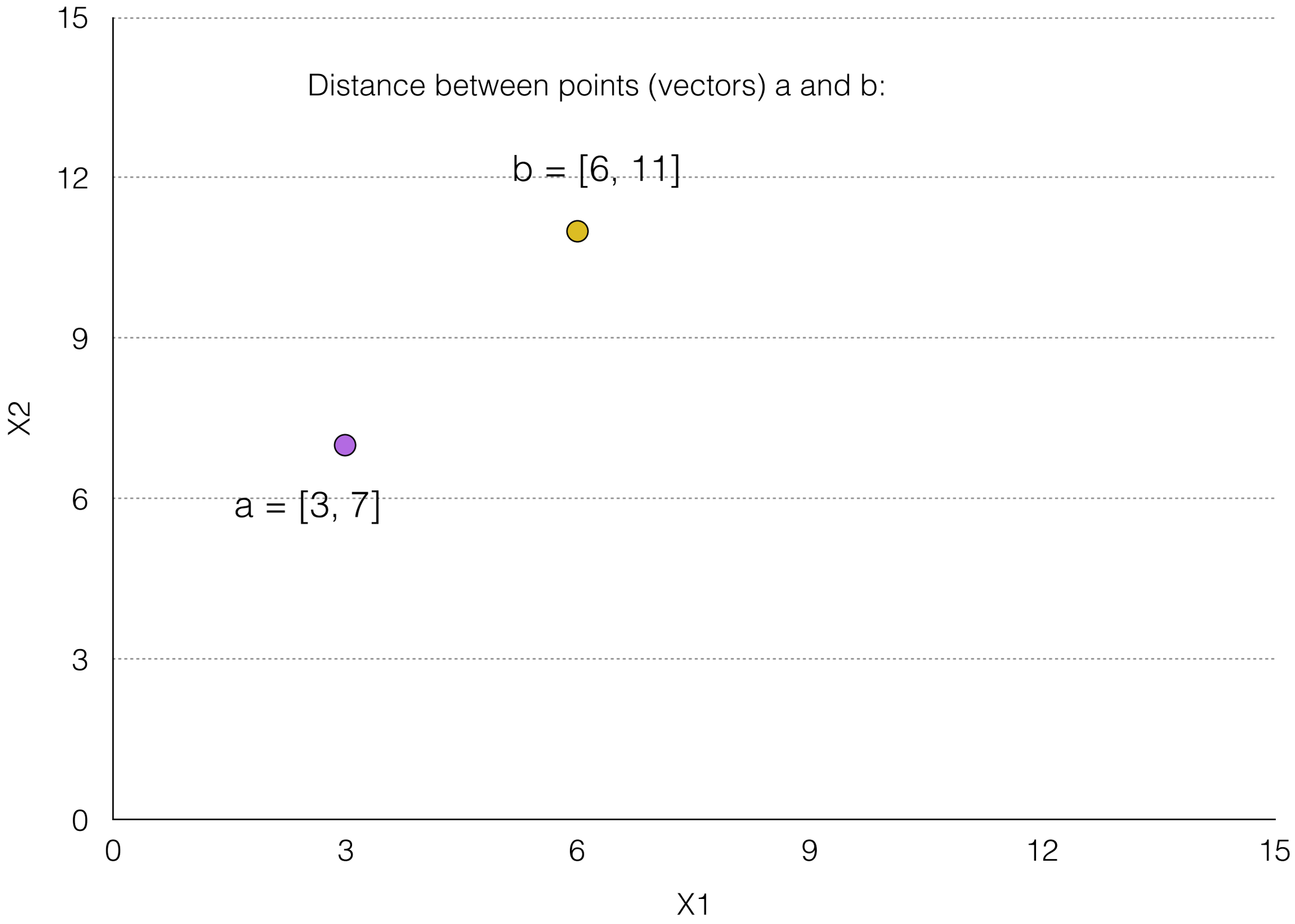
$$\sqrt{(-3)^2+(-4)^2}$$

$$\sqrt{9+16}$$

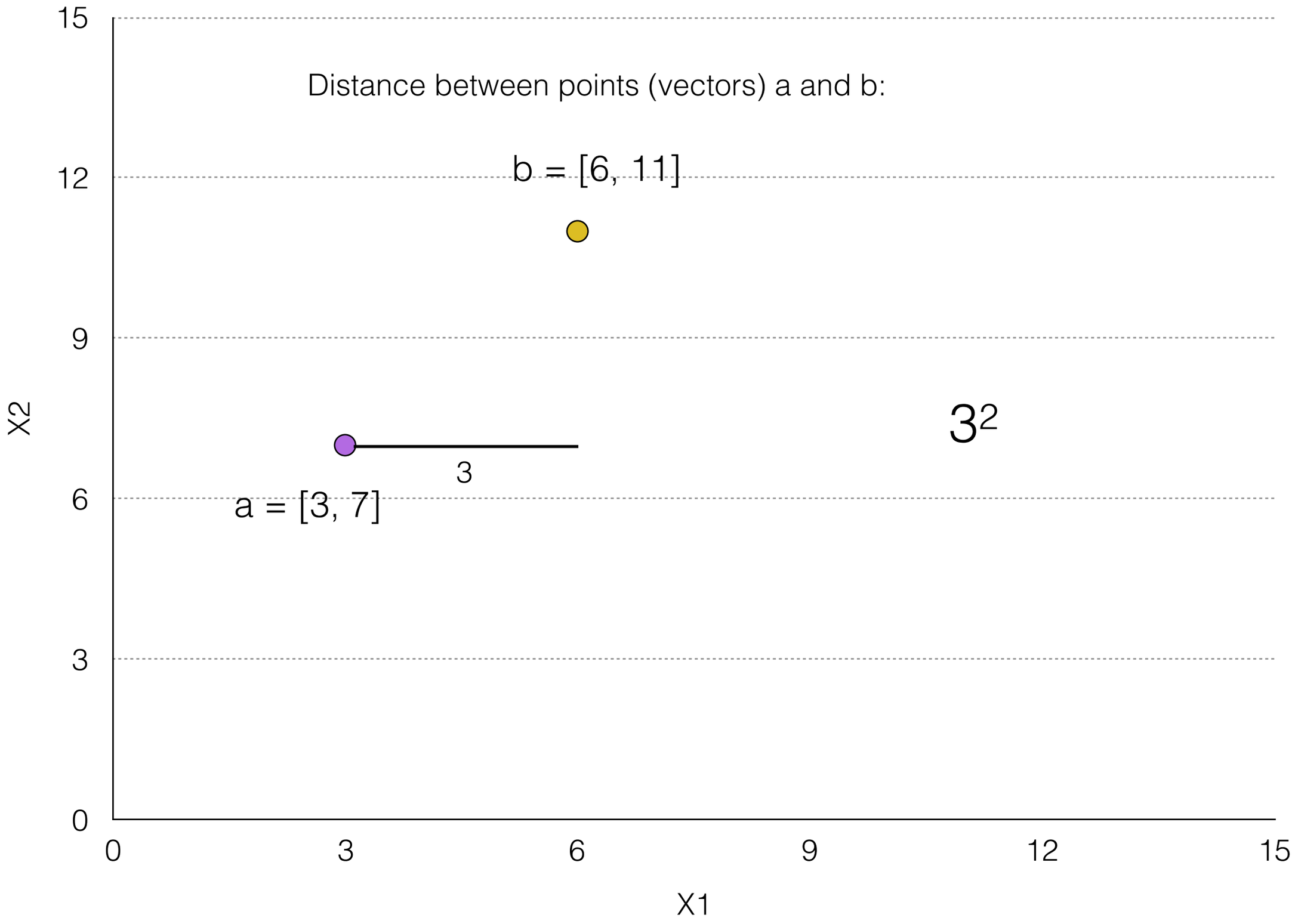
Distance between points (vectors) a and b:

$$\sqrt{25}=5$$

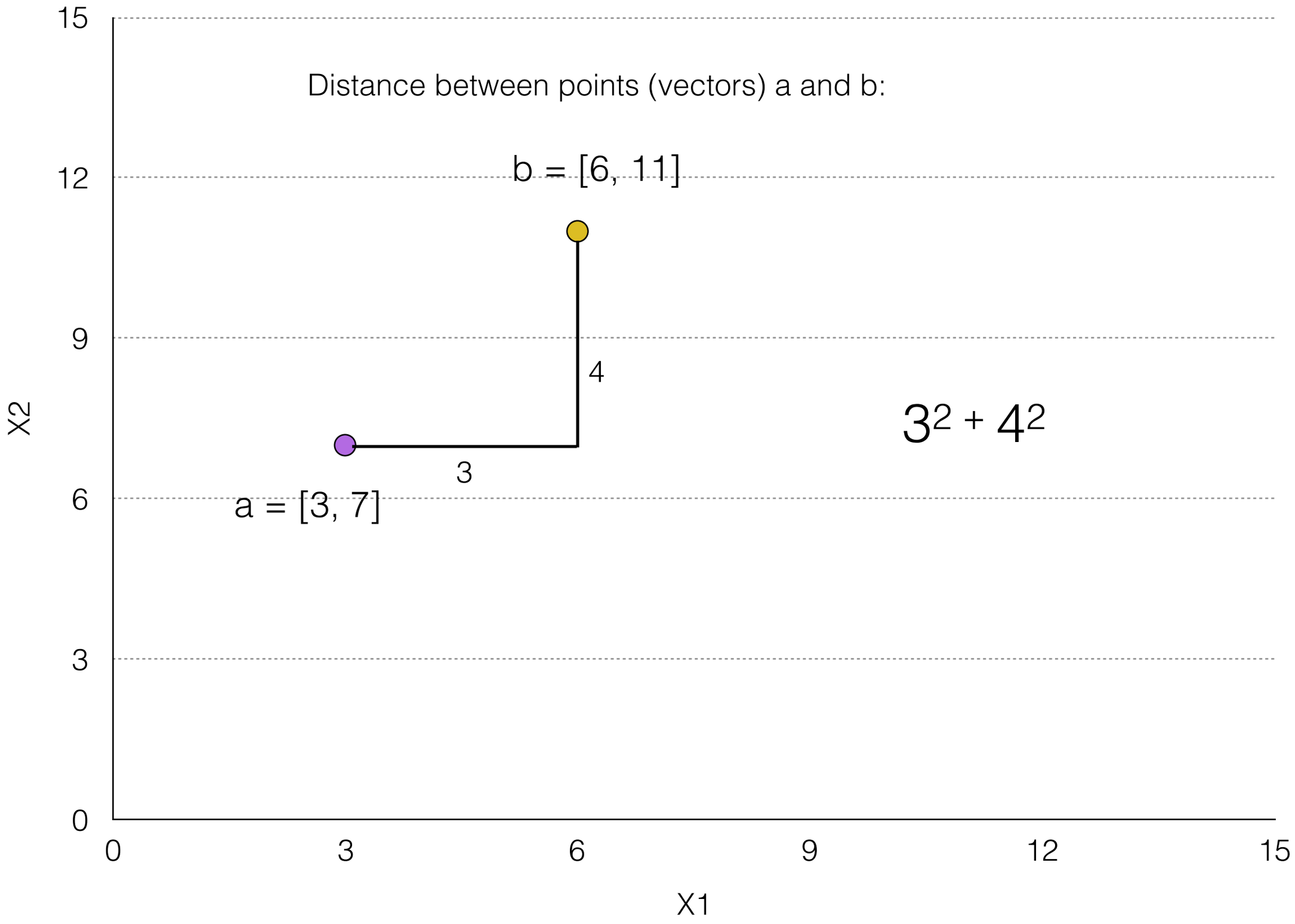
Euclidean Distance



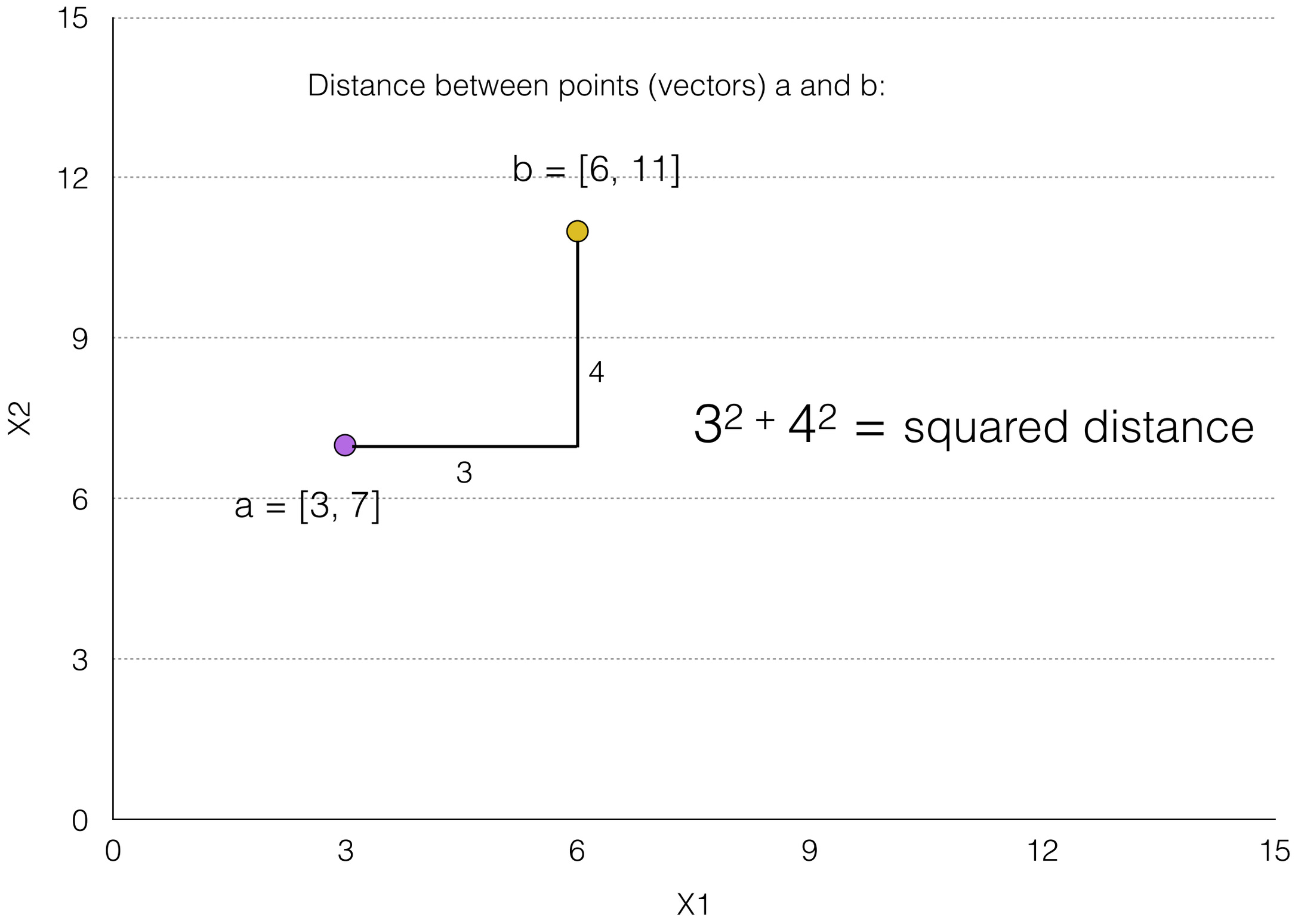
Euclidean Distance



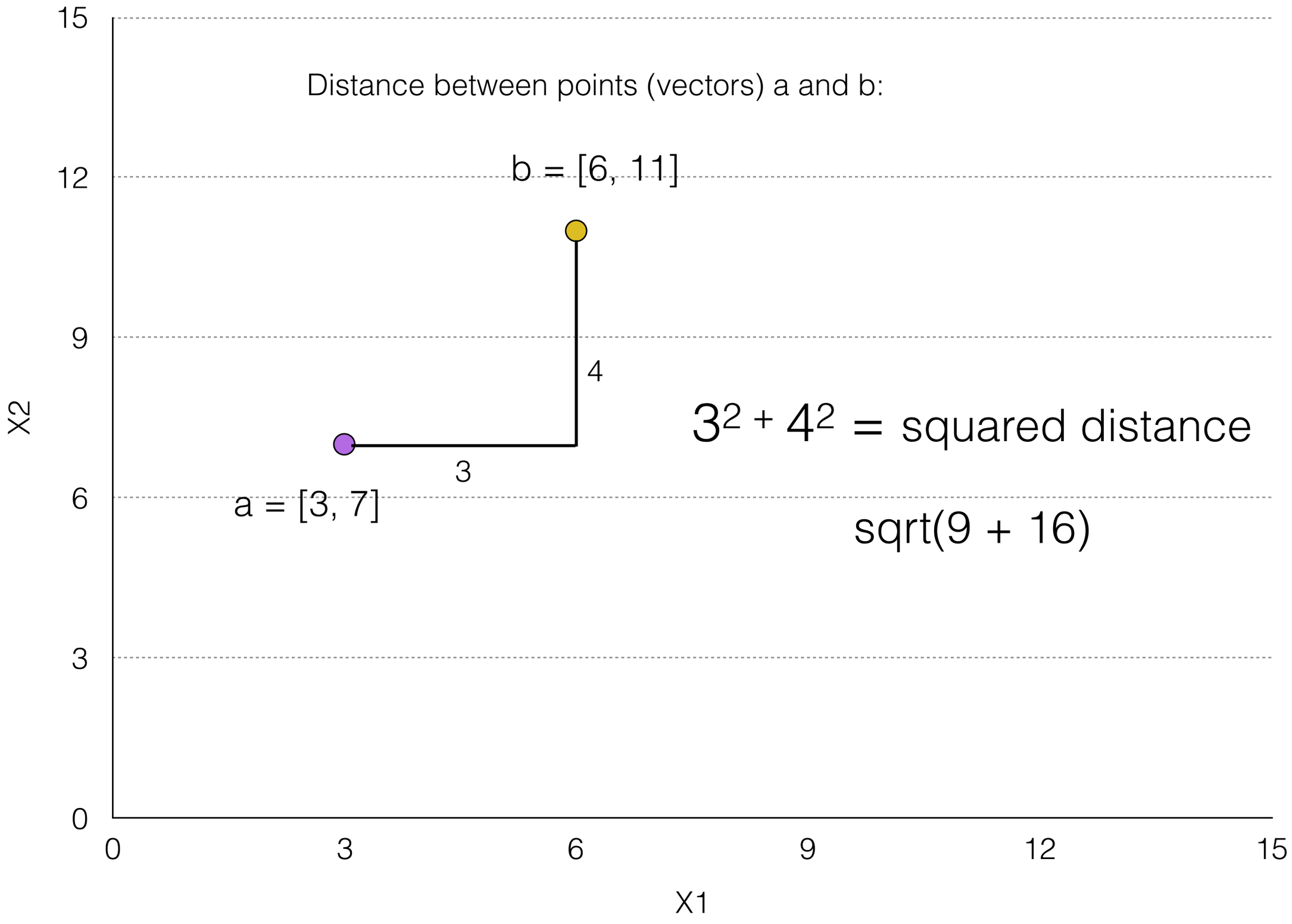
Euclidean Distance



Euclidean Distance



Euclidean Distance



Euclidean Distance

Distance between points (vectors) a and b:

