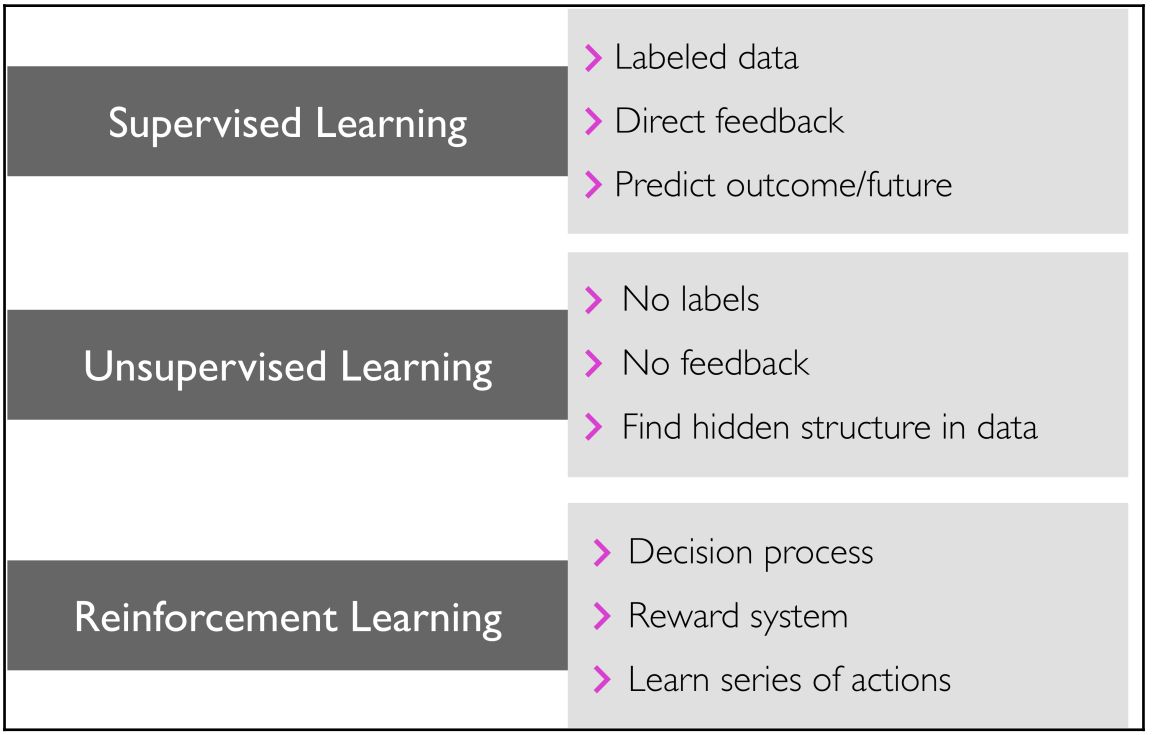
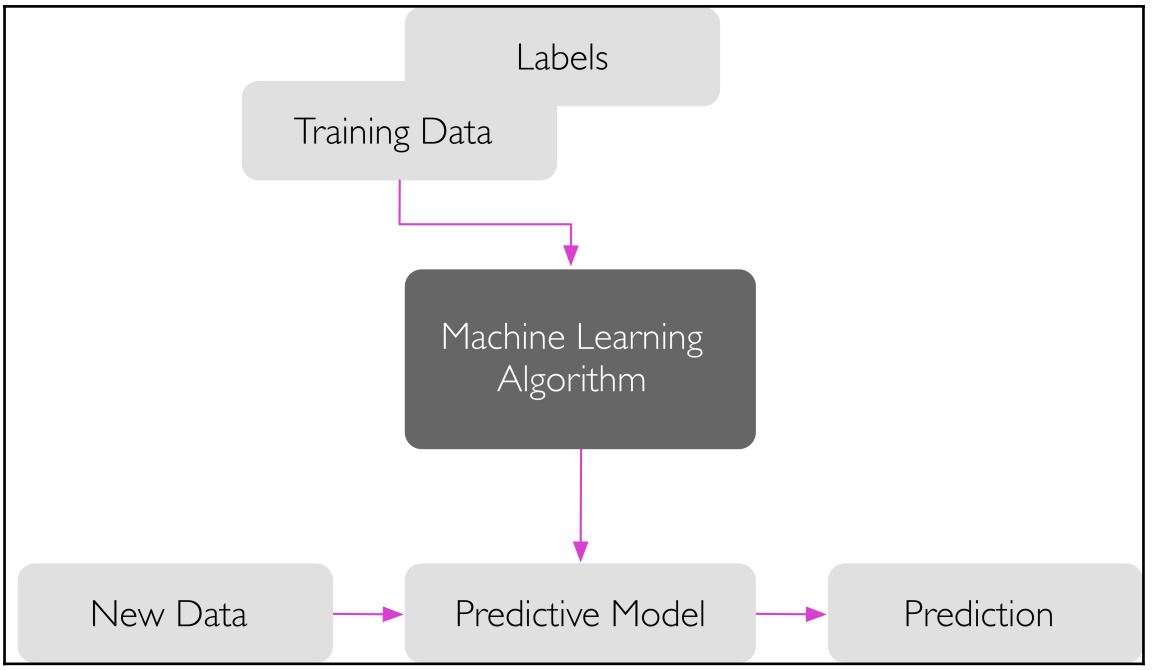
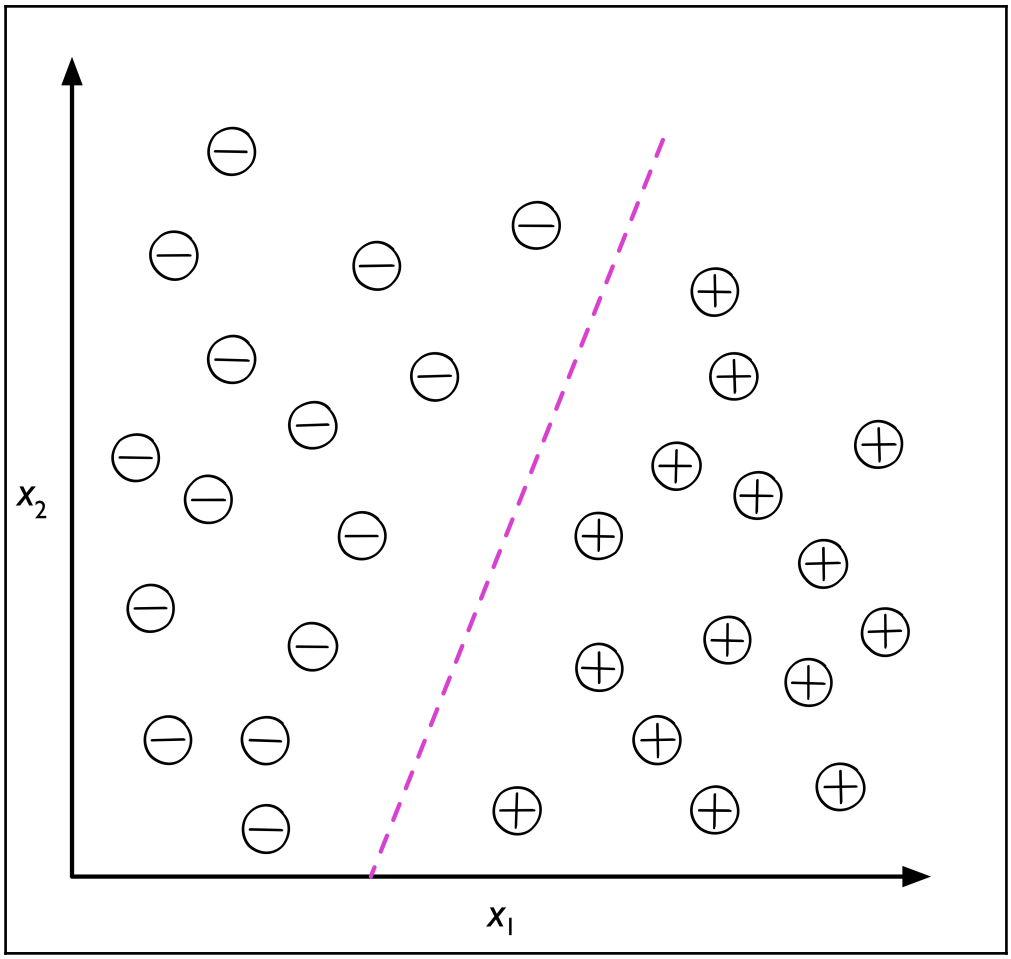
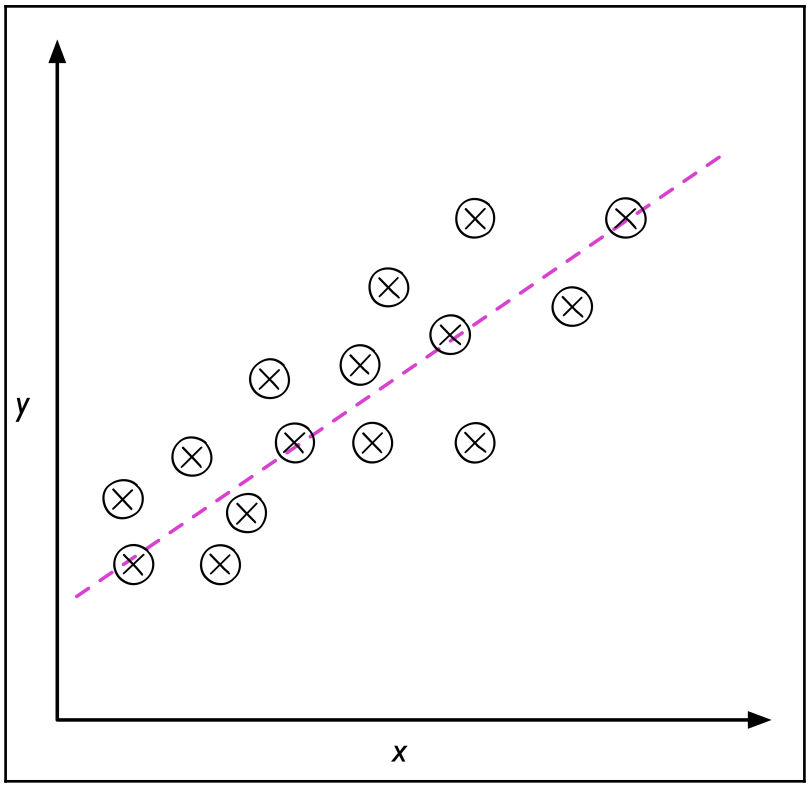
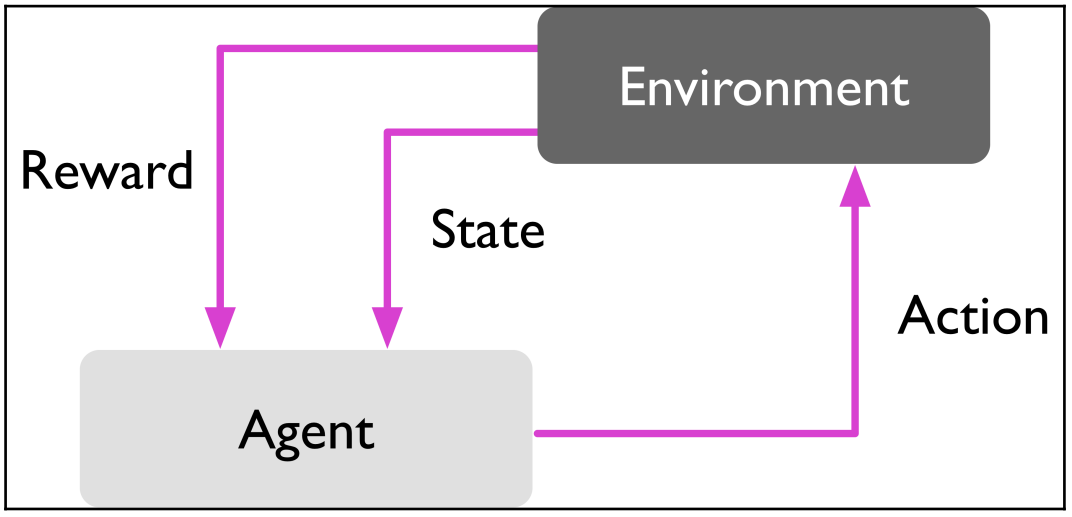
Chapter 1: Giving Computers the Ability to Learn from Data

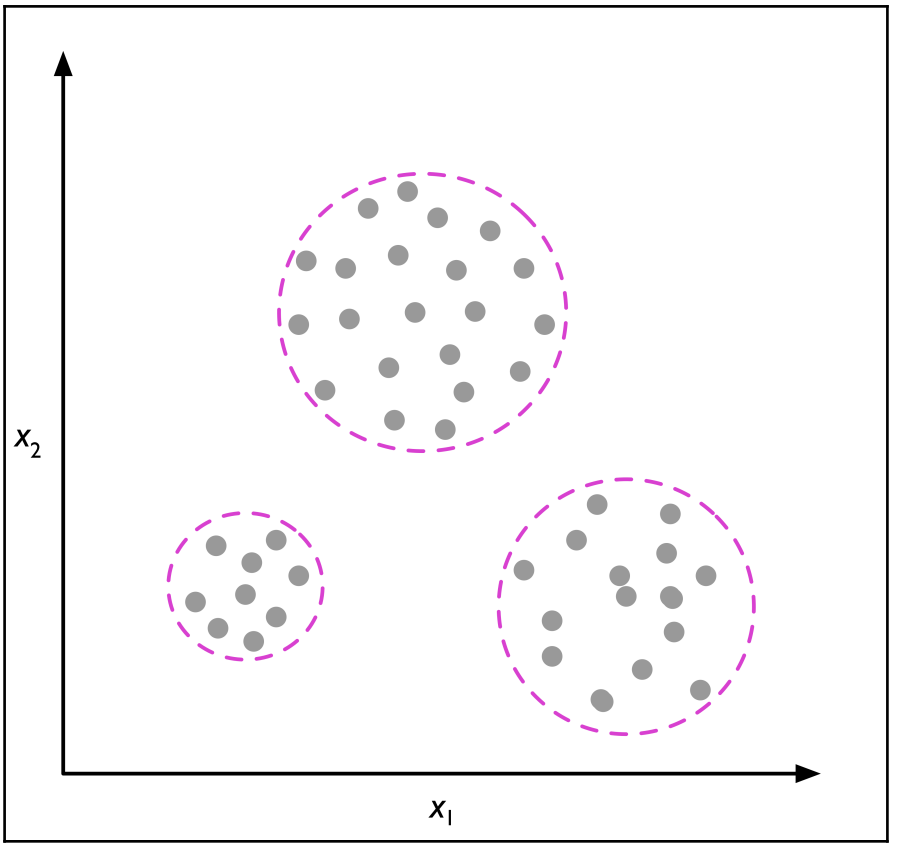


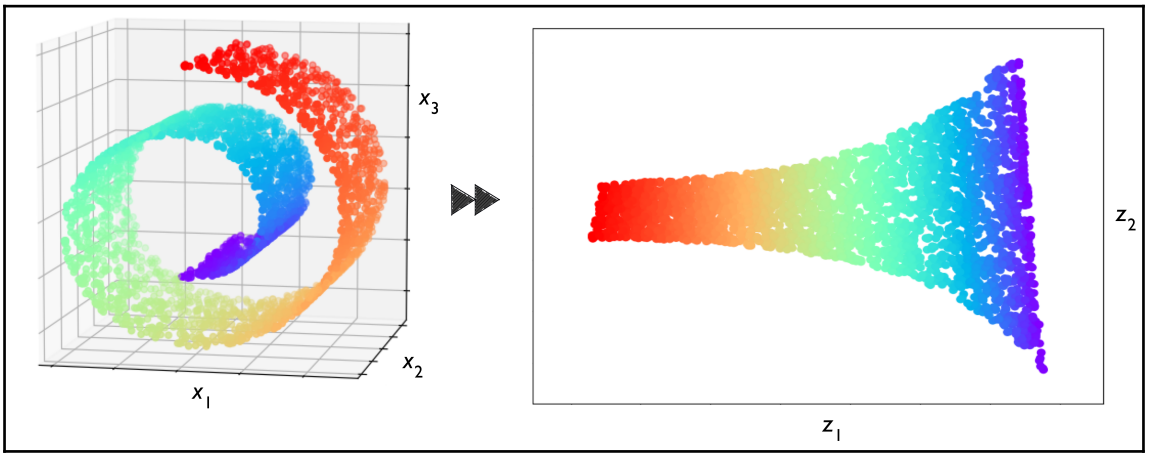

Labels
Training Data
Machine Learning Algorithm
New Data
Predictive Model
Prediction

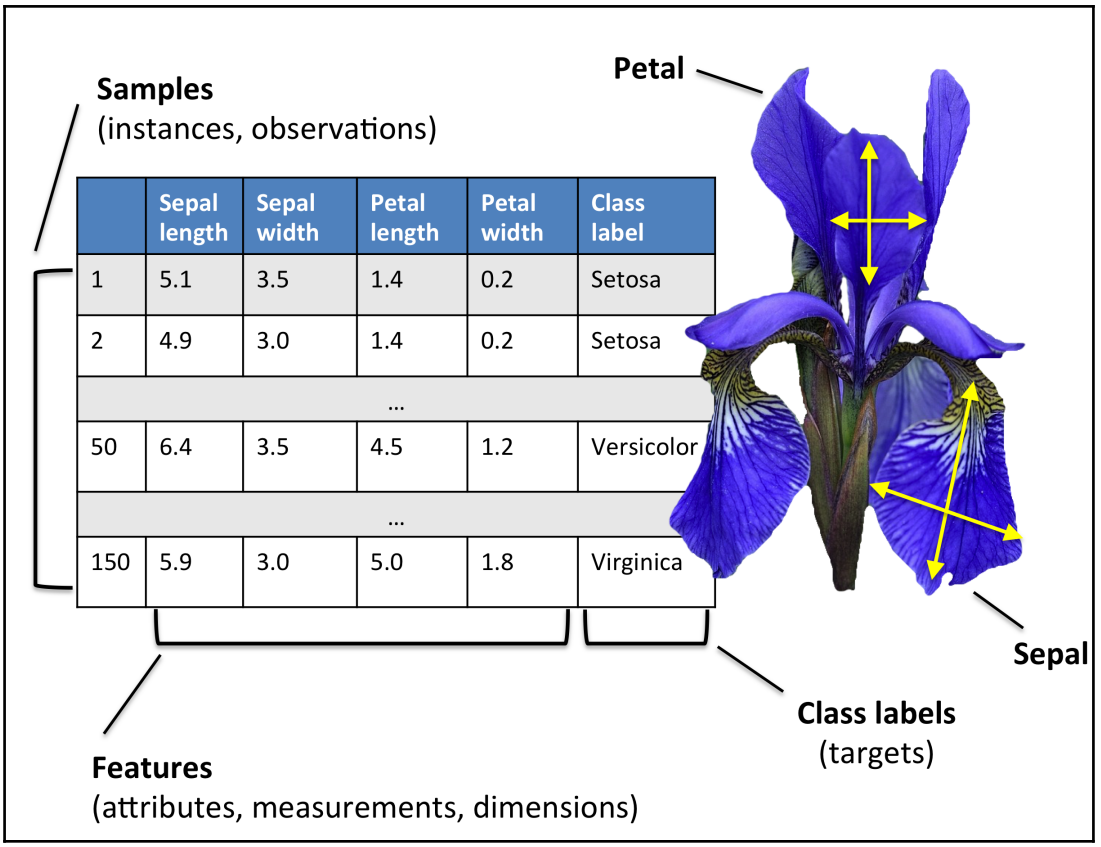

x_2
x_1


y
x

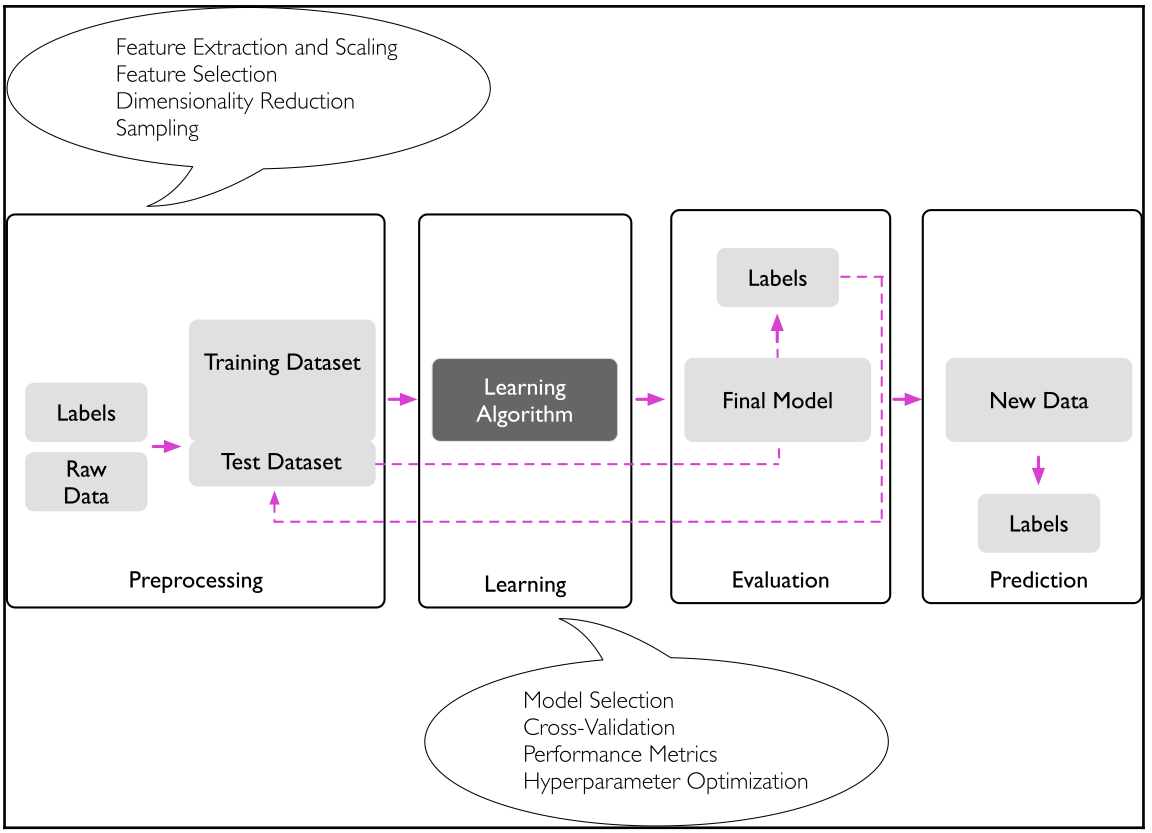

Environment
Reward
State
Action
Agent


x_2
x_1

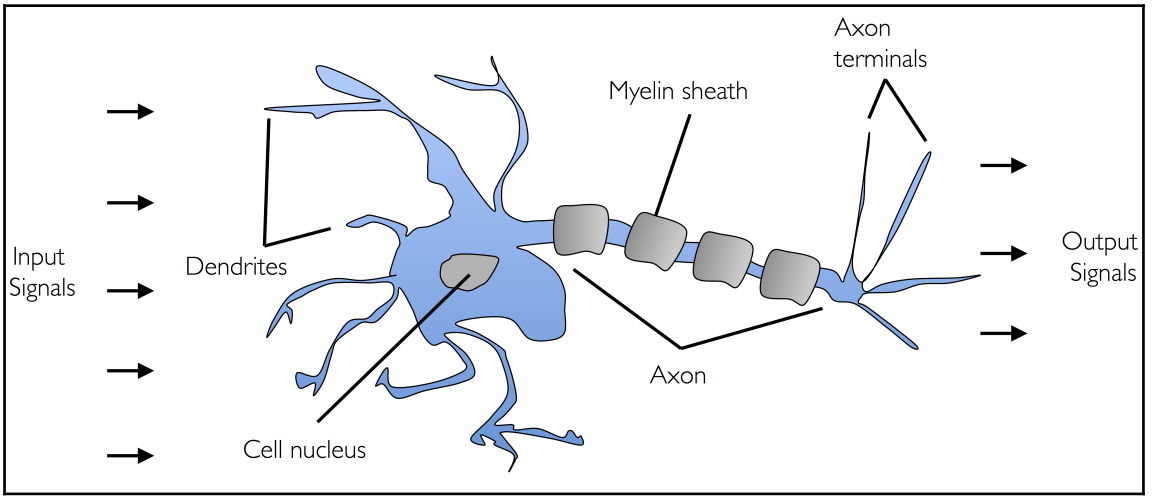

x_3
x_2
x_1
z_2
z_1

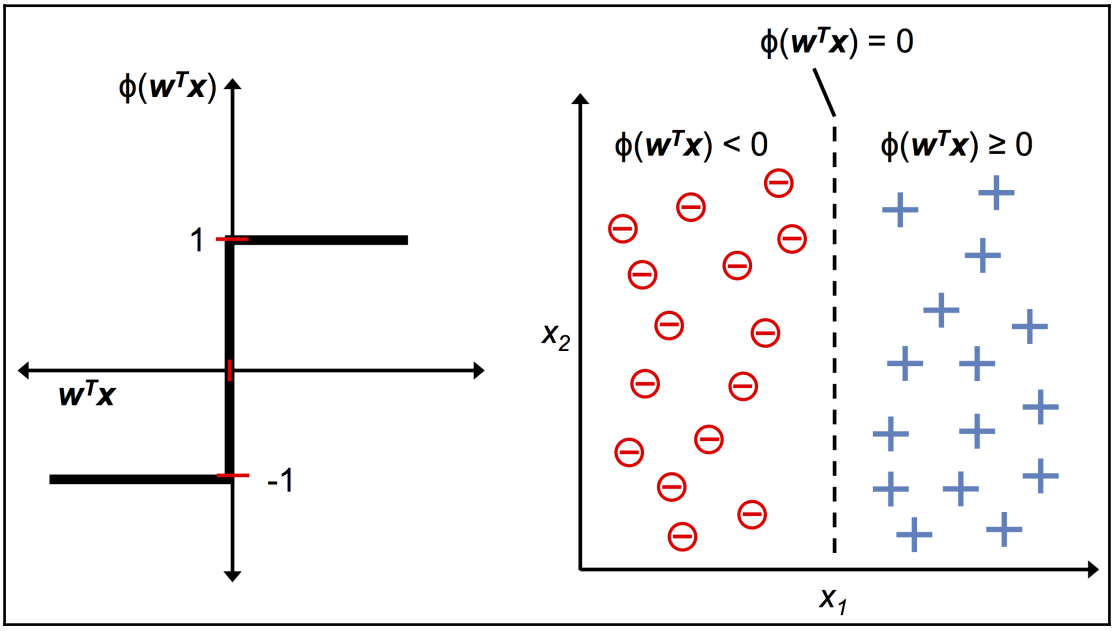


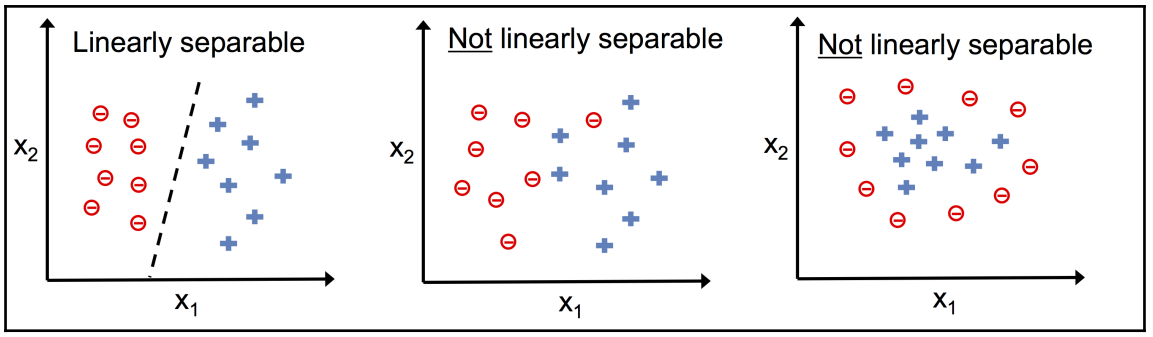
	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica

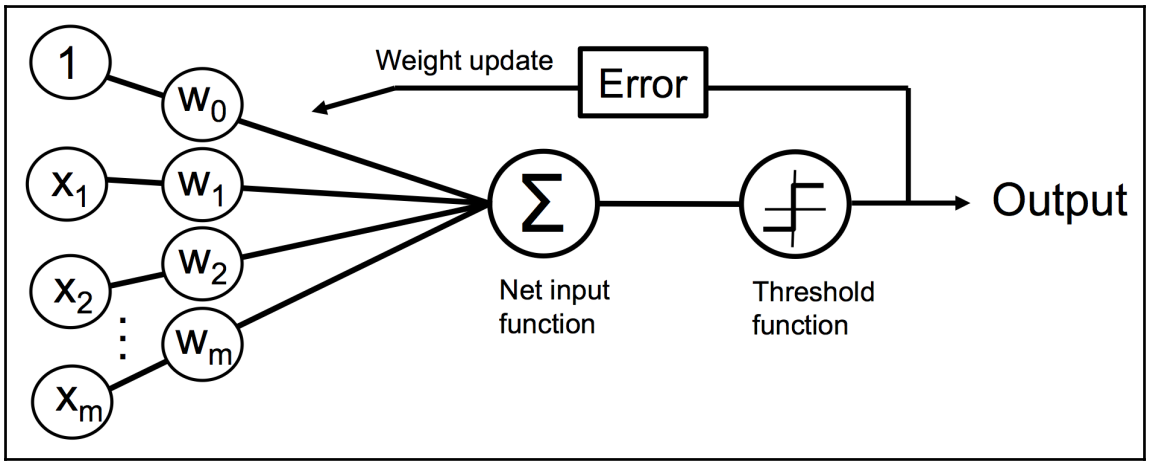

Feature Extraction and Scaling
Feature Selection
Dimensionality Reduction
Sampling
Labels
Raw Data
Training Dataset
Test Dataset
Preprocessing
Learning Algorithm
Learning
Labels
Final Model
Evaluation
New Data
Labels
Prediction
Model Selection
Cross-Validation
Performance Metrics
Hyperparameter Optimization

Chapter 2: Training Simple Machine Learning Algorithms for Classification

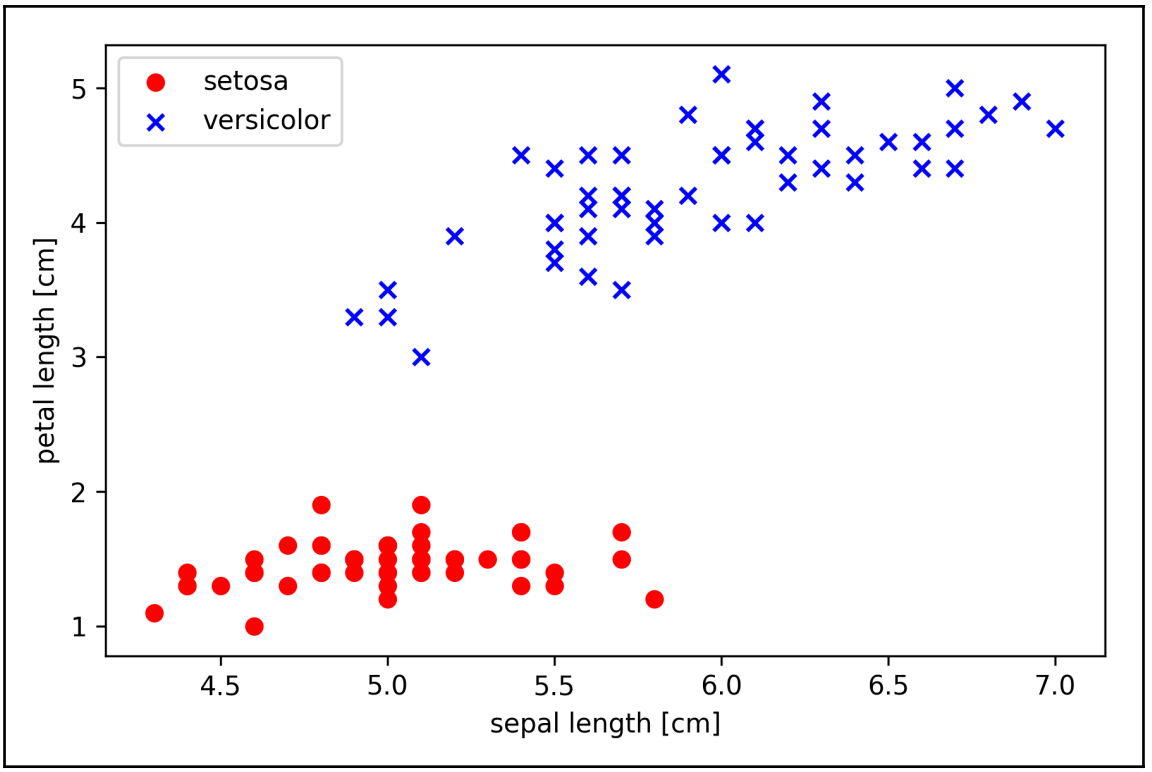


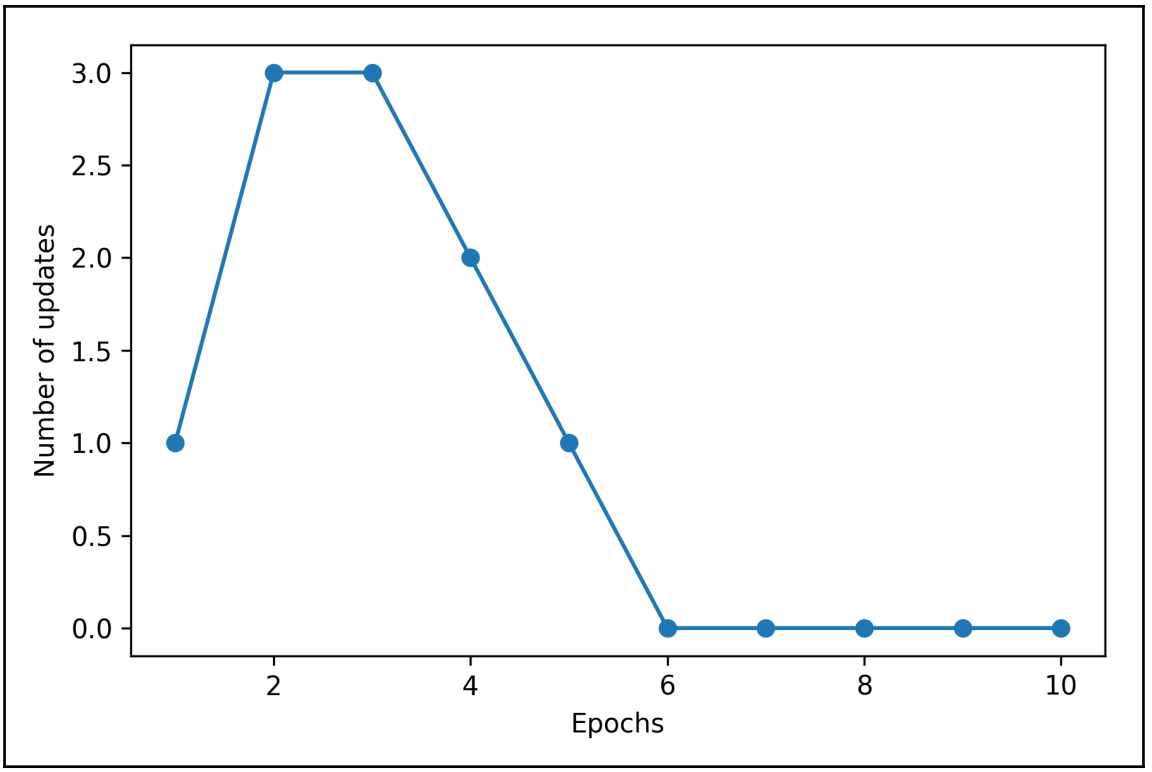


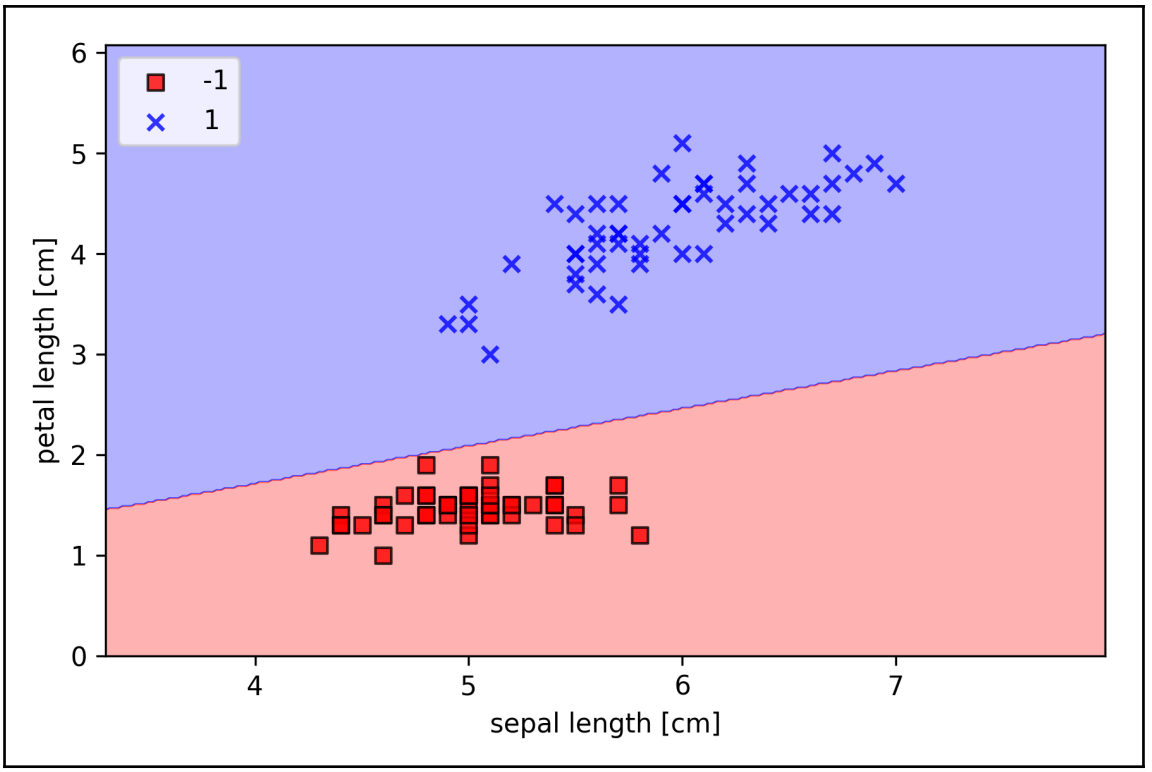

Linearly separable
Not linearly separable
Not linearly separable
x2
x1
x2
x1
x2
x1

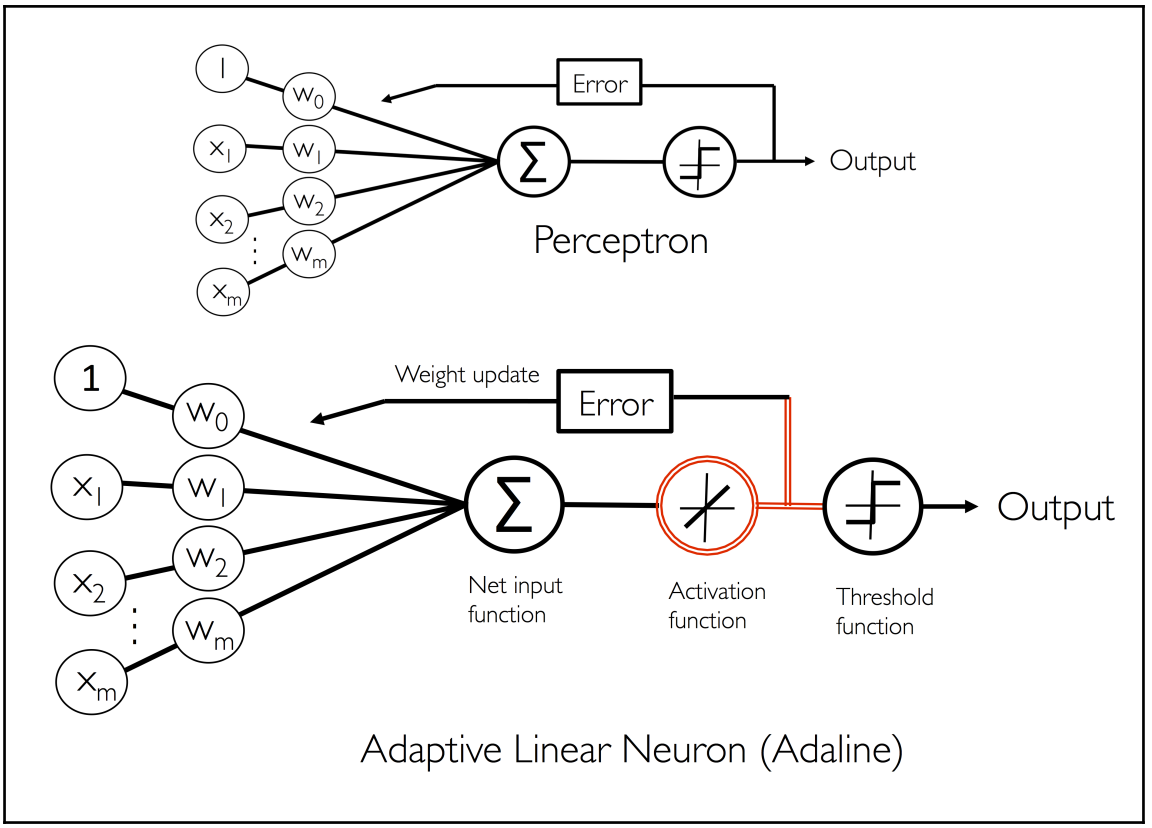

1
w0
x1
w1
x2
w2
wm
xm
Weight update
Error
Σ
Output
Net input function
Threshold function

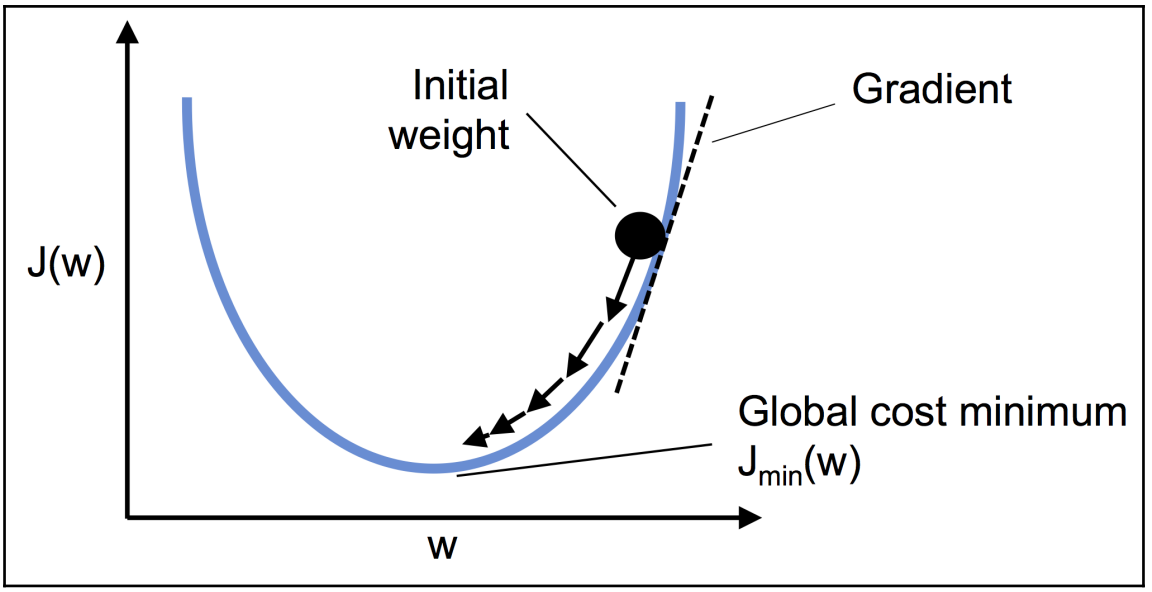
	0	1	2	3	4
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

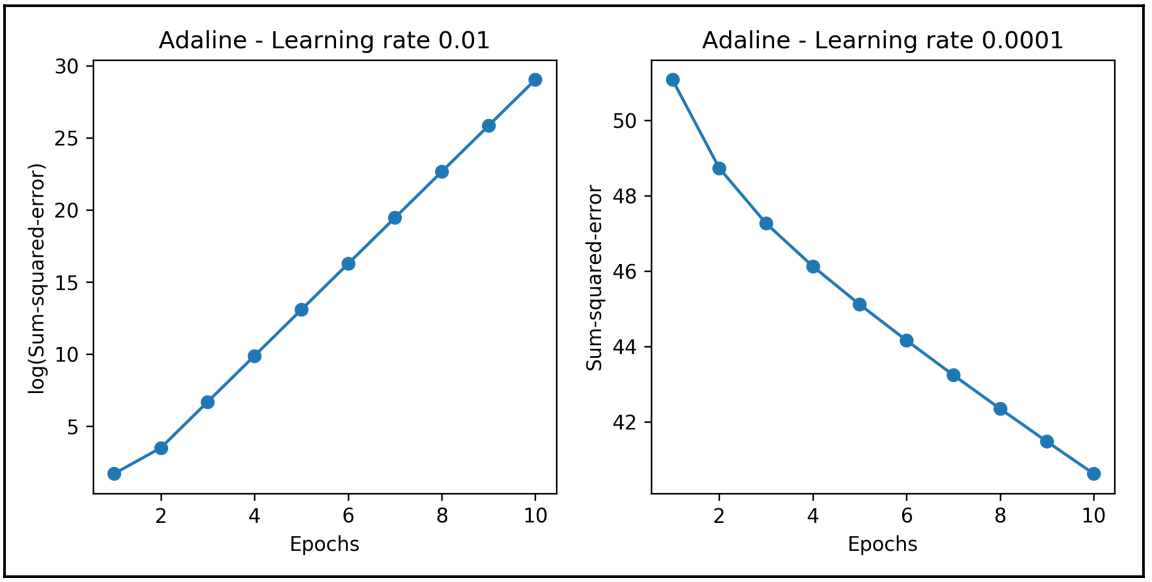

setosa
versicolor
petal length [cm]
sepal length [cm]
1
2
3
4
5
4.5
5.0
5.5
6.0
6.5
7.0

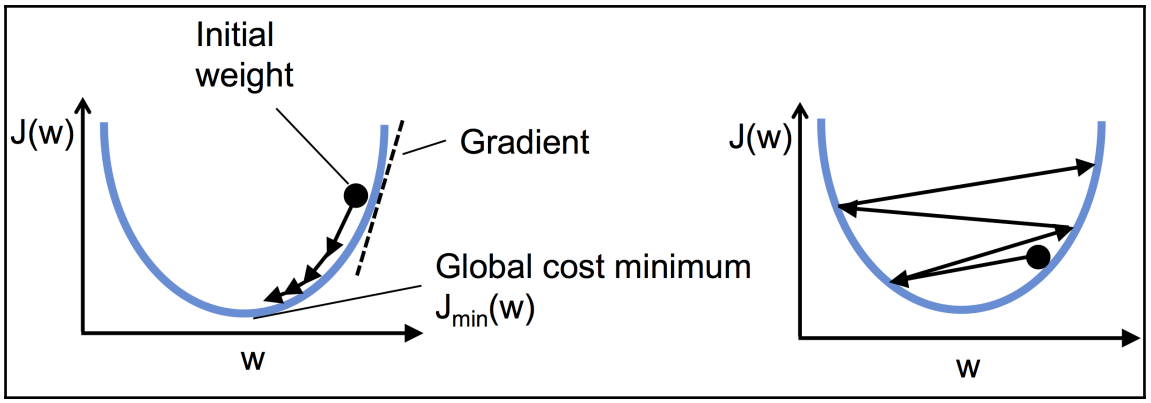

3.0
2.5
2.0
1.5
1.0
0.5
0.0
Number of updates
2
4
6
8
10
Epochs

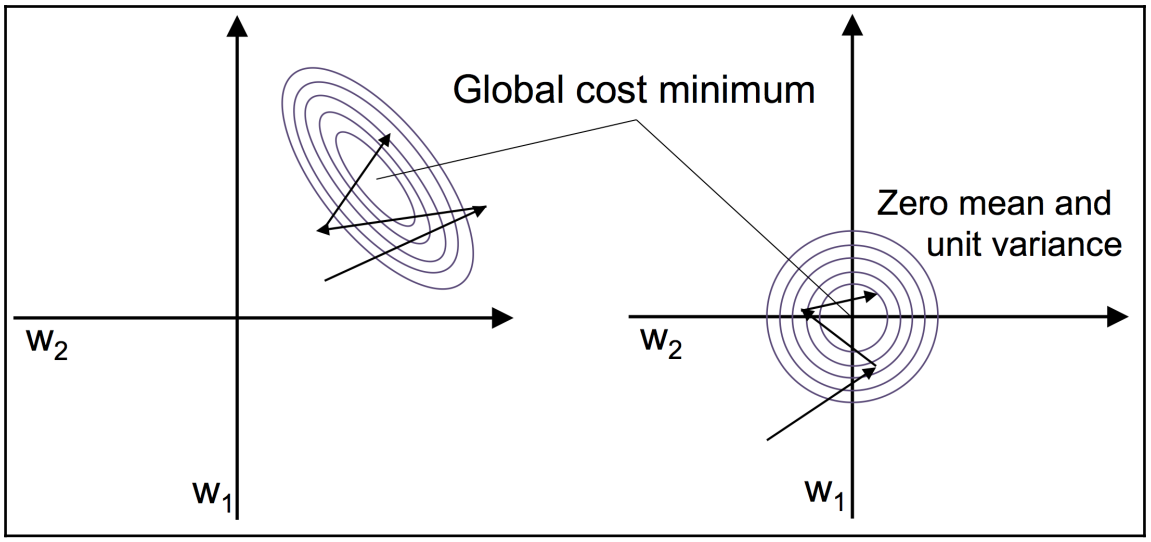

-1
1
petal length [cm]
sepal length [cm]
0
1
2
3
4
5
6
4
5
6
7

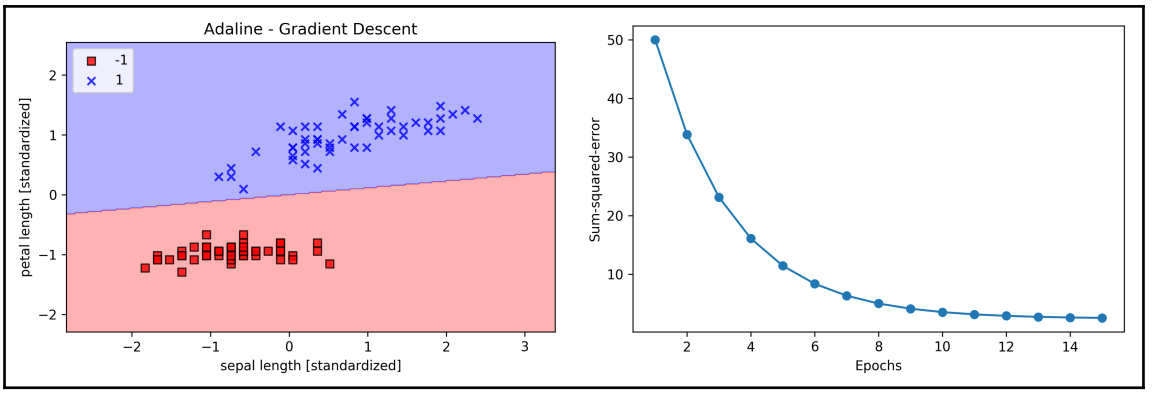

1
w_0
x_1
w_1
w_2
x_2
w_m
x_m
Error
Output
Perceptron
1
w_0
x_1
w_1
w_2
x_2
w_m
x_m
Weight update
Error
Output
Net input function
Activation function
Threshold function
Adaptive Linear Neuron (Adaline)

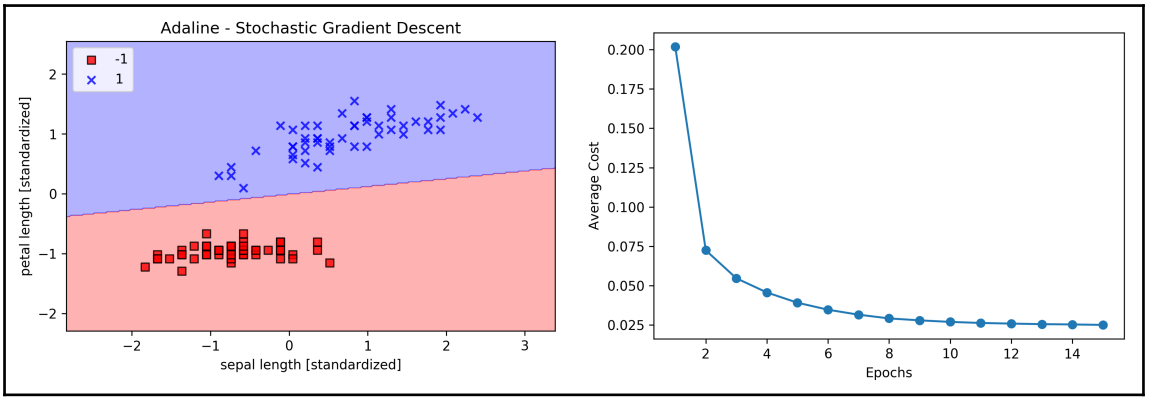

Initial
weight
Gradient
J(w)
Global cost minimum
$J_{min}(w)$
w


Adaline - Learning rate 0.01
log(Sum-squared-error)
Epochs
Adaline - Learning rate 0.0001
Sum-squared-error
Epochs

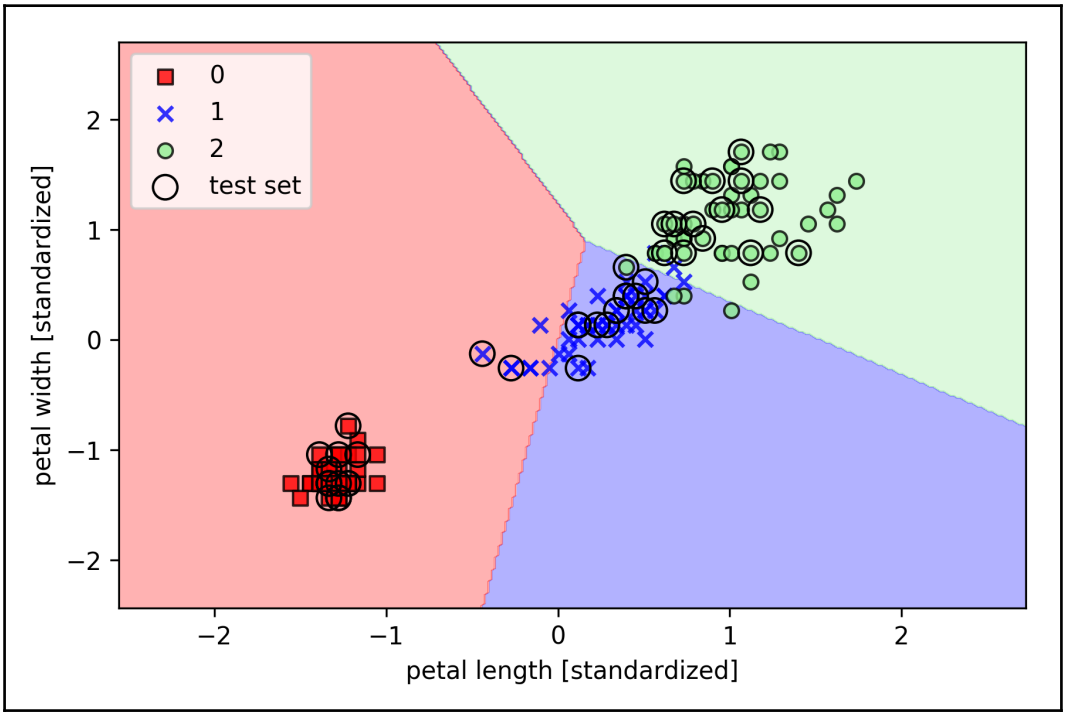

Initial
weight
J(w)
Gradient
Global cost minimum
$J_{min}(w)$
w
J(w)
w

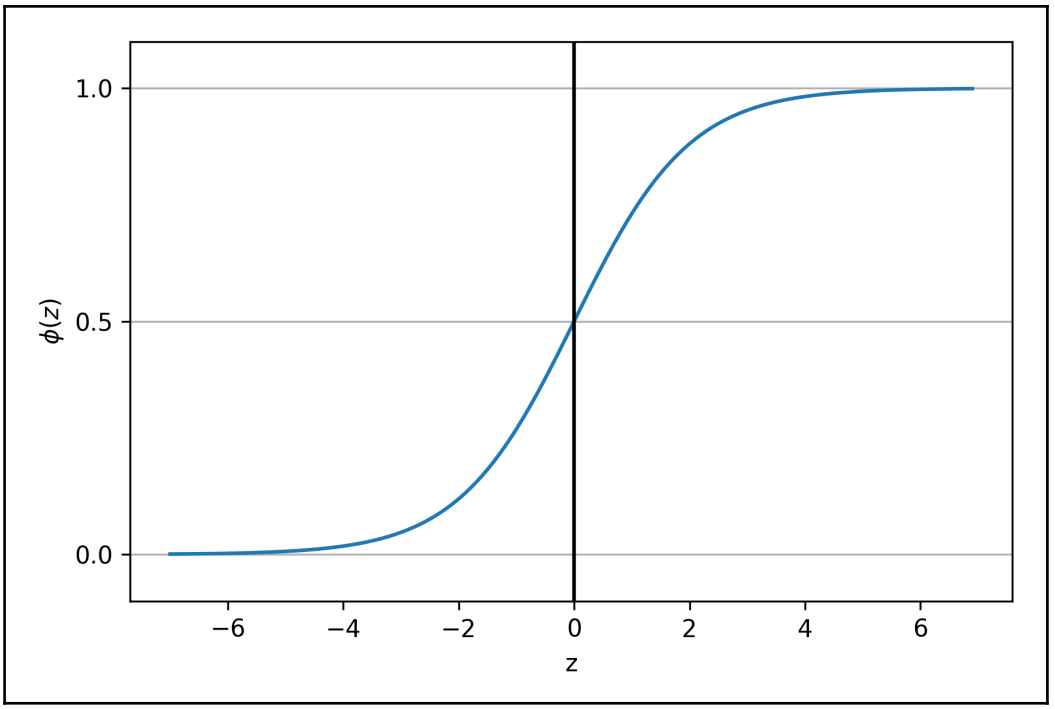

Global cost minimum
Zero mean and
unit variance
w_2
w_1
w_2
w_1

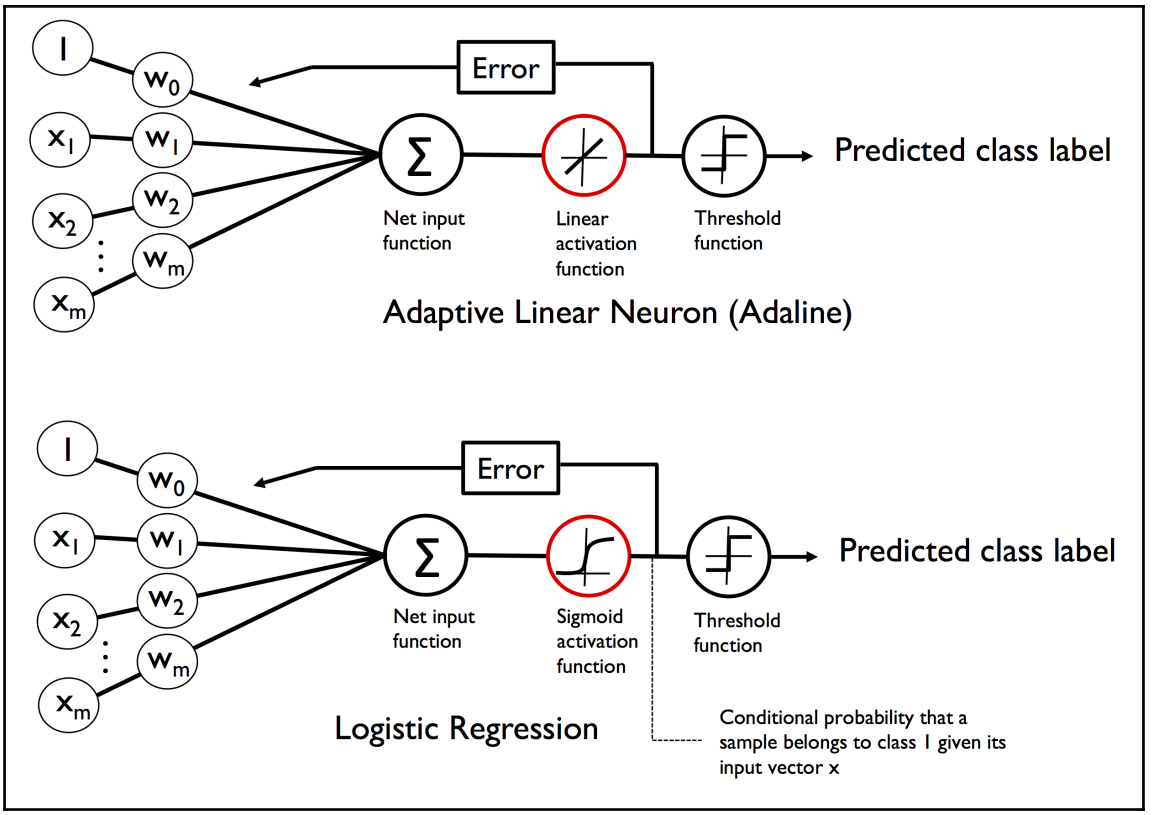

Adaline - Gradient Descent
-1
1
petal length [standardized]
sepal length [standardized]
Sum-squared-error
Epochs

