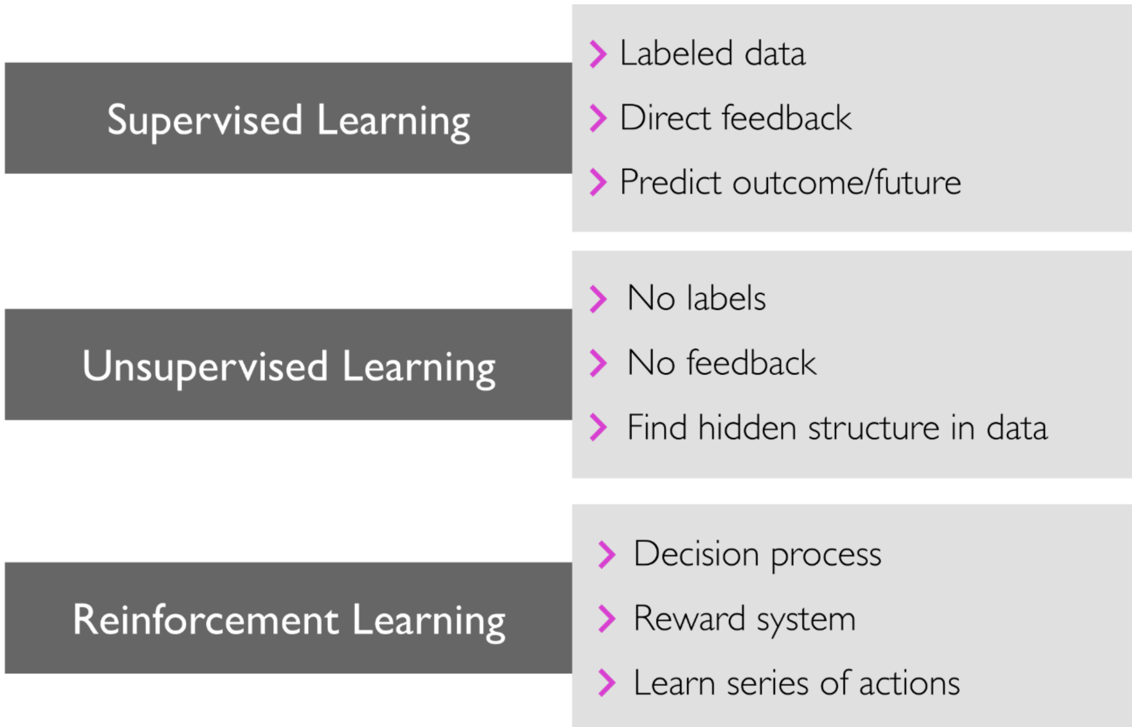
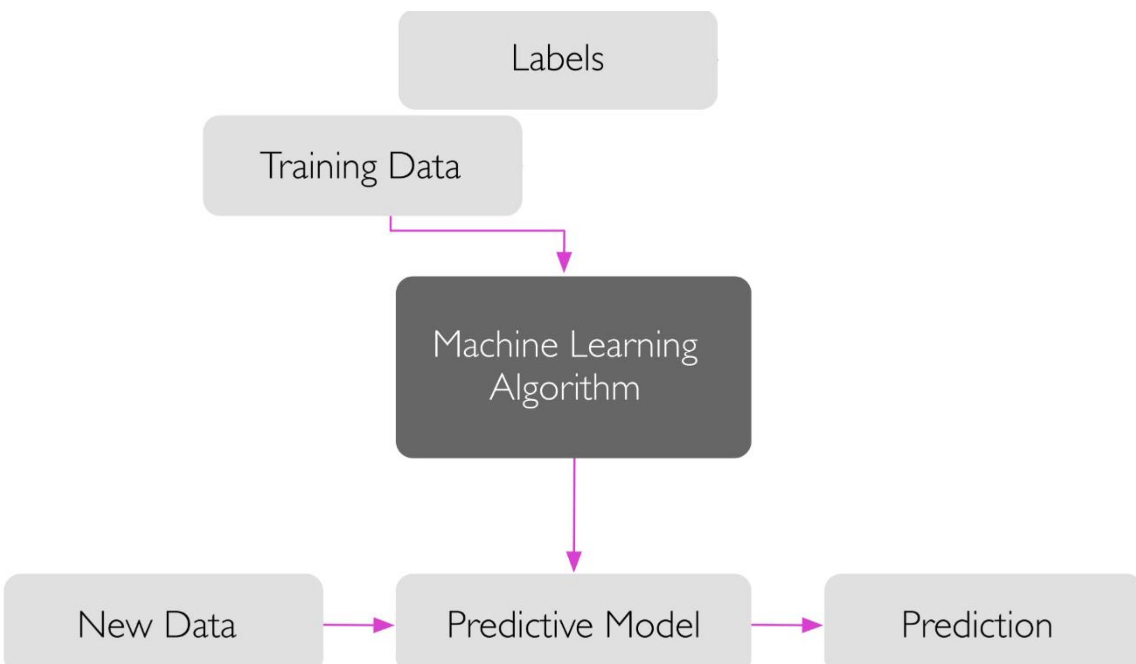
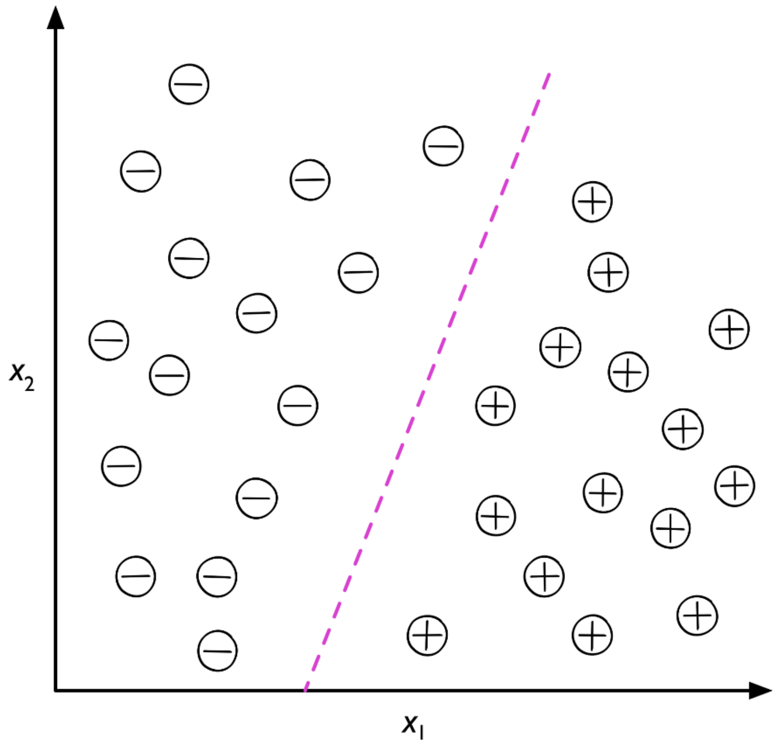
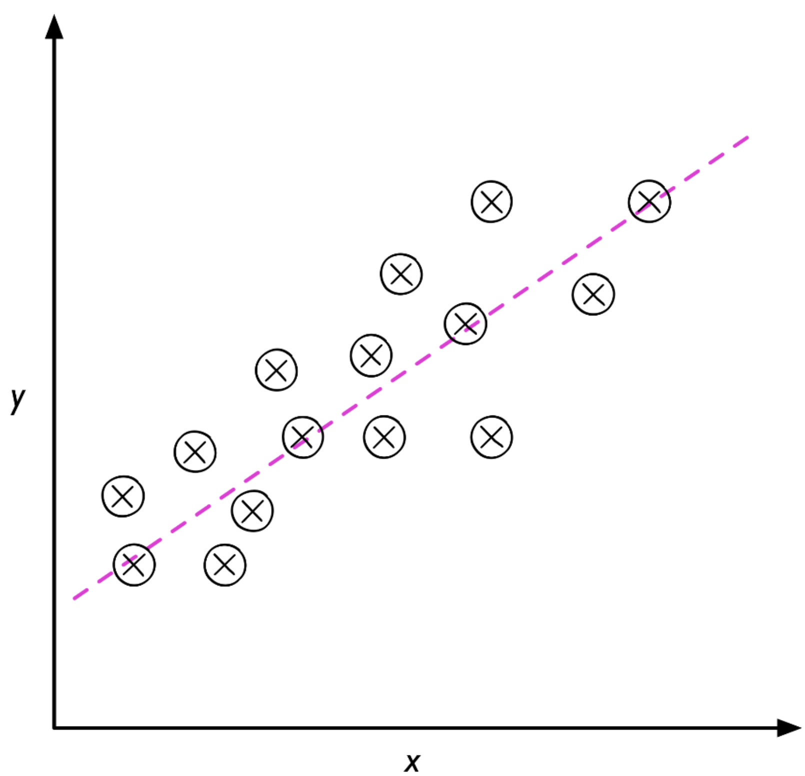
Chapter 01: Giving Computers the Ability to Learn from Data



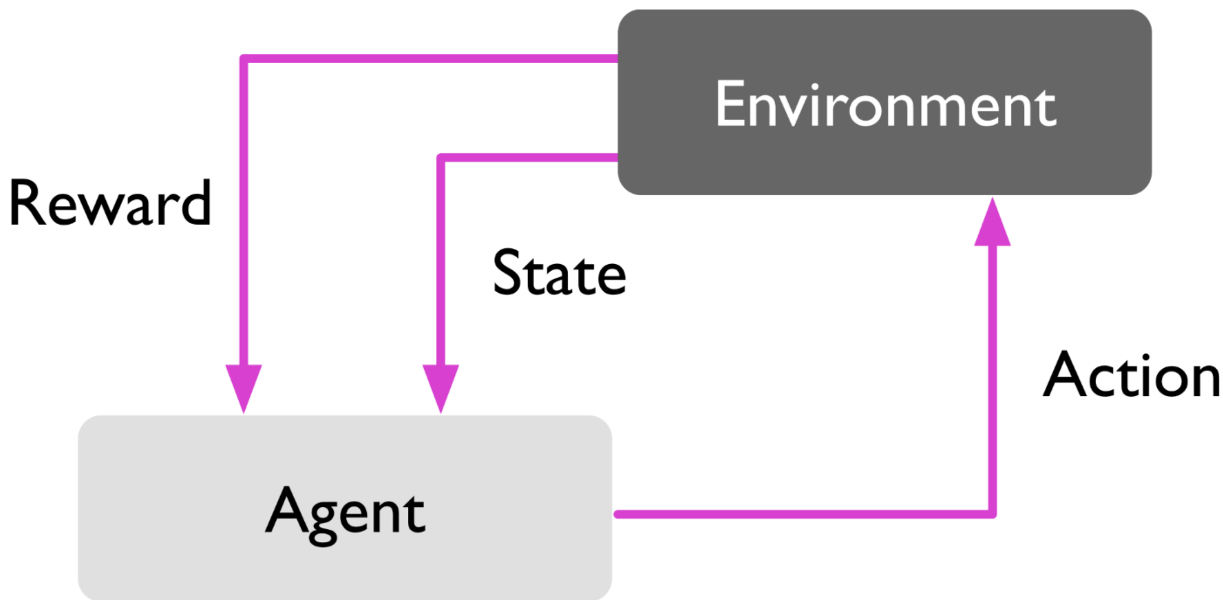


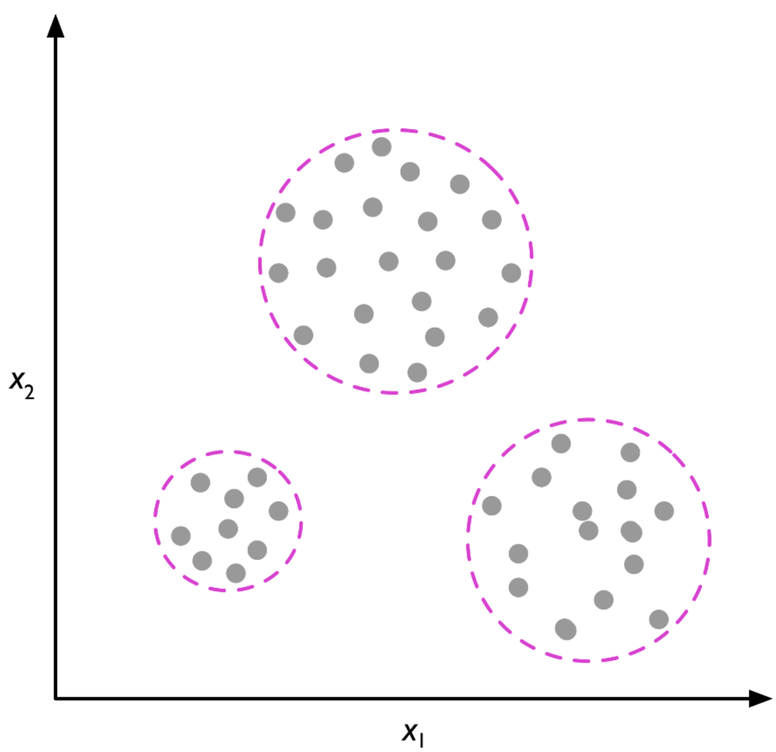


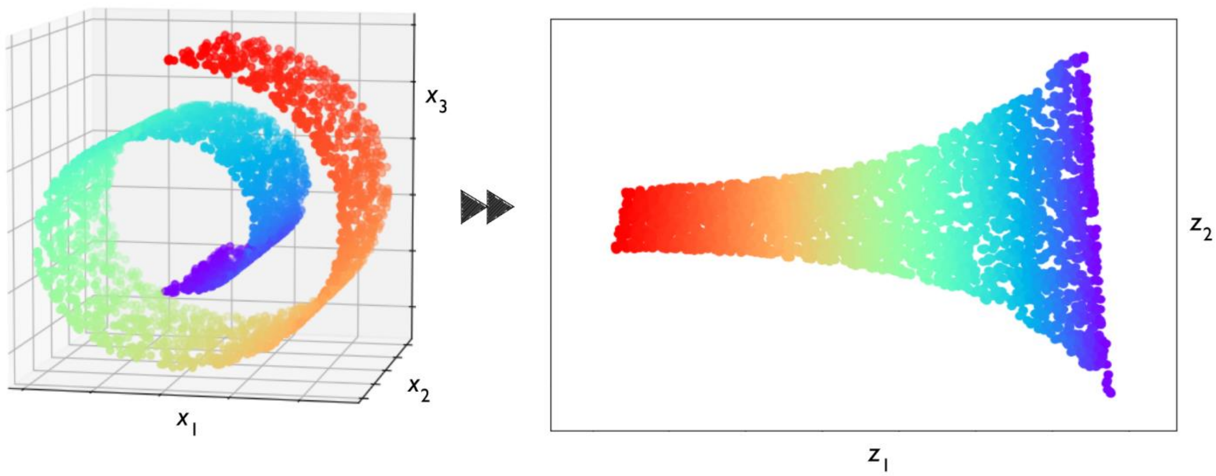
x_2
x_1



y
x

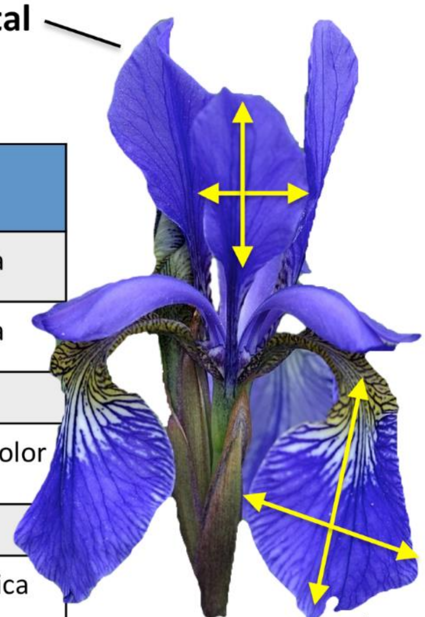

Environment
Reward
State
Action
Agent


x_2
x_1



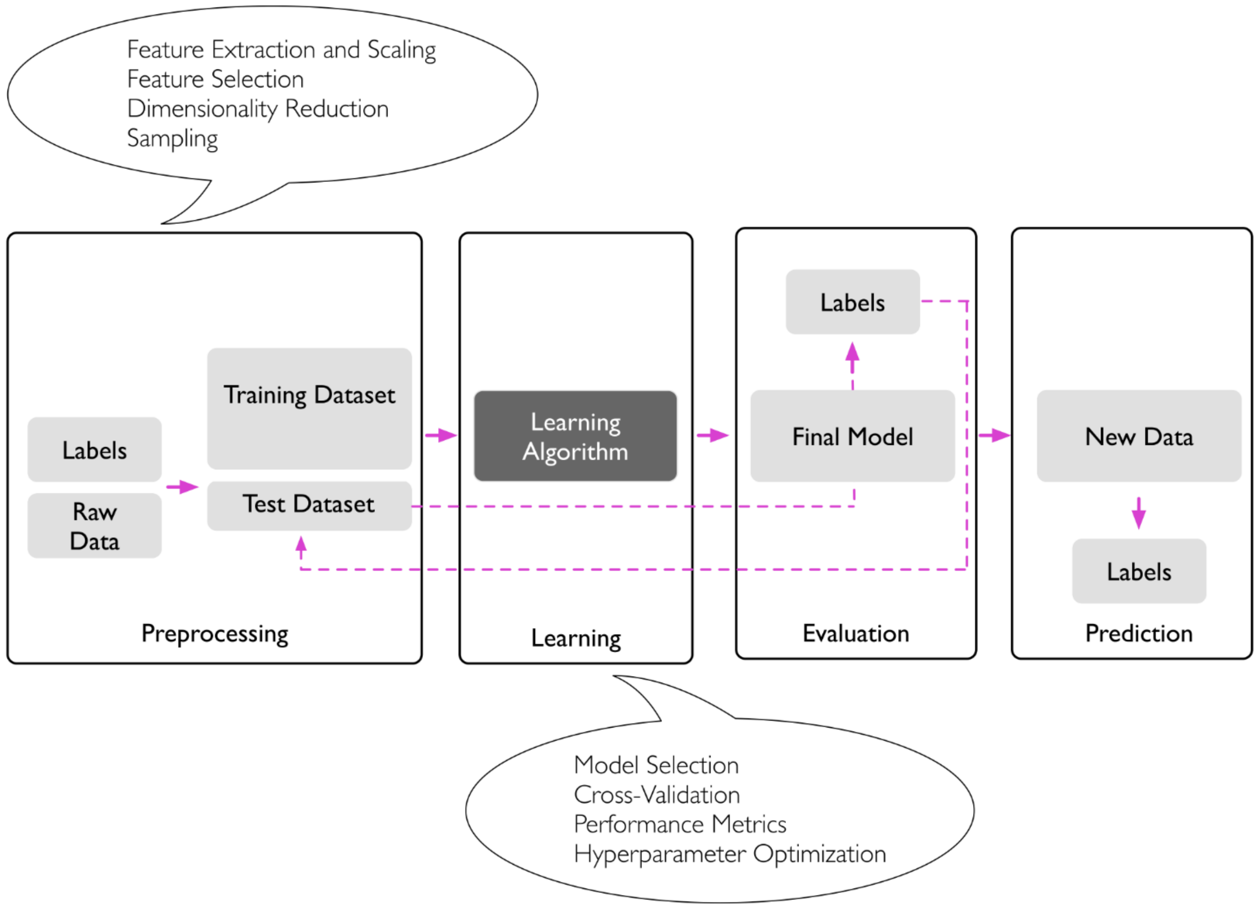
Samples
(instances, observations)

	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

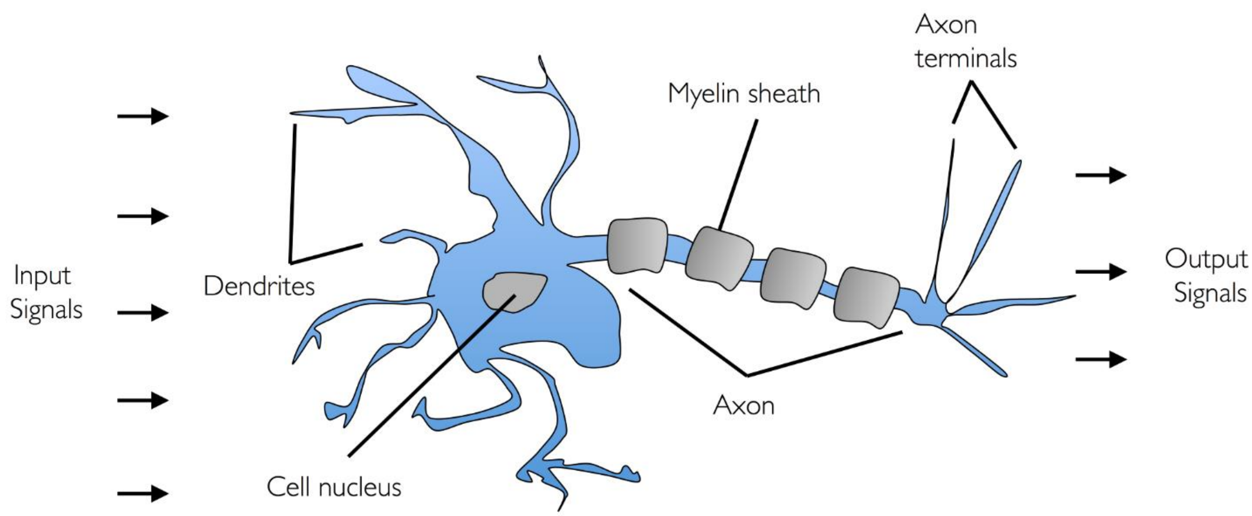


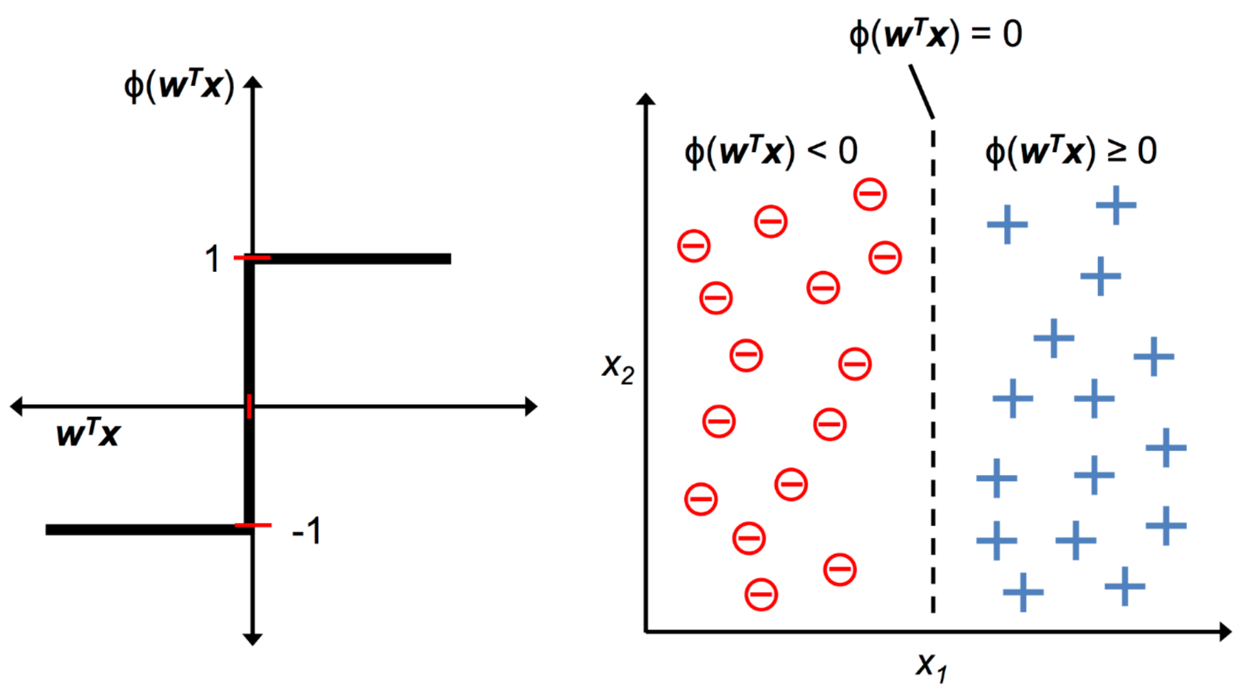
Class labels
(targets)

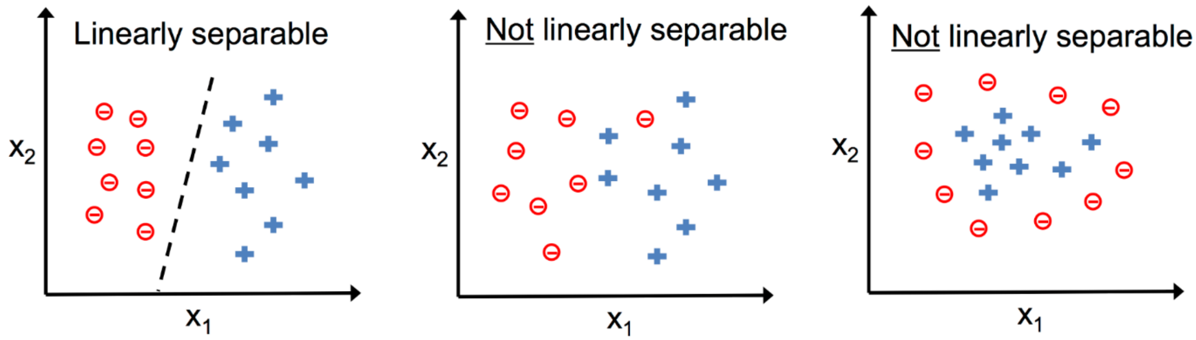
Features
(attributes, measurements, dimensions)

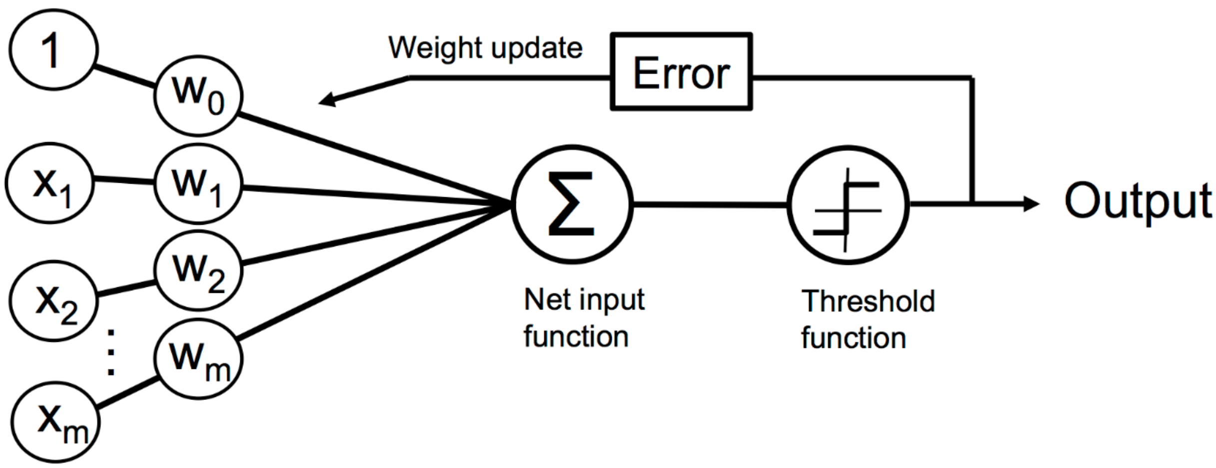

Feature Extraction and Scaling
Feature Selection
Dimensionality Reduction
Sampling
Labels
Raw Data
Training Dataset
Test Dataset
Preprocessing
Learning Algorithm
Learning
Labels
Final Model
Evaluation
New Data
Labels
Prediction
Model Selection
Cross-Validation
Performance Metrics
Hyperparameter Optimization