b = [6, 11]

distance = **5**

4

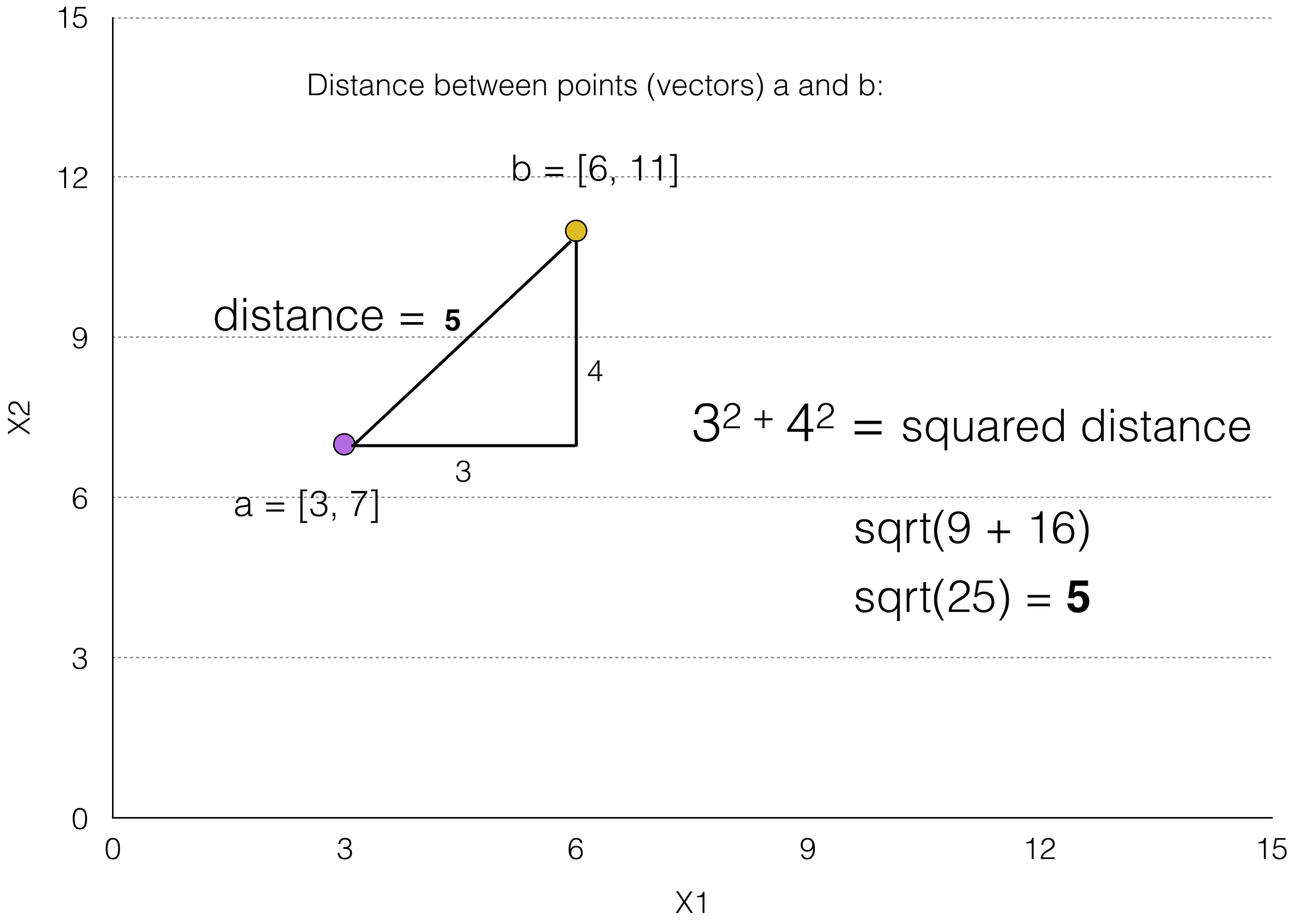
3

a = [3, 7]

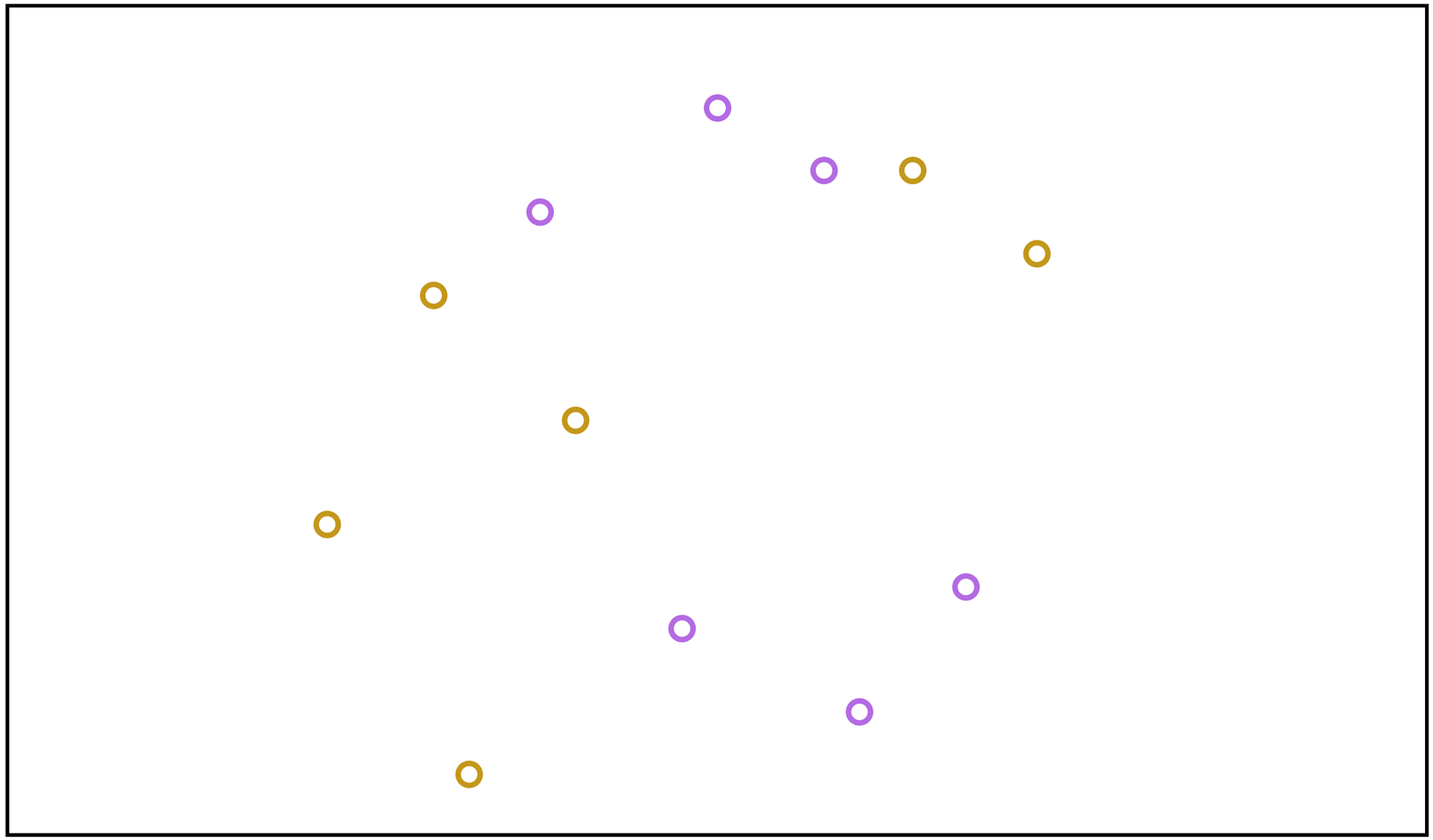
$3^2 + 4^2$ = squared distance

sqrt(9 + 16)