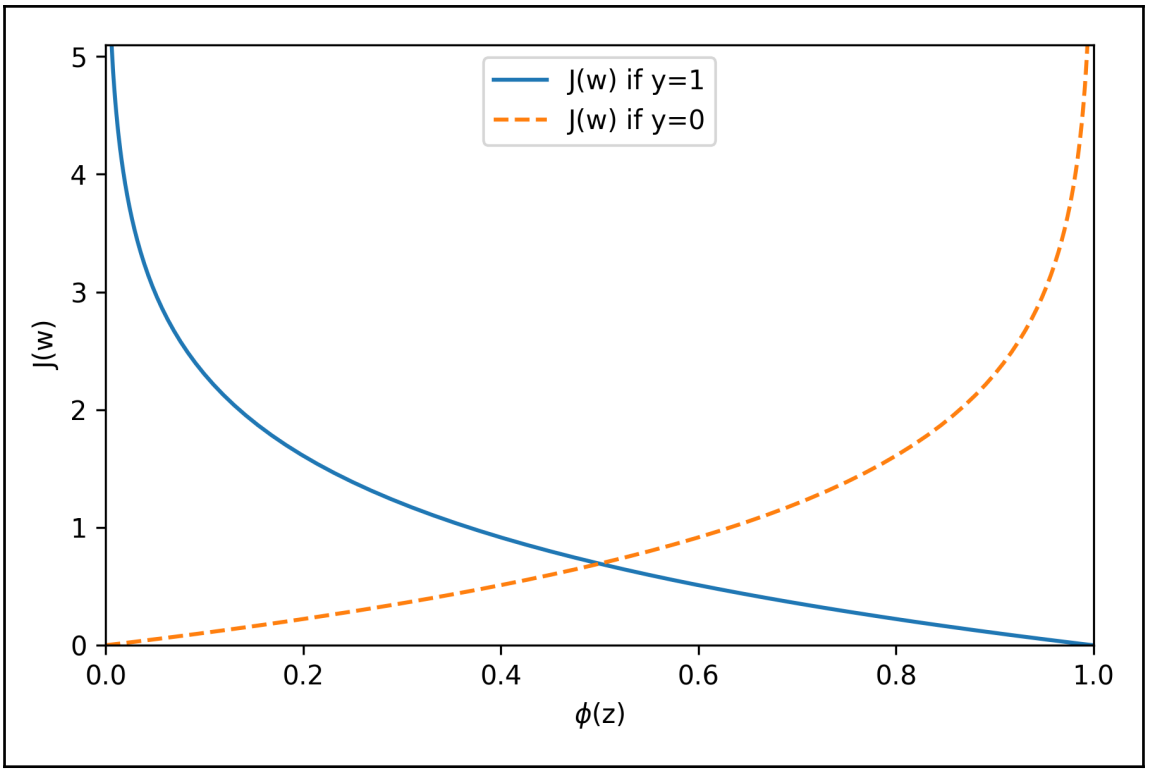

Adaline - Stochastic Gradient Descent
-1
1
petal length [standardized]
sepal length [standardized]
Average Cost
Epochs

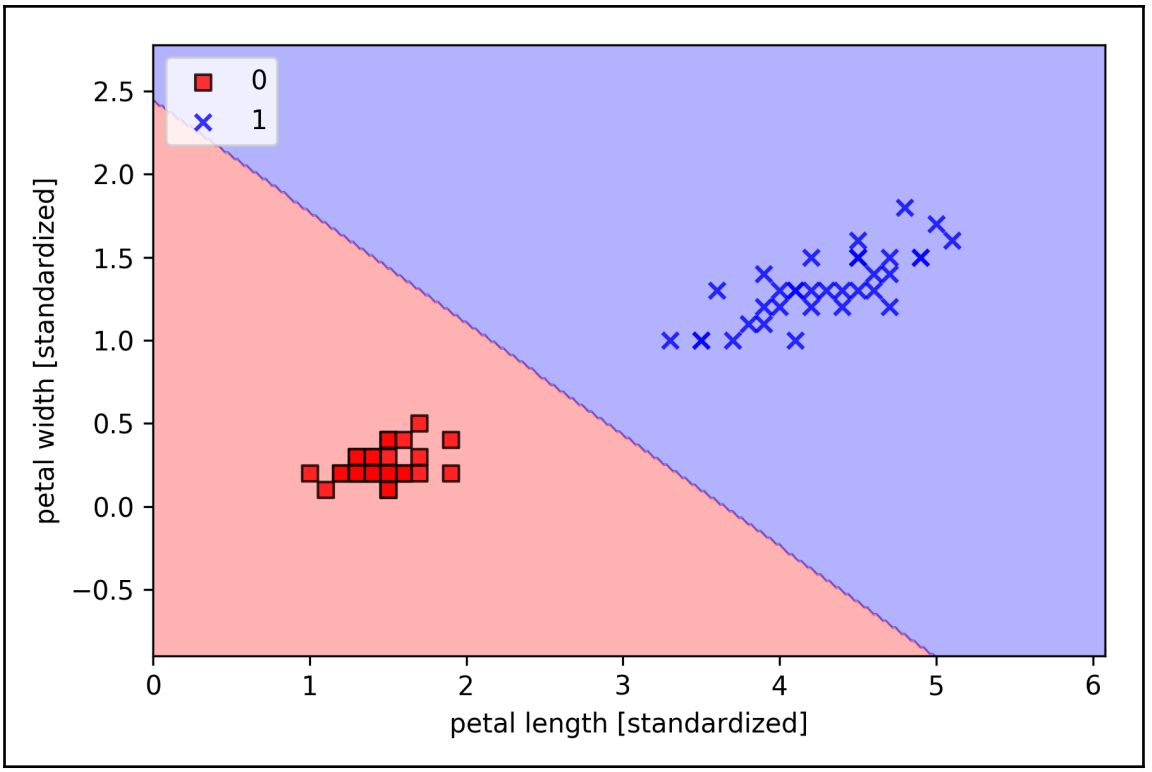
Chapter 3: A Tour of Machine Learning Classifiers Using scikit-learn



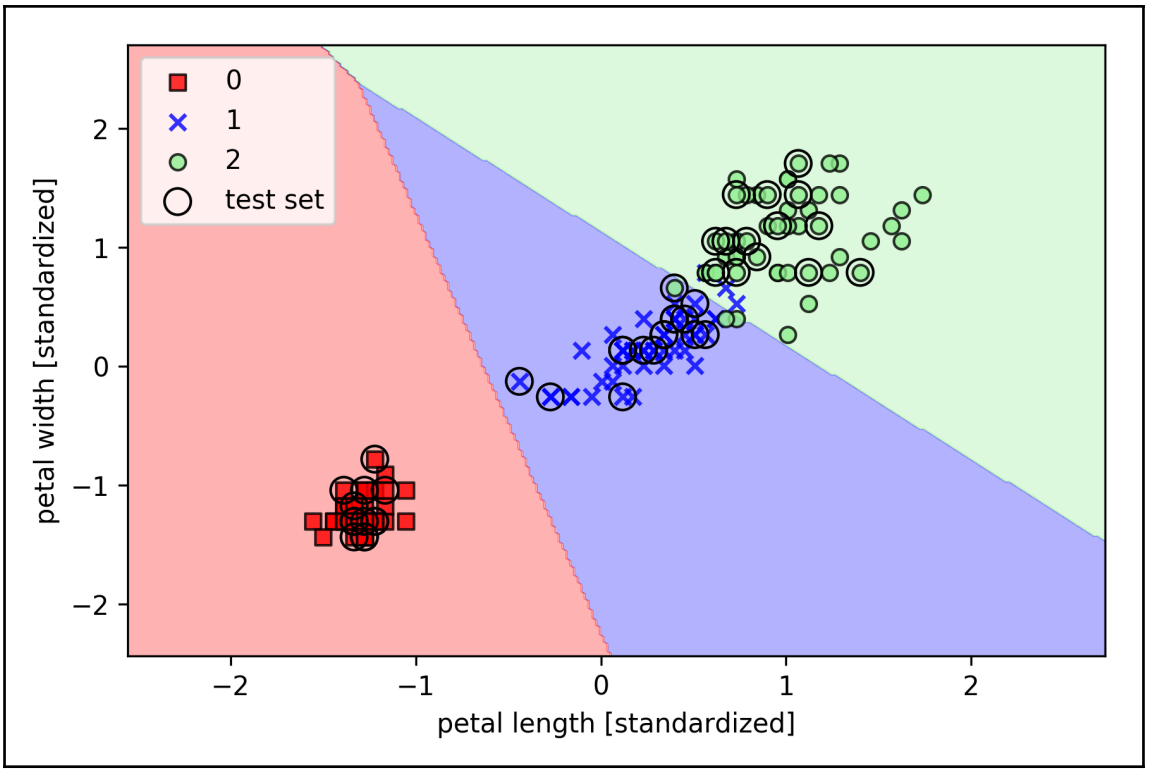

1.0
0.5
0.0
$\phi(z)$
−6
−4
−2
0
2
4
6
z

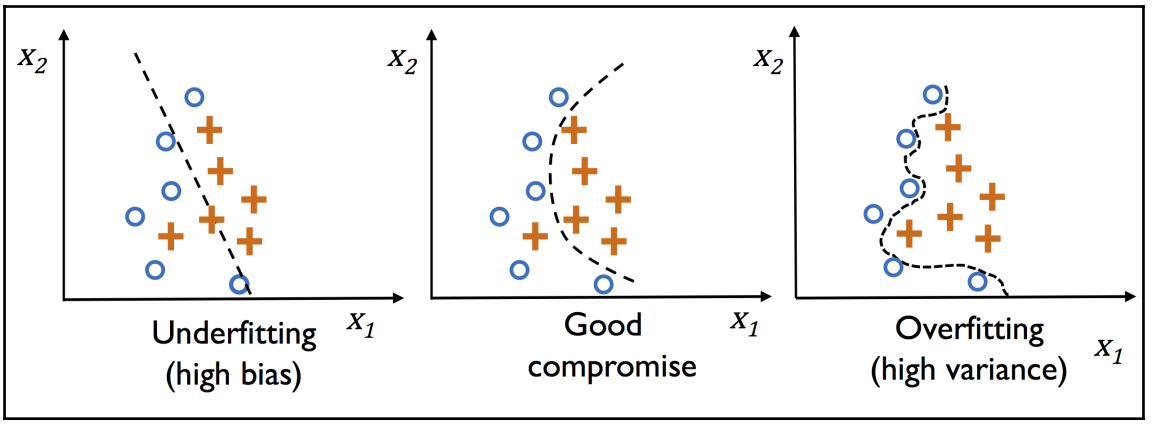

1
w0
x1
w1
x2
w2
wm
xm
Error
Σ
Net input function
Linear activation function
Threshold function
Predicted class label
Adaptive Linear Neuron (Adaline)
1
w0
x1
w1
x2
w2
wm
xm
Error
Σ
Net input function
Sigmoid activation function
Threshold function
Predicted class label
Logistic Regression
Conditional probability that a sample belongs to class 1 given its input vector x

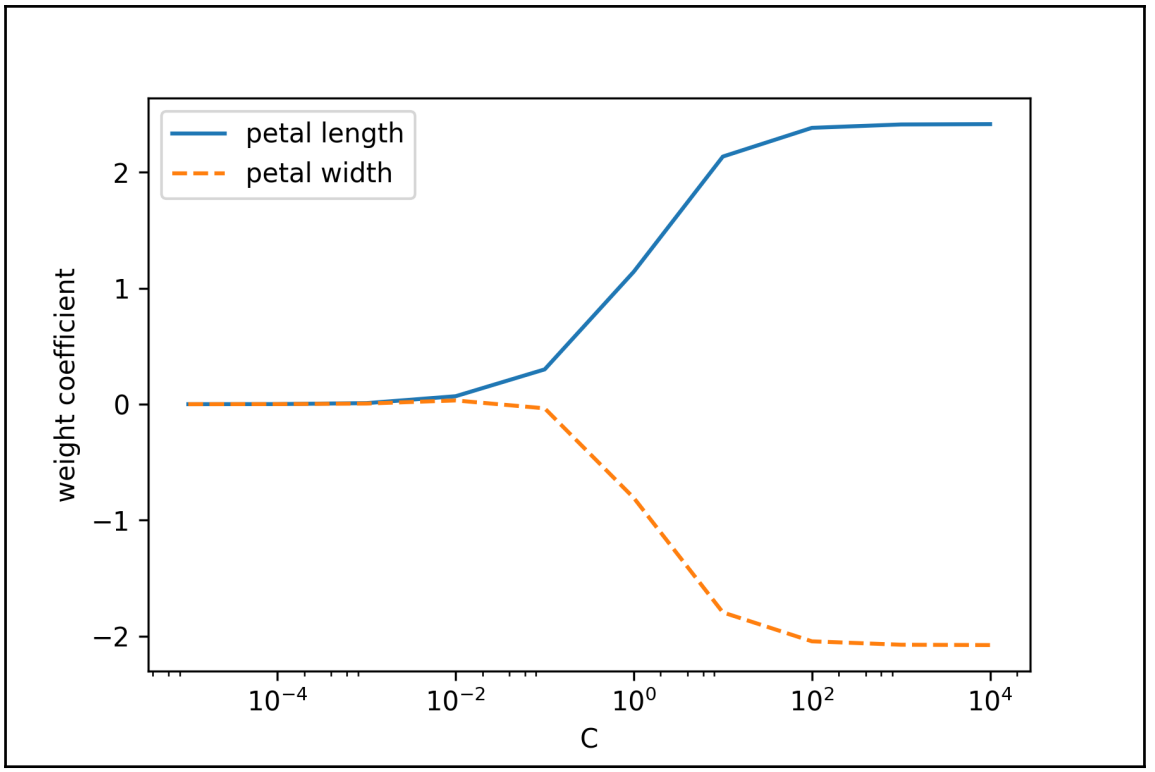

J(w) if y=1
J(w) if y=0
J(w)
ϕ(z)
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0

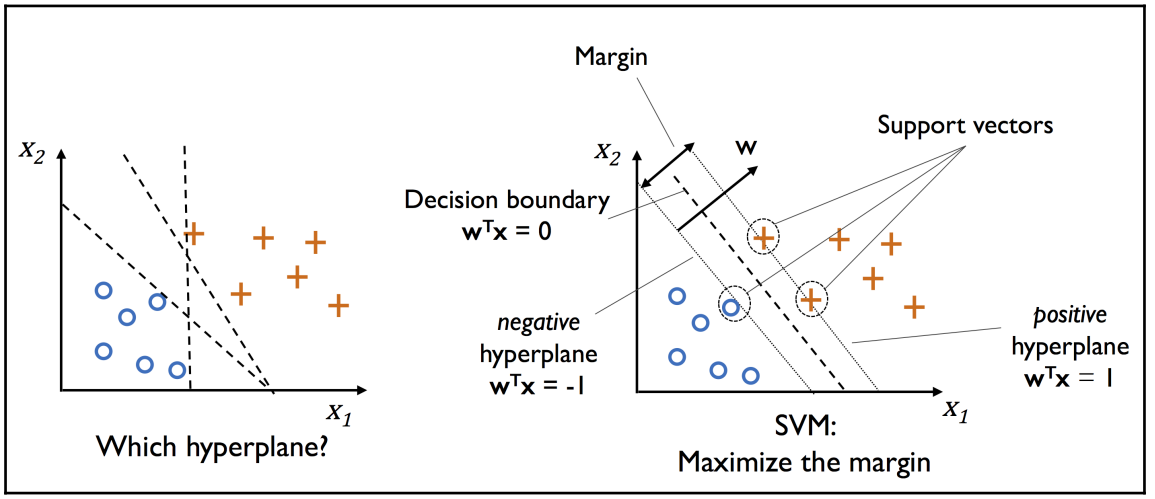

0
1
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
petal width [standardized]
0
1
2
3
4
5
6
petal length [standardized]

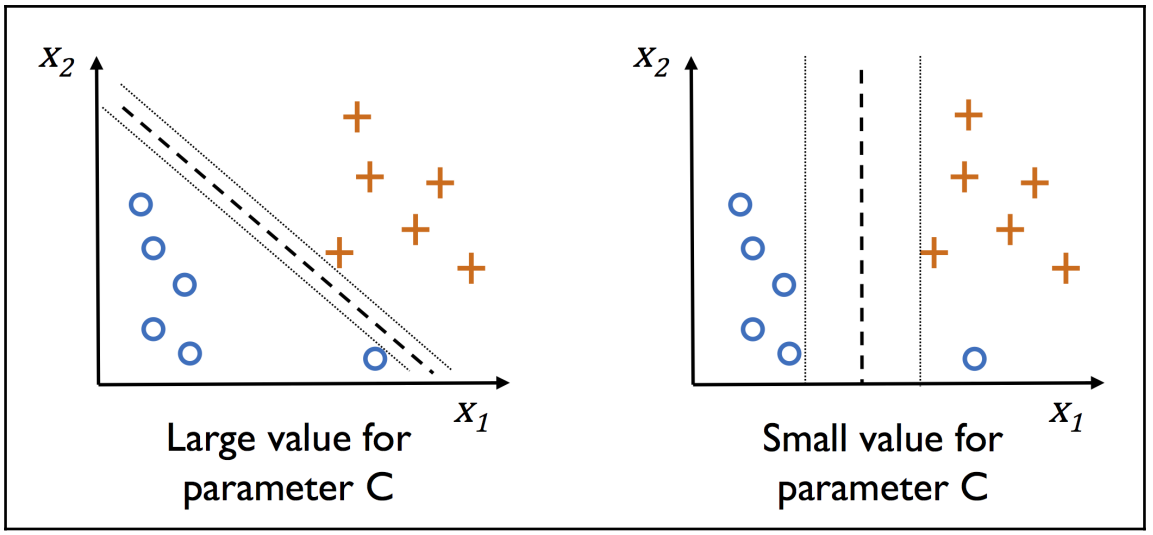
>

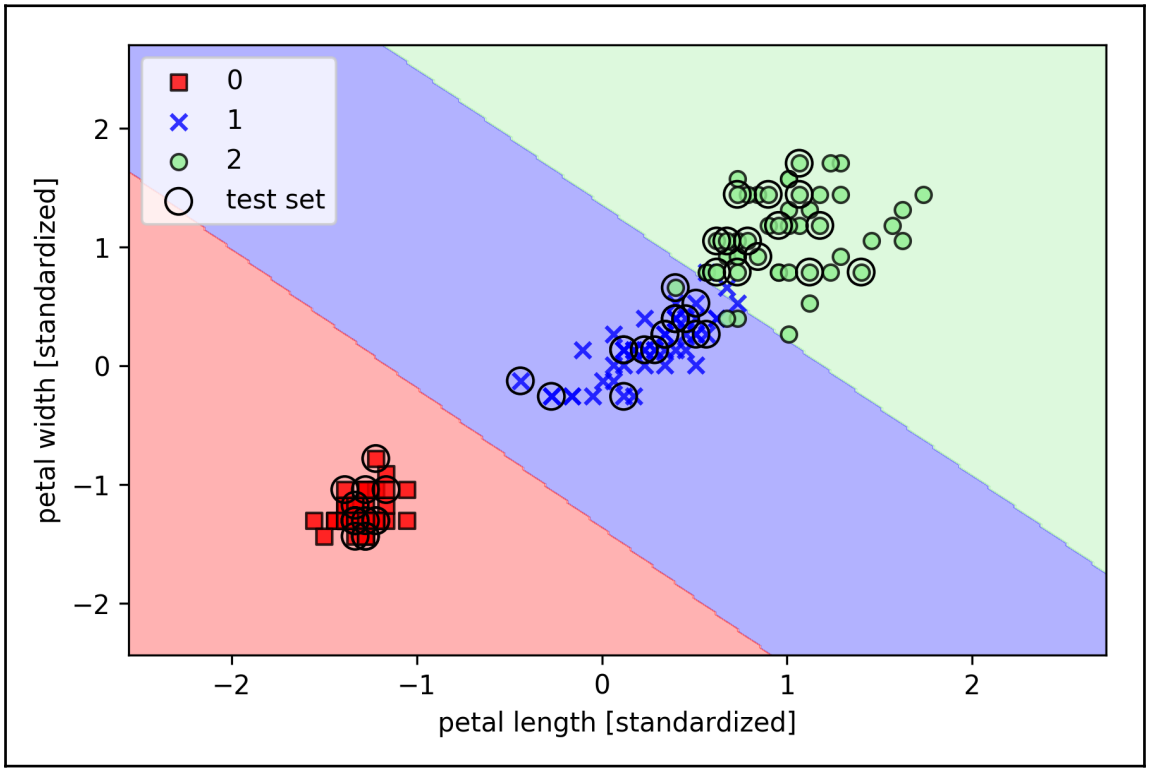

0
1
2
test set
petal width [standardized]
petal length [standardized]
2
1
0
−1
−2
−2
−1
0
1
2

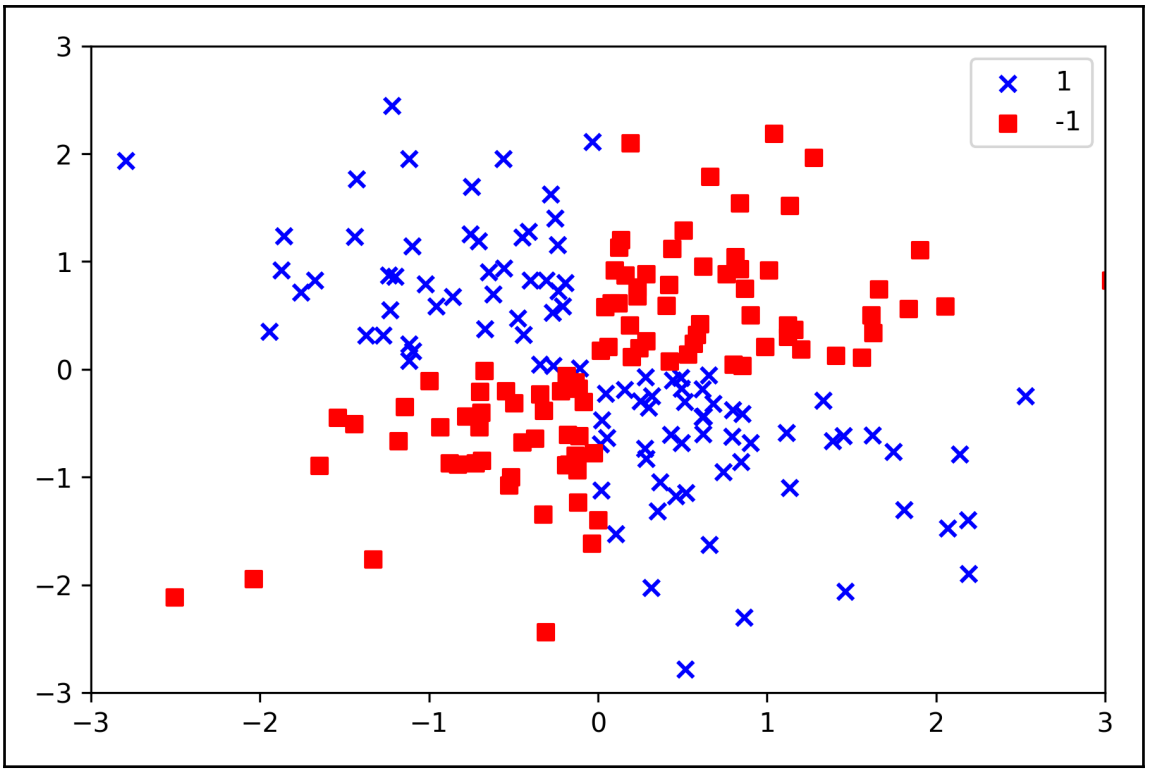

x_2
x_1
Underfitting
(high bias)
x_2
x_1
Good
compromise
x_2
x_1
Overfitting
(high variance)

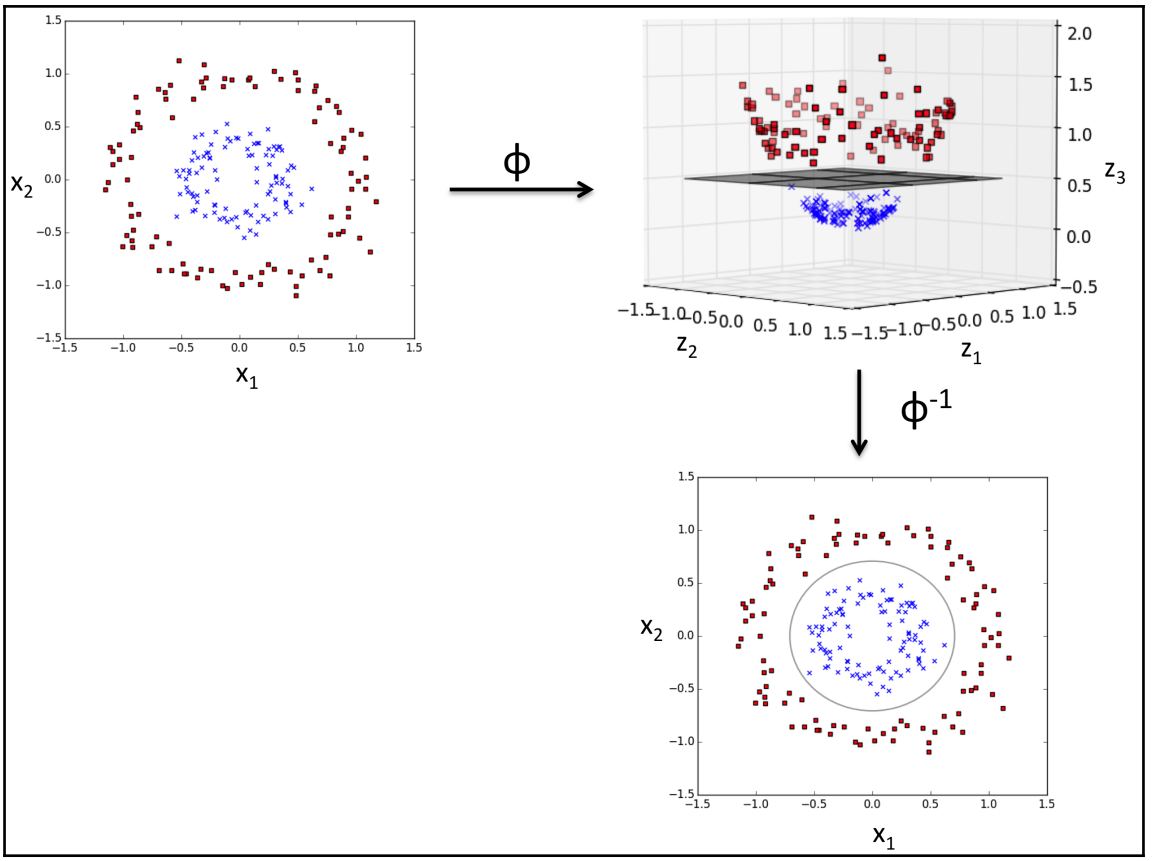

petal length
petal width
weight coefficient
2
1
0
−1
−2
10^{-4}
10^{-2}
10^{0}
10^{2}
10^{4}
C

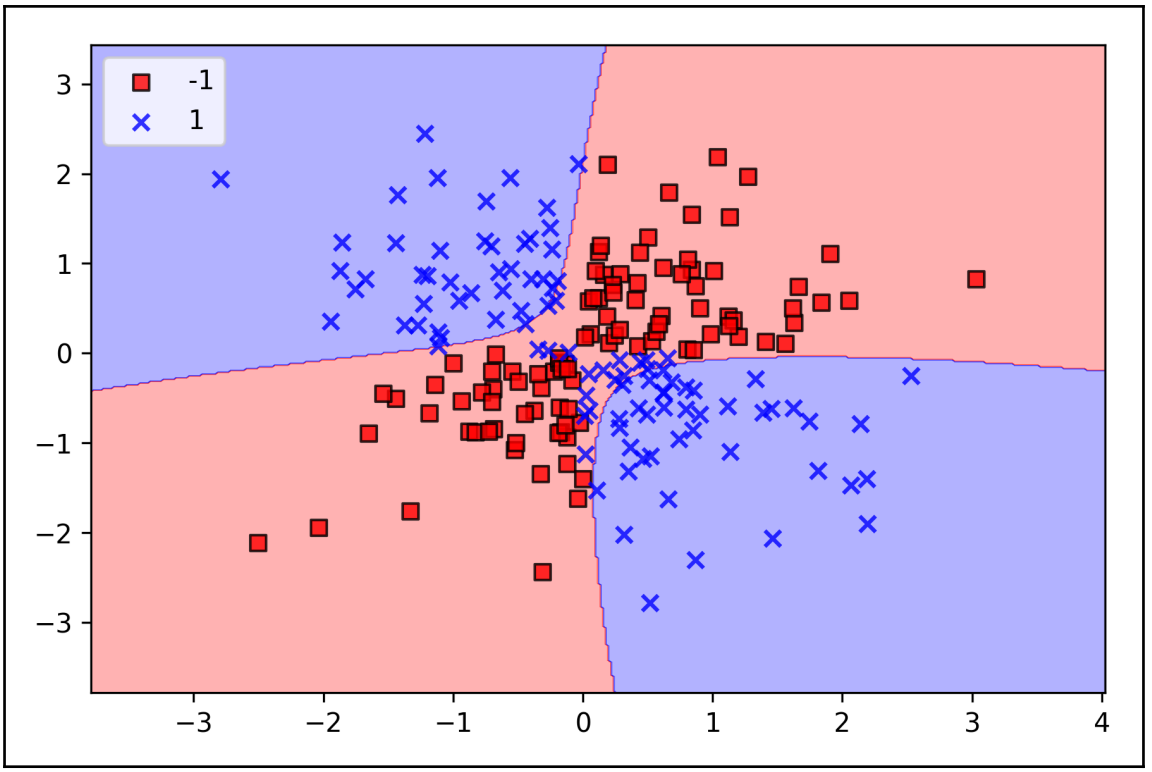

x_2
x_1
Which hyperplane?
Margin
Support vectors
x_2
w
Decision boundary
$w^Tx = 0$
negative
hyperplane
$w^Tx = -1$
positive
hyperplane
$w^Tx = 1$
x_1
SVM:
Maximize the margin

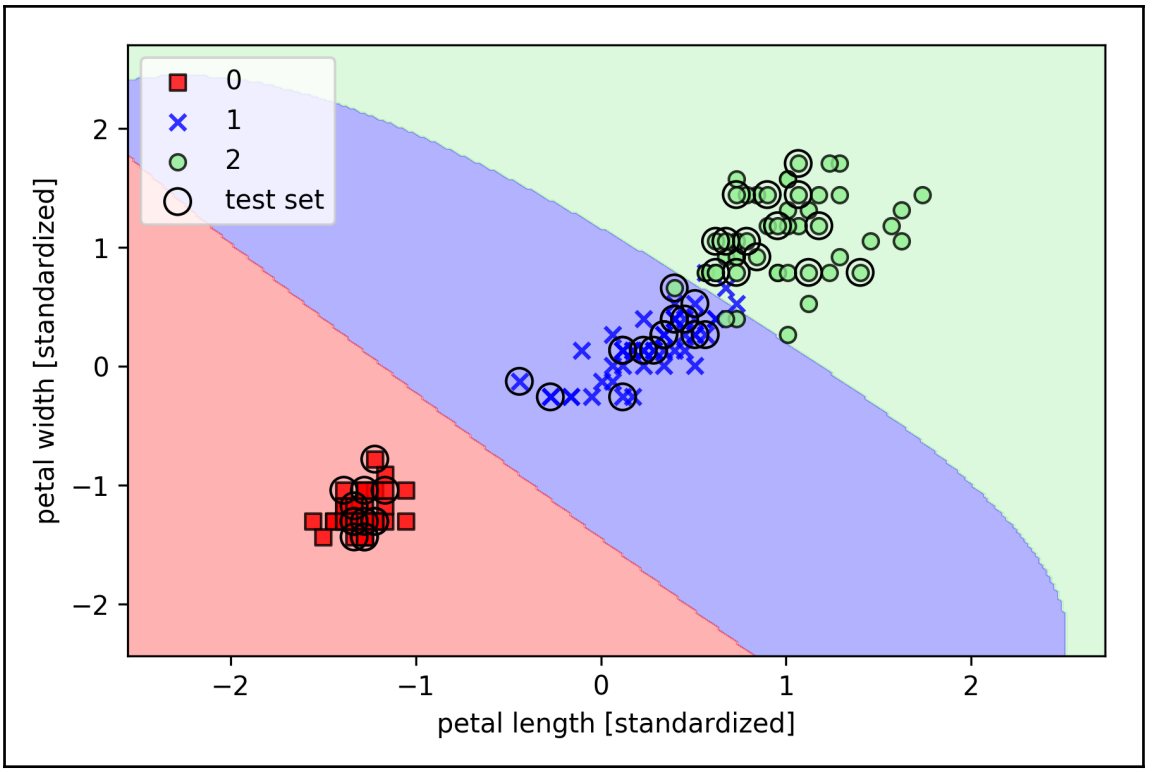

x_2
x_1
Large value for parameter C
x_2
x_1
Small value for parameter C

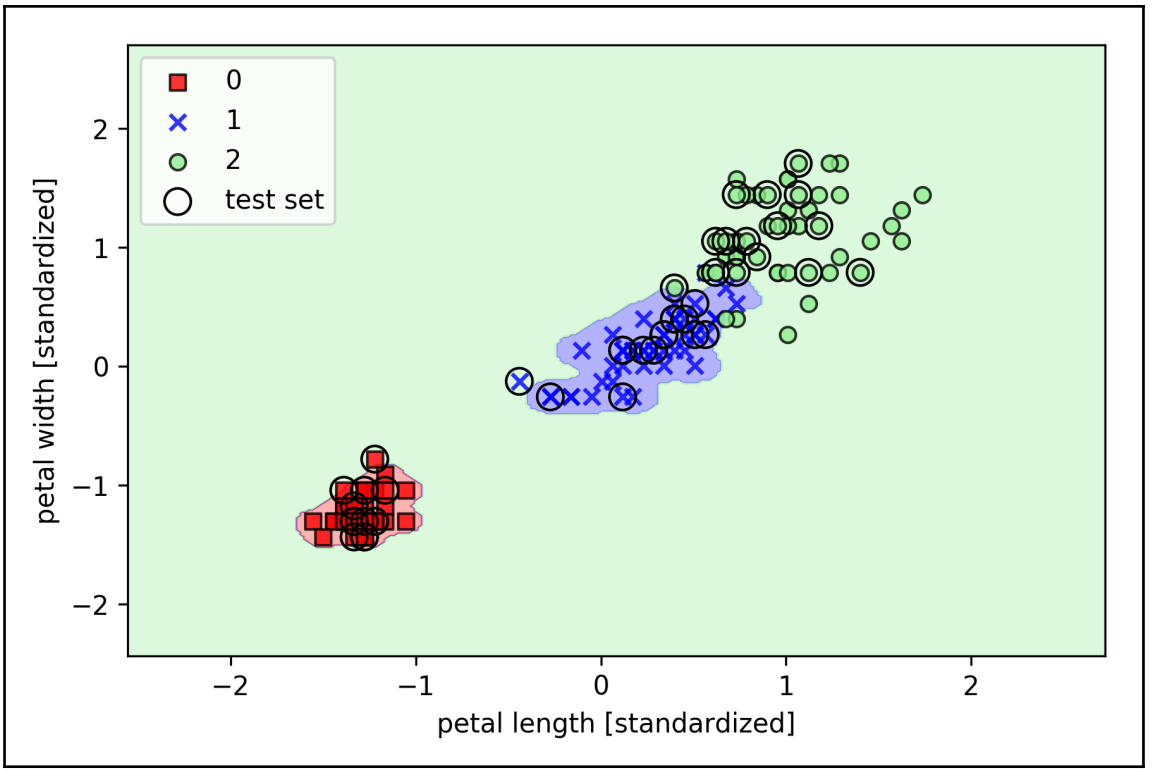

0
1
2
test set
petal width [standardized]
petal length [standardized]
2
1
0
−1
−2
−2
−1
0
1
2

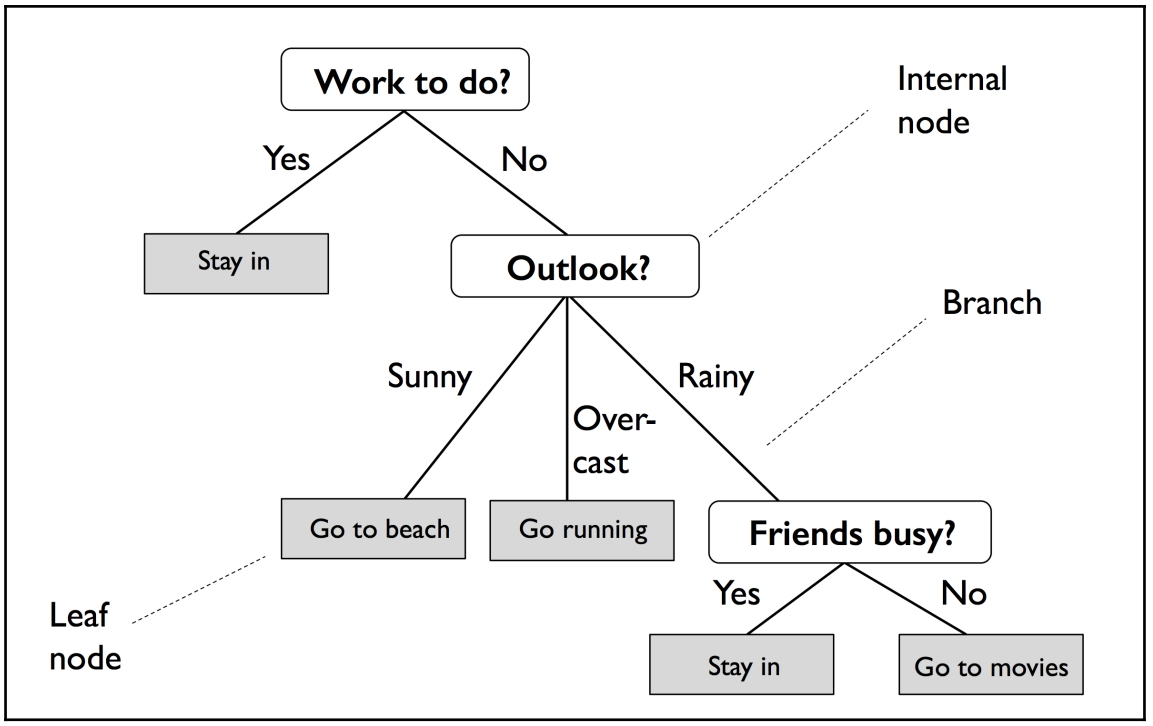

1
-1
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

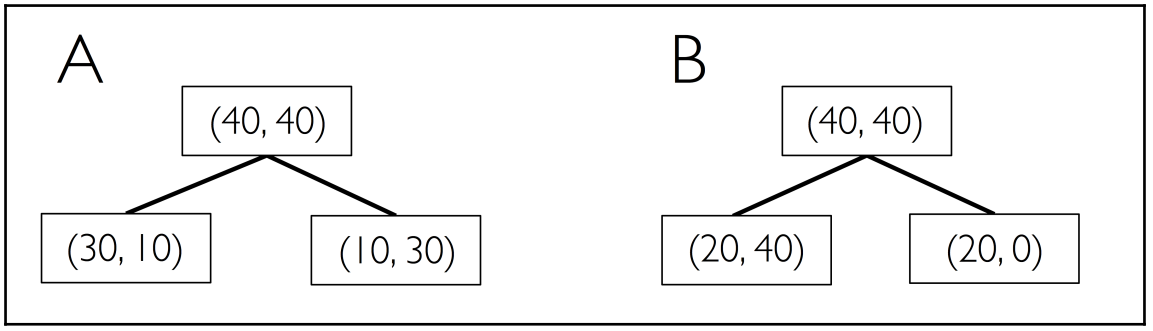

x2
x1
ϕ
z3
z2
z1
ϕ-1
x2
x1

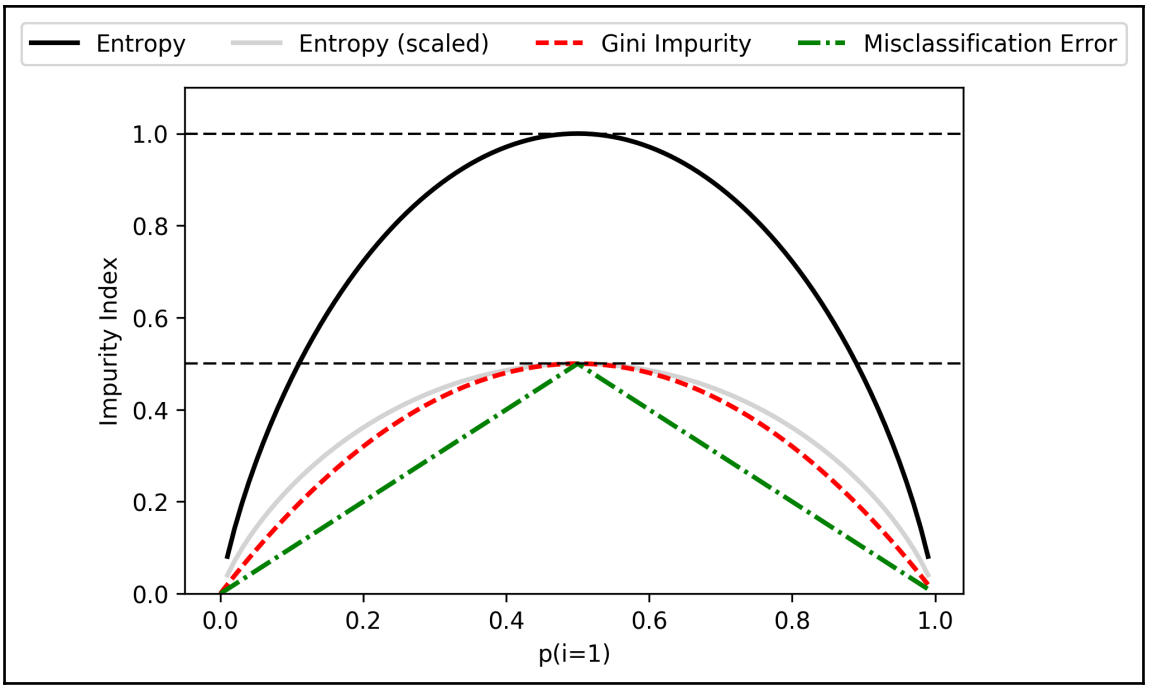

-1
1
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3
4

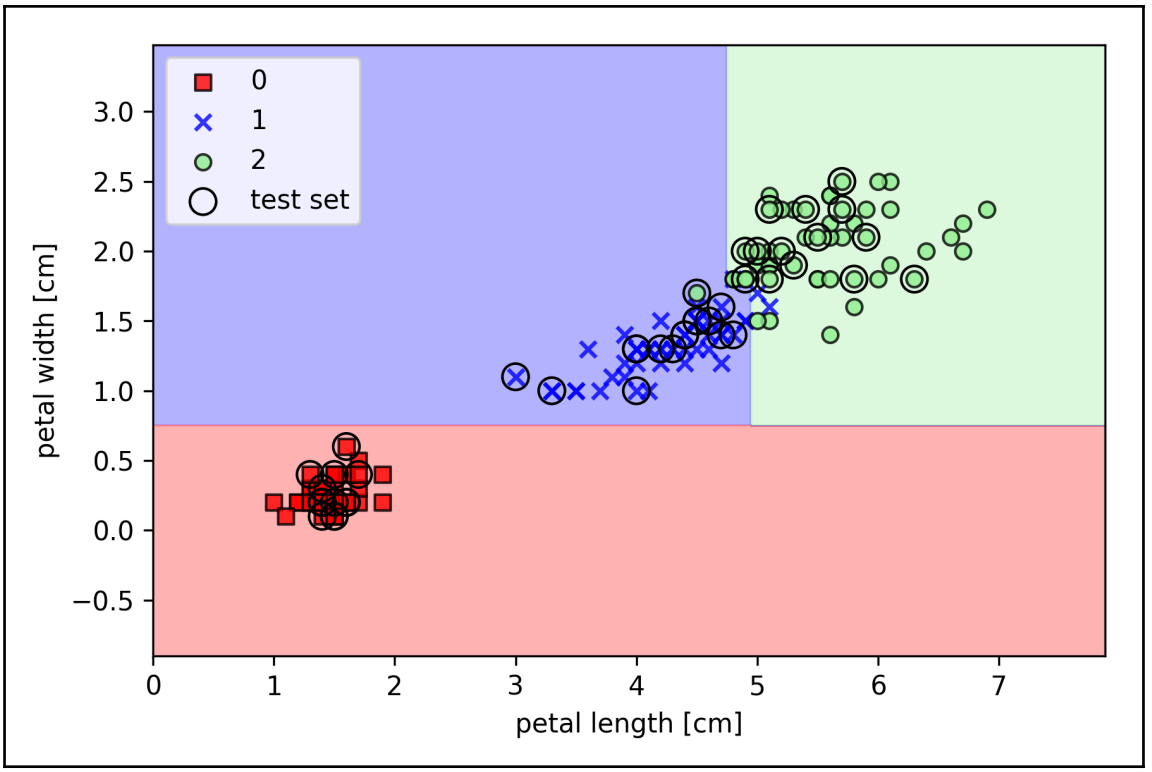

0
1
2
test set
petal width [standardized]
petal length [standardized]
−2
−1
0
1
2

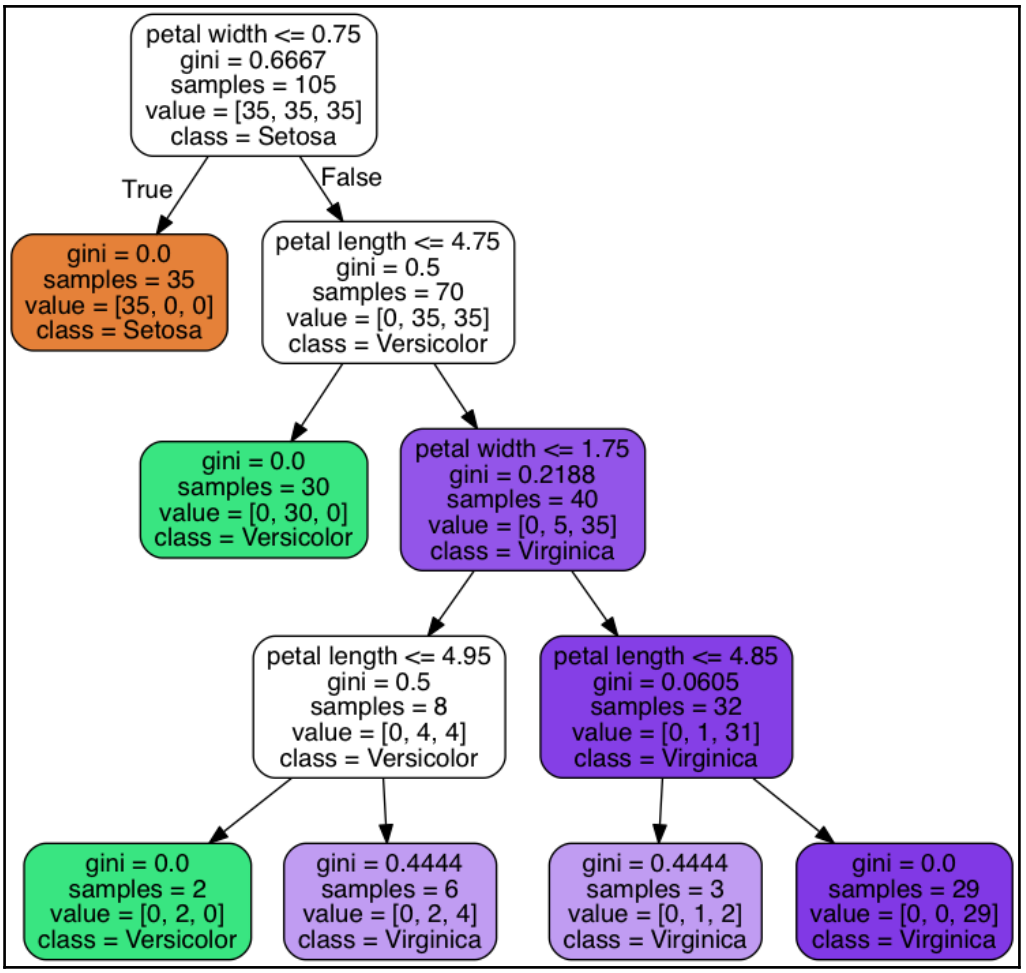

0
1
2
test set
petal width [standardized]
petal length [standardized]
−2
−1
0
1
2

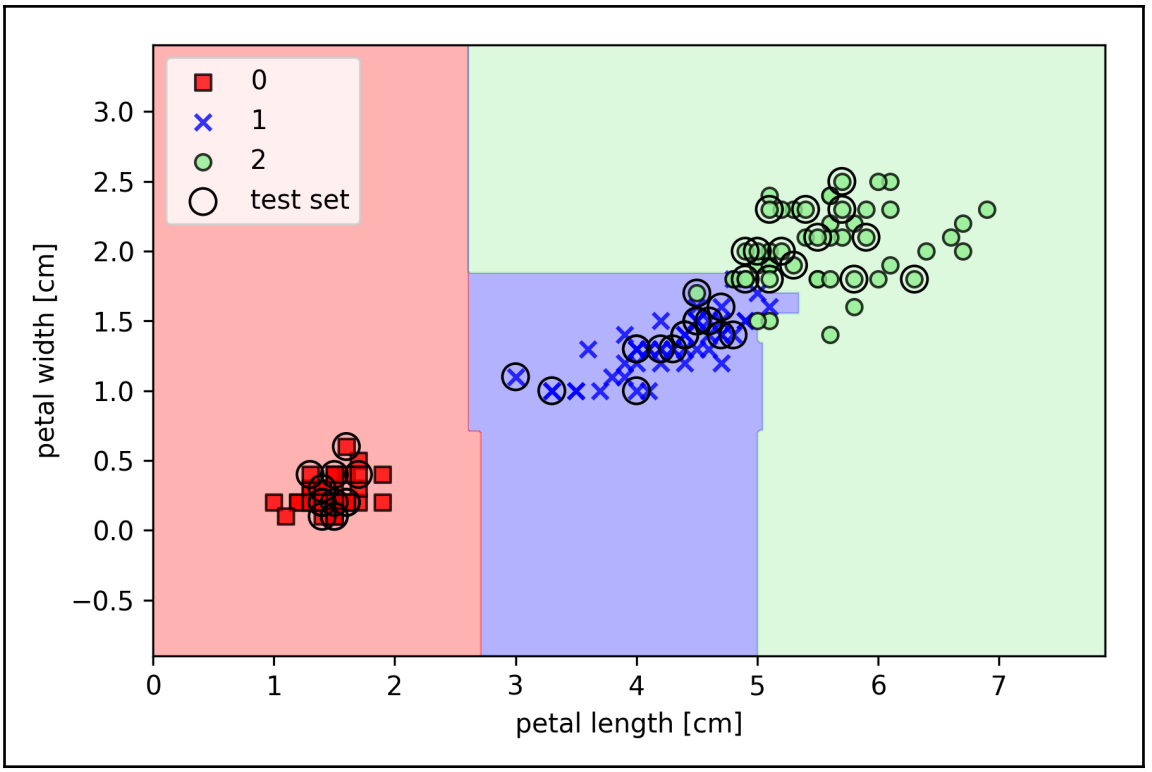

Work to do?
Yes
No
Stay in
Outlook?
Internal node
Branch
Sunny
Over-cast
Rainy
Go to beach
Go running
Friends busy?
Yes
No
Leaf node
Stay in
Go to movies

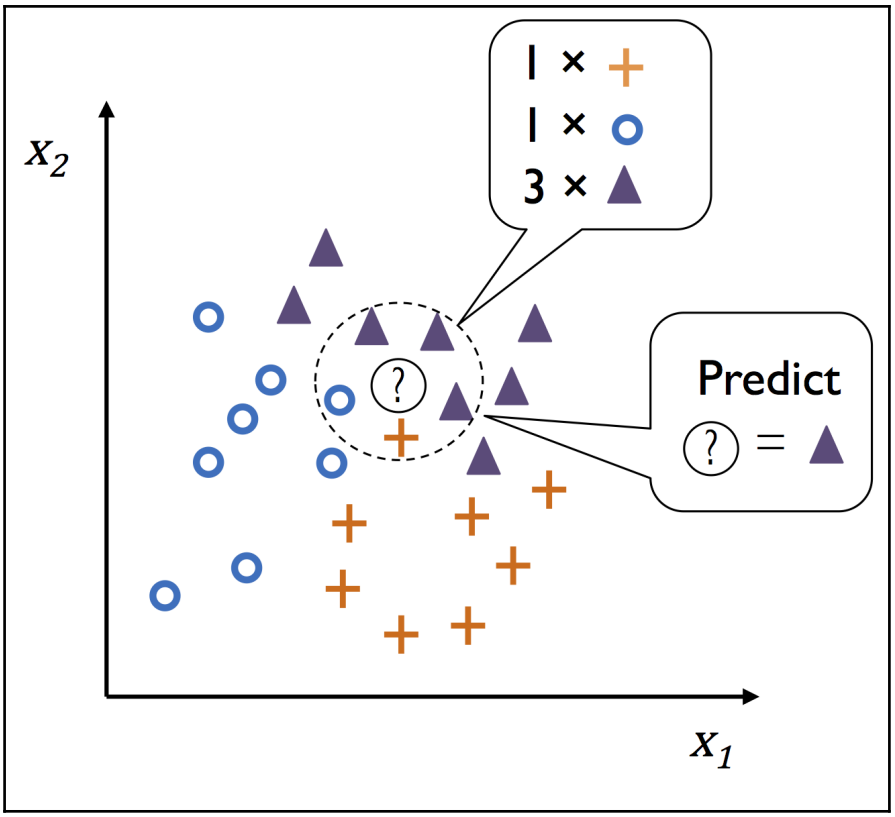

A
(40, 40)
(30, 10)
(10, 30)
B
(40, 40)
(20, 40)
(20, 0)

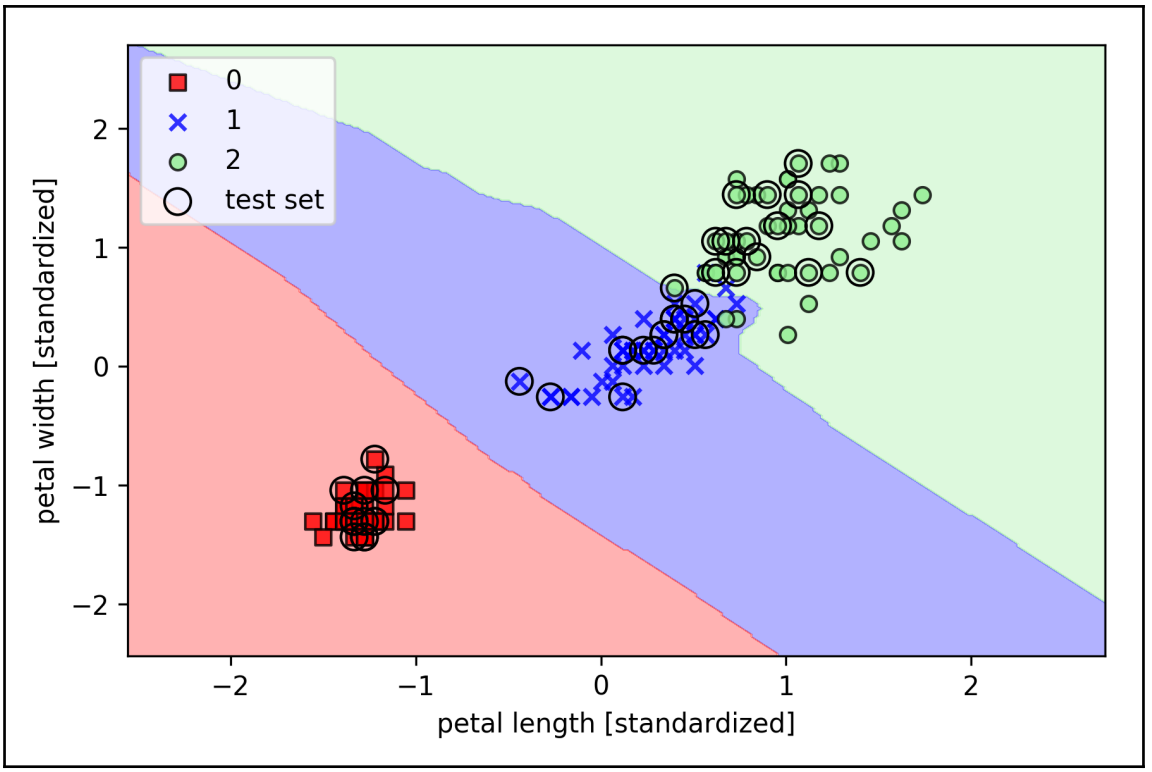

Entropy
Entropy (scaled)
Gini Impurity
Misclassification Error
Impurity Index
p(i=1)
0.0
0.2
0.4
0.6
0.8
1.0


0
1
2
test set
petal width [cm]
petal length [cm]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
1
2
3
4
5
6
7

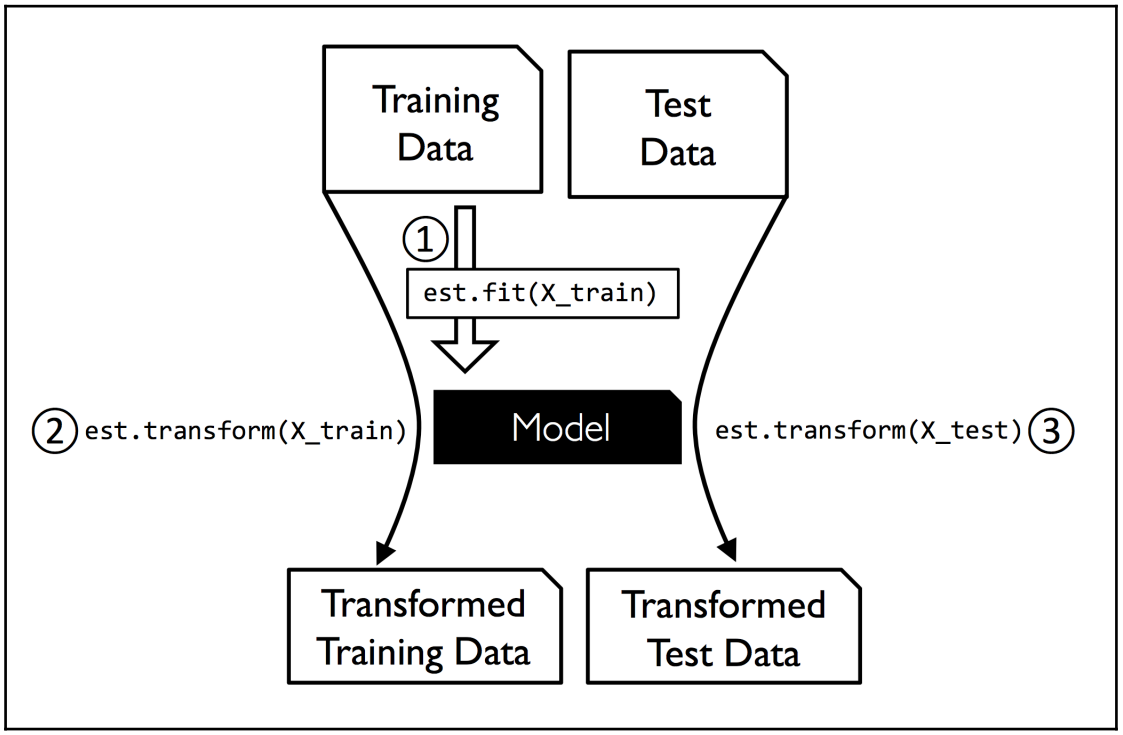

petal width <= 0.75
gini = 0.6667
samples = 105
value = [35, 35, 35]
class = Setosa
True
False
gini = 0.0
samples = 35
value = [35, 0, 0]
class = Setosa
petal length <= 4.75
gini = 0.5
samples = 70
value = [0, 35, 35]
class = Versicolor
gini = 0.0
samples = 30
value = [0, 30, 0]
class = Versicolor
petal width <= 1.75
gini = 0.2188
samples = 40
value = [0, 5, 35]
class = Virginica
petal length <= 4.95
gini = 0.5
samples = 8
value = [0, 4, 4]
class = Versicolor
petal length <= 4.85
gini = 0.0605
samples = 32
value = [0, 1, 31]
class = Virginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = Versicolor
gini = 0.4444
samples = 6
value = [0, 2, 4]
class = Virginica
gini = 0.4444
samples = 3
value = [0, 1, 2]
class = Virginica
gini = 0.0
samples = 29
value = [0, 0, 29]
class = Virginica