Chapter 02: Training Simple Machine Learning Algorithms for Classification

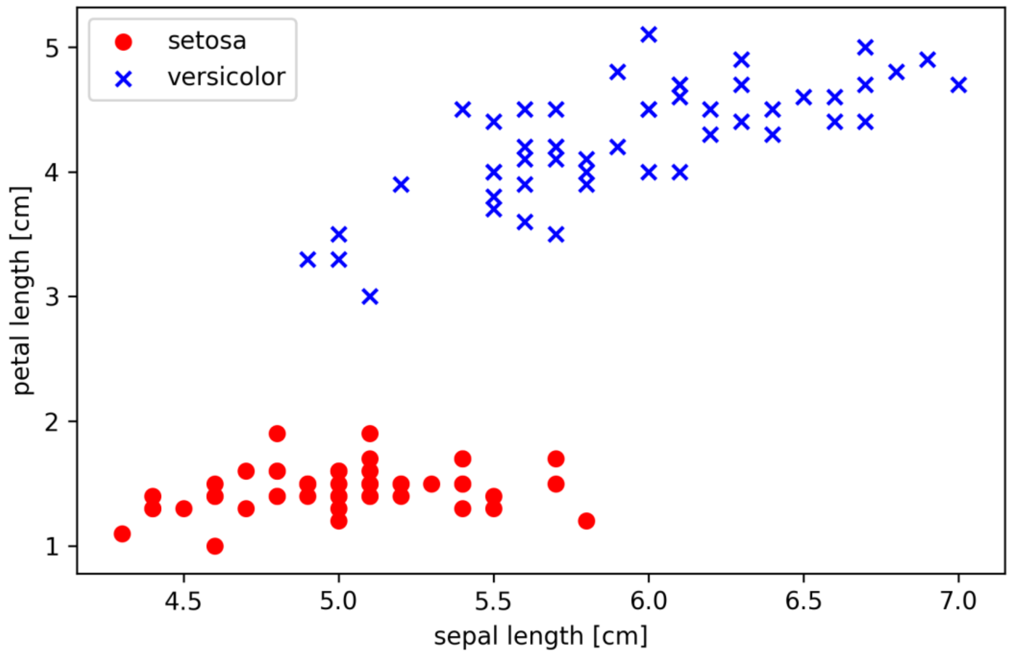


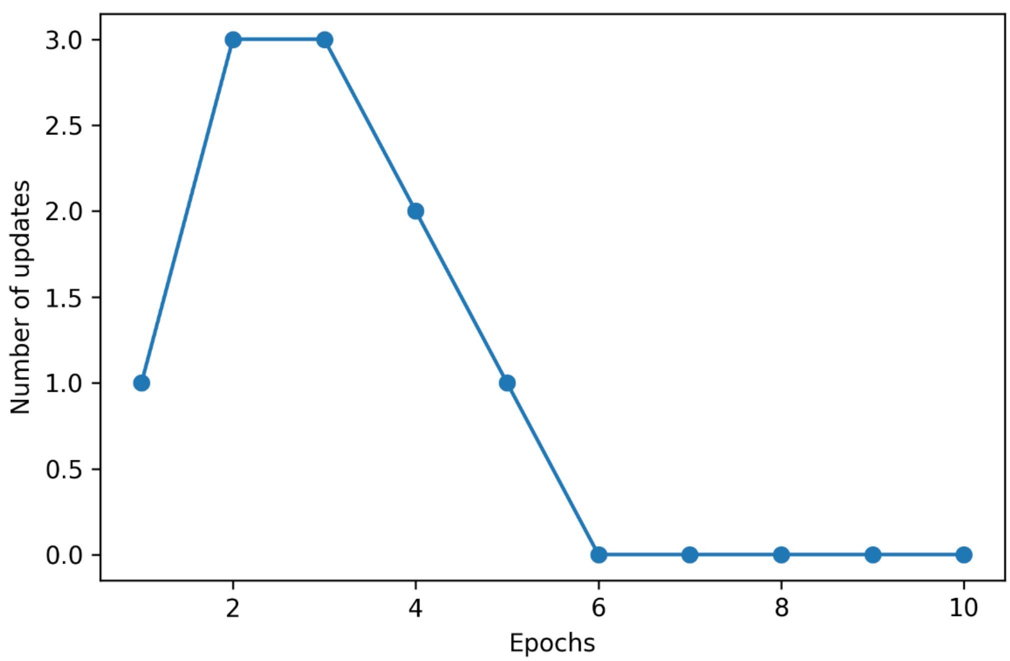


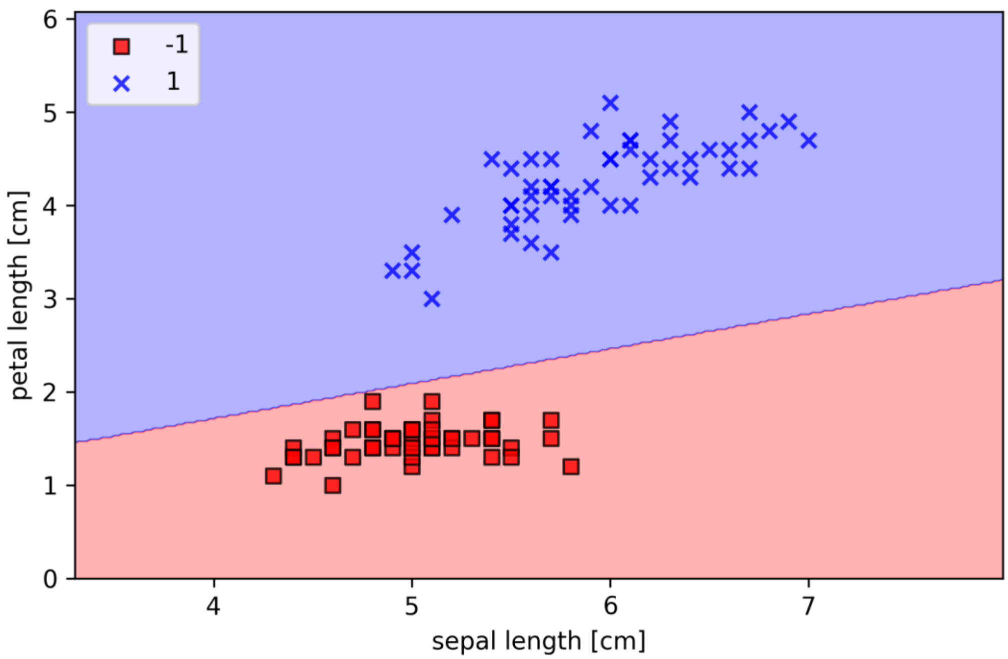


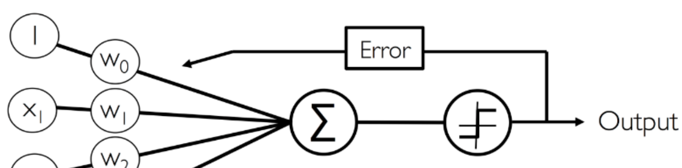


	0	1	2	3	4
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

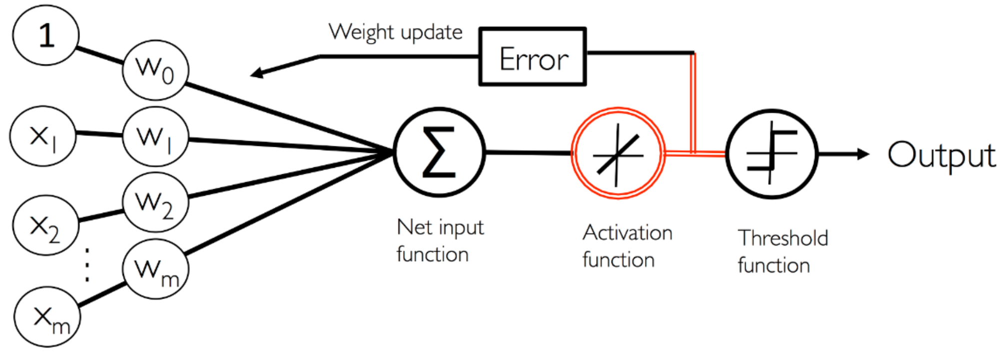

setosa
versicolor
petal length [cm]
sepal length [cm]
5
4
3
2
1
4.5
5.0
5.5
6.0
6.5
7.0


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

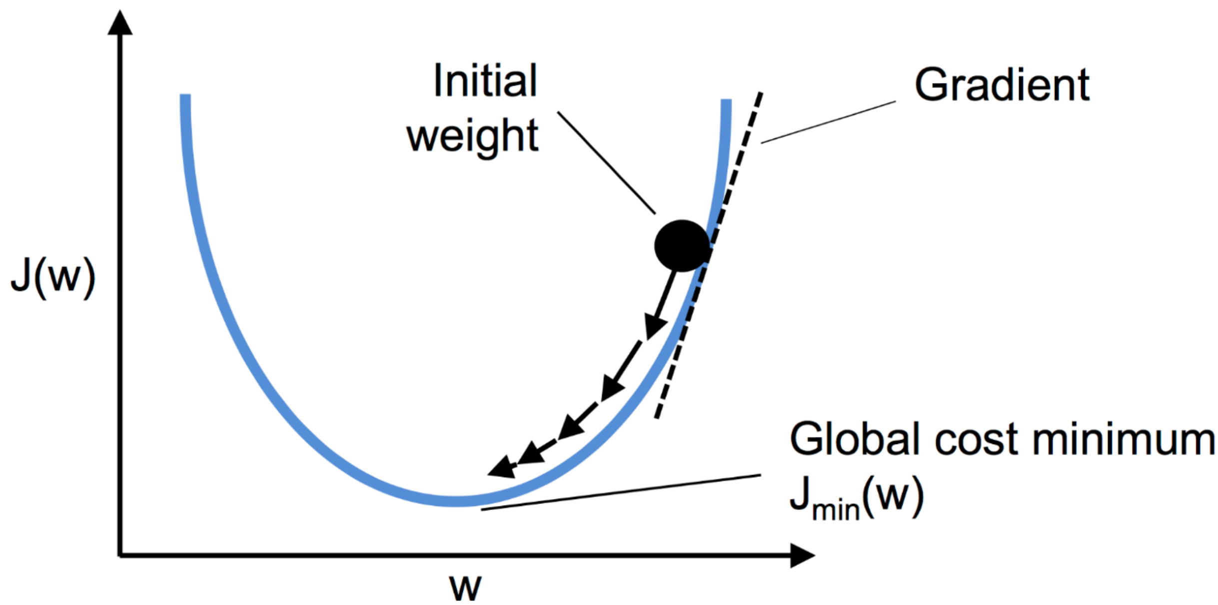


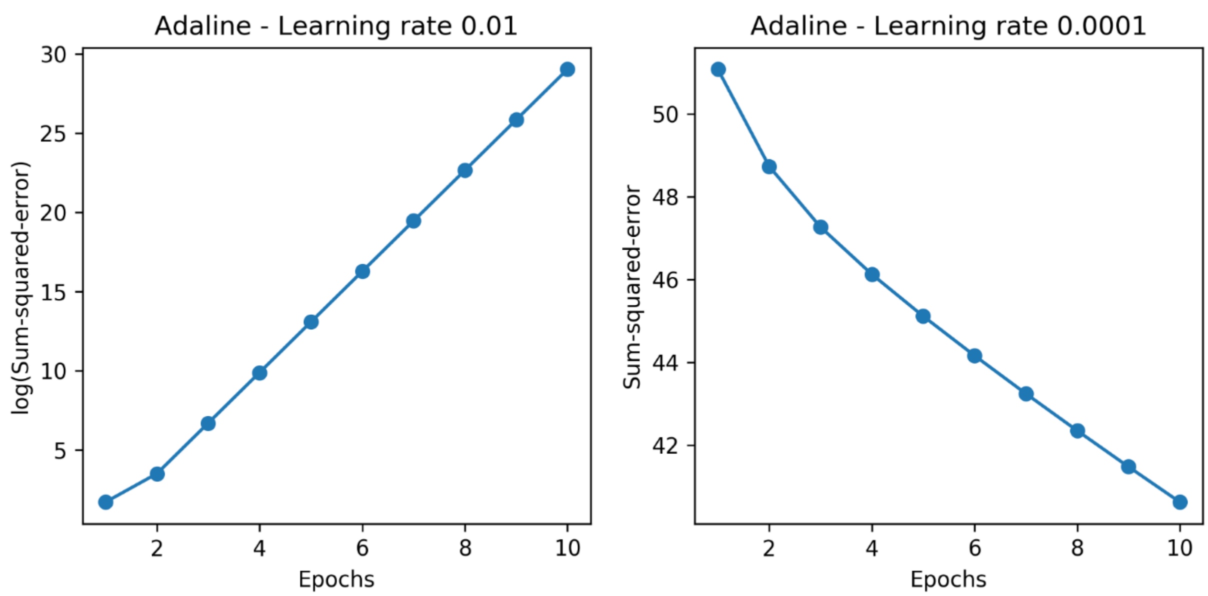


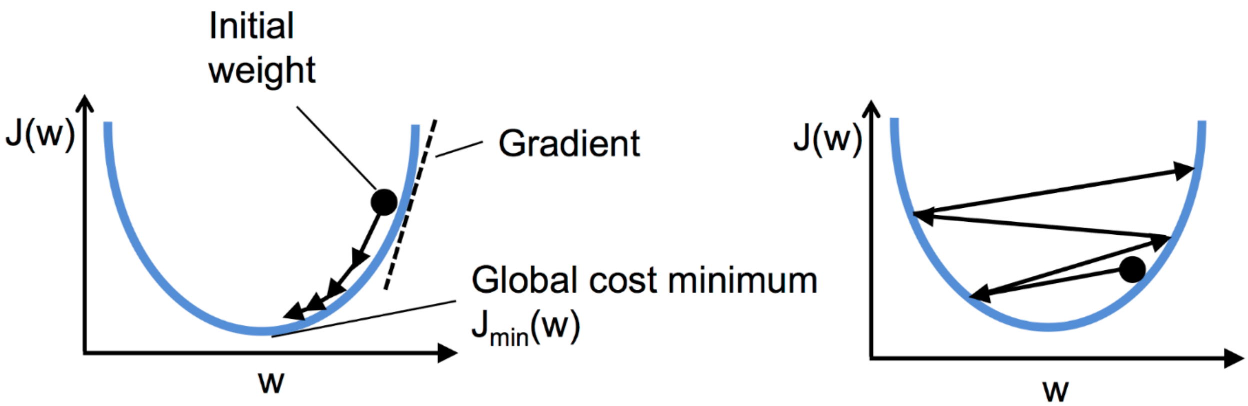
Perceptron

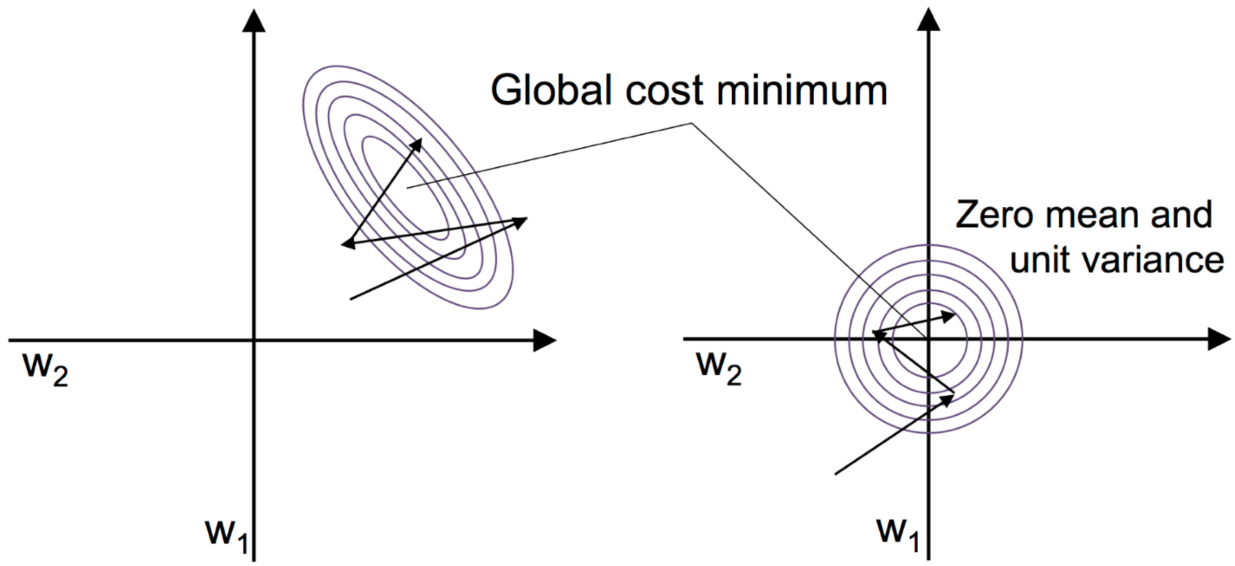


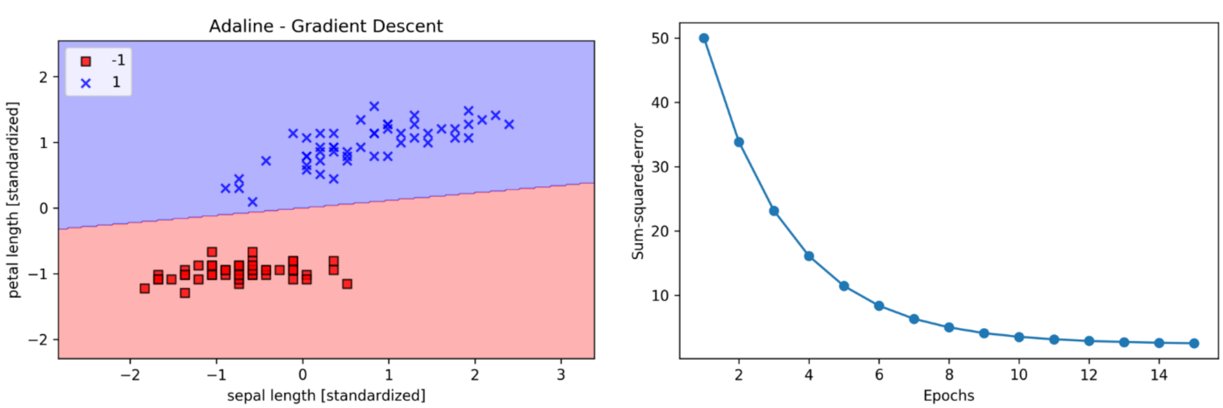
Adaptive Linear Neuron (Adaline)

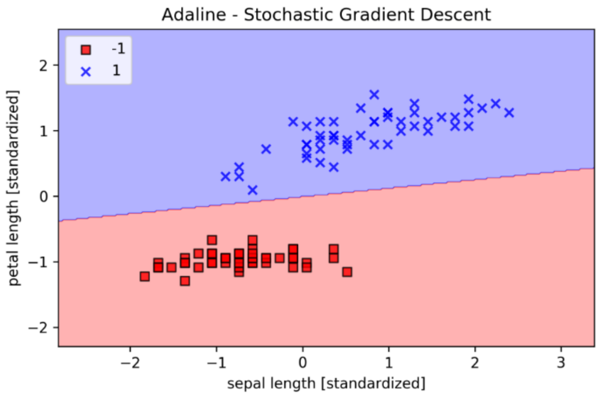

Initial
weight
Gradient
J(w)
Global cost minimum
J_min(w)
w

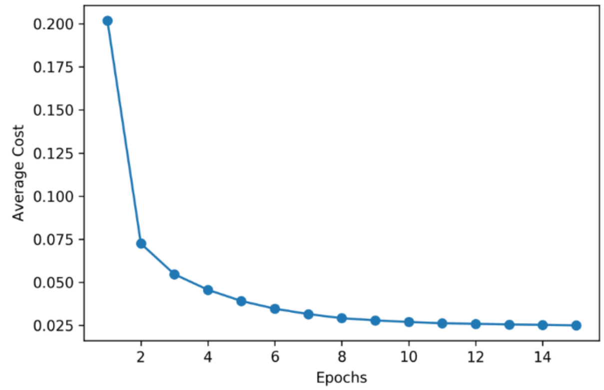

Adaline - Learning rate 0.01
log(Sum-squared-error)
Epochs
Adaline - Learning rate 0.0001
Sum-squared-error
Epochs


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

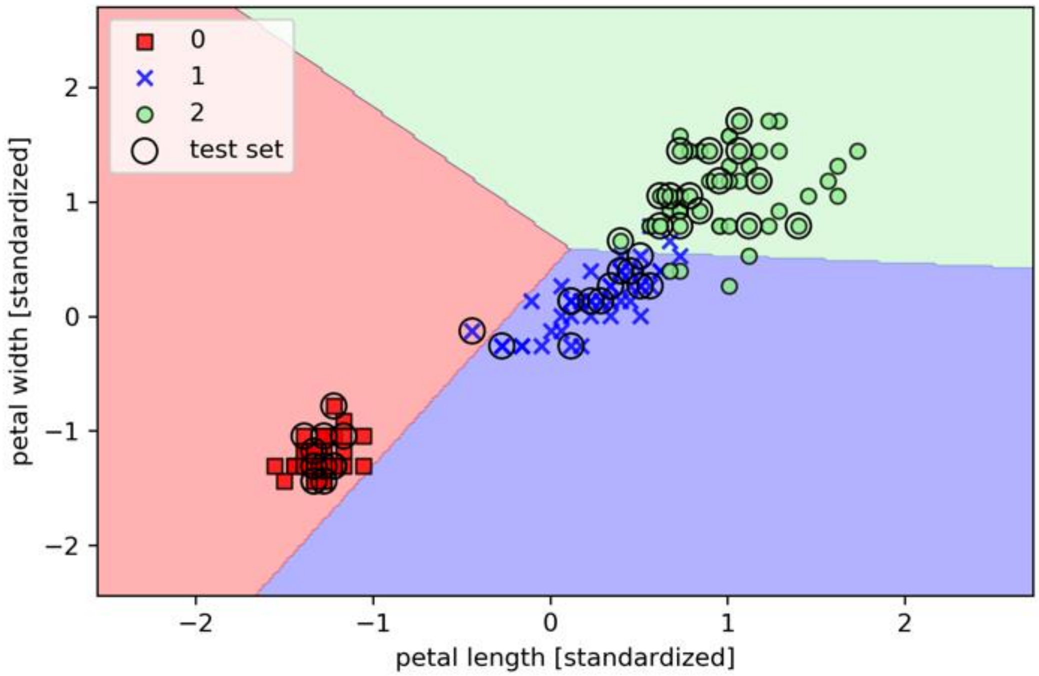

Global cost minimum
Zero mean and
unit variance
w_2
w_1
w_2
w_1

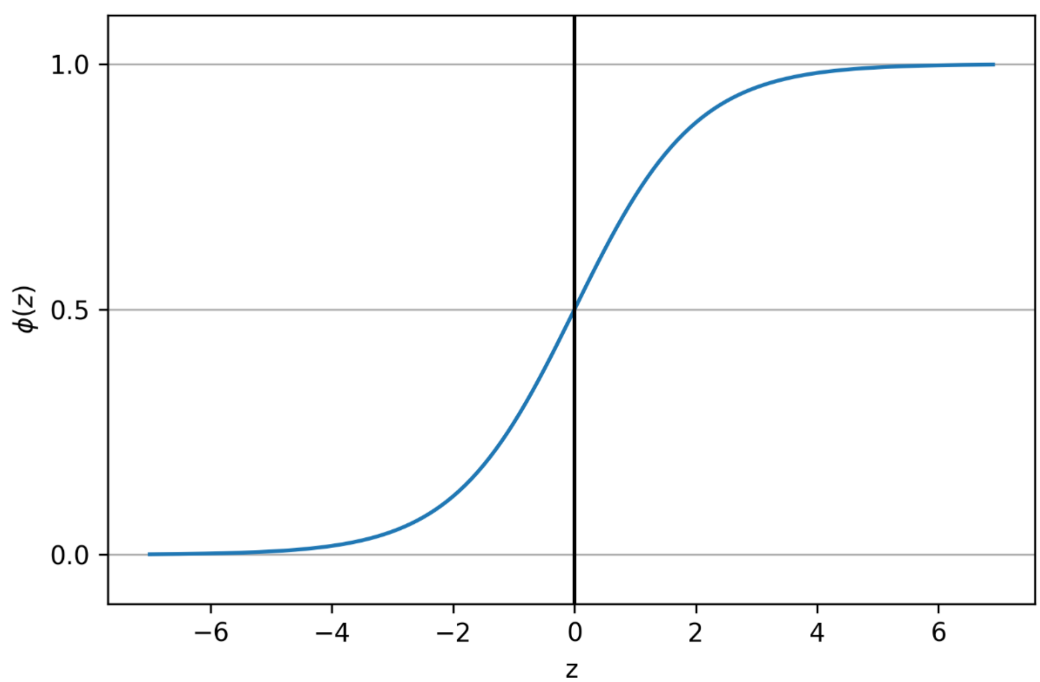

Adaline - Gradient Descent
-1
1
petal length [standardized]
sepal length [standardized]
Sum-squared-error
Epochs

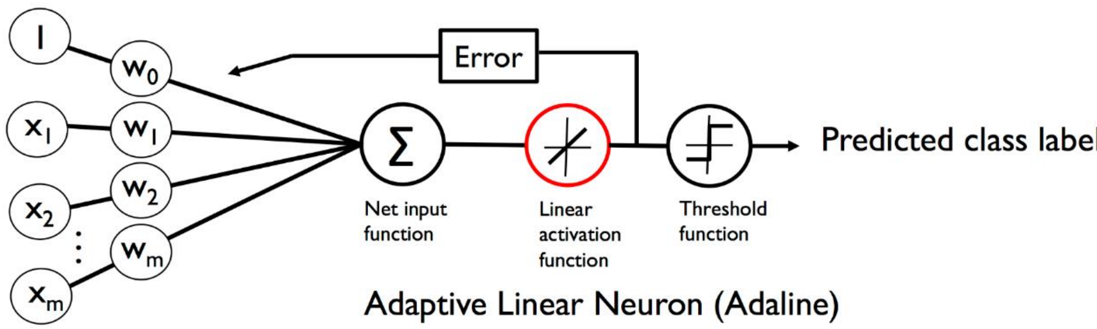

Adaline - Stochastic Gradient Descent
-1
1
petal length [standardized]
sepal length [standardized]


Average Cost
Epochs

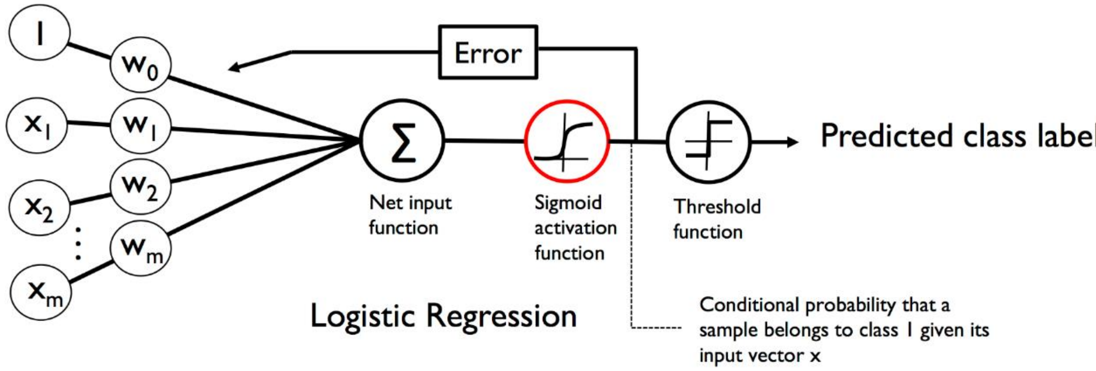
Chapter 03: A Tour of Machine Learning Classifiers Using scikit-learn



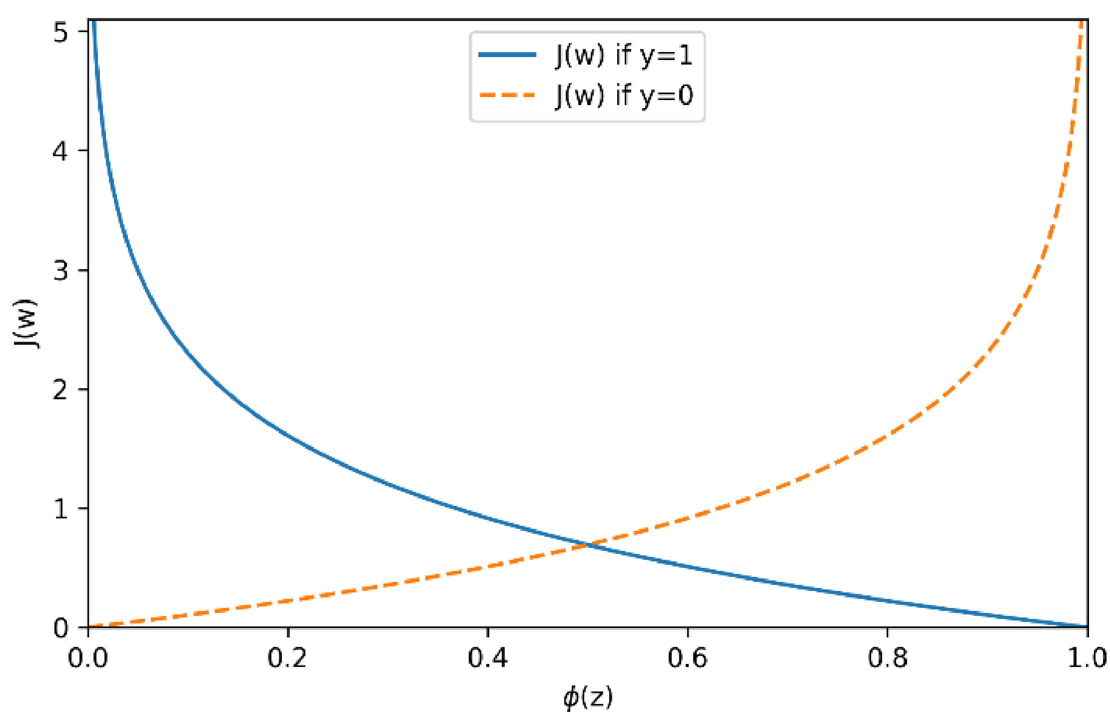




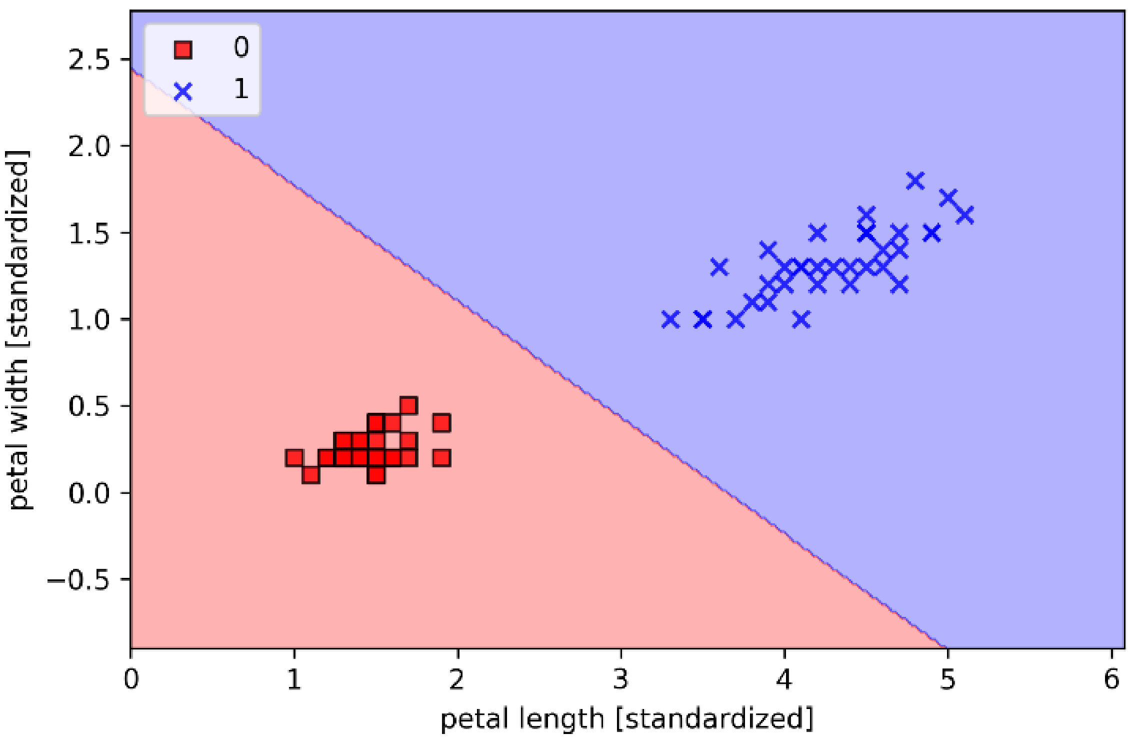
1
w0
x1
w1
x2
w2
wm
xm
Error
Net input function
Linear activation function
Threshold function
Predicted class label
Adaptive Linear Neuron (Adaline)