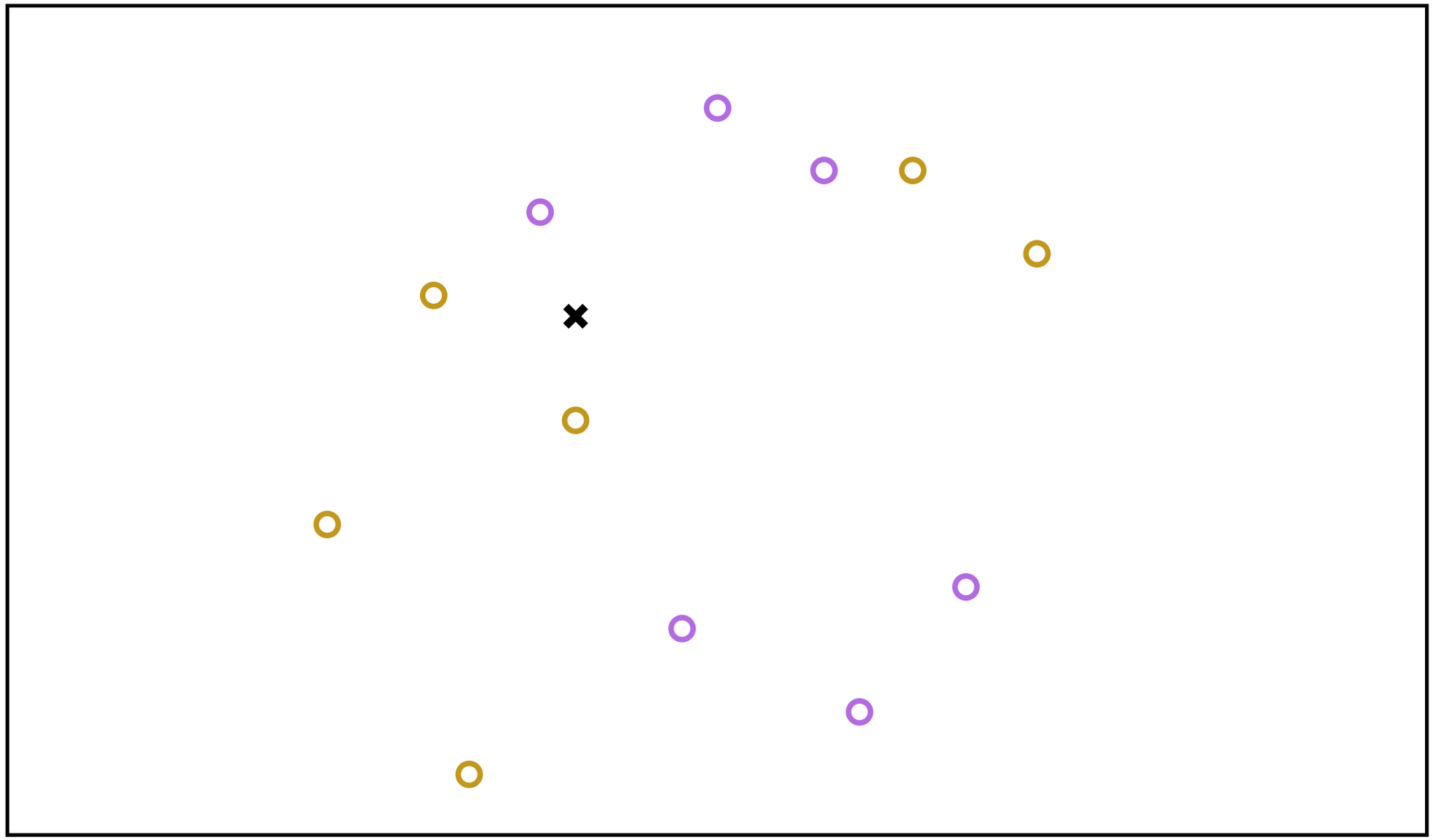
sqrt(25) = **5**



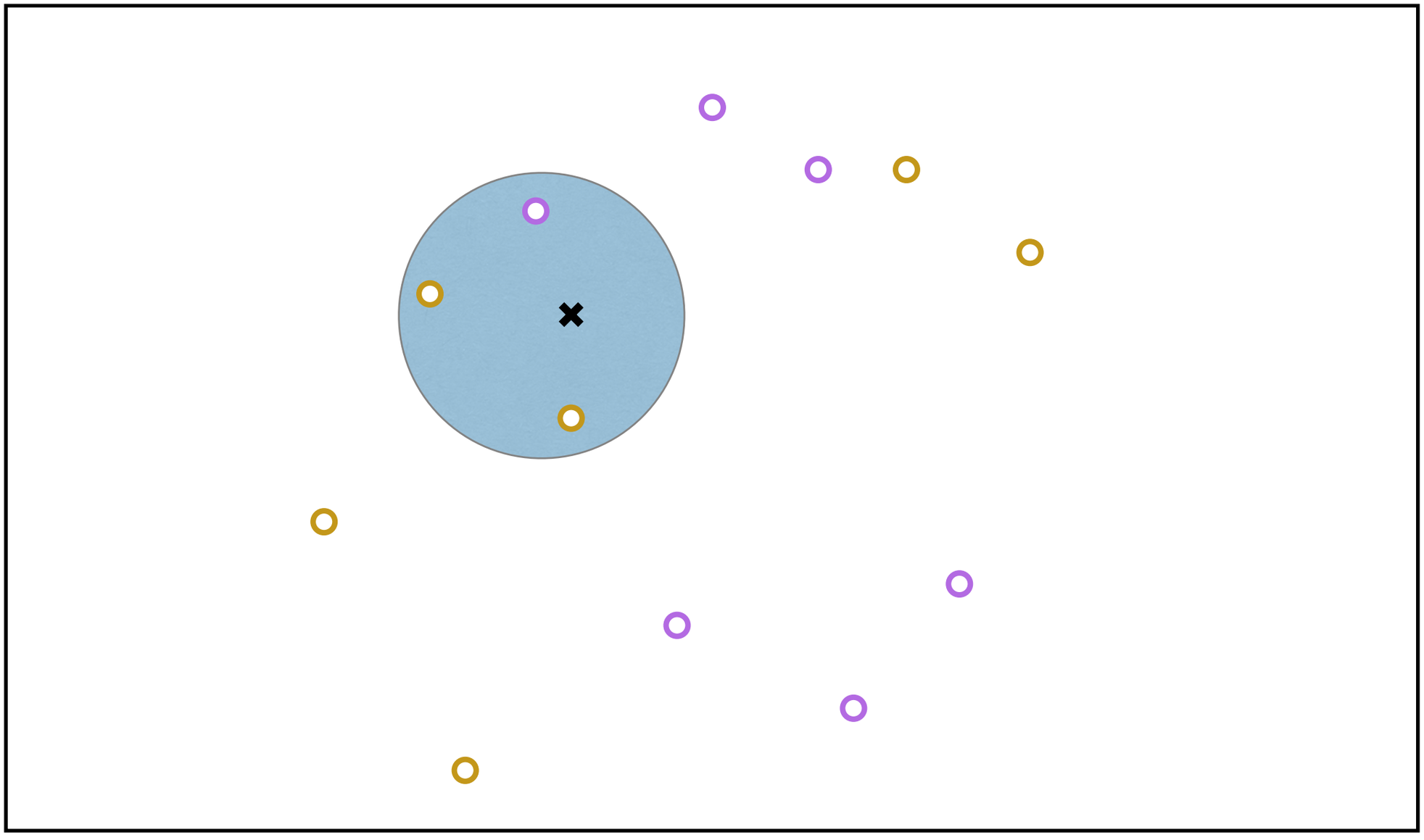
K-nearest Neighbor



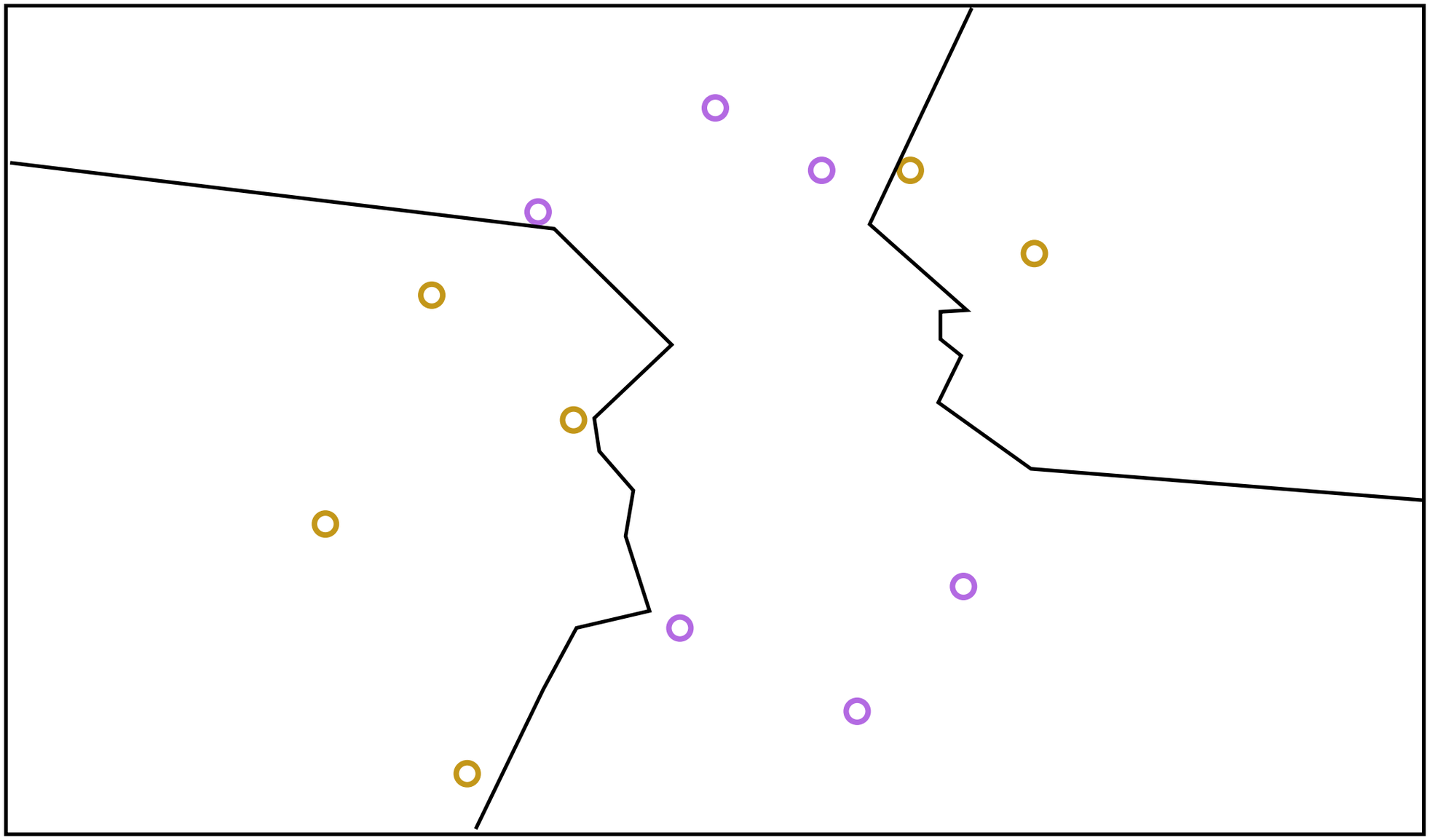
K-nearest Neighbor



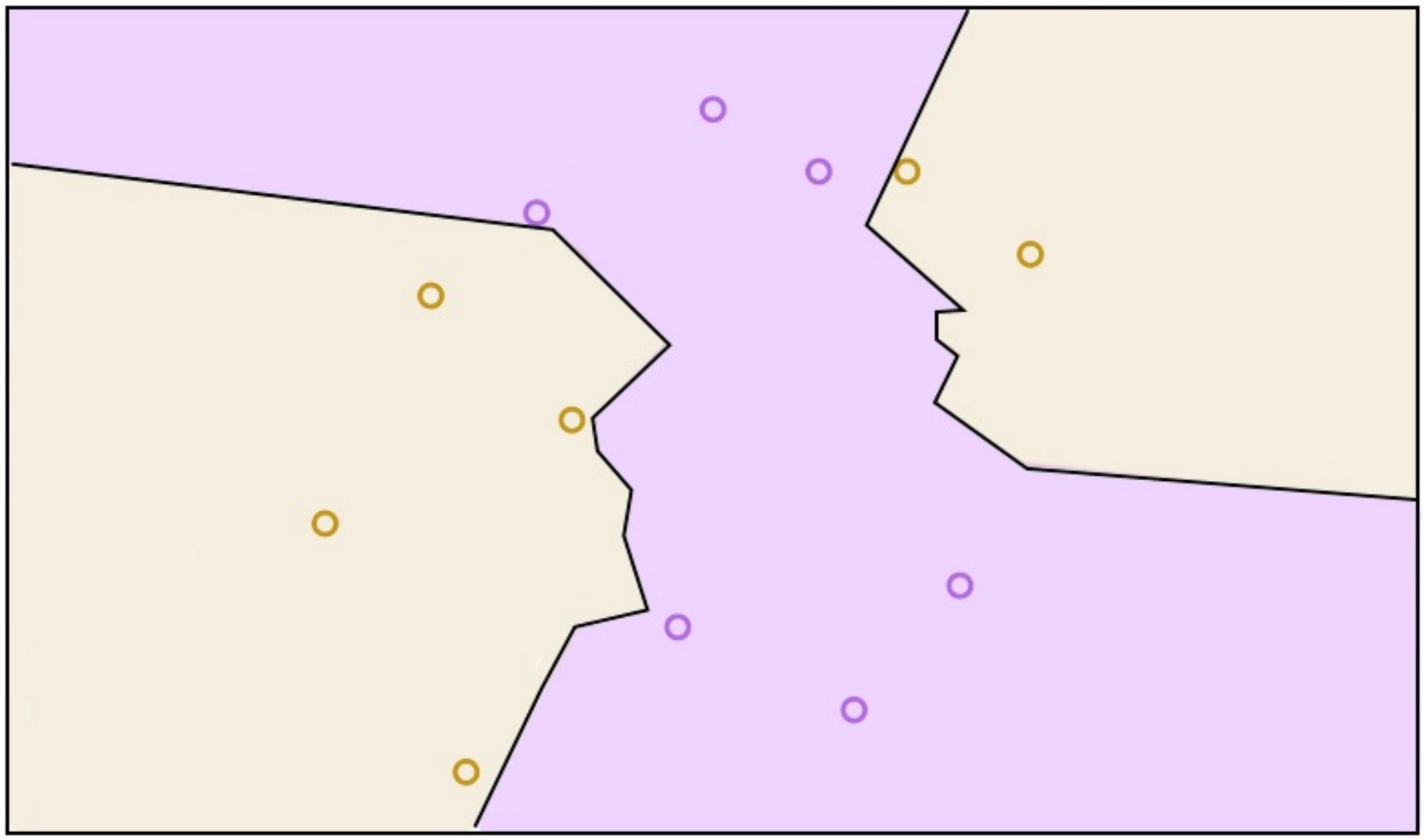
K = 3

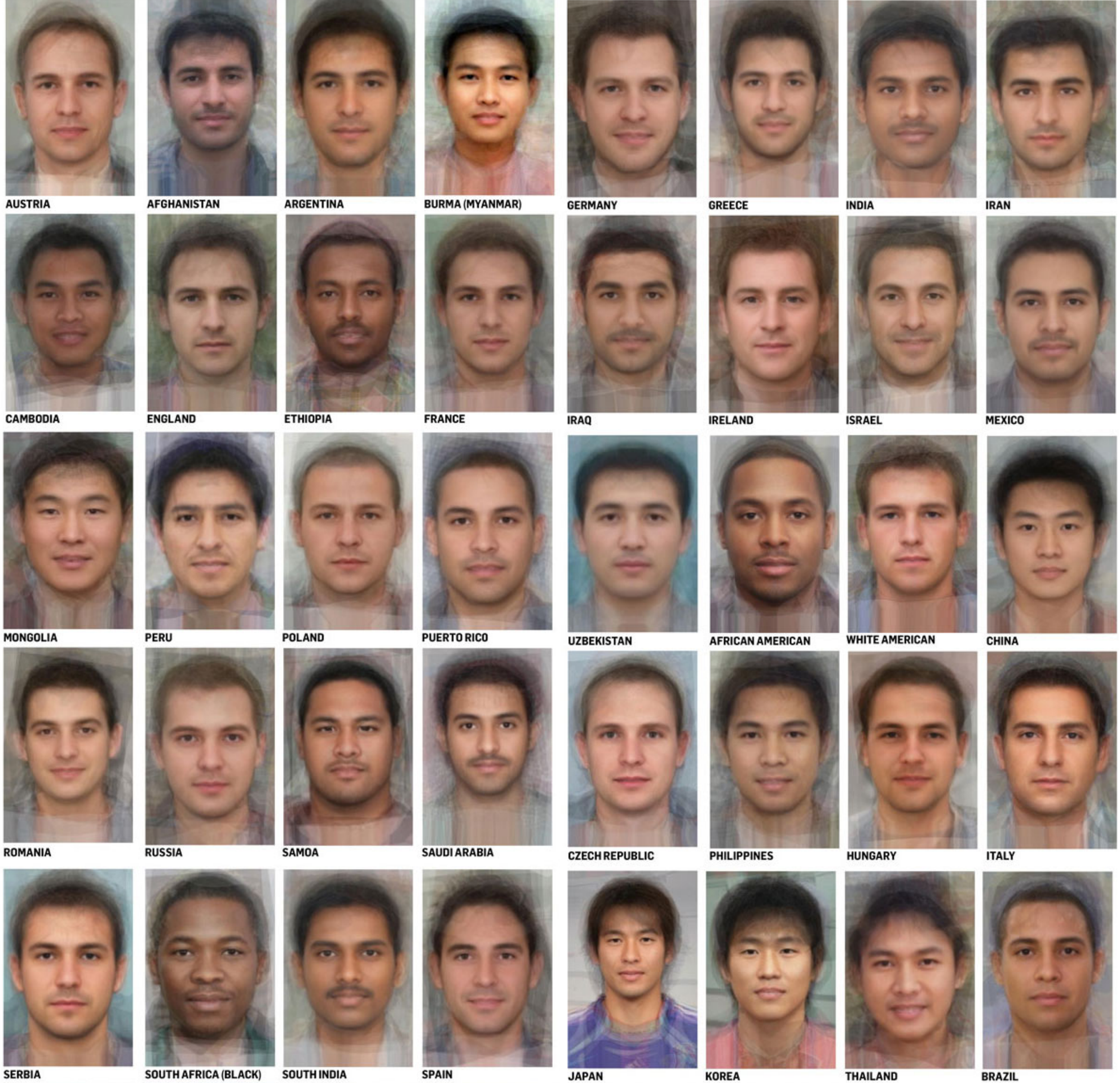


K = 3



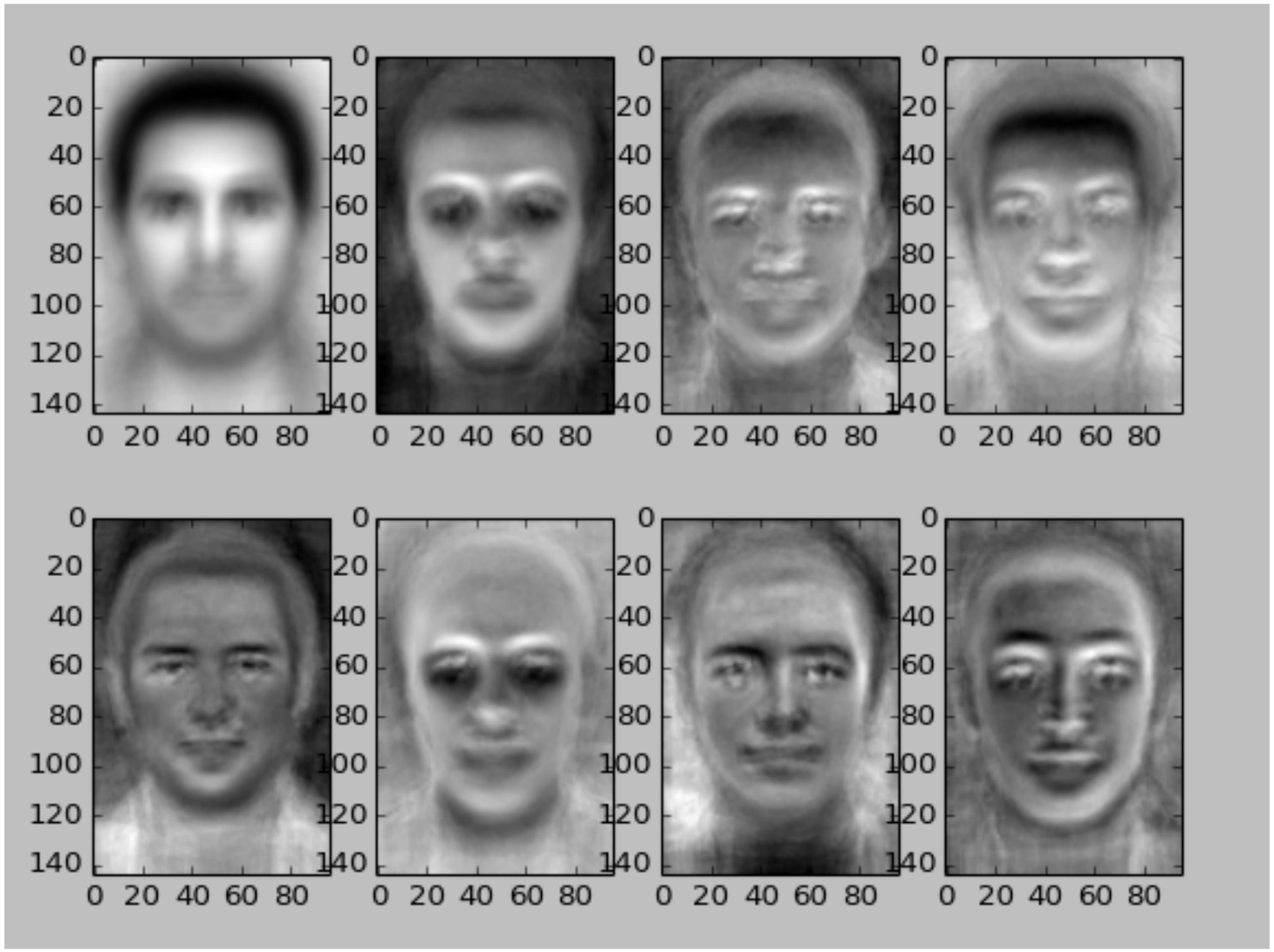
K = 3

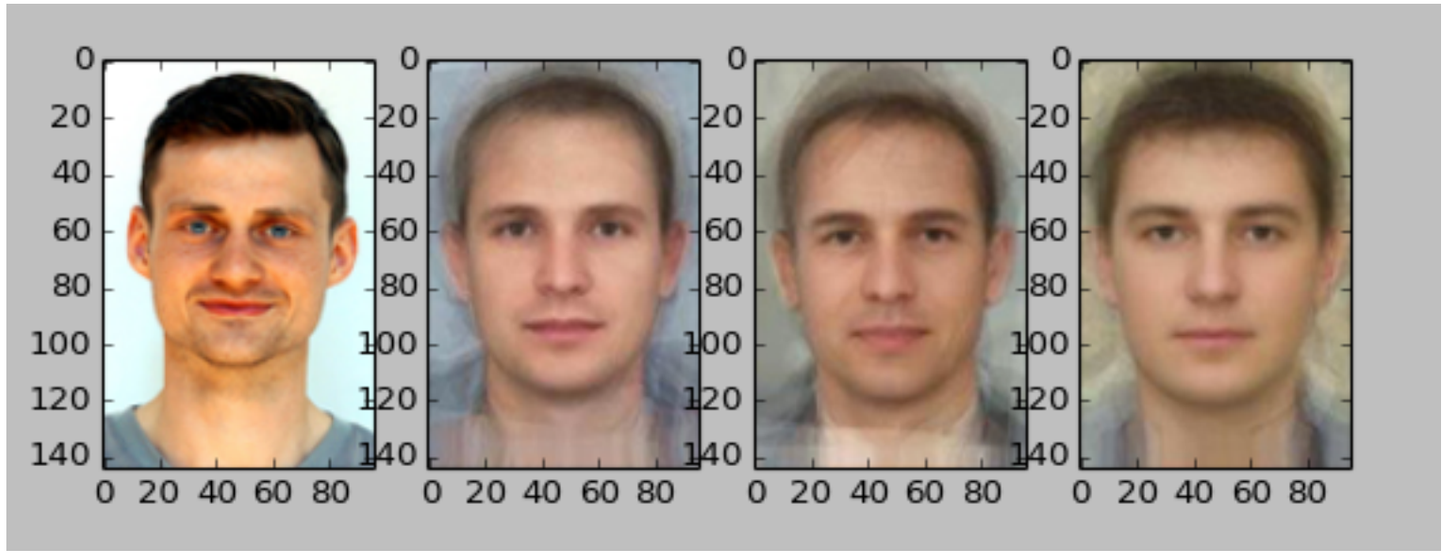



AUSTRIA
AFGHANISTAN
ARGENTINA
BURMA (MYANMAR)
GERMANY
GREECE
INDIA
IRAN
CAMBODIA
ENGLAND
ETHIOPIA
FRANCE
IRAQ
IRELAND
ISRAEL
MEXICO
MONGOLIA
PERU
POLAND
PUERTO RICO
UZBEKISTAN
AFRICAN AMERICAN
WHITE AMERICAN
CHINA
ROMANIA
RUSSIA
SAMOA
SAUDI ARABIA
CZECH REPUBLIC
PHILIPPINES
HUNGARY
ITALY
SERBIA
SOUTH AFRICA (BLACK)
SOUTH INDIA
SPAIN
JAPAN
KOREA
THAILAND
BRAZIL

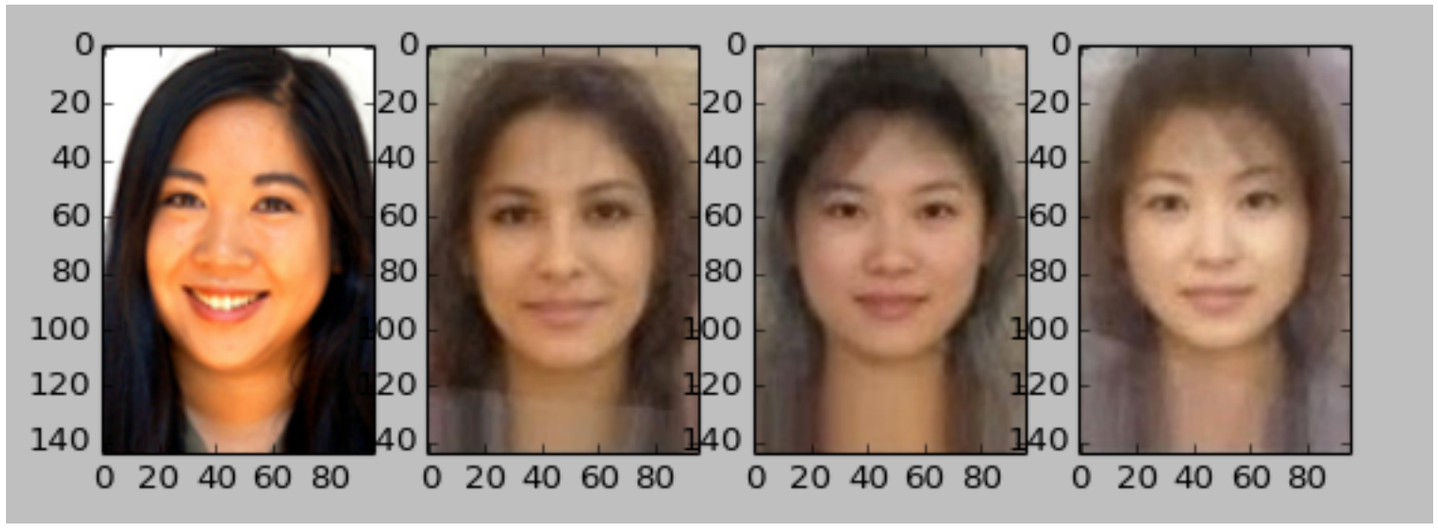

germany
france
england
netherlands
ukraine
sweden
poland
russia
mexico
peru
argentina
brazil
serbia
czech republic
hungary
romania
philippines
cambodia
vietnam
thailand
south india
india
burma
samoa
greece
italy
spain
israel
turkey
lebanon
afghanistan
iran
china (han)
korea
japan
mongolia
chad / cameroon
ethiopia
south africa
west africa