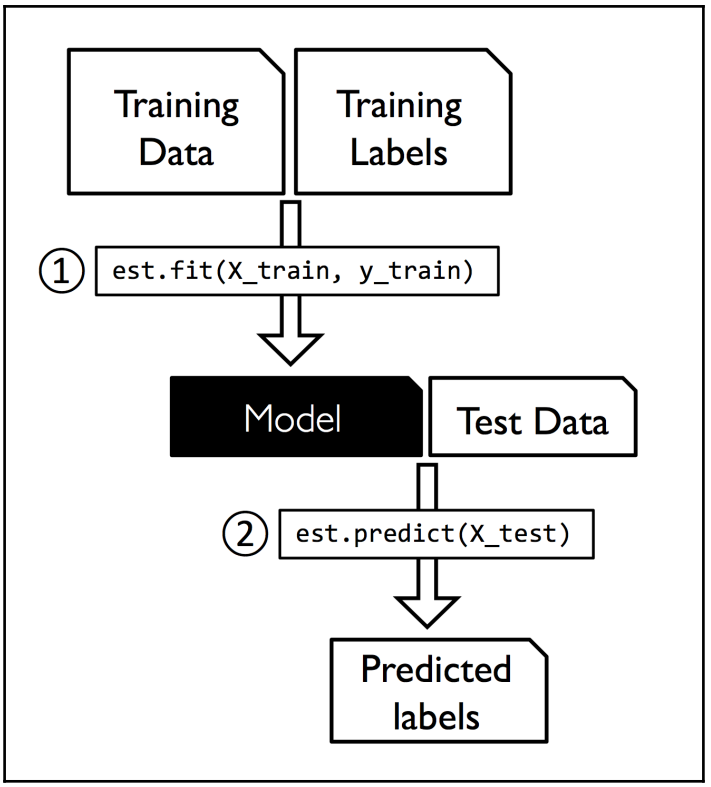

0
1
2
test set
petal width [cm]
petal length [cm]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
1
2
3
4
5
6
7


1 ×
1 ×
3 ×
Predict
? =
?
x_2
x_1

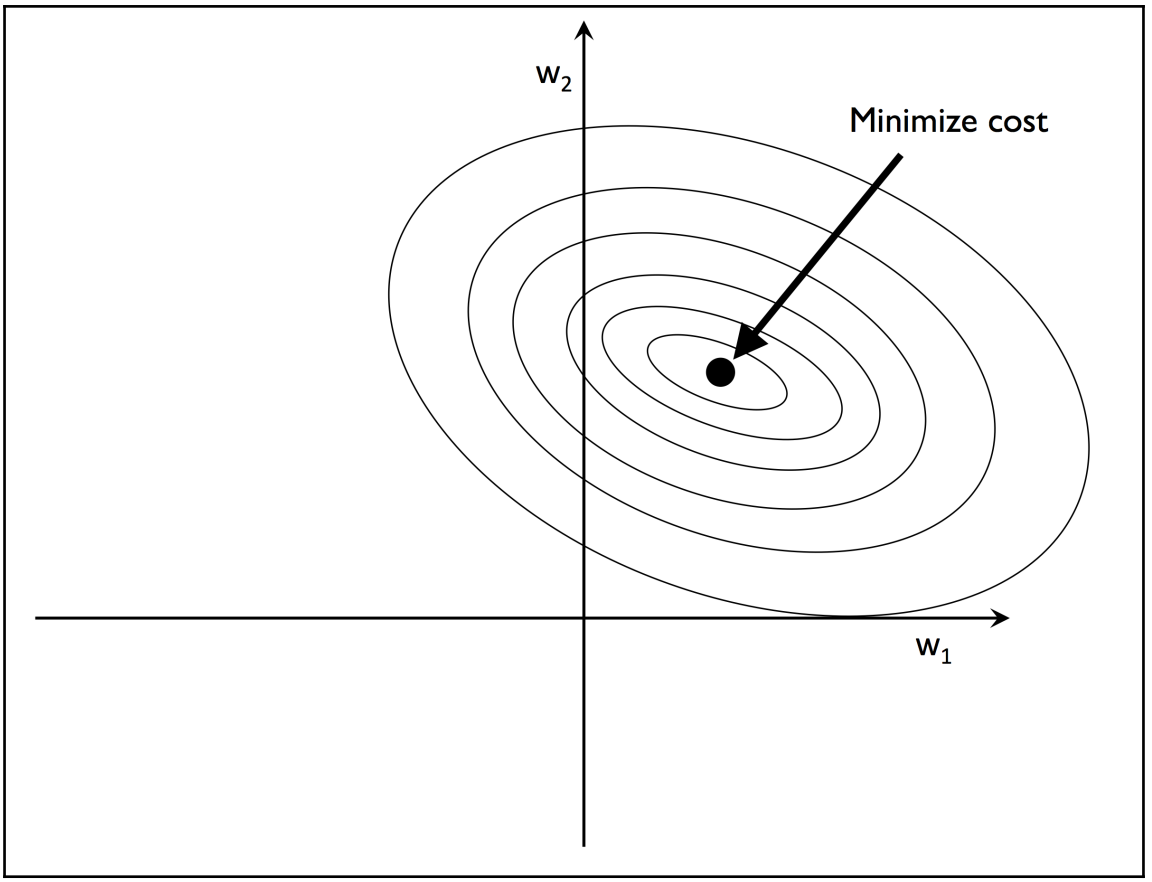

0
1
2
test set
petal width [standardized]
petal length [standardized]
2
1
0
−1
−2
−2
−1
0
1
2

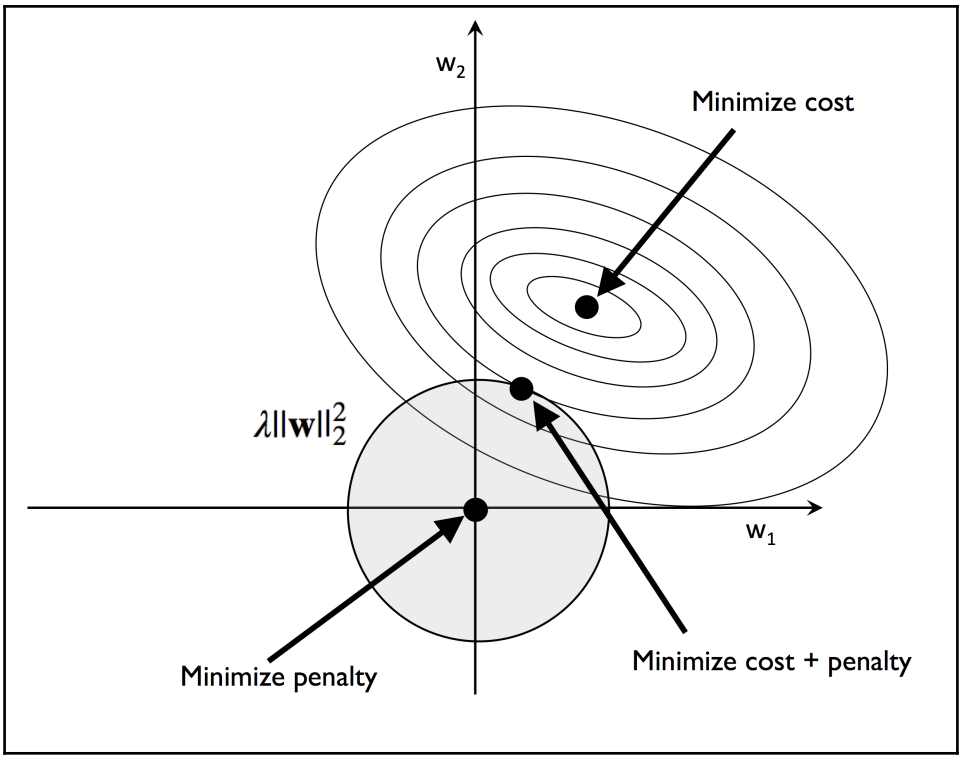
Chapter 4: Building Good Training Sets – Data Preprocessing

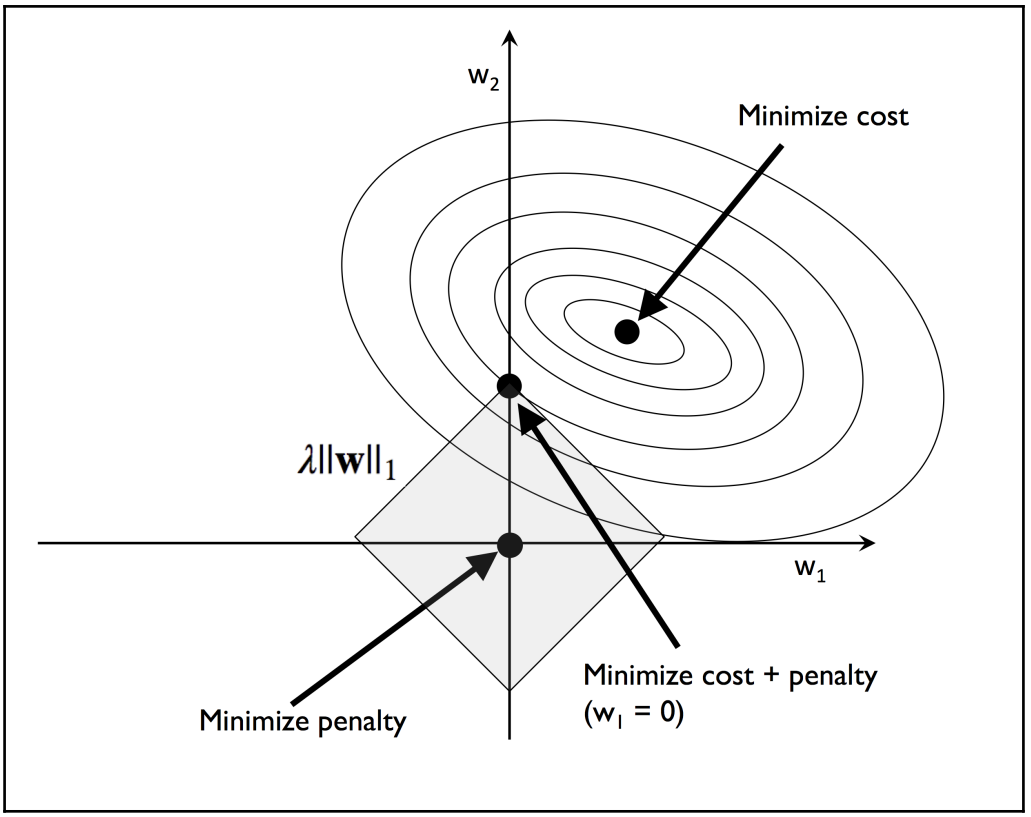


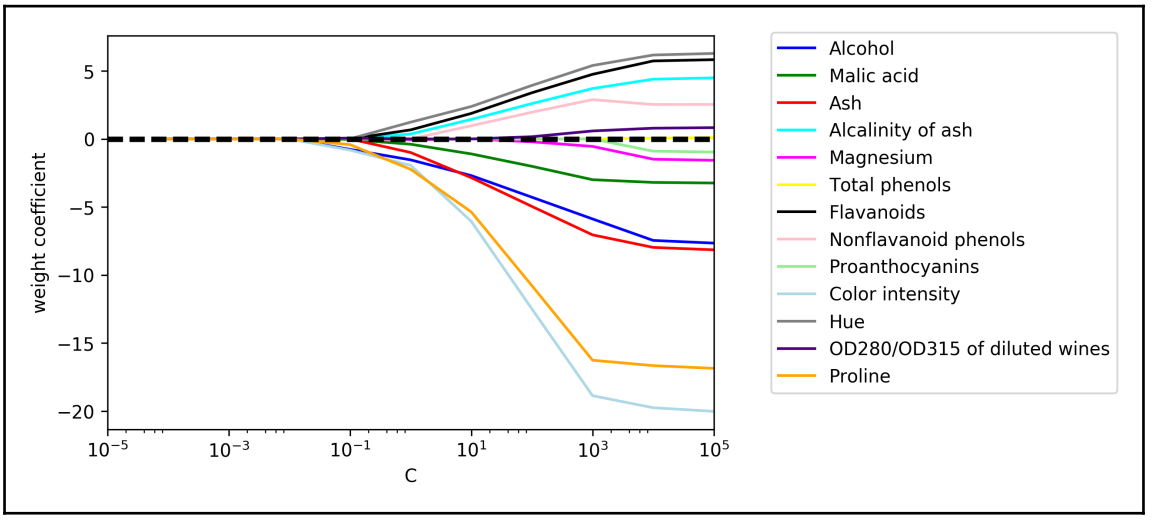


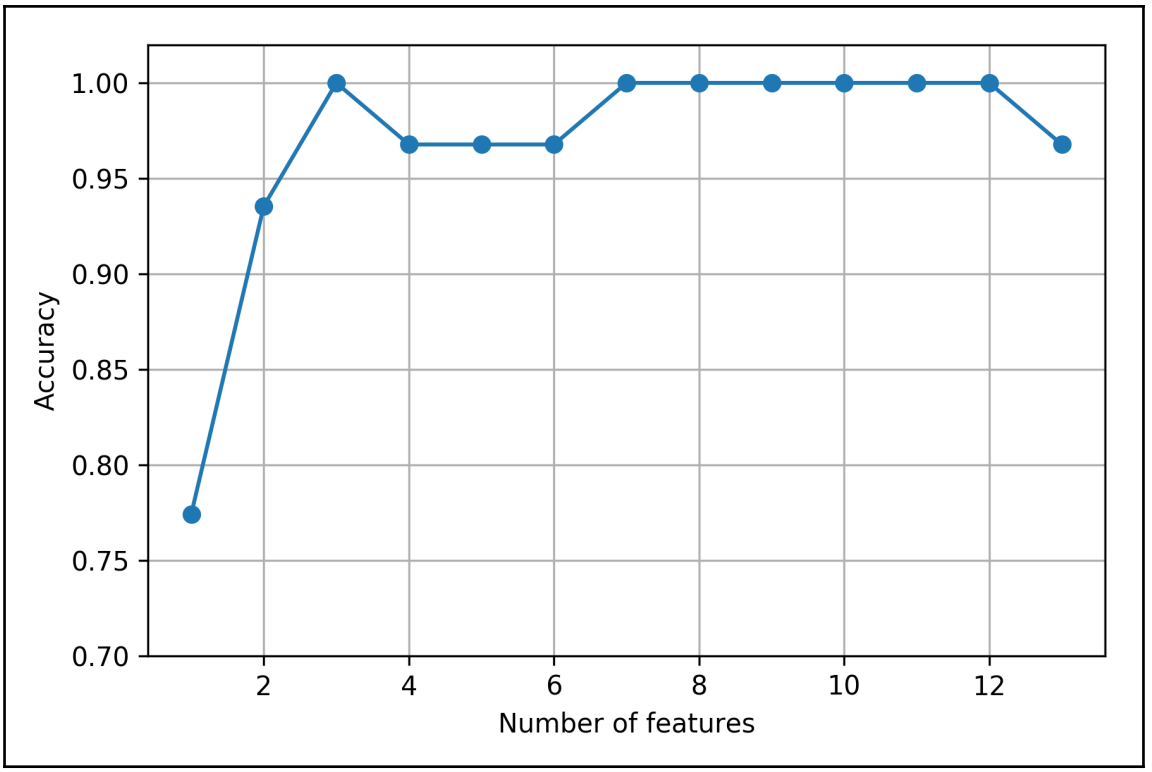
	Class label	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735

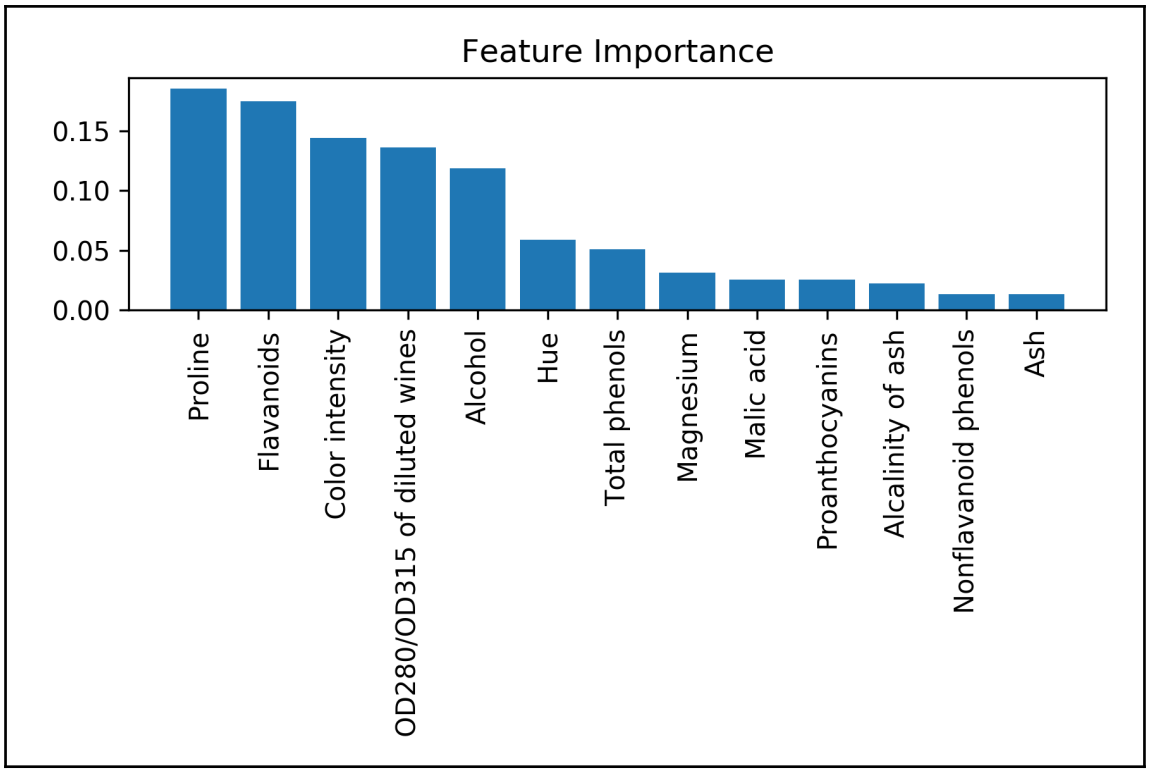

w_2
Minimize cost
w_1


w2
Minimize cost
$\lambda \|\mathbf{w}\|_2^2$
w1
Minimize penalty
Minimize cost + penalty

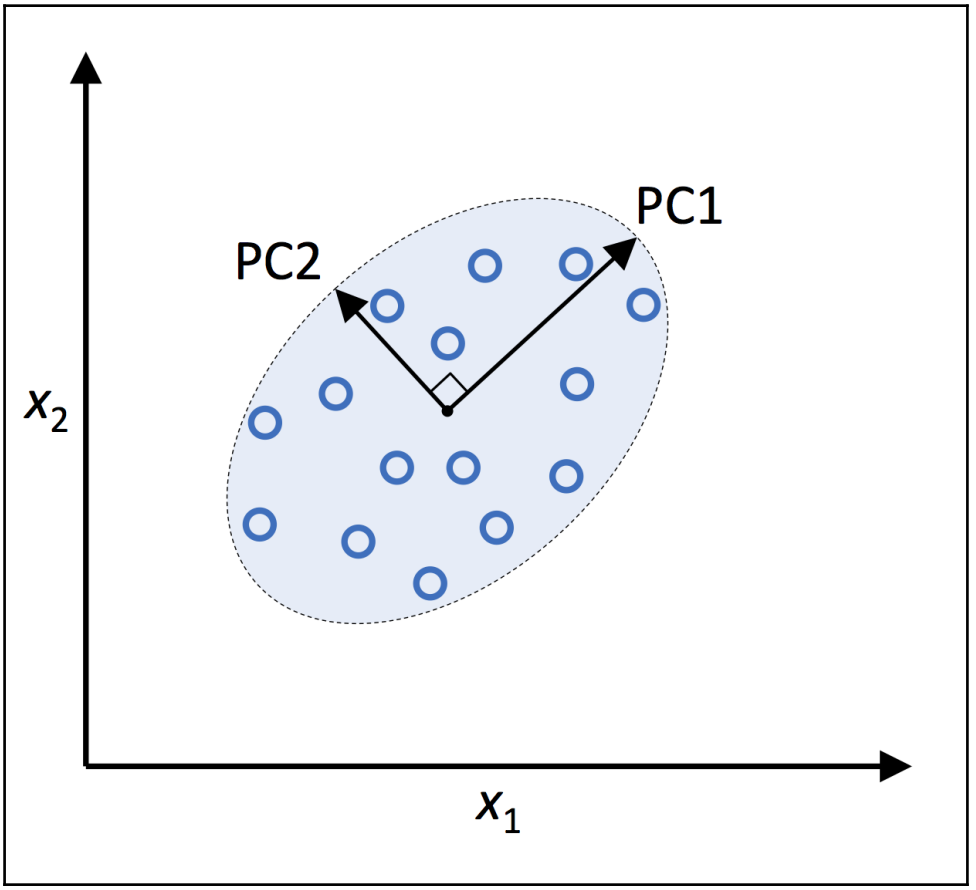

w2
Minimize cost
λ||w||1
w1
Minimize cost + penalty
(w1 = 0)
Minimize penalty

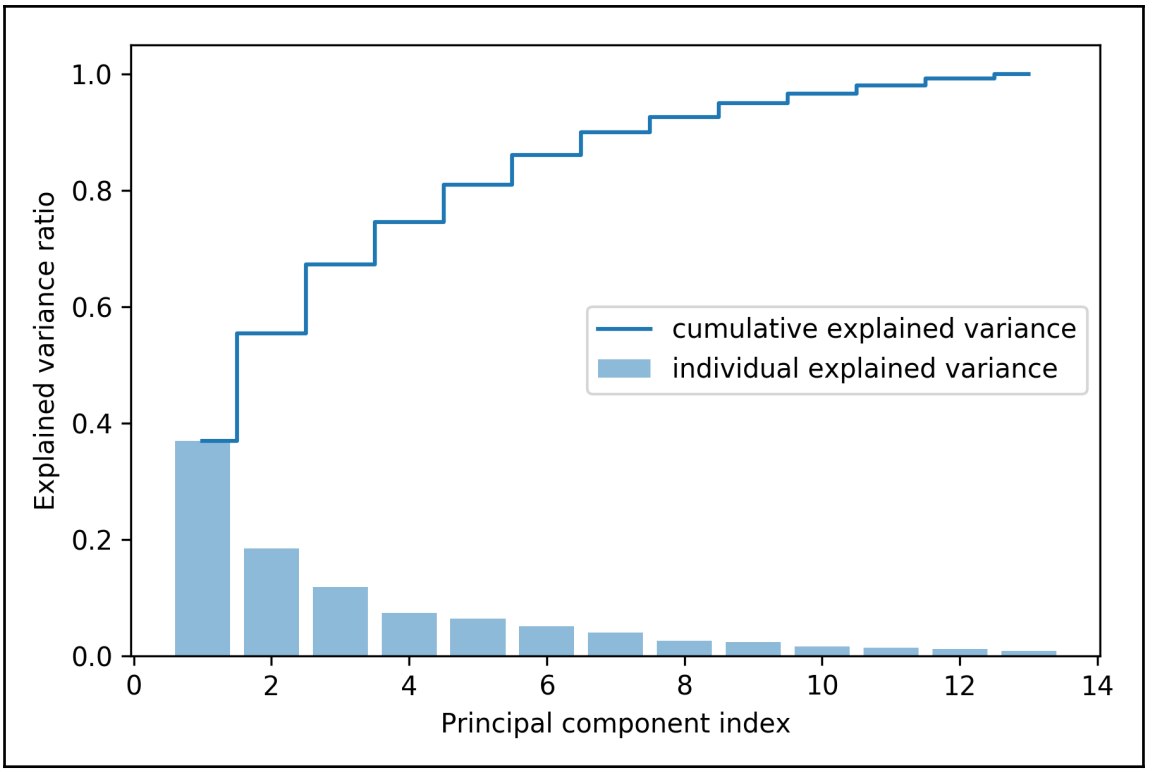

weight coefficient
5
0
−5
−10
−15
−20
10−5
10−3
10−1
101
103
105
C
Alcohol
Malic acid
Ash
Alcalinity of ash
Magnesium
Total phenols
Flavanoids
Nonflavanoid phenols
Proanthocyanins
Color intensity
Hue
OD280/OD315 of diluted wines
Proline

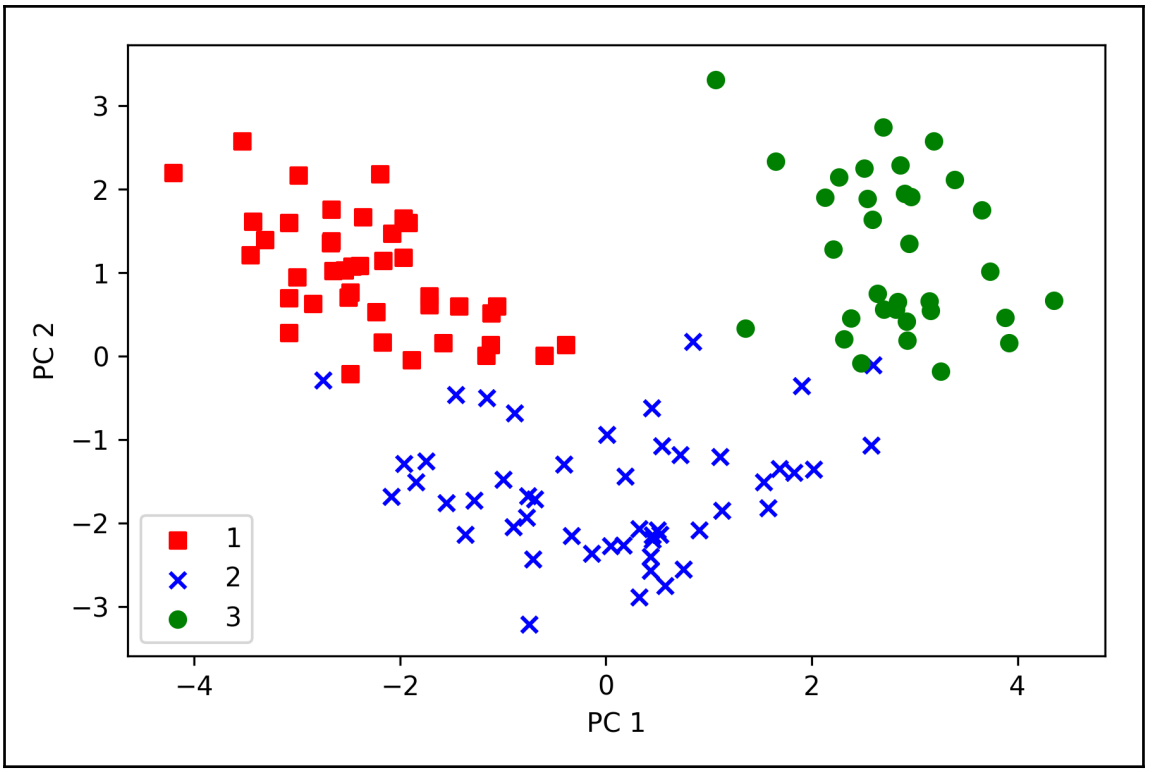

1.00
0.95
0.90
0.85
0.80
0.75
0.70
Accuracy
2
4
6
8
10
12
Number of features

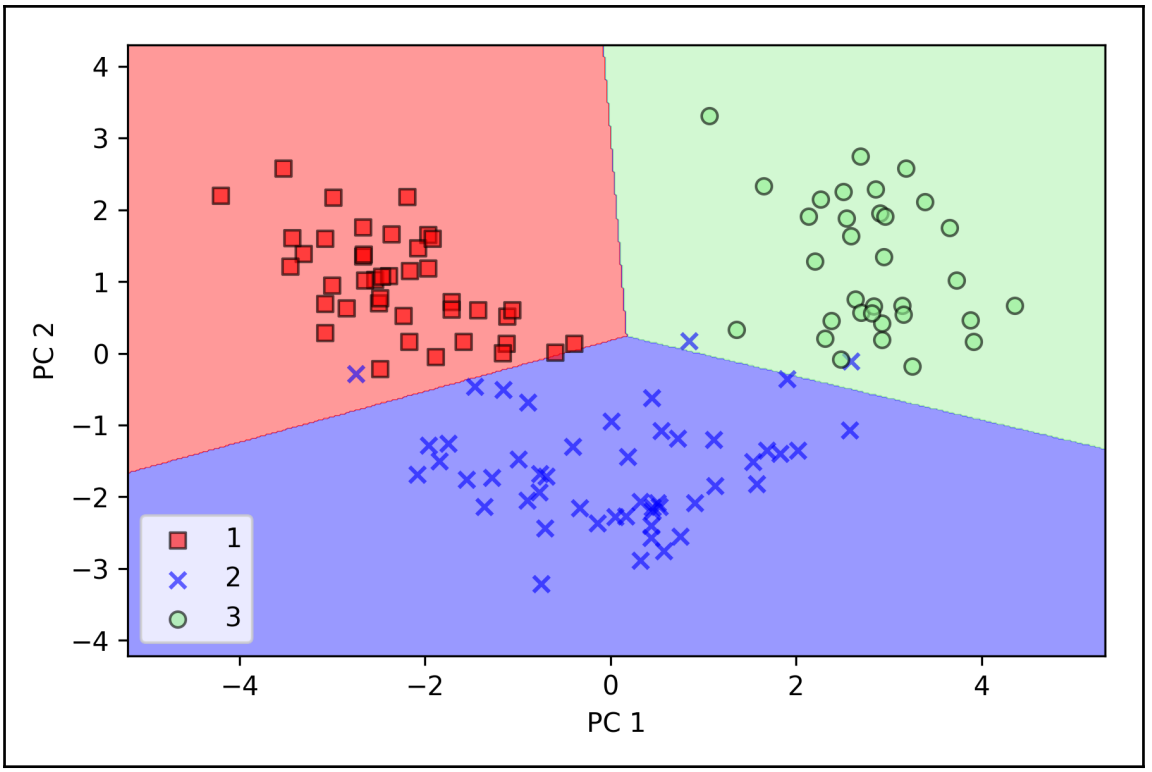

Feature Importance
0.15
0.10
0.05
0.00
Proline
Flavanoids
Color intensity
OD280/OD315 of diluted wines
Alcohol
Hue
Total phenols
Magnesium
Malic acid
Proanthocyanins
Alcalinity of ash
Nonflavanoid phenols
Ash

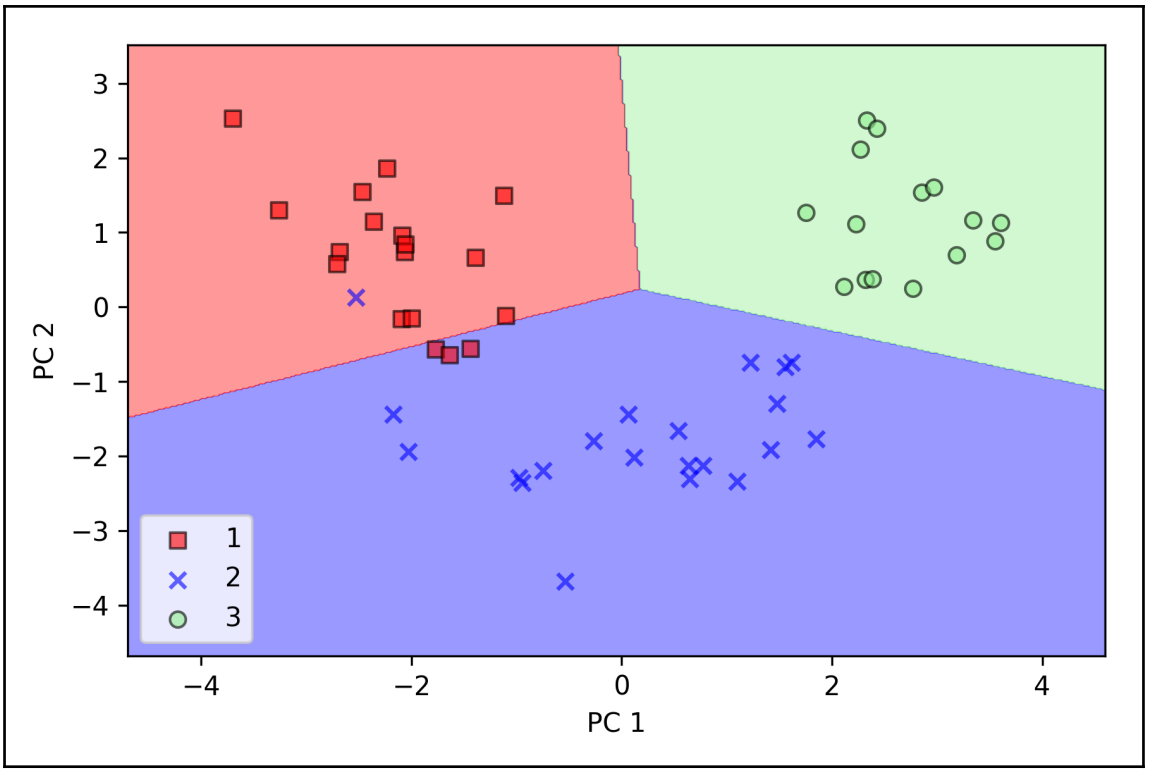
Chapter 5: Compressing Data via Dimensionality Reduction

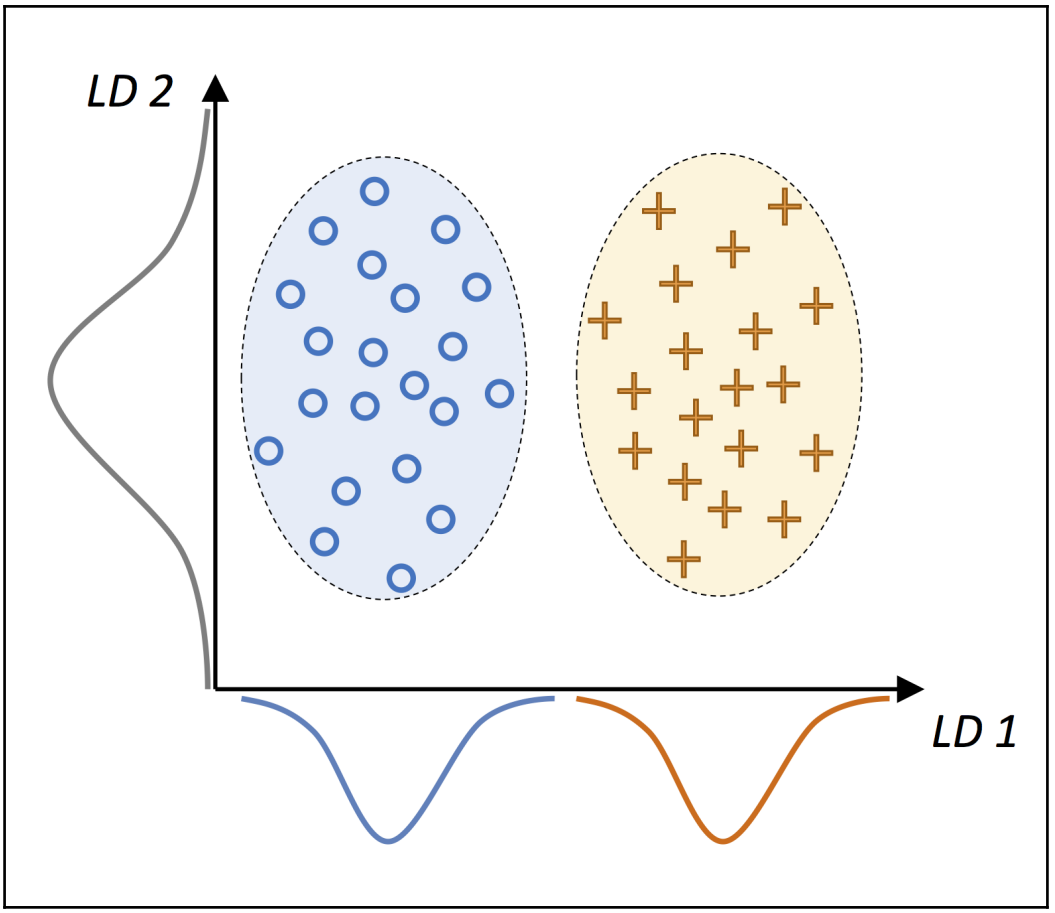


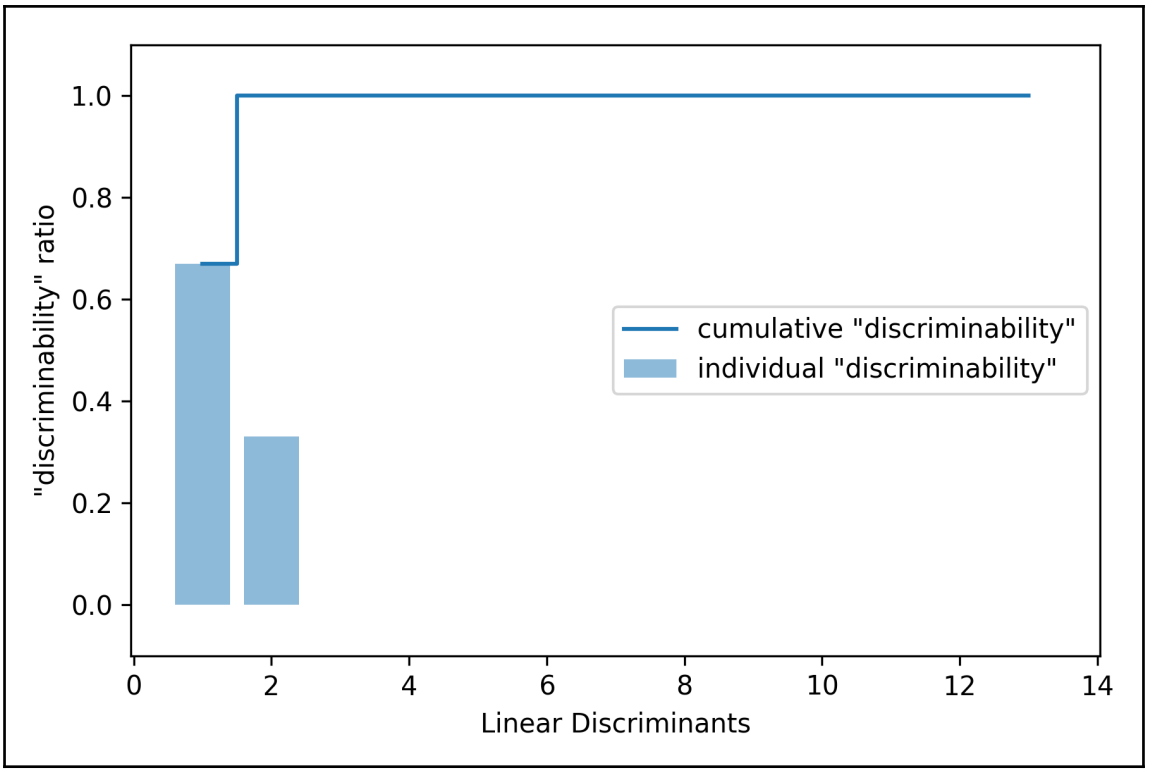

1.0
0.8
0.6
0.4
0.2
0.0
Explained variance ratio
0
2
4
6
8
10
12
14
Principal component index
cumulative explained variance
individual explained variance

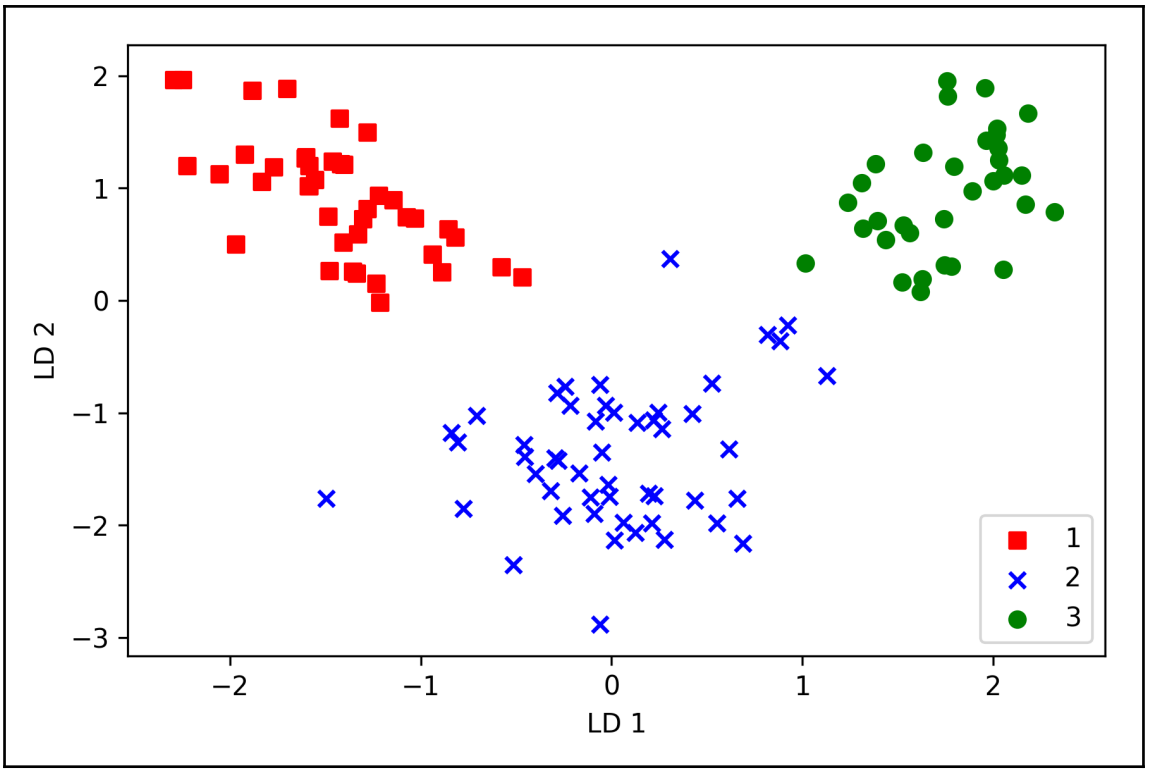

PC 2
PC 1
3
2
1
0
−1
−2
−3
−4
−2
0
2
4
1
2
3

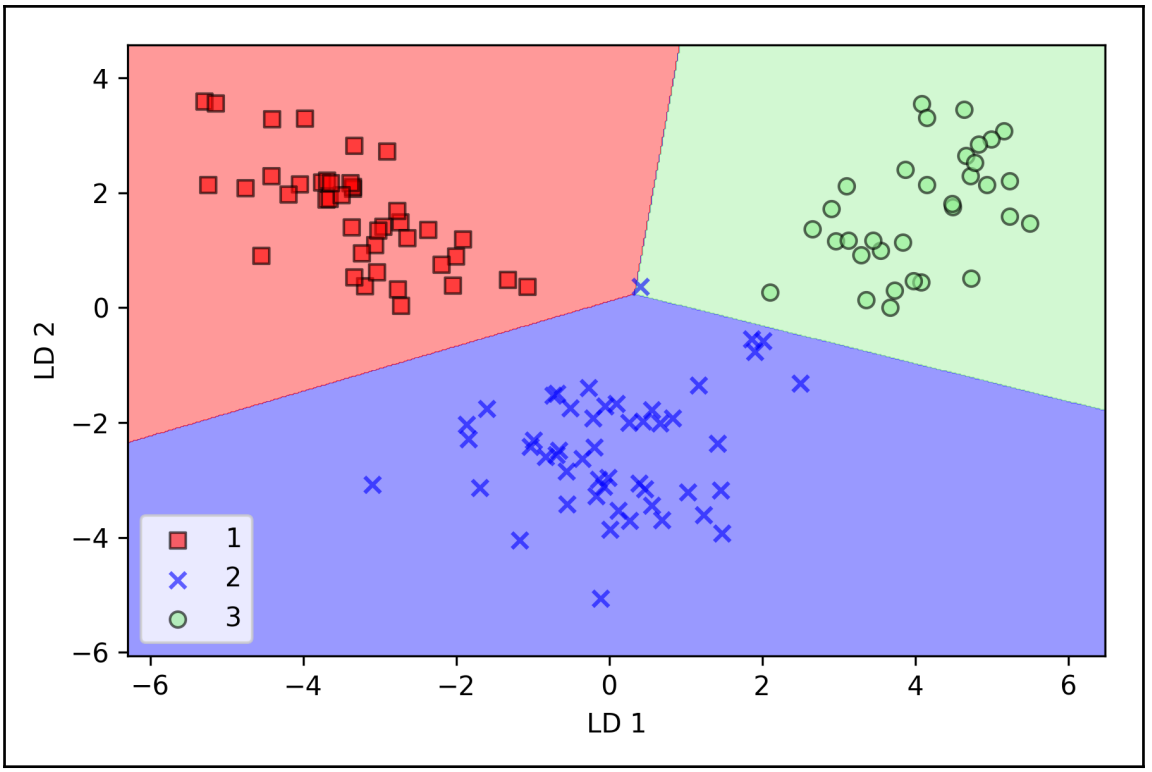

4
3
2
1
0
−1
−2
−3
−4
PC 2
−4
−2
0
2
4
PC 1
1
2
3

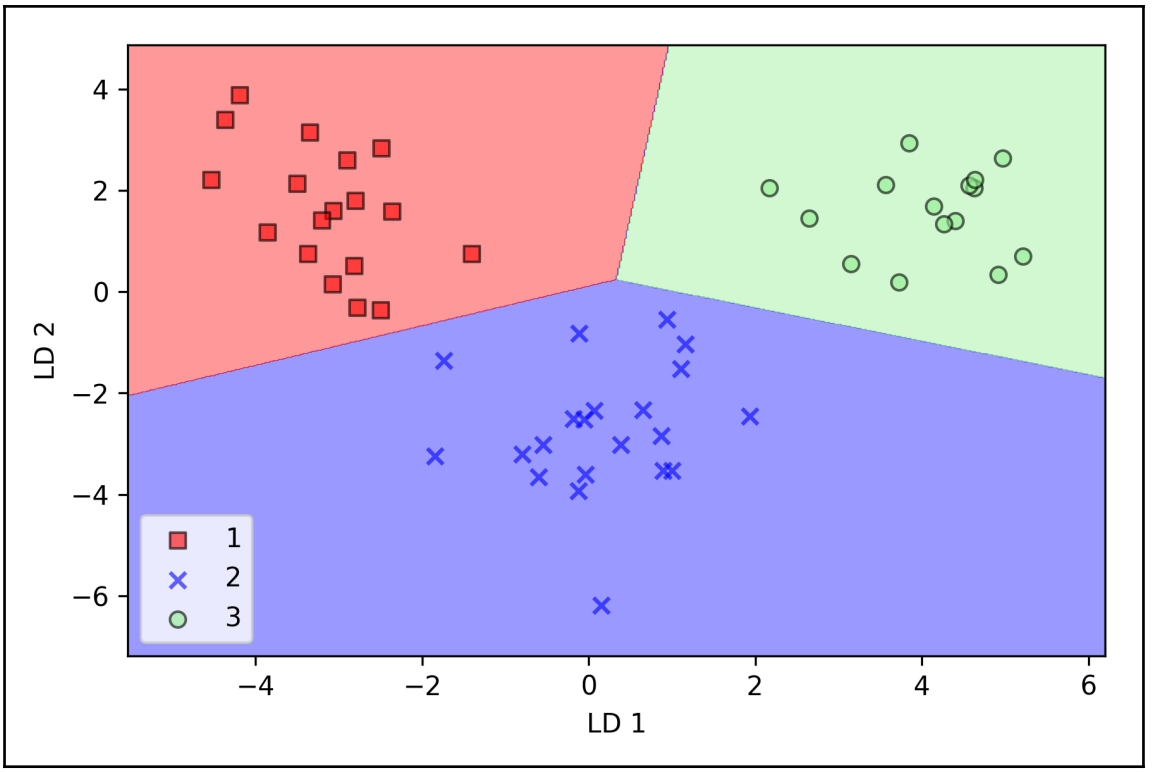

3
2
1
0
−1
−2
−3
−4
PC 2
−4
−2
0
2
4
PC 1
1
2
3

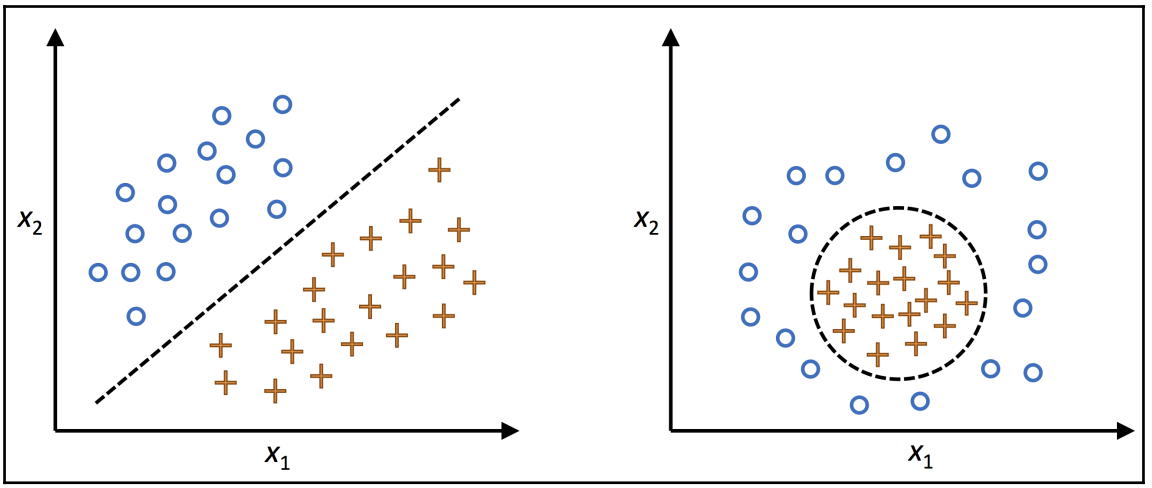

LD 2
LD 1

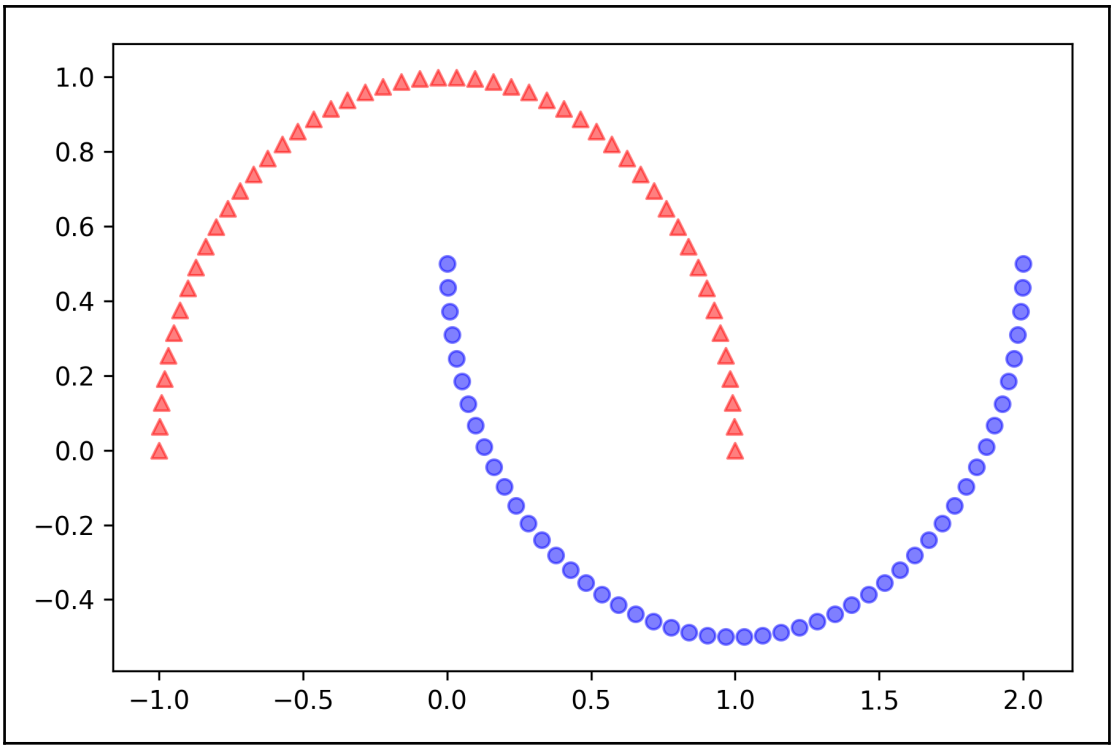

1.0
0.8
0.6
0.4
0.2
0.0
"discriminability" ratio
cumulative "discriminability"
individual "discriminability"
0
2
4
6
8
10
12
14
Linear Discriminants

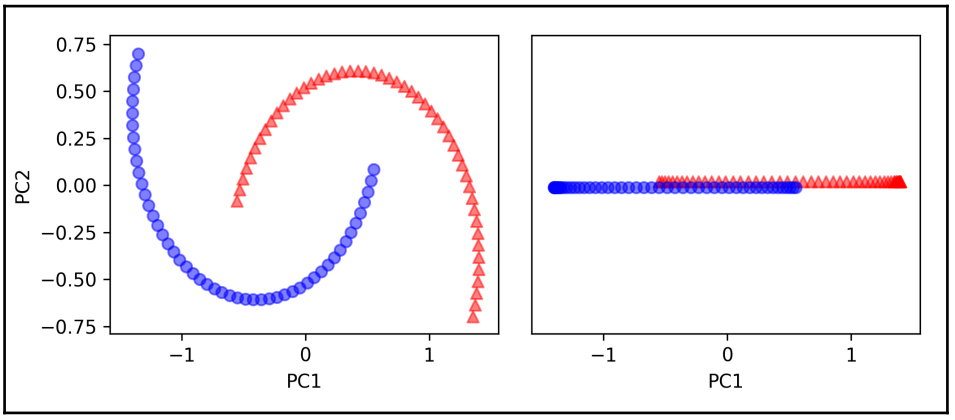

2
1
0
−1
−2
−3
LD 2
−2
−1
0
1
2
LD 1
1
2
3

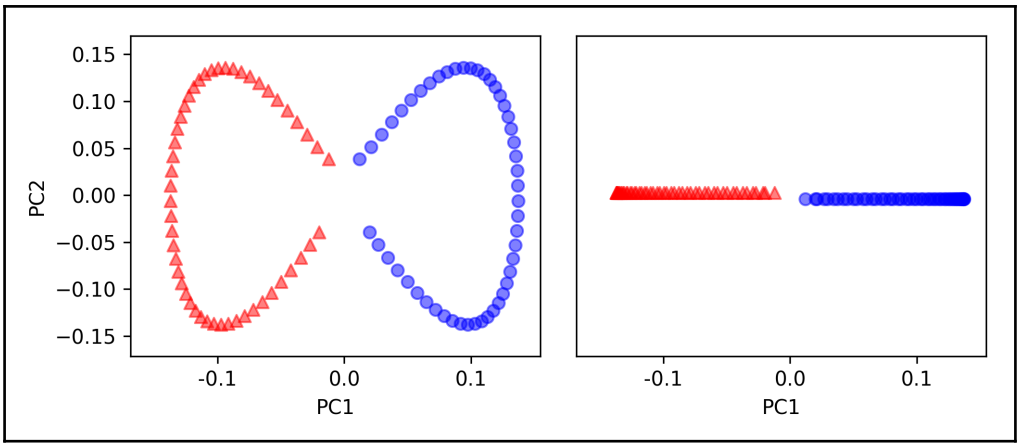

4
2
0
−2
−4
−6
LD 2
−6
−4
−2
0
2
4
6
LD 1
1
2
3

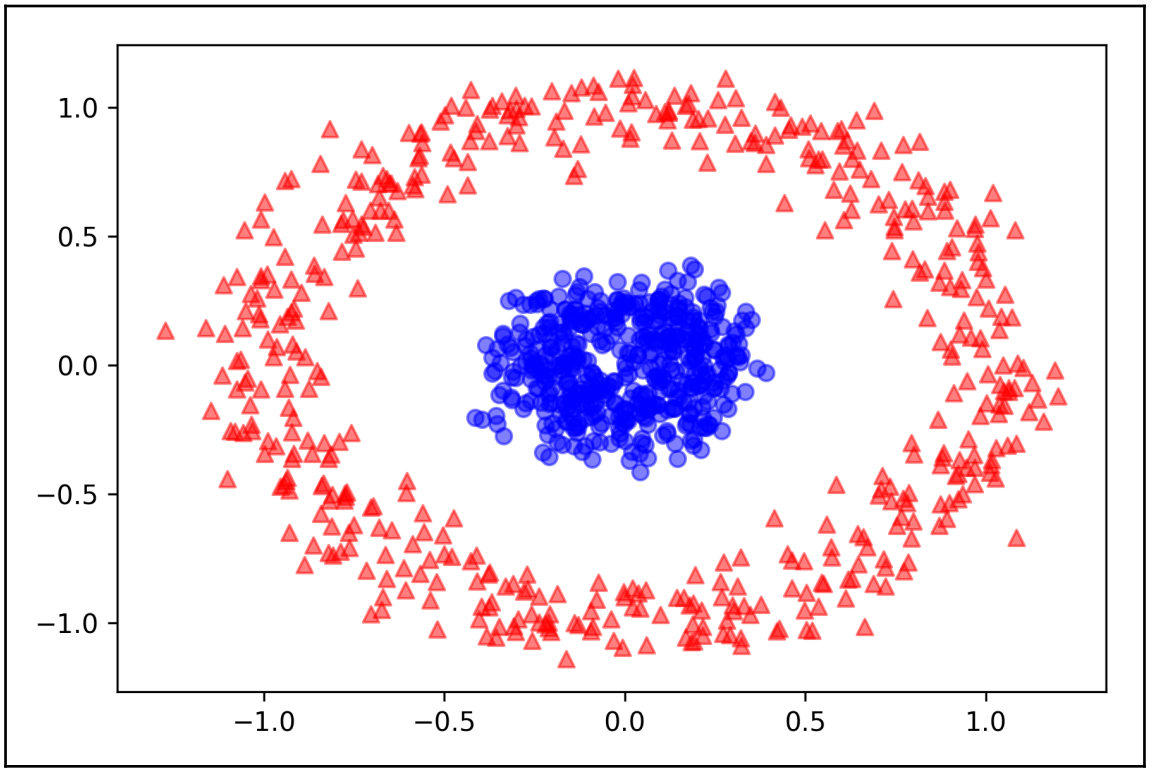

LD 2
LD 1
1
2
3

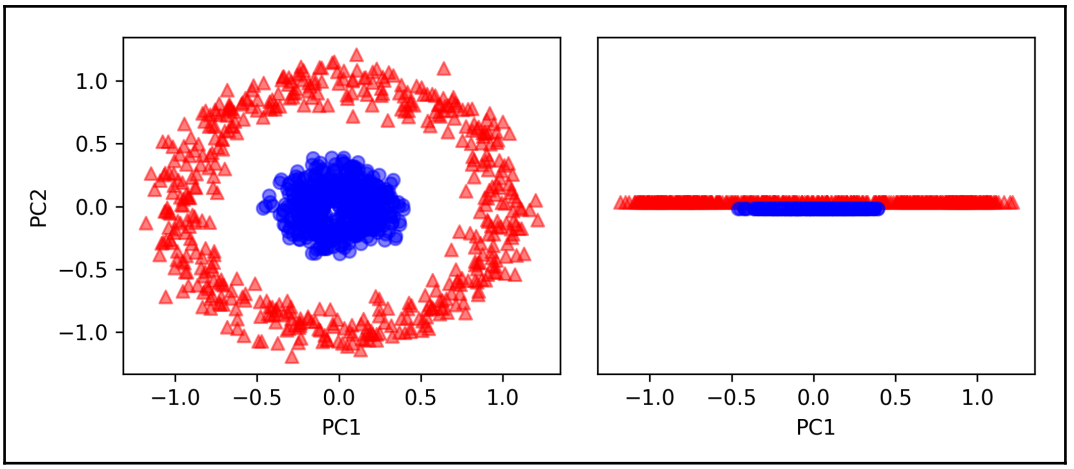

x_2
x_1
x_2
x_1

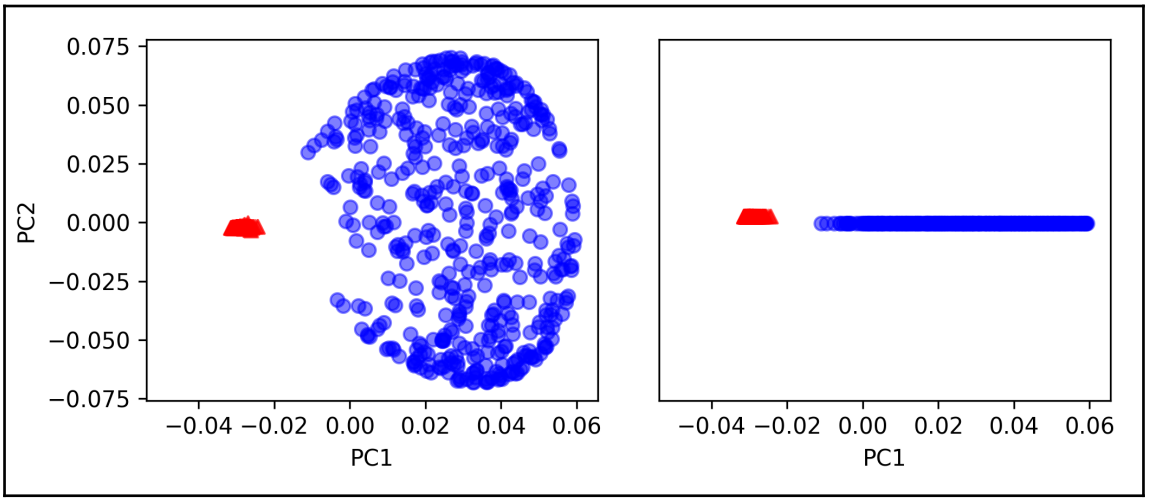

1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

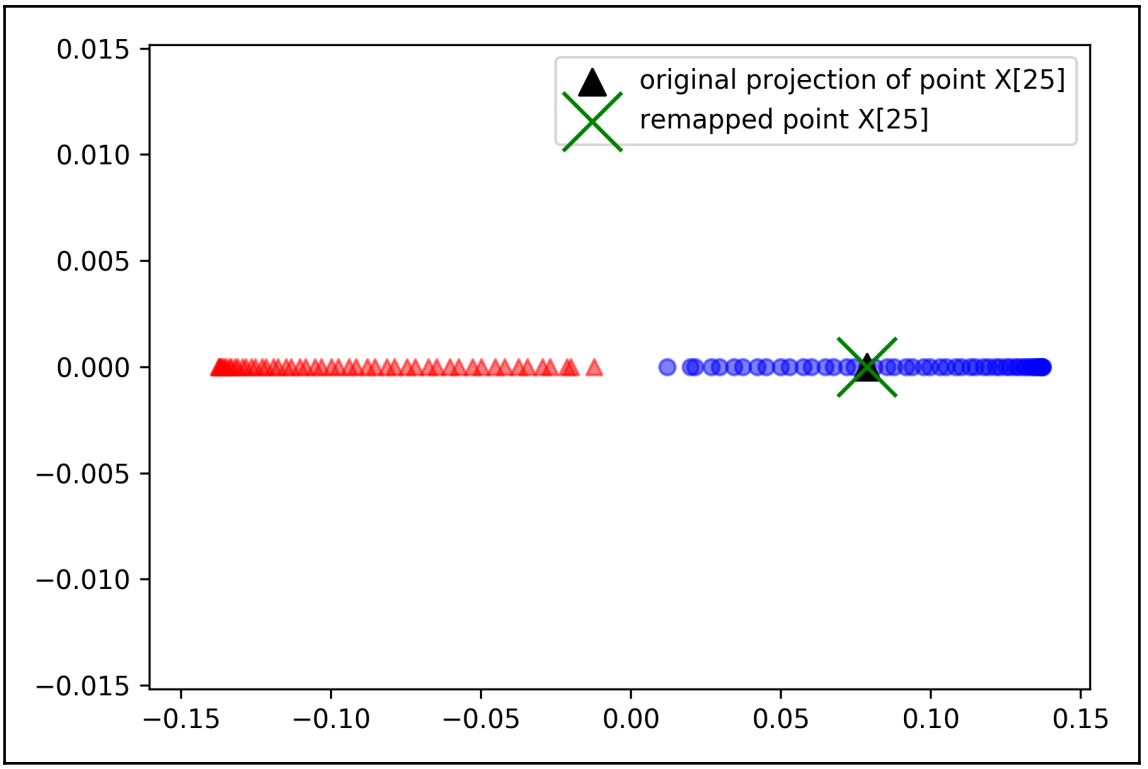

PC2
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1
0
1
PC1
−1
0
1
PC1

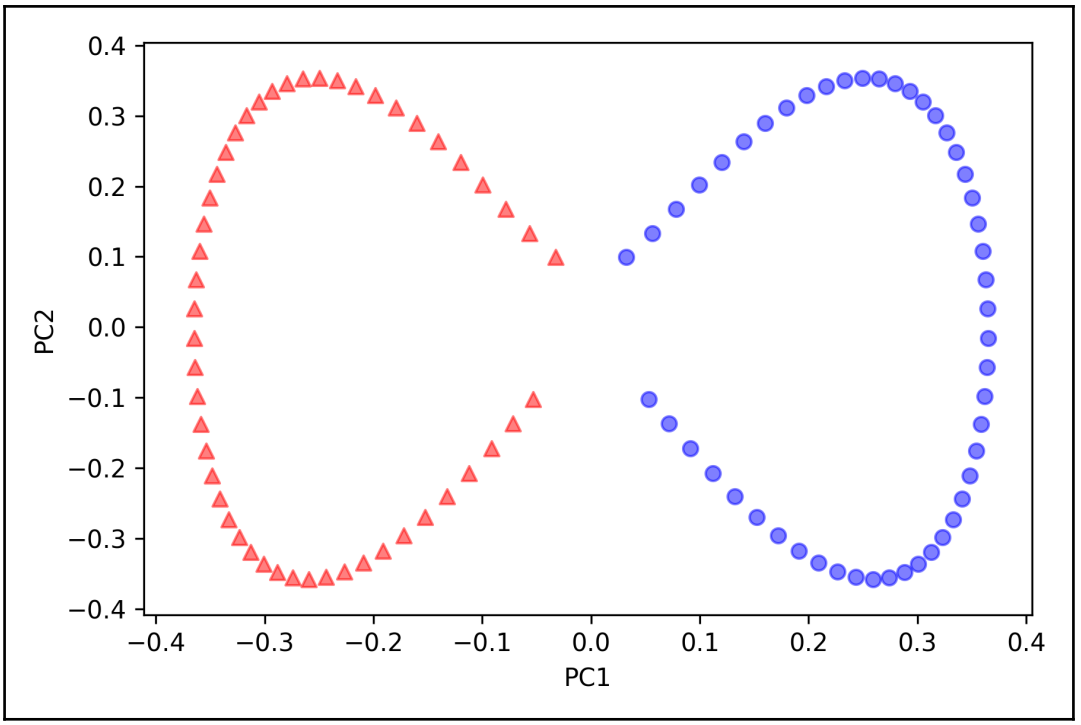

0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
PC2
-0.1
0.0
0.1
PC1
-0.1
0.0
0.1
PC1