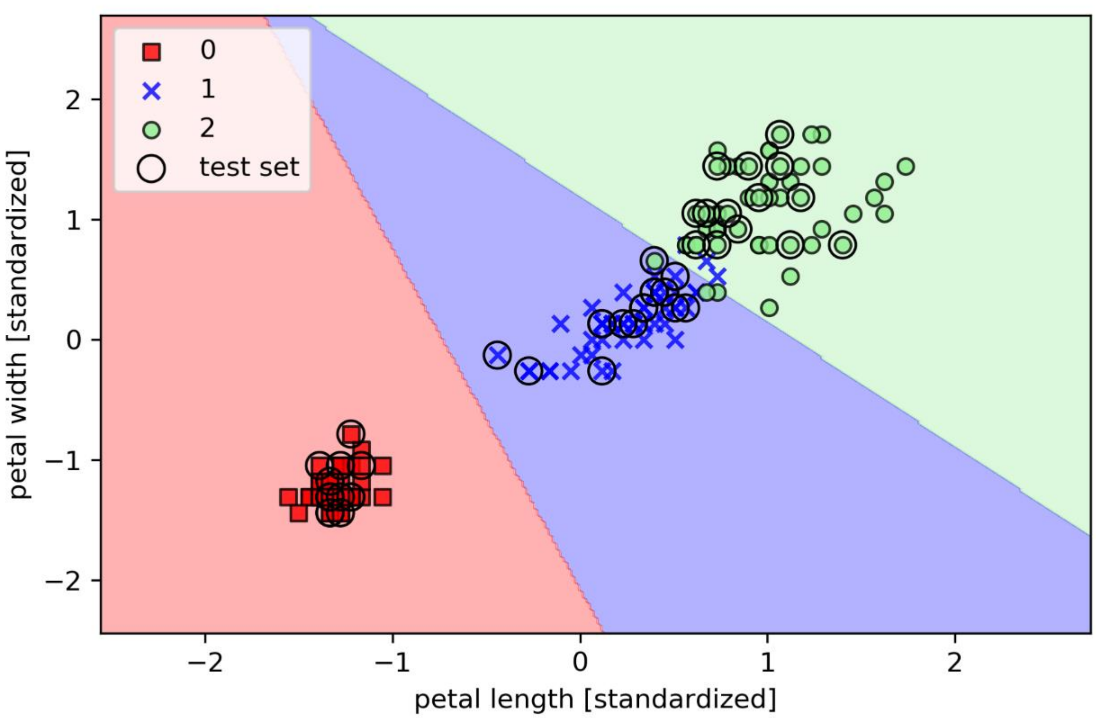


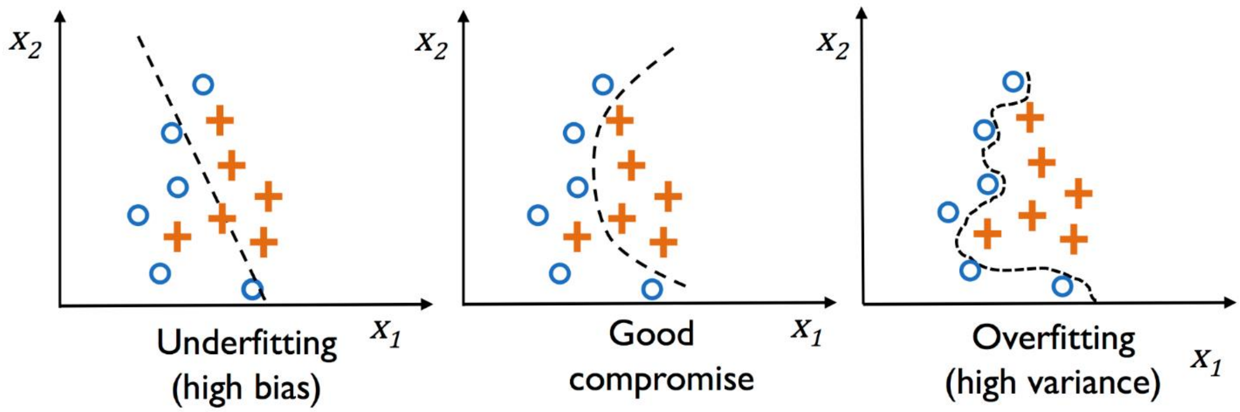
1
w0
x1
w1
x2
w2
wm
xm
Error
Net input function
Sigmoid activation function
Threshold function
Predicted class label
Logistic Regression
Conditional probability that a sample belongs to class 1 given its input vector x

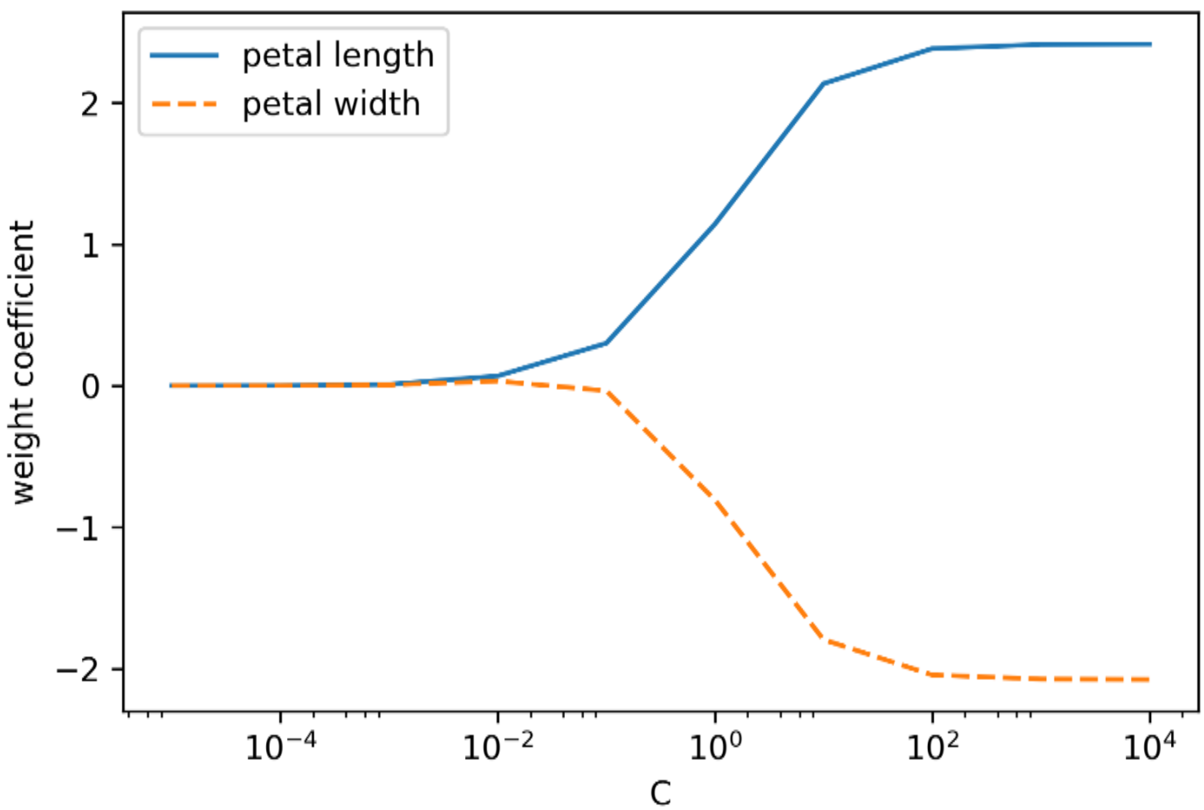


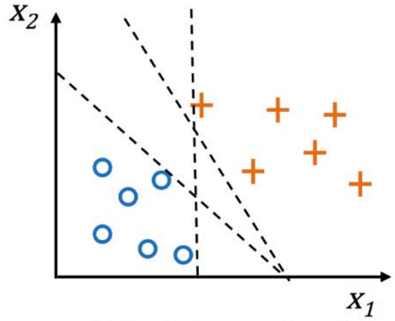
J(w) if y=1
J(w) if y=0
J(w)
φ(z)
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0


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]

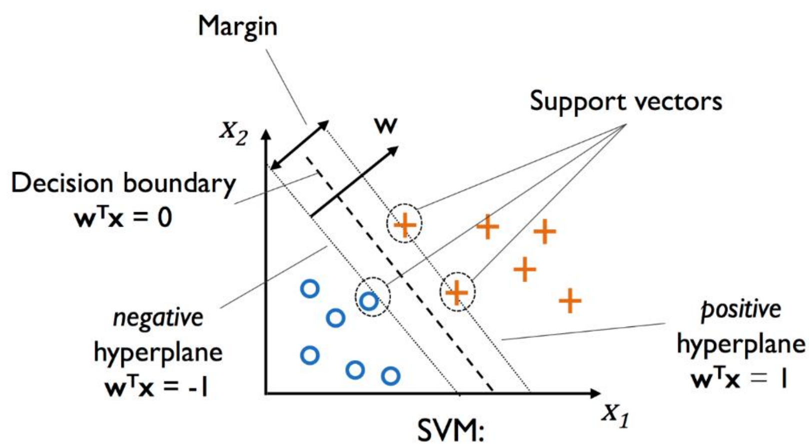

0
1
2
test set
2
1
0
−1
−2
petal width [standardized]
−2
−1
0
1
2
petal length [standardized]


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

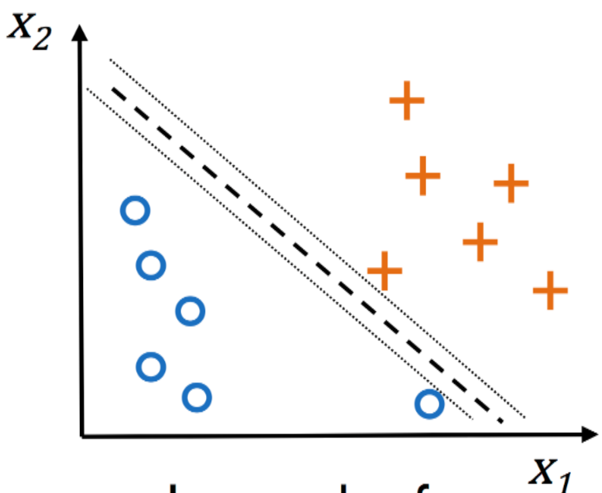

petal length
petal width
weight coefficient
2
1
0
−1
−2
10^{-4}
10^{-2}
10^{0}
10^{2}
10^{4}
C



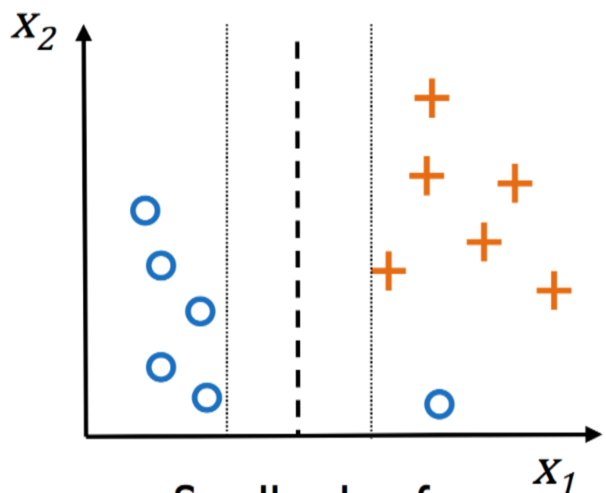
Which hyperplane?