Eigenfaces





Lithuanian Czech Republic Austria Ukraine

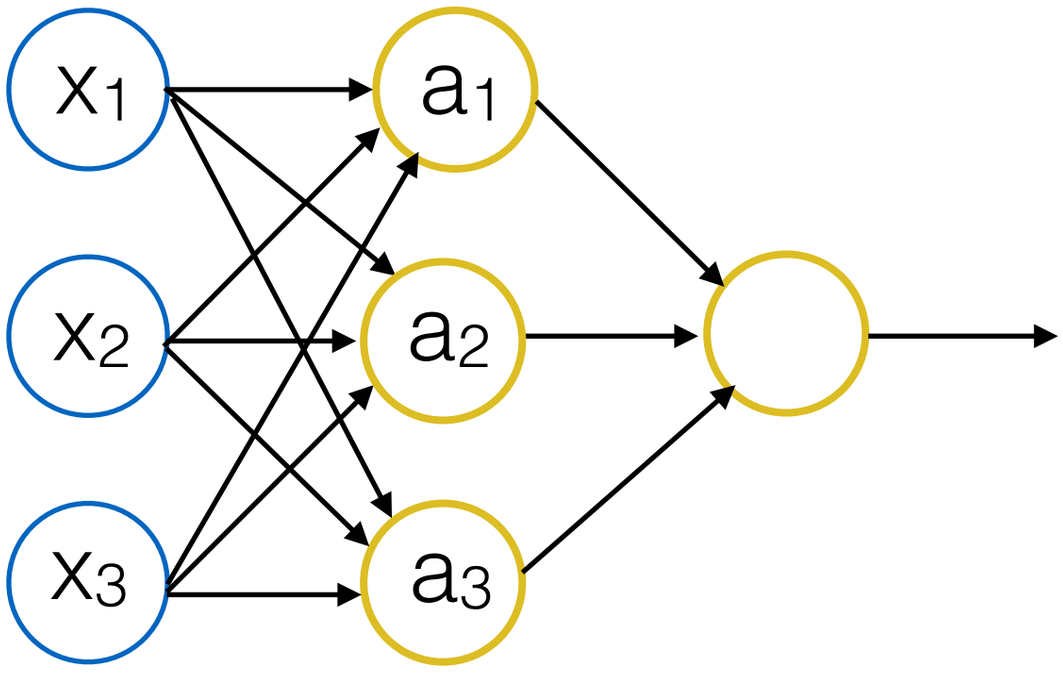


Chinese Peruvian Chinese Japanese

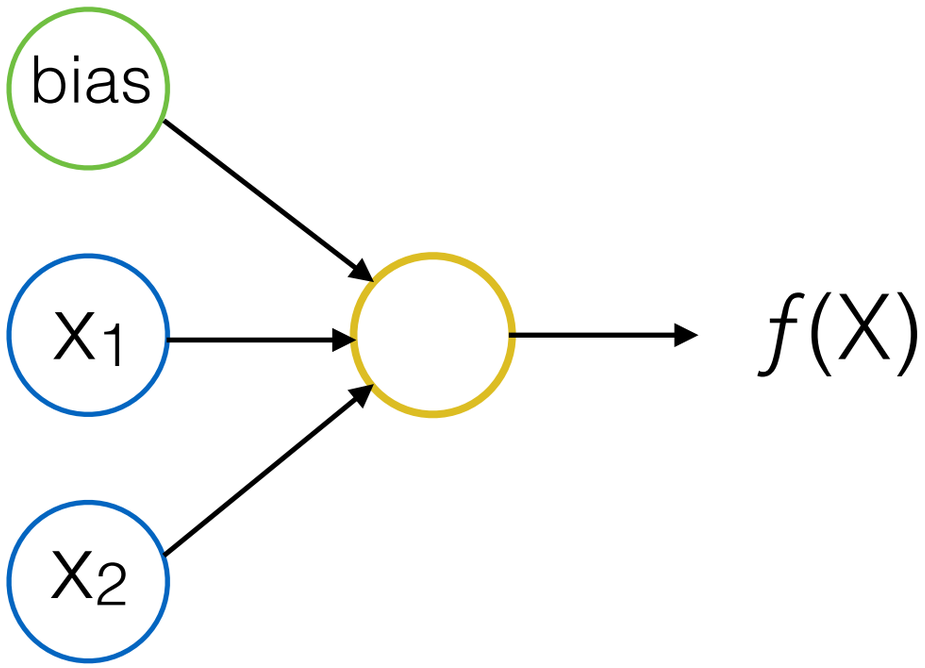
K-Means

Neural Network

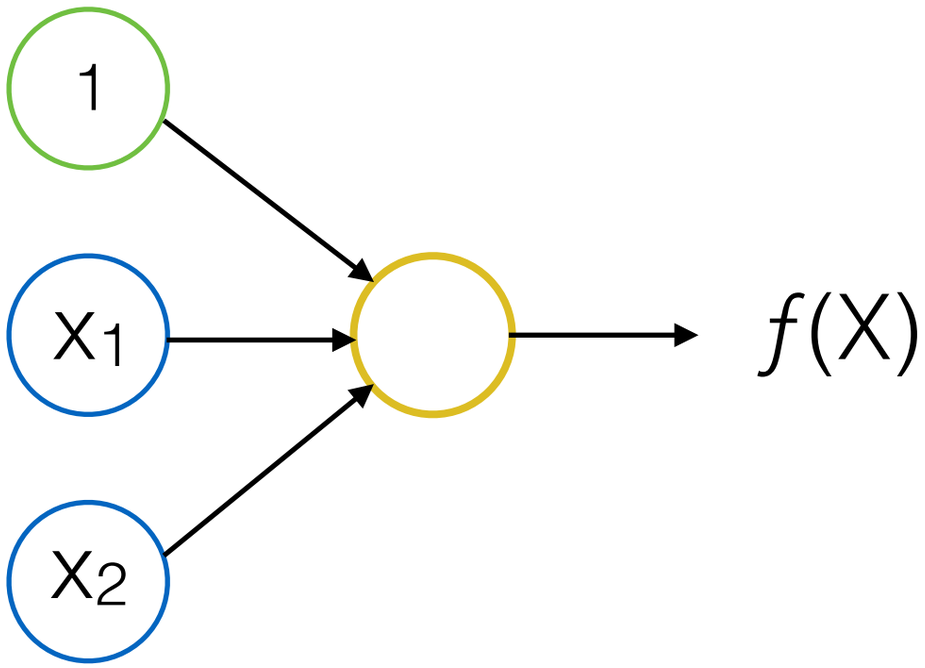
Neural Network



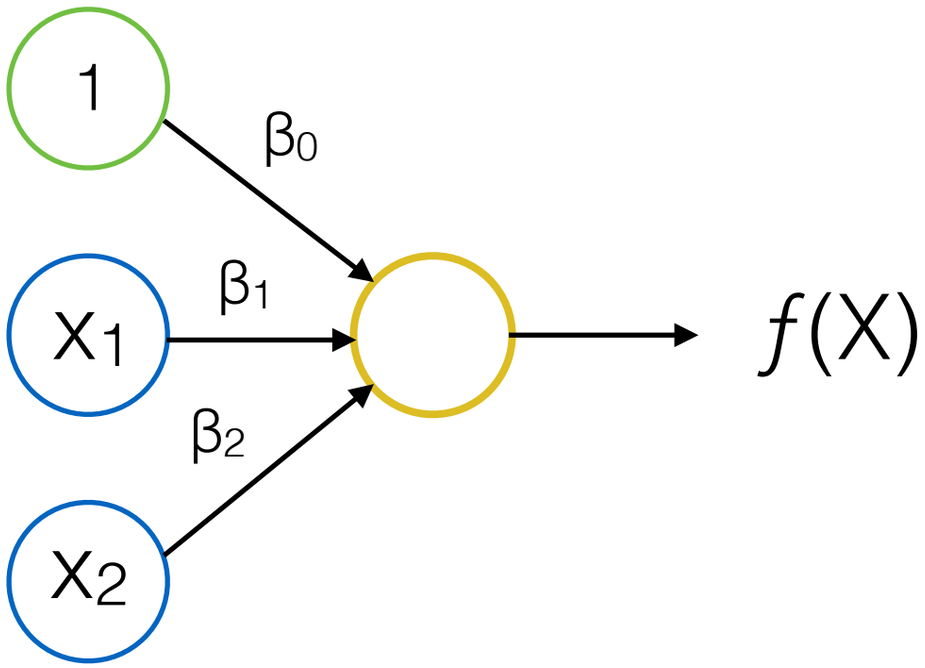
Perceptron



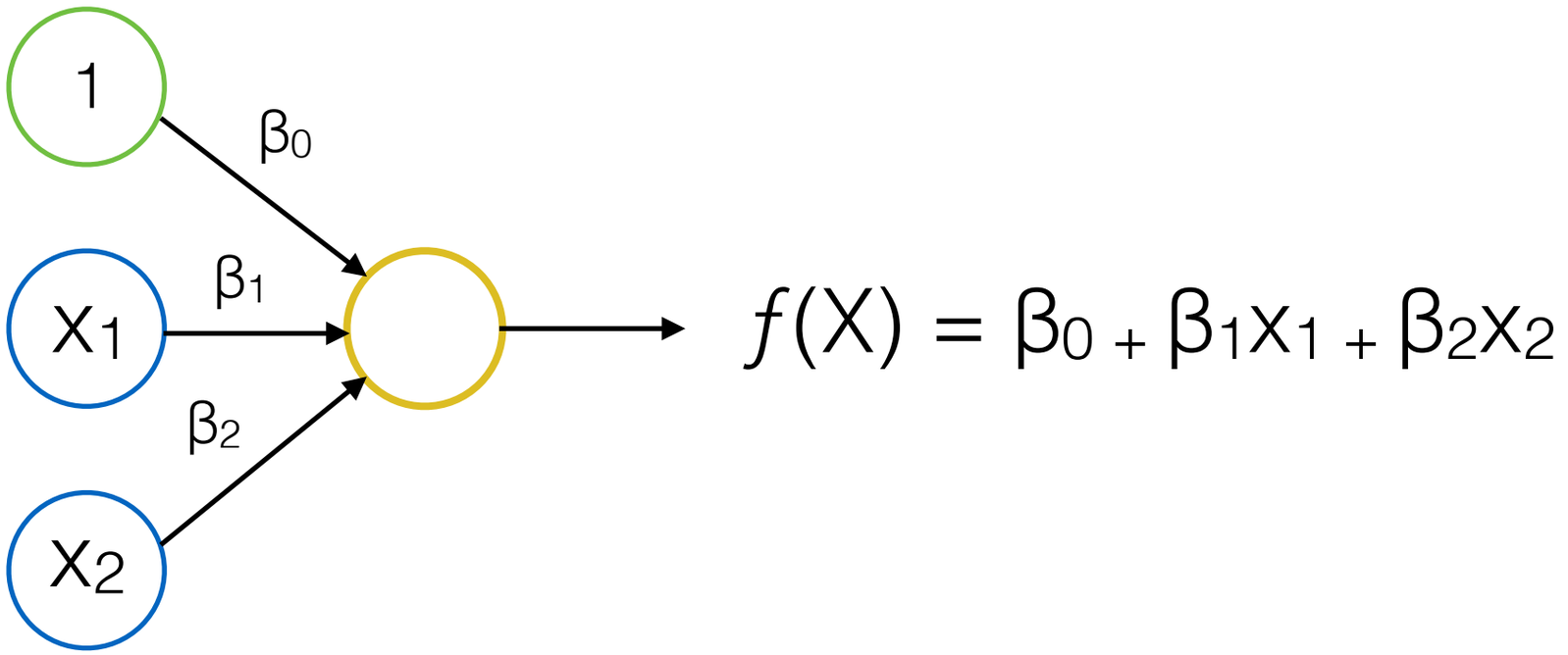
Perceptron



Perceptron



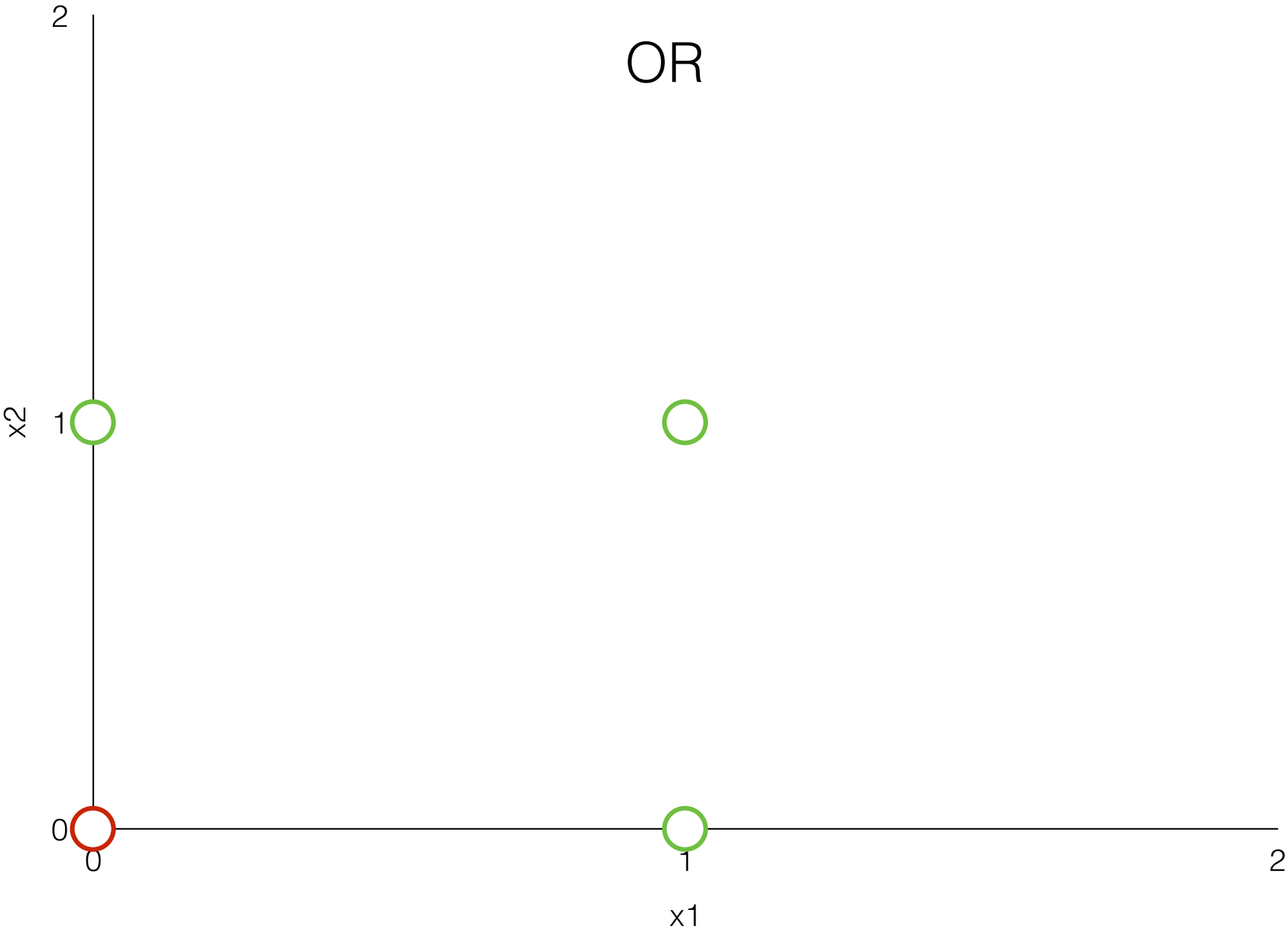
Perceptron



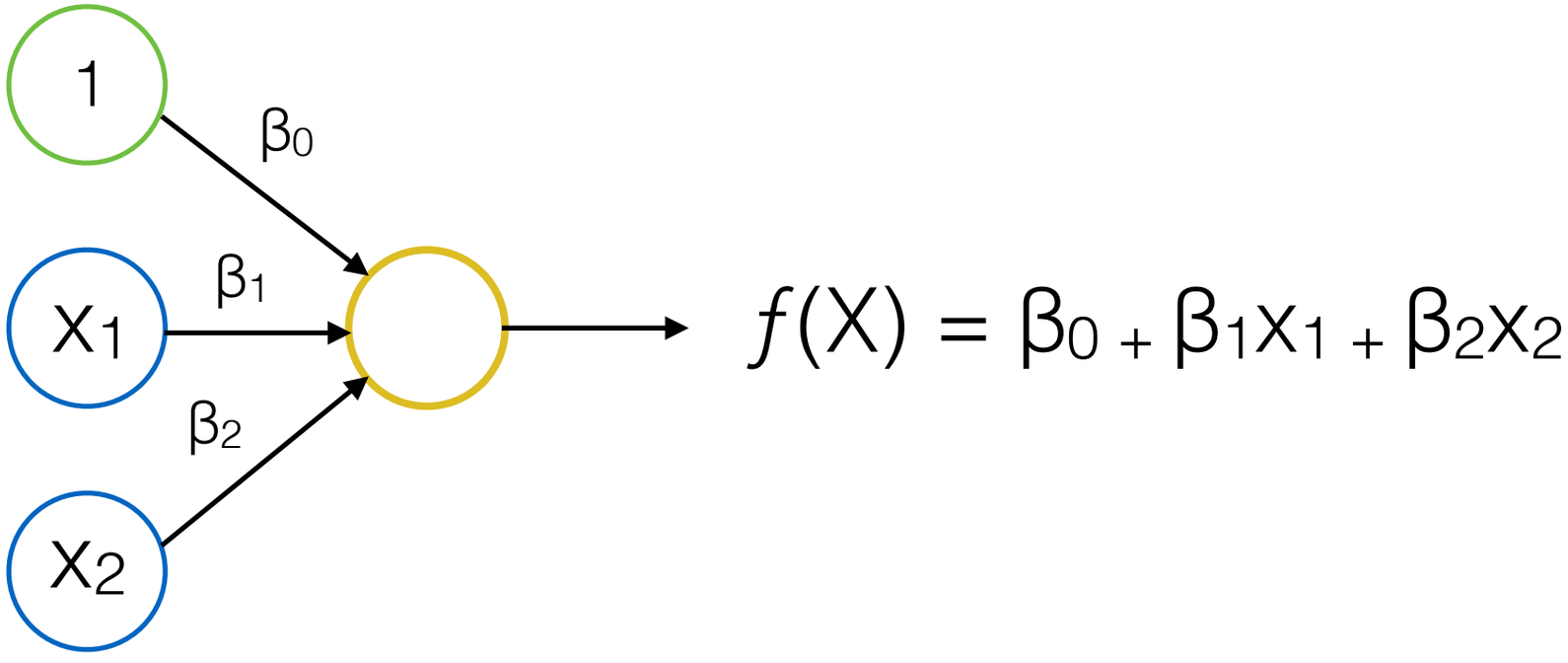
OR

Feature 1	Feature 2	Target
0	0	0
1	0	1
0	1	1
1	1	1

OR



Feature 1	Feature 2	Target
0	0	0
1	0	1
0	1	1
1	1	1