1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

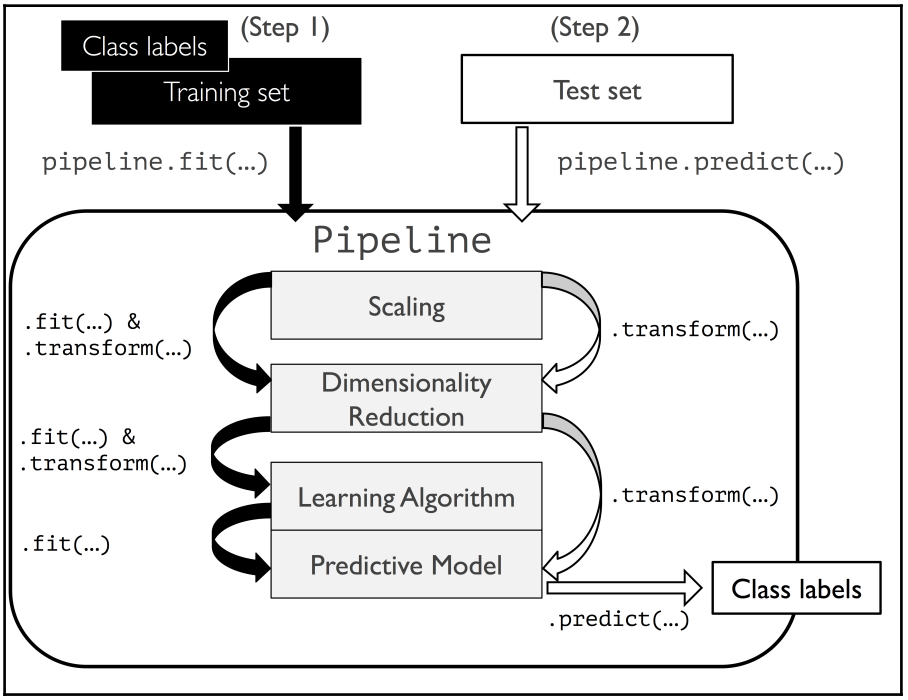

PC2
1.0
0.5
0.0
−0.5
−1.0
−1.0 −0.5 0.0 0.5 1.0
PC1
−1.0 −0.5 0.0 0.5 1.0
PC1

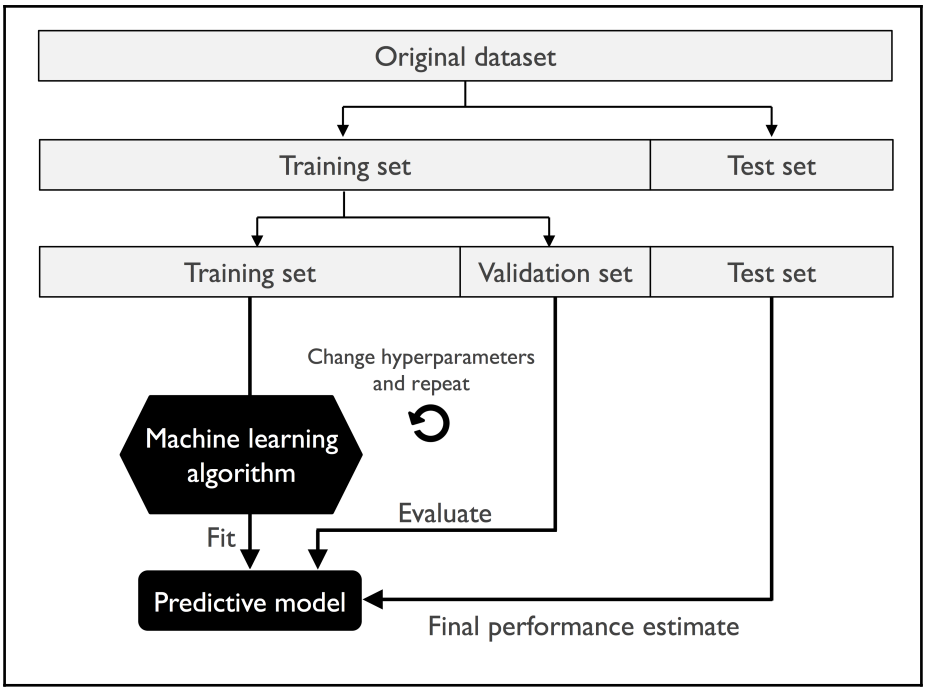

PC2
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.04 −0.02 0.00 0.02 0.04 0.06
PC1
−0.04 −0.02 0.00 0.02 0.04 0.06
PC1

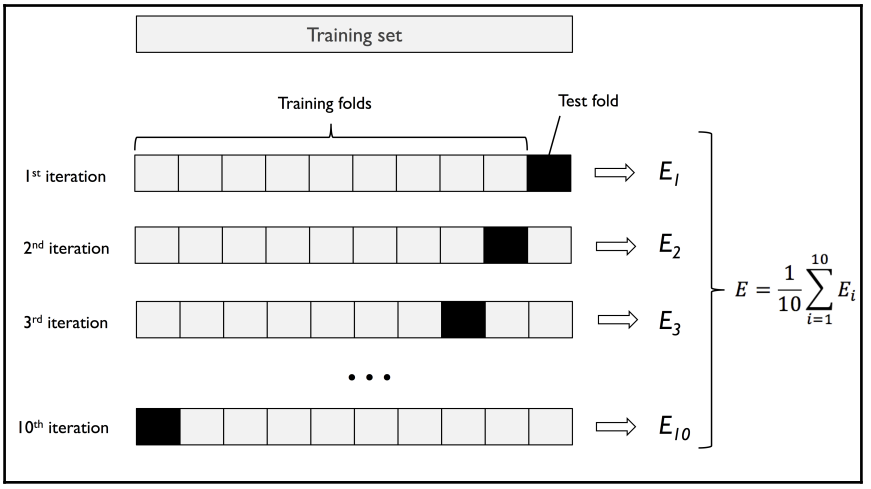

original projection of point X[25]
remapped point X[25]
0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15

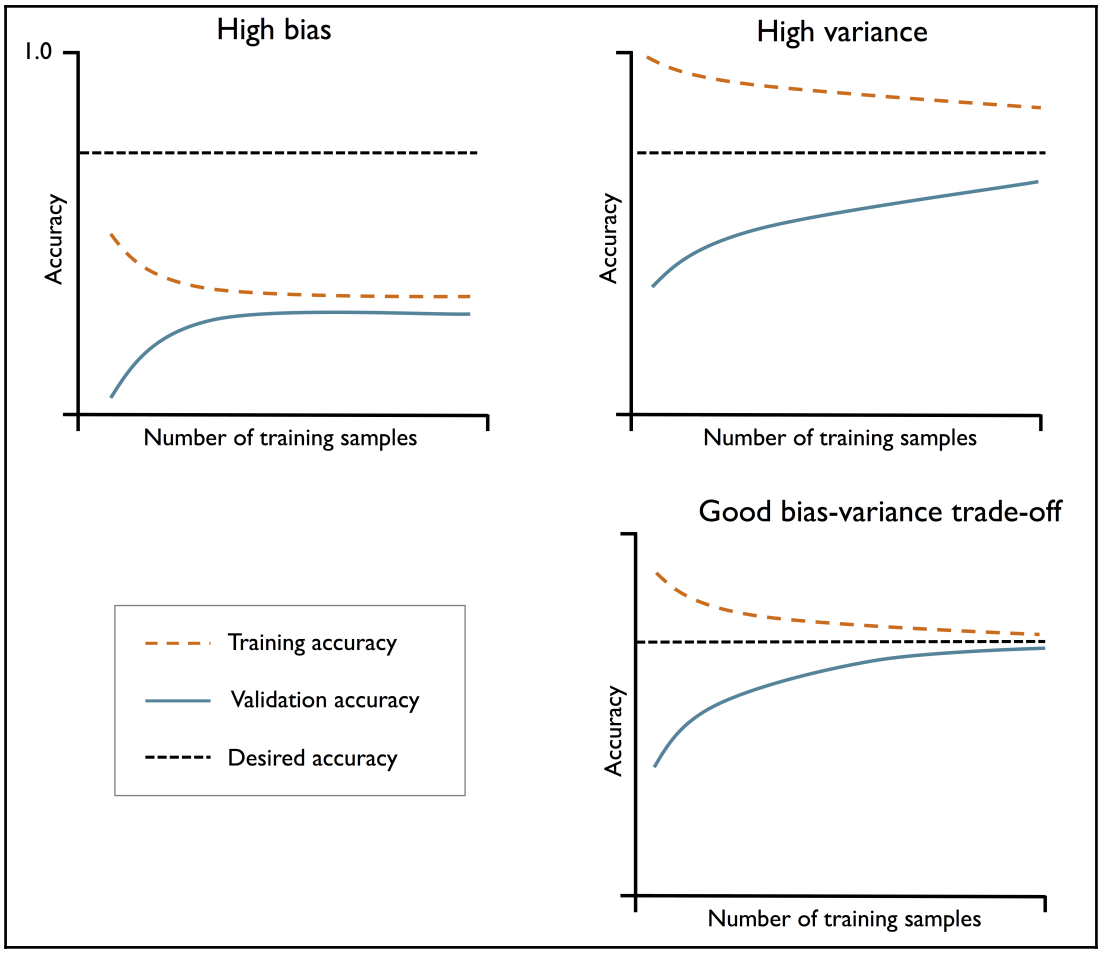

0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
PC2
−0.4
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
PC1

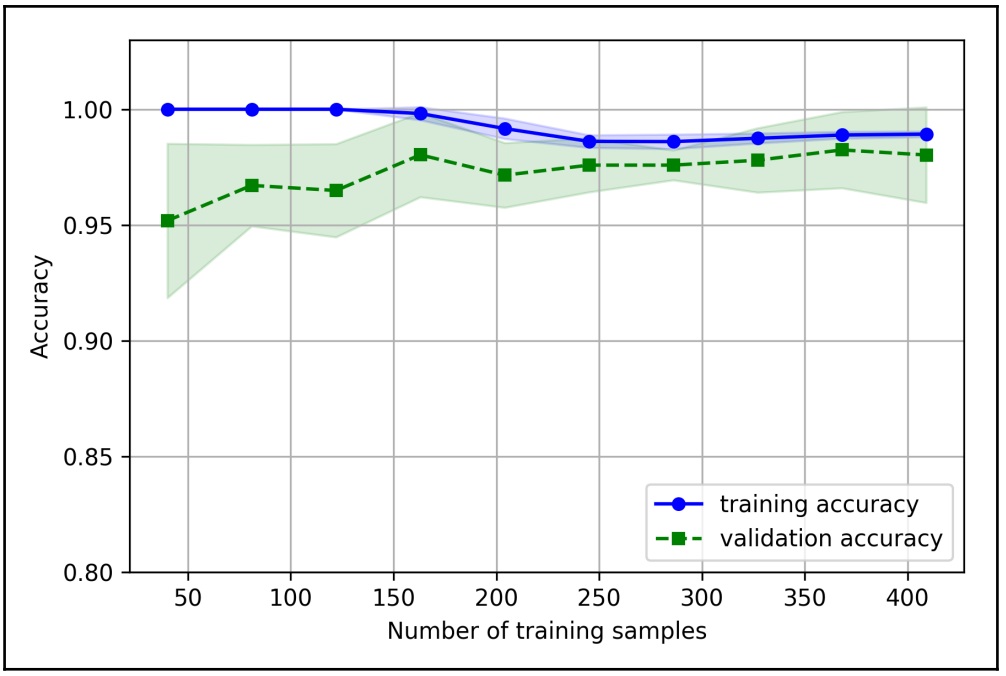
Chapter 6: Learning Best Practices for Model Evaluation and Hyperparameter Tuning

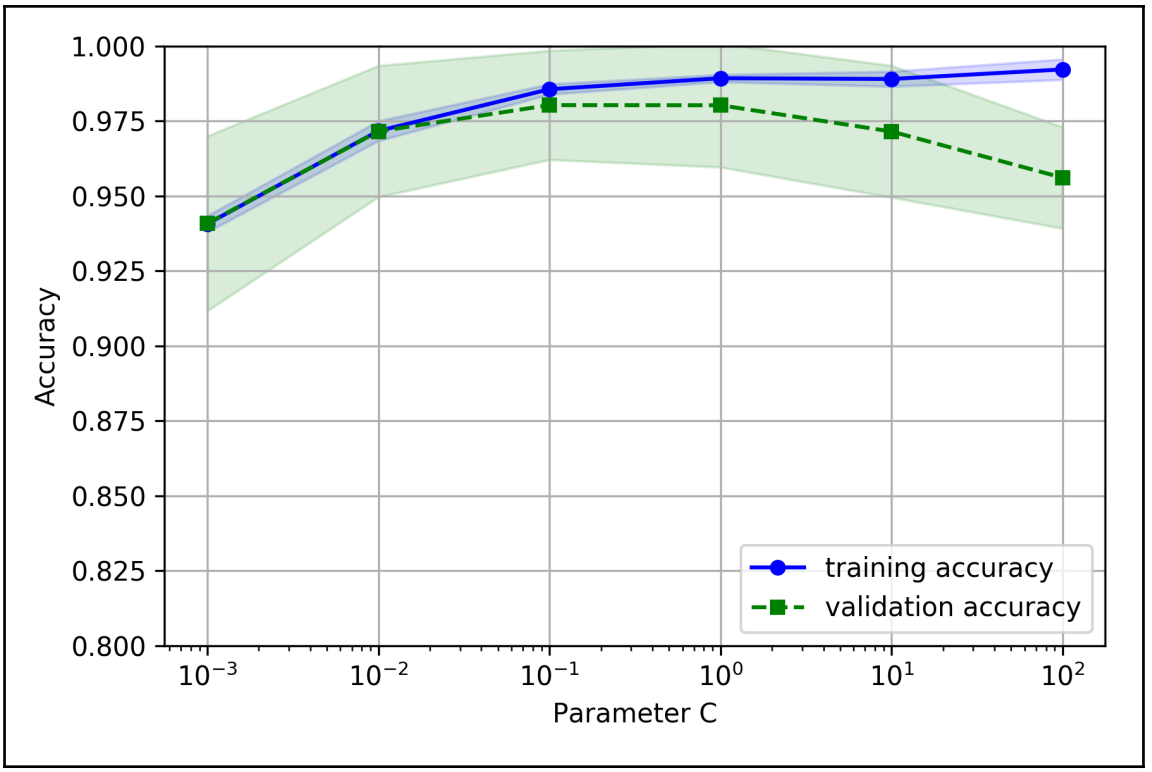


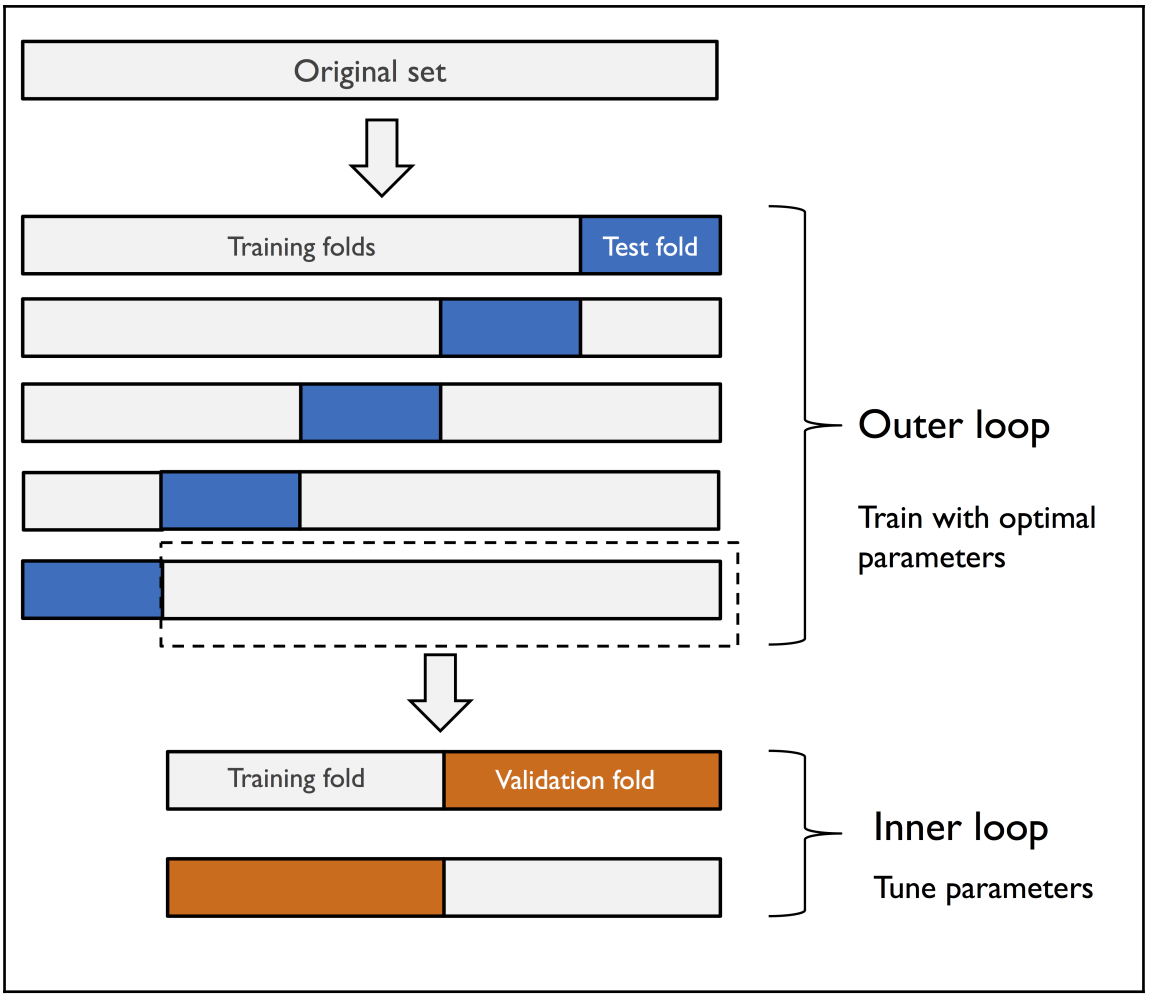

Original dataset
Training set
Test set
Training set
Validation set
Test set
Change hyperparameters and repeat
Machine learning algorithm
Evaluate
Fit
Predictive model
Final performance estimate

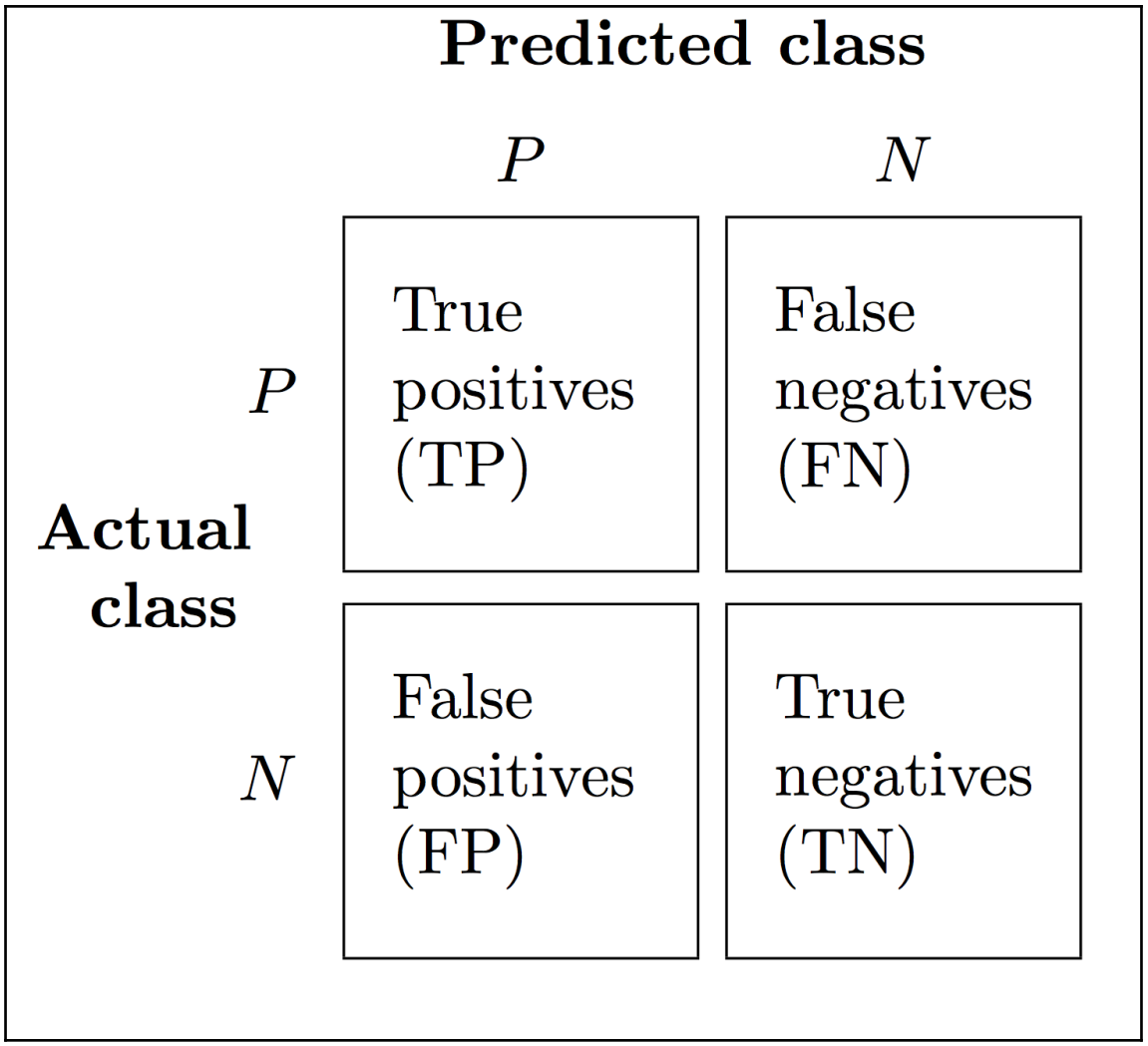

Training set
Training folds
Test fold
1st iteration
2nd iteration
3rd iteration
10th iteration
E_1
E_2
E_3
E_{10}
$E = \frac{1}{10}\sum_{i=1}^{10} E_i$

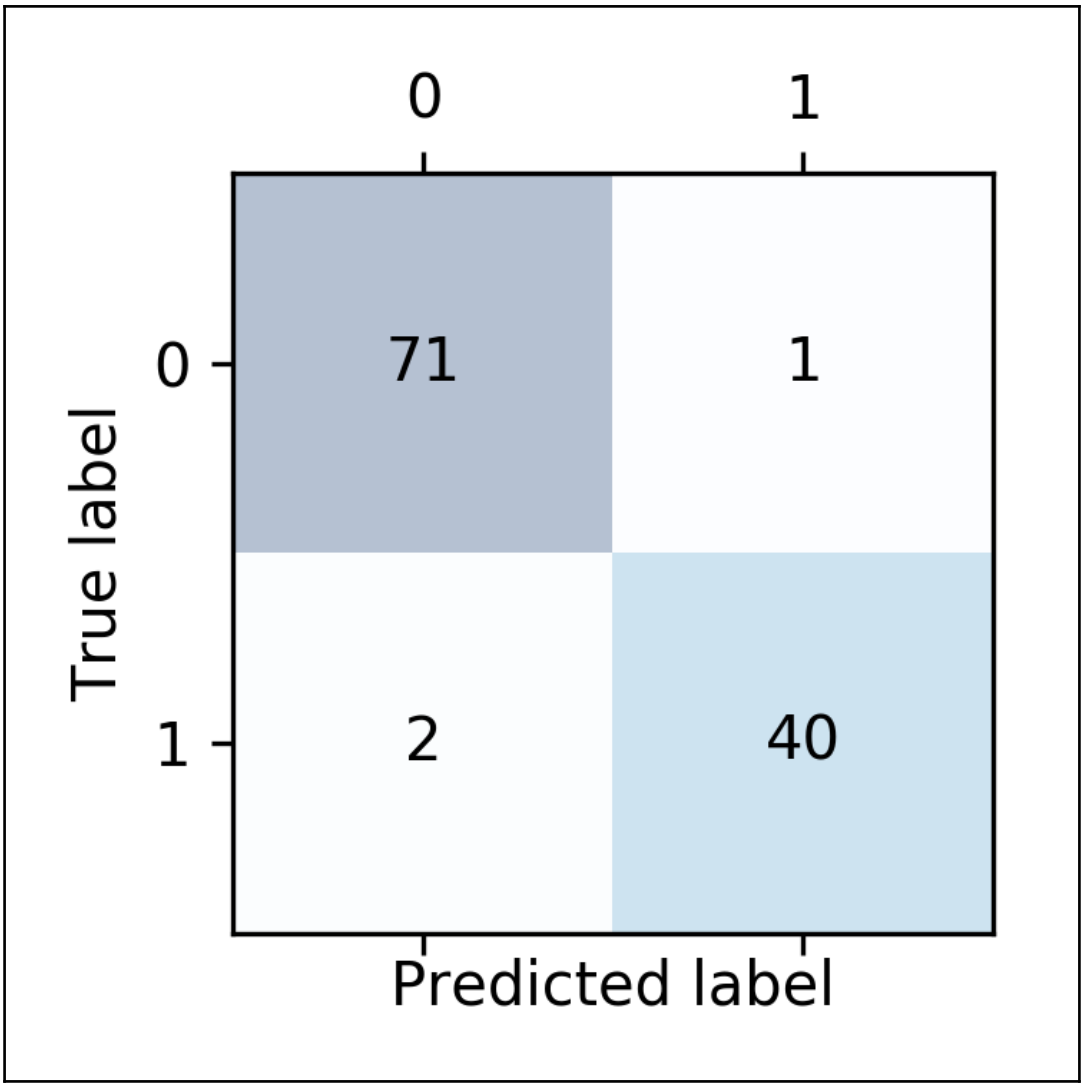

High bias
1.0
Accuracy
Number of training samples
High variance
Accuracy
Number of training samples
Good bias-variance trade-off
Accuracy
Number of training samples
Training accuracy
Validation accuracy
Desired accuracy

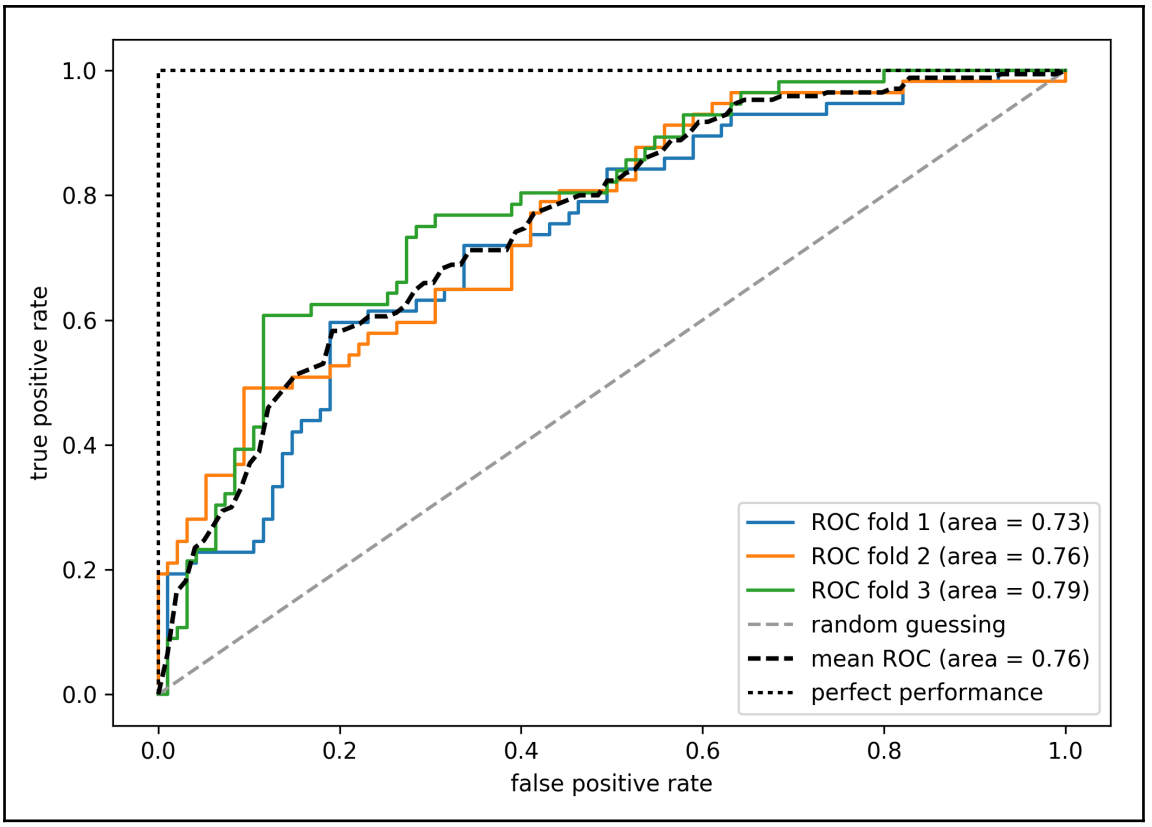

1.00
0.95
0.90
0.85
0.80
Accuracy
50
100
150
200
250
300
350
400
Number of training samples
training accuracy
validation accuracy


1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
Accuracy
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Parameter C
training accuracy
validation accuracy

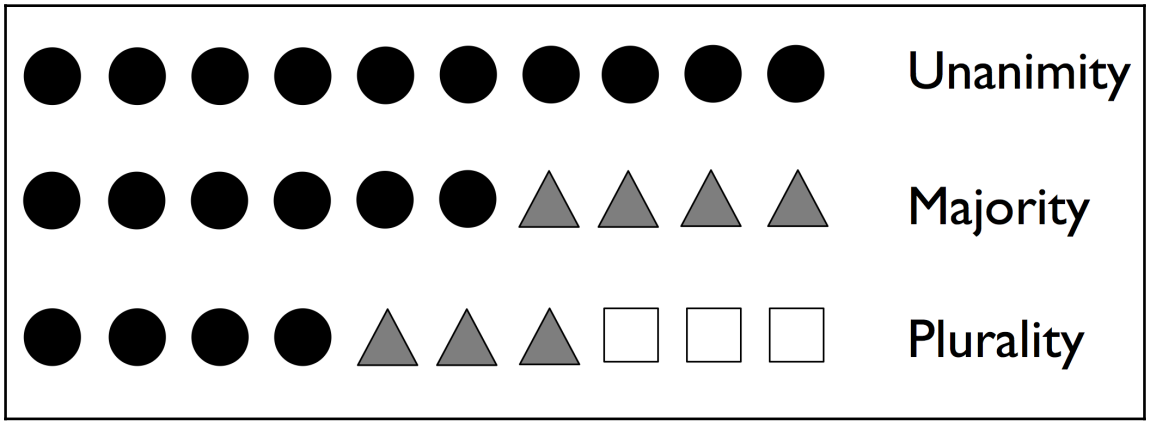

Original set
Training folds
Test fold
Outer loop
Train with optimal parameters
Training fold
Validation fold
Inner loop
Tune parameters

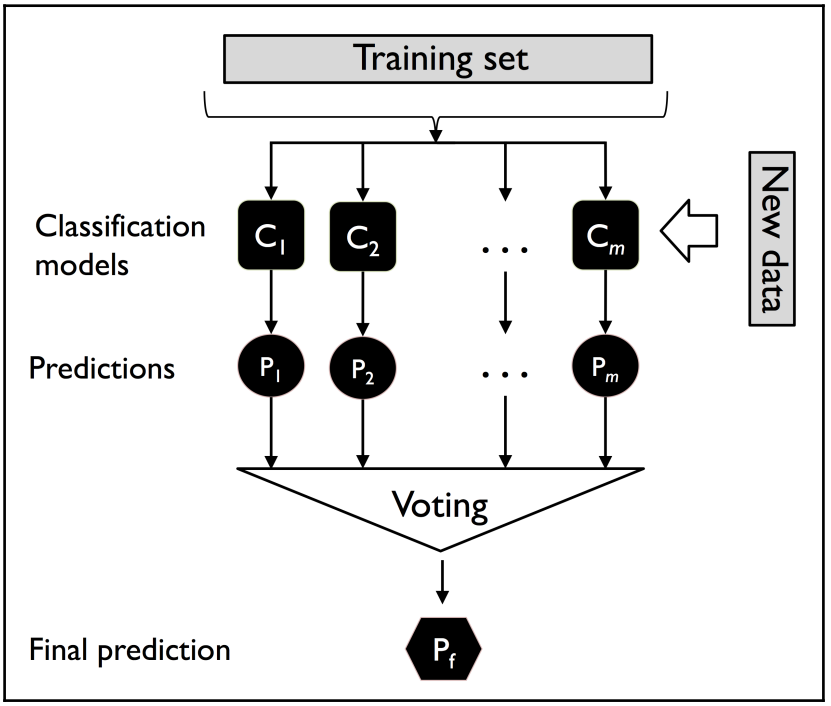

Predicted class
P
N
Actual class
P
N
True positives (TP)
False negatives (FN)
False positives (FP)
True negatives (TN)

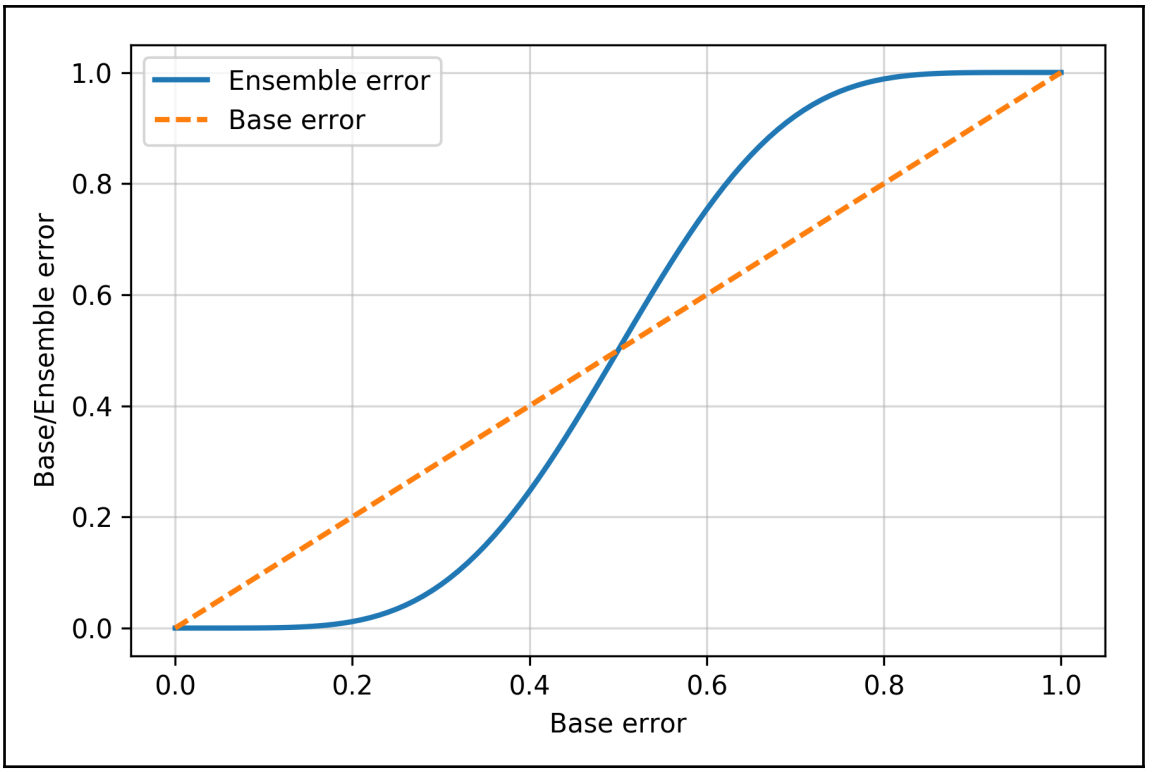

0
1
0
1
71
1
2
40
True label
Predicted label

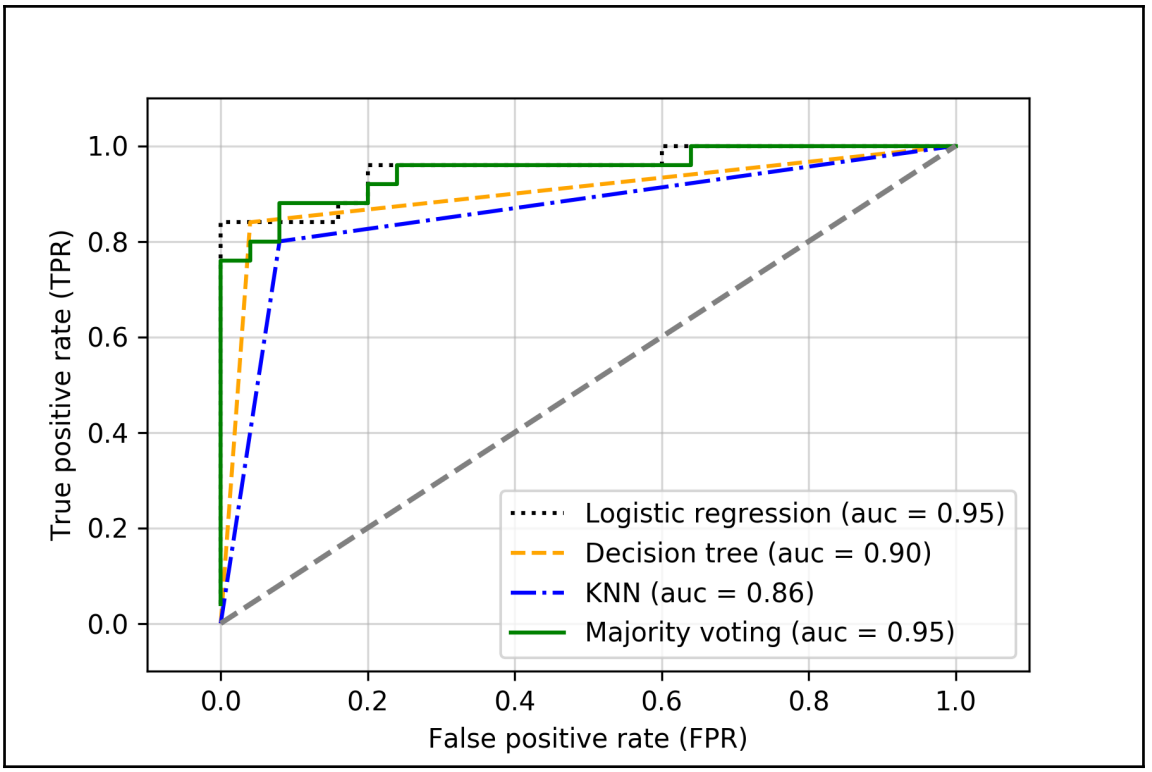

true positive rate
false positive rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC fold 1 (area = 0.73)
ROC fold 2 (area = 0.76)
ROC fold 3 (area = 0.79)
random guessing
mean ROC (area = 0.76)
perfect performance

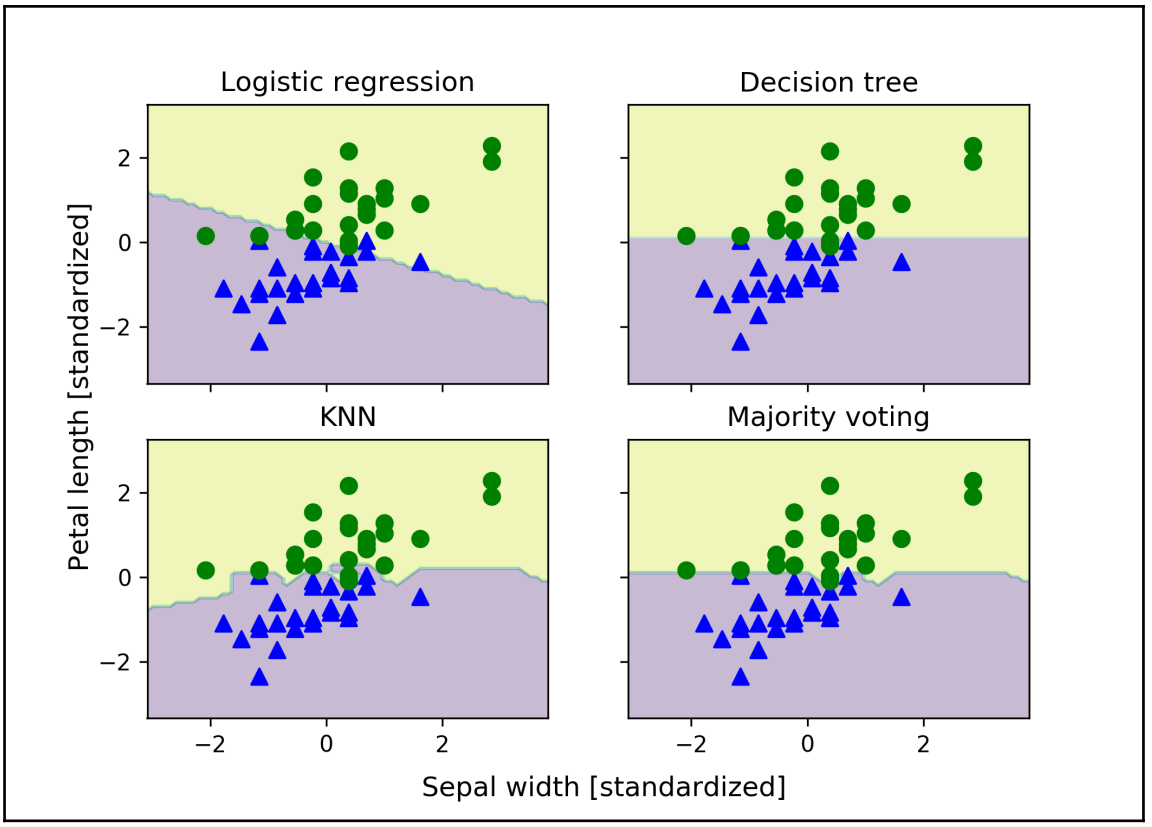
Chapter 7: Combining Different Models for Ensemble Learning

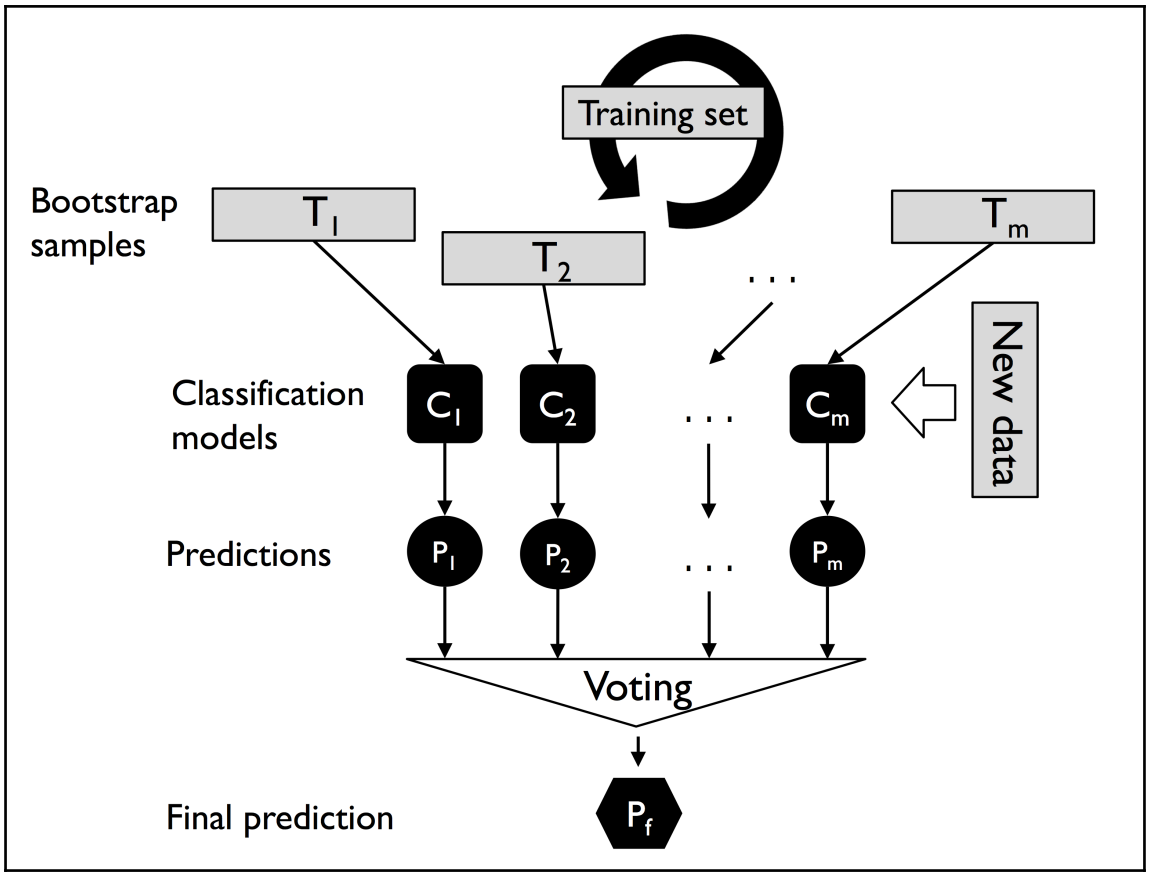




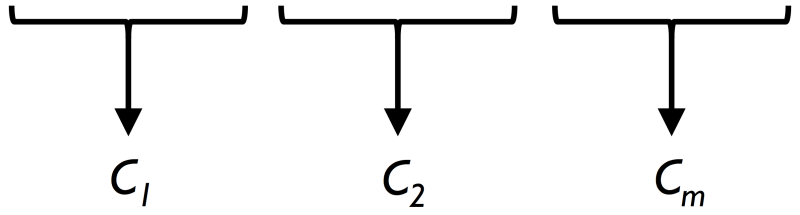

Ensemble error
Base error
Base/Ensemble error
Base error
0.0
0.2
0.4
0.6
0.8
1.0

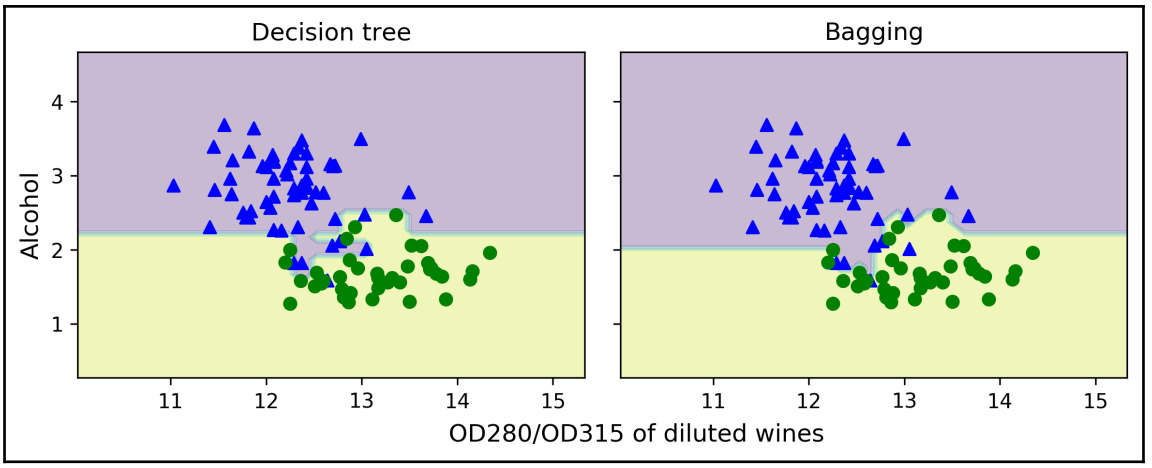

True positive rate (TPR)
False positive rate (FPR)
0.0
0.2
0.4
0.6
0.8
1.0
Logistic regression (auc = 0.95)
Decision tree (auc = 0.90)
KNN (auc = 0.86)
Majority voting (auc = 0.95)

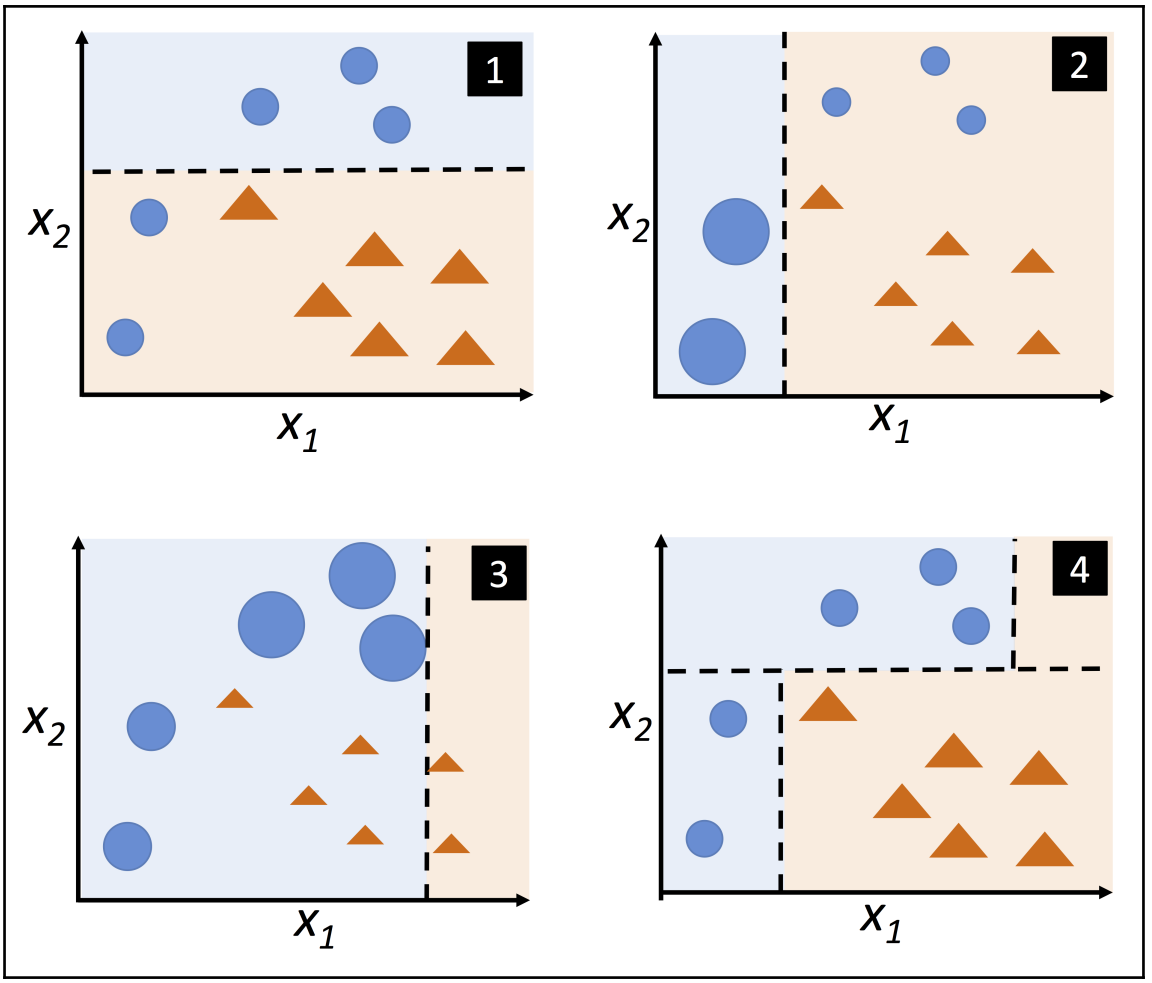

Logistic regression
Decision tree
KNN
Majority voting
Petal length [standardized]
Sepal width [standardized]
2
0
−2
−2
0
2


Training set
Bootstrap samples
T_1
T_2
T_m
Classification models
C_1
C_2
C_m
New data
Predictions
P_1
P_2
P_m
Voting
Final prediction
P_f

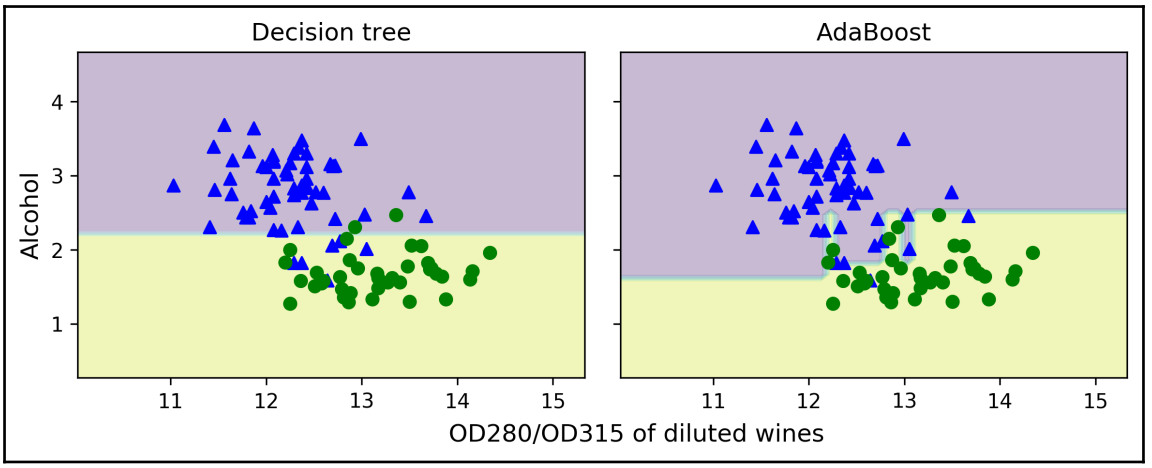
Sample indices	Bagging round 1	Bagging round 2	…
1	2	7	…
2	2	3	…
3	1	2	…
4	3	1	…
5	7	1	…
6	2	7	…
7	4	7	…




Decision tree
Bagging
Alcohol
4
3
2
1
11
12
13
14
15
OD280/OD315 of diluted wines