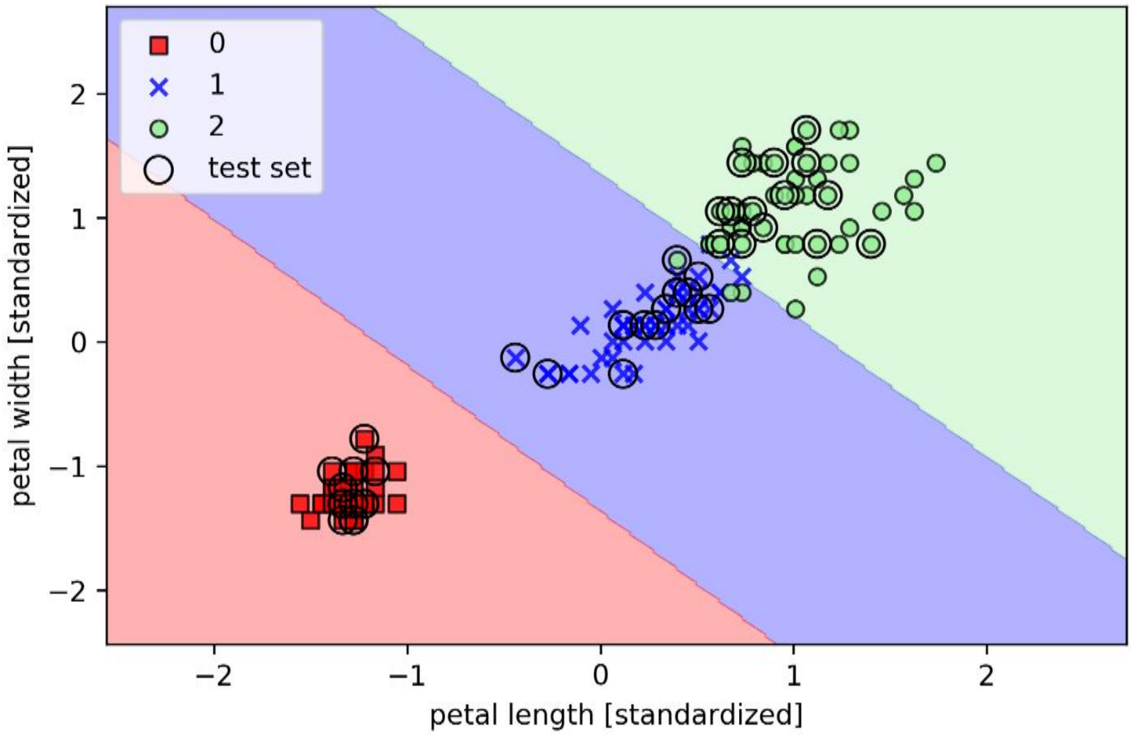
SVM:
Maximize the margin

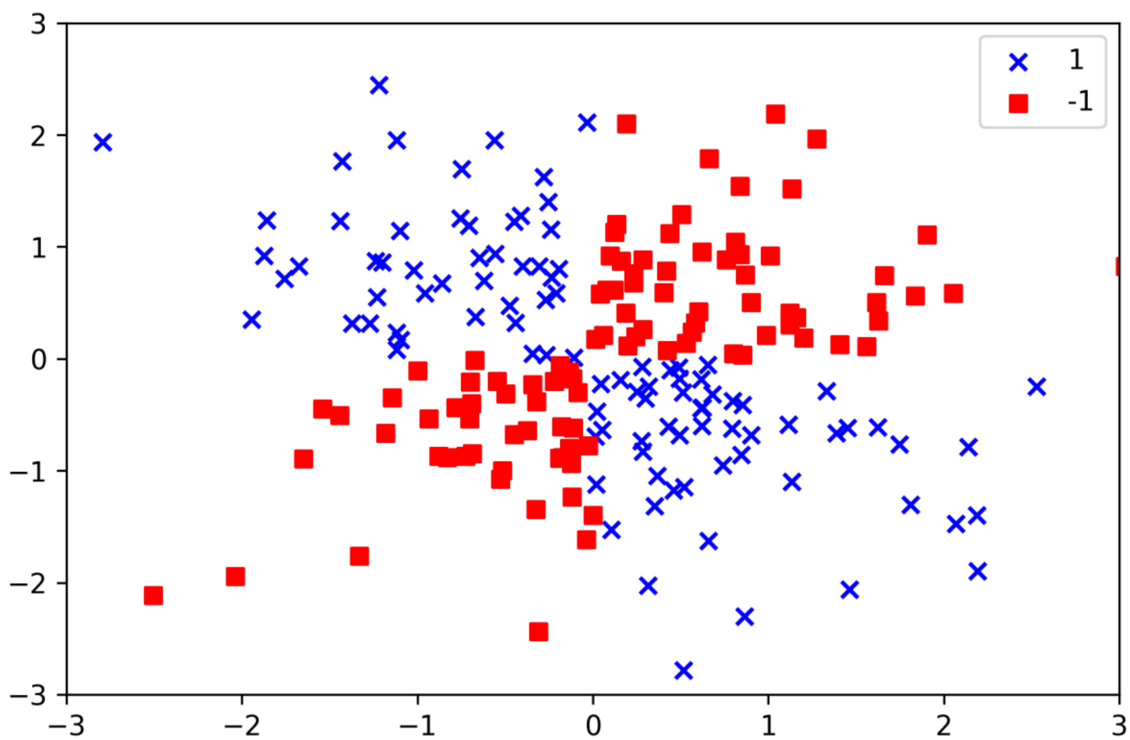


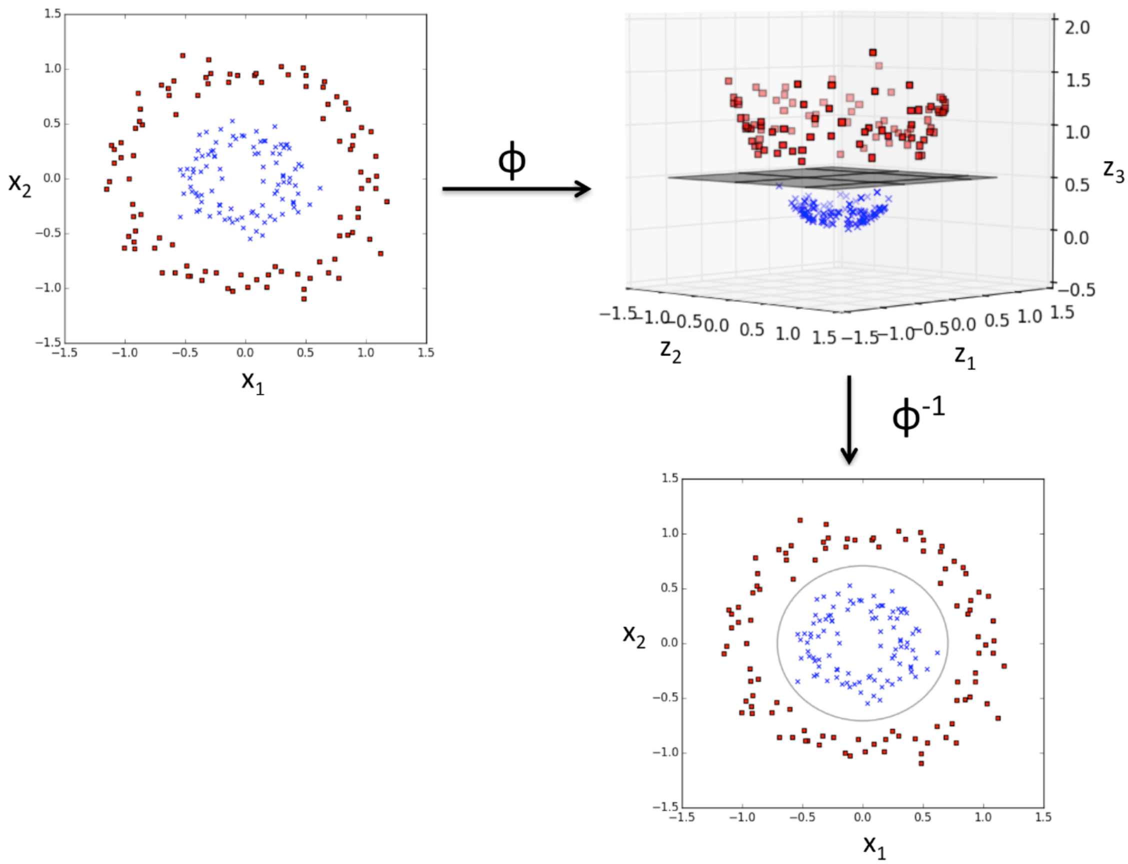
Large value for parameter C



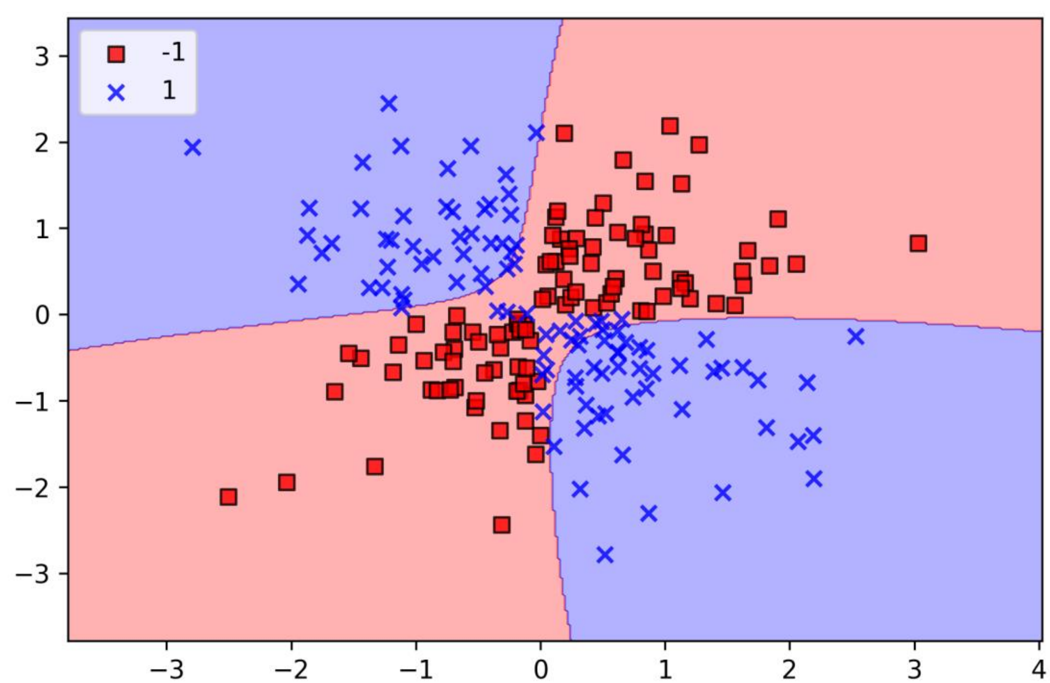
Small value for parameter C


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

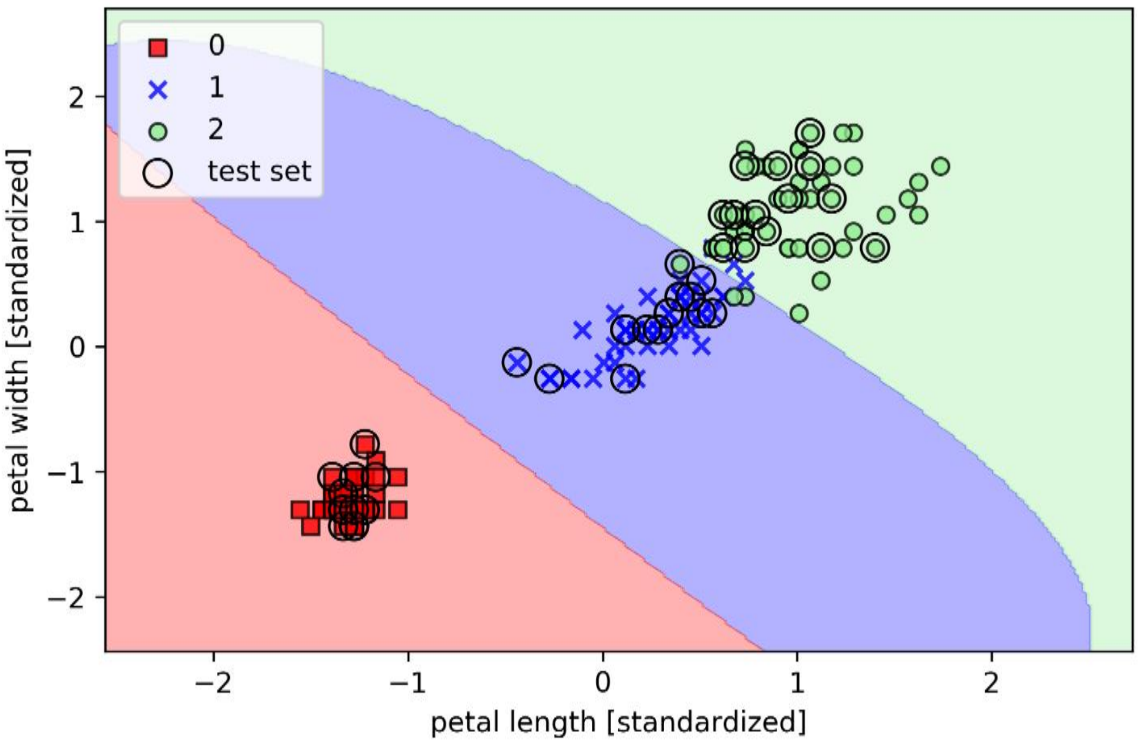

1
-1
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

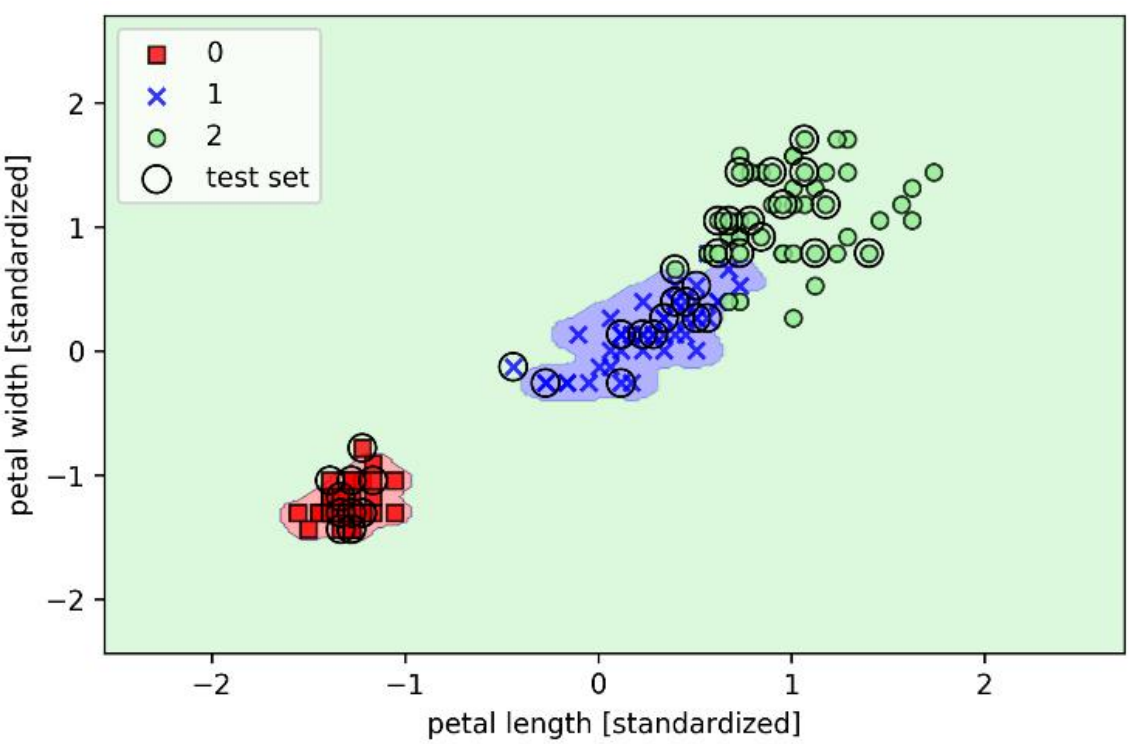


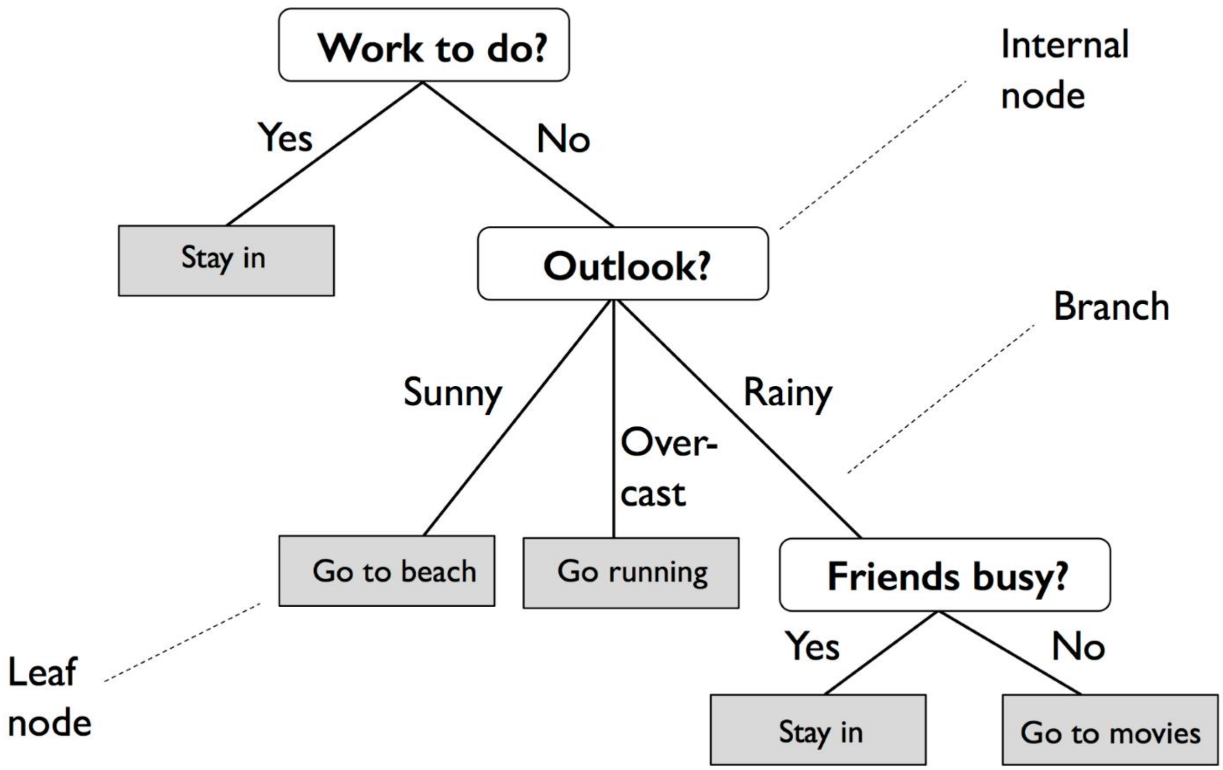
x2
x1
ϕ
z3
z2
z1
ϕ-1
x2
x1

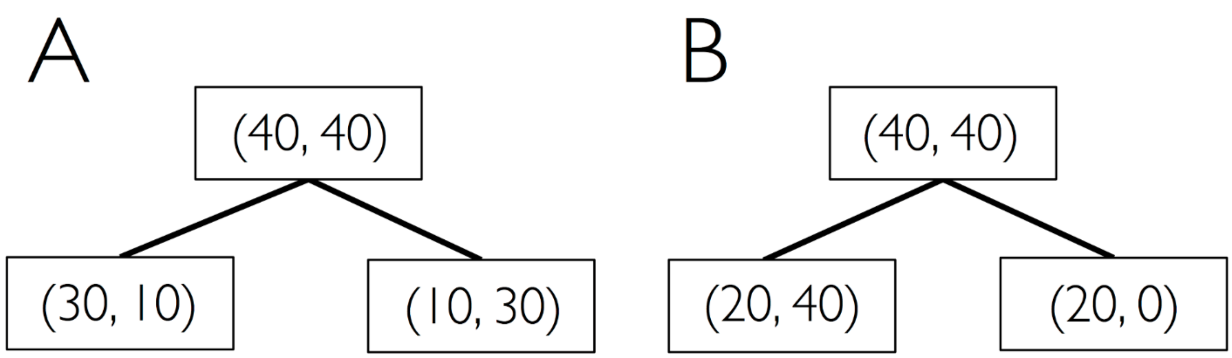


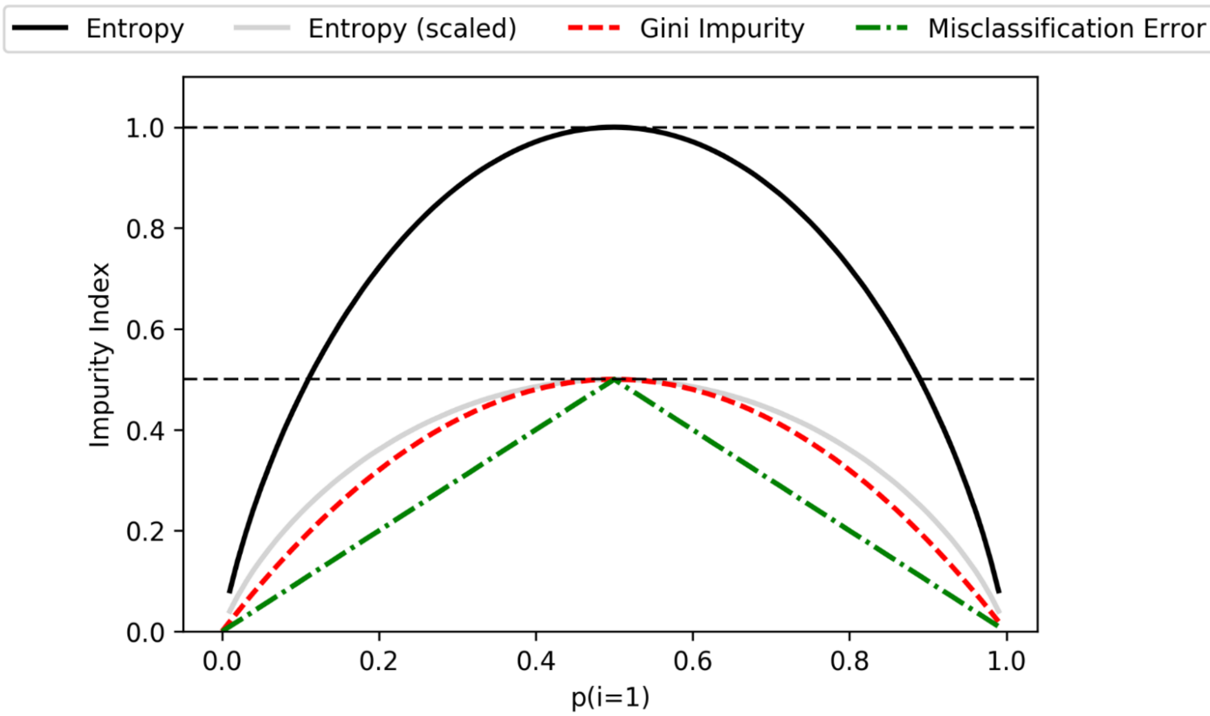
-1
1

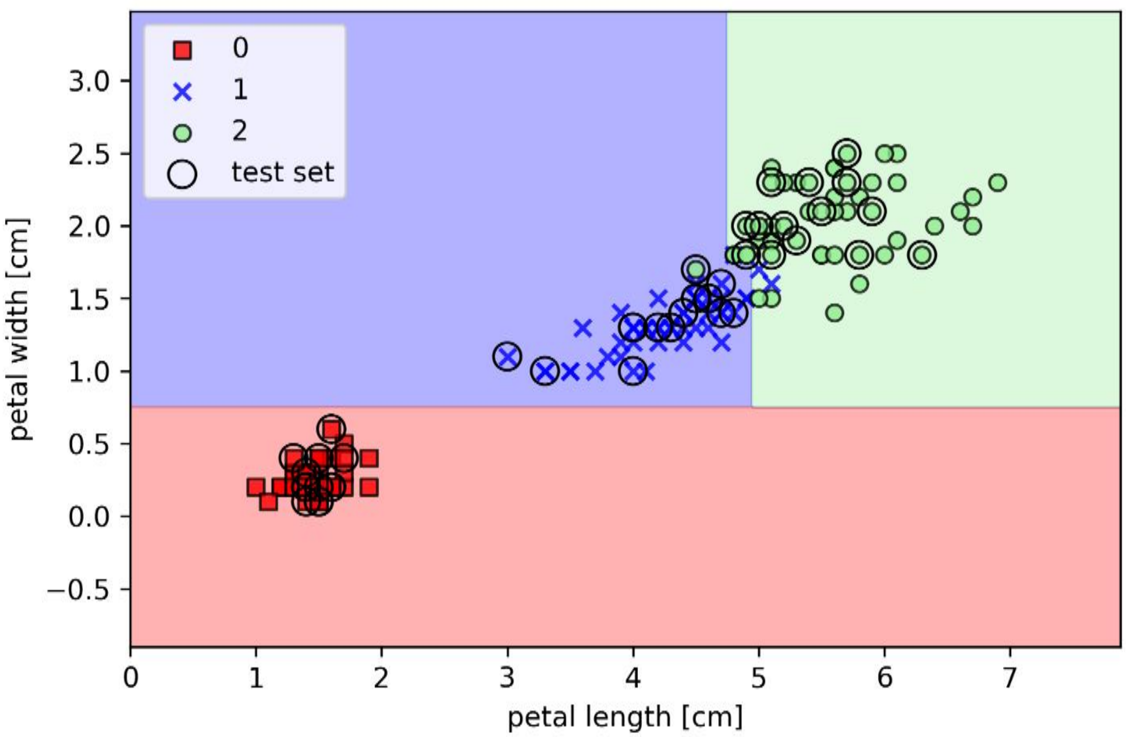

0
1
2
test set
petal width [standardized]
petal length [standardized]
−2
−1
0
1
2

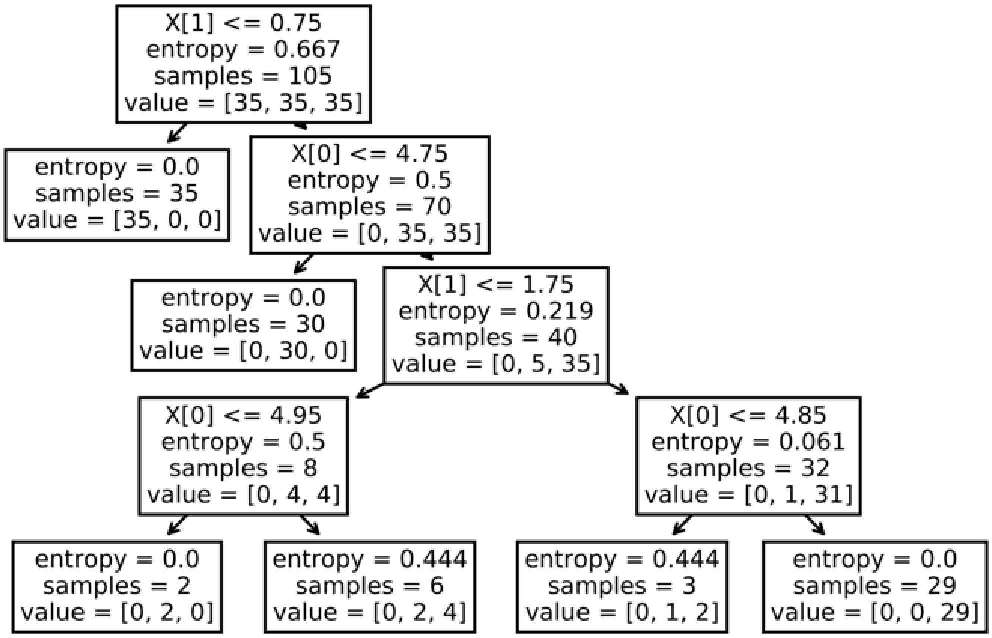

0
1
2
test set
petal width [standardized]
petal length [standardized]
−2
−1
0
1
2

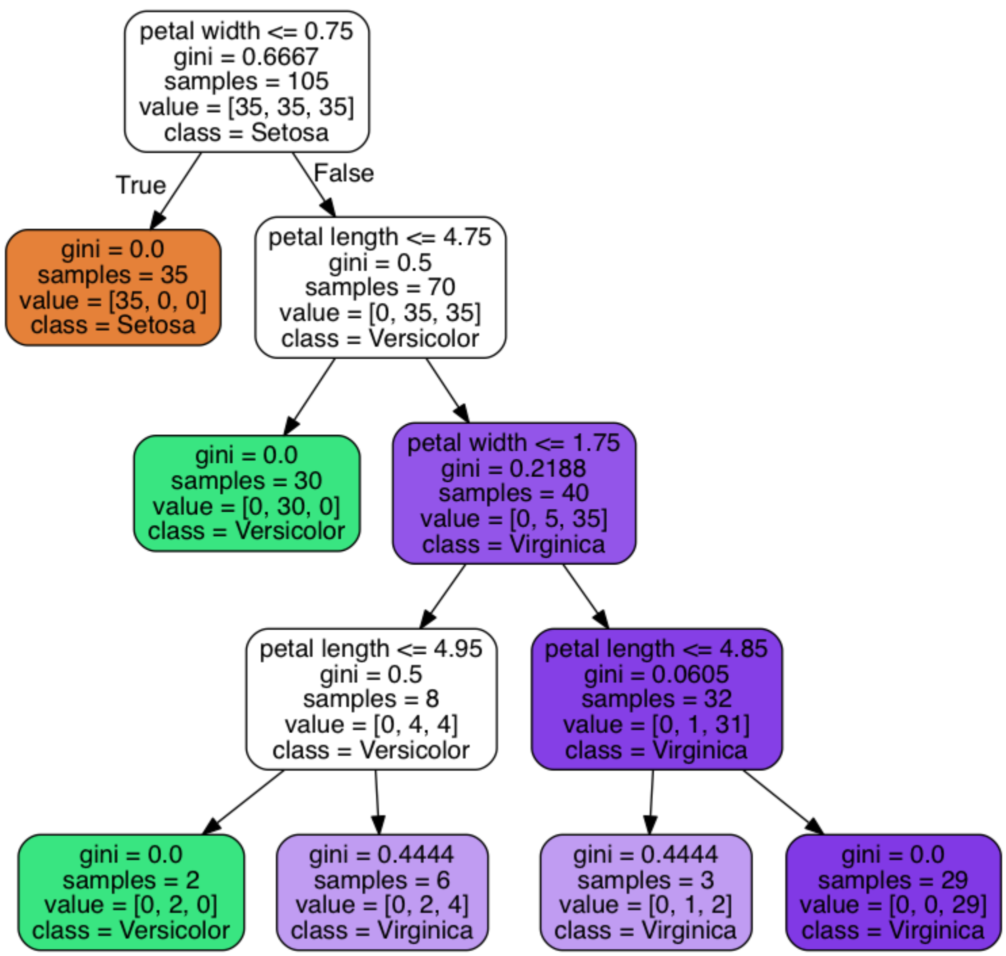

Work to do?
Yes
No
Stay in
Outlook?
Internal
node
Branch
Sunny
Over-
cast
Rainy
Go to beach
Go running
Friends busy?
Yes
No
Leaf
node
Stay in
Go to movies

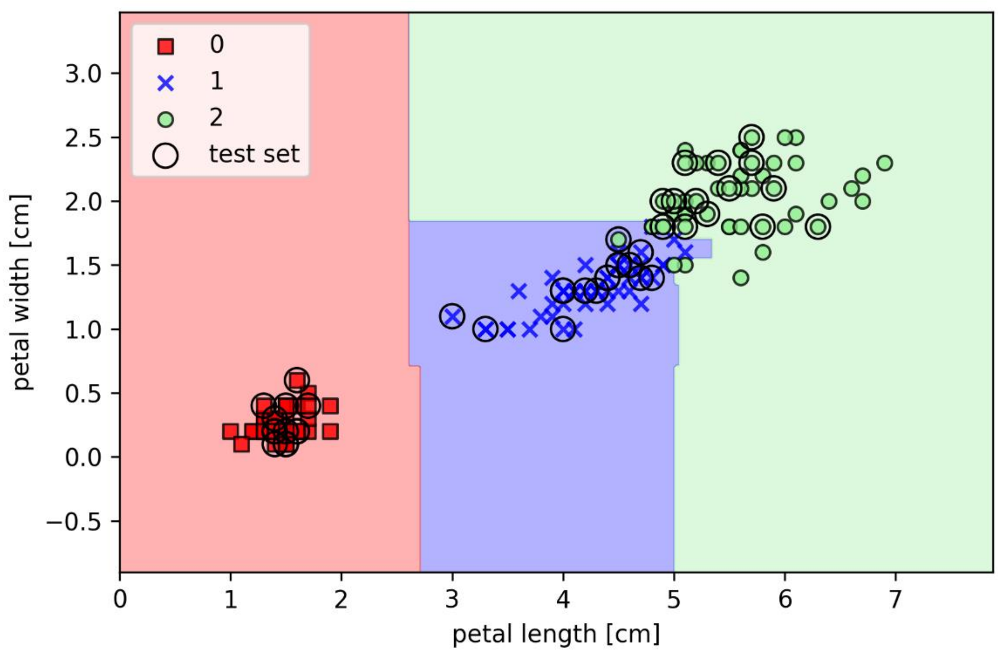

A
(40, 40)
(30, 10)
(10, 30)
B
(40, 40)
(20, 40)
(20, 0)


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

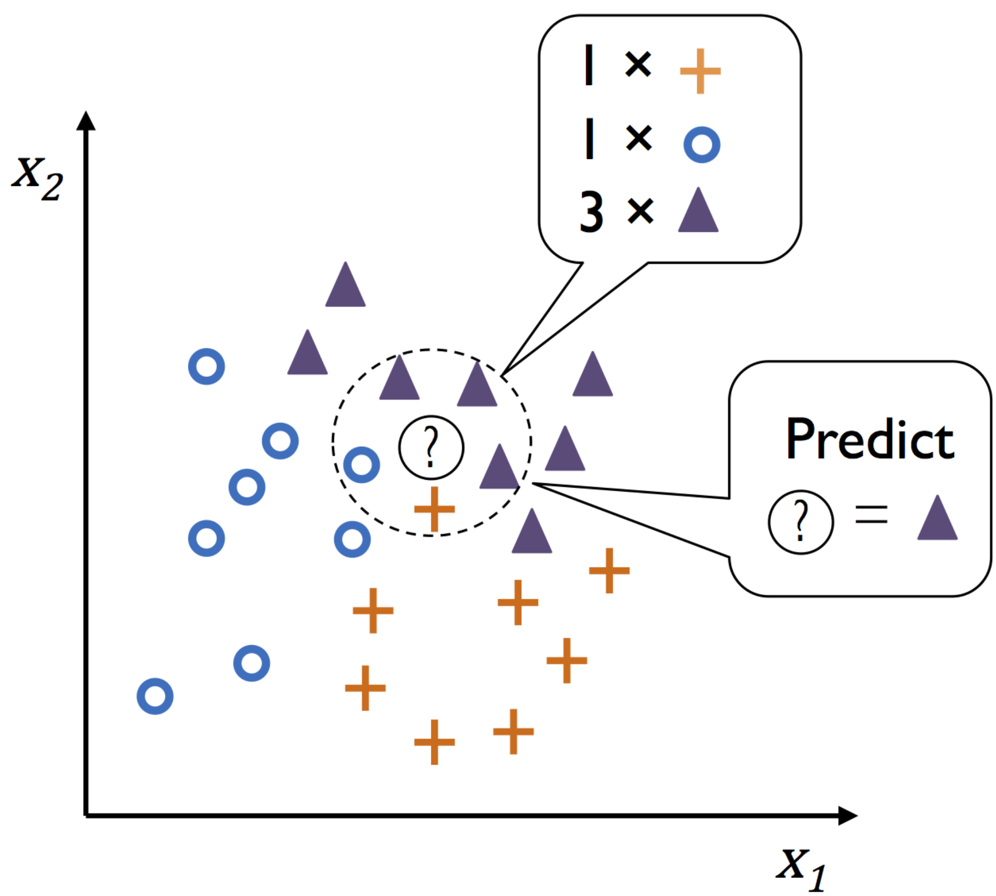

0
1
2
test set
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
petal width [cm]
0
1
2
3
4
5
6
7
petal length [cm]


X[1] <= 0.75
entropy = 0.667
samples = 105
value = [35, 35, 35]
entropy = 0.0
samples = 35
value = [35, 0, 0]
X[0] <= 4.75
entropy = 0.5
samples = 70
value = [0, 35, 35]
entropy = 0.0
samples = 30
value = [0, 30, 0]
X[1] <= 1.75
entropy = 0.219
samples = 40
value = [0, 5, 35]
X[0] <= 4.95
entropy = 0.5
samples = 8
value = [0, 4, 4]
X[0] <= 4.85
entropy = 0.061
samples = 32
value = [0, 1, 31]
entropy = 0.0
samples = 2
value = [0, 2, 0]
entropy = 0.444
samples = 6
value = [0, 2, 4]
entropy = 0.444
samples = 3
value = [0, 1, 2]
entropy = 0.0
samples = 29
value = [0, 0, 29]

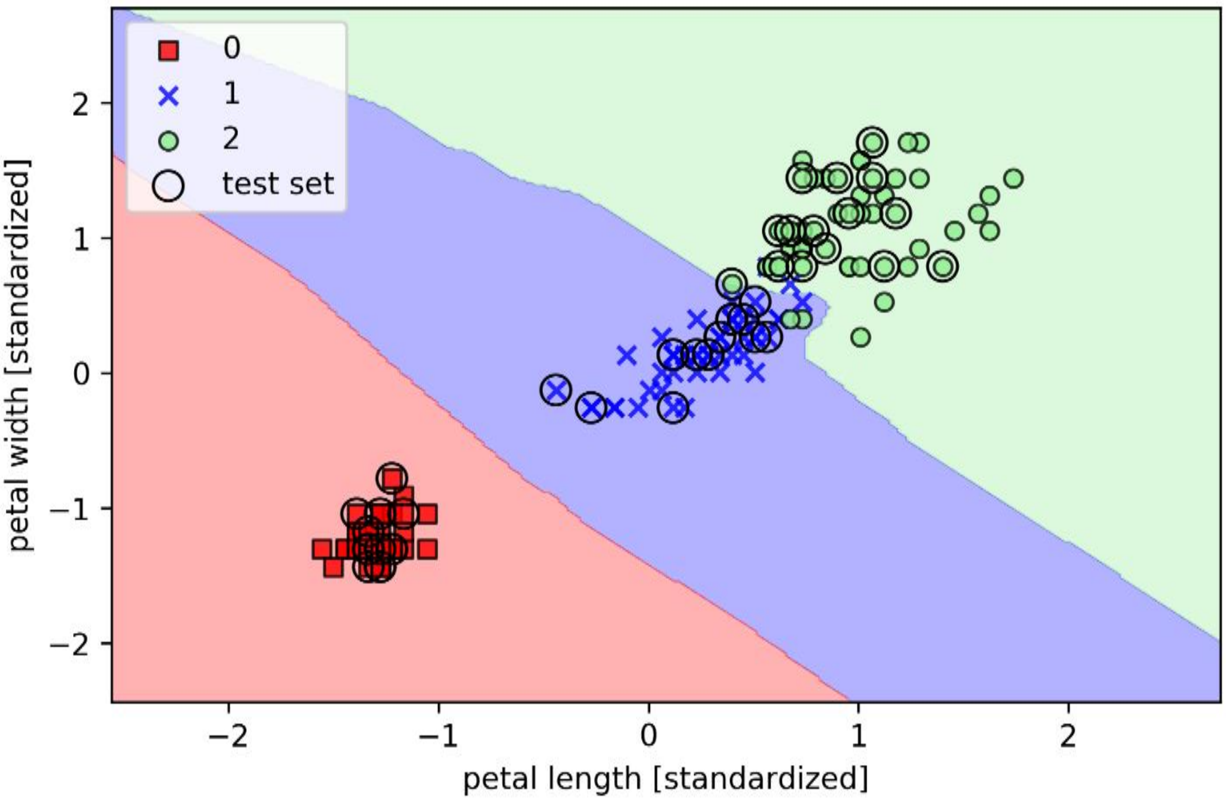

petal width <= 0.75
gini = 0.6667
samples = 105
value = [35, 35, 35]
class = Setosa
True
False
gini = 0.0
samples = 35
value = [35, 0, 0]
class = Setosa
petal length <= 4.75
gini = 0.5
samples = 70
value = [0, 35, 35]
class = Versicolor
gini = 0.0
samples = 30
value = [0, 30, 0]
class = Versicolor
petal 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
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
petal width [cm]
0
1
2
3
4
5
6
7
petal length [cm]

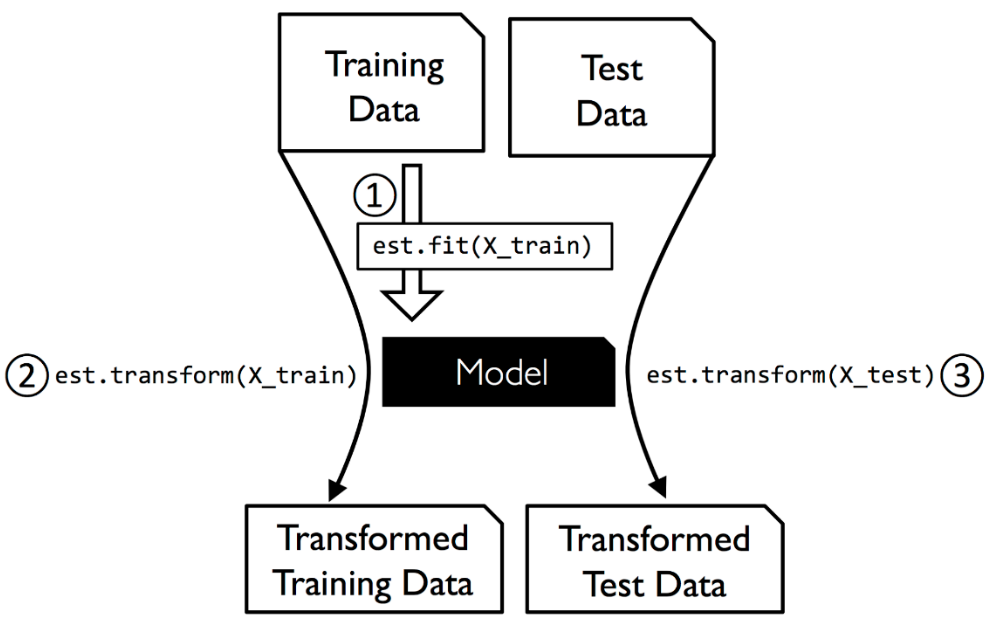


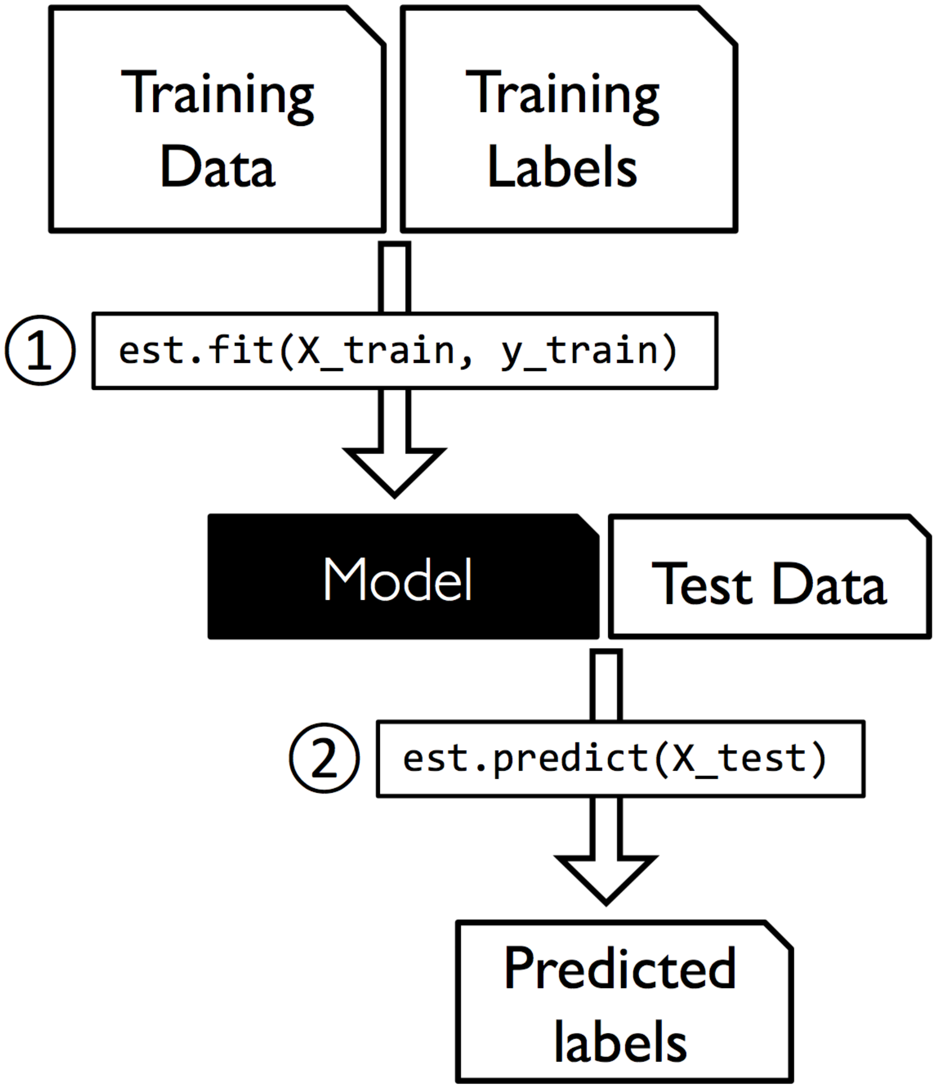
1 × +
1 × ○
3 × ▲
x_2
Predict
? = ▲
x_1


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

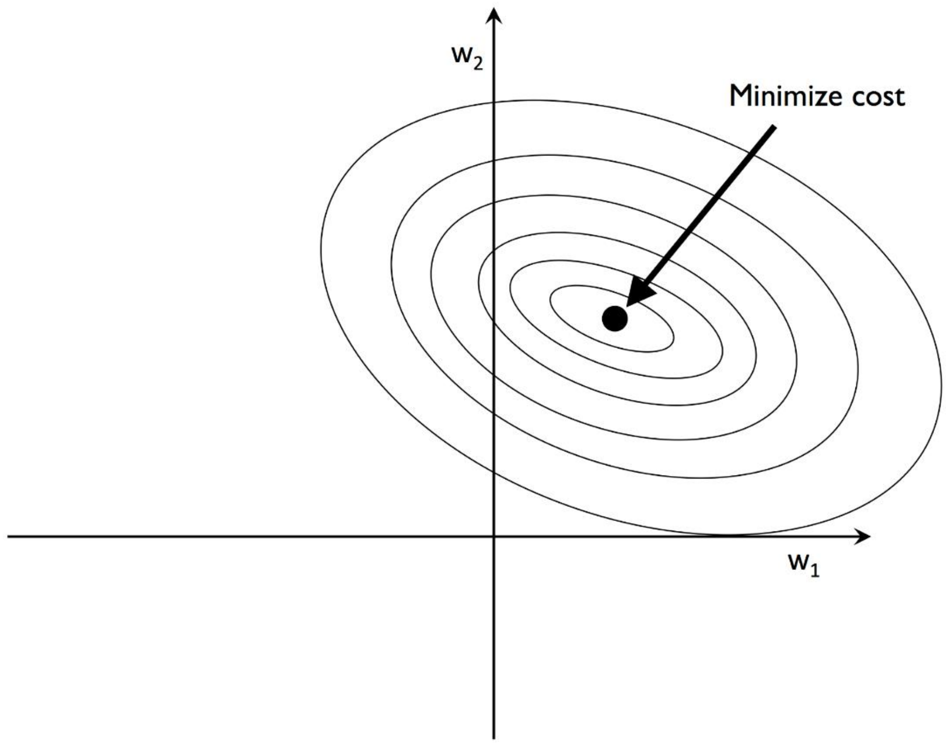
Chapter 04: Building Good Training Datasets – Data Preprocessing