Activation Function: Threshold

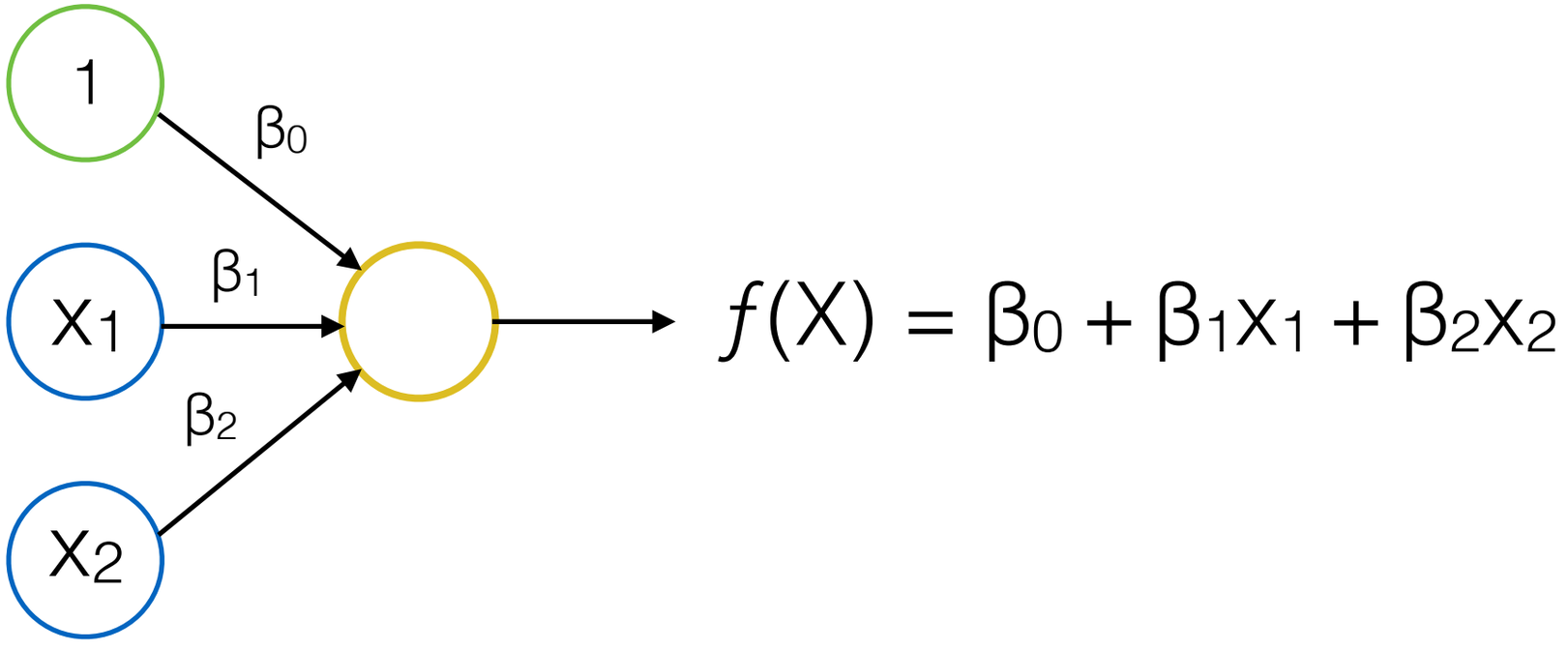
if $\beta_0 + \beta_1 x_1 + \beta_2 x_2 > 0$: 1

Else: 0

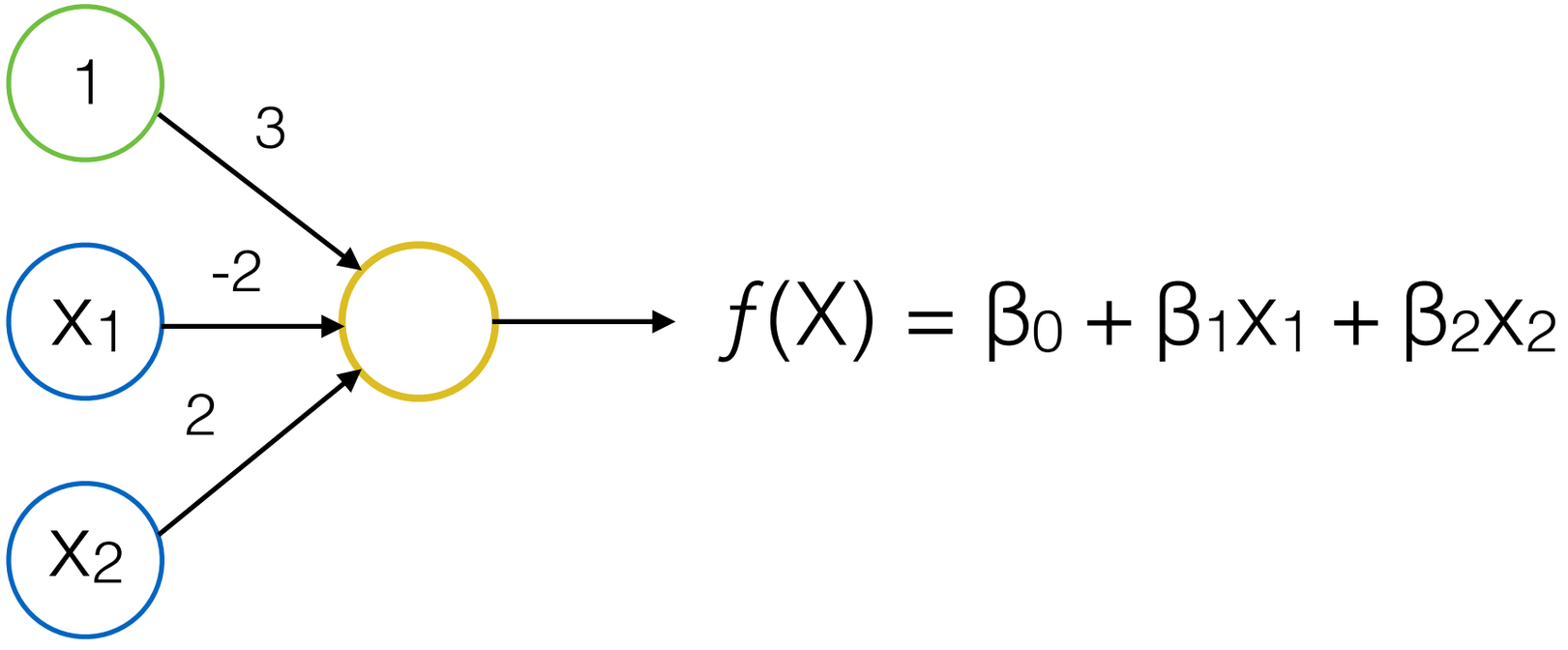
Update Rule:

updated weight_i = weight_i - (output - target) * input_i

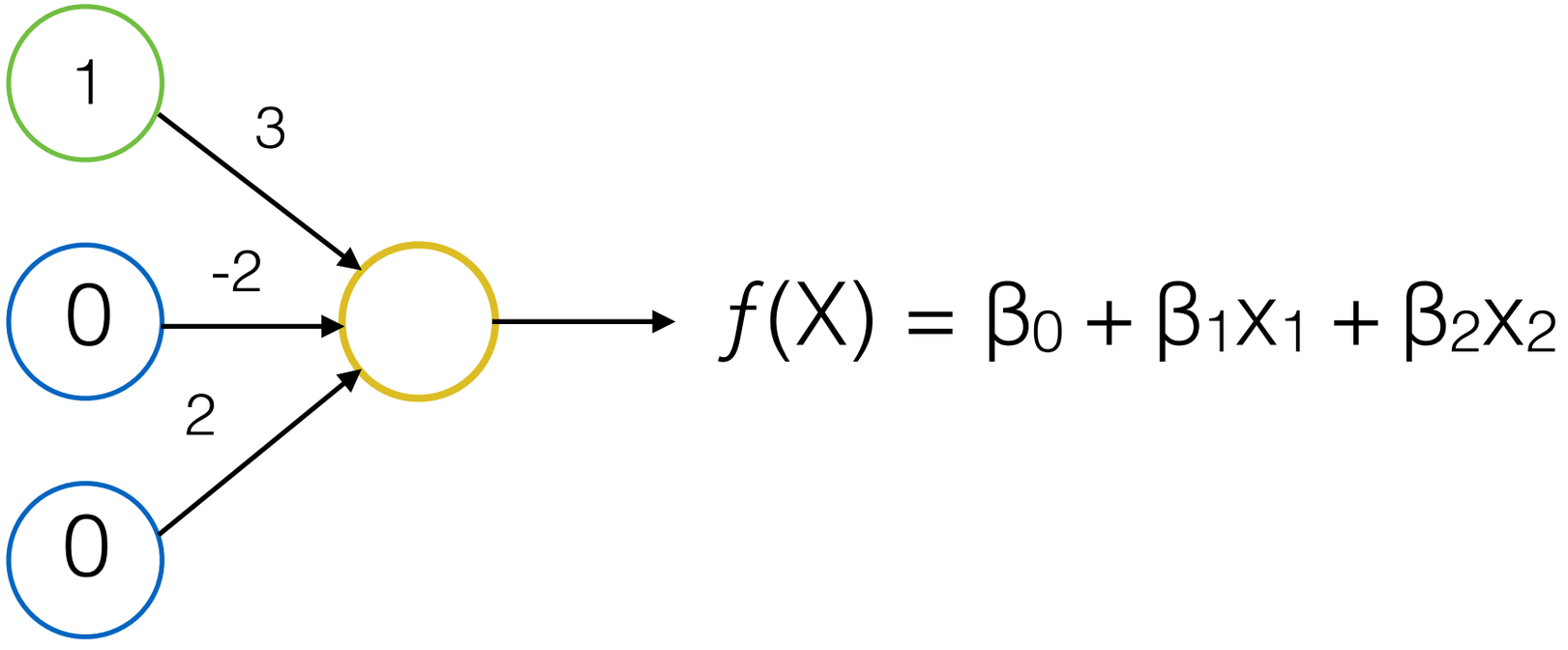
Feature 1	Feature 2	Target
0	0	0
1	0	1
0	1	1
1	1	1



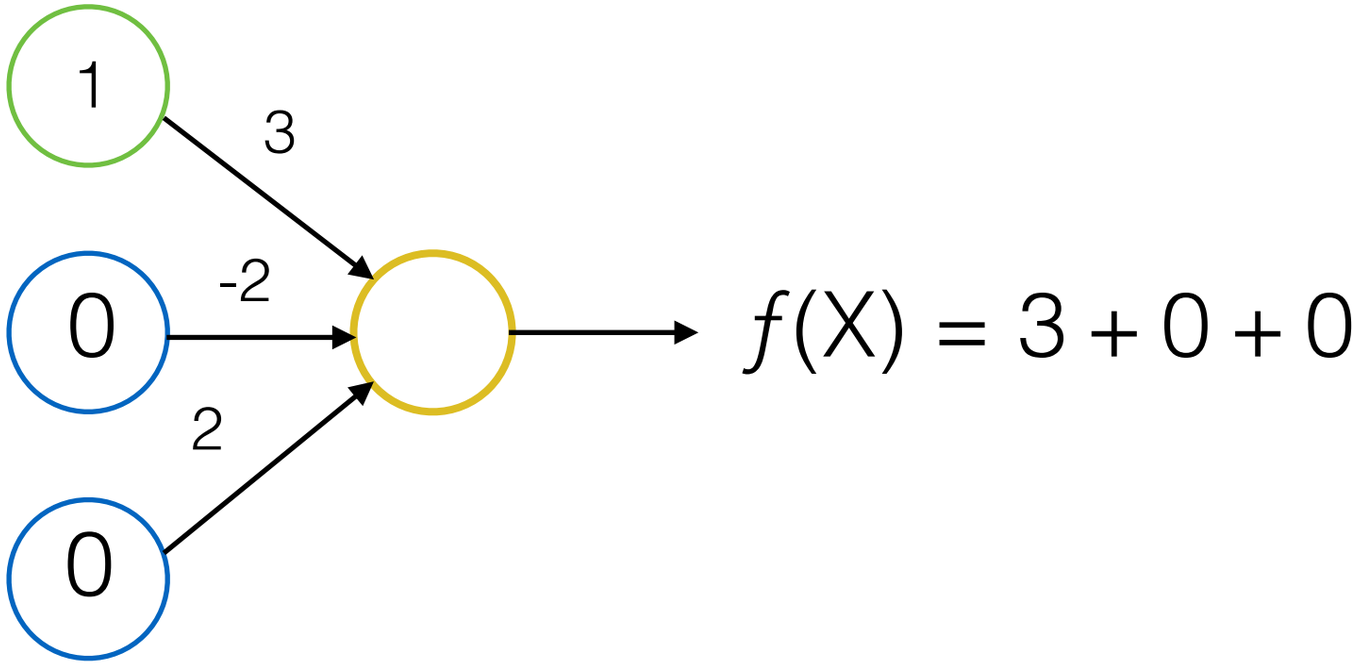
Feature 1	Feature 2	Target
0	0	0
1	0	1
0	1	1
1	1	1



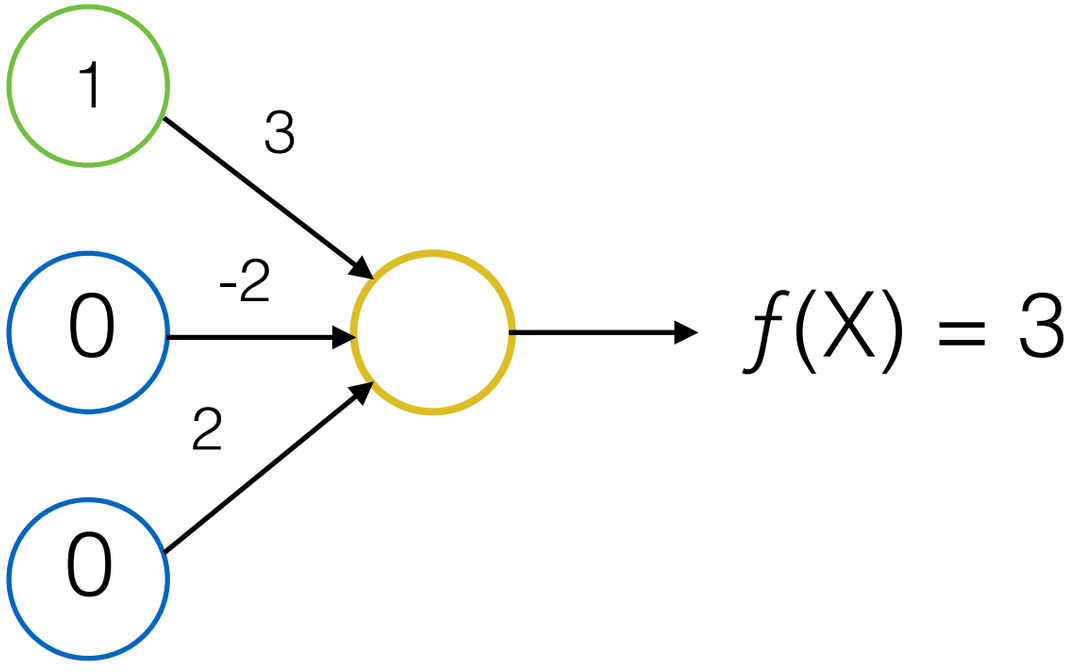
Feature 1	Feature 2	Target
0	0	0
1	0	1
0	1	1
1	1	1



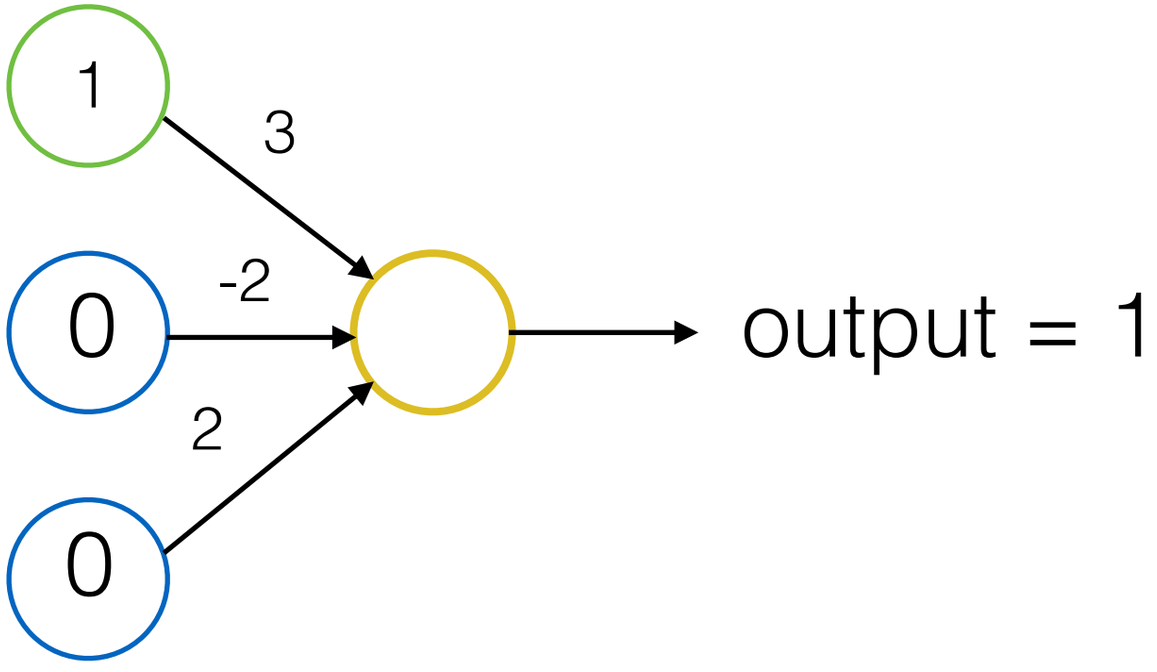
Feature 1	Feature 2	Target
0	0	0
1	0	1
0	1	1
1	1	1