1
x_2
x_1
2
x_2
x_1
3
x_2
x_1
4
x_2
x_1

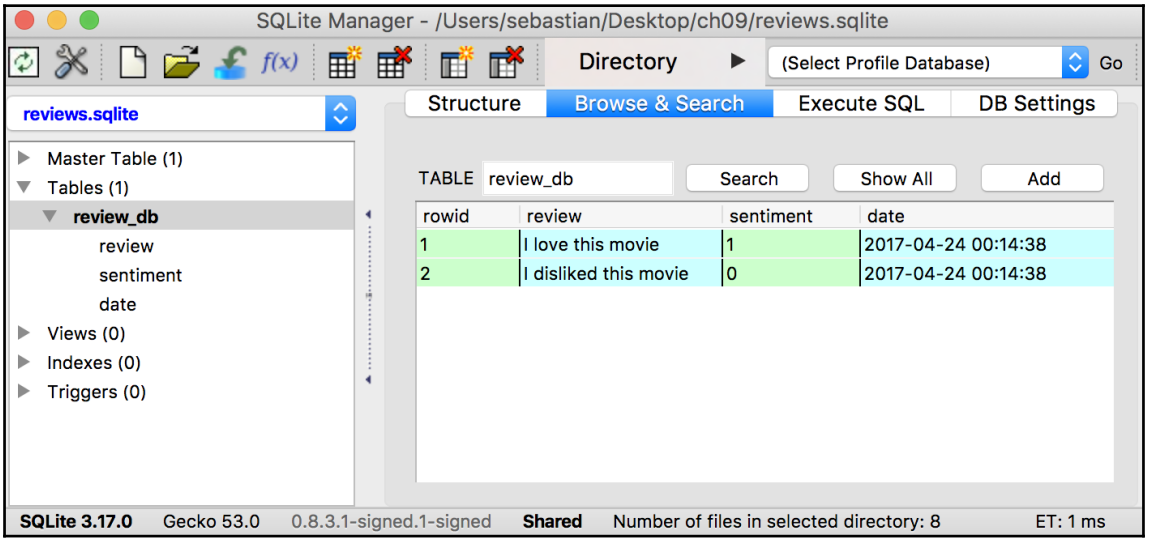
Sample indices	x	y	Weights	$\hat{y}(x <= 3.0)$?	Correct?	Updated weights
1	1.0	1	0.1	1	Yes	0.072
2	2.0	1	0.1	1	Yes	0.072
3	3.0	1	0.1	1	Yes	0.072
4	4.0	-1	0.1	-1	Yes	0.072
5	5.0	-1	0.1	-1	Yes	0.072
6	6.0	-1	0.1	-1	Yes	0.072
7	7.0	1	0.1	-1	No	0.167
8	8.0	1	0.1	-1	No	0.167
9	9.0	1	0.1	-1	No	0.167
10	10.0	-1	0.1	-1	Yes	0.072

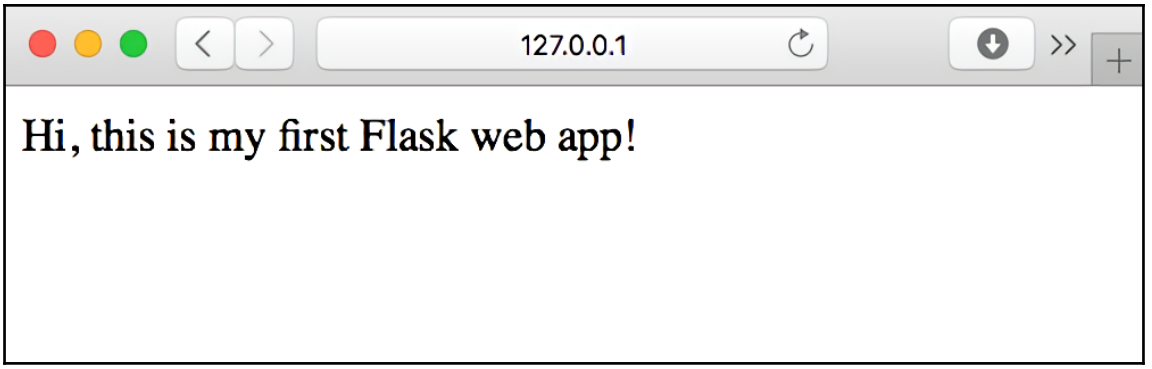


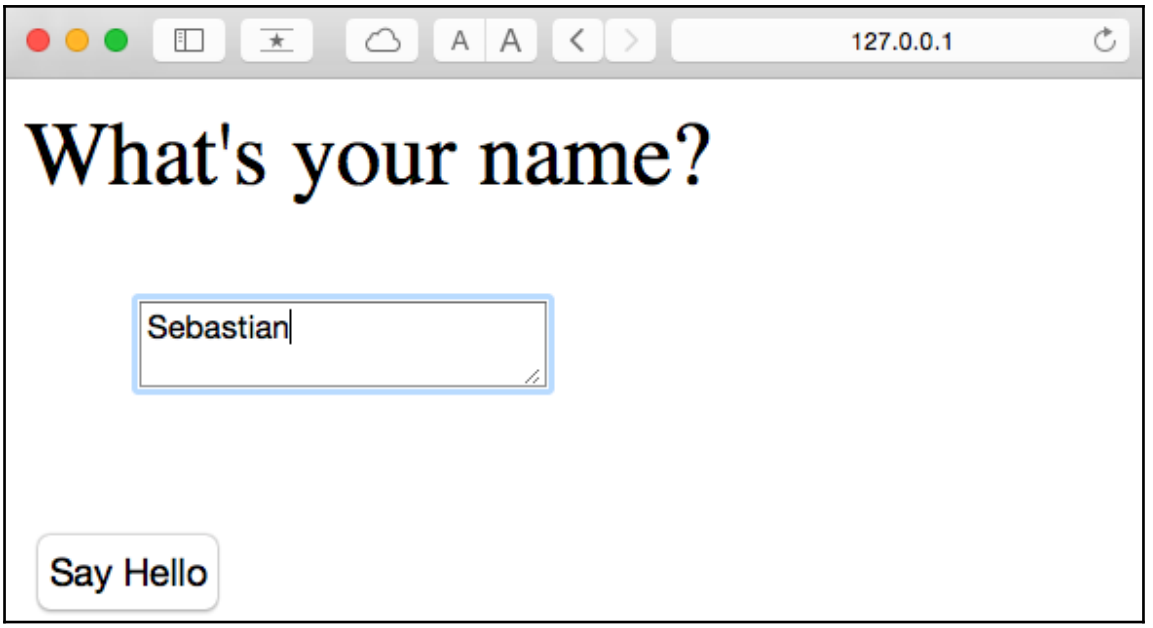
Chapter 8: Applying Machine Learning to Sentiment Analysis

	review	sentiment
0	In 1974, the teenager Martha Moxley (Maggie Gr...	1
1	OK... so... I really like Kris Kristofferson a...	0
2	***SPOILER*** Do not read this, if you think a...	0

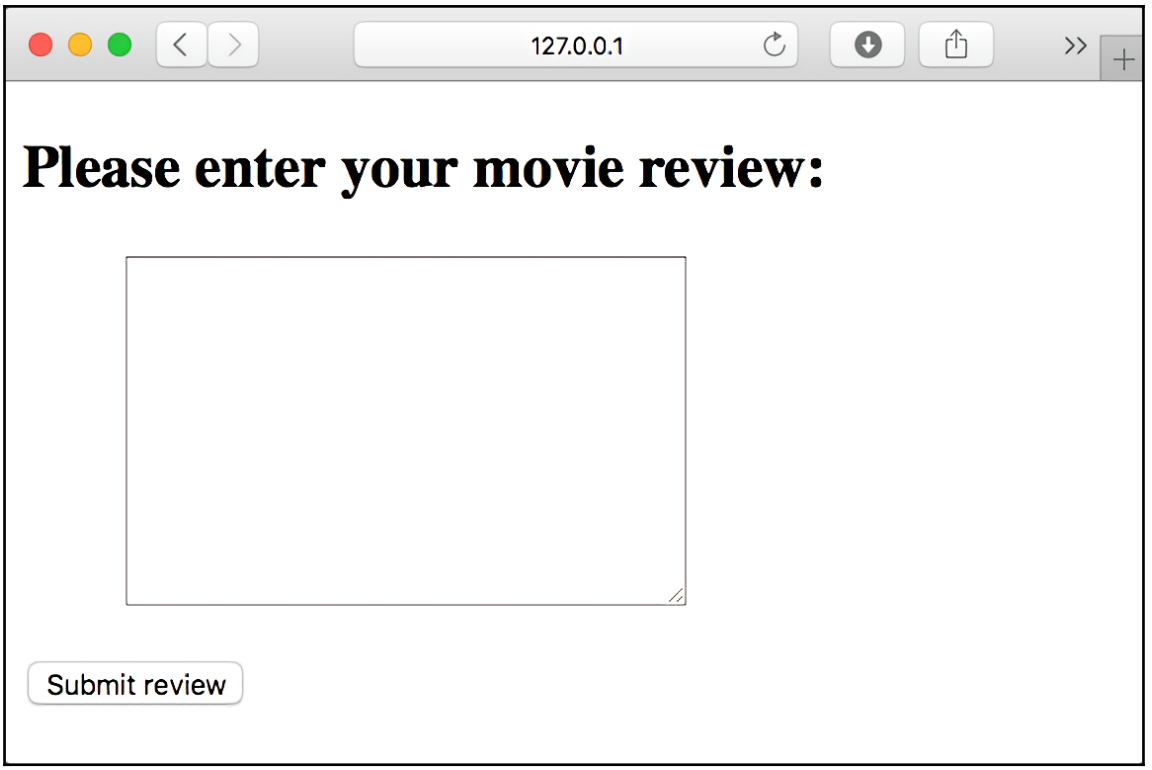
Chapter 9: Embedding a Machine Learning Model into a Web Application

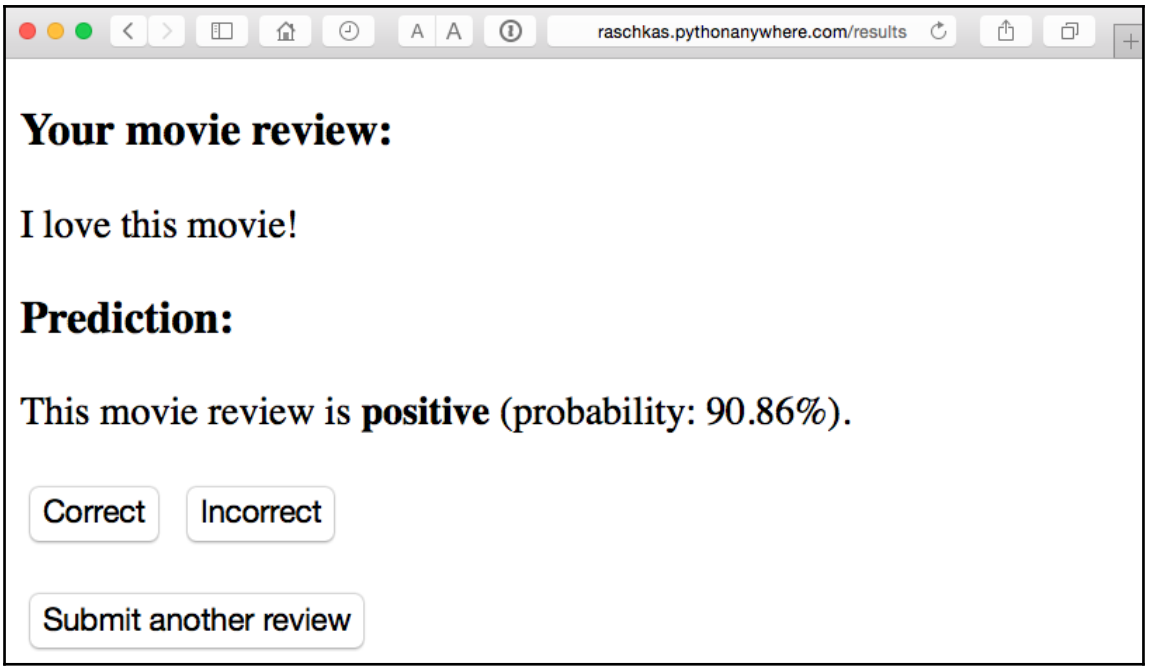


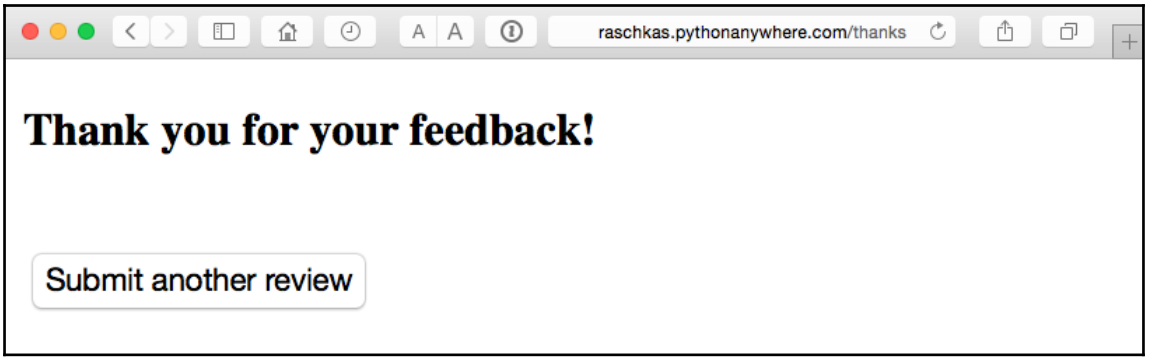


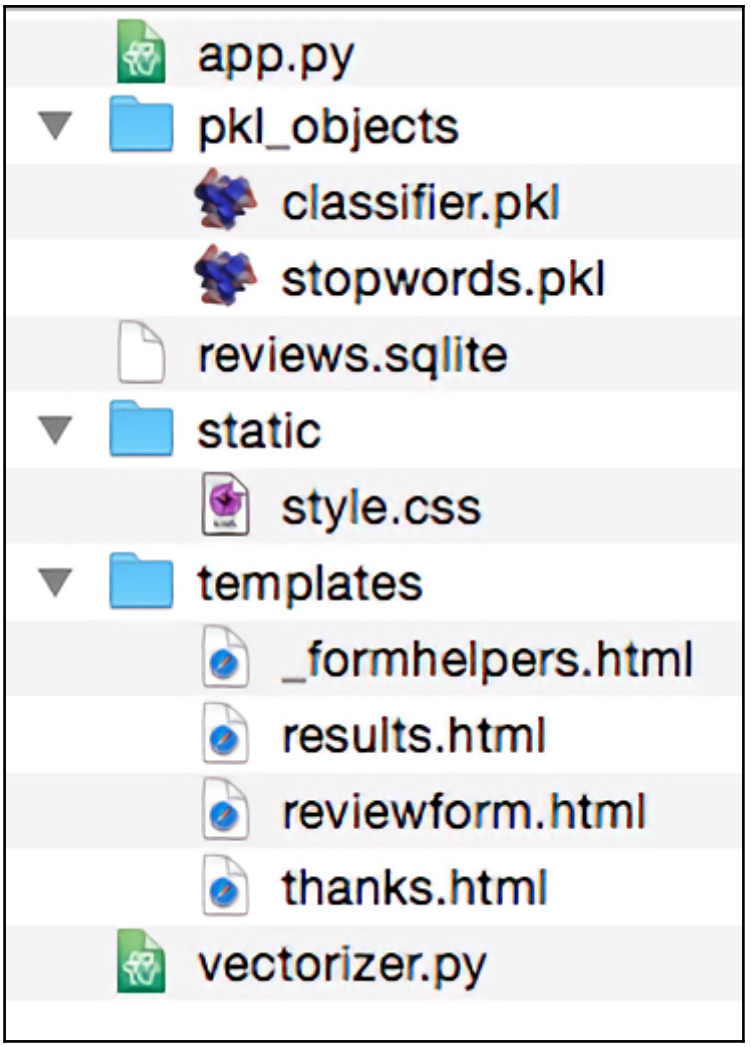

127.0.0.1
What's your name?
Sebastian
Say Hello

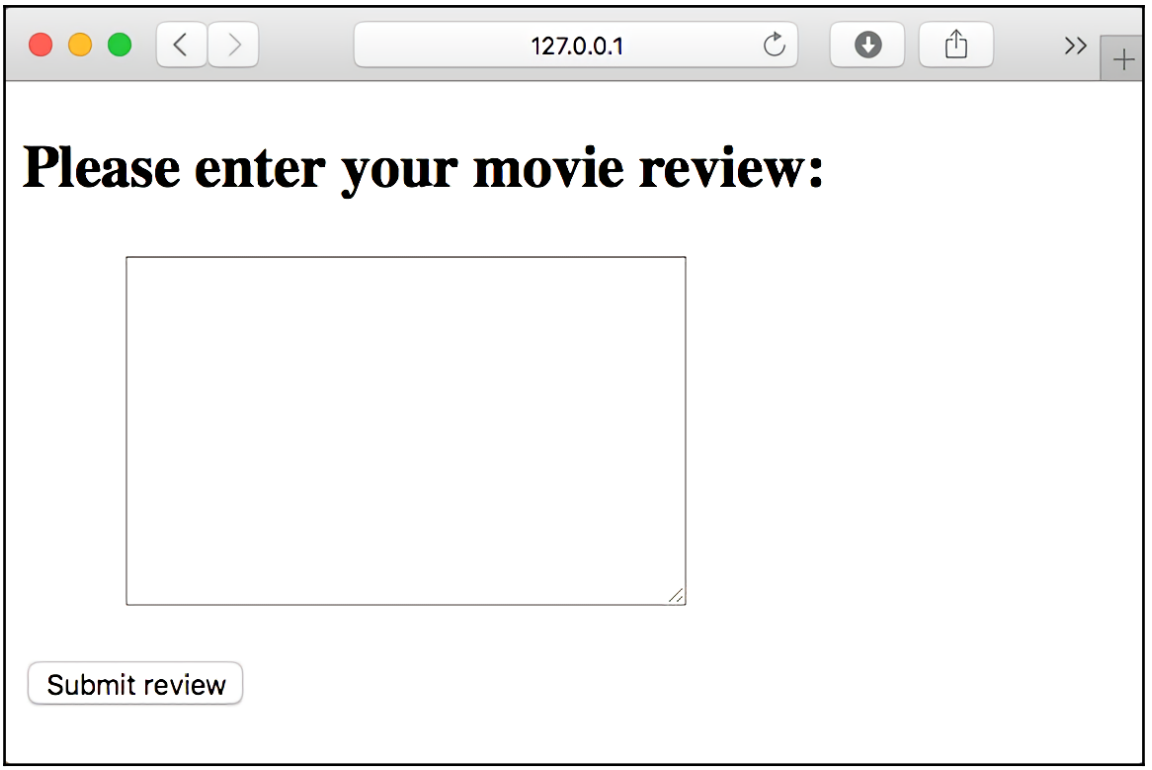

127.0.0.1:5000/hello
Hello Sebastian

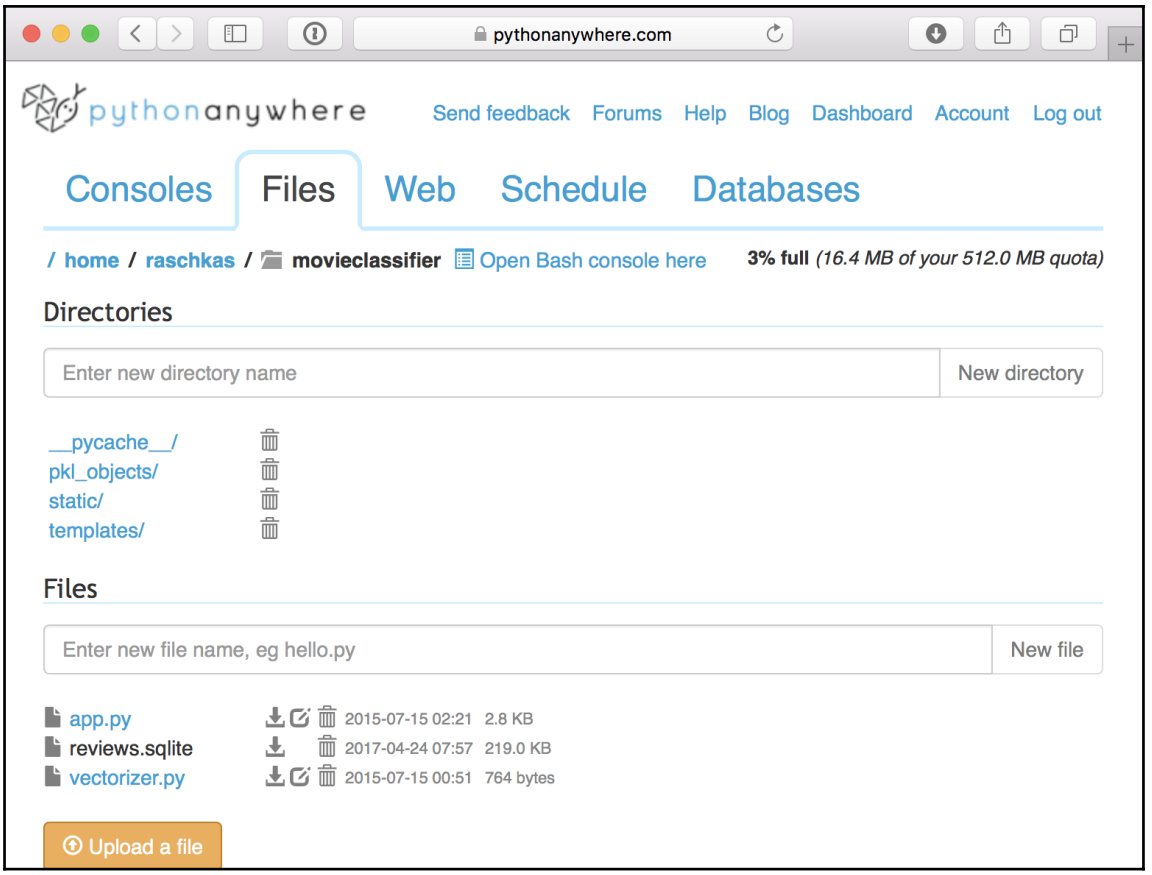

127.0.0.1
Please enter your movie review:
Submit review


raschkas.pythonanywhere.com/results
Your movie review:
I love this movie!
Prediction:
This movie review is positive (probability: 90.86%).
Correct
Incorrect
Submit another review

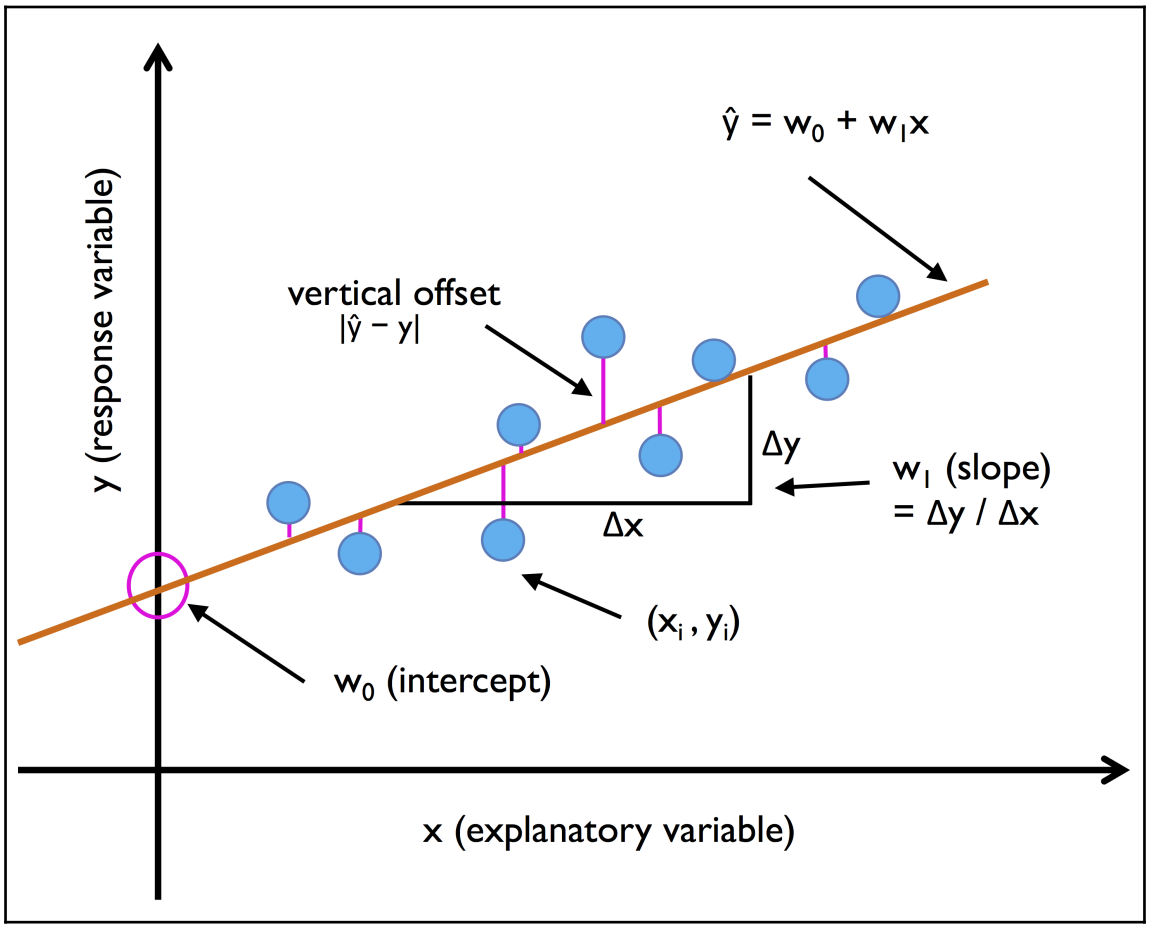

raschkas.pythonanywhere.com/thanks
Thank you for your feedback!
Submit another review

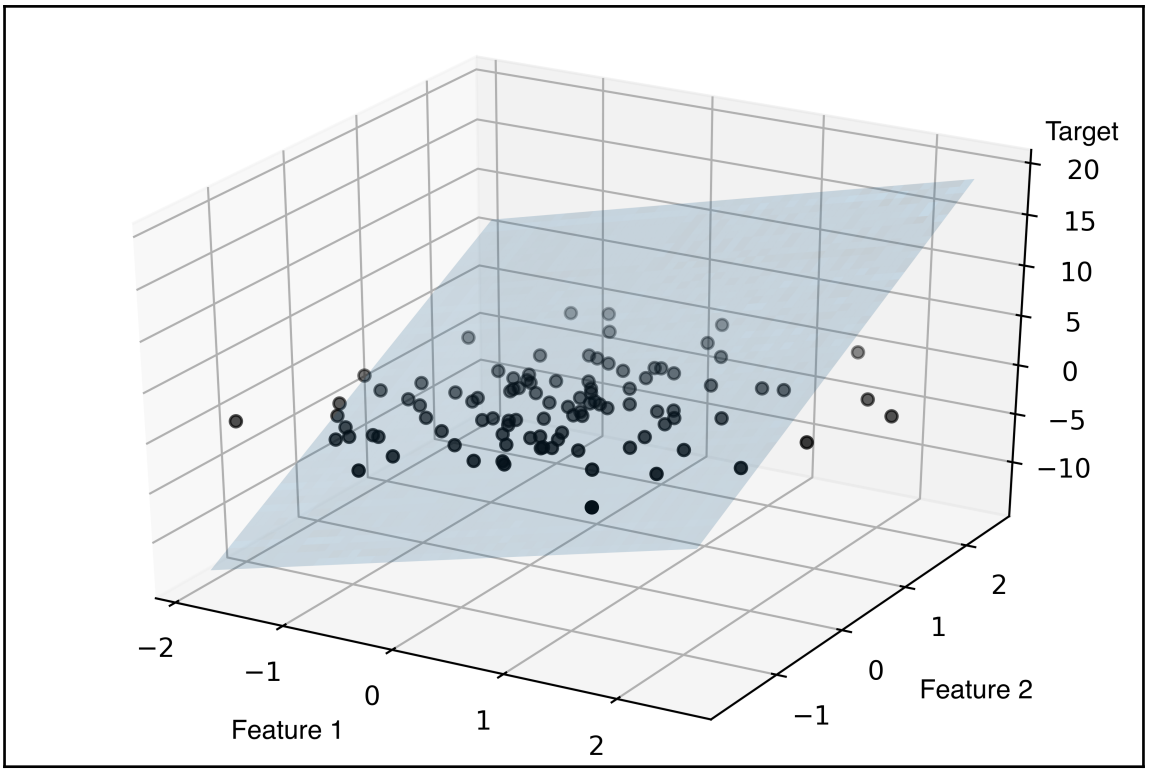

app.py
pkl_objects
classifier.pkl
stopwords.pkl
reviews.sqlite
static
style.css
templates
_formhelpers.html
results.html
reviewform.html
thanks.html
vectorizer.py


127.0.0.1
Please enter your movie review:
Submit review

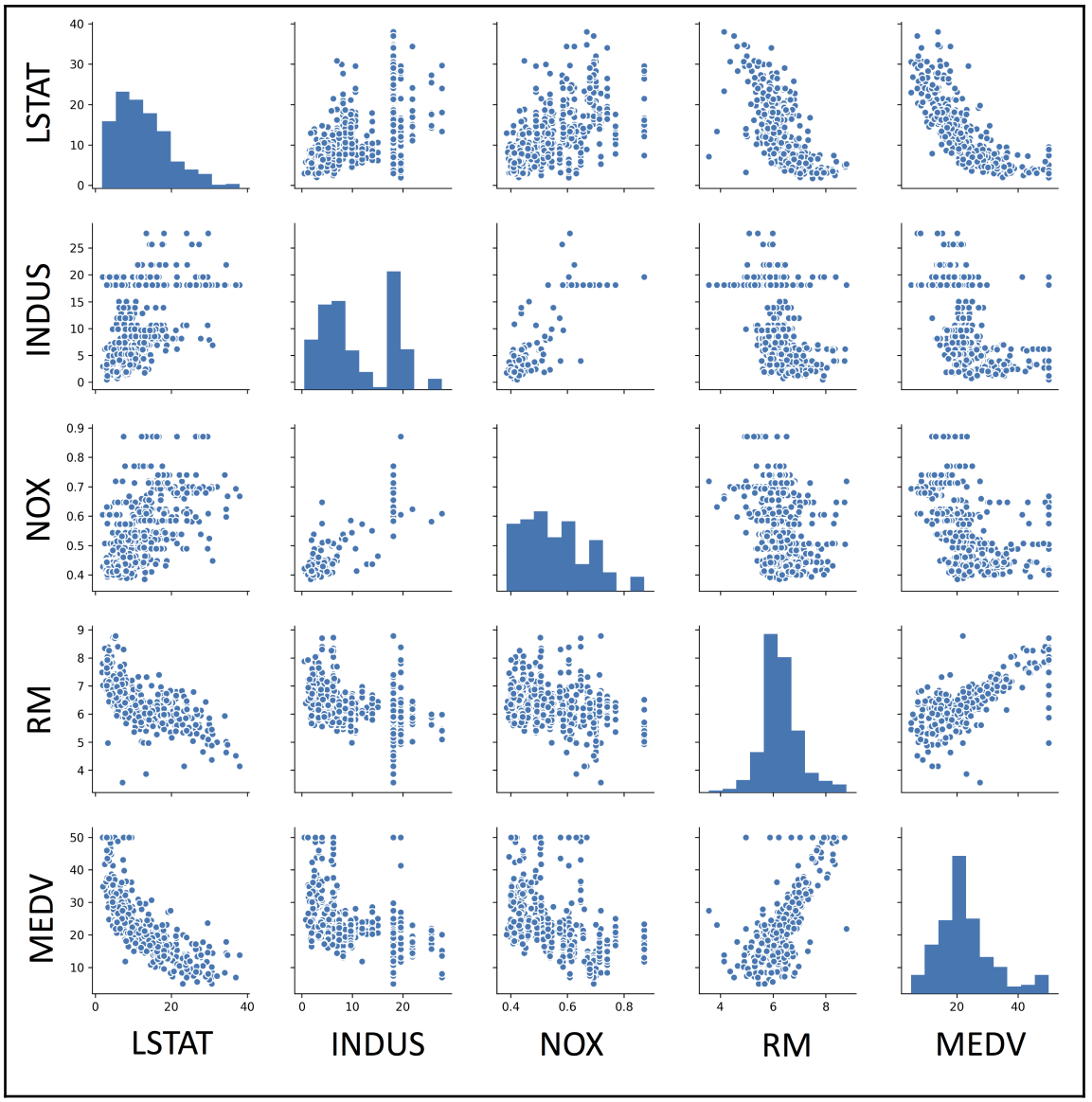

pythonanywhere.com
pythonanywhere
Send feedback
Forums
Help
Blog
Dashboard
Account
Log out
Consoles
Files
Web
Schedule
Databases
/ home / raschkas / movieclassifier
Open Bash console here
3% full (16.4 MB of your 512.0 MB quota)
Directories
Enter new directory name
New directory
__pycache__/
pkl_objects/
static/
templates/
Files
Enter new file name, eg hello.py
New file
app.py 2015-07-15 02:21 2.8 KB
reviews.sqlite 2017-04-24 07:57 219.0 KB
vectorizer.py 2015-07-15 00:51 764 bytes
Upload a file

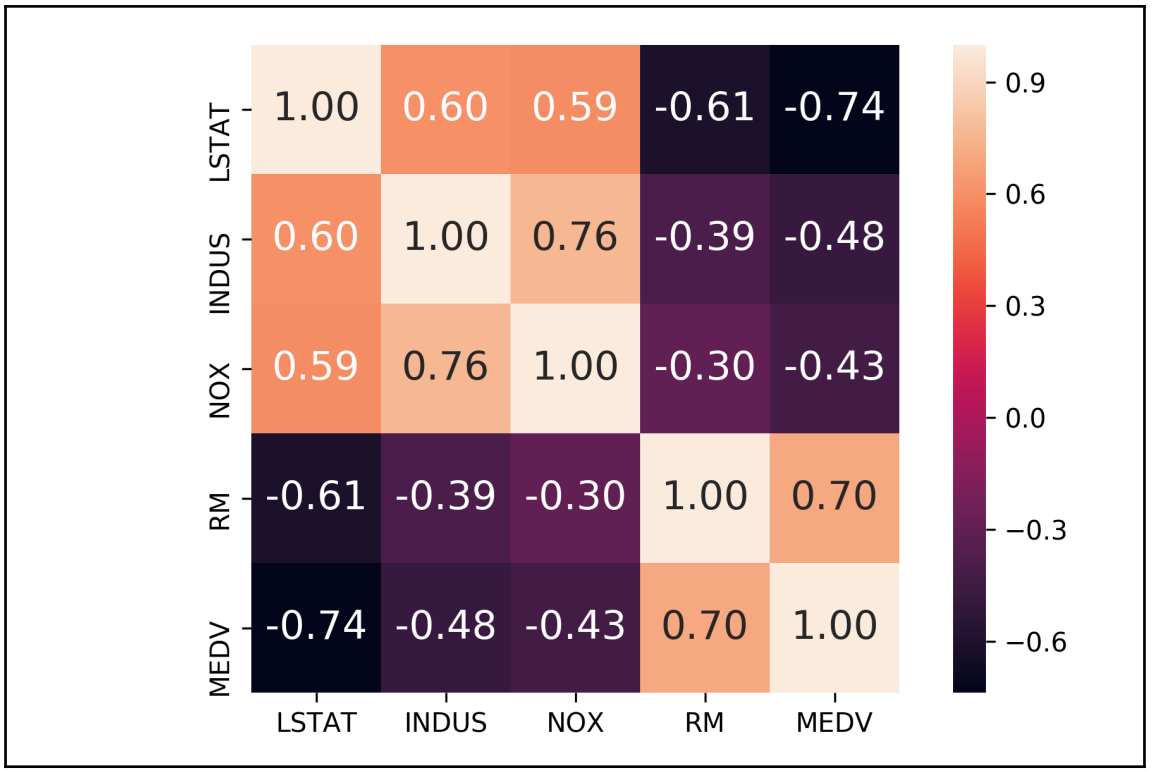
Chapter 10: Predicting Continuous Target Variables with Regression Analysis

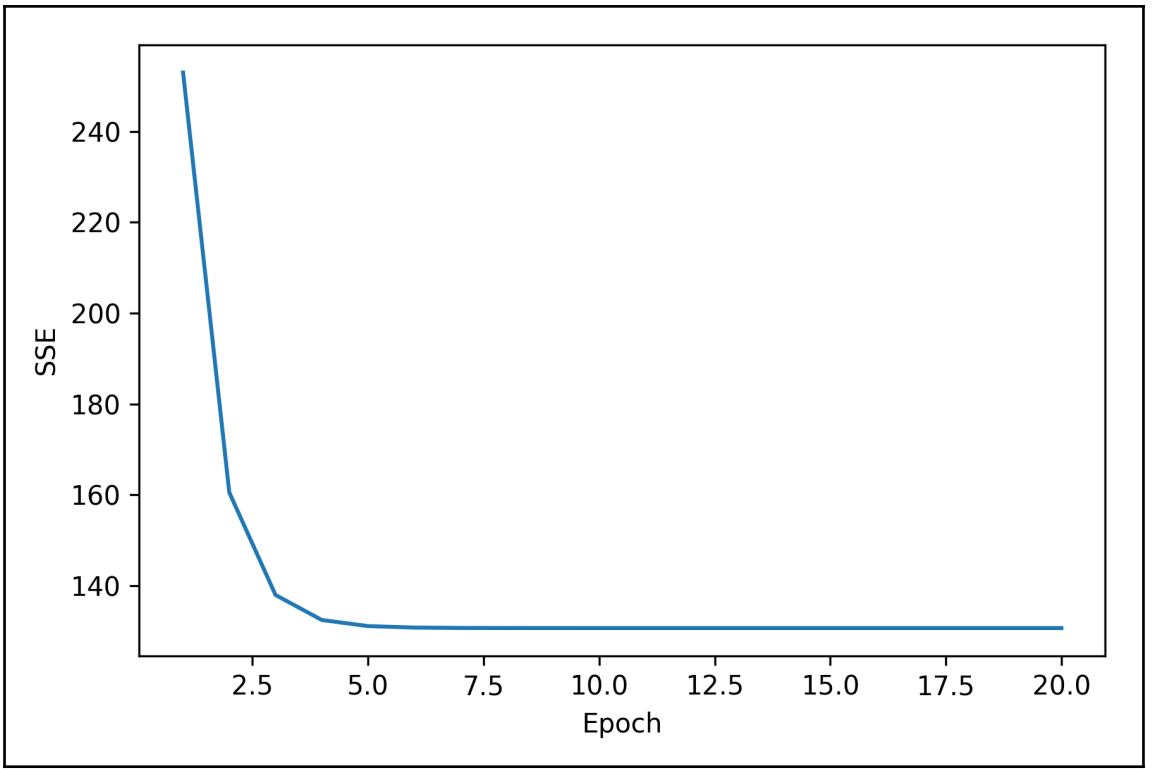


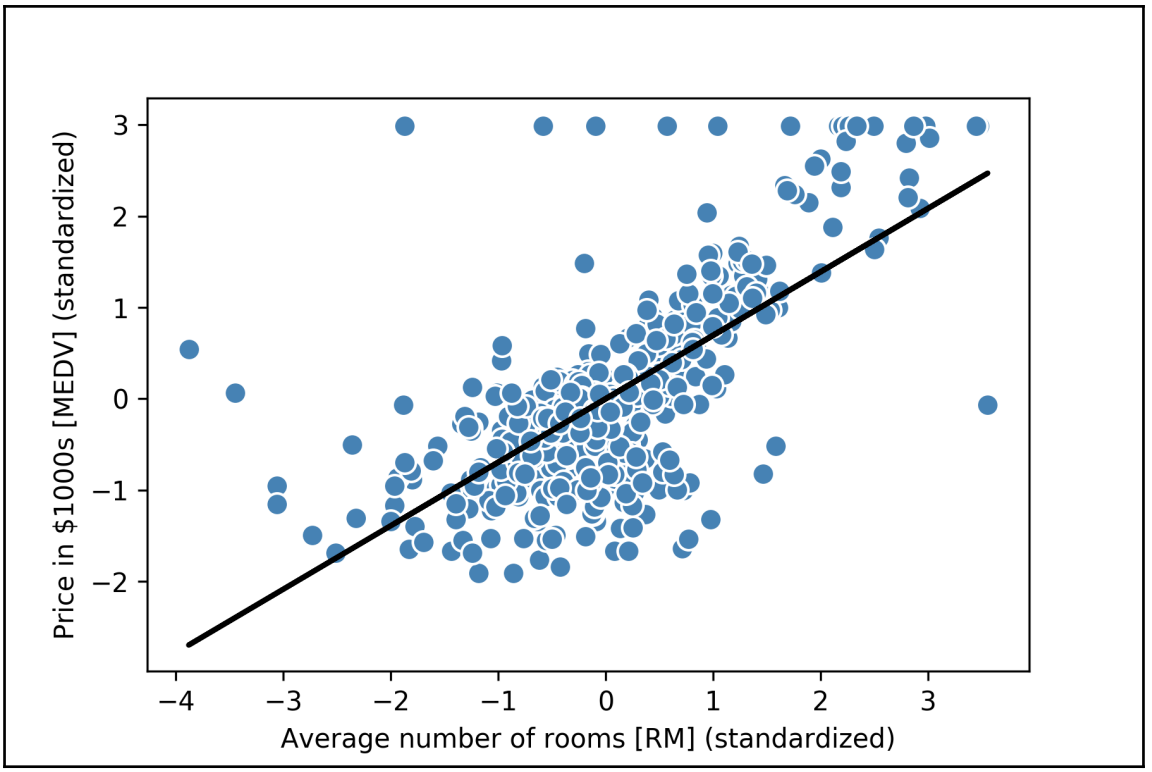


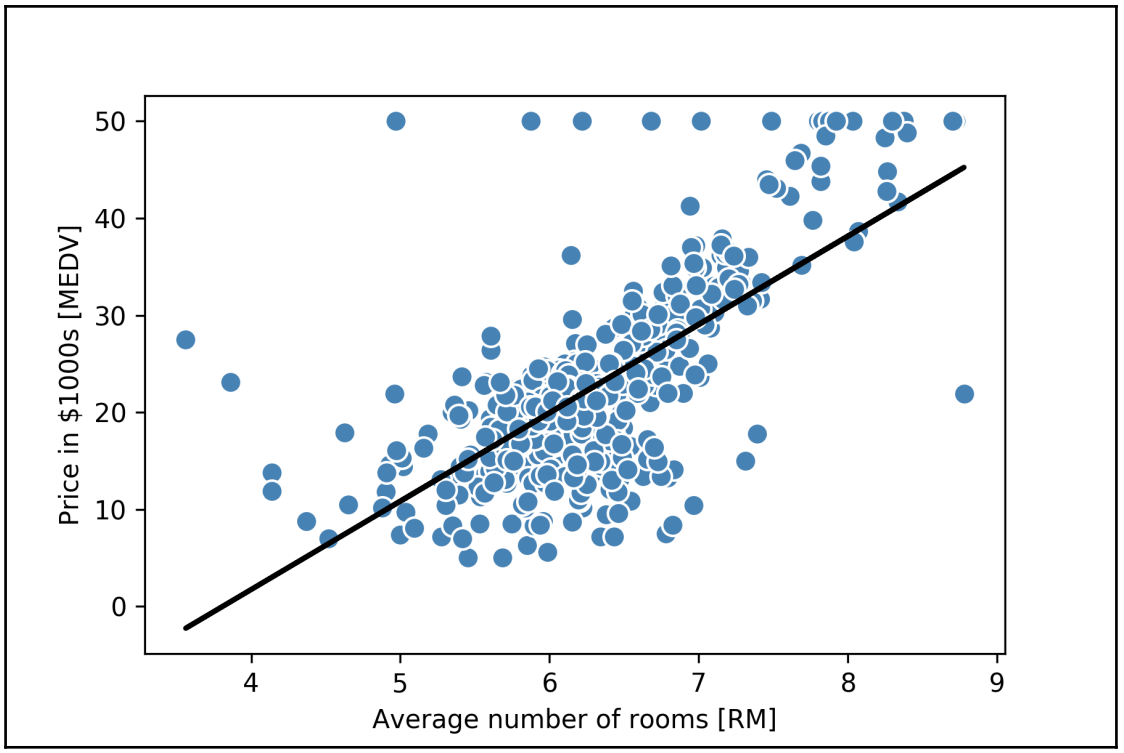
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

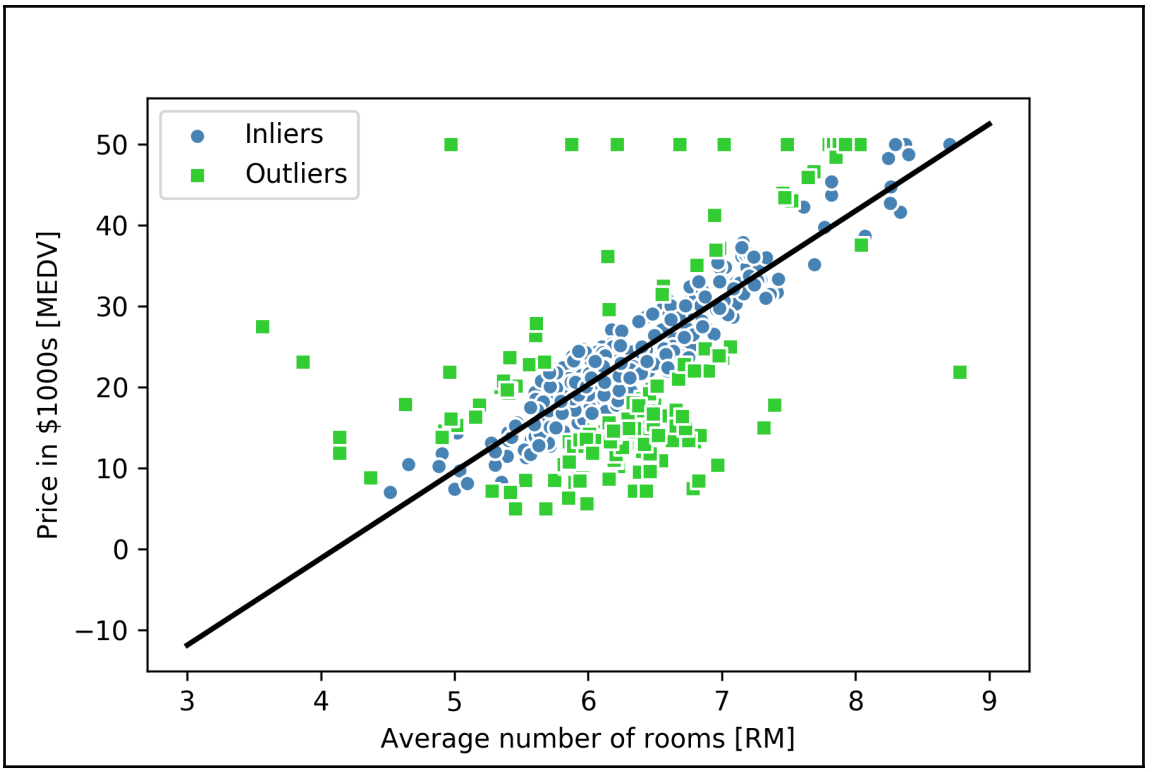

LSTAT
INDUS
NOX
RM
MEDV
LSTAT
INDUS
NOX
RM
MEDV

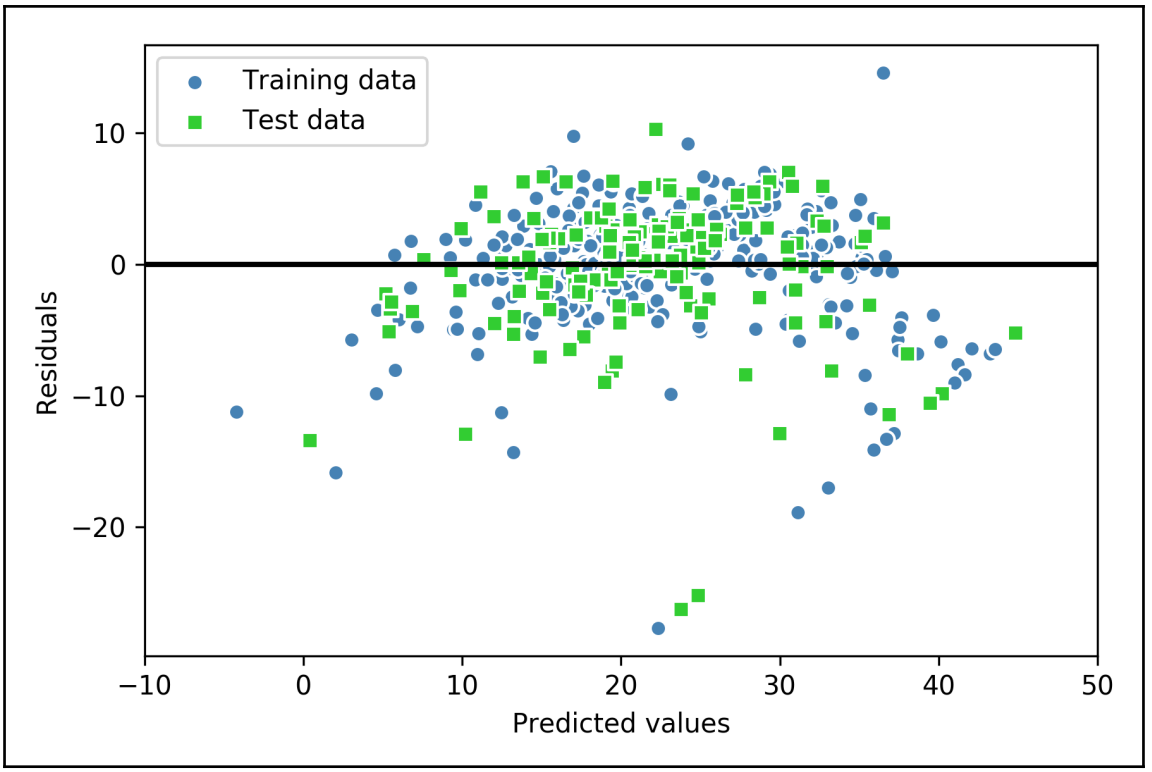

LSTAT
INDUS
NOX
RM
MEDV
1.00 0.60 0.59 -0.61 -0.74
0.60 1.00 0.76 -0.39 -0.48
0.59 0.76 1.00 -0.30 -0.43
-0.61 -0.39 -0.30 1.00 0.70
-0.74 -0.48 -0.43 0.70 1.00
LSTAT
INDUS
NOX
RM
MEDV
0.9
0.6
0.3
0.0
−0.3
−0.6

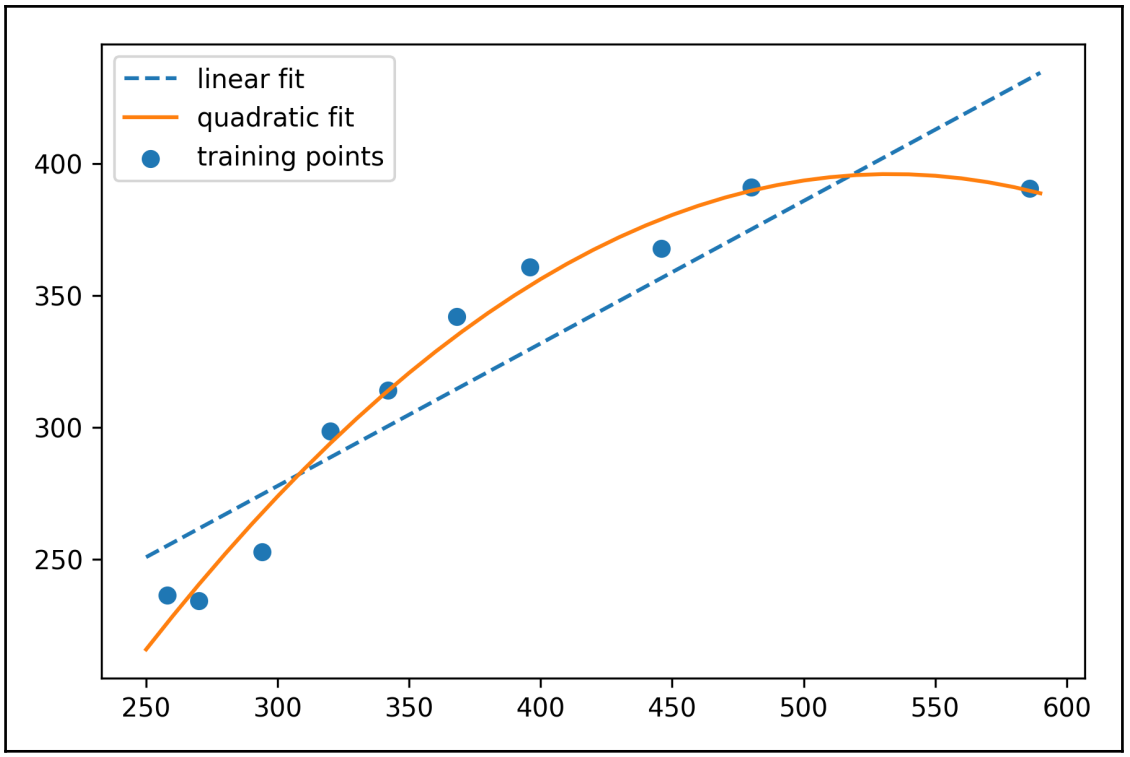

240
220
200
180
160
140
SSE
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epoch

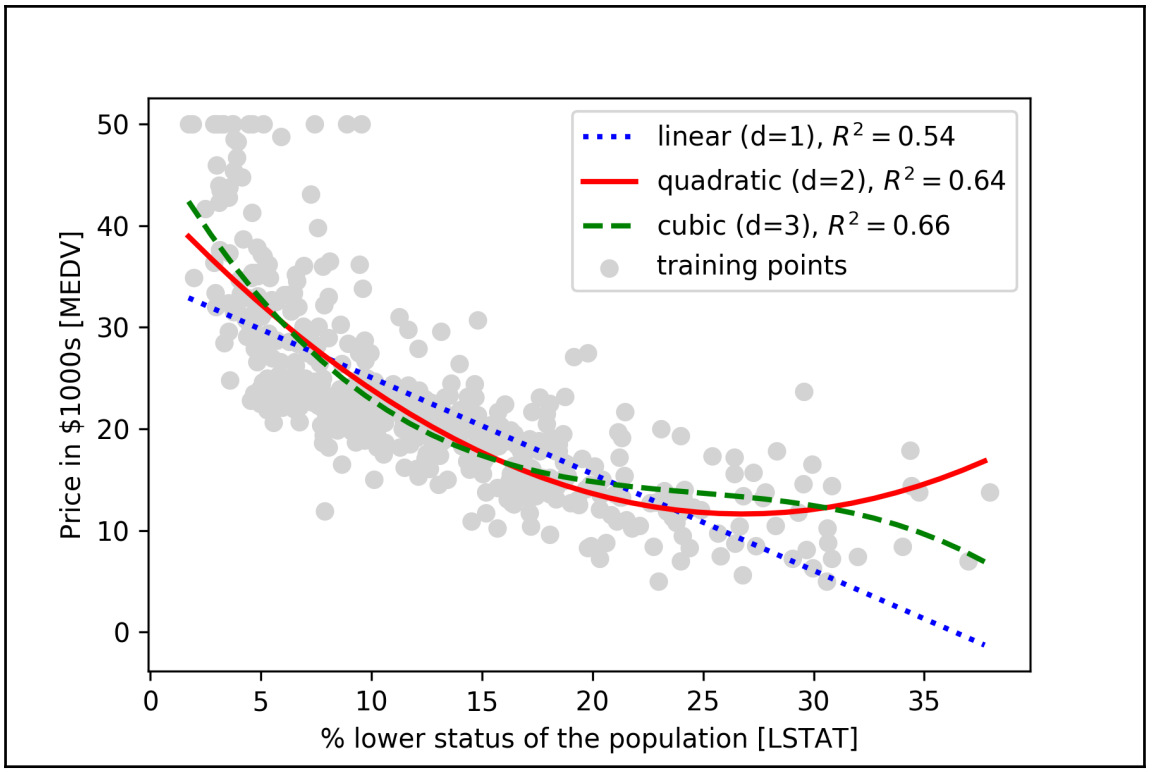

Price in $1000s [MEDV] (standardized)
3
2
1
0
−1
−2
−4
−3
−2
−1
0
1
2
3
Average number of rooms [RM] (standardized)

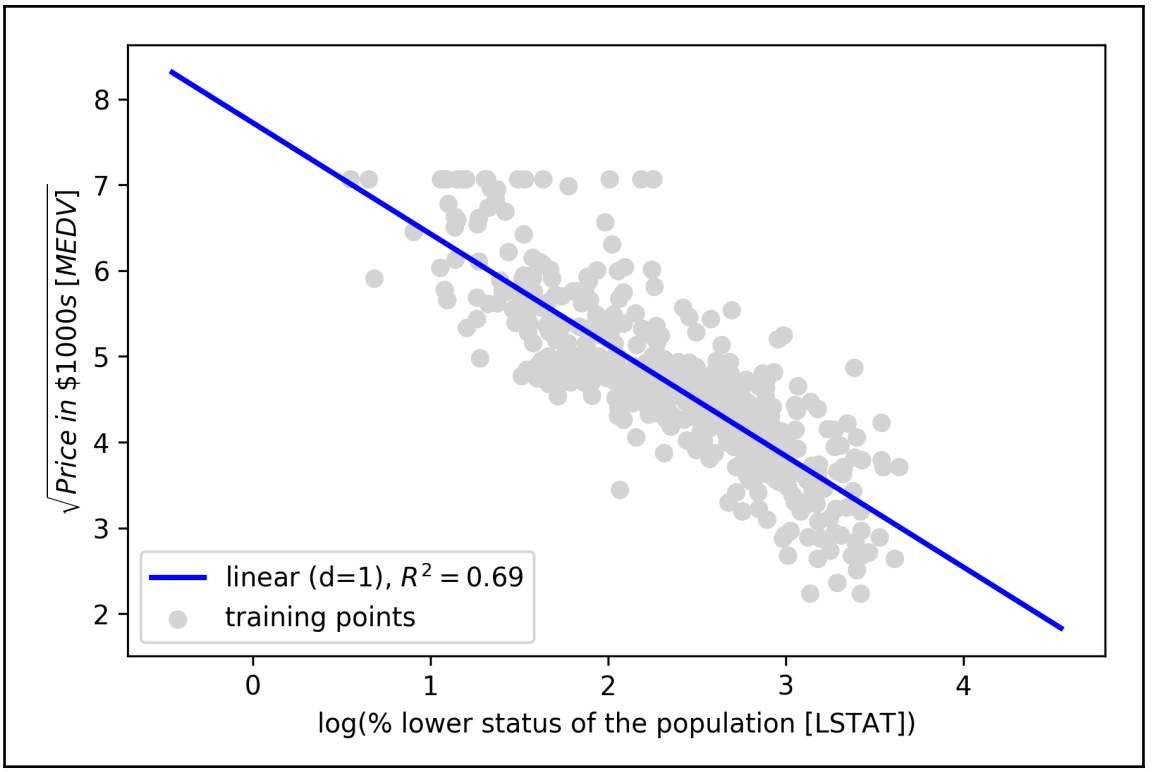

Price in $1000s [MEDV]
Average number of rooms [RM]
0
10
20
30
40
50
4
5
6
7
8
9

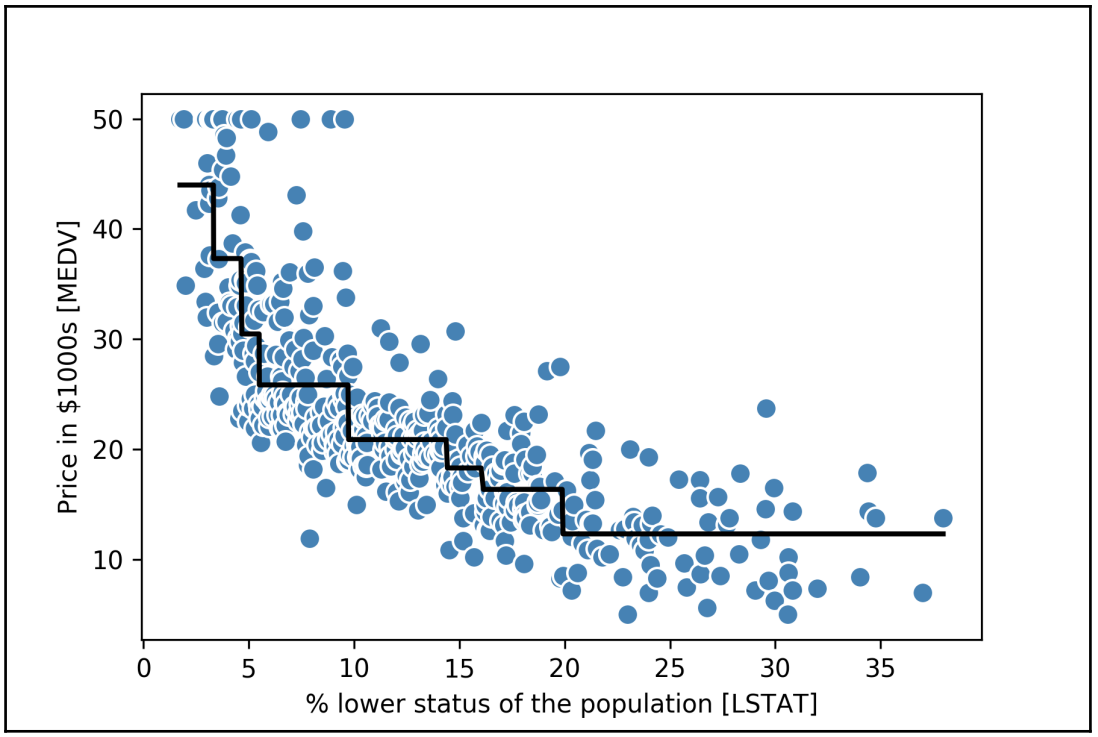

Inliers
Outliers
Price in $1000s [MEDV]
Average number of rooms [RM]
50
40
30
20
10
0
−10
3
4
5
6
7
8
9