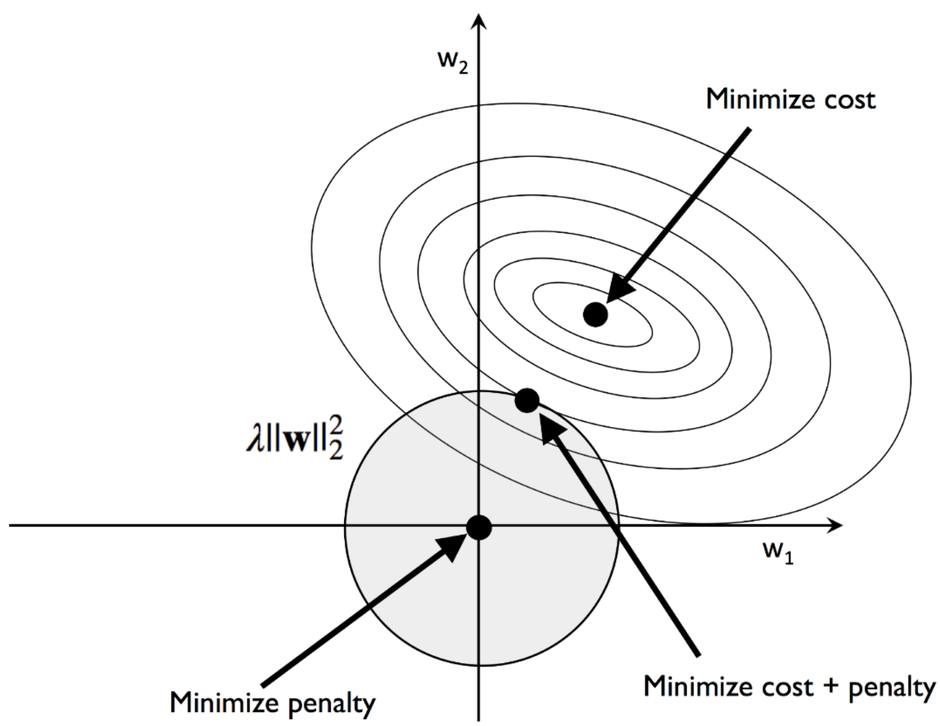
	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	7.5	8.0
2	10.0	11.0	12.0	6.0

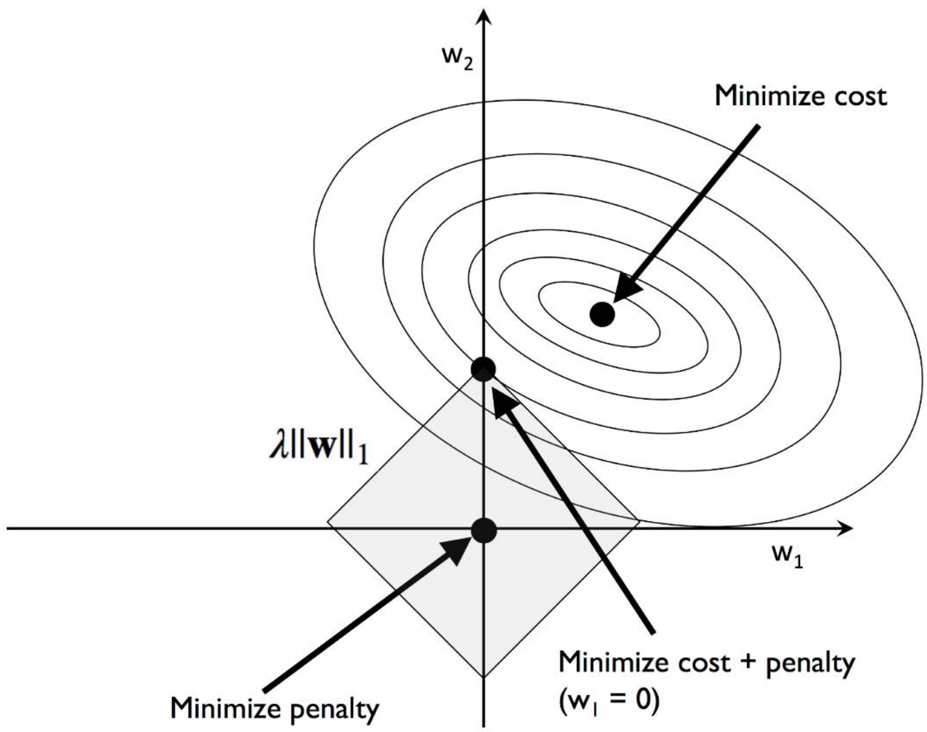




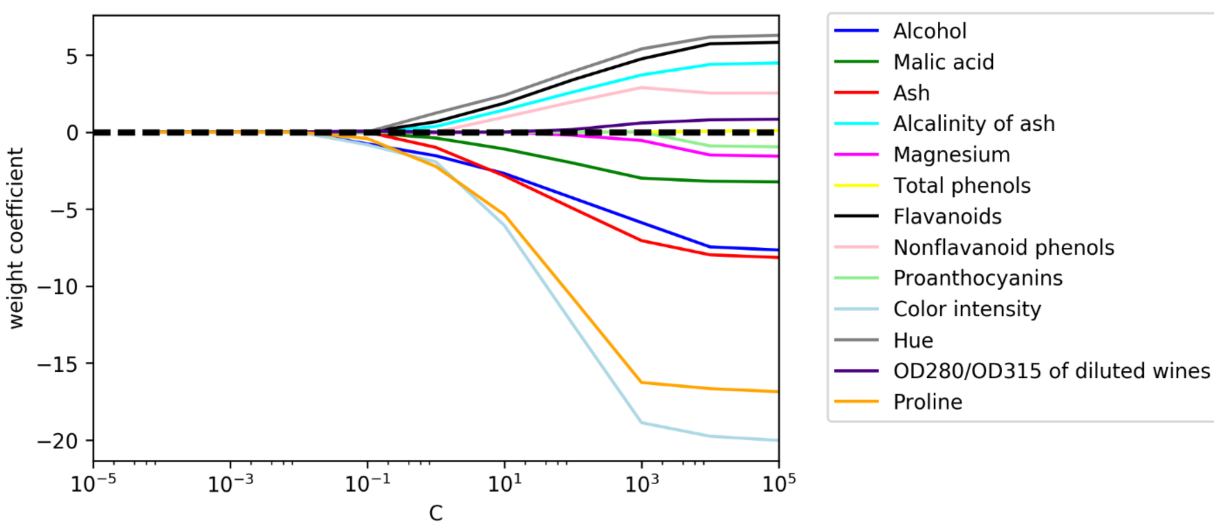
	Class label	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	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

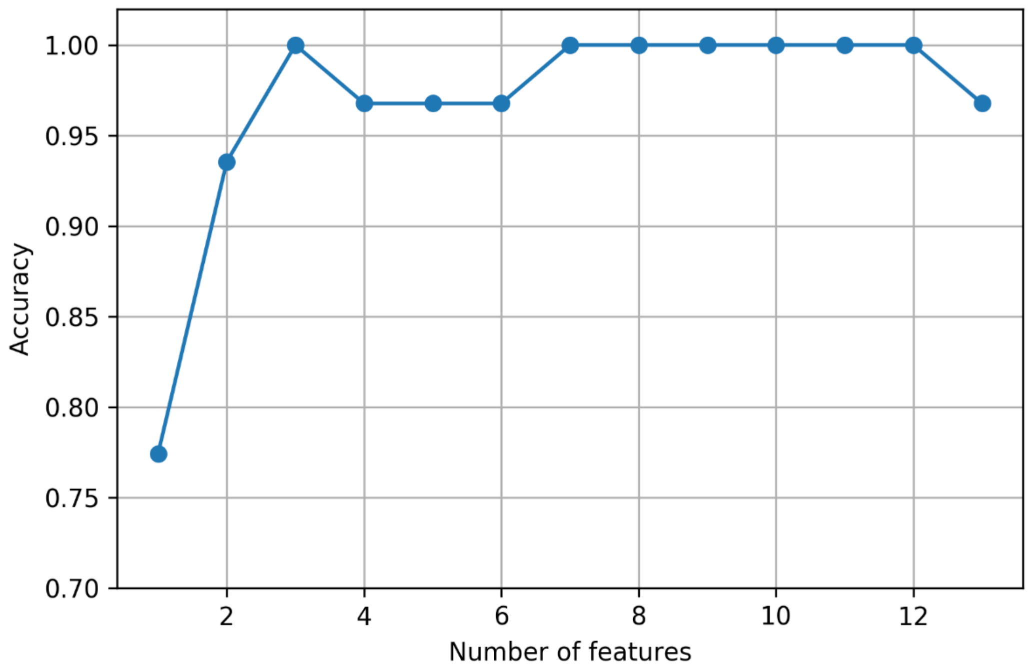

w_2
Minimize cost
$\lambda \|\mathbf{w}\|_2^2$
w_1
Minimize penalty
Minimize cost + penalty



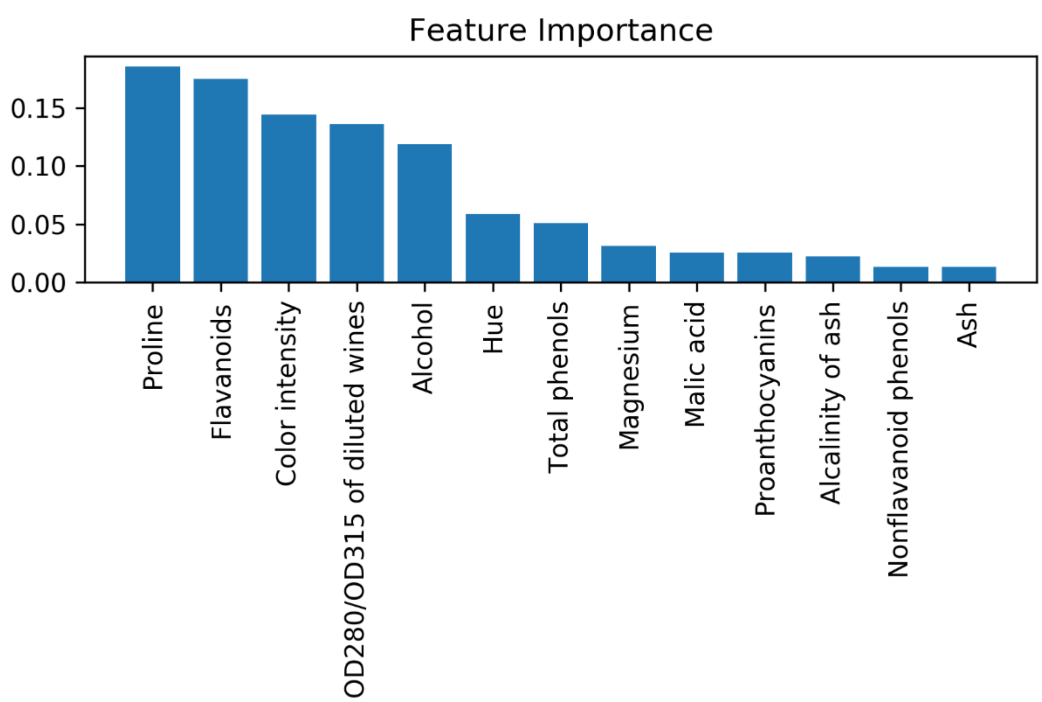
w_2
Minimize cost
$\lambda\|\mathbf{w}\|_1$
w_1
Minimize cost + penalty
(w_1 = 0)
Minimize penalty



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

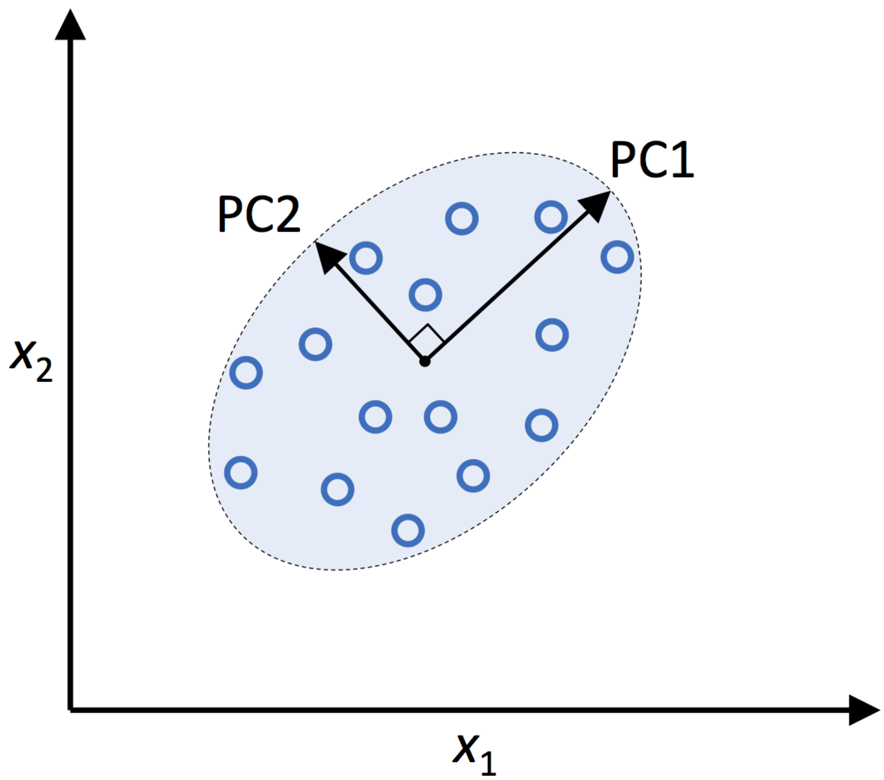


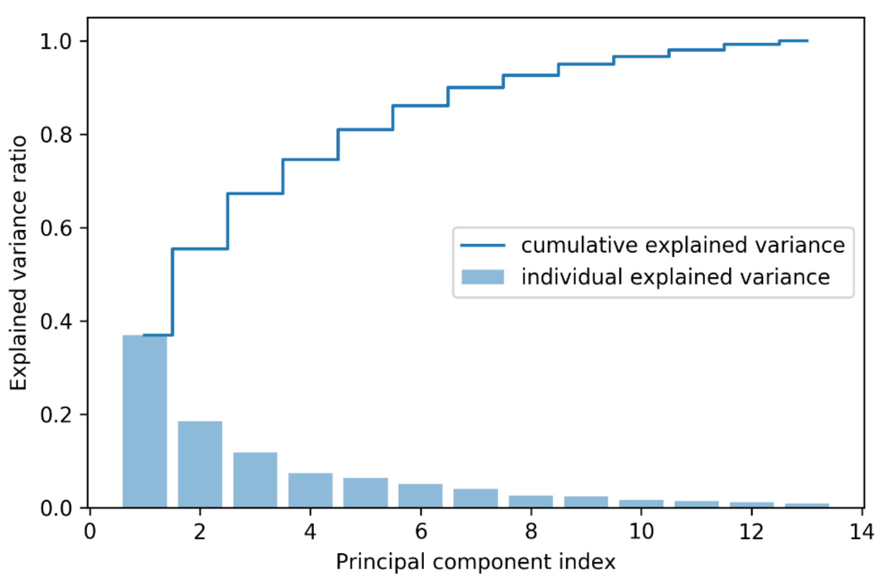
1.00
0.95
0.90
0.85
0.80
0.75
0.70
Accuracy
2
4
6
8
10
12
Number of features

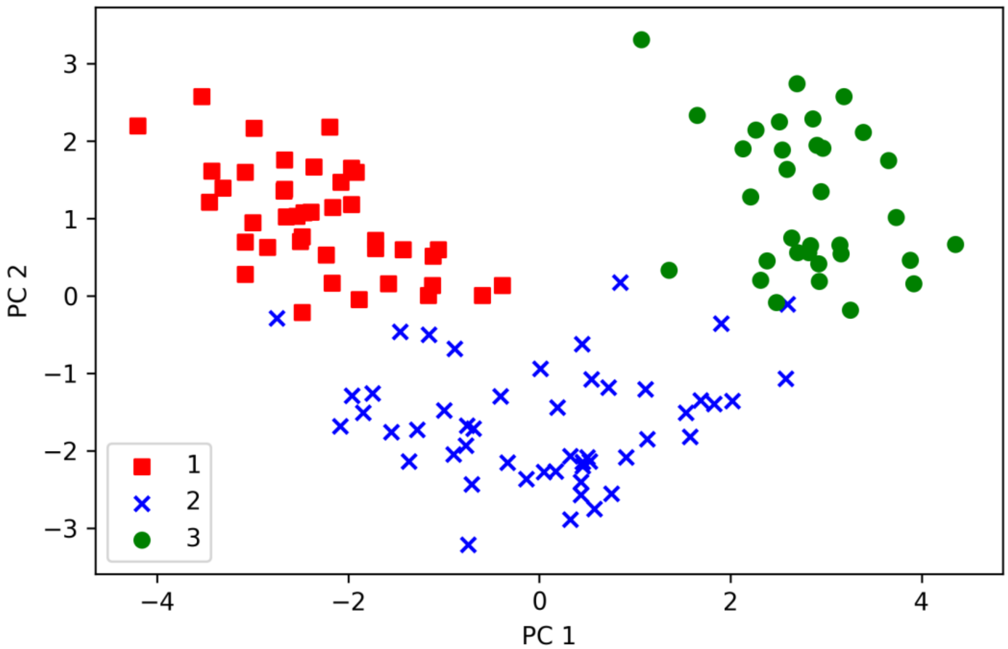


Feature Importance
0.15
0.10
0.05
0.00
Proline
Flavanoids
Color intensity
OD280/OD315 of diluted wines
Alcohol
Hue
Total phenols
Magnesium
Malic acid
Proanthocyanins
Alcalinity of ash
Nonflavanoid phenols
Ash

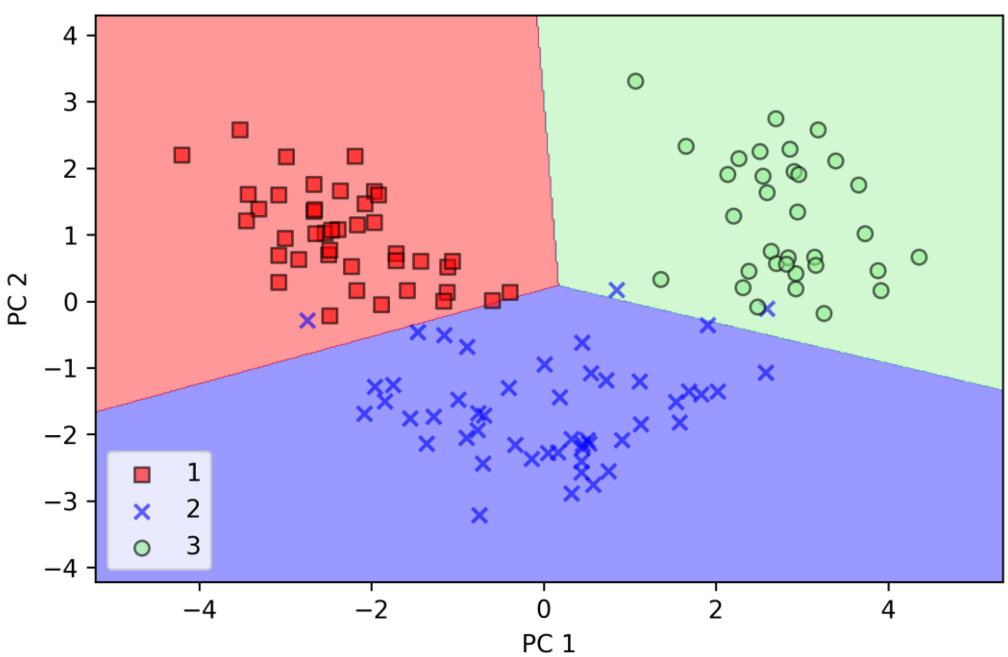
Chapter 05: Compressing Data via Dimensionality Reduction



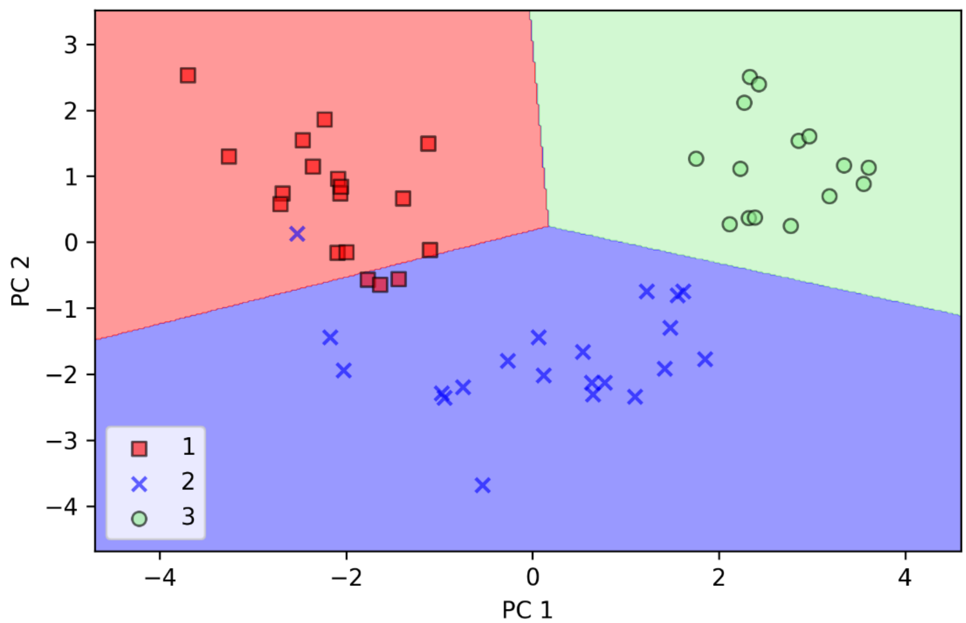




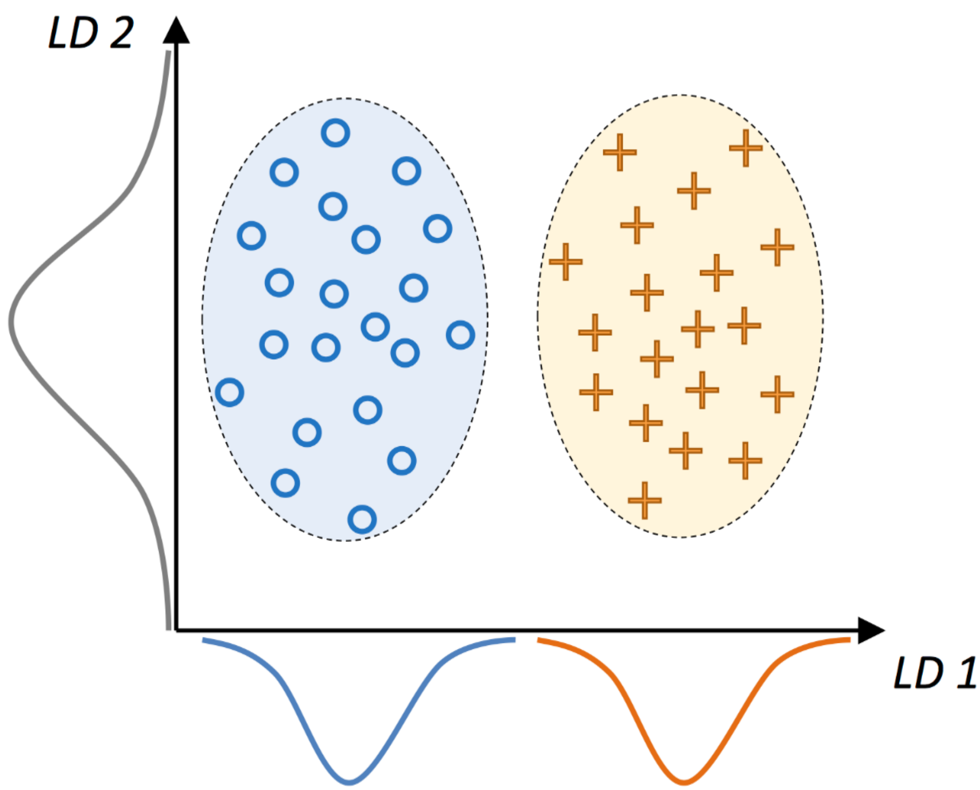
PC 2
PC 1
1
2
3



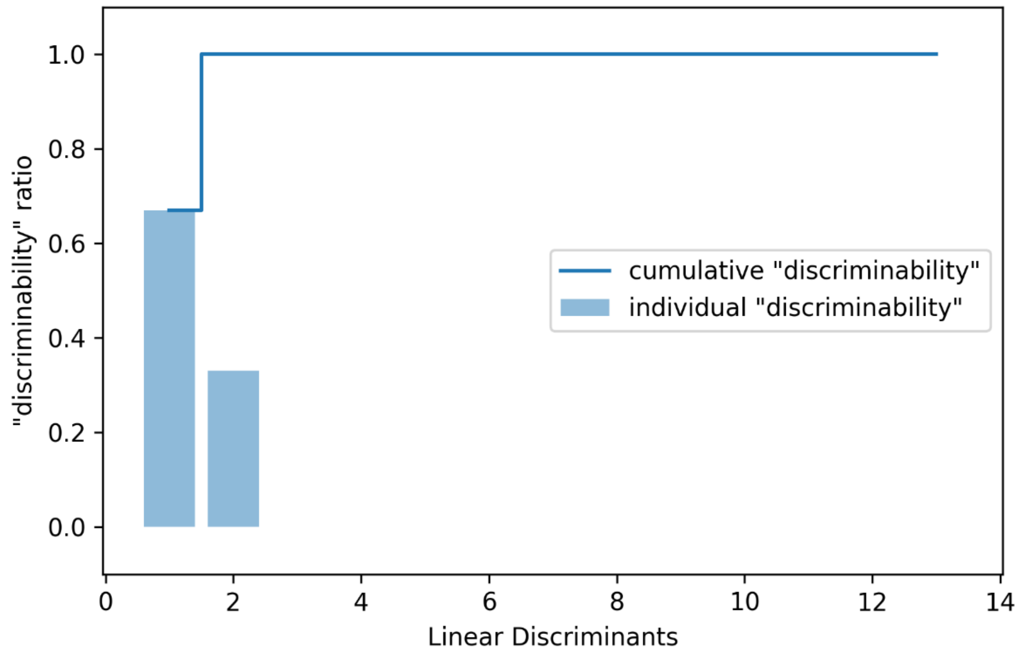
PC 2
PC 1
1
2
3



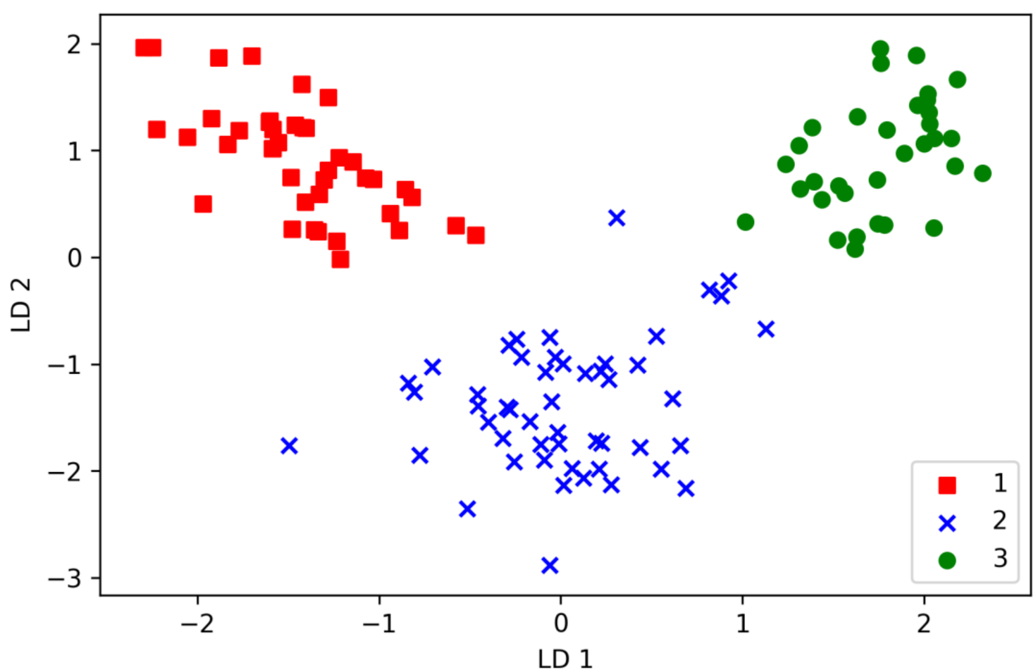
3
2
1
0
−1
−2
−3
−4
PC 2
−4
−2
0
2
4
PC 1
1
2
3