Feature 1	Feature 2	Target
0	0	0
1	0	1
0	1	1
1	1	1



Feature 1	Feature 2	Target
0	0	0
1	0	1
0	1	1
1	1	1



weights: 3, -2, 2

input: 1, 0, 0

output: 1

target: 0

updated $weight_0$ = $weight_0$ - (output - target) * $input_0$

updated $weight_1$ = $weight_1$ - (output - target) * $input_1$

updated $weight_2$ = $weight_2$ - (output - target) * $input_2$

weights: 3, -2, 2

input: 1, 0, 0

output: 1

target: 0

updated $\text{weight}_0 = 3 - (\text{output} - \text{target}) * \text{input}_0$

updated $\text{weight}_1 = -2 - (\text{output} - \text{target}) * \text{input}_1$

updated $\text{weight}_2 = 2 - (\text{output} - \text{target}) * \text{input}_2$

weights: 3, -2, 2

input: 1, 0, 0

output: 1

target: 0

updated $weight_0 = 3 - (1 - target) * input_0$

updated $weight_1 = -2 - (1 - target) * input_1$

updated $weight_2 = 2 - (1 - target) * input_2$

weights: 3, -2, 2

input: 1, 0, 0

output: 1

target: 0

$$\text{updated weight}_0 = 3 - (1 - 0) * \text{input}_0$$

$$\text{updated weight}_1 = -2 - (1 - 0) * \text{input}_1$$

$$\text{updated weight}_2 = 2 - (1 - 0) * \text{input}_2$$

weights: 3, -2, 2 **output**: 1

input: 1, 0, 0 **target**: 0

$$\text{updated weight}_0 = 3 - (1 - 0) * 1$$

$$\text{updated weight}_1 = -2 - (1 - 0) * 0$$

$$\text{updated weight}_2 = 2 - (1 - 0) * 0$$

weights: 3, -2, 2

input: 1, 0, 0

output: 1

target: 0

updated $\text{weight}_0 = 3 - 1$

updated $\text{weight}_1 = -2 - 0$

updated $\text{weight}_2 = 2 - 0$

weights: 3, -2, 2

input: 1, 0, 0

output: 1

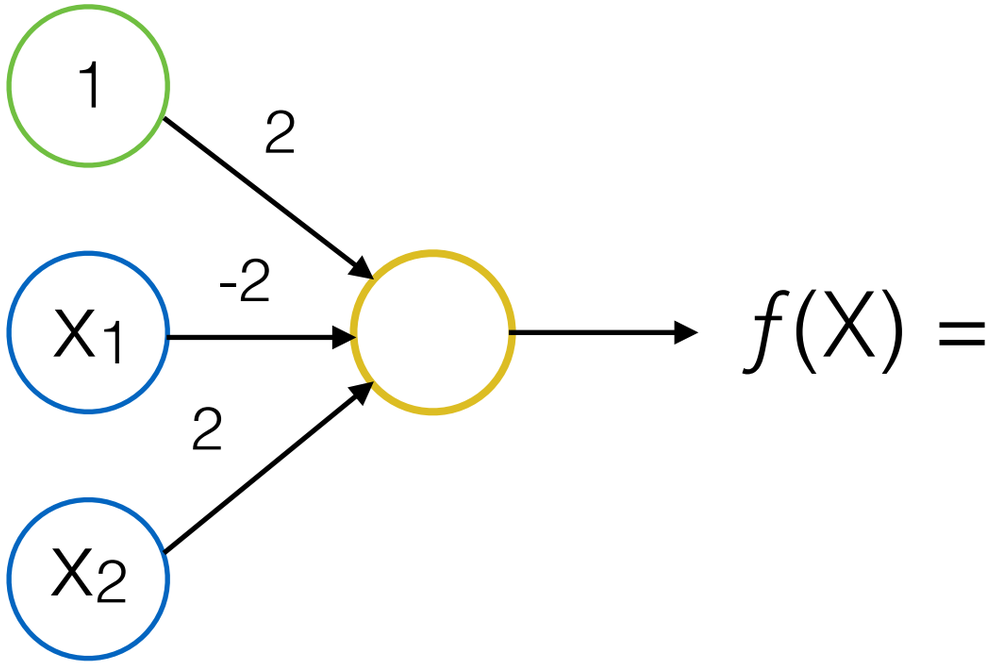
target: 0

updated $\text{weight}_0 = 2$

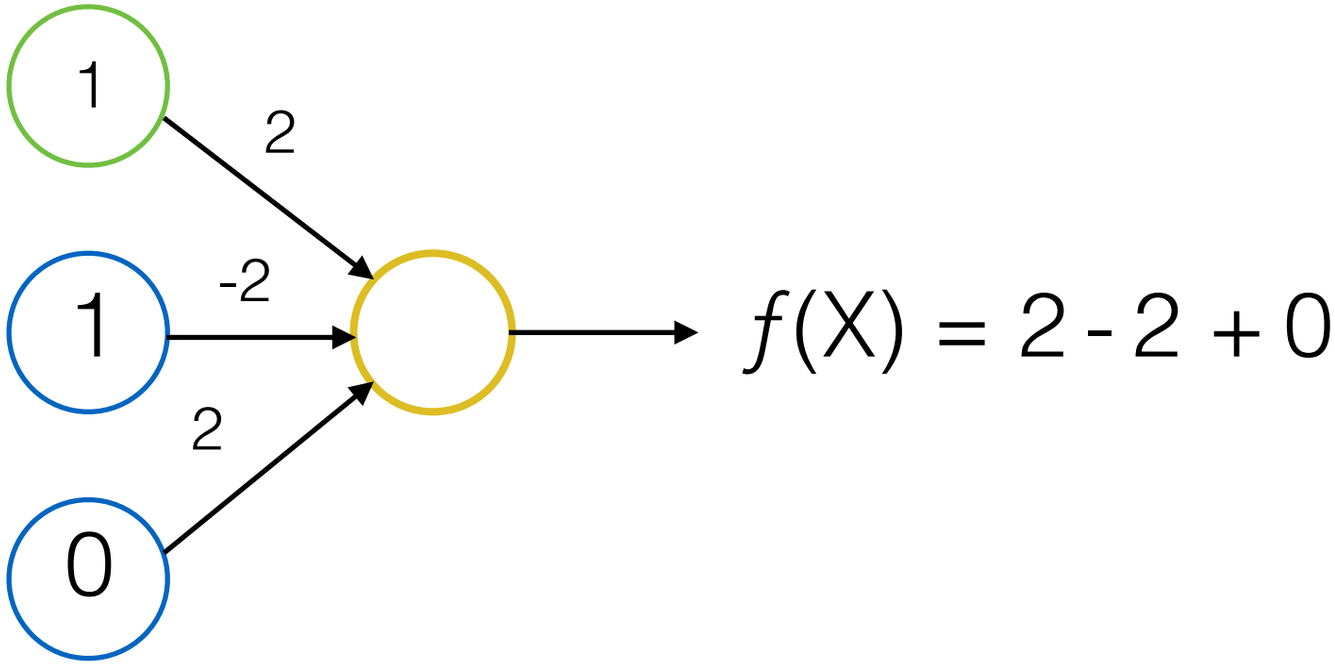
updated $\text{weight}_1 = -2$

updated $\text{weight}_2 = 2$

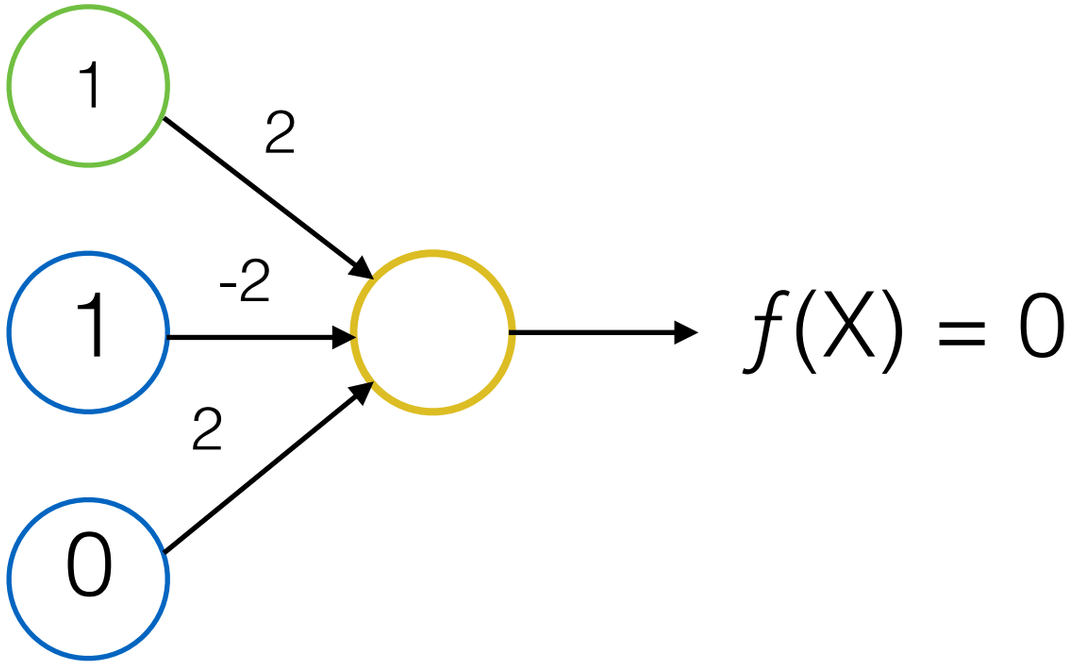
Feature 1	Feature 2	Target
0	0	0
1	0	1
0	1	1
1	1	1



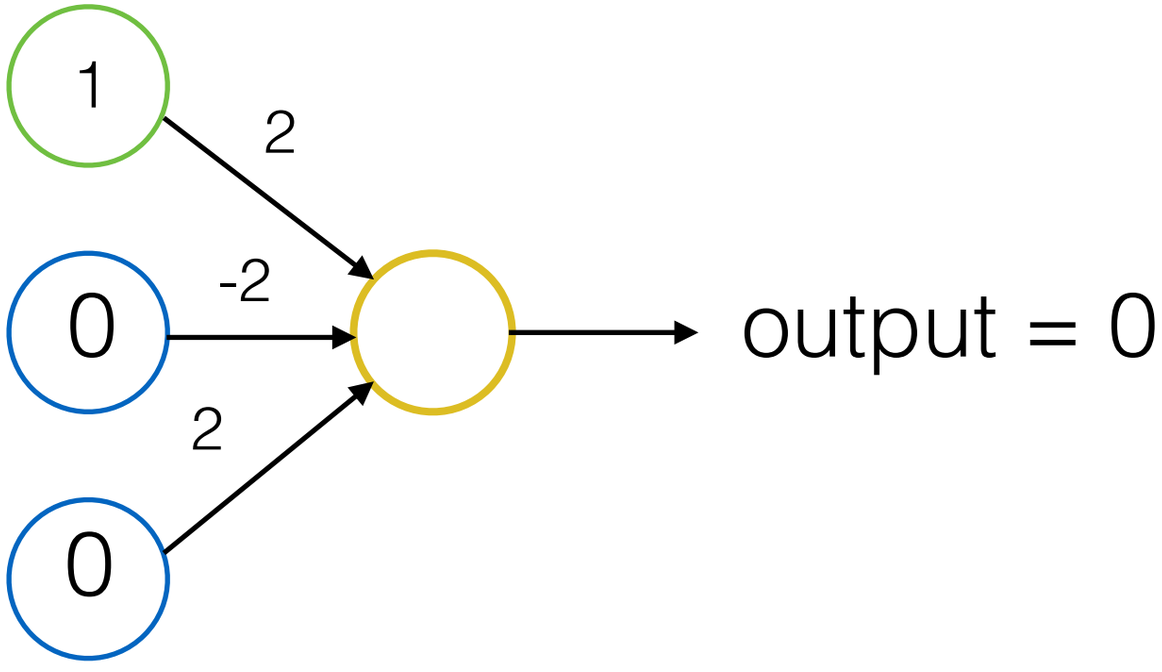
Feature 1	Feature 2	Target
0	0	0
1	0	1
0	1	1
1	1	1



Feature 1	Feature 2	Target
0	0	0
1	0	1
0	1	1
1	1	1



Feature 1	Feature 2	Target
0	0	0
1	0	1
0	1	1
1	1	1



weights: 2, -2, 2

input: 1, 1, 0

output: 0

target: 1

updated $weight_0$ = $weight_0$ - (output - target) * $input_0$

updated $weight_1$ = $weight_1$ - (output - target) * $input_1$

updated $weight_2$ = $weight_2$ - (output - target) * $input_2$

weights: 2, -2, 2

input: 1, 1, 0

output: 0

target: 1

updated $\text{weight}_0 = 2 - (\text{output} - \text{target}) * \text{input}_0$

updated $\text{weight}_1 = -2 - (\text{output} - \text{target}) * \text{input}_1$