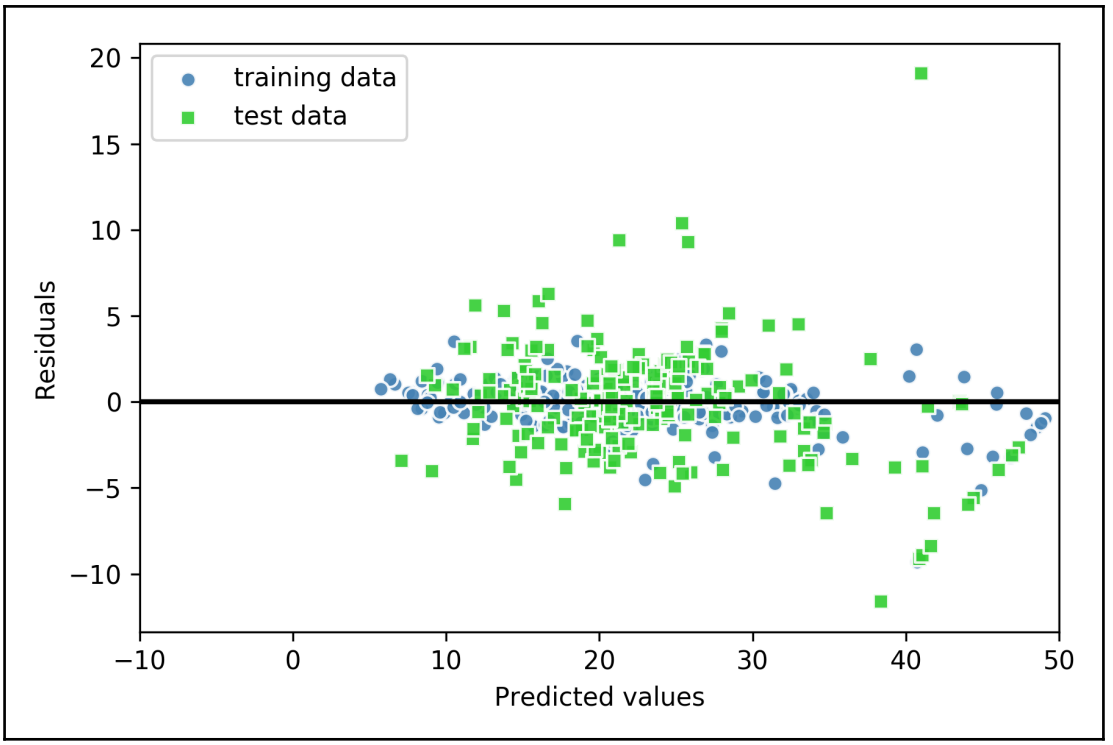

Training data
Test data
Residuals
Predicted values
10
0
−10
−20
−10
0
10
20
30
40
50


linear fit
quadratic fit
training points
400
350
300
250
250
300
350
400
450
500
550
600

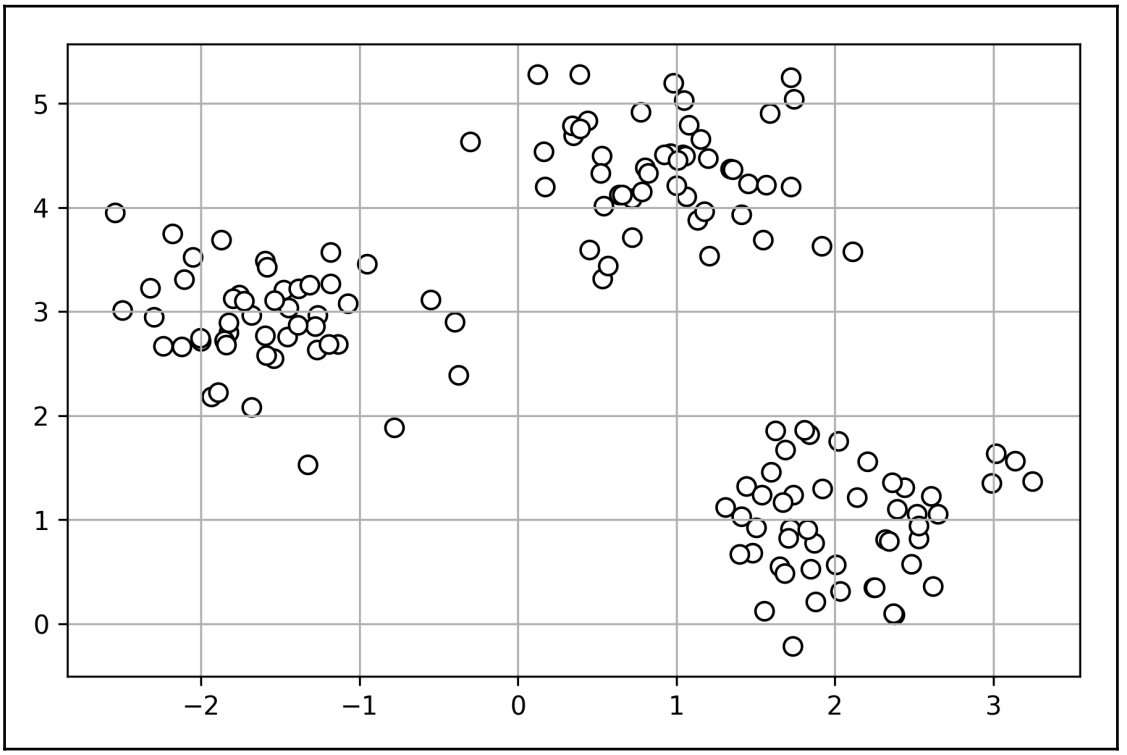

linear (d=1), $R^2 = 0.54$
quadratic (d=2), $R^2 = 0.64$
cubic (d=3), $R^2 = 0.66$
training points
Price in $1000s [MEDV]
% lower status of the population [LSTAT]
0
10
20
30
40
50
0
5
10
15
20
25
30
35

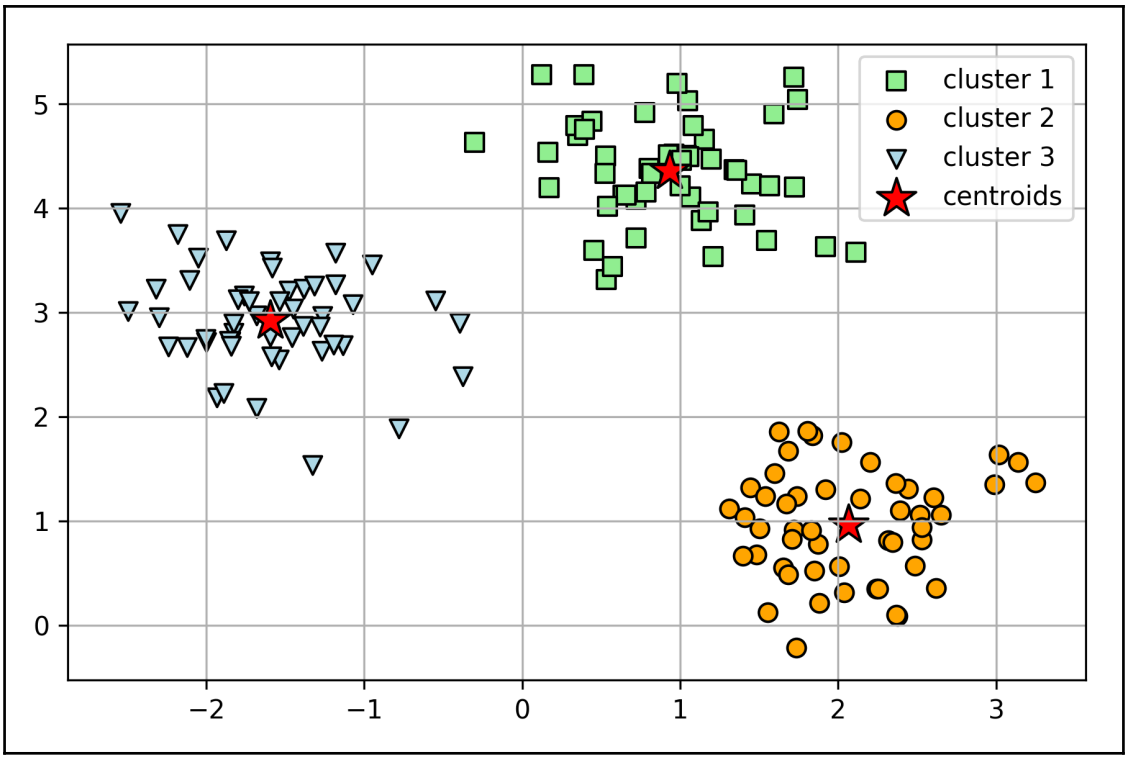

8
7
6
5
4
3
2
$\sqrt{Price\ in\ \$1000s\ [MEDV]}$
linear (d=1), $R^2 = 0.69$
training points
0
1
2
3
4
log(% lower status of the population [LSTAT])

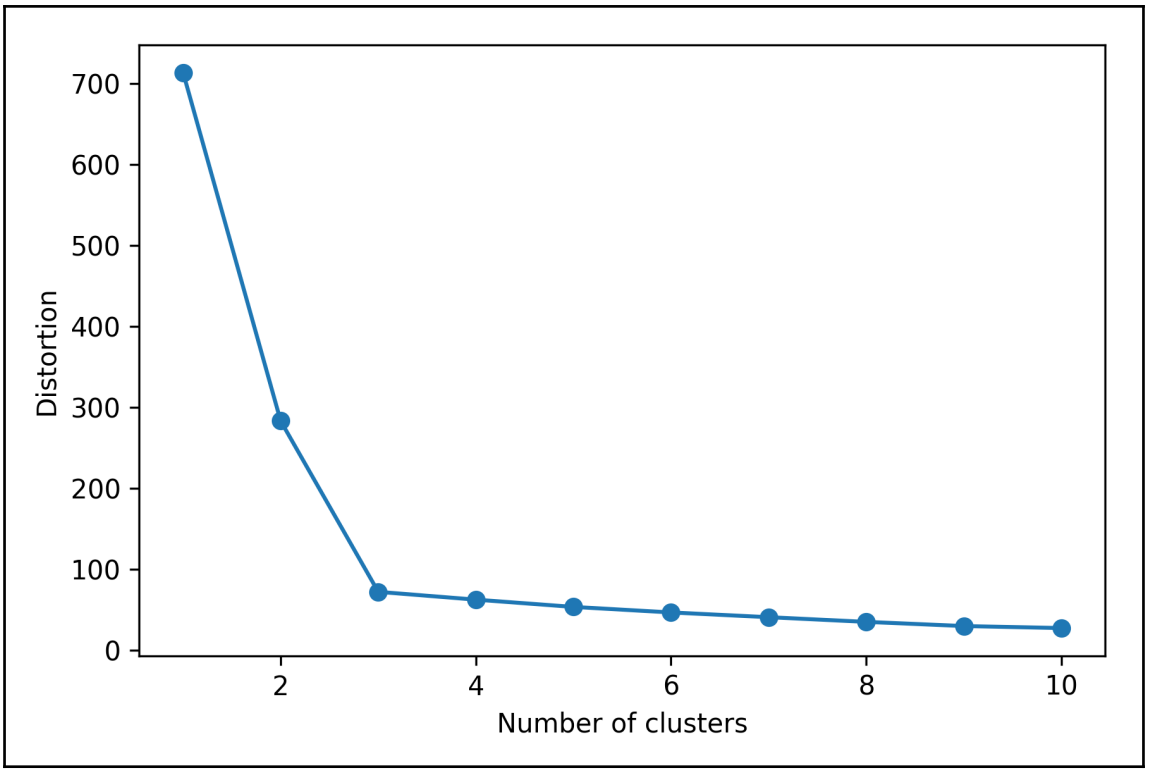

Price in $1000s [MEDV]
50
40
30
20
10
0
5
10
15
20
25
30
35
% lower status of the population [LSTAT]

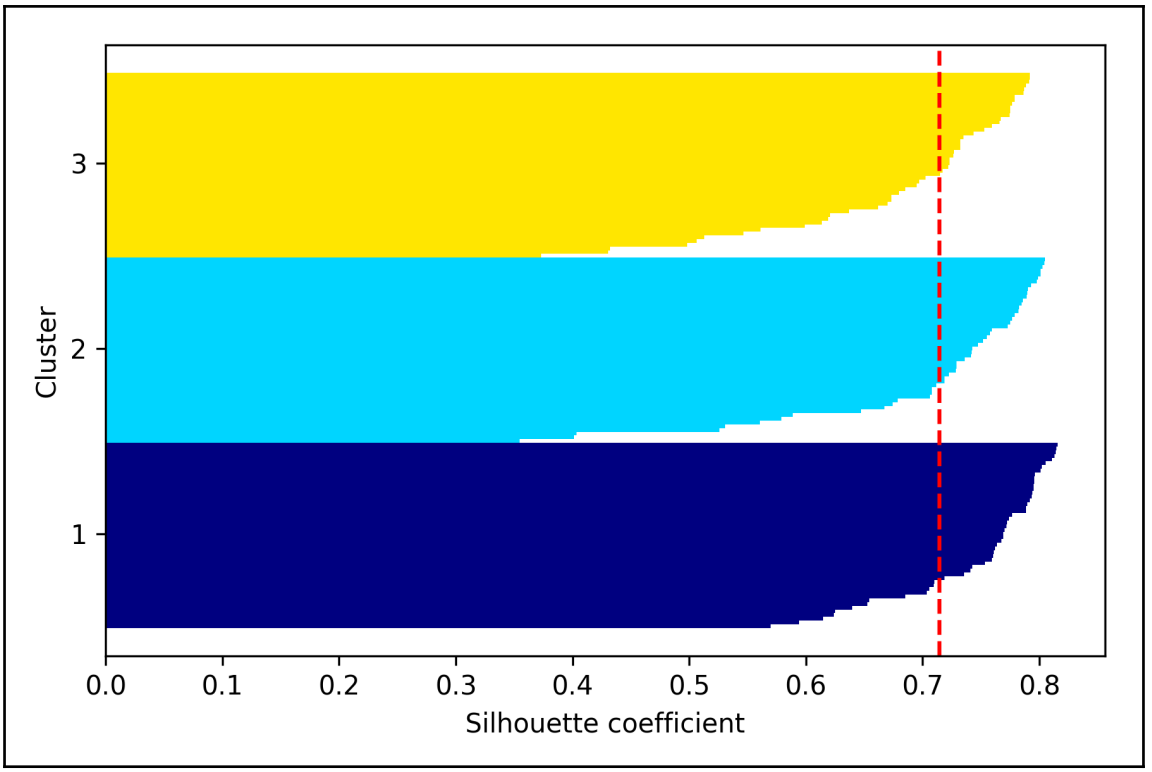

training data
test data
Residuals
Predicted values
20
15
10
5
0
−5
−10
−10
0
10
20
30
40
50

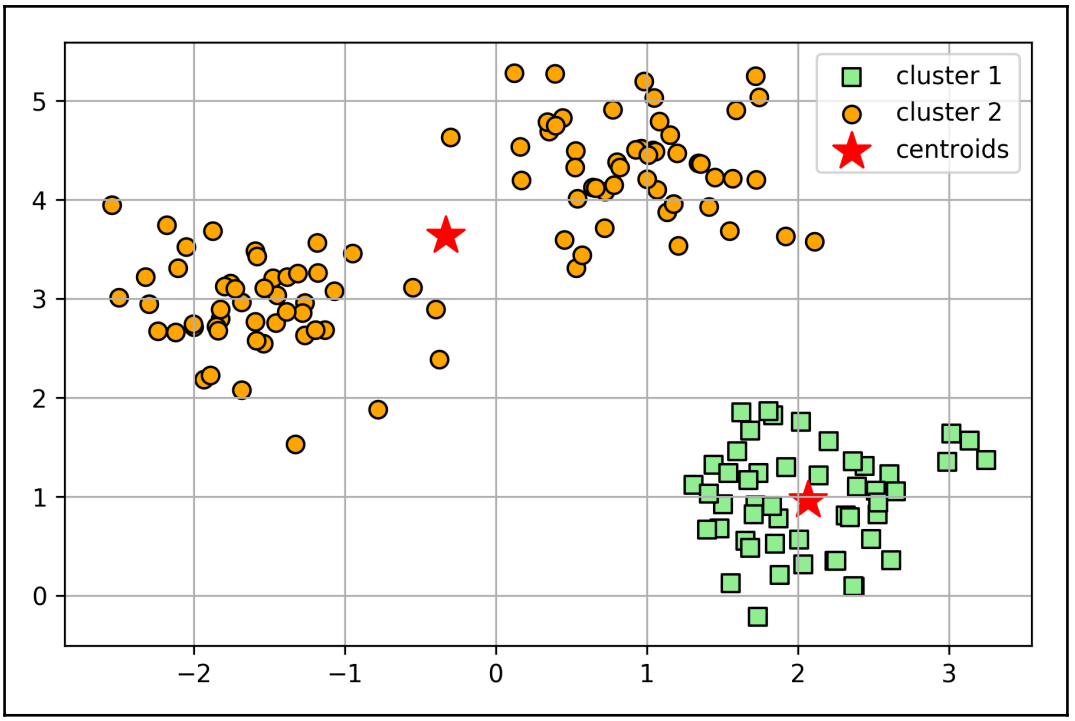
Chapter 11: Working with Unlabeled Data – Clustering Analysis

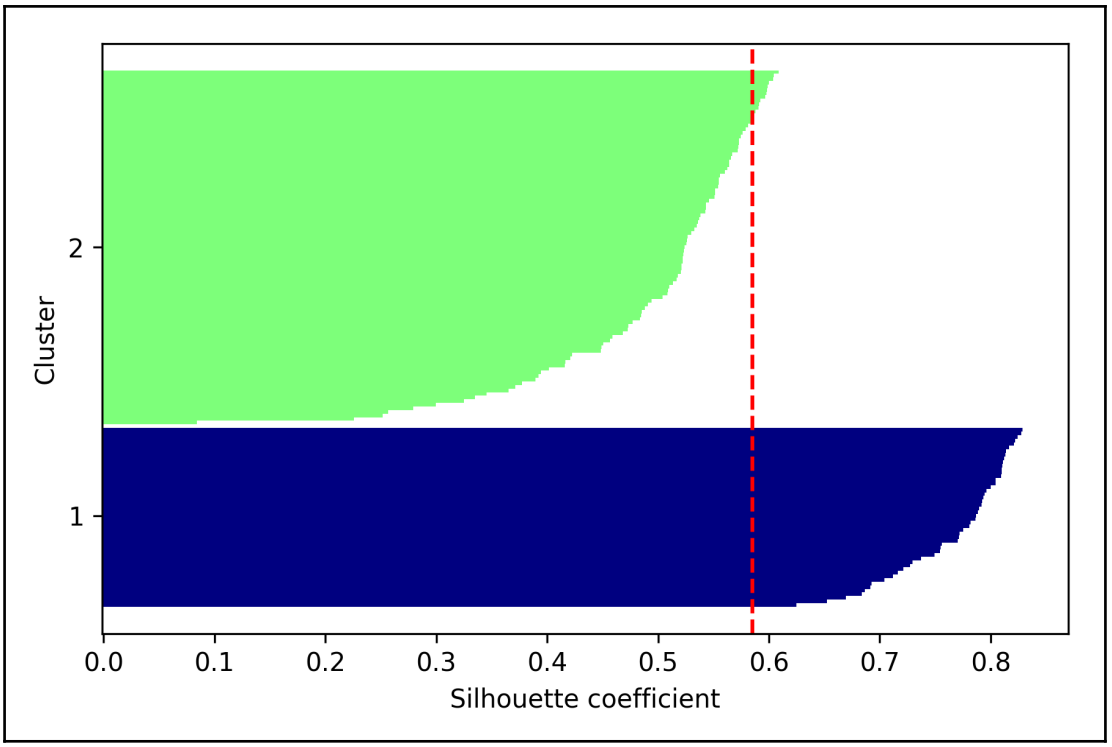


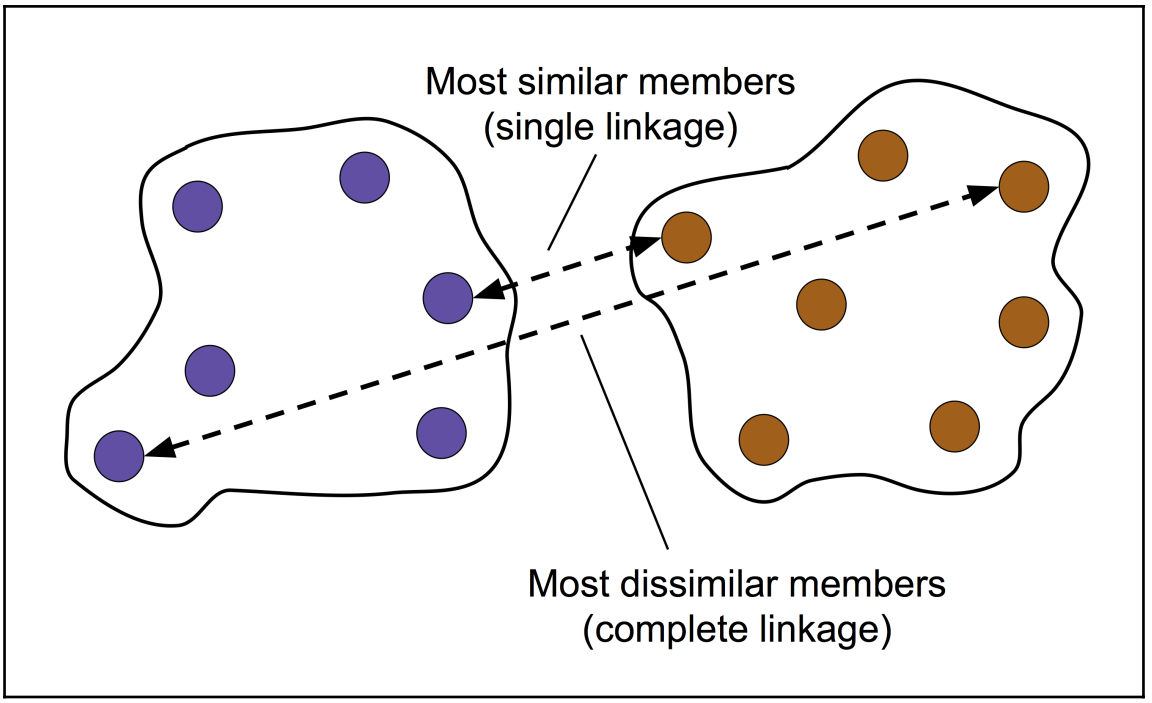

cluster 1
cluster 2
cluster 3
centroids
5
4
3
2
1
0
−2
−1
0
1
2
3


Distortion
700
600
500
400
300
200
100
0
2
4
6
8
10
Number of clusters


Cluster
3
2
1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Silhouette coefficient


cluster 1
cluster 2
centroids
5
4
3
2
1
0
−2
−1
0
1
2
3

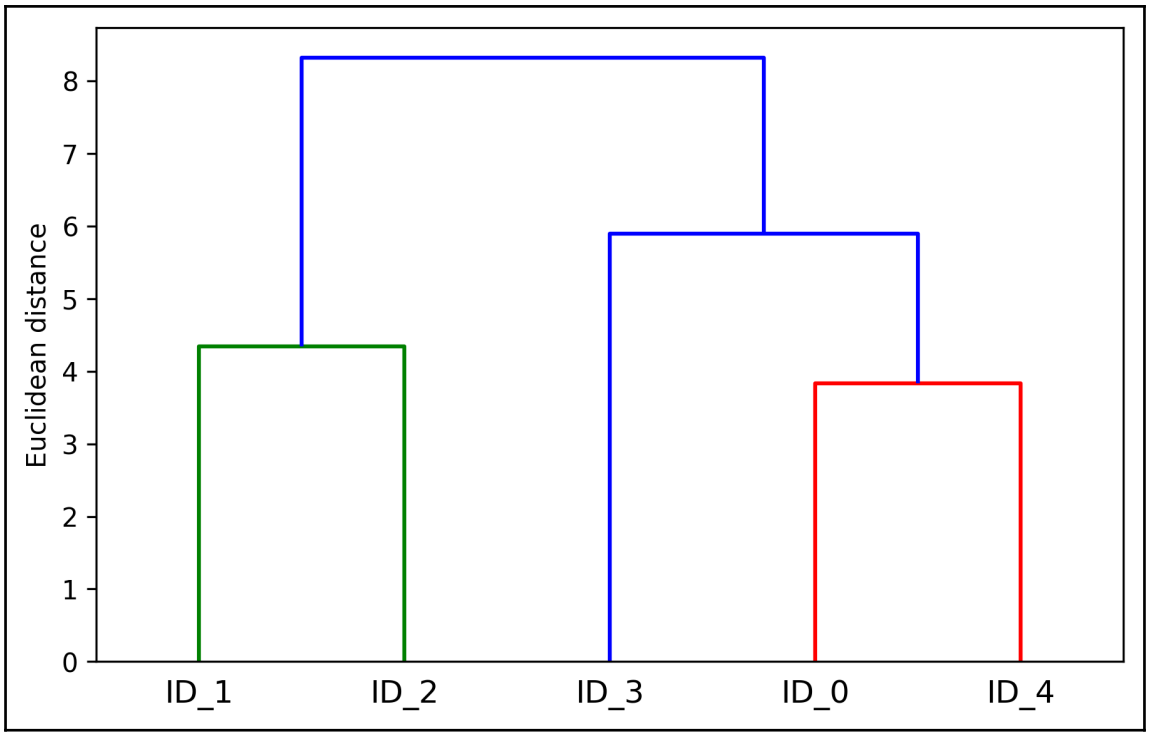

Cluster
2
1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Silhouette coefficient

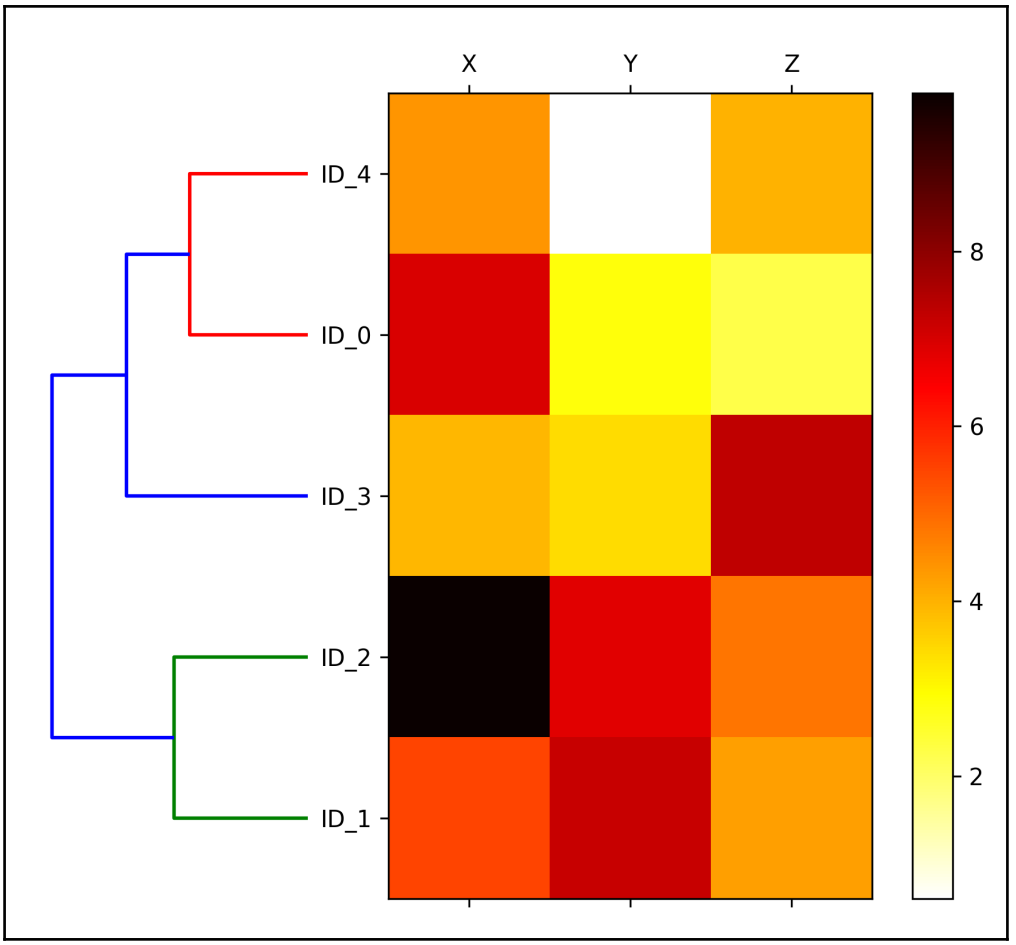


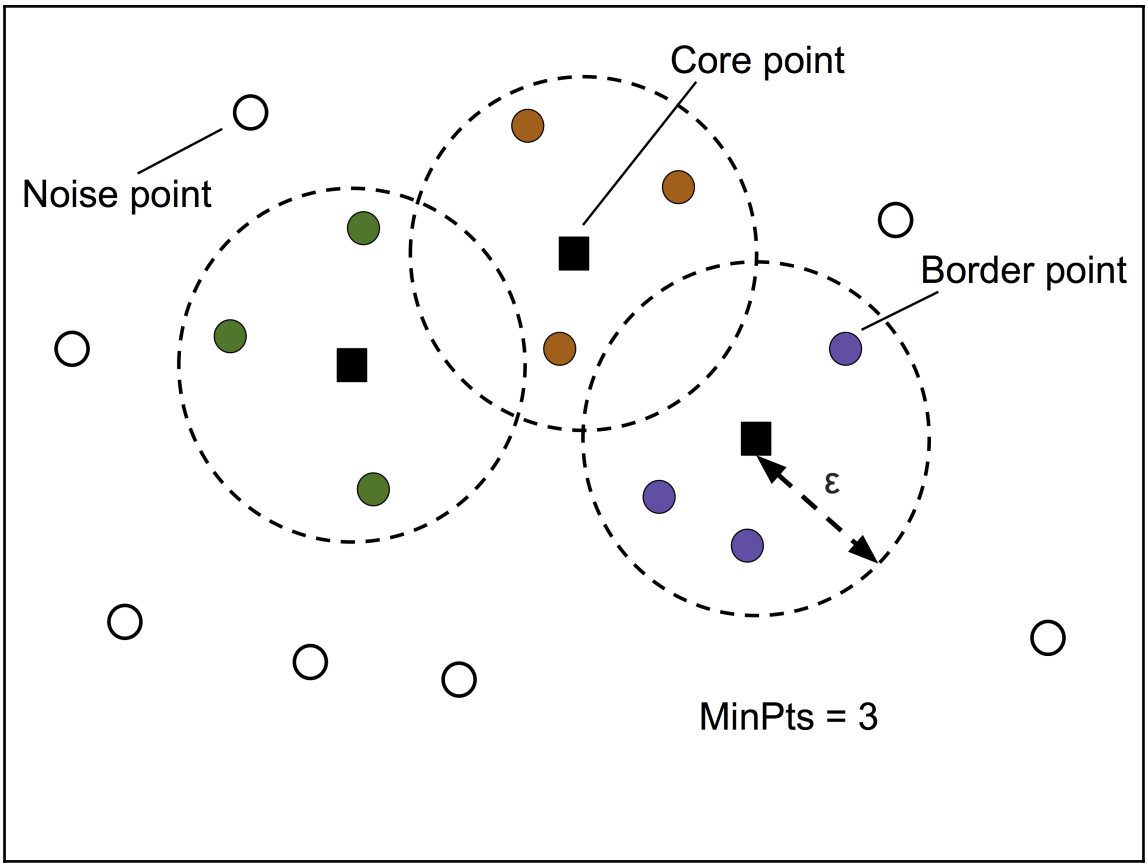
	X	Y	Z
ID_0	6.964692	2.861393	2.268515
ID_1	5.513148	7.194690	4.231065
ID_2	9.807642	6.848297	4.809319
ID_3	3.921175	3.431780	7.290497
ID_4	4.385722	0.596779	3.980443

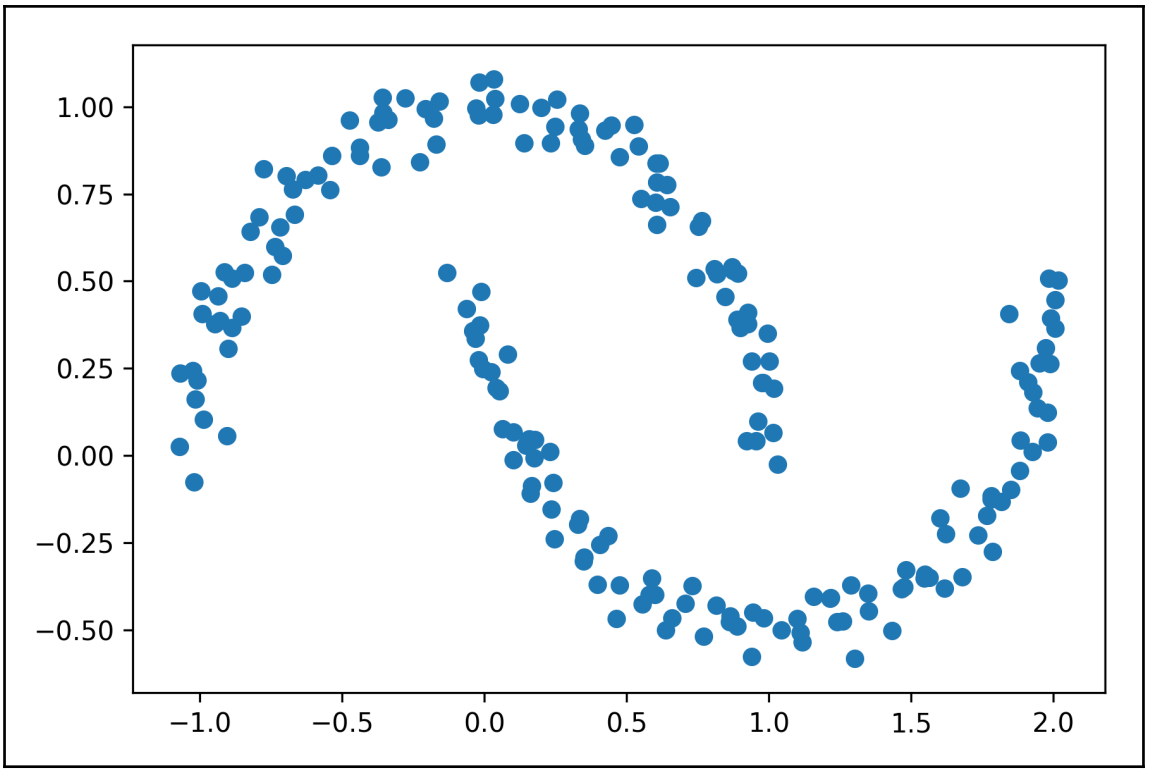
	X	Y	Z
ID_0	6.964692	2.861393	2.268515
ID_1	5.513148	7.194690	4.231065
ID_2	9.807642	6.848297	4.809319
ID_3	3.921175	3.431780	7.290497
ID_4	4.385722	0.596779	3.980443

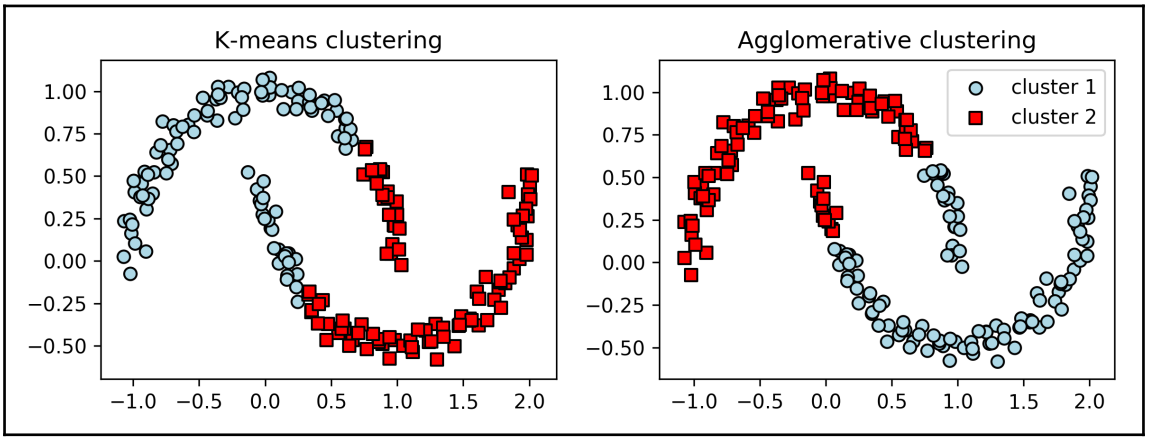
	row label 1	row label 2	distance	no. of items in clust.
cluster 1	0.0	4.0	3.835396	2.0
cluster 2	1.0	2.0	4.347073	2.0
cluster 3	3.0	5.0	5.899885	3.0
cluster 4	6.0	7.0	8.316594	5.0


Euclidean distance
0
1
2
3
4
5
6
7
8
ID_1
ID_2
ID_3
ID_0
ID_4

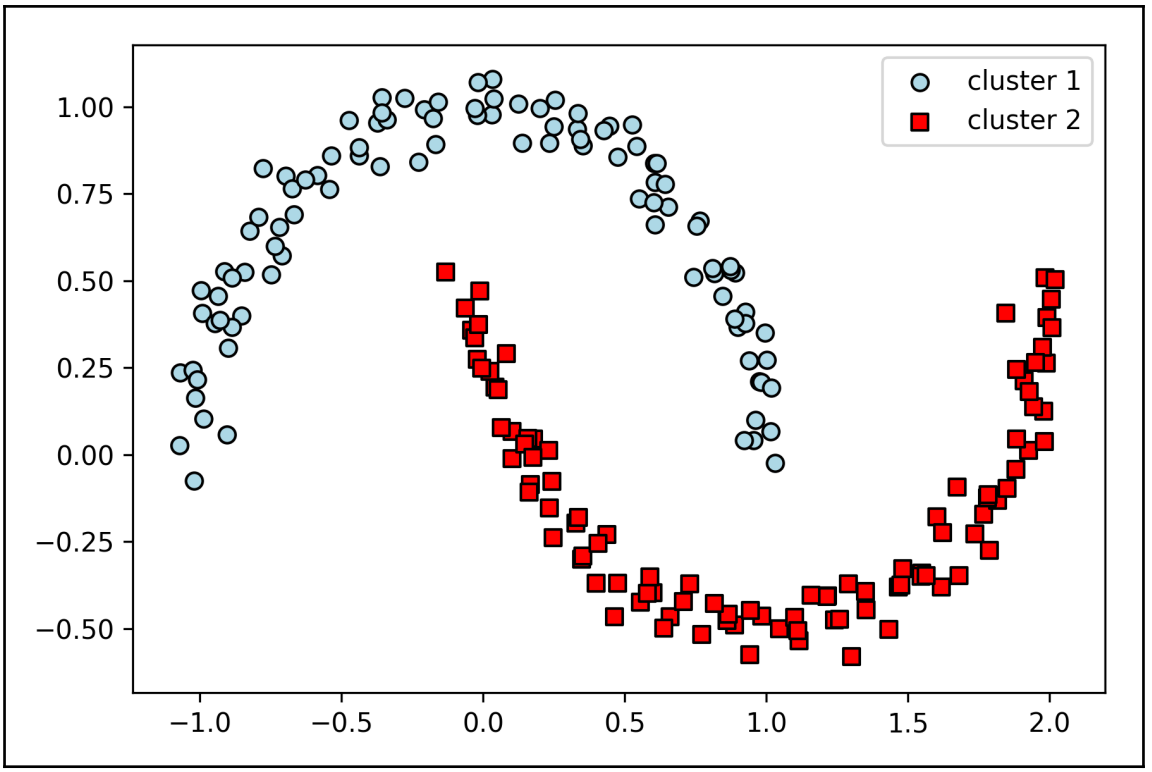

X
Y
Z
ID_4
ID_0
ID_3
ID_2
ID_1
8
6
4
2


Core point
Noise point
Border point
ε
MinPts = 3

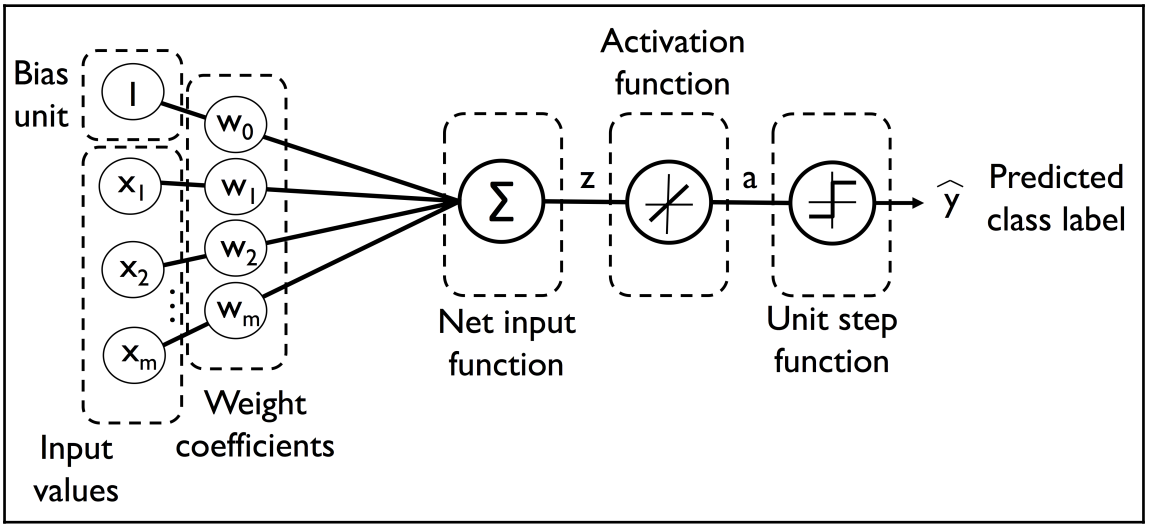

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

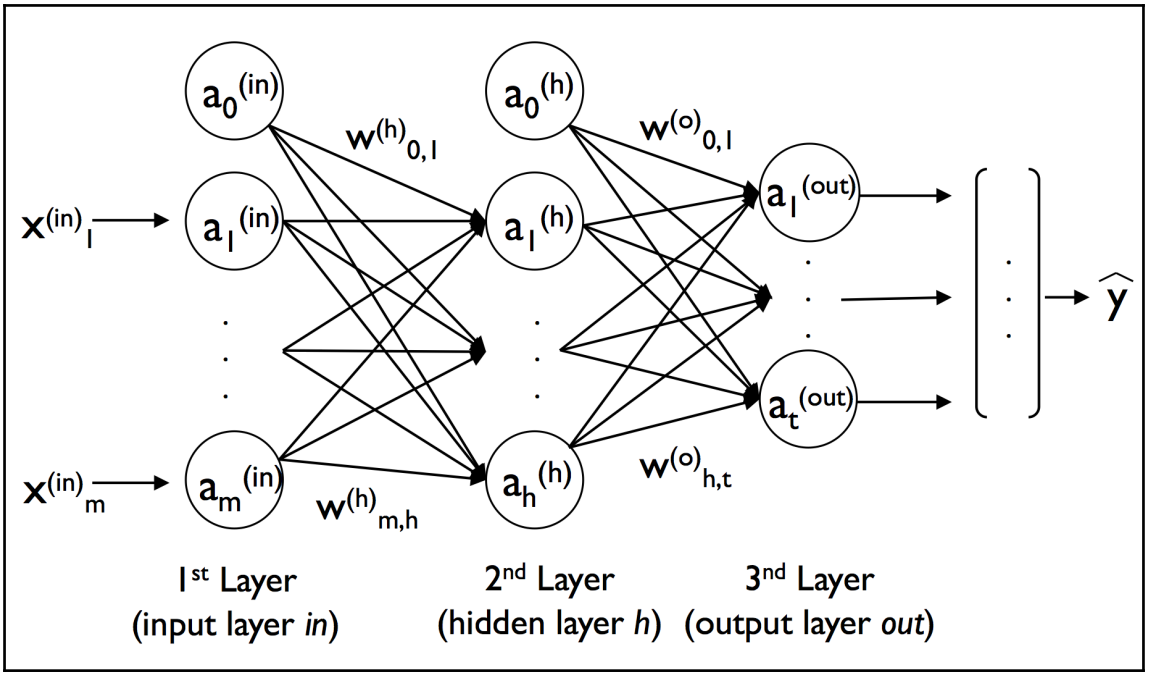

K-means clustering
Agglomerative clustering
cluster 1
cluster 2

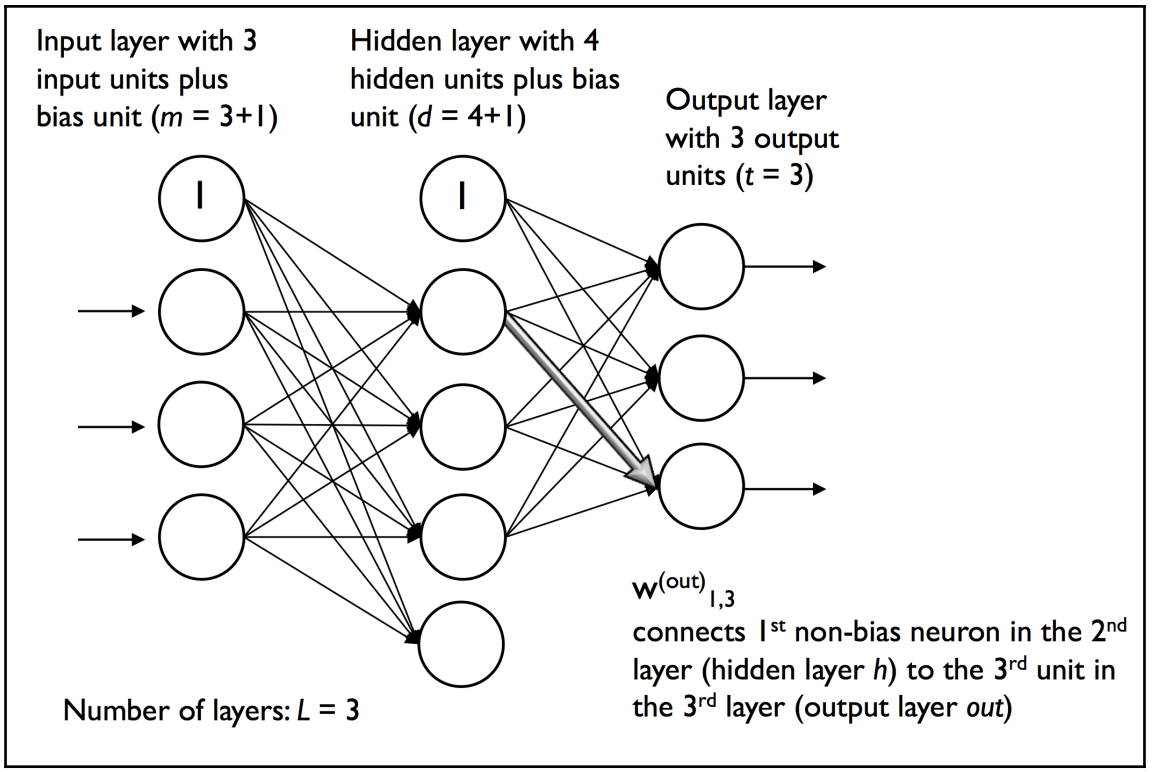
>

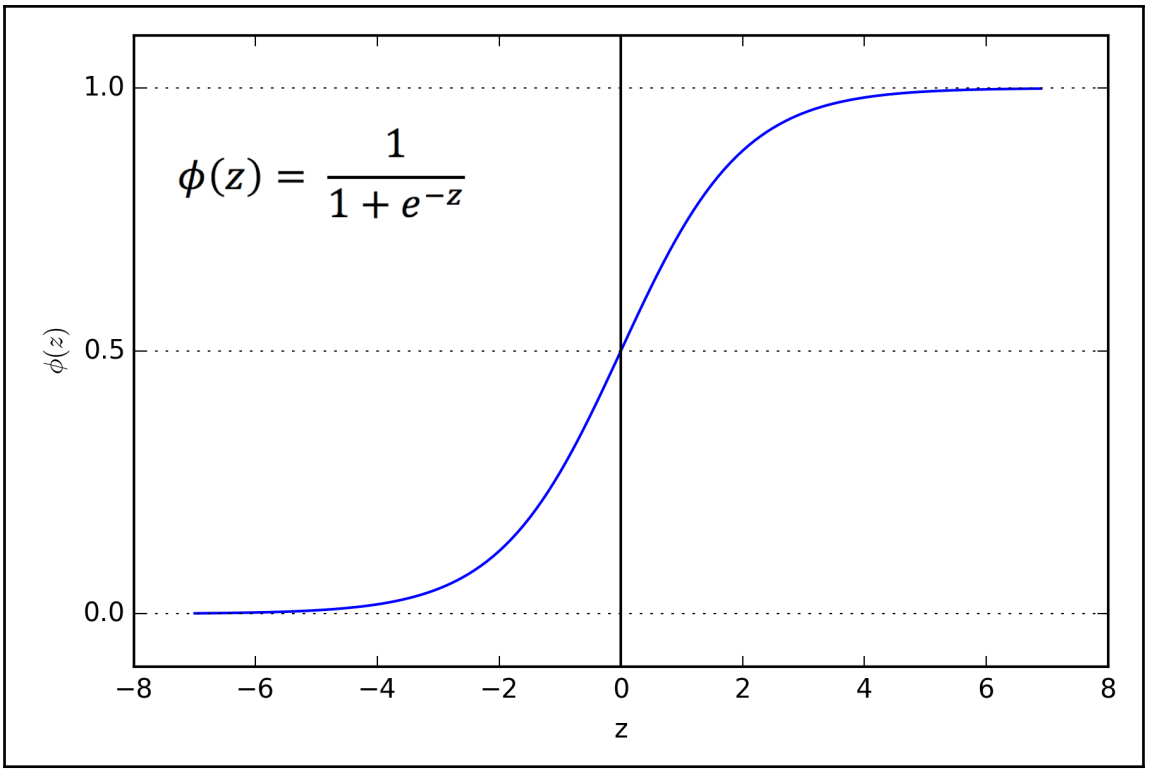

cluster 1
cluster 2
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

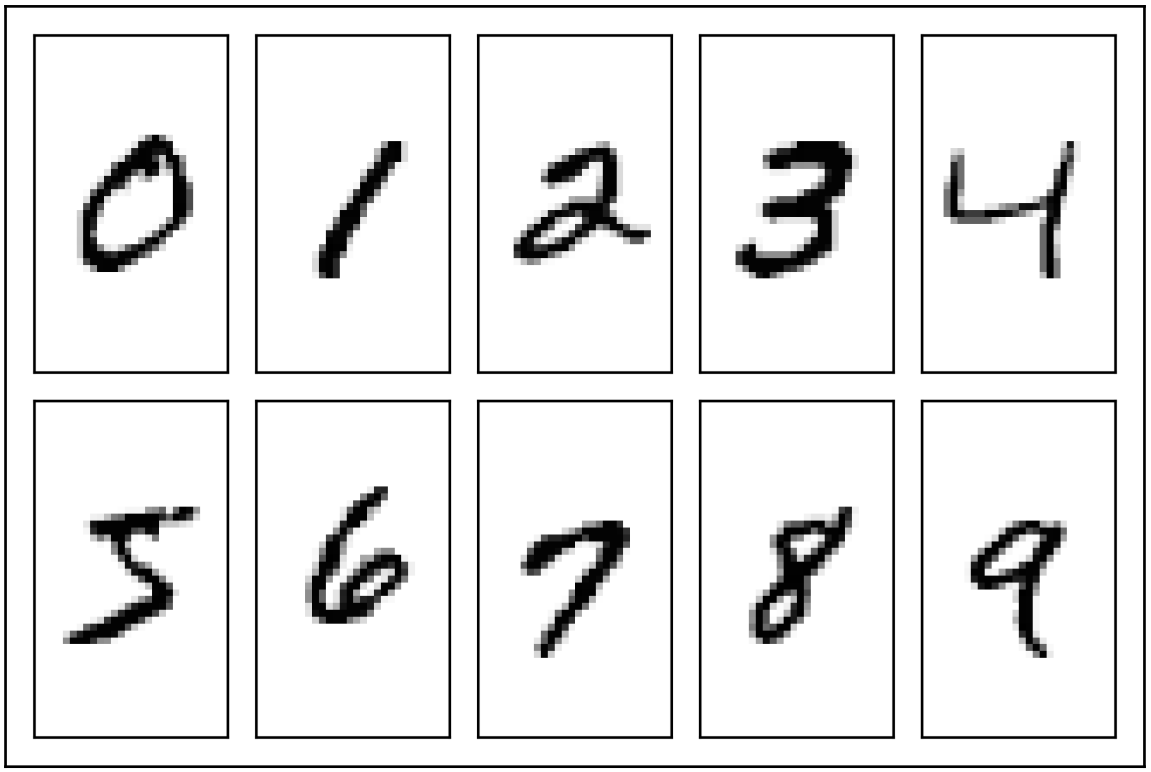
Chapter 12: Implementing a Multilayer Artificial Neural Network from Scratch

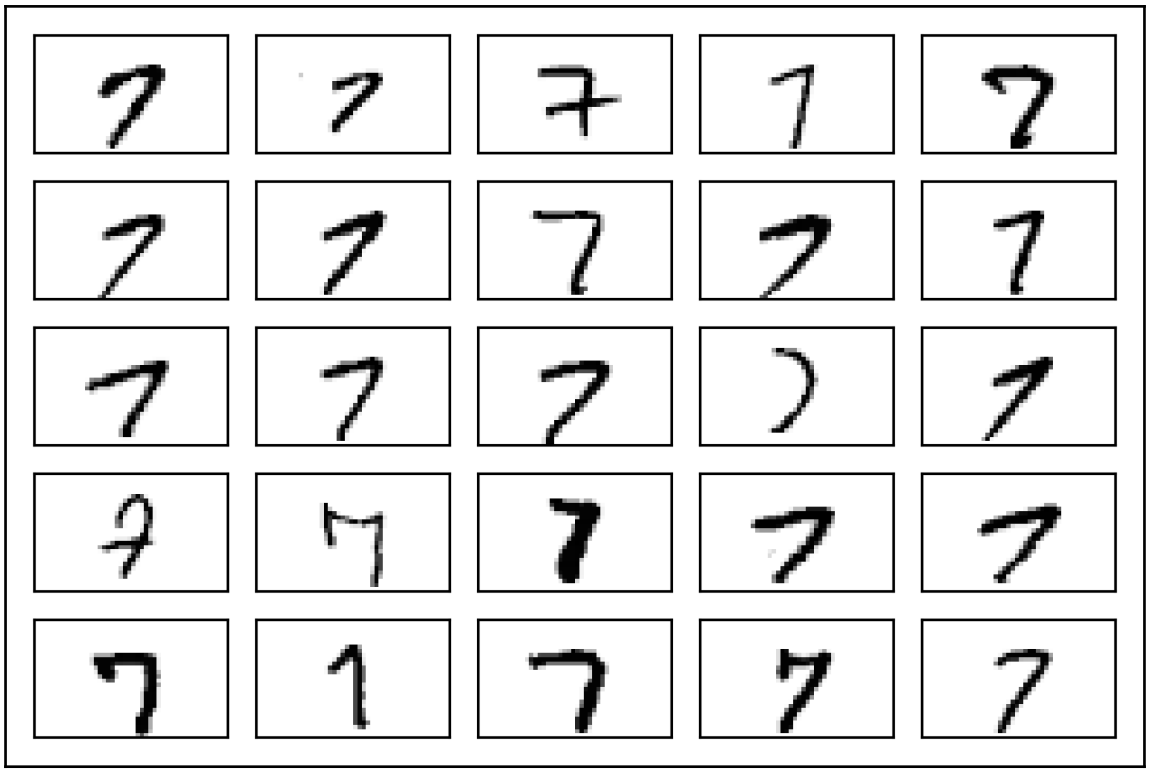


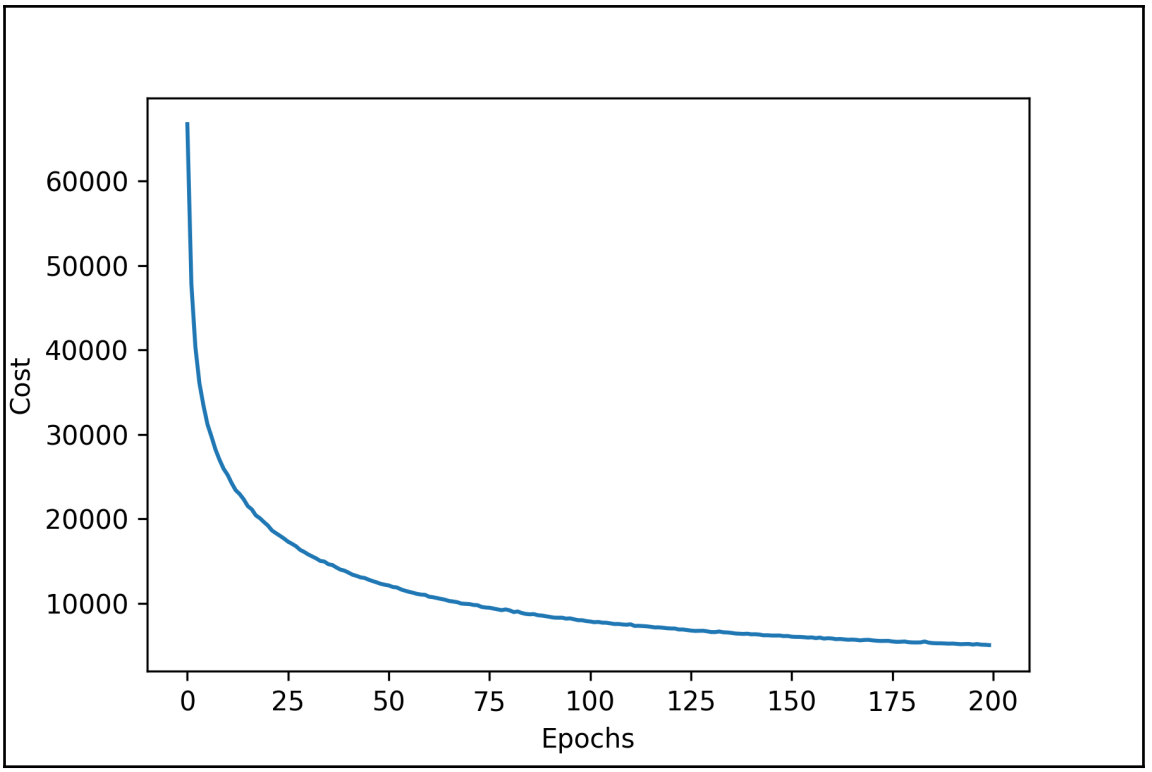


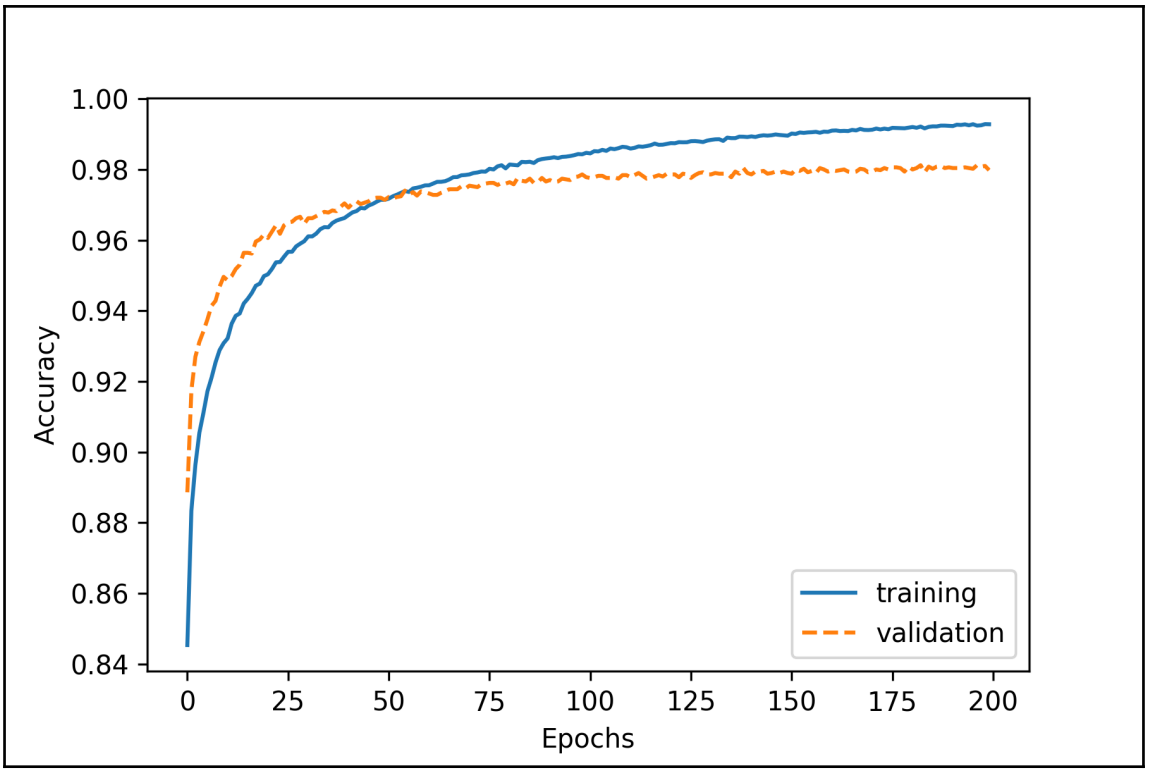

Input layer with 3 input units plus bias unit (m = 3+1)
Hidden layer with 4 hidden units plus bias unit (d = 4+1)
Output layer with 3 output units (t = 3)
1
1
$w^{(out)}_{1,3}$
connects 1st non-bias neuron in the 2nd layer (hidden layer h) to the 3rd unit in the 3rd layer (output layer out)
Number of layers: L = 3