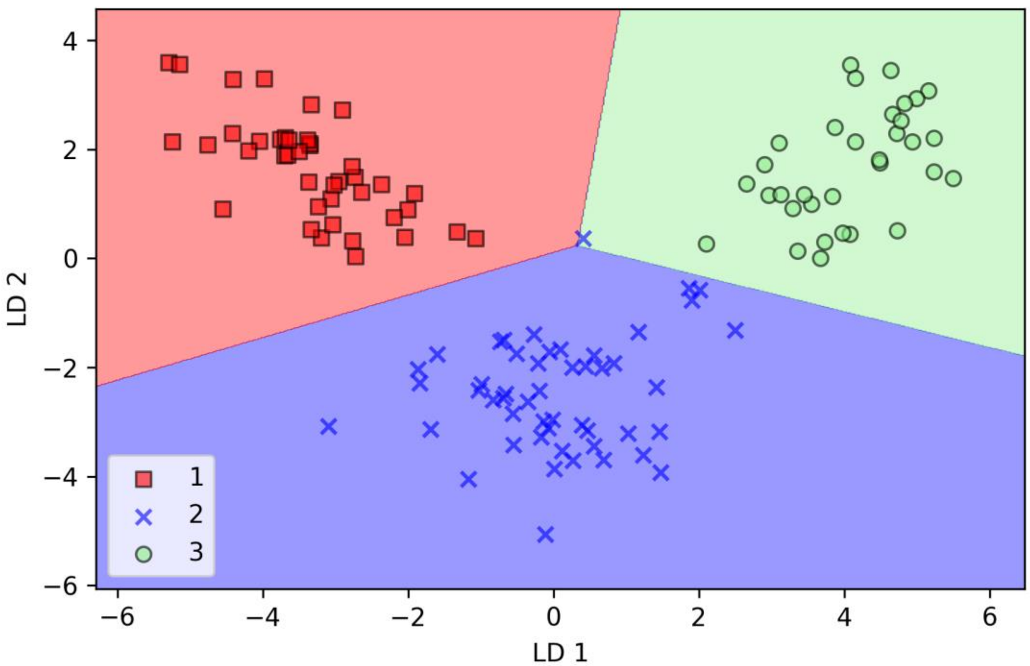
LD 2
LD 1



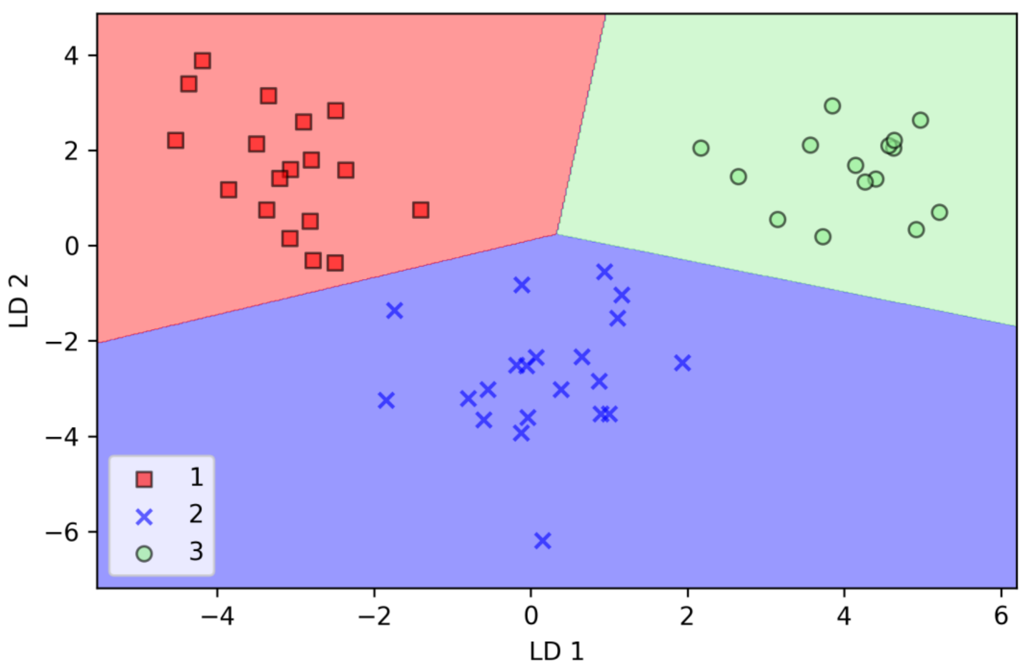
"discriminability" ratio
Linear Discriminants
cumulative "discriminability"
individual "discriminability"



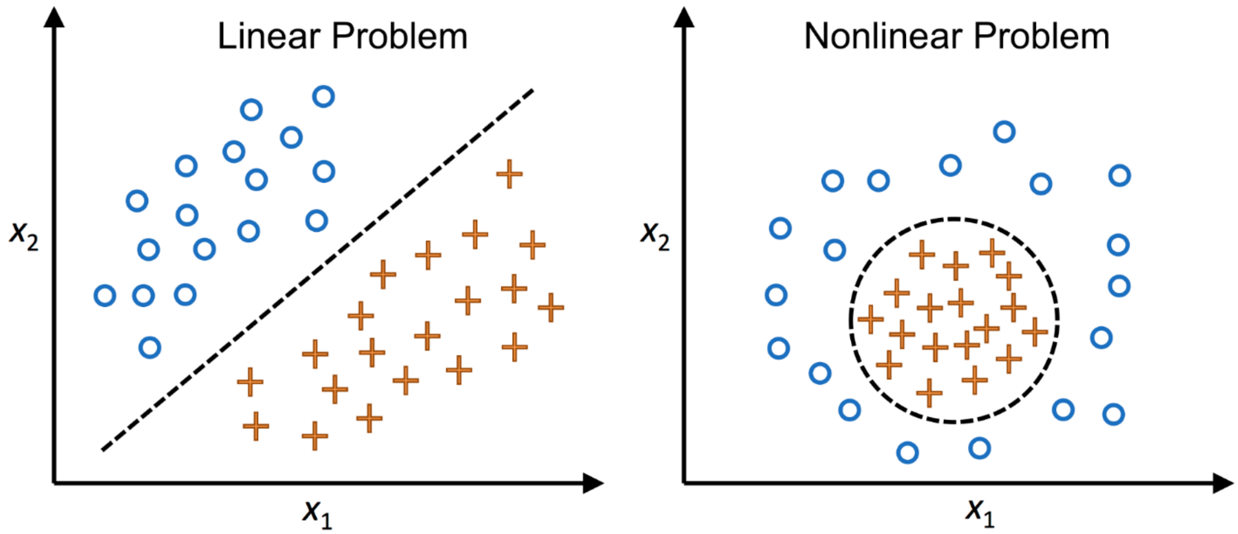
LD 1
LD 2
1
2
3

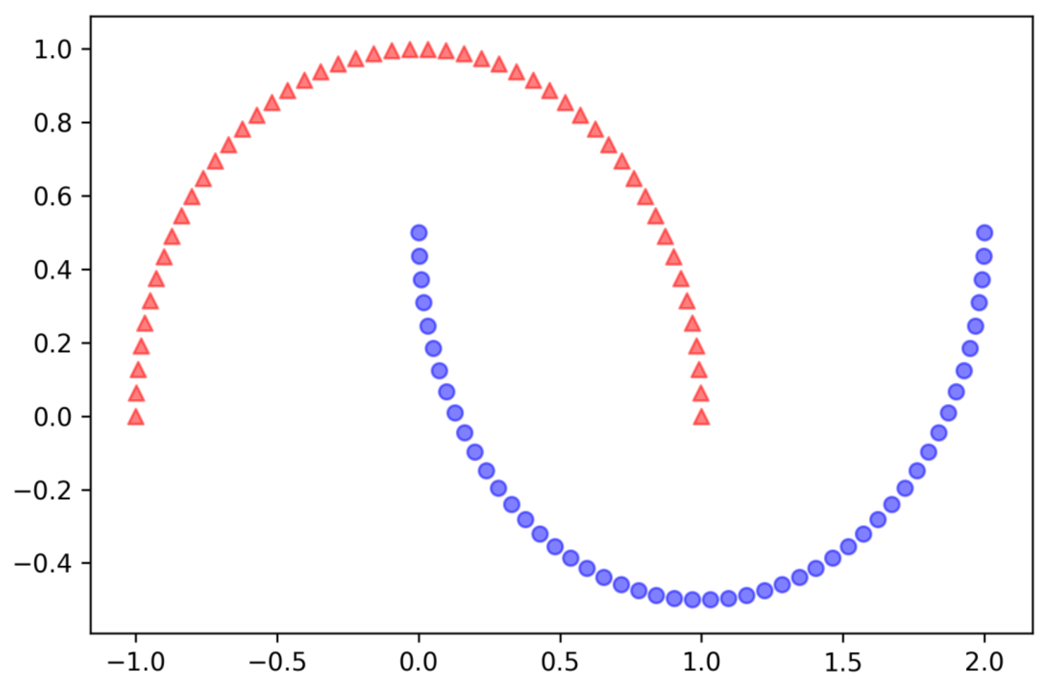


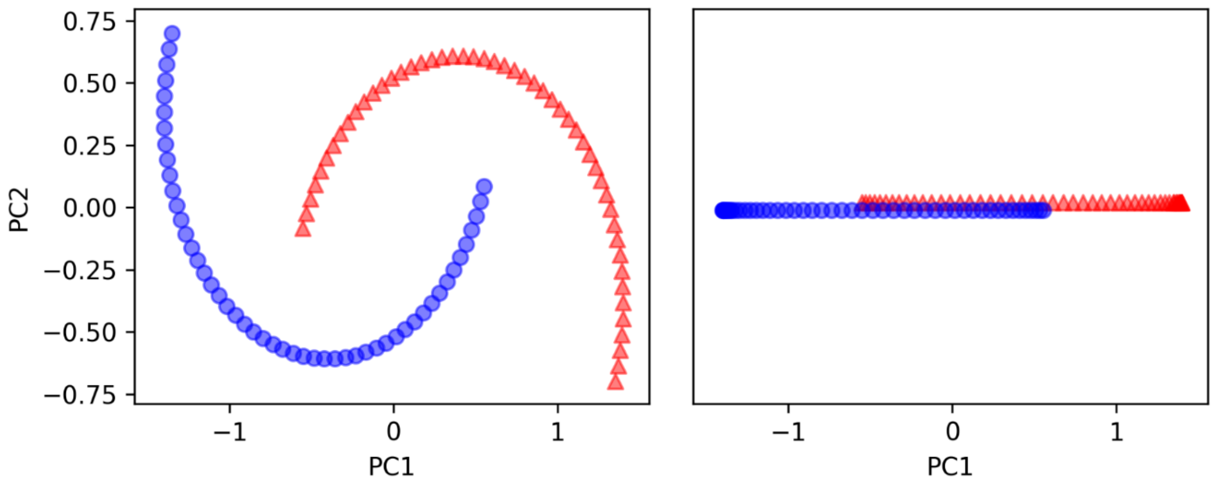
4
2
0
−2
−4
−6
LD 2
−6
−4
−2
0
2
4
6
LD 1
1
2
3



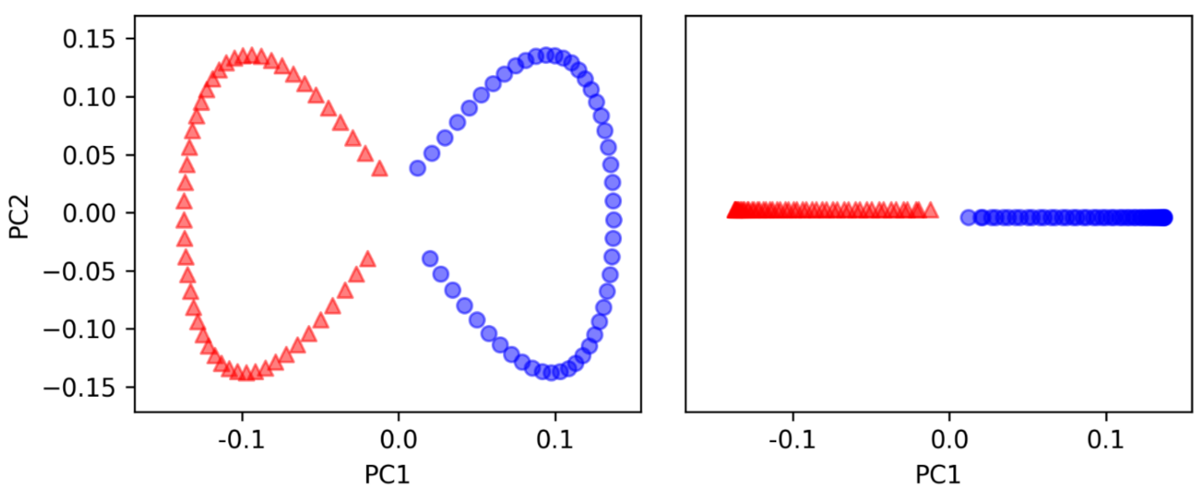
4
2
0
−2
−4
−6
LD 2
−4
−2
0
2
4
6
LD 1
1
2
3


Linear Problem
x_2
x_1
Nonlinear Problem
x_2
x_1

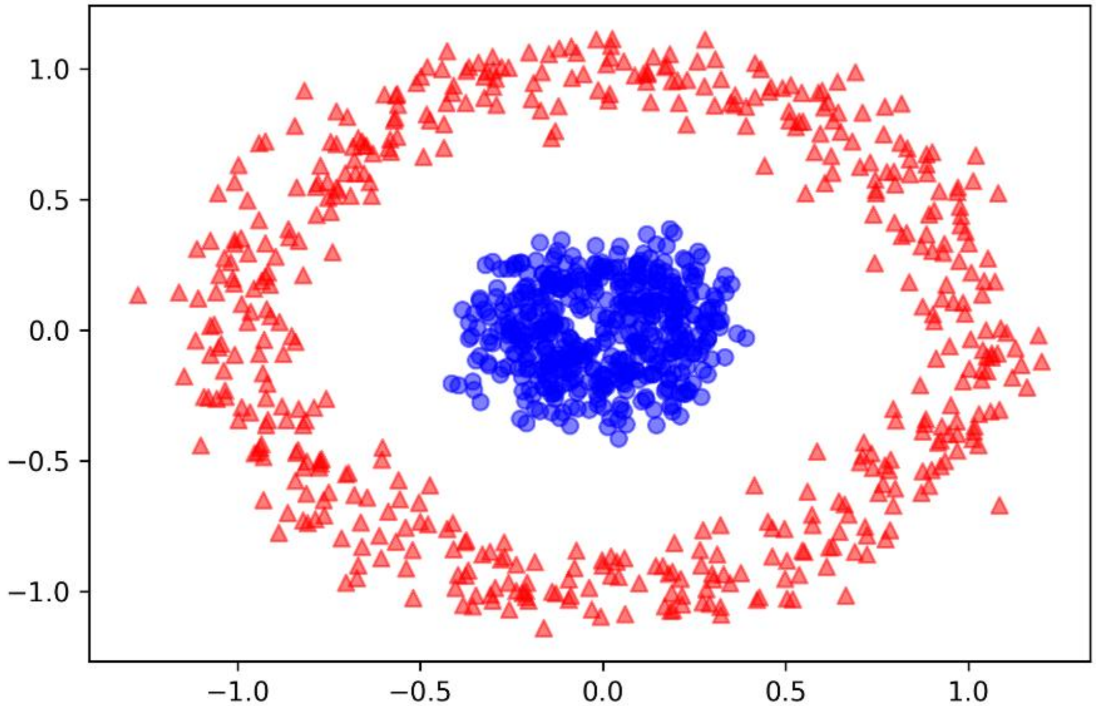

1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

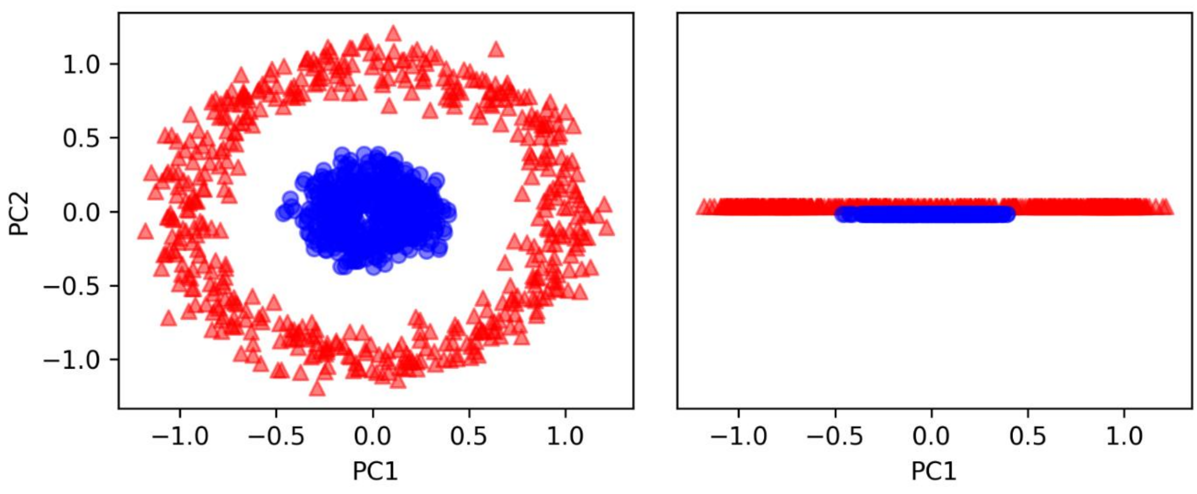


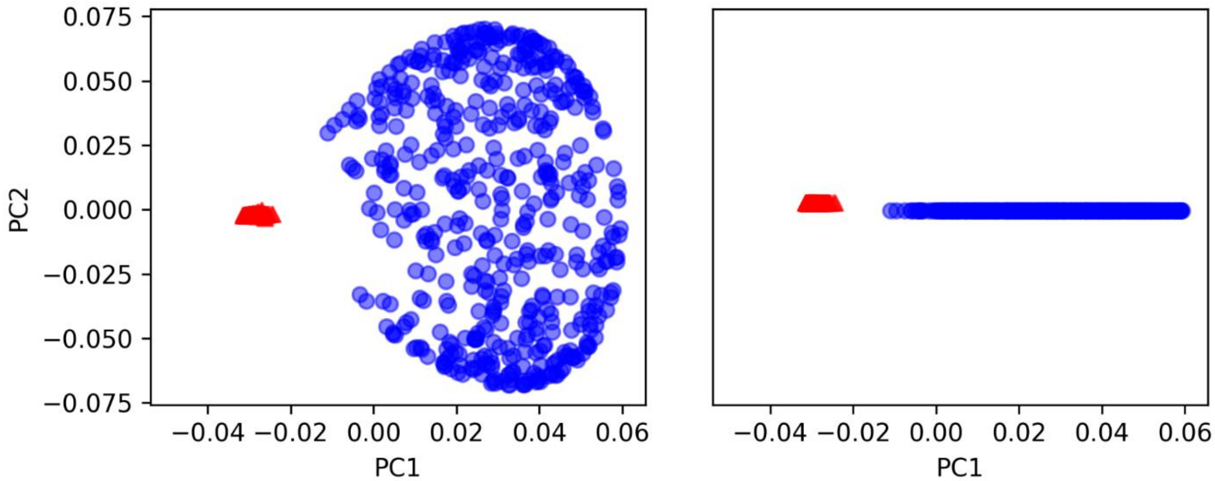
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
PC2
−1
0
1
PC1
−1
0
1
PC1

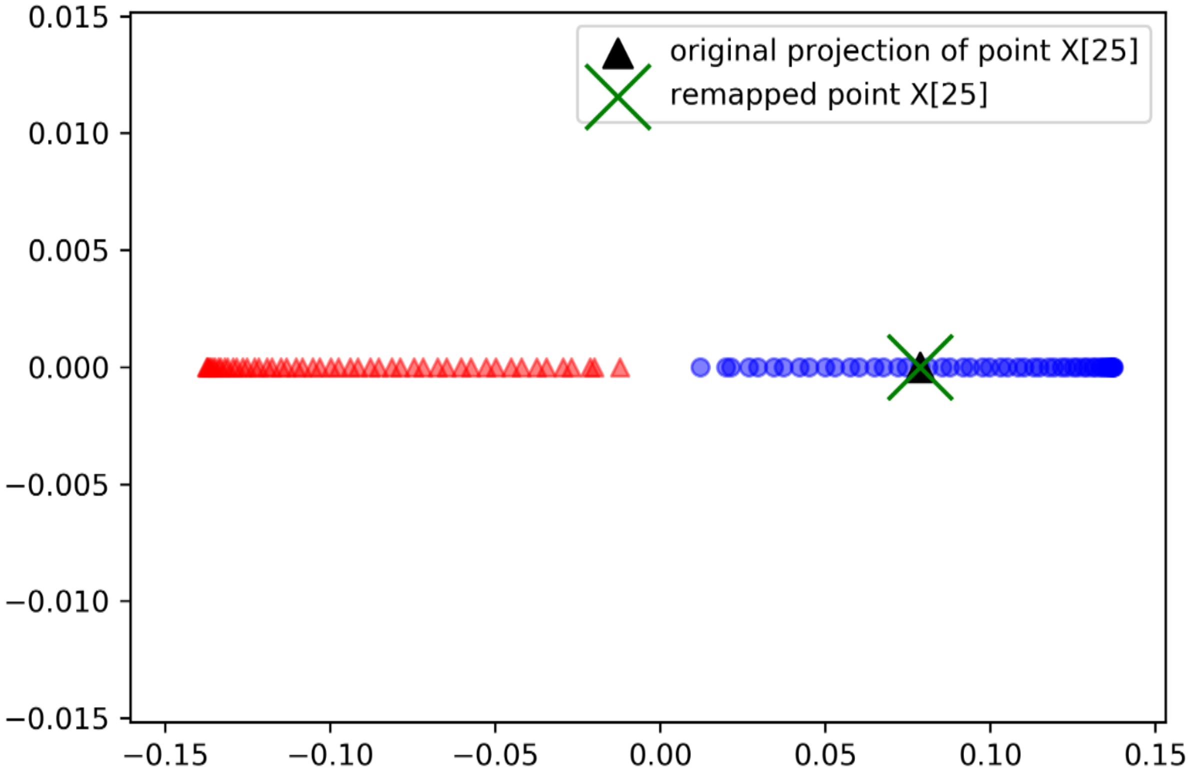


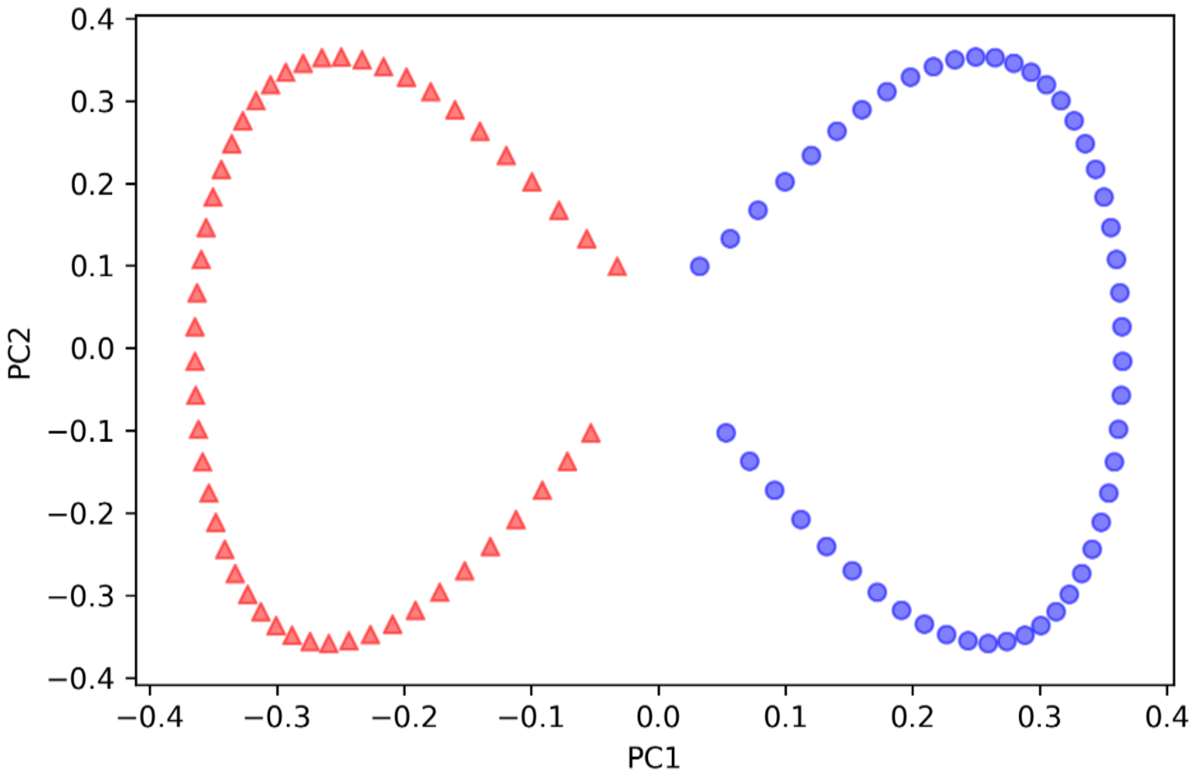
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
PC2
-0.1
0.0
0.1
PC1
-0.1
0.0
0.1
PC1


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

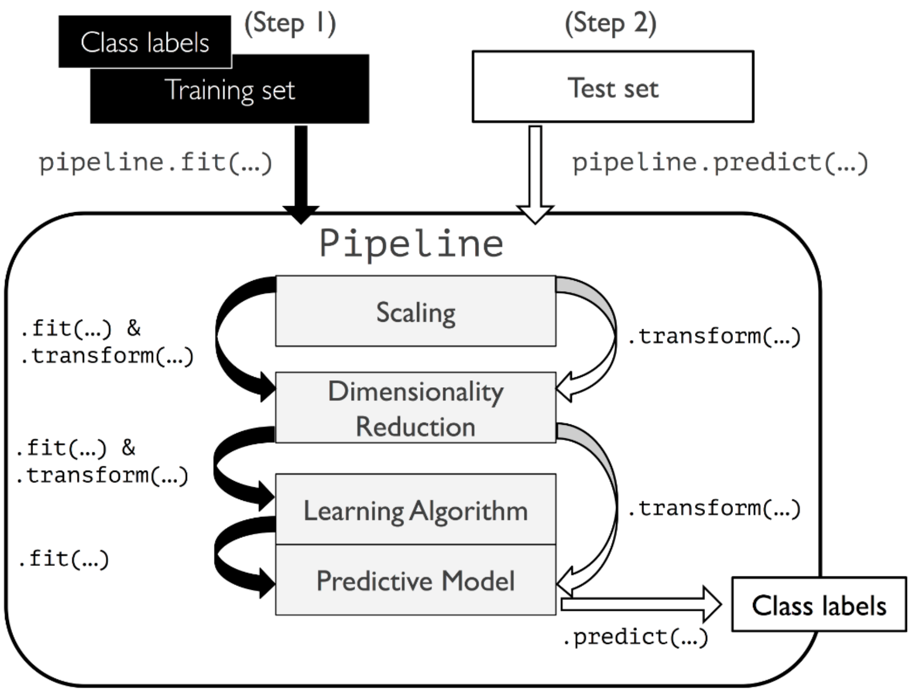

PC2
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
PC1
PC1

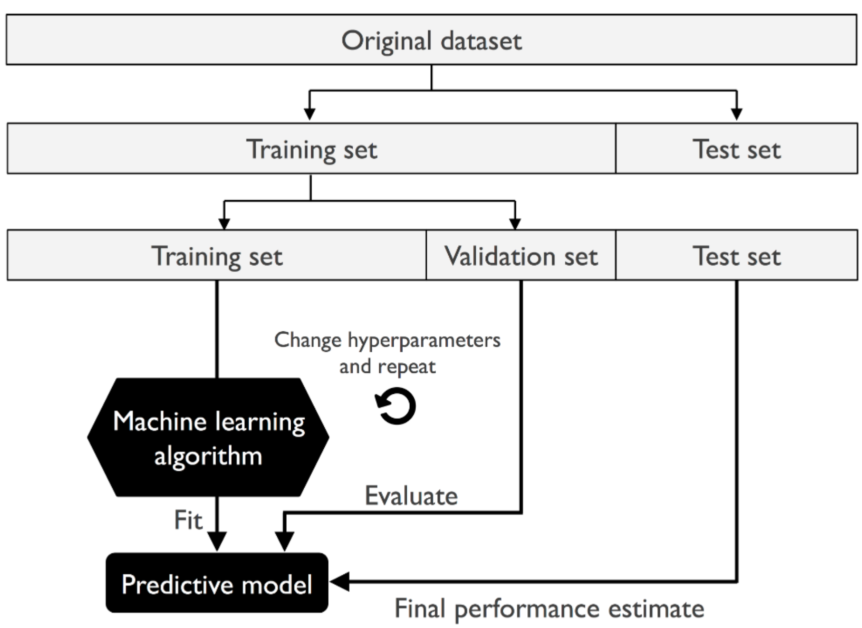

PC2
PC1
PC1

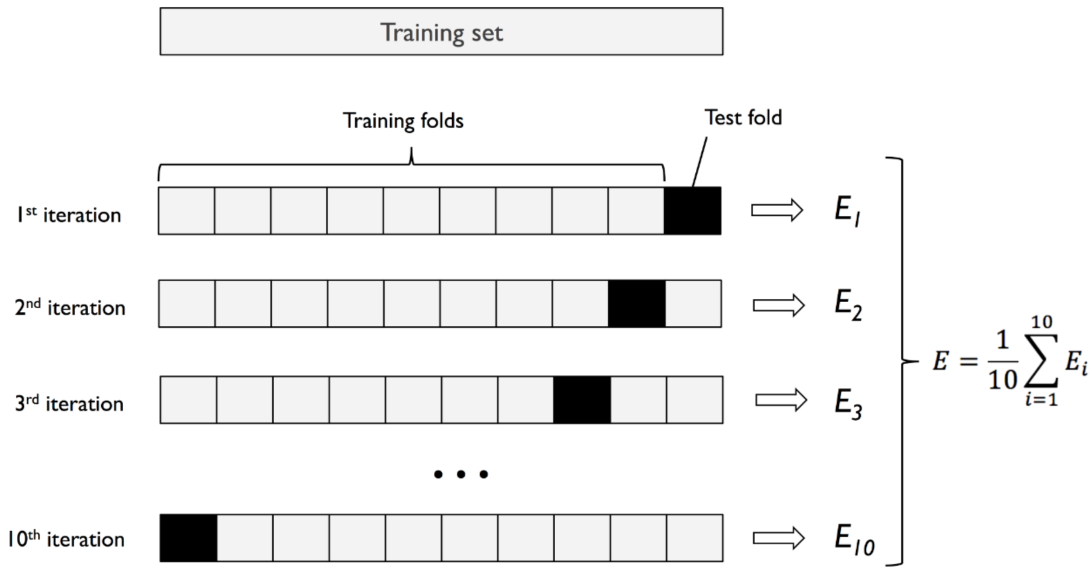

original projection of point X[25]
remapped point X[25]