updated $\text{weight}_2 = 2 - (\text{output} - \text{target}) * \text{input}_2$

weights: 2, -2, 2

input: 1, 1, 0

output: 0

target: 1

updated $weight_0 = 2 - (0 - target) * input_0$

updated $weight_1 = -2 - (0 - target) * input_1$

updated $weight_2 = 2 - (0 - target) * input_2$

weights: 2, -2, 2

input: 1, 1, 0

output: 0

target: 1

$$\text{updated weight}_0 = 2 - (0 - 1) * \text{input}_0$$

$$\text{updated weight}_1 = -2 - (0 - 1) * \text{input}_1$$

$$\text{updated weight}_2 = 2 - (0 - 1) * \text{input}_2$$

weights: 2, -2, 2 **output**: 0

input: 1, 1, 0 **target**: 1

$$\text{updated weight}_0 = 2 - (0 - 1) * 1$$

$$\text{updated weight}_1 = -2 - (0 - 1) * 1$$

$$\text{updated weight}_2 = 2 - (0 - 1) * 0$$

weights: 2, -2, 2

output: 0

input: 1, 1, 0

target: 1

$$\text{updated weight}_0 = 2 - (-1)$$

$$\text{updated weight}_1 = -2 - (-1)$$

$$\text{updated weight}_2 = 2 - 0$$

weights: 2, -2, 2

input: 1, 1, 0

output: 0

target: 1

$$\text{updated weight}_0 = 2 + 1$$

$$\text{updated weight}_1 = -2 + 1$$

$$\text{updated weight}_2 = 2 - 0$$

weights: 2, -2, 2

input: 1, 1, 0

output: 0

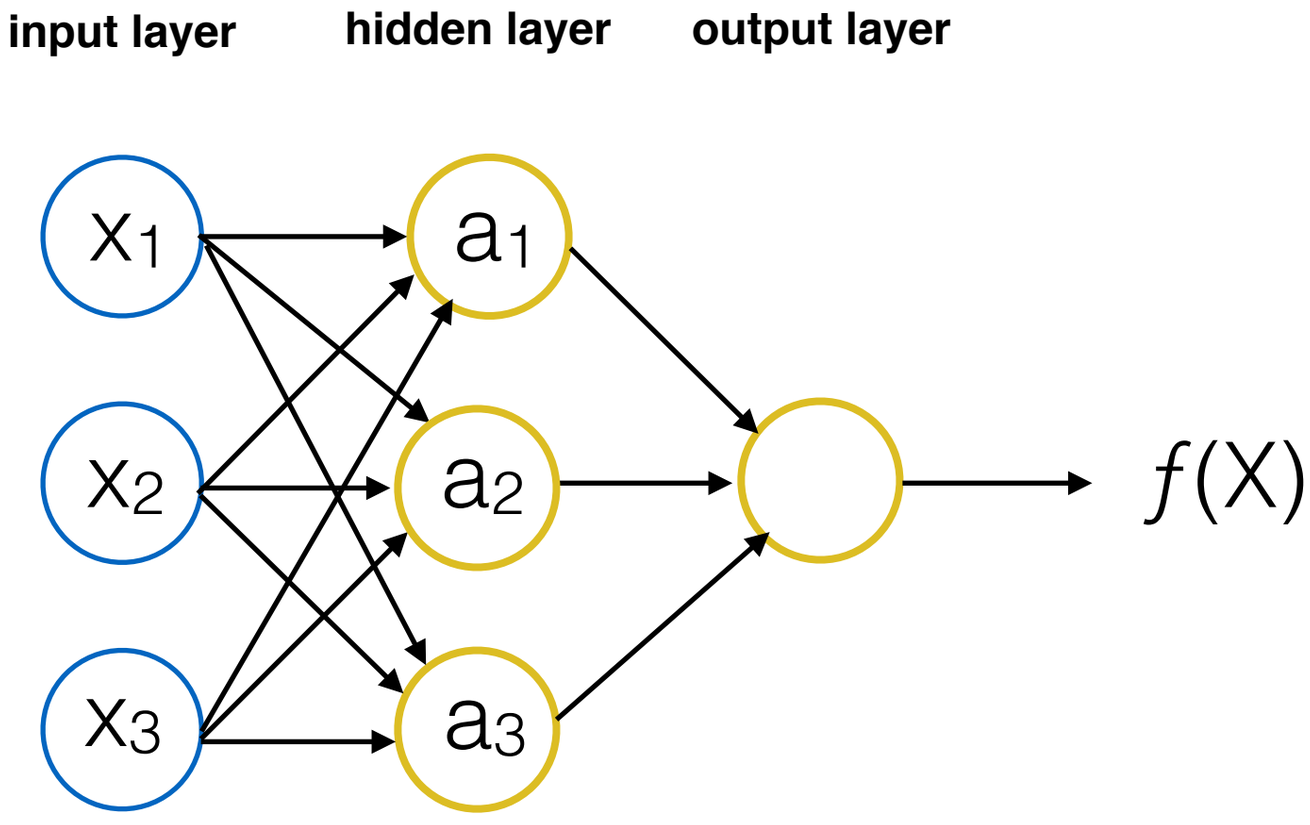
target: 1

updated $\text{weight}_0 = 3$

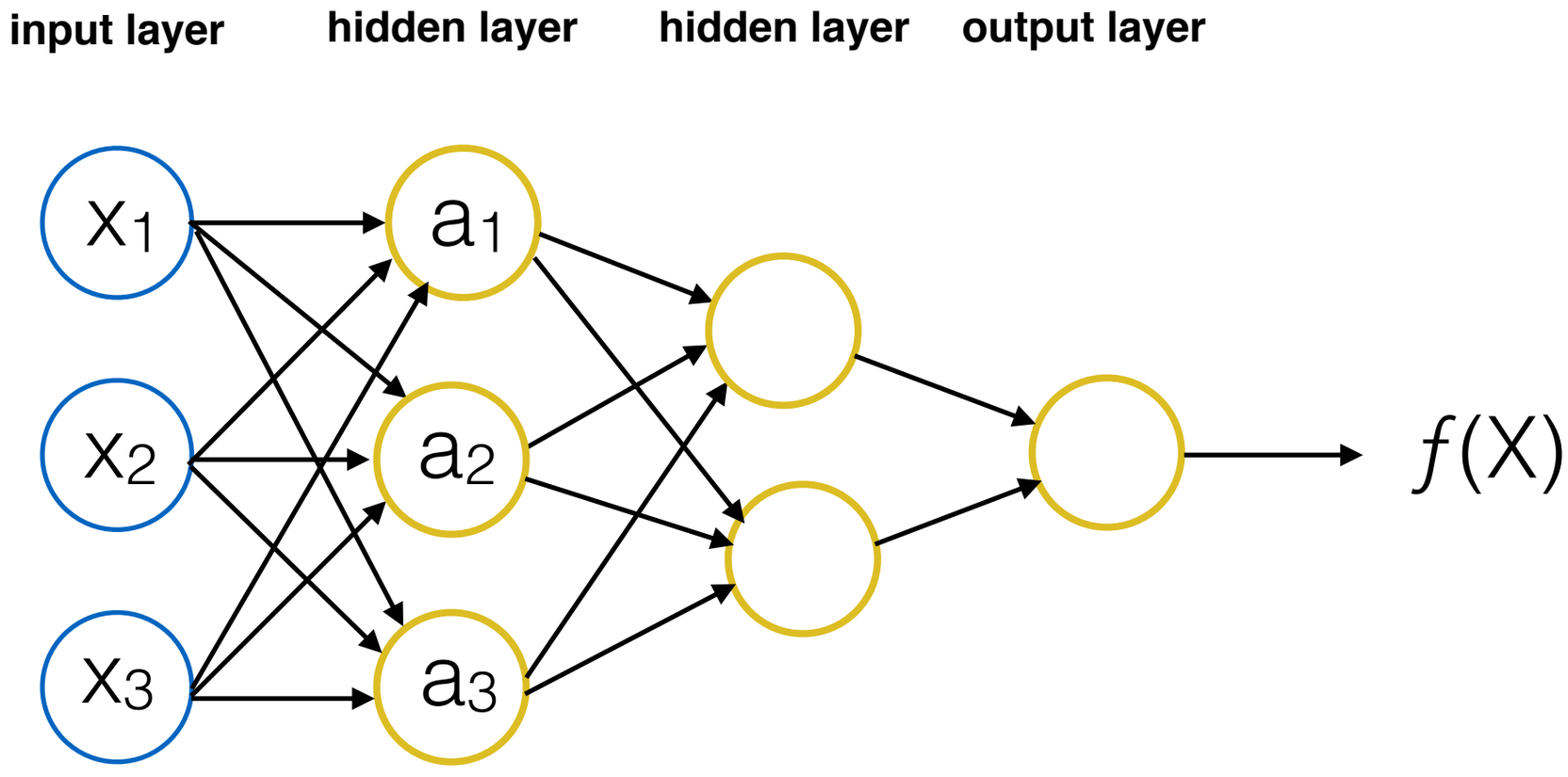
updated $\text{weight}_1 = -1$

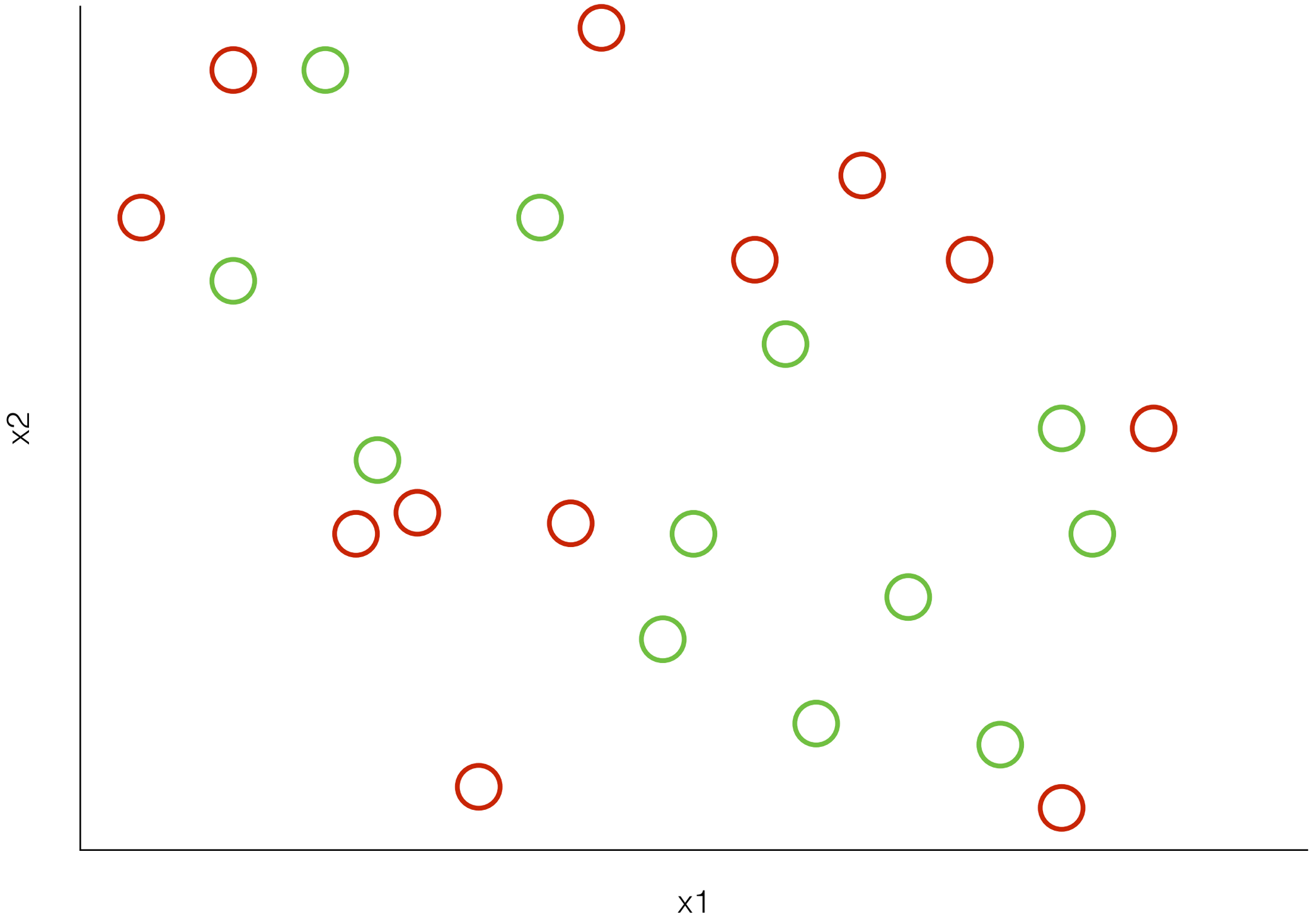
updated $\text{weight}_2 = 2$

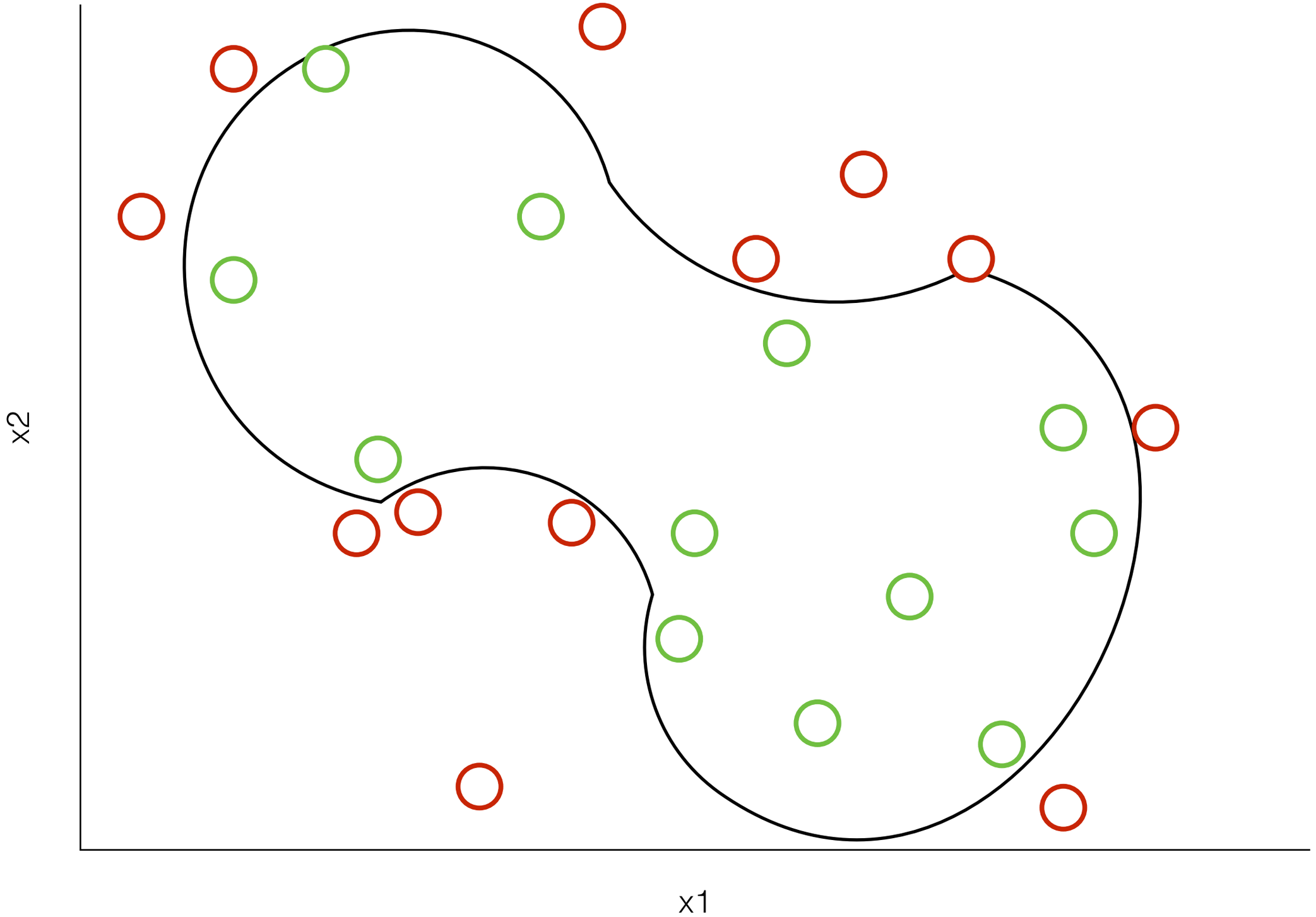
Multi-Layer Perceptron (MLP)



Multi-Layer Perceptron (MLP)