$\phi(z) = \frac{1}{1 + e^{-z}}$
1.0
0.5
0.0
$\phi(z)$
−8
−6
−4
−2
0
2
4
6
8
z






60000
50000
40000
30000
20000
10000
Cost
0
25
50
75
100
125
150
175
200
Epochs


1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
Accuracy
0
25
50
75
100
125
150
175
200
Epochs
training
validation

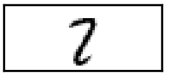
1) t: 5 p: 6

2) t: 4 p: 9

3) t: 4 p: 2

4) t: 6 p: 0

5) t: 2 p: 7



6) t: 5 p: 3

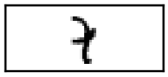
7) t: 3 p: 7

8) t: 6 p: 0

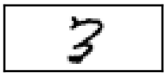
9) t: 3 p: 5

10) t: 8 p: 0

11) t: 7 p: 1



12) t: 3 p: 7

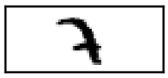


13) t: 1 p: 8

14) t: 2 p: 6

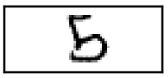
15) t: 2 p: 8

16) t: 7 p: 3

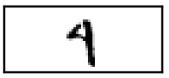


17) t: 8 p: 4

18) t: 5 p: 8



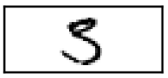
19) t: 4 p: 9



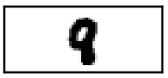
20) t: 9 p: 7

21) t: 2 p: 7

22) t: 3 p: 5

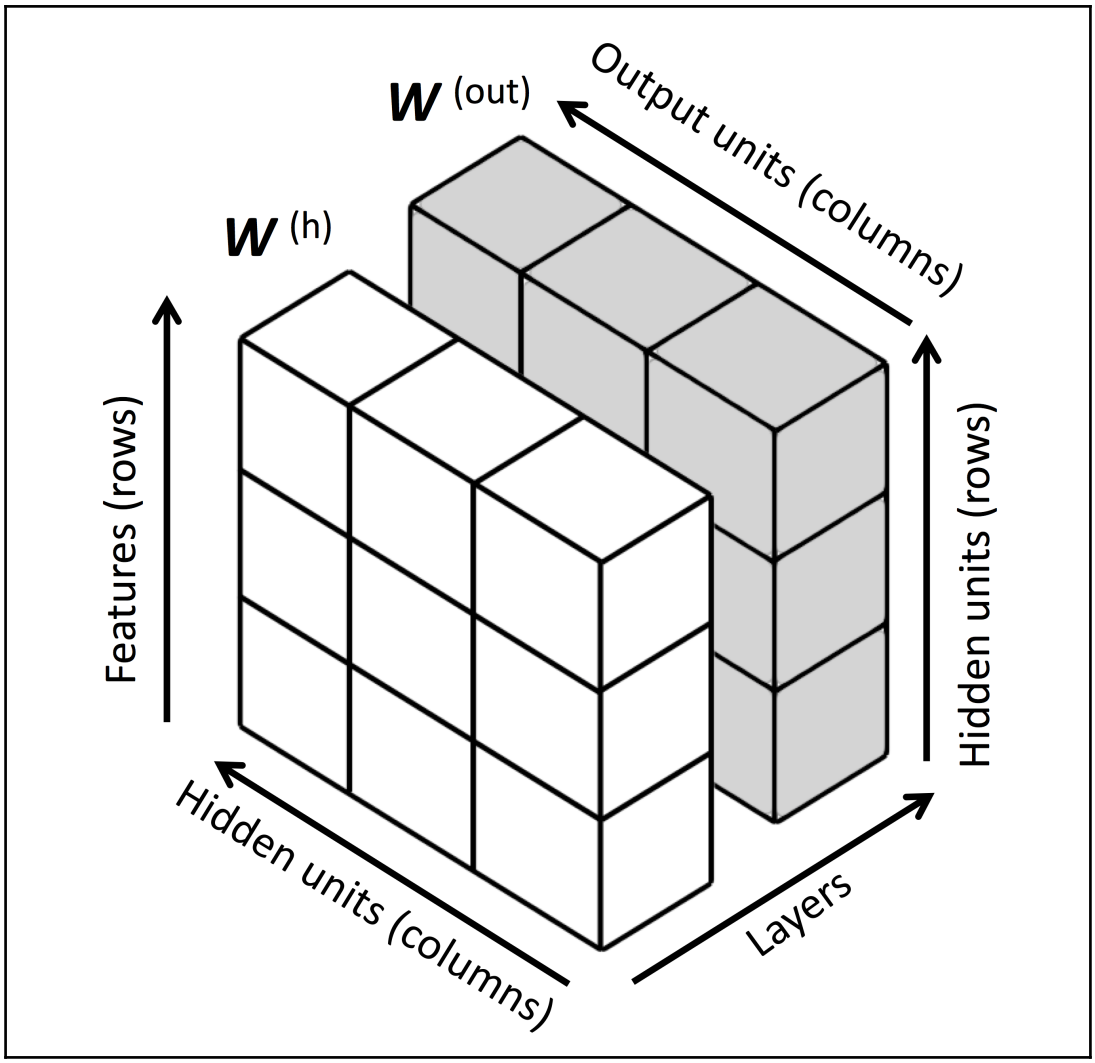


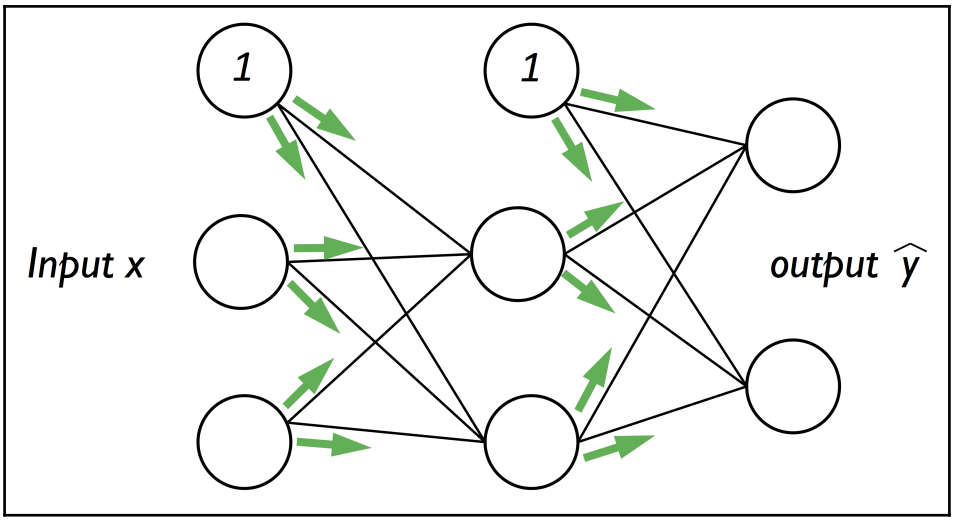
23) t: 8 p: 9

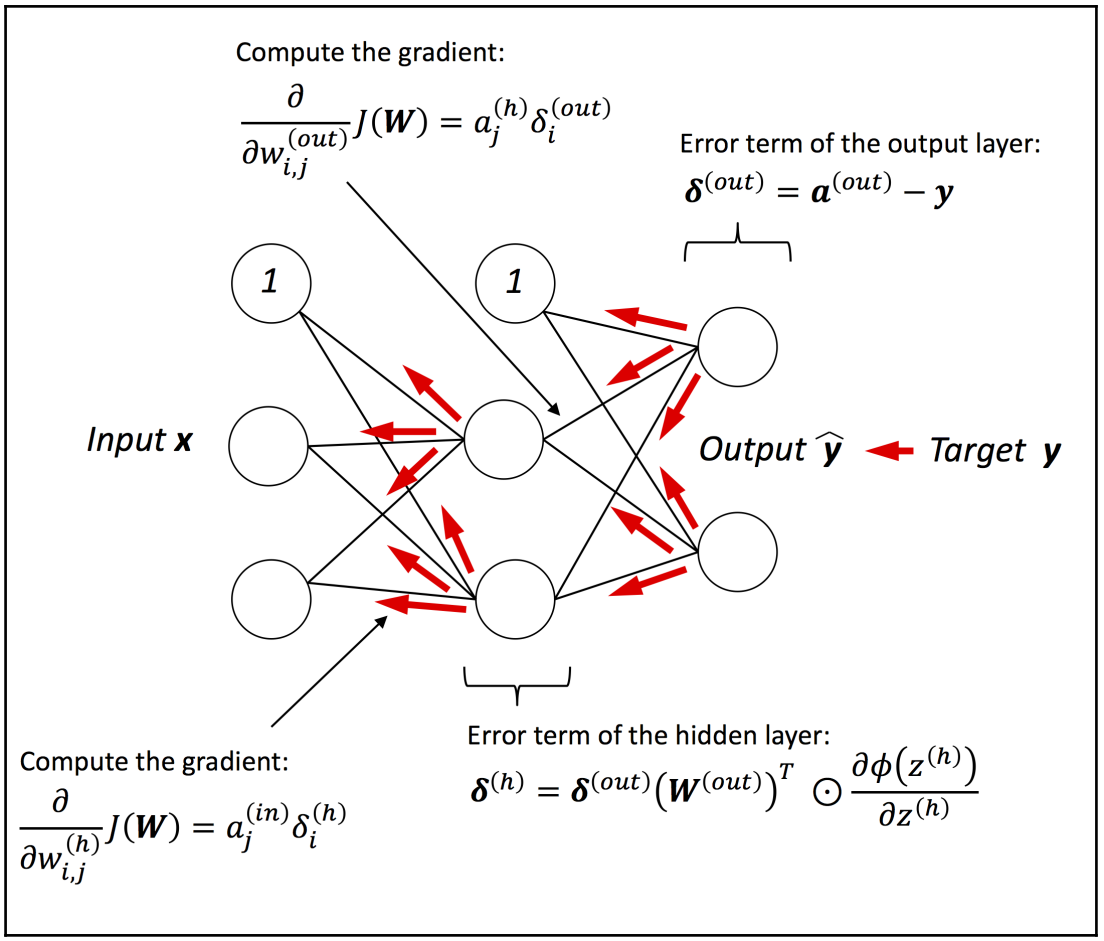


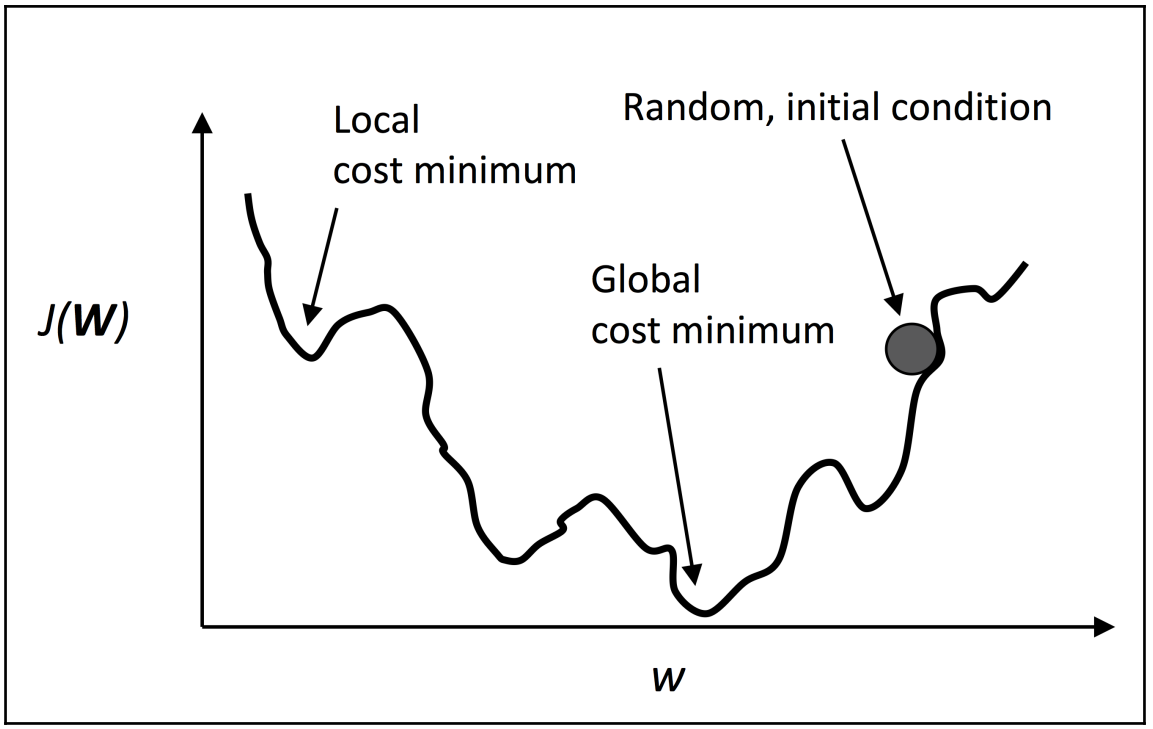
24) t: 5 p: 4

25) t: 1 p: 2


W (out)
Output units (columns)
W (h)
Features (rows)
Hidden units (rows)
Hidden units (columns)
Layers

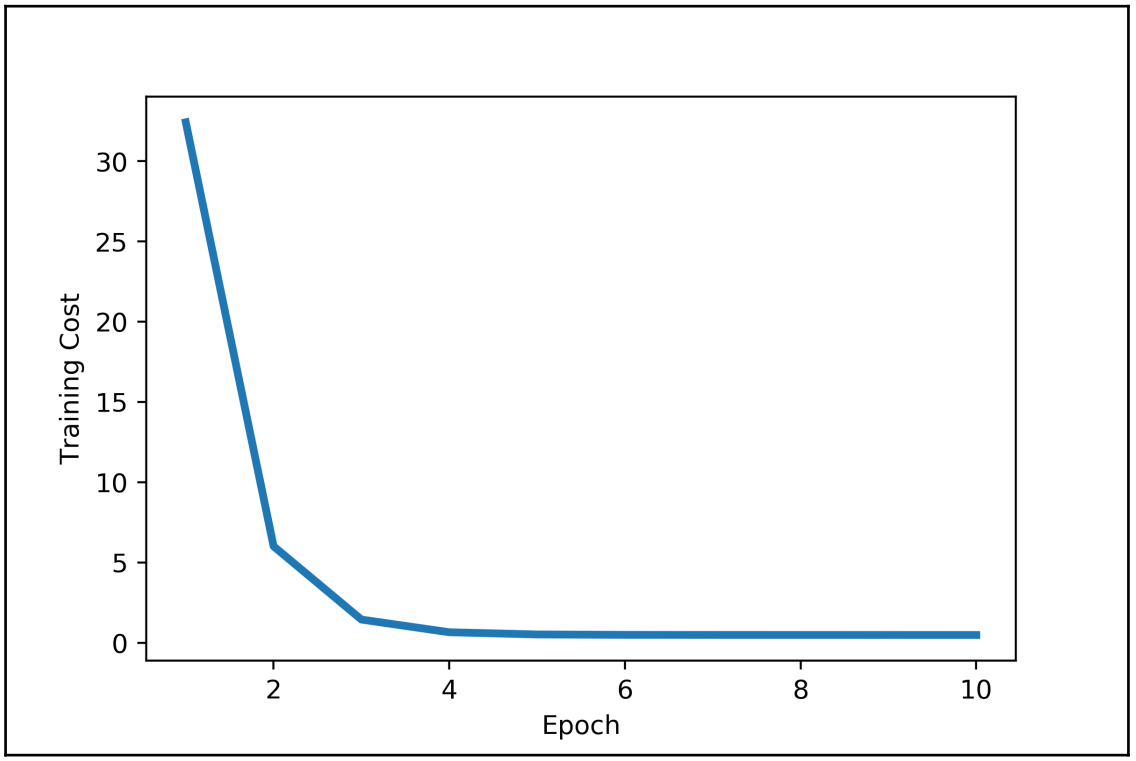

1
1
Input x
output $\hat{y}$

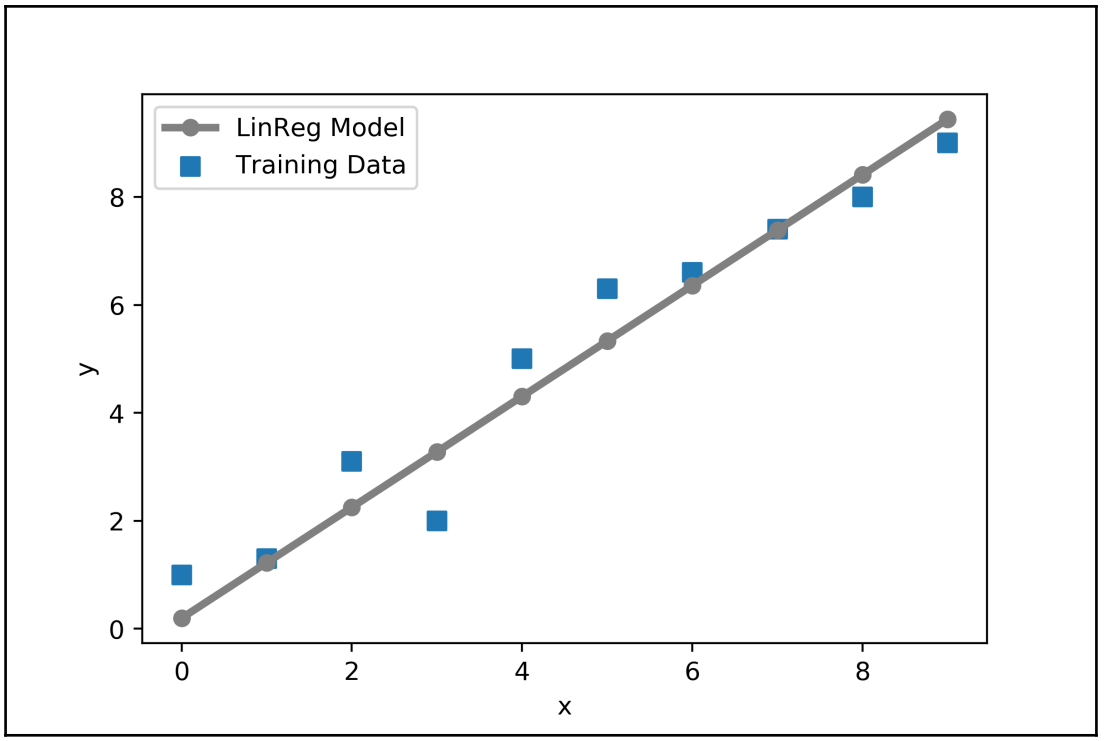

Compute the gradient:
$\frac{\partial}{\partial w_{i,j}^{(out)}} J(\boldsymbol{W}) = a_j^{(h)} \delta_i^{(out)}$
Error term of the output layer:
$\boldsymbol{\delta}^{(out)} = \boldsymbol{a}^{(out)} - \boldsymbol{y}$
1
1
Input $\boldsymbol{x}$
Output $\hat{\boldsymbol{y}}$
Target $\boldsymbol{y}$
Error term of the hidden layer:
$\boldsymbol{\delta}^{(h)} = \boldsymbol{\delta}^{(out)} \left(\boldsymbol{W}^{(out)}\right)^T \odot \frac{\partial \phi\left(z^{(h)}\right)}{\partial z^{(h)}}$
Compute the gradient:
$\frac{\partial}{\partial w_{i,j}^{(h)}} J(\boldsymbol{W}) = a_j^{(in)} \delta_i^{(h)}$

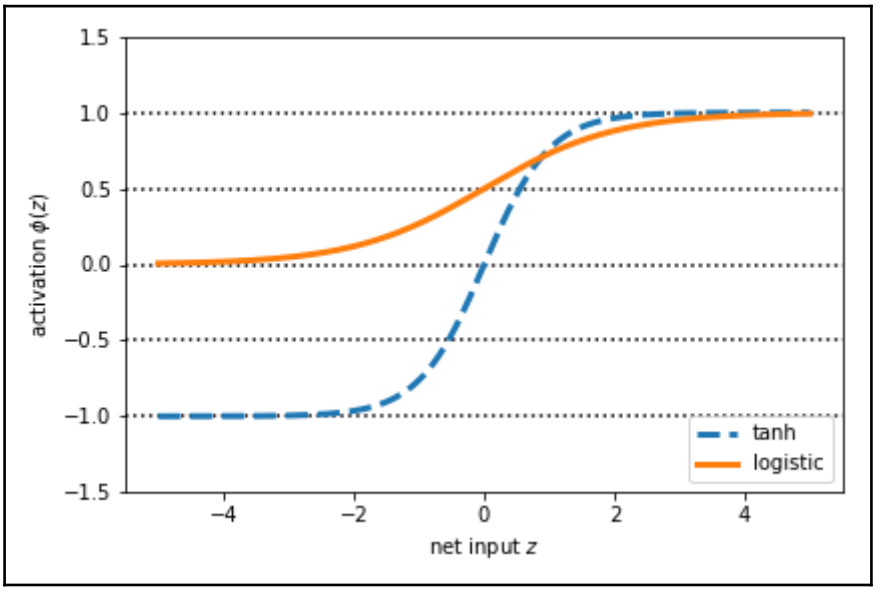

Local
cost minimum
Random, initial condition
Global
cost minimum
J(W)
w

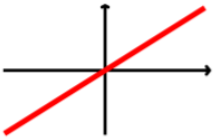
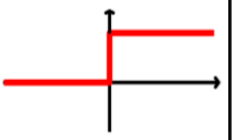
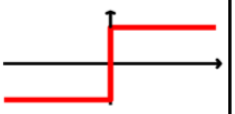
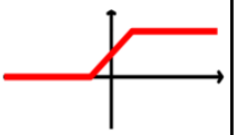
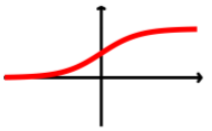
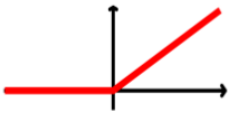
Chapter 13: Parallelizing Neural Network Training with TensorFlow

Specifications	Intel® Core™ i7-6900K Processor Extreme Ed.	NVIDIA GeForce® GTX™ 1080 Ti
Base Clock Frequency	3.2 GHz	< 1.5 GHz
Cores	8	3584
Memory Bandwidth	64 GB/s	484 GB/s
Floating-Point Calculations	409 GFLOPS	11300 GFLOPS
Cost	~ $1000.00	~ $700.00

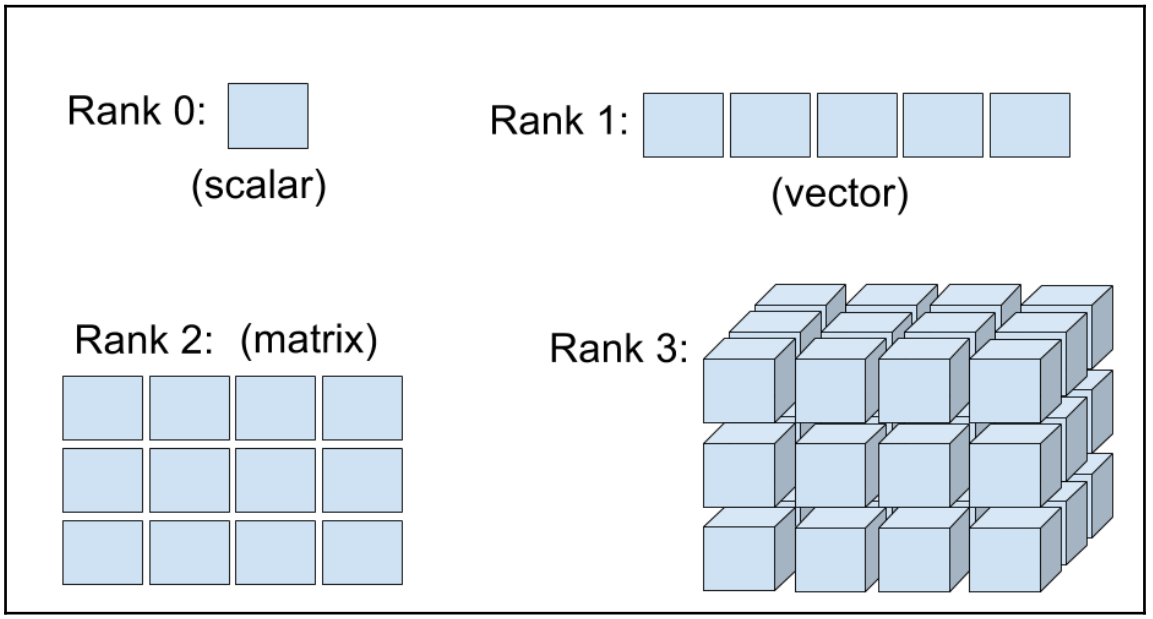


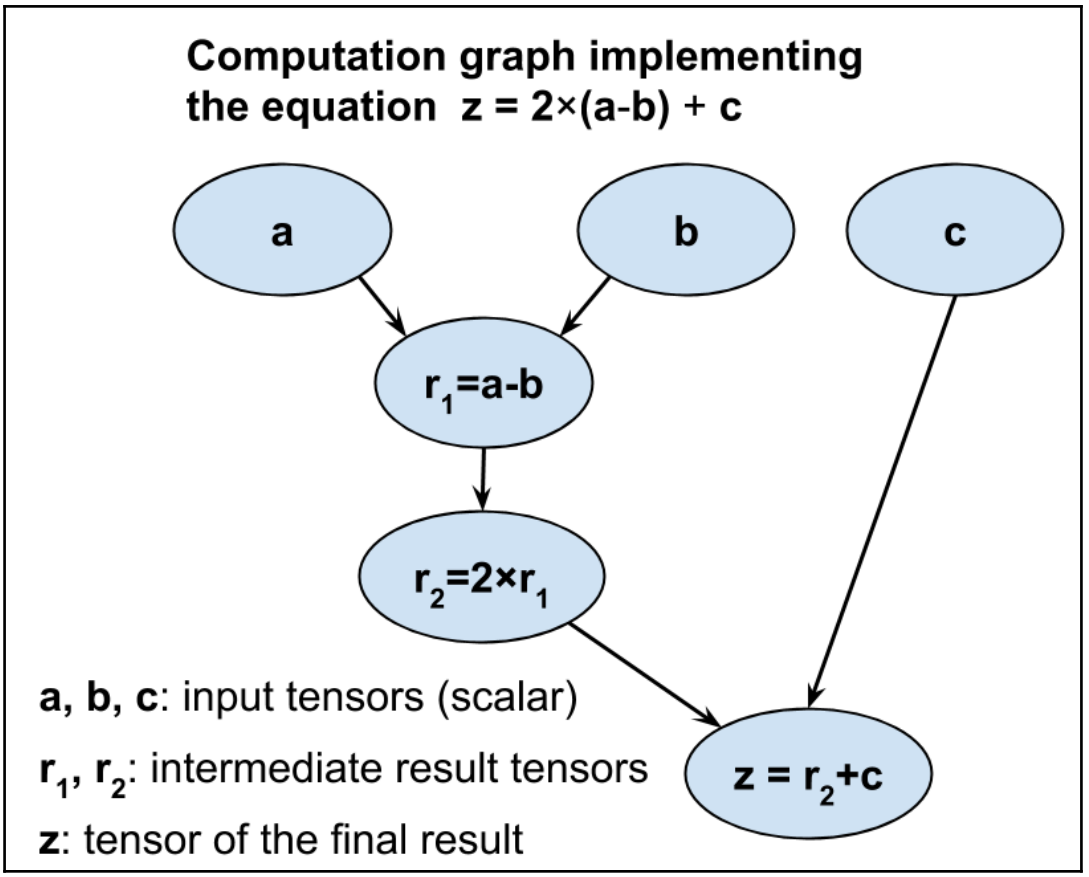

LinReg Model
Training Data
y
x
0
2
4
6
8

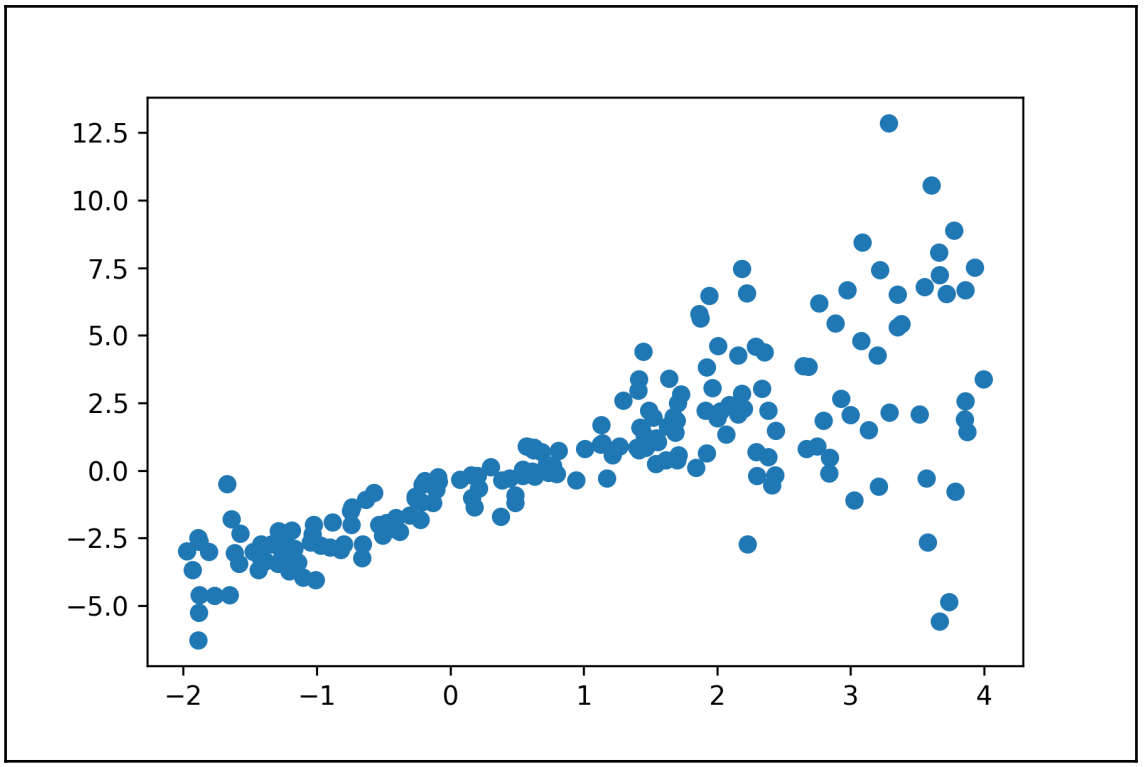

activation ϕ(z)
net input z
tanh
logistic
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−4
−2
0
2
4

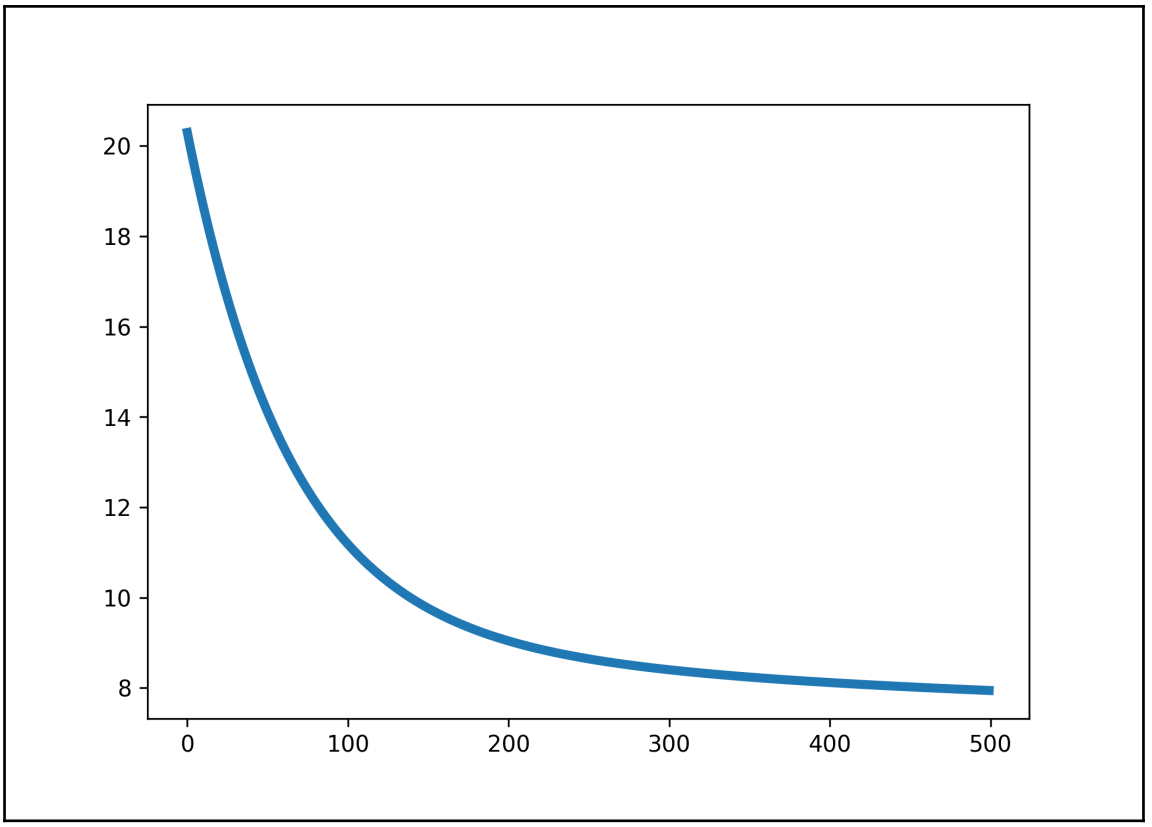
Activation Function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Unit Step (Heaviside Function)	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\phi(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Piece-wise Linear	$\phi(z) = \begin{cases} 0 & z \leq -½ \\ z + ½ & -½ \leq z \leq ½ \\ 1 & z \geq ½ \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multilayer NN	
Hyperbolic Tangent (tanh)	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multilayer NN, RNNs	
ReLU	$\phi(z) = \begin{cases} 0 & z < 0 \\ z & z > 0 \end{cases}$	Multilayer NN, CNNs	

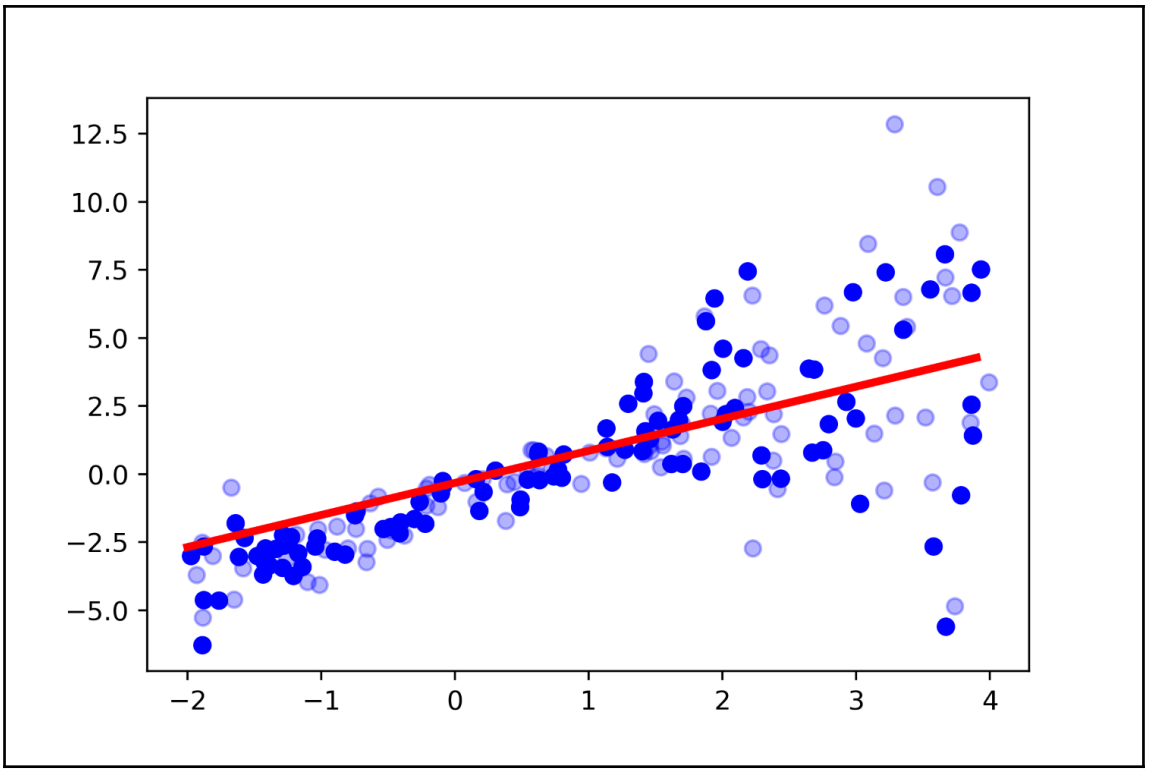
Chapter 14: Going Deeper – The Mechanics of TensorFlow

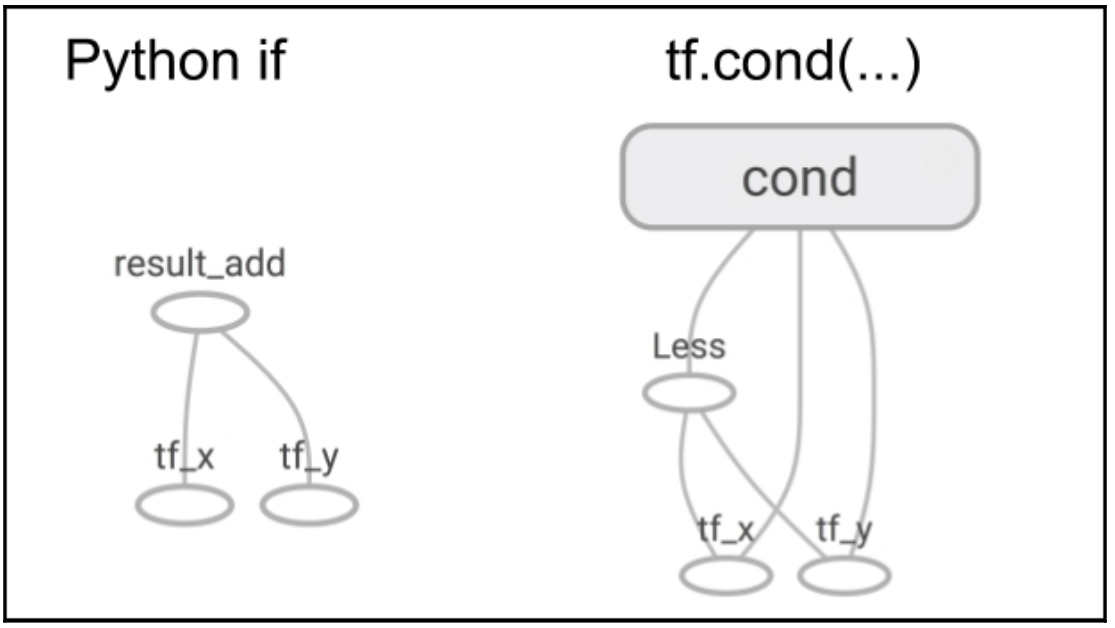


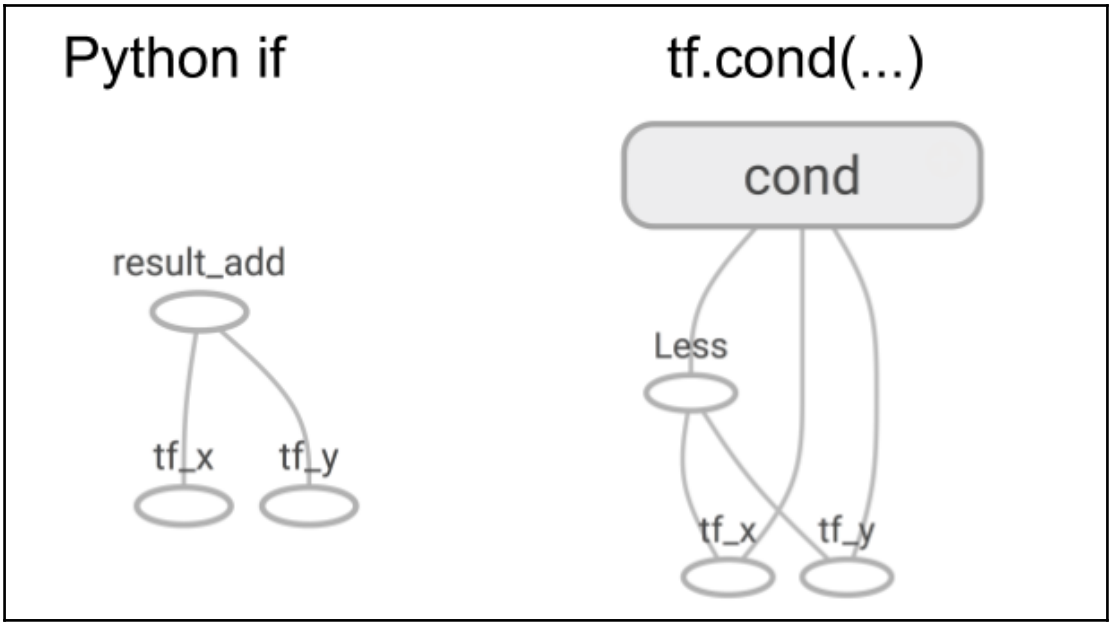

Computation graph implementing
the equation z = 2×(a-b) + c
a
b
c
r1=a-b
r2=2×r1
z = r2+c
a, b, c: input tensors (scalar)
r1, r2: intermediate result tensors
z: tensor of the final result

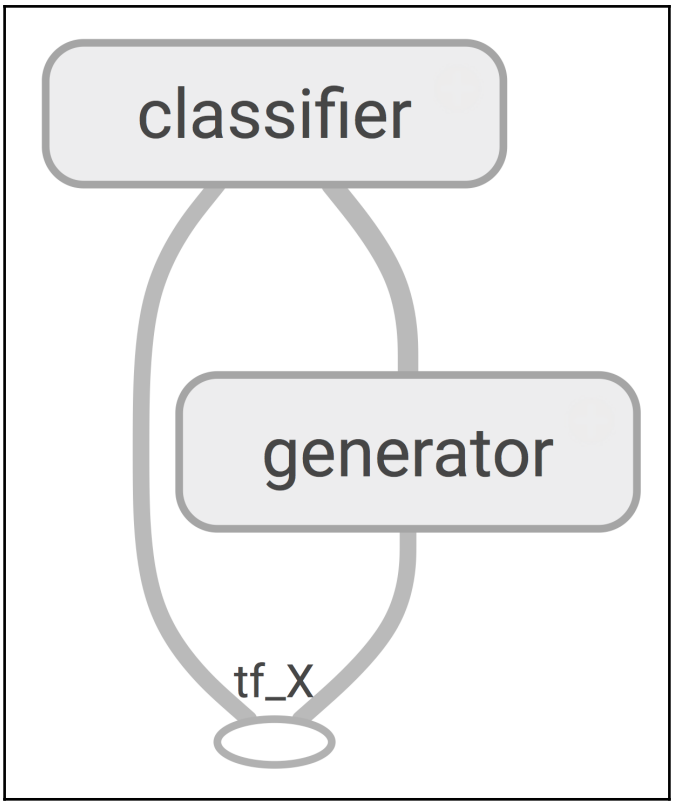

12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−2
−1
0
1
2
3
4

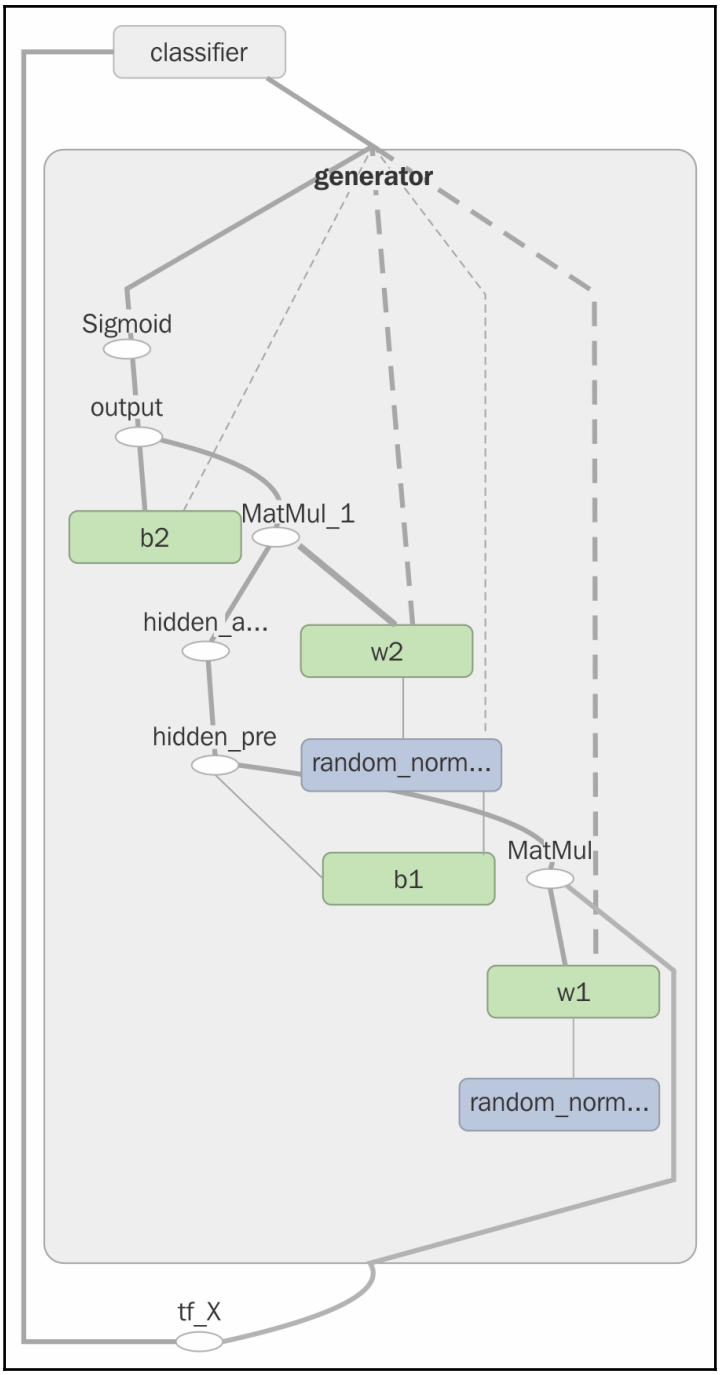

20
18
16
14
12
10
8
0
100
200
300
400
500

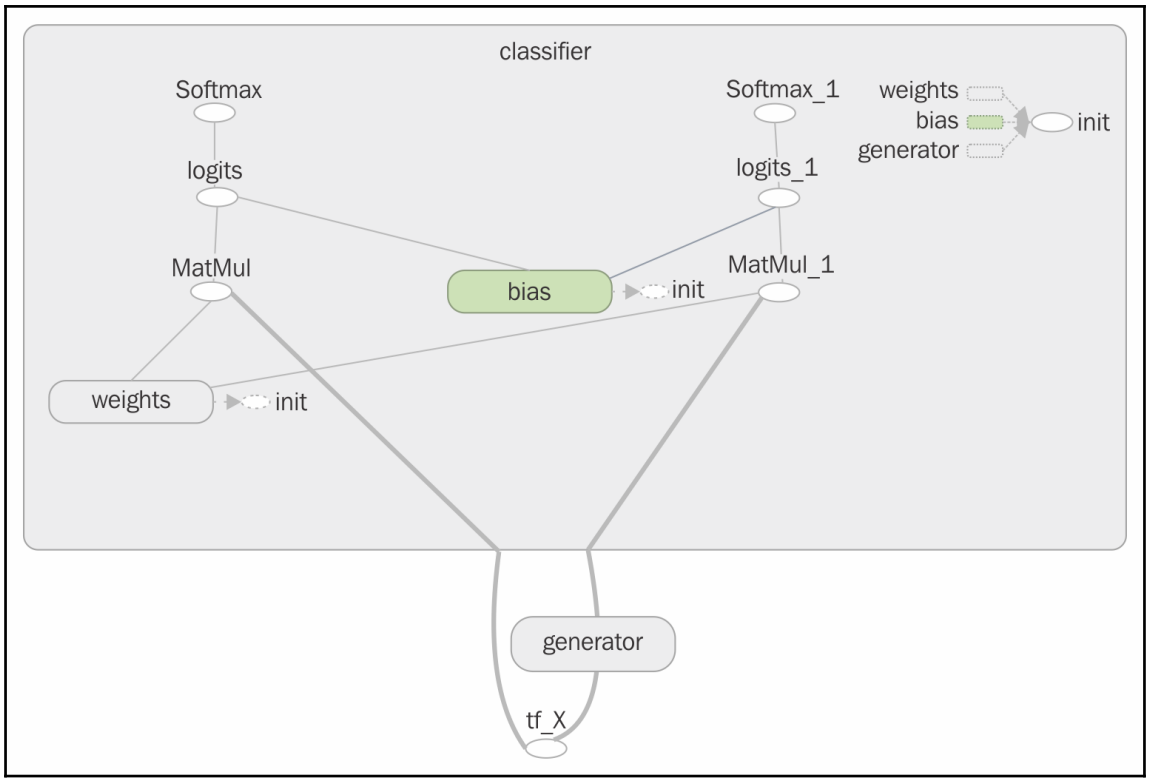

12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−2
−1
0
1
2
3
4


Python if
tf.cond(...)
result_add
tf_x
tf_y
cond
Less
tf_x
tf_y

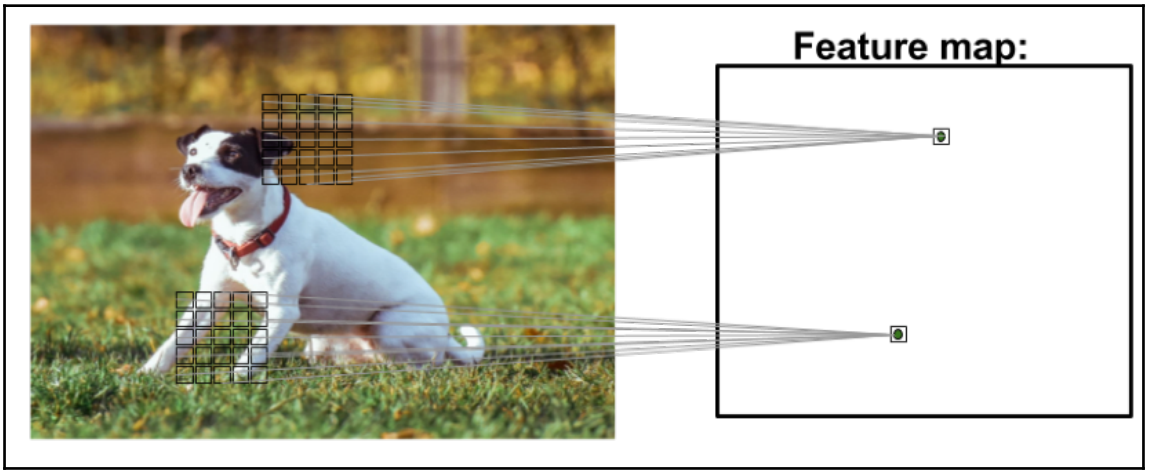

Python if
tf.cond(...)
result_add
tf_x
tf_y
cond
Less
tf_x
tf_y

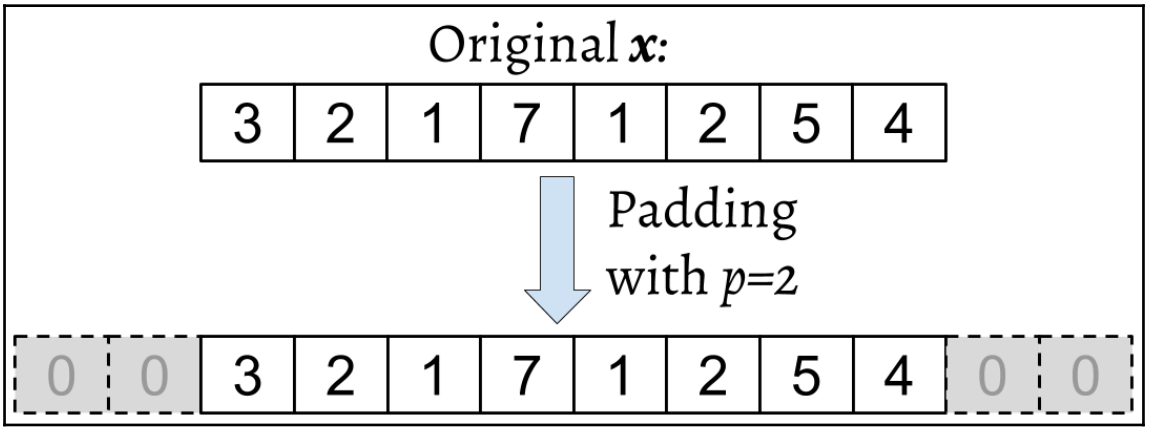

classifier
generator
tf_X

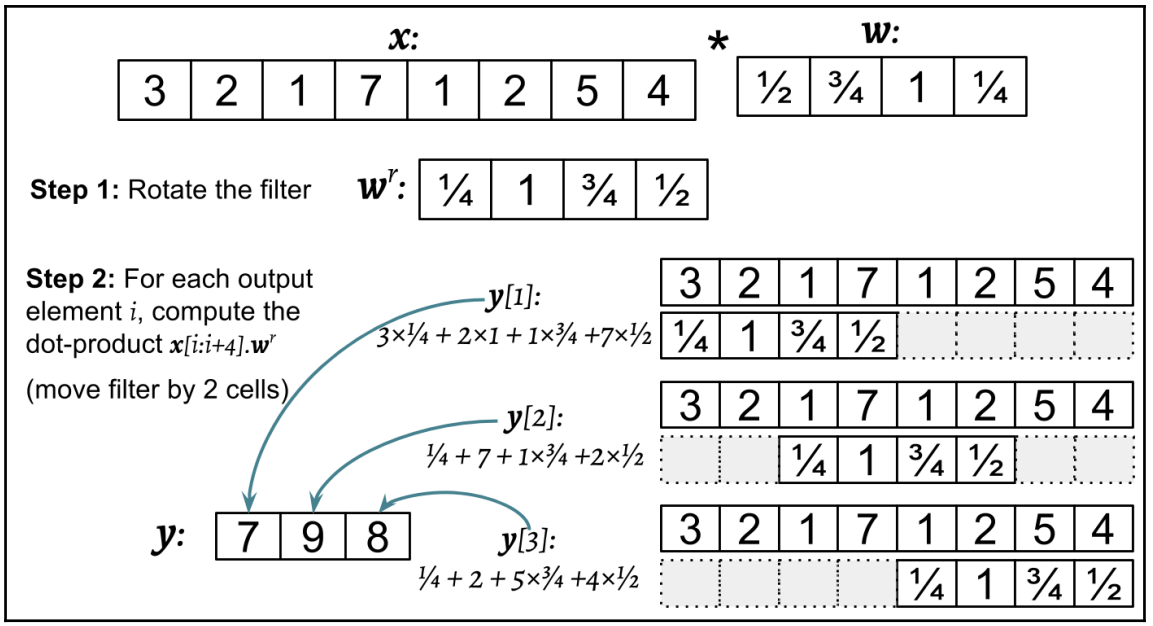

classifier
generator
Sigmoid
output
b2
MatMul_1
hidden_a...
w2
hidden_pre
random_norm...
b1
MatMul
w1
random_norm...
tf_X

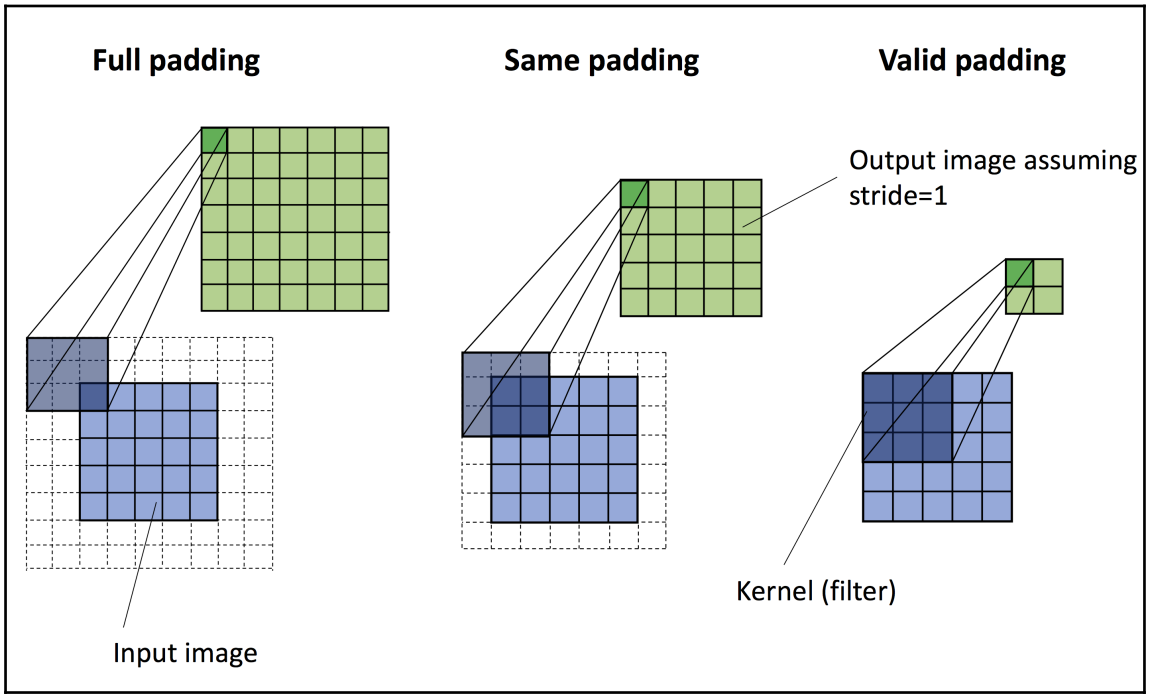

classifier
Softmax
logits
MatMul
weights
init
bias
init
Softmax_1
logits_1
MatMul_1
weights
bias
generator
init
generator
tf_X