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

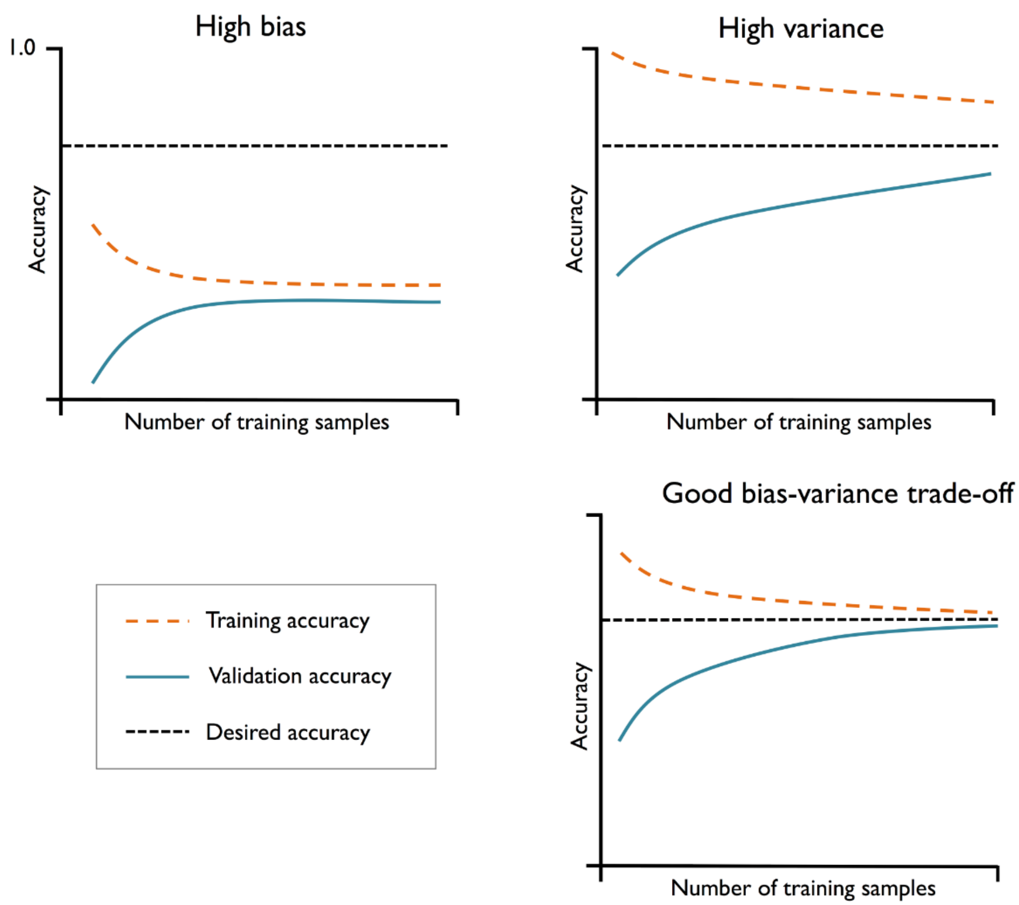
Chapter 06: Learning Best Practices for Model Evaluation and Hyperparameter Tuning



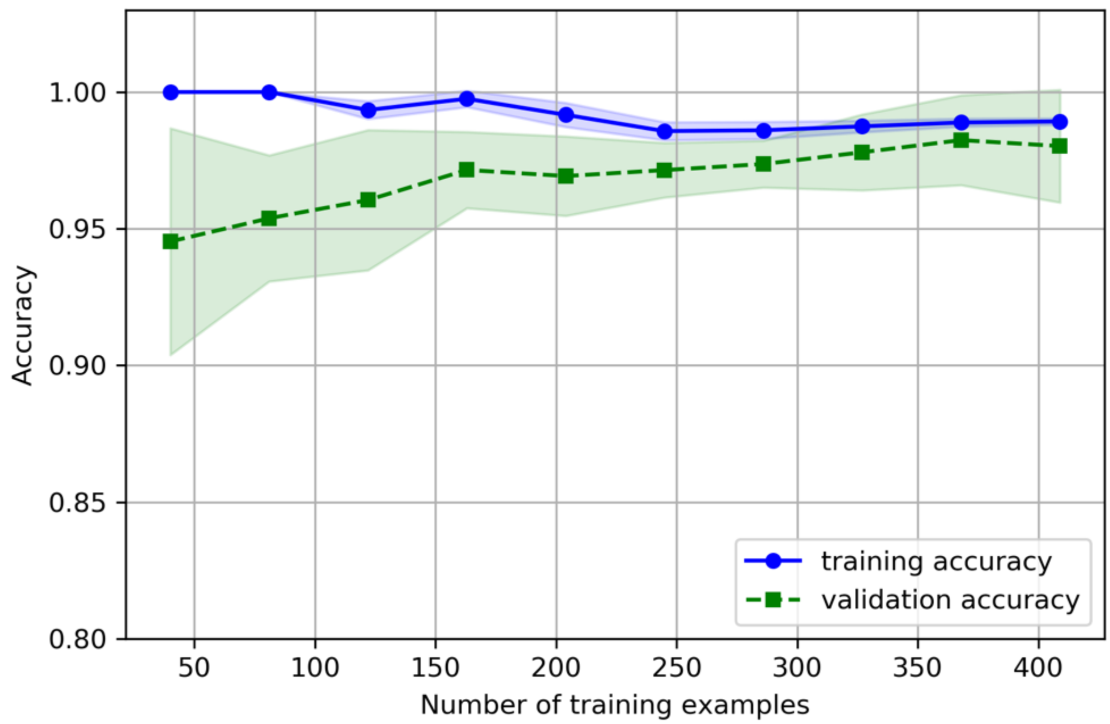


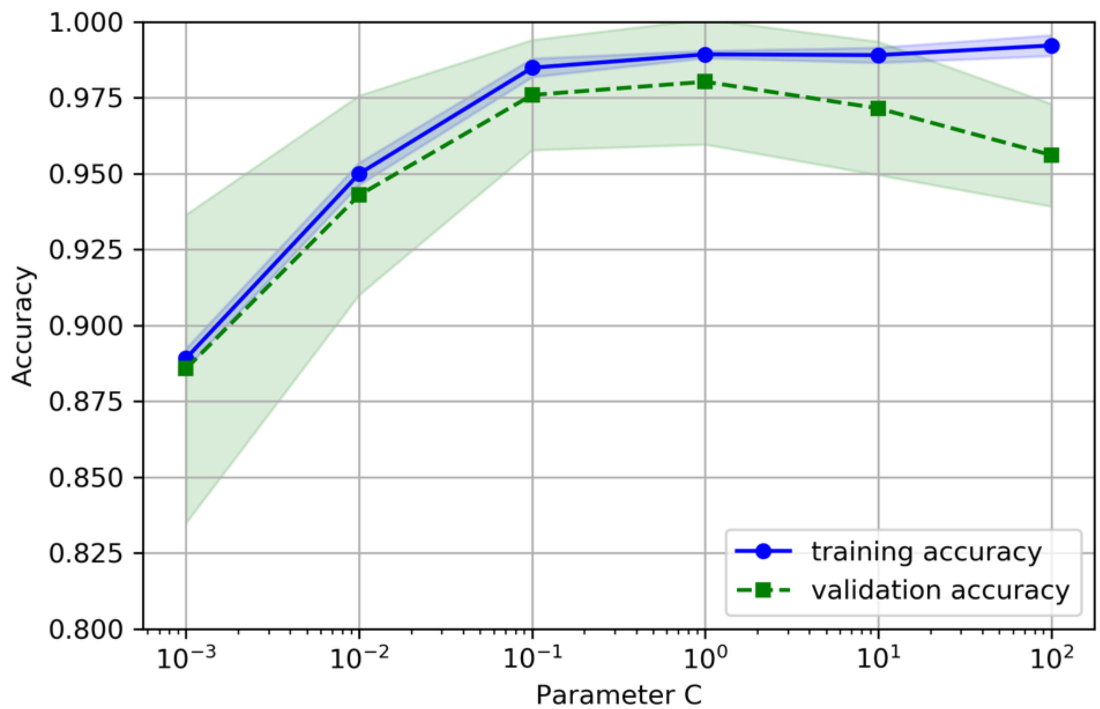


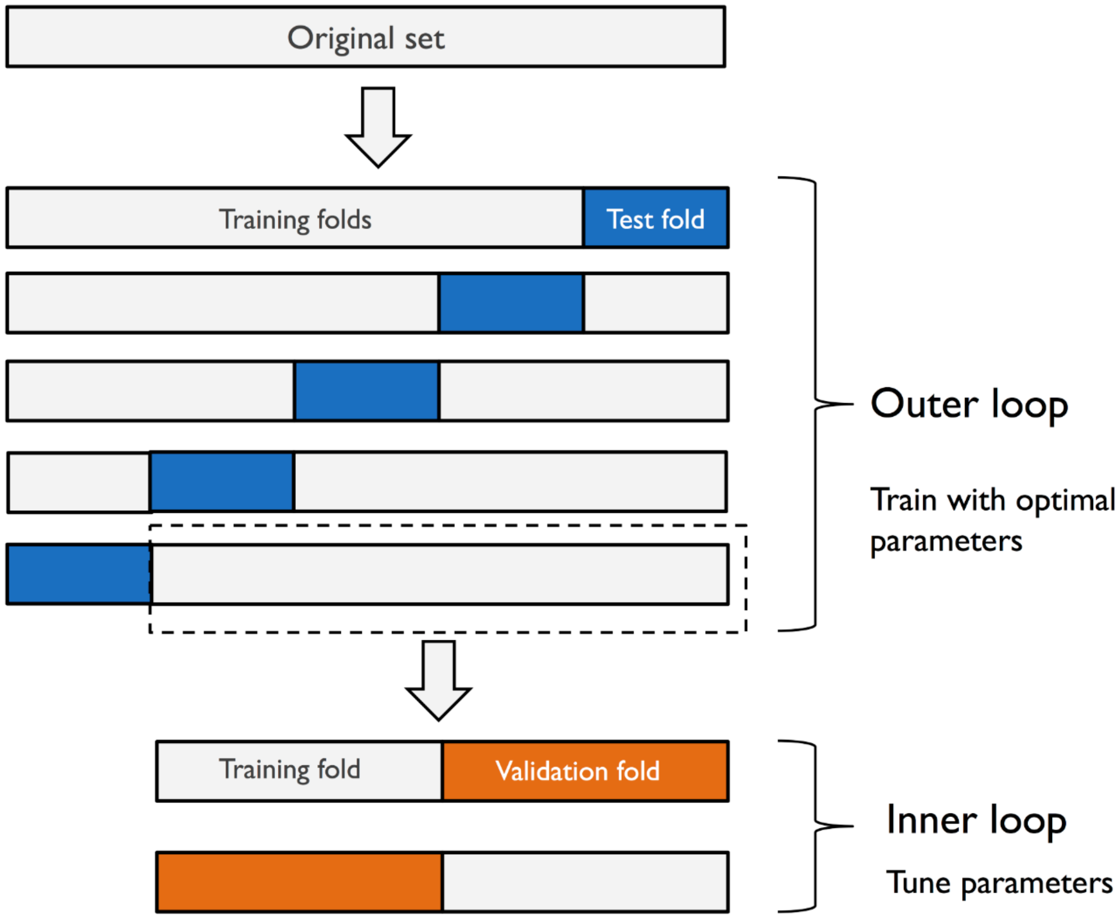
Training set
Training folds
Test fold
1st iteration
2nd iteration
3rd iteration
10th iteration
E_1
E_2
E_3
E_{10}
$E = \frac{1}{10}\sum_{i=1}^{10} E_i$

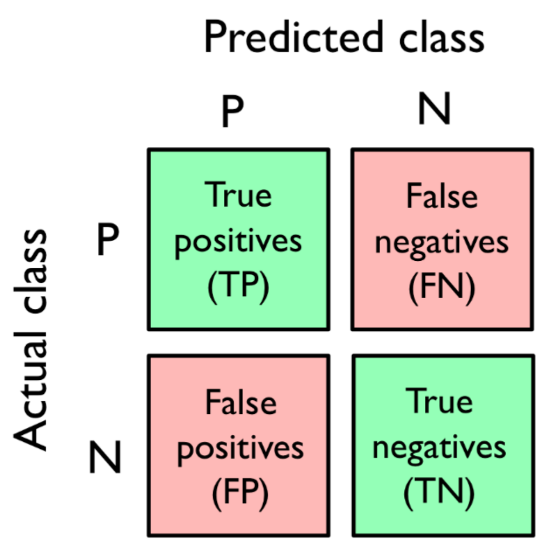


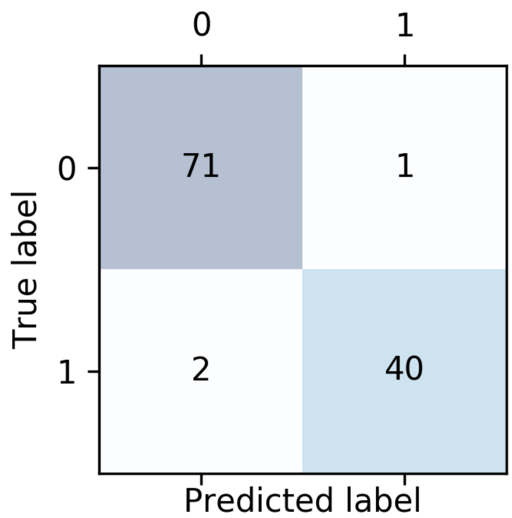
High bias
1.0
Accuracy
Number of training samples
High variance
Accuracy
Number of training samples
Good bias-variance trade-off
Accuracy
Number of training samples
Training accuracy
Validation accuracy
Desired accuracy


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

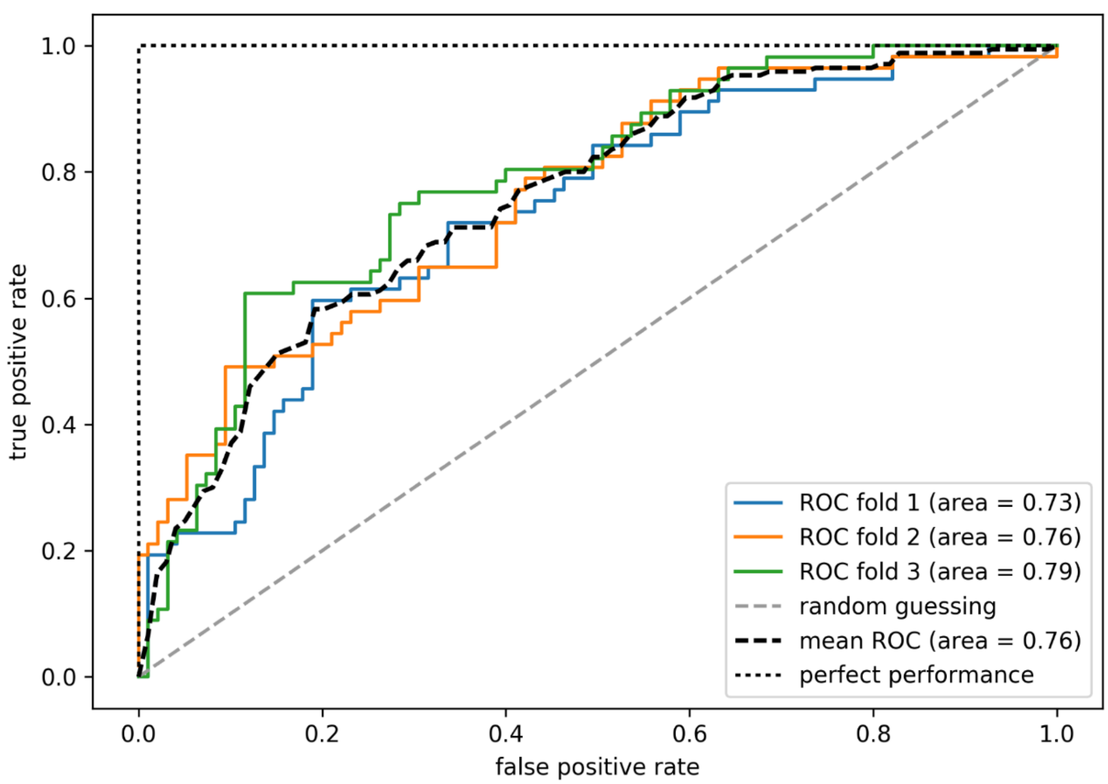

1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
Accuracy
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Parameter C
training accuracy
validation accuracy


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)

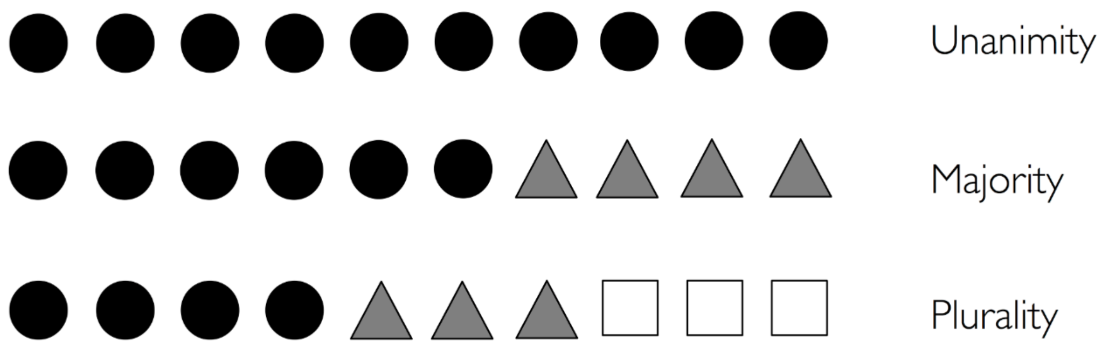


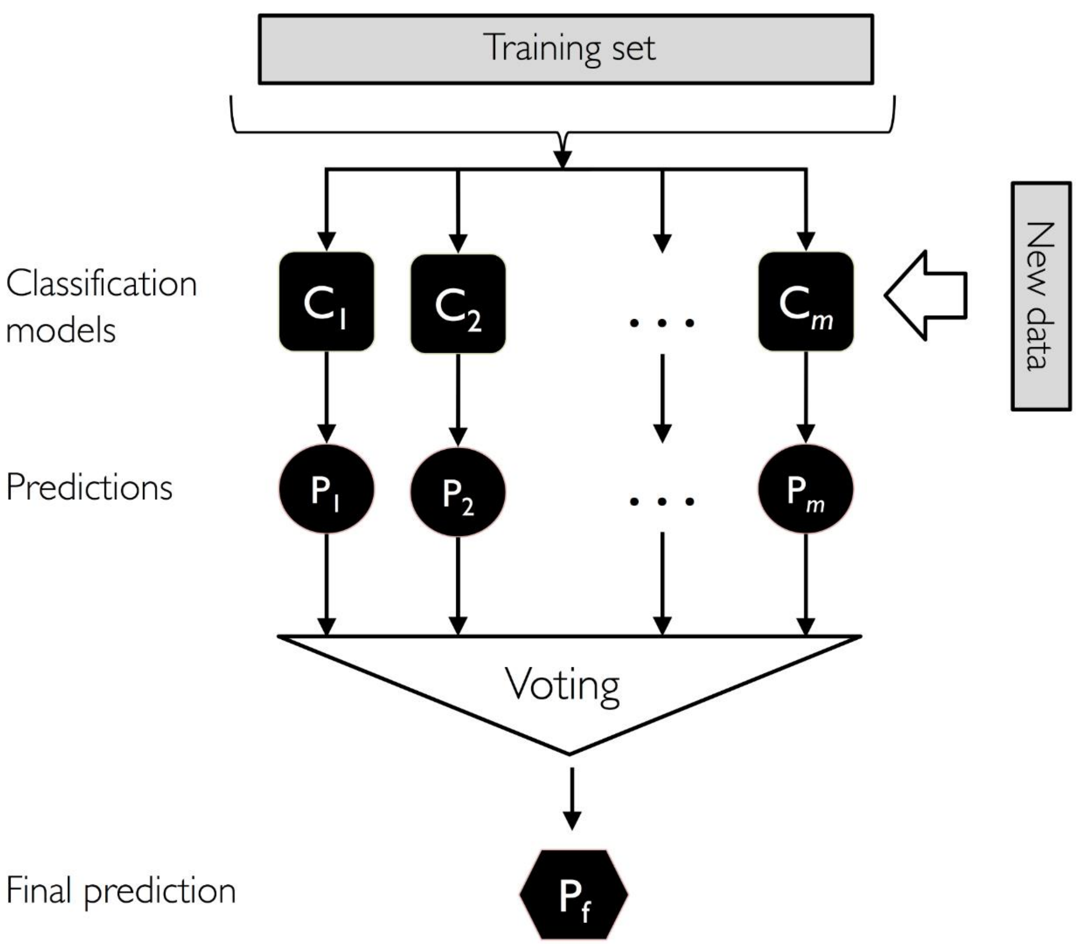
0
1
0
1
71
1
2
40
True label
Predicted label

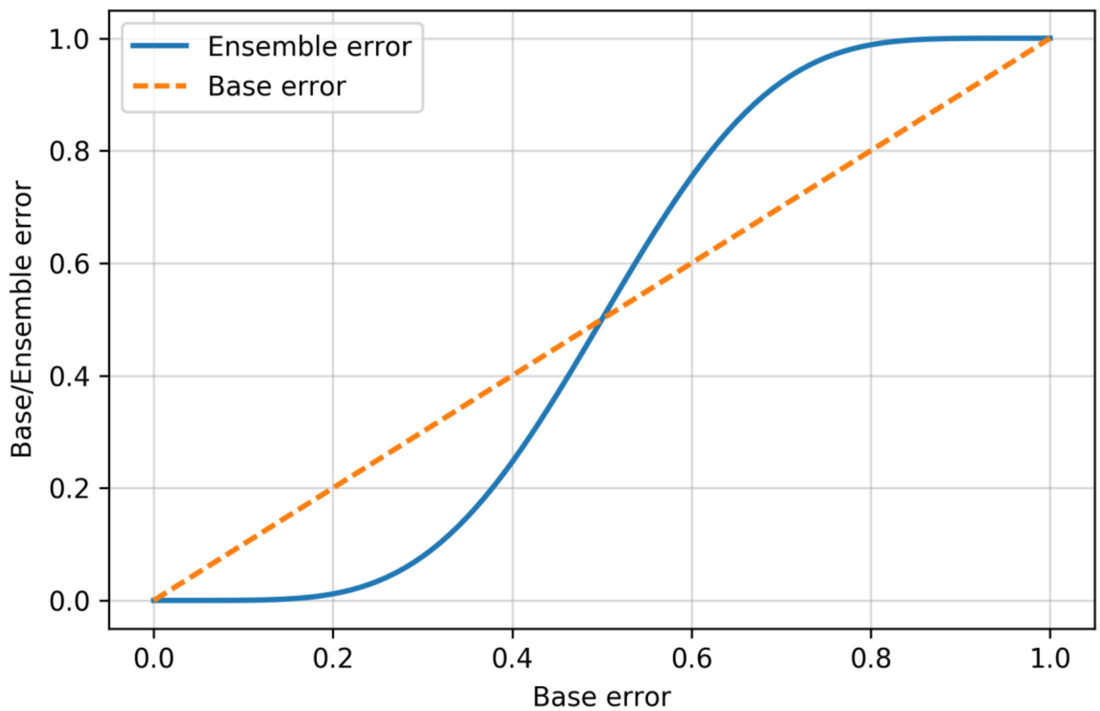


ROC fold 1 (area = 0.73)
ROC fold 2 (area = 0.76)
ROC fold 3 (area = 0.79)
random guessing
mean ROC (area = 0.76)
perfect performance
true positive rate
false positive rate
0.0
0.2
0.4
0.6
0.8
1.0

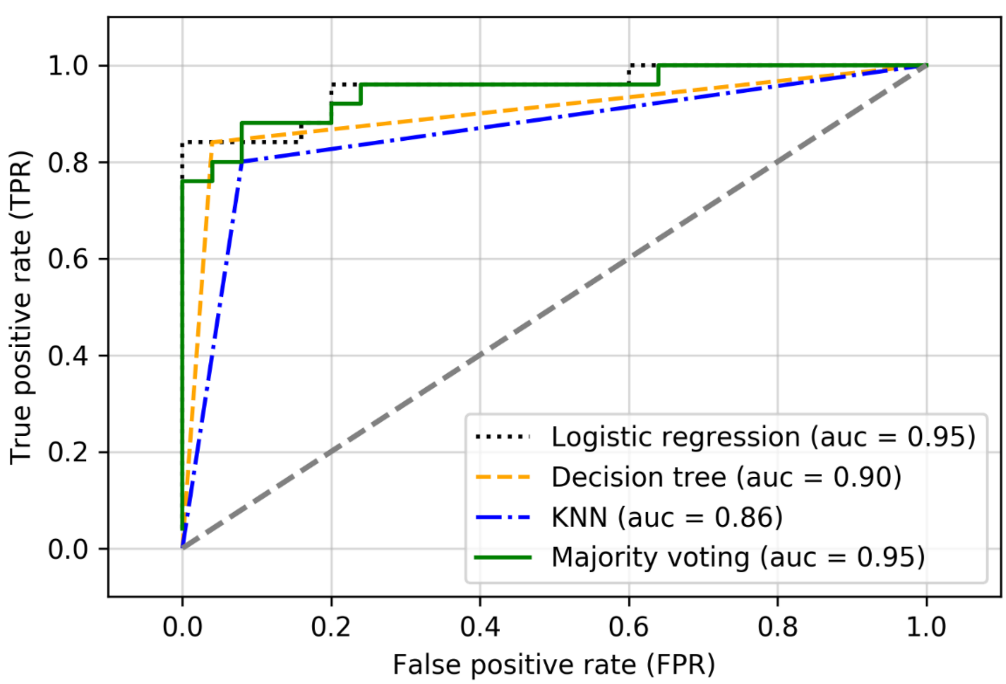
Chapter 07: Learning Best Practices for Model Evaluation and Hyperparameter Tuning