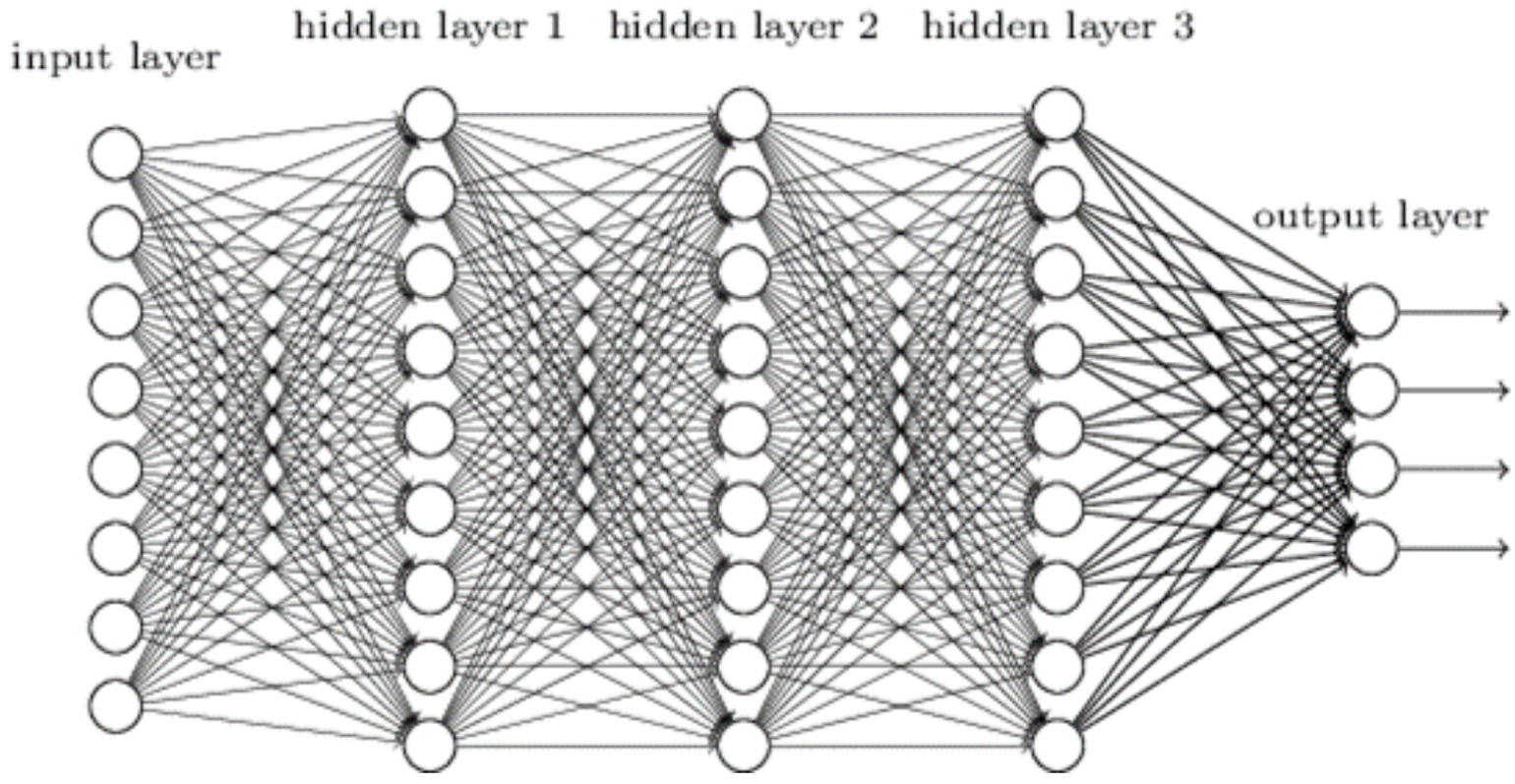



x2
x1


x2
x1

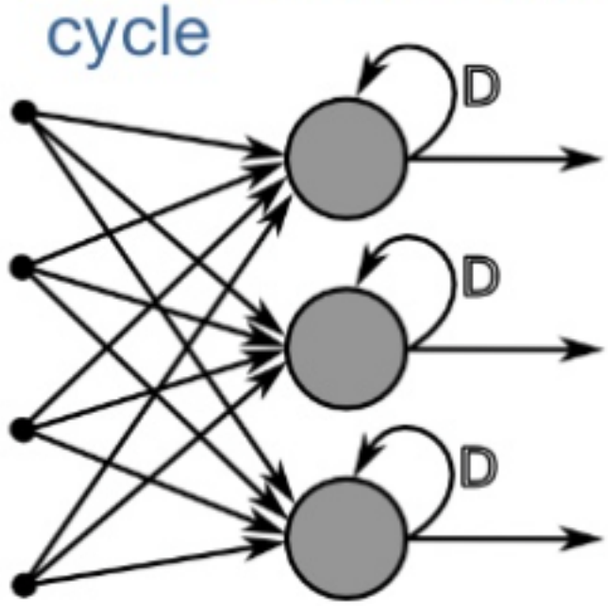
[[216, 203, 125, 10, 84, 241, 149, 159, 212, 118, 135, 158, 11, 91, 36, 177, 176, 253, 132, 210, 159, 20, 153, 131, 132, 55,16, 132],
[184, 34, 95, 225, 60, 218, 49, 193, 93, 119, 68, 133, 195,104, 248, 18, 18, 136, 90, 71, 81, 41, 233, 53, 46, 87,86, 243],
[85, 61, 220, 170, 206, 34, 141, 97, 66, 217, 124, 143, 241,205, 76, 123, 66, 72, 231, 116, 244, 74, 155, 144, 47, 230,171, 165],
[156, 87, 181, 90, 160, 2, 184, 112, 108, 62, 223, 153, 93, 244, 83, 187, 83, 18, 134, 28, 121, 244, 202, 176, 228, 233,76, 13],
[76, 238, 128, 183, 119, 130, 34, 12, 112, 254, 90, 167, 64,89, 170, 221, 196, 69, 82, 11, 65, 86, 254, 111, 134, 0,148, 246],
[105, 178, 254, 31, 32, 133, 57, 40, 6, 85, 115, 56, 132,84, 35, 119, 158, 182, 106, 77, 84, 106, 164, 230, 54, 42,55, 130],
[25, 86, 222, 59, 242, 111, 59, 183, 236, 214, 251, 7, 142,90, 179, 80, 163, 159, 26, 143, 108, 109, 229, 223, 220, 196,21, 18],
[21, 42, 109, 188, 91, 93, 246, 236, 125, 48, 151, 12, 178,26, 118, 135, 77, 84, 179, 208, 114, 224, 99, 246, 68, 21, 69, 39],
[253, 66, 78, 55, 39, 107, 248, 90, 124, 107, 51, 92, 150,234, 91, 177, 146, 80, 8, 179, 148, 229, 233, 59, 164, 199,252, 43],
[79, 60, 5, 70, 37, 218, 19, 9, 90, 74, 198, 129, 61,160, 206, 11, 37, 171, 44, 241, 228, 190, 232, 99, 7, 100, 83, 225],
[211, 38, 52, 167, 206, 139, 215, 209, 202, 102, 122, 77, 86,117, 134, 22, 176, 94, 22, 201, 6, 73, 156, 226, 36, 0,50, 119],
[159, 24, 197, 215, 16, 243, 177, 13, 108, 211, 6, 97, 75,214, 121, 92, 154, 109, 213, 163, 123, 20, 190, 174, 89, 6,136, 164],
[183, 136, 245, 175, 233, 62, 141, 117, 150, 74, 182, 175, 36,230, 93, 109, 212, 43, 10, 75, 234, 124, 70, 244, 161, 76,241, 223],
[150, 7, 184, 20, 133, 22, 112, 212, 48, 30, 156, 113, 127,207, 219, 173, 223, 127, 202, 172, 39, 98, 134, 124, 130, 34,210, 101],
[101, 77, 87, 37, 152, 112, 34, 106, 30, 23, 79, 214, 245,152, 129, 243, 109, 213, 170, 190, 220, 25, 76, 205, 135, 227,225, 165],
[108, 184, 172, 121, 8, 83, 106, 116, 235, 55, 73, 204, 50,40, 124, 153, 225, 157, 13, 28, 105, 62, 242, 214, 56, 159,137, 67],
[14, 75, 26, 47, 74, 205, 45, 219, 27, 18, 79, 28, 49,224, 85, 214, 180, 105, 183, 87, 18, 64, 7, 61, 125, 87,38, 98],
[122, 146, 4, 72, 150, 249, 77, 90, 6, 132, 134, 151, 164,29, 94, 188, 251, 177, 0, 206, 193, 182, 231, 43, 32, 32,80, 147],
[26, 39, 76, 12, 35, 81, 103, 233, 204, 138, 82, 28, 5,68, 229, 197, 52, 215, 224, 117, 101, 4, 154, 4, 205, 50,251, 114],
[68, 176, 23, 246, 11, 57, 62, 25, 38, 17, 136, 106, 113,140, 254, 43, 231, 150, 12, 114, 77, 8, 214, 187, 92, 66,195, 70],
[20, 241, 148, 151, 37, 4, 14, 231, 225, 53, 232, 240, 223,59, 234, 134, 247, 242, 212, 63, 201, 38, 63, 200, 128, 139,167, 173],
[60, 244, 33, 111, 143, 127, 168, 237, 189, 63, 125, 181, 92,91, 14, 211, 21, 26, 253, 109, 174, 100, 138, 138, 221, 204,29, 230],
[81, 174, 217, 93, 65, 134, 7, 36, 176, 122, 226, 23, 223,28, 202, 5, 54, 205, 169, 14, 88, 178, 84, 198, 96, 201,230, 193],
[215, 168, 125, 92, 70, 151, 183, 210, 36, 32, 19, 51, 42,64, 19, 146, 183, 246, 0, 184, 236, 7, 226, 118, 113, 241, 85, 89],
[31, 158, 210, 16, 199, 58, 224, 7, 203, 86, 103, 45, 28,54, 92, 204, 243, 117, 75, 208, 248, 223, 87, 250, 14, 43,102, 66],
[13, 236, 138, 67, 236, 109, 113, 46, 115, 19, 214, 154, 199,248, 55, 172, 214, 249, 125, 154, 139, 141, 188, 78, 107, 200, 196, 16],
[65, 150, 158, 254, 114, 177, 120, 15, 65, 58, 79, 171, 118,32, 250, 81, 27, 85, 128, 146, 144, 234, 139, 26, 6, 68,133, 205],
[123, 68, 216, 34, 139, 34, 34, 175, 213, 72, 76, 19, 32,138, 132, 111, 242, 249, 177, 89, 61, 72, 252, 79, 20, 171,174, 177]]

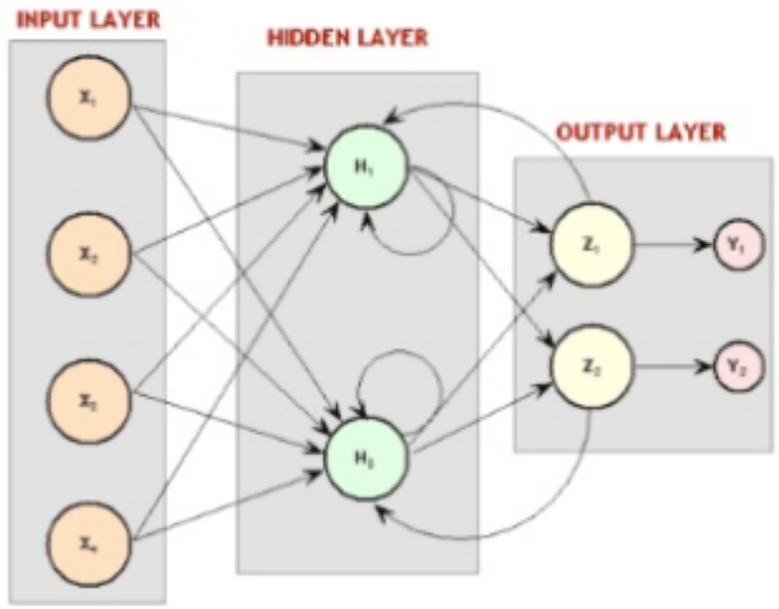
Deep Neural Network



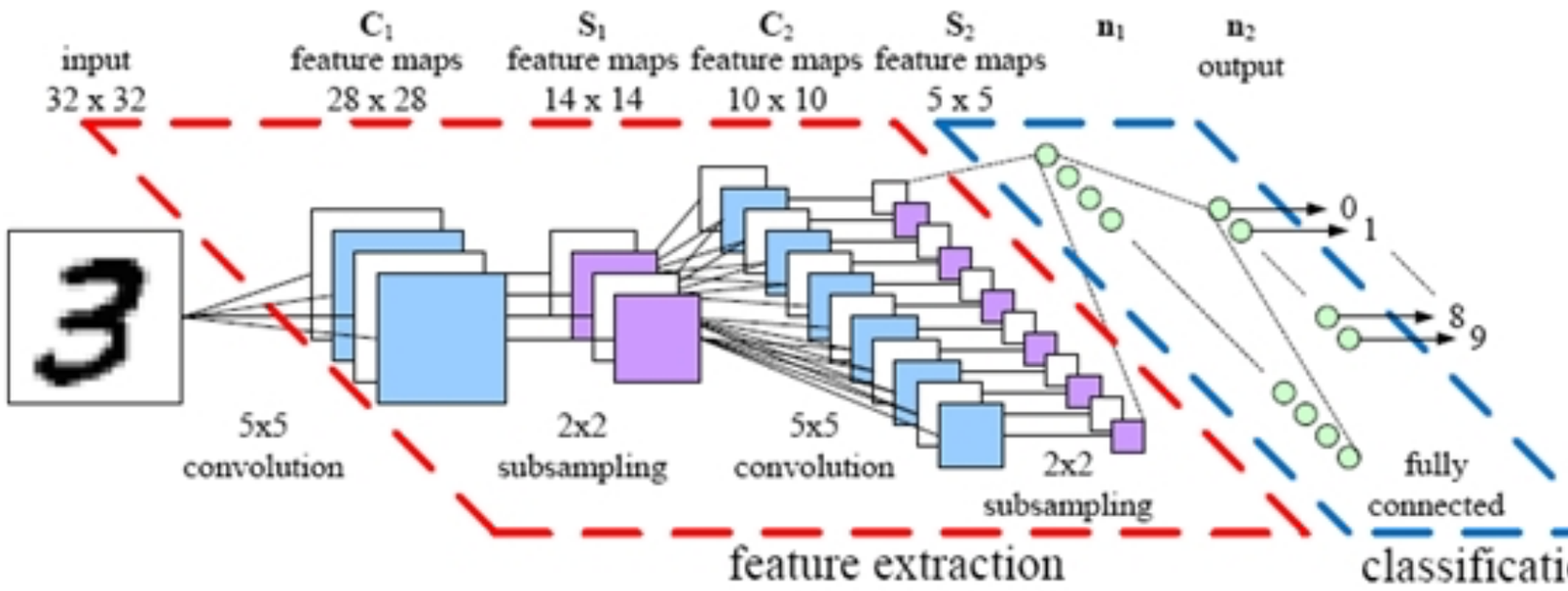
Recurrent neural network

- A class of artificial neural network where connections between units form a directed cycle

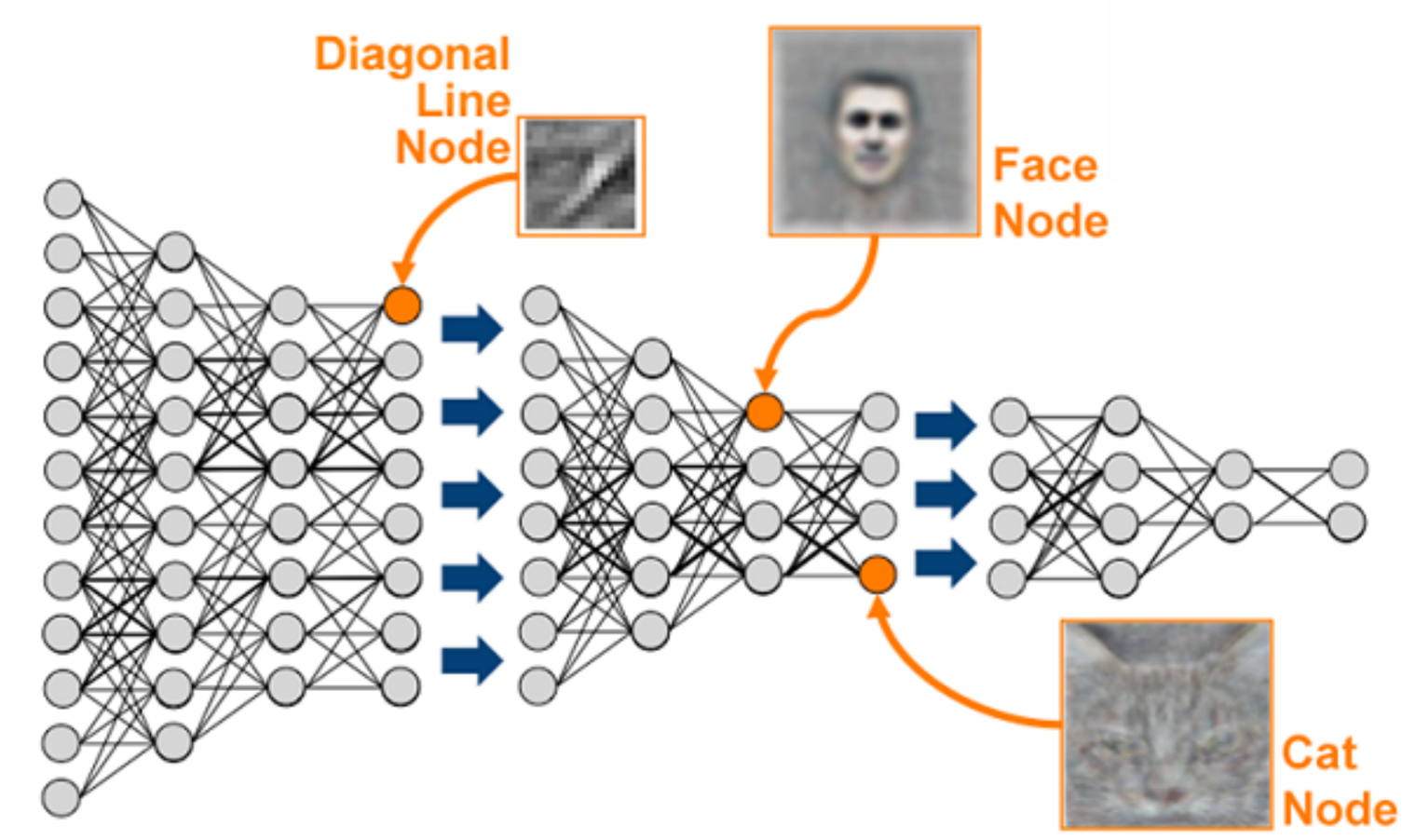




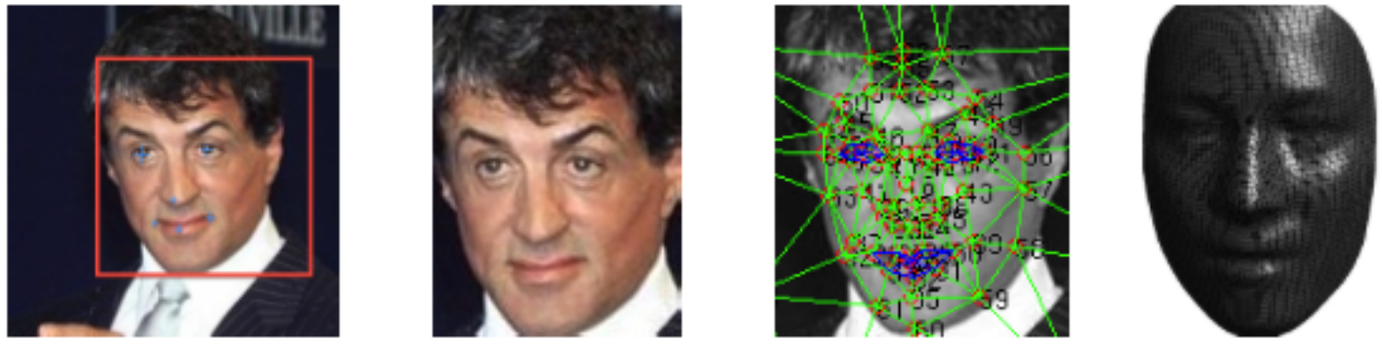
Convolutional Neural Network



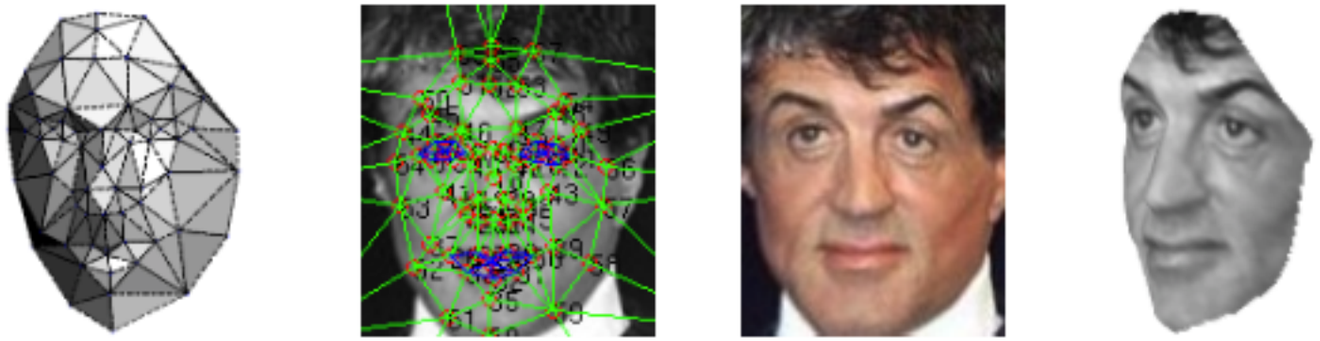
Convolutional Neural Network



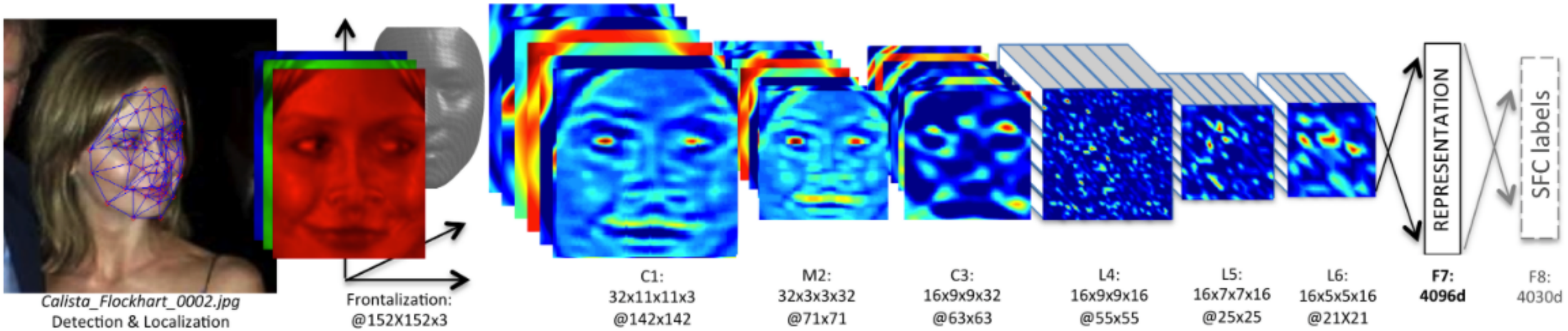
Facebook - DeepFace



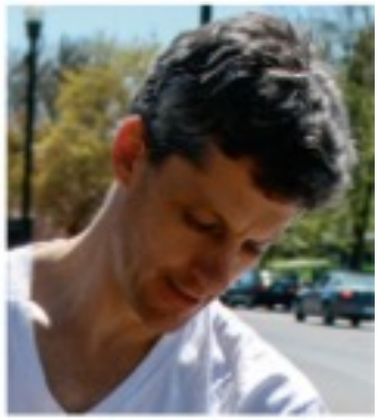
(a) (b) (c) (d)



(e) (f) (g) (h)


Calista_Flockhart_0002.jpg
Detection & Localization
Frontalization:
@152X152x3
C1:
32x11x11x3
@142x142
M2:
32x3x3x32
@71x71
C3:
16x9x9x32
@63x63
L4:
16x9x9x16
@55x55
L5:
16x7x7x16
@25x25
L6:
16x5x5x16
@21X21
F7:
4096d
REPRESENTATION
F8:
4030d
SFC labels









Useful Machine Learning and Data Science books for beginners to intermediate:

Book	Author
An Introduction to Statistical Learning	Robert Tibshirani and Trevor Hastie
Python Machine Learning	Sebastian Raschka
Introduction to Machine Learning with Python	Andreas C. Müller; Sarah Guido
Data Smart: Using Data Science to Transform Information Into Insight	John W. Foreman
Naked Statistics: Stripping the Dread from the Data	Charles Wheelan