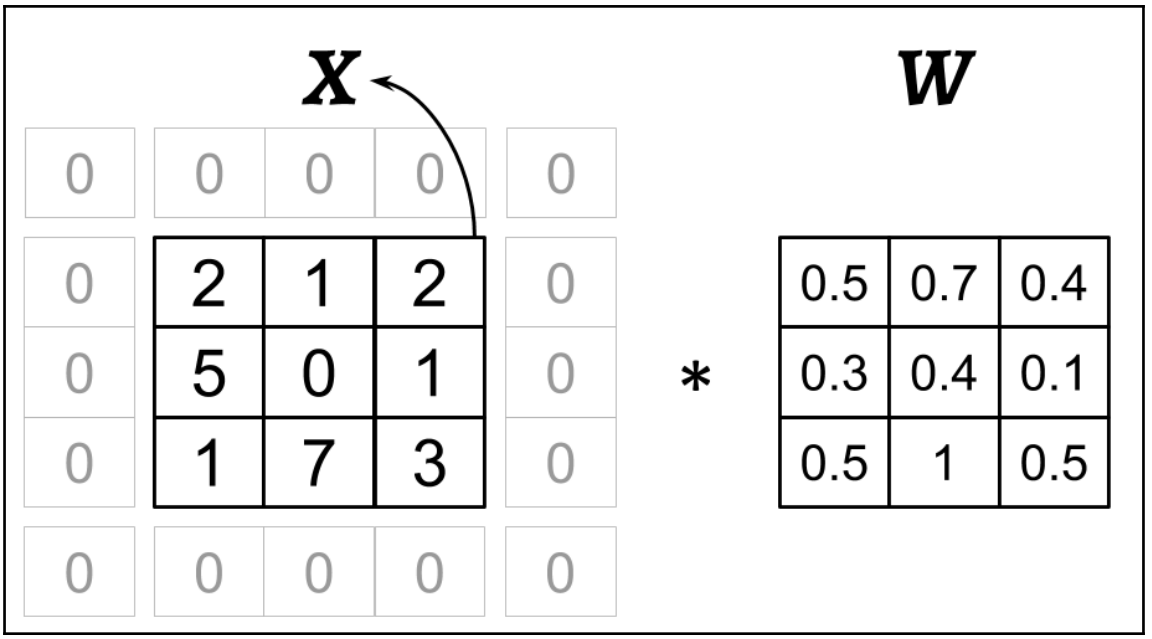
Chapter 15: Classifying Images with Deep Convolutional Neural Networks

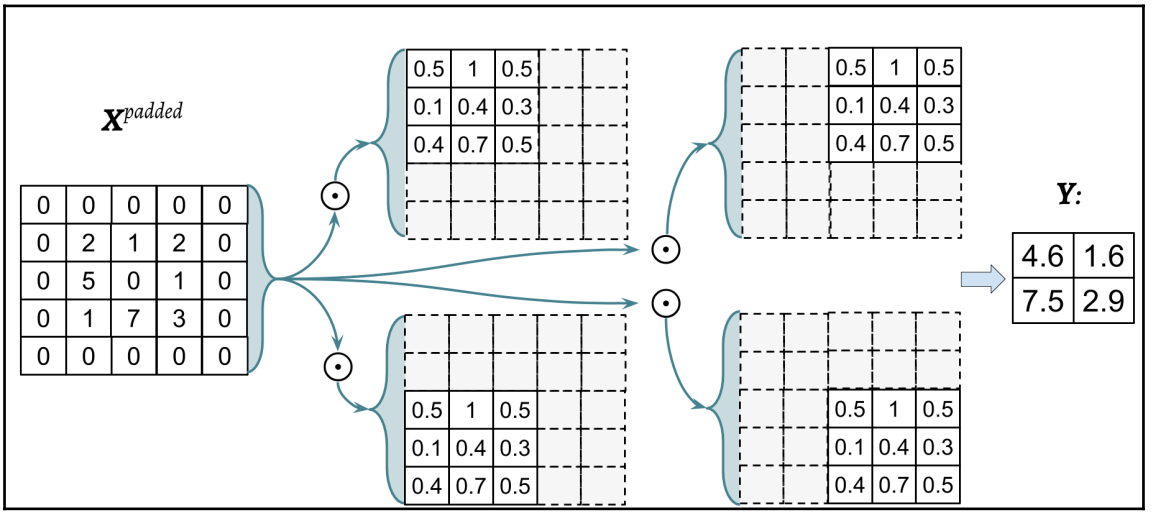


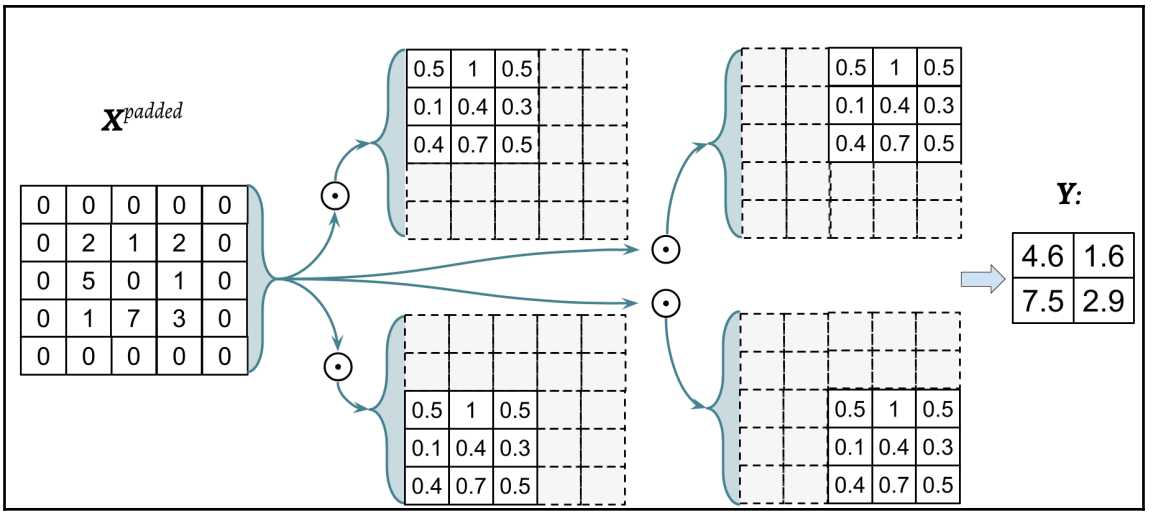



x:
3 2 1 7 1 2 5 4
*
w:
½ ¾ 1 ¼
Step 1: Rotate the filter
w^r:
¼ 1 ¾ ½
Step 2: For each output element i, compute the dot-product x[i:i+4].w^r
(move filter by 2 cells)
y[1]:
3×¼ + 2×1 + 1×¾ +7×½
y[2]:
¼ + 7 + 1×¾ +2×½
y[3]:
¼ + 2 + 5×¾ +4×½
y:
7 9 8


Full padding
Same padding
Valid padding
Output image assuming stride=1
Kernel (filter)
Input image


X
W
0 0 0 0 0
0 2 1 2 0
0 5 0 1 0
0 1 7 3 0
0 0 0 0 0
*
0.5 0.7 0.4
0.3 0.4 0.1
0.5 1 0.5

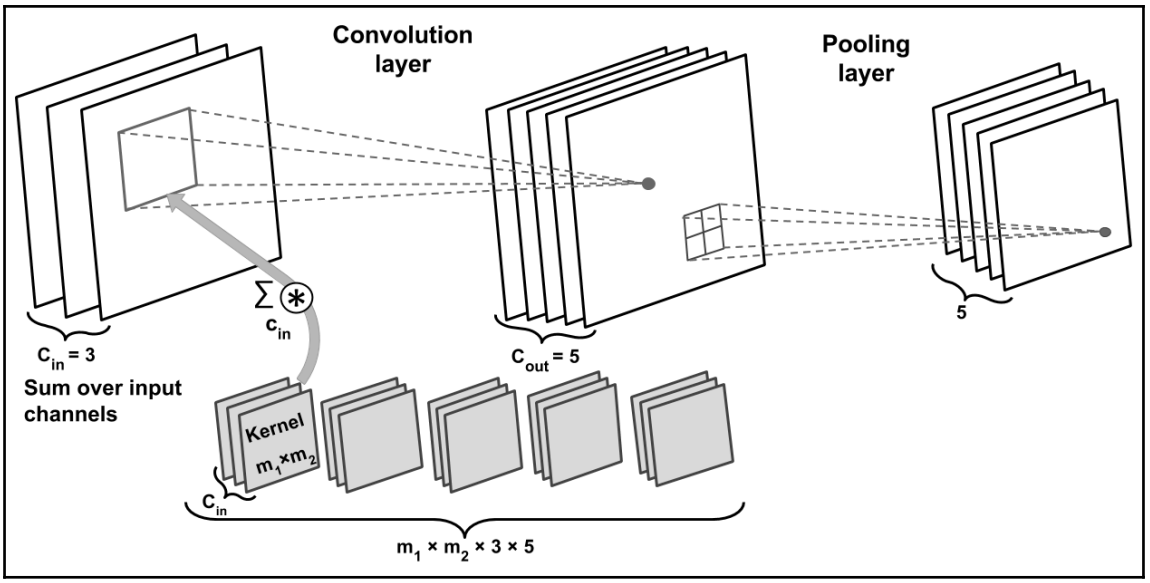

X^padded
0 0 0 0 0
0 2 1 2 0
0 5 0 1 0
0 1 7 3 0
0 0 0 0 0
0.5 1 0.5
0.1 0.4 0.3
0.4 0.7 0.5
Y:
4.6 1.6
7.5 2.9

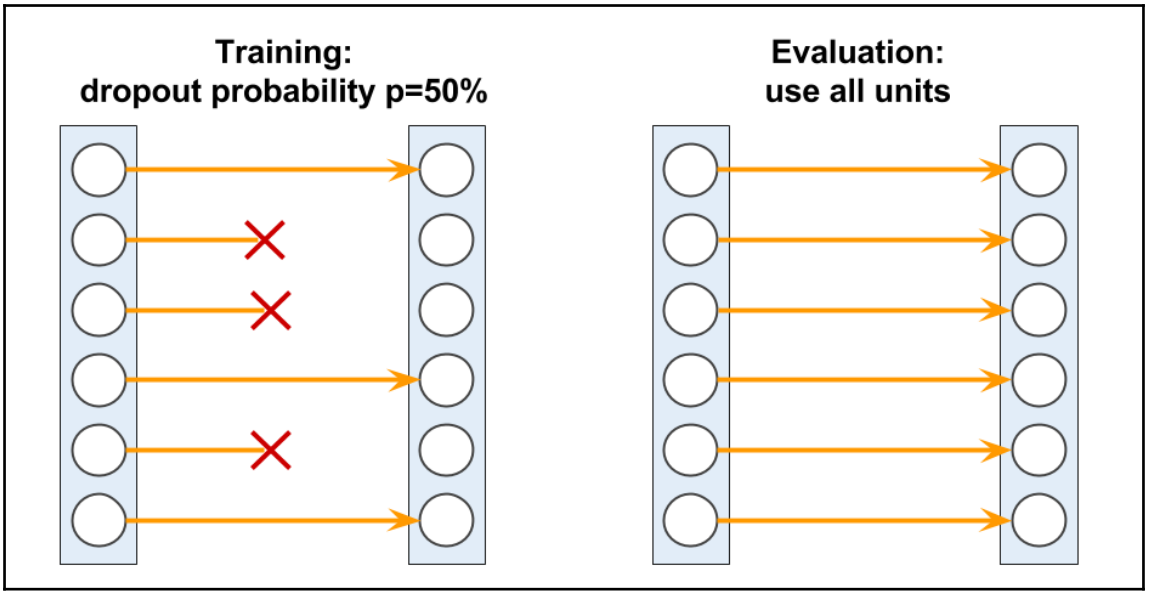


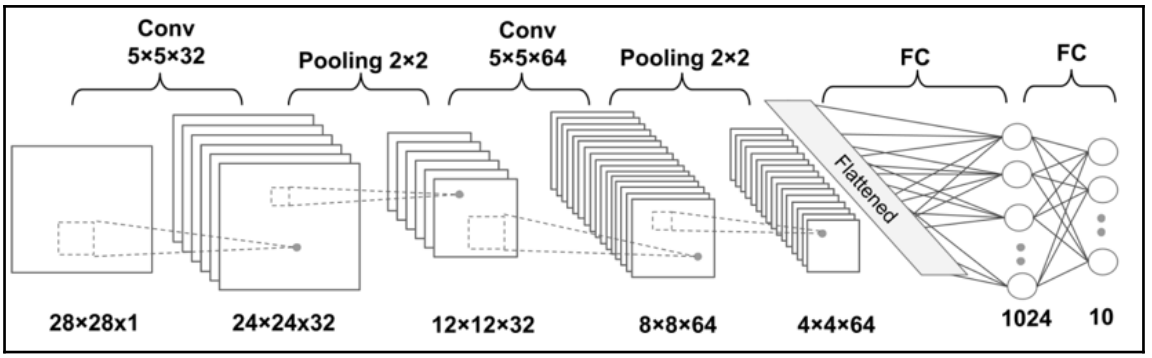
$$X_1 = \begin{bmatrix} 10 & 255 & 125 & 0 & 170 & 100 \\ 70 & 255 & 105 & 25 & 25 & 70 \\ 255 & 0 & 150 & 0 & 10 & 10 \\ 0 & 255 & 10 & 10 & 150 & 20 \\ 70 & 15 & 200 & 100 & 95 & 0 \\ 35 & 25 & 100 & 20 & 0 & 60 \end{bmatrix}$$

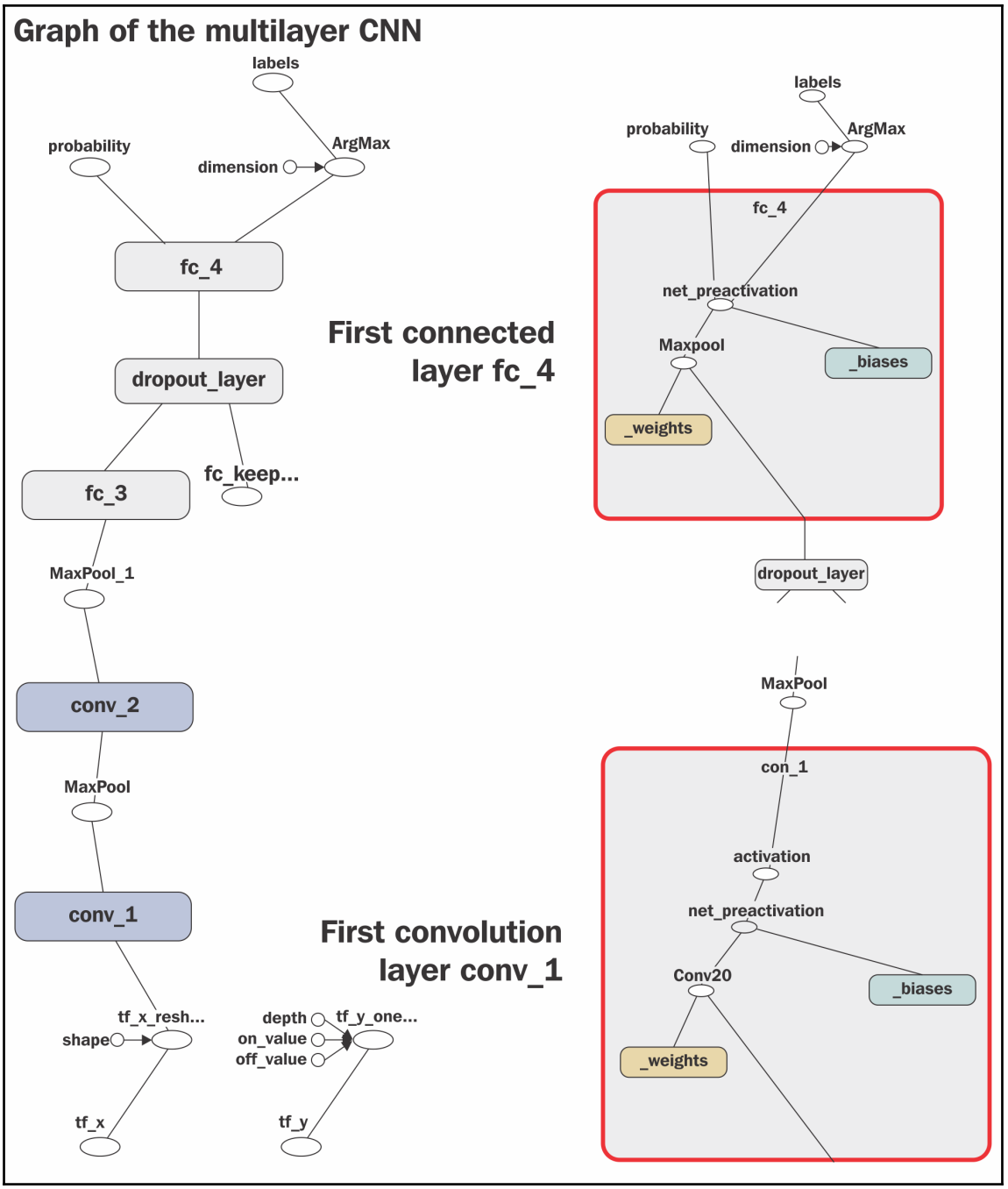
$$X_2 = \begin{bmatrix} 100 & 100 & 100 & 50 & 100 & 50 \\ 95 & 255 & 100 & 125 & 125 & 170 \\ 80 & 40 & 10 & 10 & 125 & 150 \\ 255 & 30 & 150 & 20 & 120 & 125 \\ 30 & 30 & 150 & 100 & 70 & 70 \\ 70 & 30 & 100 & 200 & 70 & 95 \end{bmatrix}$$

$$\xrightarrow{\text{max-pooling } P_{2\times 2}} \begin{bmatrix} 255 & 125 & 170 \\ 255 & 150 & 150 \\ 70 & 200 & 95 \end{bmatrix}$$

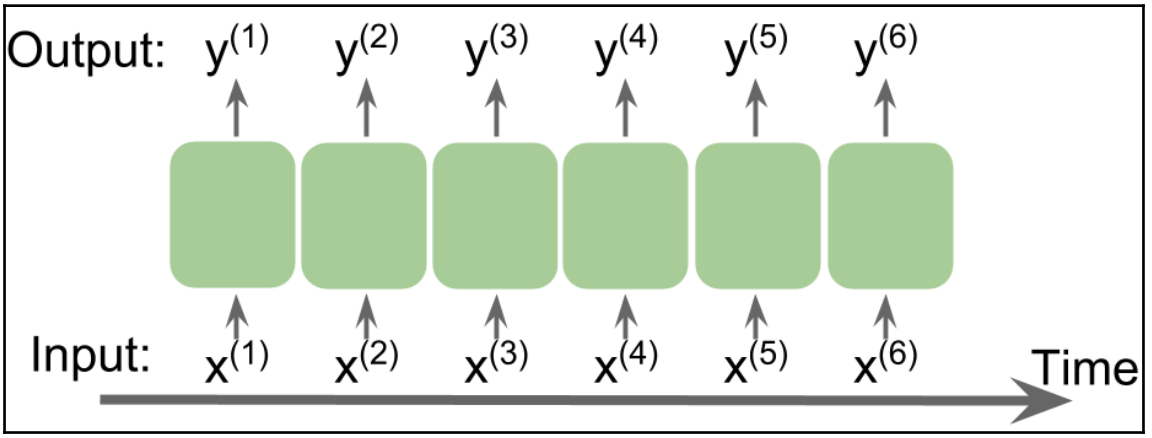

Convolution layer
Pooling layer
$C_{in} = 3$
Sum over input channels
c_{in}
$C_{out} = 5$
5
Kernel
$m_1 \times m_2$
C_{in}
$m_1 \times m_2 \times 3 \times 5$

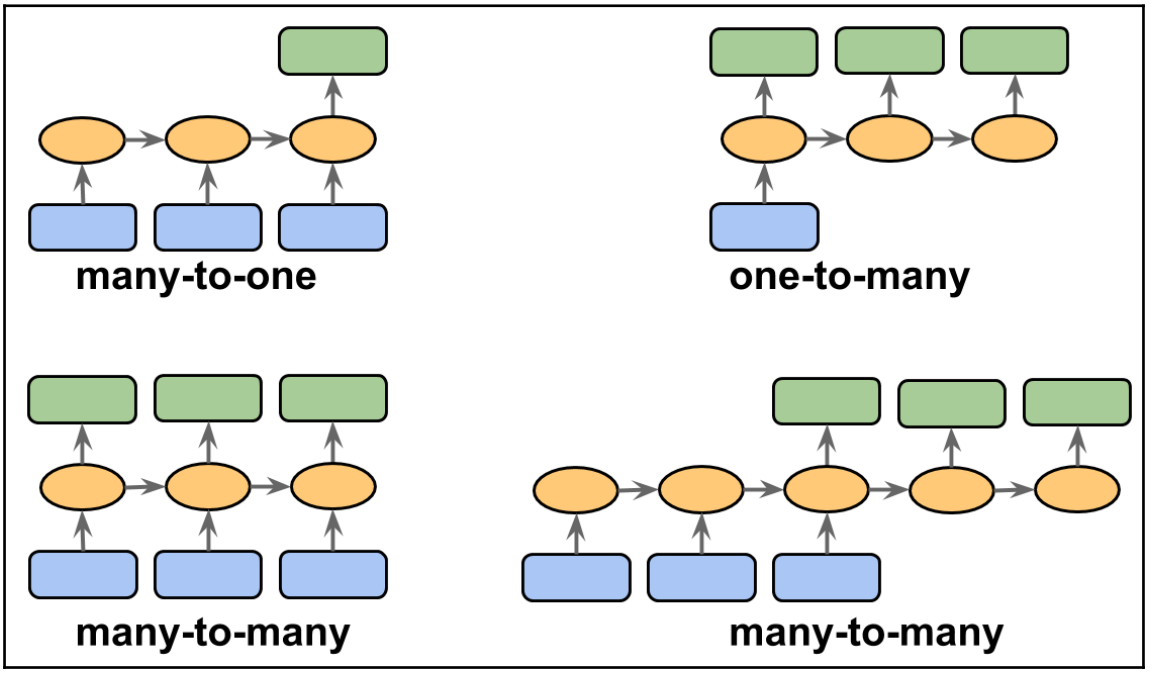

Training:
dropout probability p=50%
Evaluation:
use all units


Conv
5×5×32
Pooling 2×2
Conv
5×5×64
Pooling 2×2
FC
FC
Flattened
28×28x1
24×24x32
12×12×32
8×8×64
4×4×64
1024
10

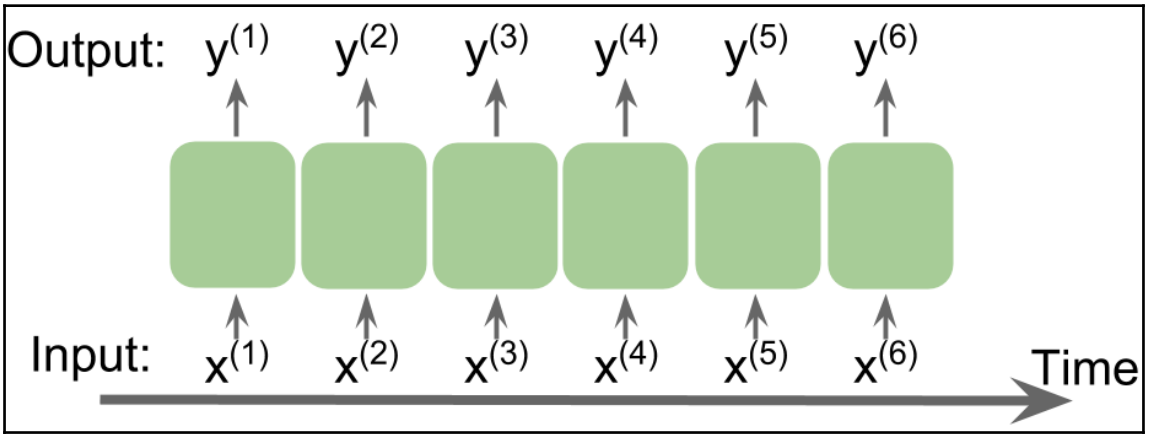

Graph of the multilayer CNN
labels
probability
ArgMax
dimension
fc_4
dropout_layer
fc_3
fc_keep...
MaxPool_1
conv_2
MaxPool
conv_1
tf_x_resh...
shape
tf_x
depth
on_value
off_value
tf_y_one...
tf_y
First connected layer fc_4
labels
probability
ArgMax
dimension
fc_4
net_preactivation
Maxpool
_biases
_weights
dropout_layer
First convolution layer conv_1
MaxPool
con_1
activation
net_preactivation
Conv20
_biases
_weights

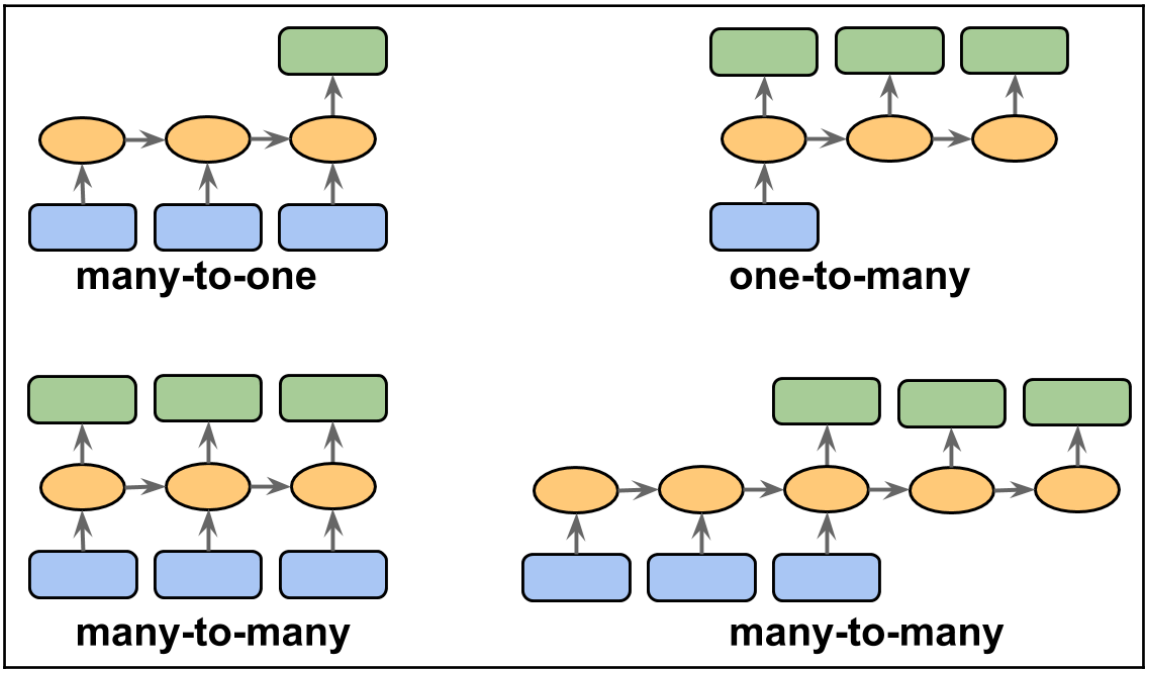
Chapter 16: Modeling Sequential Data Using Recurrent Neural Networks

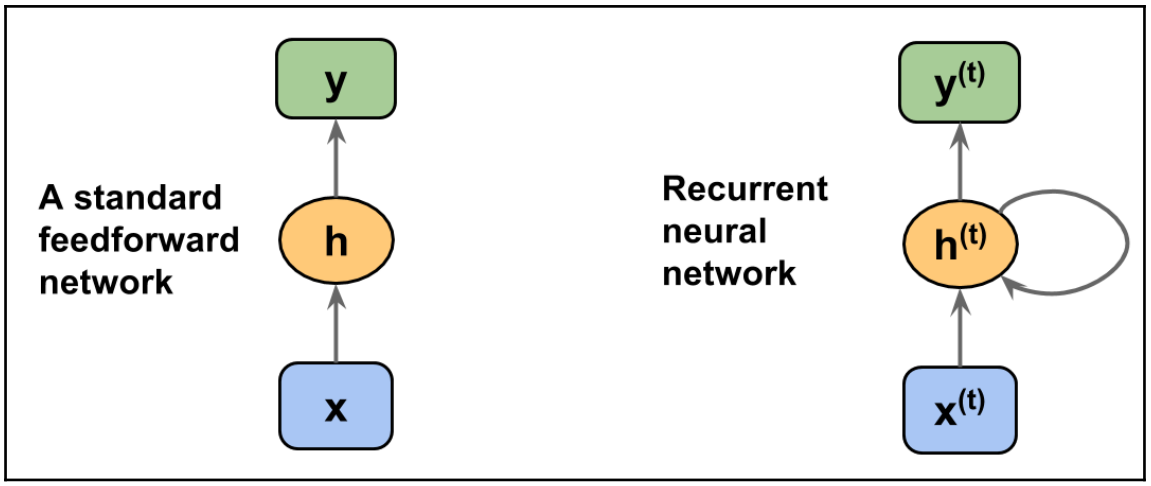


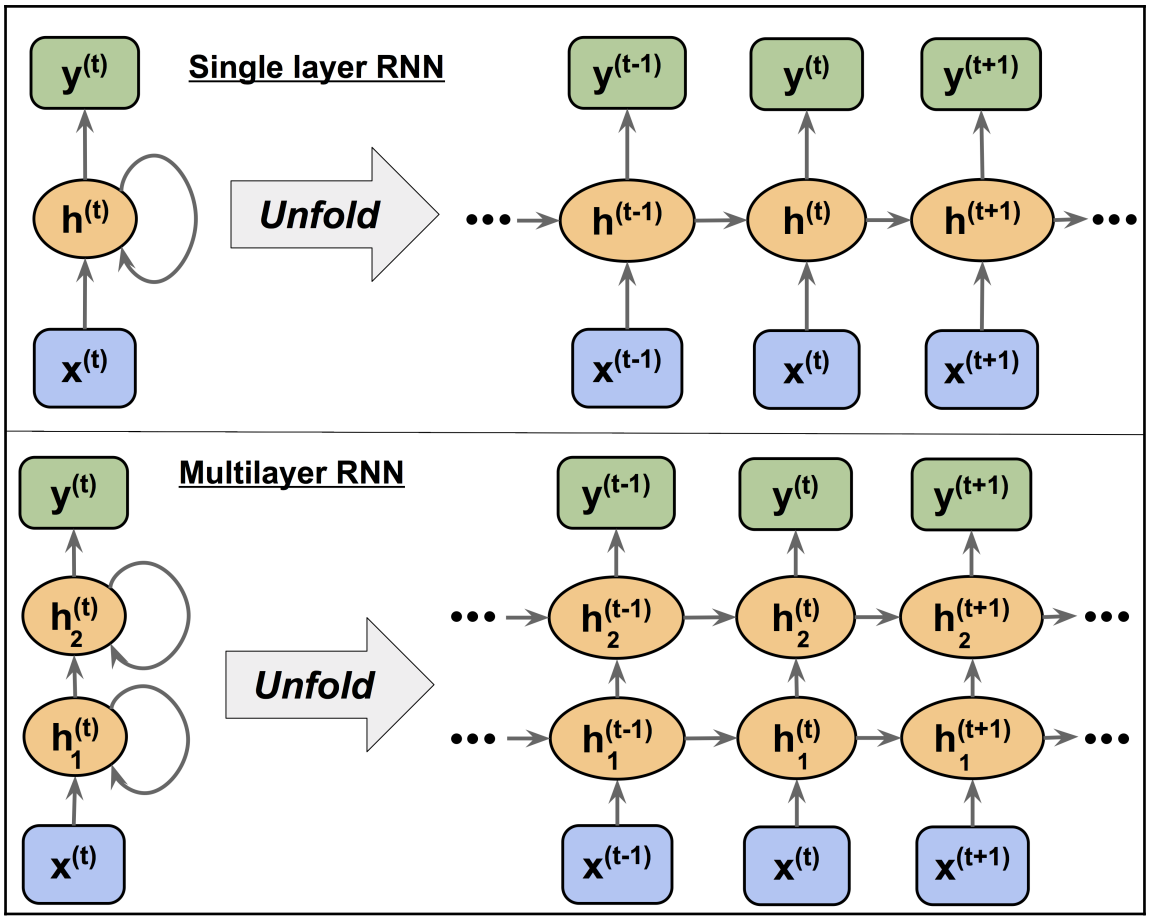


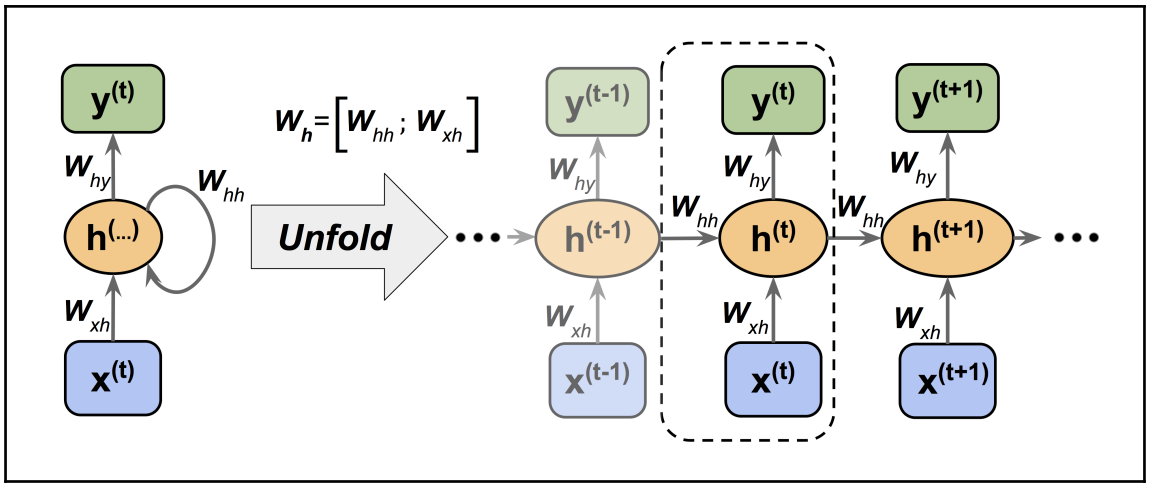
Chapter 16: Modeling Sequential Data Using Recurrent Neural Networks

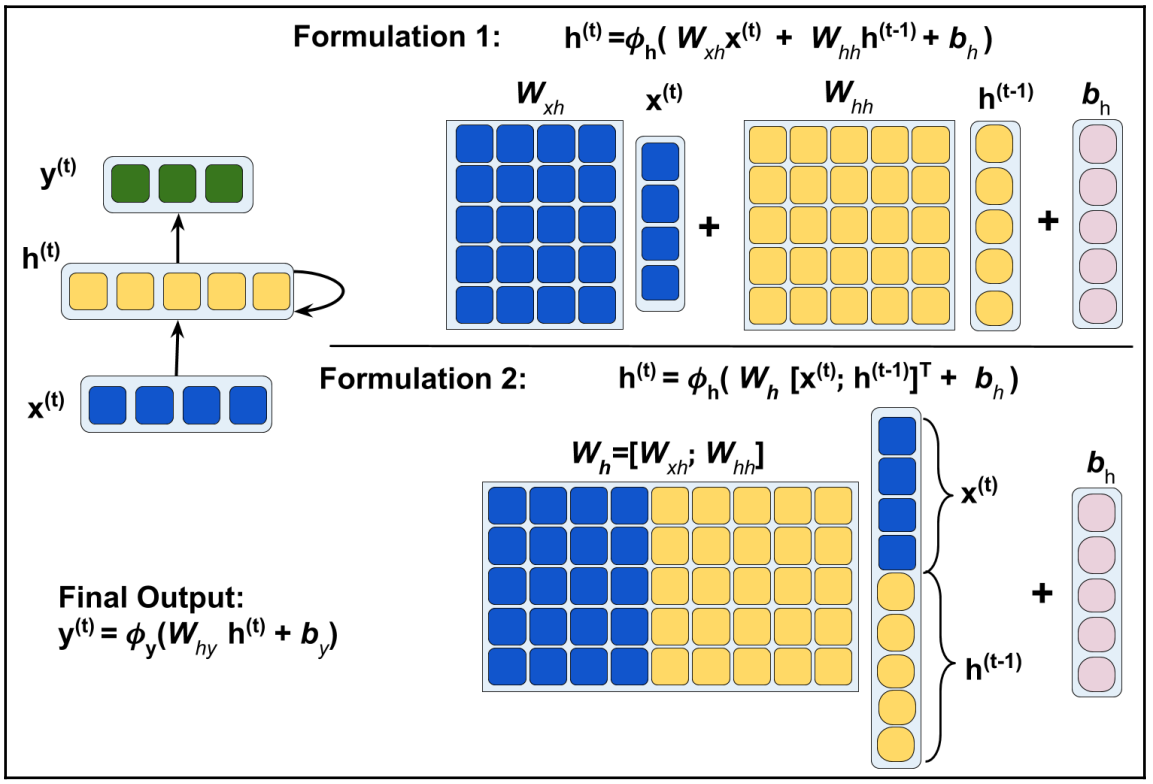


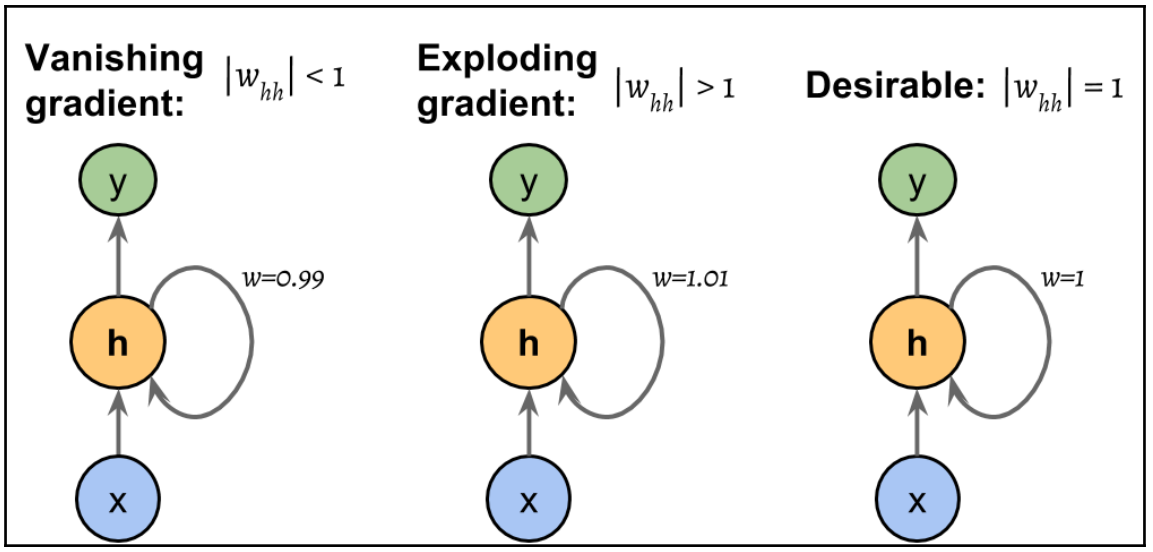


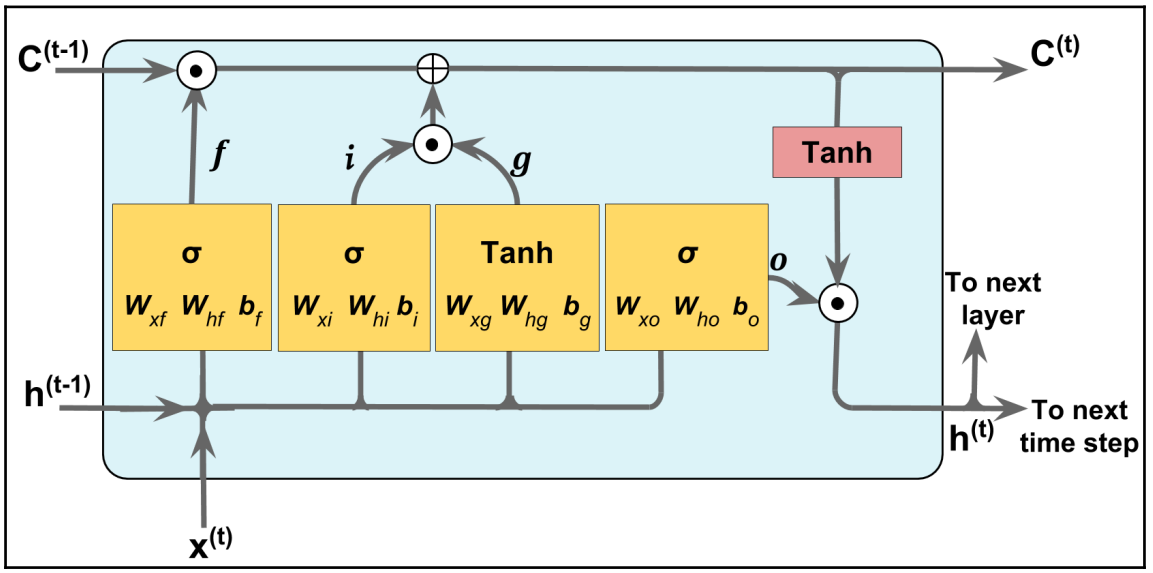

A standard feedforward network
y
h
x
Recurrent neural network
y(t)
h(t)
x(t)

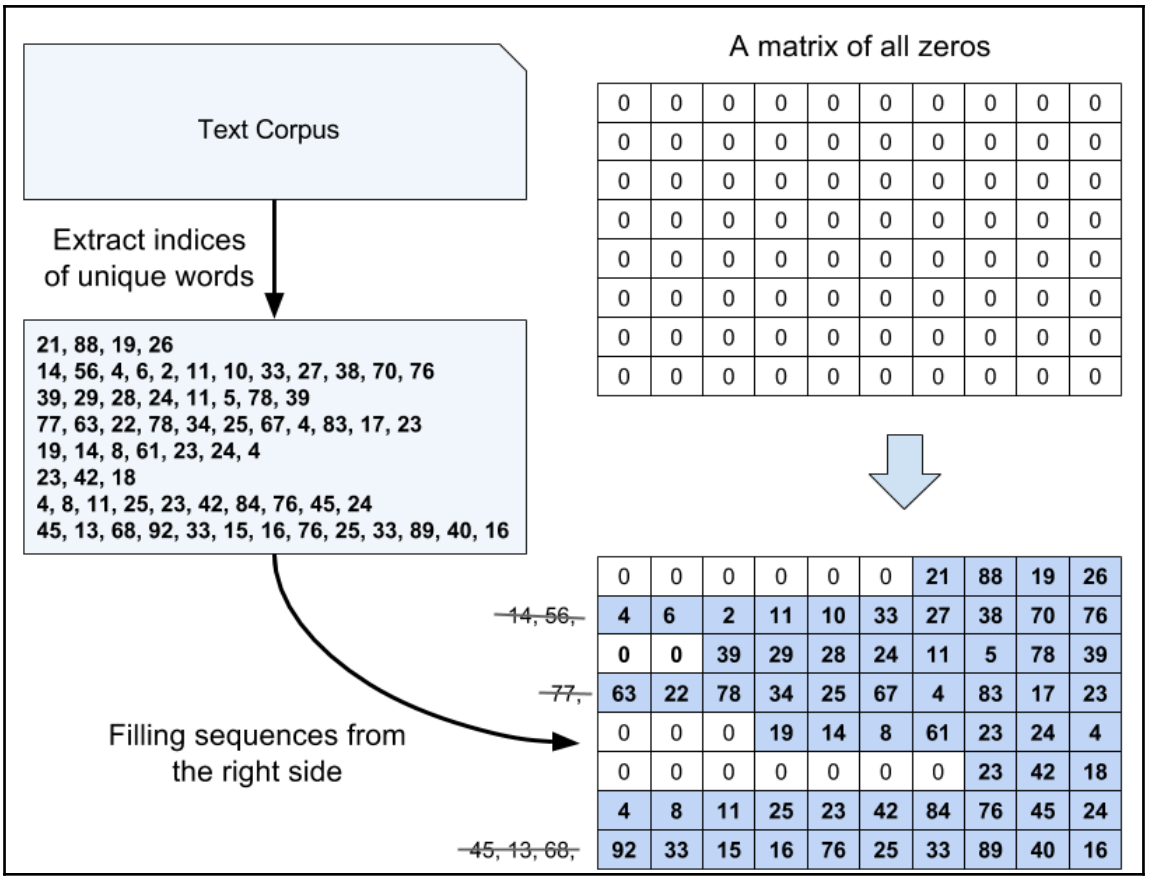

Single layer RNN
Unfold
Multilayer RNN
Unfold

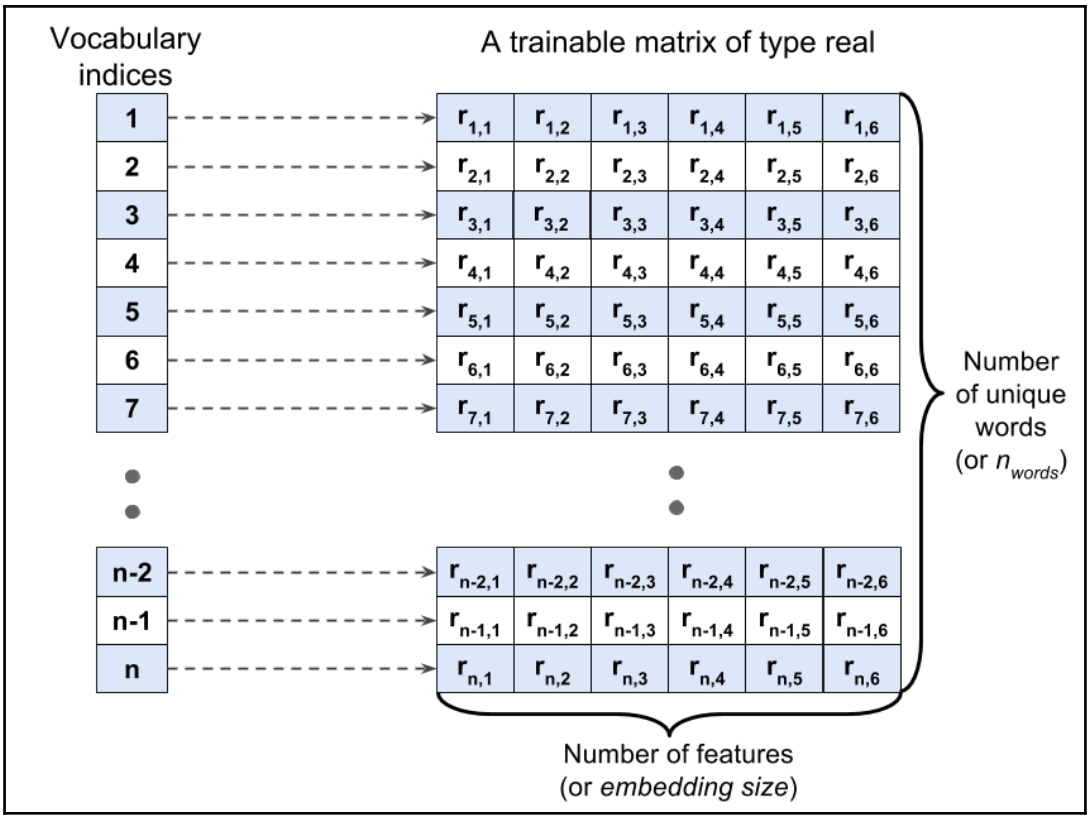

y(t)
W_hy
W_hh
h(...)
W_xh
x(t)
W_h=[W_hh ; W_xh]
Unfold
y(t-1)
W_hy
h(t-1)
W_xh
x(t-1)
y(t)
W_hy
W_hh
h(t)
W_xh
x(t)
y(t+1)
W_hy
W_hh
h(t+1)
W_xh
x(t+1)

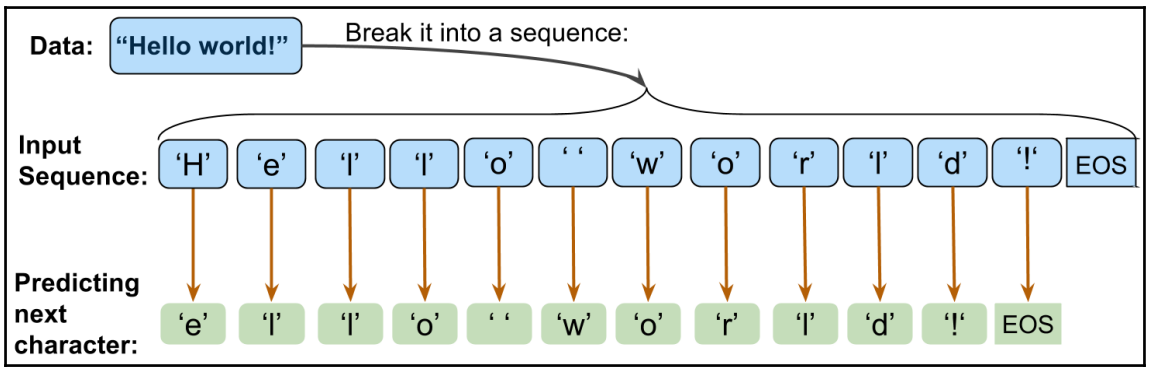

Formulation 1: h(t) =φh(Wxh x(t) + Whh h(t-1) + bh)
Wxh
x(t)
Whh
h(t-1)
bh
y(t)
h(t)
x(t)
Formulation 2: h(t) = φh(Wh [x(t); h(t-1)]T + bh)
Wh=[Wxh; Whh]
x(t)
h(t-1)
bh
Final Output:
y(t) = φy(Why h(t) + by)

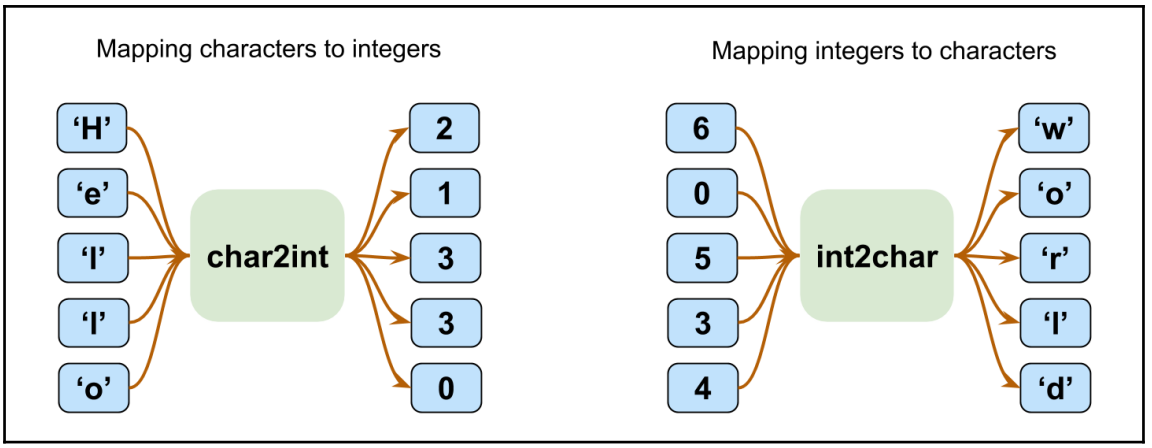

Vanishing gradient: $|w_{hh}| < 1$
Exploding gradient: $|w_{hh}| > 1$
Desirable: $|w_{hh}| = 1$
y
h
x
w=0.99
w=1.01
w=1

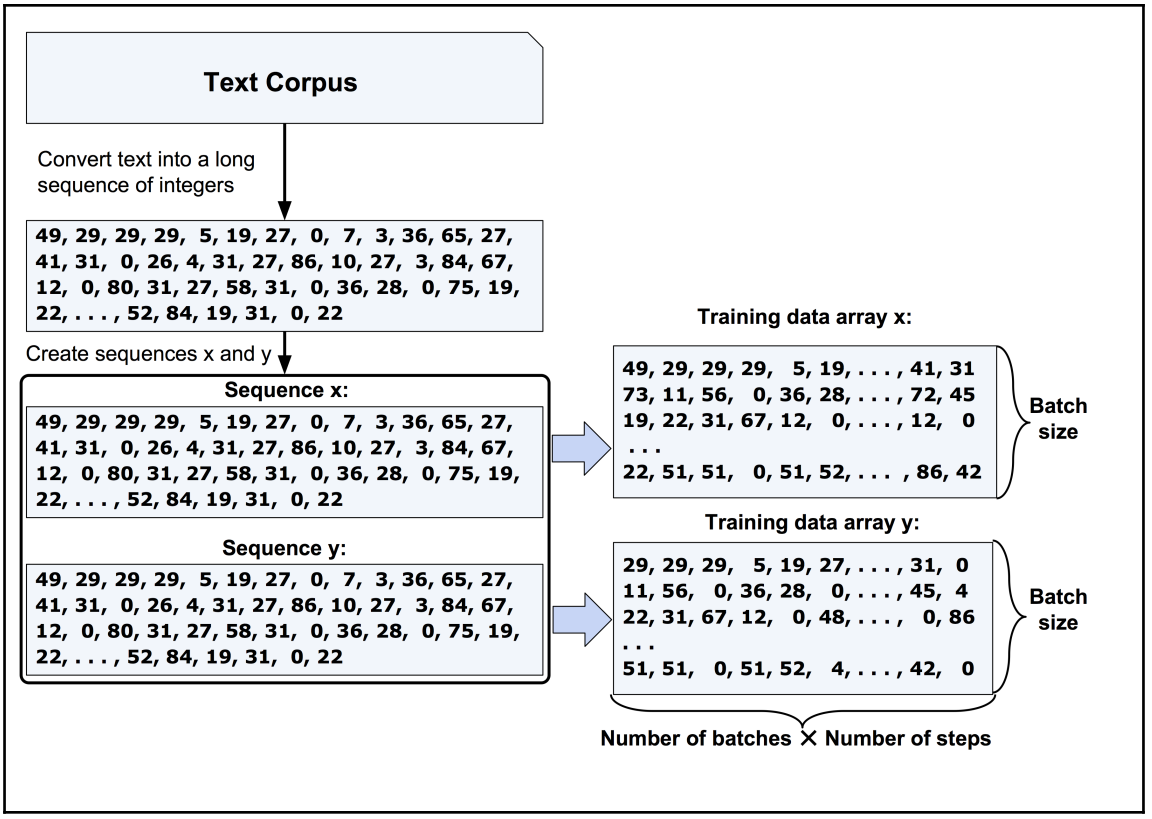

C(t-1)
C(t)
f
i
g
Tanh
σ
σ
Tanh
σ
o
Wxf Whf bf
Wxi Whi bi
Wxg Whg bg
Wxo Who bo
To next layer
h(t-1)
To next time step
h(t)
x(t)

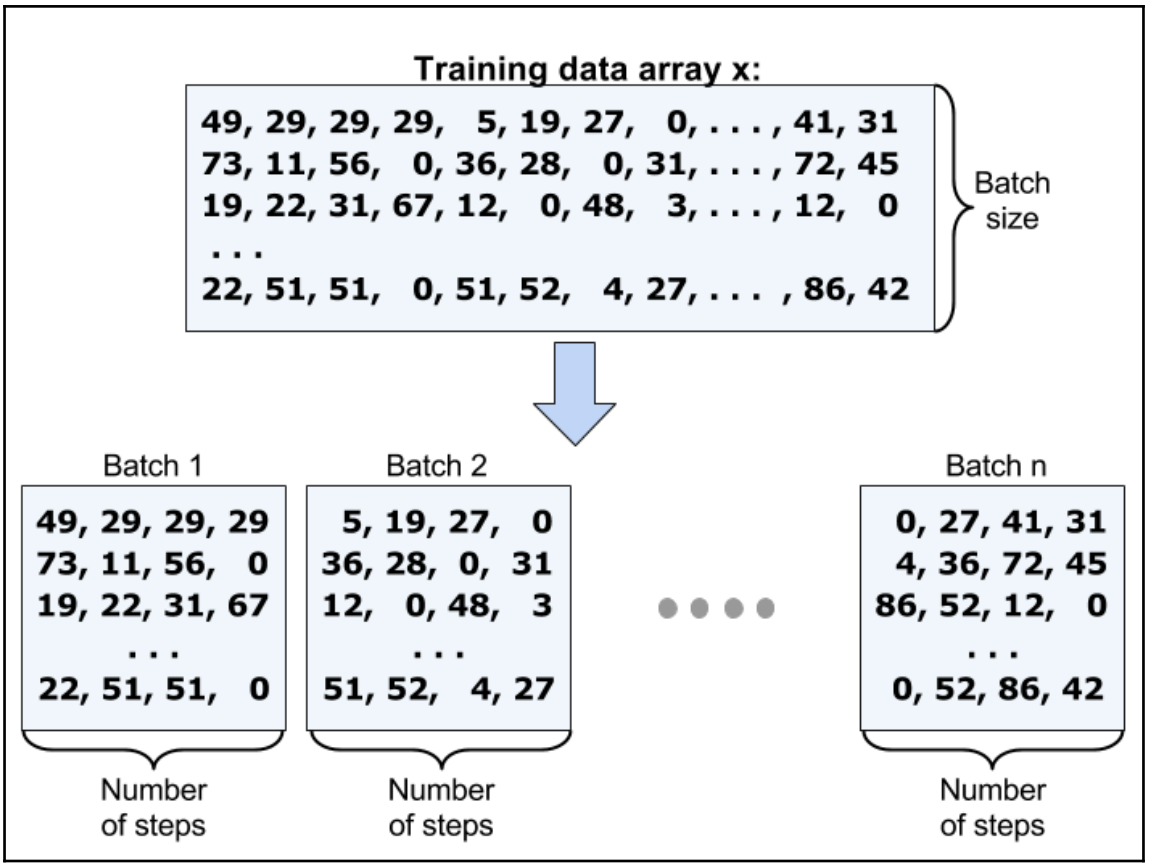

Text Corpus
Extract indices of unique words
21, 88, 19, 26
14, 56, 4, 6, 2, 11, 10, 33, 27, 38, 70, 76
39, 29, 28, 24, 11, 5, 78, 39
77, 63, 22, 78, 34, 25, 67, 4, 83, 17, 23
19, 14, 8, 61, 23, 24, 4
23, 42, 18
4, 8, 11, 25, 23, 42, 84, 76, 45, 24
45, 13, 68, 92, 33, 15, 16, 76, 25, 33, 89, 40, 16
Filling sequences from the right side
A matrix of all zeros
~~14, 56,~~
~~77,~~
~~45, 13, 68,~~


Vocabulary indices
A trainable matrix of type real
1
2
3
4
5
6
7
n-2
n-1
n
Number of unique words (or n_{words})
Number of features (or *embedding size*)


Data:
"Hello world!"
Break it into a sequence:
Input Sequence:
'H' 'e' 'l' 'l' 'o' ' ' 'w' 'o' 'r' 'l' 'd' '!' EOS
Predicting next character:
'e' 'l' 'l' 'o' ' ' 'w' 'o' 'r' 'l' 'd' '!' EOS


Mapping characters to integers
'H'
'e'
'l'
'l'
'o'
char2int
2
1
3
3
0
Mapping integers to characters
6
0
5
3
4
int2char
'w'
'o'
'r'
'l'
'd'


Text Corpus
Convert text into a long sequence of integers
49, 29, 29, 29, 5, 19, 27, 0, 7, 3, 36, 65, 27,
41, 31, 0, 26, 4, 31, 27, 86, 10, 27, 3, 84, 67,
12, 0, 80, 31, 27, 58, 31, 0, 36, 28, 0, 75, 19,
22, . . . , 52, 84, 19, 31, 0, 22
Create sequences x and y
Sequence x:
49, 29, 29, 29, 5, 19, 27, 0, 7, 3, 36, 65, 27,
41, 31, 0, 26, 4, 31, 27, 86, 10, 27, 3, 84, 67,
12, 0, 80, 31, 27, 58, 31, 0, 36, 28, 0, 75, 19,
22, . . . , 52, 84, 19, 31, 0, 22
Sequence y:
49, 29, 29, 29, 5, 19, 27, 0, 7, 3, 36, 65, 27,
41, 31, 0, 26, 4, 31, 27, 86, 10, 27, 3, 84, 67,
12, 0, 80, 31, 27, 58, 31, 0, 36, 28, 0, 75, 19,
22, . . . , 52, 84, 19, 31, 0, 22
Training data array x:
49, 29, 29, 29, 5, 19, . . . , 41, 31
73, 11, 56, 0, 36, 28, . . . , 72, 45
19, 22, 31, 67, 12, 0, . . . , 12, 0
. . .
22, 51, 51, 0, 51, 52, . . . , 86, 42
Batch size
Training data array y:
29, 29, 29, 5, 19, 27, . . . , 31, 0
11, 56, 0, 36, 28, 0, . . . , 45, 4
22, 31, 67, 12, 0, 48, . . . , 0, 86
. . .
51, 51, 0, 51, 52, 4, . . . , 42, 0
Batch size
Number of batches × Number of steps




```
The stall soues tay and the hates,
The perse in there is that so the meanes this made there

   Ham. Ile teath thes are this makere of a driane,
Why shis mestend the Casst of is singe,
In this to this, to mers it is for marth,
Ase hinees sim thig tald ow a tore andere,
In histhene tistere shere this wile and my Lord:
And tit mighes the secleer allost heruen, and that hash to sall and hears,
If you his moses tonger and mout ofr mesting a forte tis at

   Pomin. Where in you dist and sintere shan shall
```