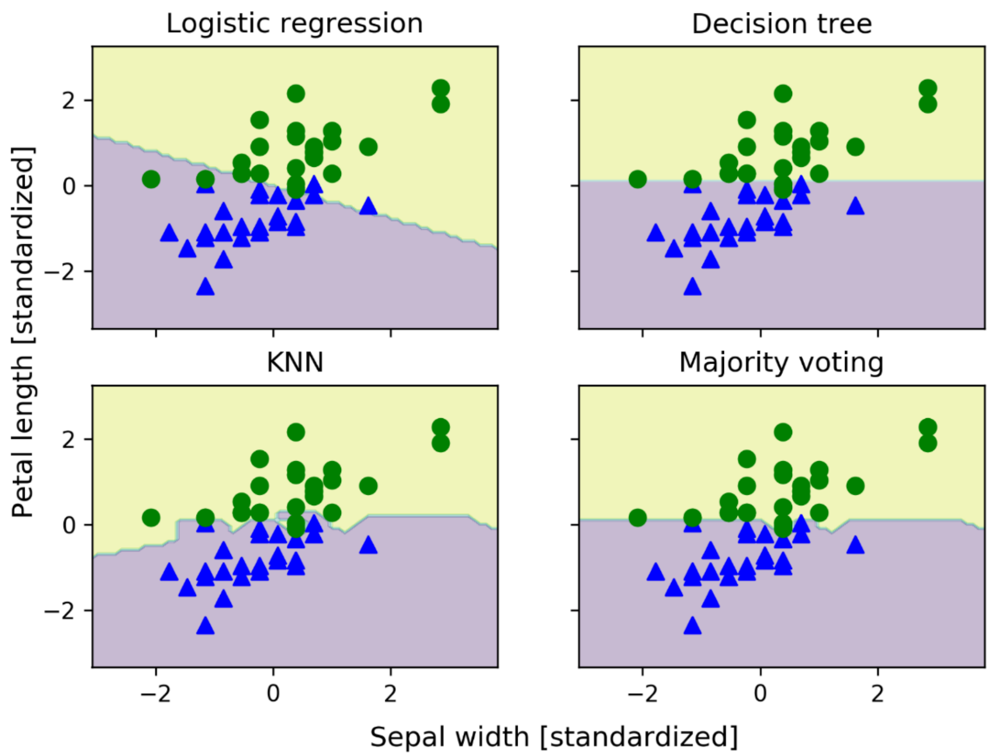


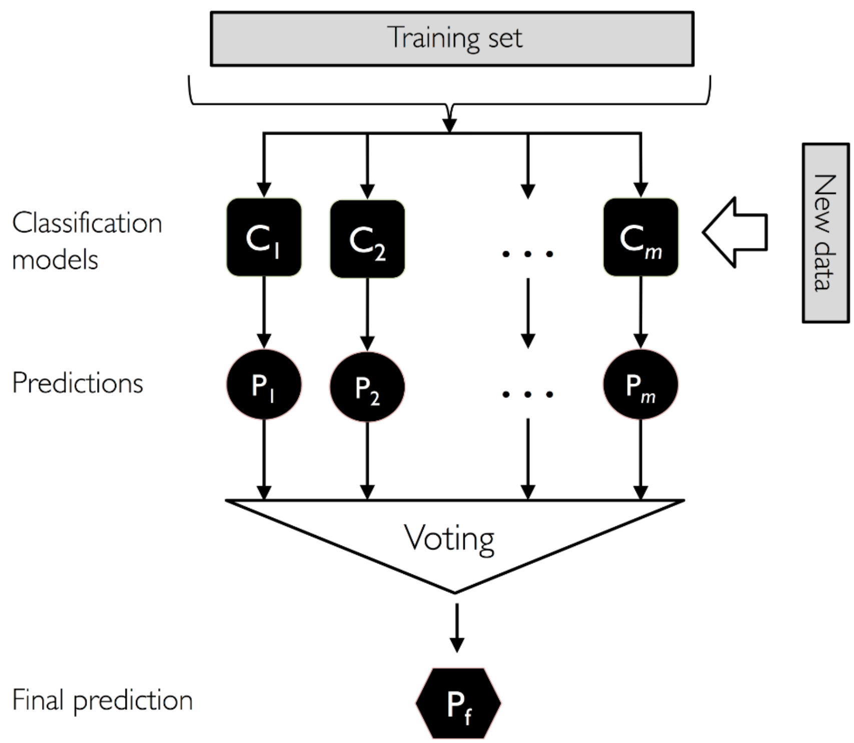


Ensemble error
Base error
Base/Ensemble error
Base error
0.0
0.2
0.4
0.6
0.8
1.0

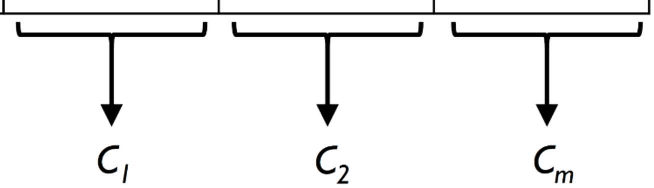


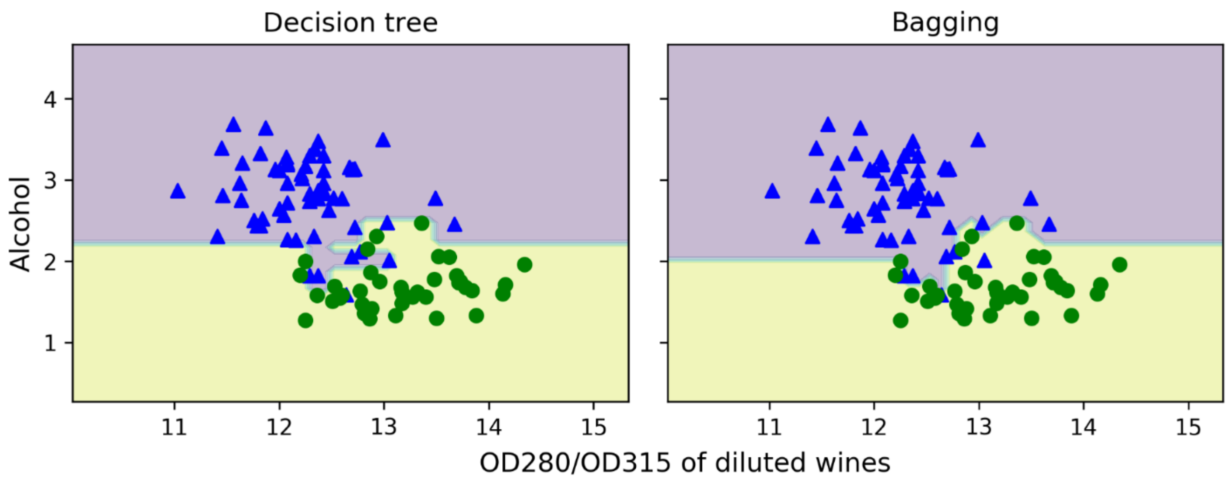
Logistic regression (auc = 0.95)
Decision tree (auc = 0.90)
KNN (auc = 0.86)
Majority voting (auc = 0.95)
True positive rate (TPR)
False positive rate (FPR)
0.0
0.2
0.4
0.6
0.8
1.0

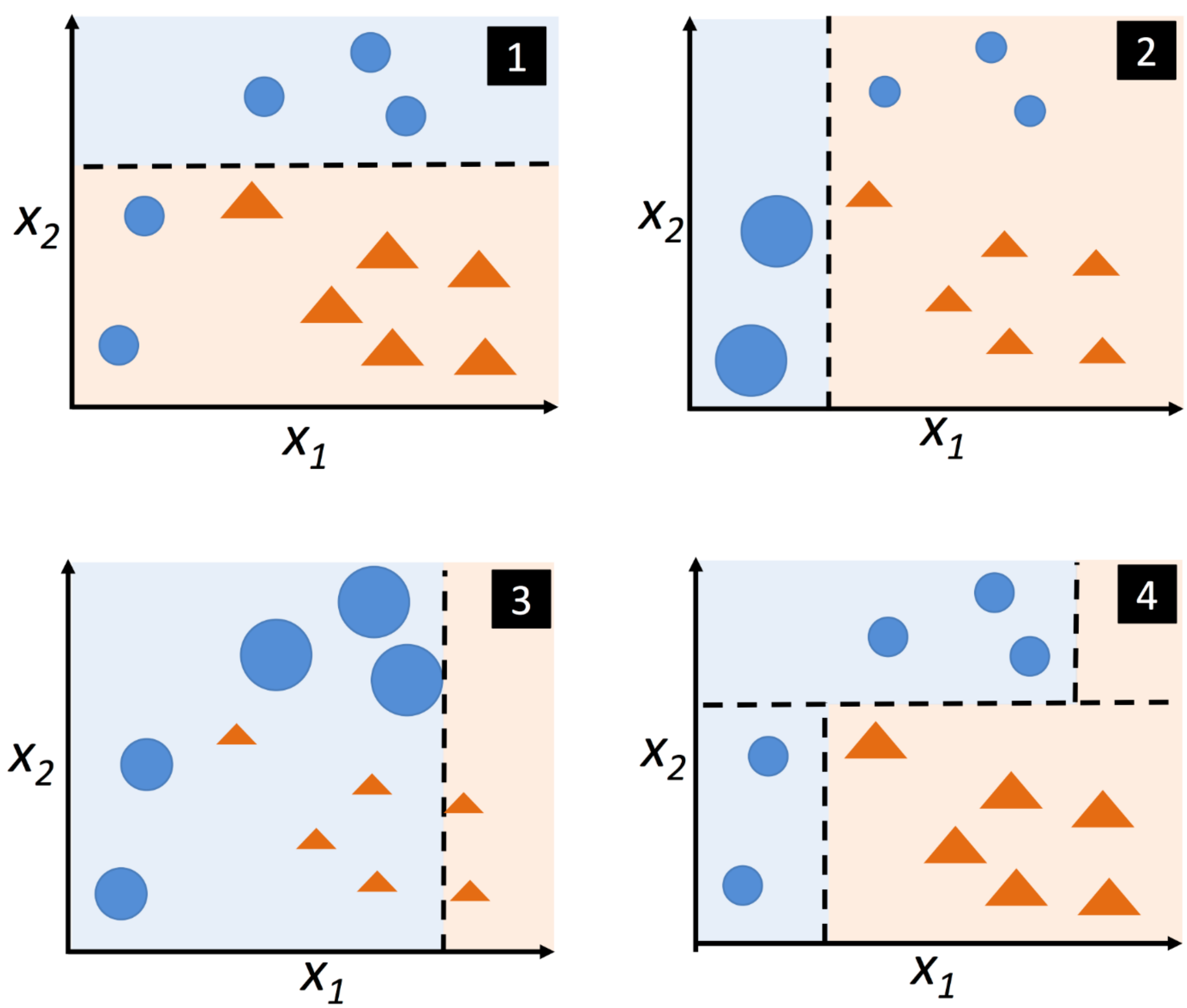

Logistic regression
Decision tree
KNN
Majority voting
Petal length [standardized]
Sepal width [standardized]
2
0
−2
−2
0
2



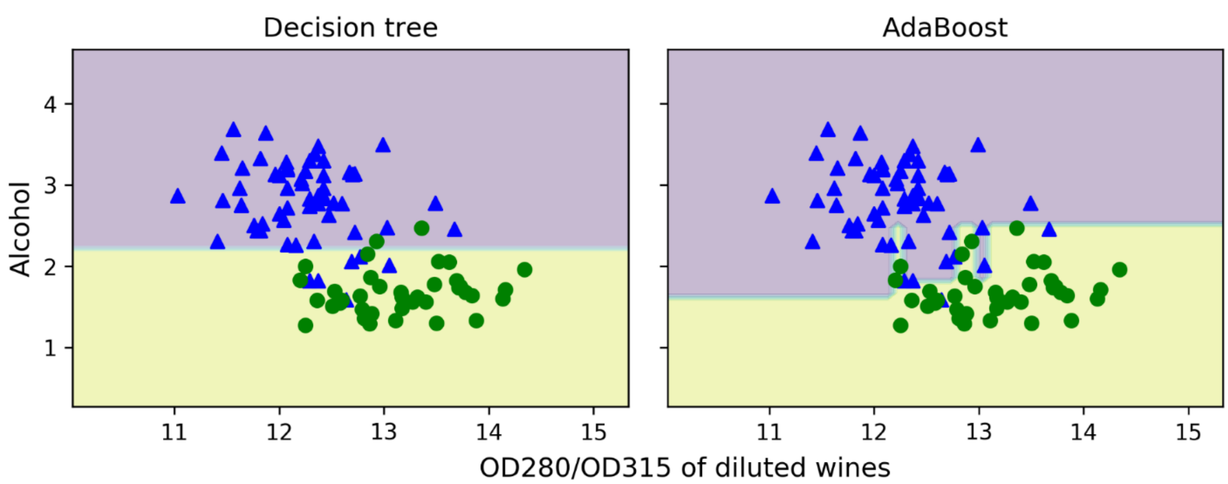
Sample indices	Bagging round 1	Bagging round 2	…
1	2	7	…
2	2	3	…
3	1	2	…
4	3	1	…
5	7	1	…
6	2	7	…
7	4	7	…




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


1
2
3
4
x_1
x_2

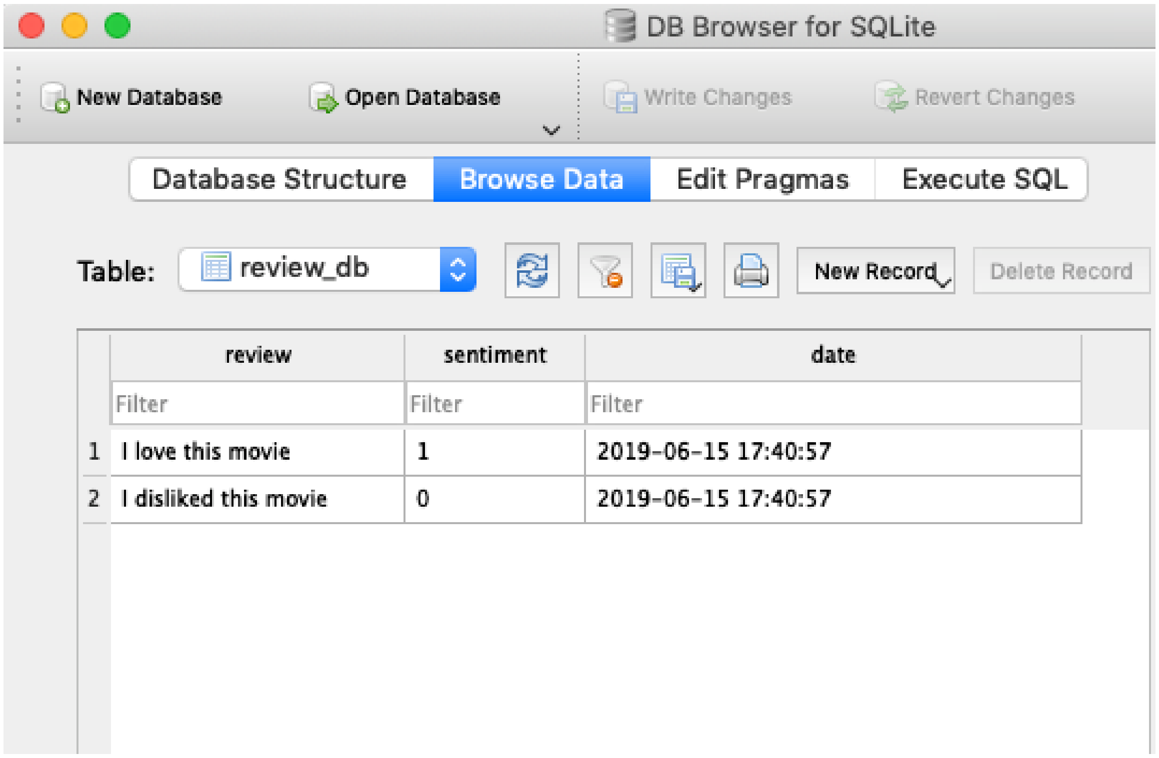
Index	x	y	Weights	$\hat{y}$(x <= 3.0)?	Correct?	Updated weights
1	1.0	1	0.1	1	Yes	0.072
2	2.0	1	0.1	1	Yes	0.072
3	3.0	1	0.1	1	Yes	0.072
4	4.0	-1	0.1	-1	Yes	0.072
5	5.0	-1	0.1	-1	Yes	0.072
6	6.0	-1	0.1	-1	Yes	0.072
7	7.0	1	0.1	-1	No	0.167
8	8.0	1	0.1	-1	No	0.167
9	9.0	1	0.1	-1	No	0.167
10	10.0	-1	0.1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

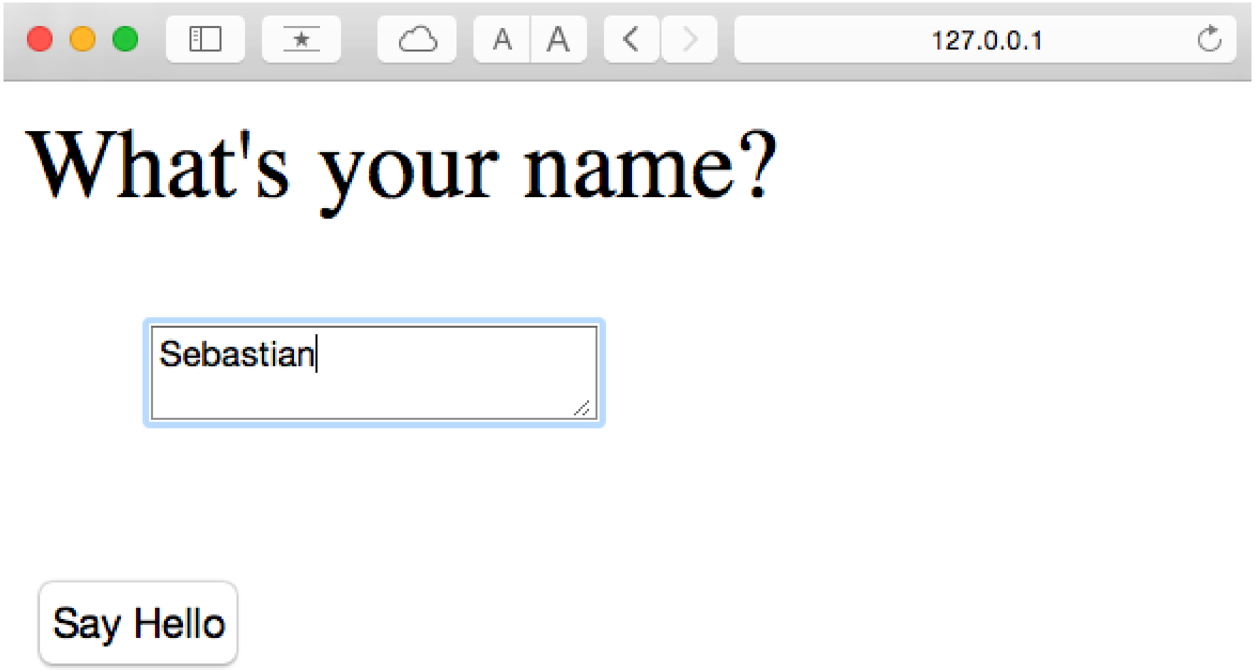


Chapter 08: Applying Machine Learning to Sentiment Analysis

	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 09: Embedding a Machine Learning Model into a Web Application





original `first_app.html` file

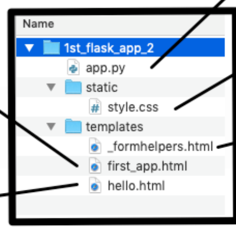
```
<!doctype html>
<html>
  <head>
    <title>First app</title>

  </head>
  <body>

  <div>
        Hi, this is my first Flask web app!
  </div>

  </body>
</html>
```

modified `first_app.html` file

```
<!doctype html>
<html>
  <head>
    <title>First app</title>
        <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
  </head>
  <body>

{% from "_formhelpers.html" import render_field %}

<div>What's your name?</div>
<form method=post action="/hello">

  <dl>
        {{ render_field(form.sayhello) }}
  </dl>

  <input type=submit value='Say Hello' name='submit_btn'>

</form>

  </body>
</html>
```

```html
<!doctype html>
<html>
  <head>
    <title>First app</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
  </head>
  <body>

{% from "_formhelpers.html" import render_field %}

<div>What's your name?</div>
<form method=post action="/hello">

  <dl>
      {{ render_field(form.sayhello) }}
  </dl>

  <input type=submit value='Say Hello' name='submit_btn'>

</form>

  </body>
</html>
```

```html
<!doctype html>
<html>
  <head>
    <title>First app</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
  </head>
  <body>

<div>Hello {{ name }}</div>


  </body>
</html>
```

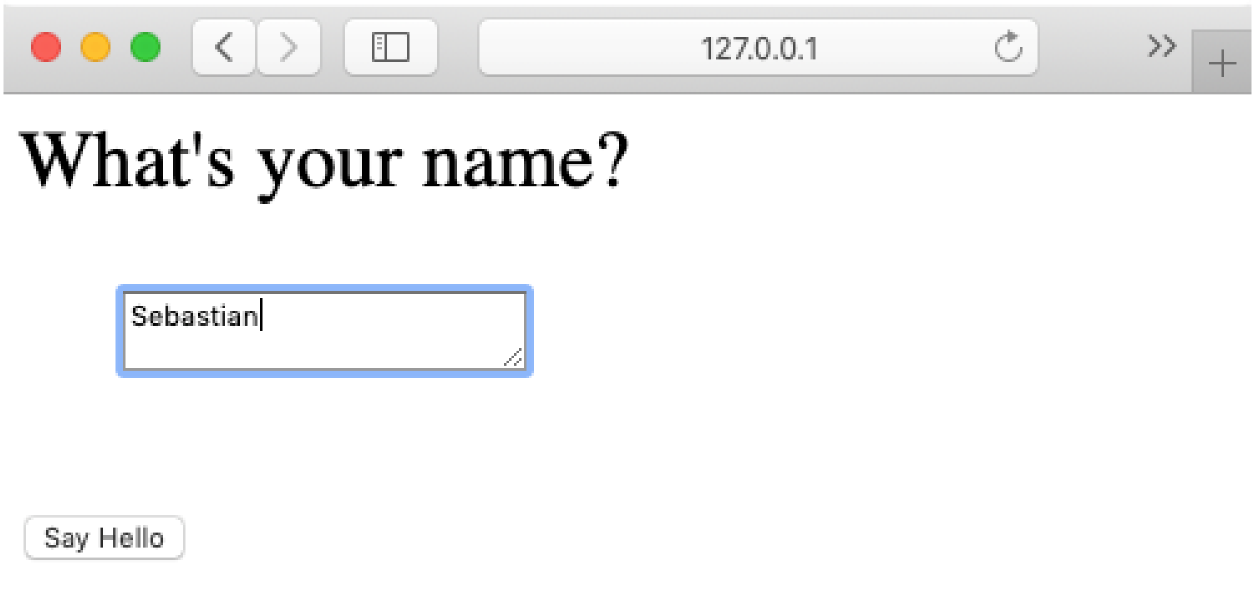



```python
from flask import Flask, render_template, request
from wtforms import Form, TextAreaField, validators

app = Flask(__name__)

class HelloForm(Form):
    sayhello = TextAreaField('',[validators.DataRequired()])

@app.route('/')
def index():
    form = HelloForm(request.form)
    return render_template('first_app.html', form=form)

@app.route('/hello', methods=['POST'])
def hello():
    form = HelloForm(request.form)
    if request.method == 'POST' and form.validate():
        name = request.form['sayhello']
        return render_template('hello.html', name=name)
    return render_template('first_app.html', form=form)

if __name__ == '__main__':
    app.run(debug=True)
```

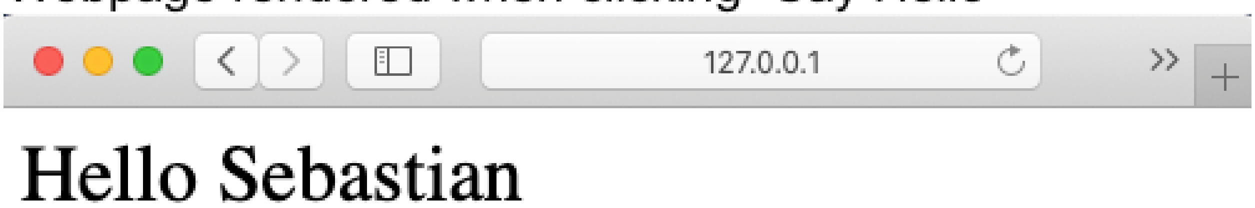
```css
body {
    font-size: 2em;
}
```

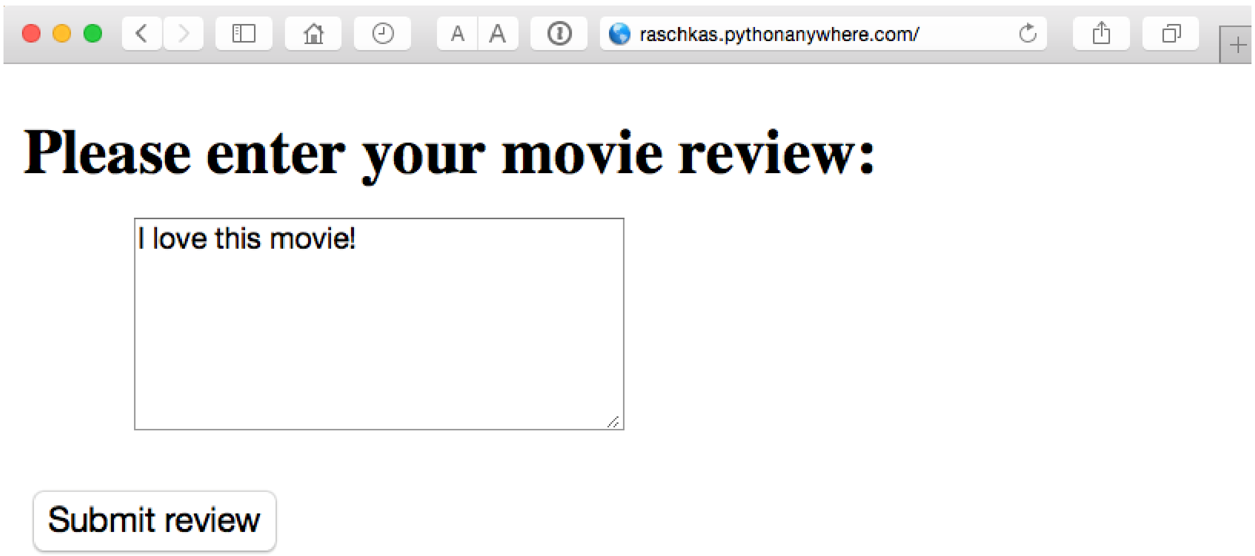
```html
{% macro render_field(field) %}
  <dt>{{ field.label }}
  <dd>{{ field(**kwargs)|safe }}
  {% if field.errors %}
    <ul class=errors>
    {% for error in field.errors %}
      <li>{{ error }}</li>
    {% endfor %}
    </ul>
  {% endif %}
  </dd>
  </dt>
{% endmacro %}
```

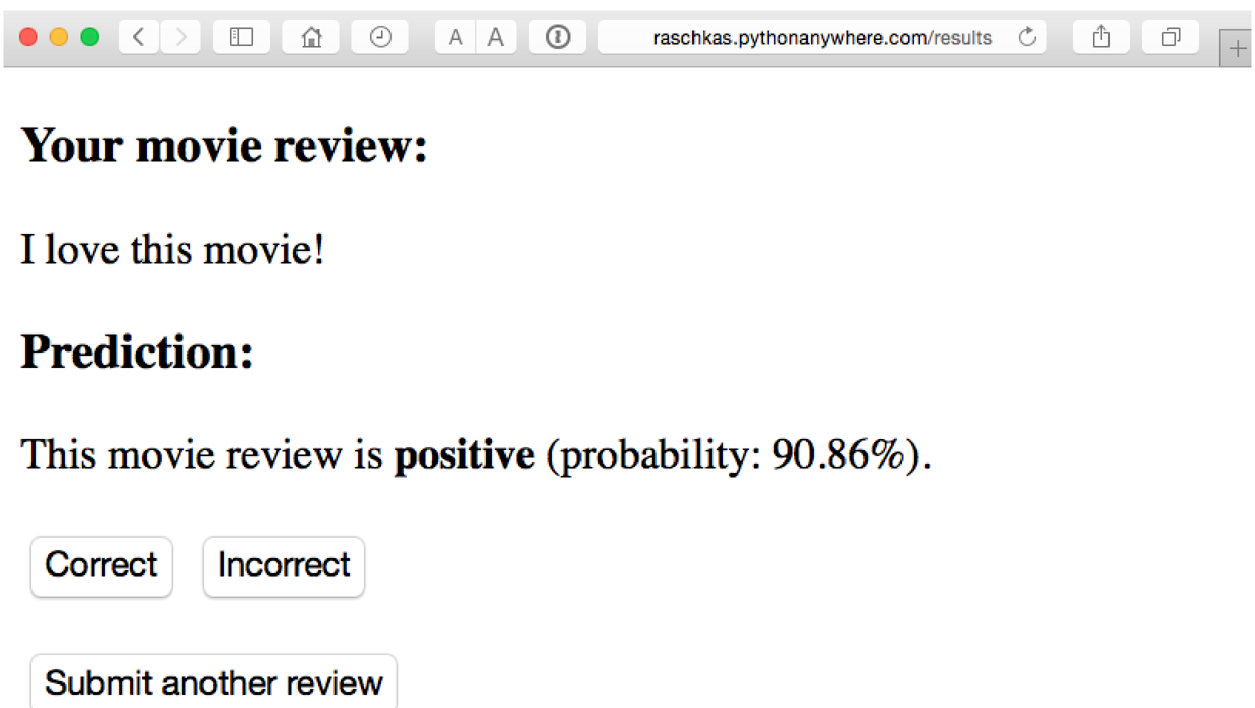
Webpage rendered when opening
`http://127.0.0.1:5000/`

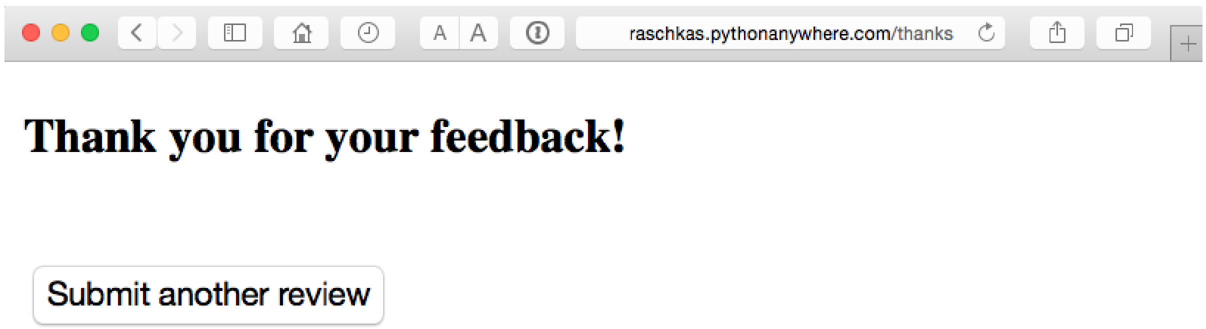


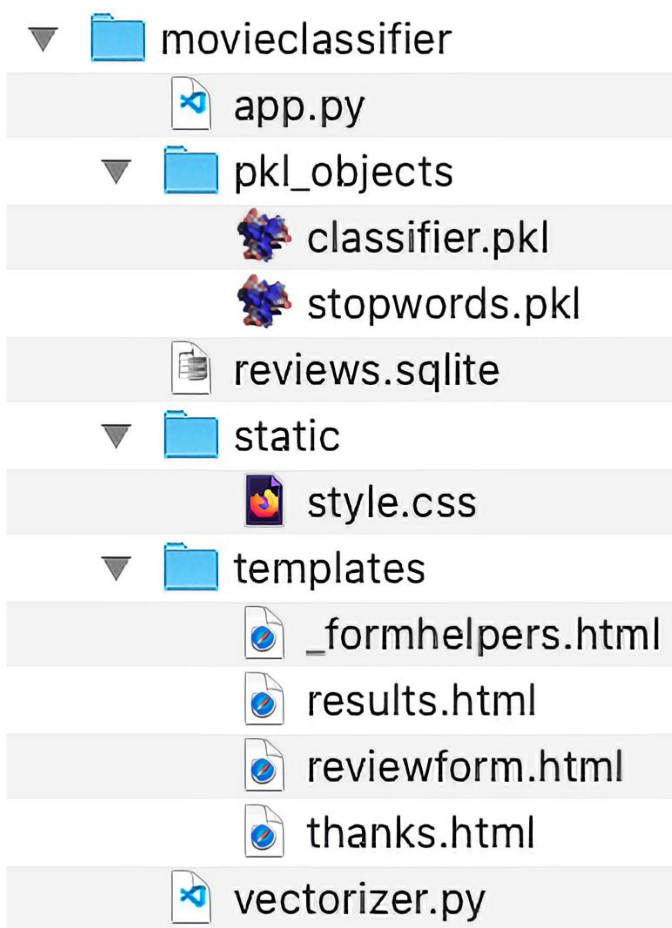
Webpage rendered when clicking "Say Hello"

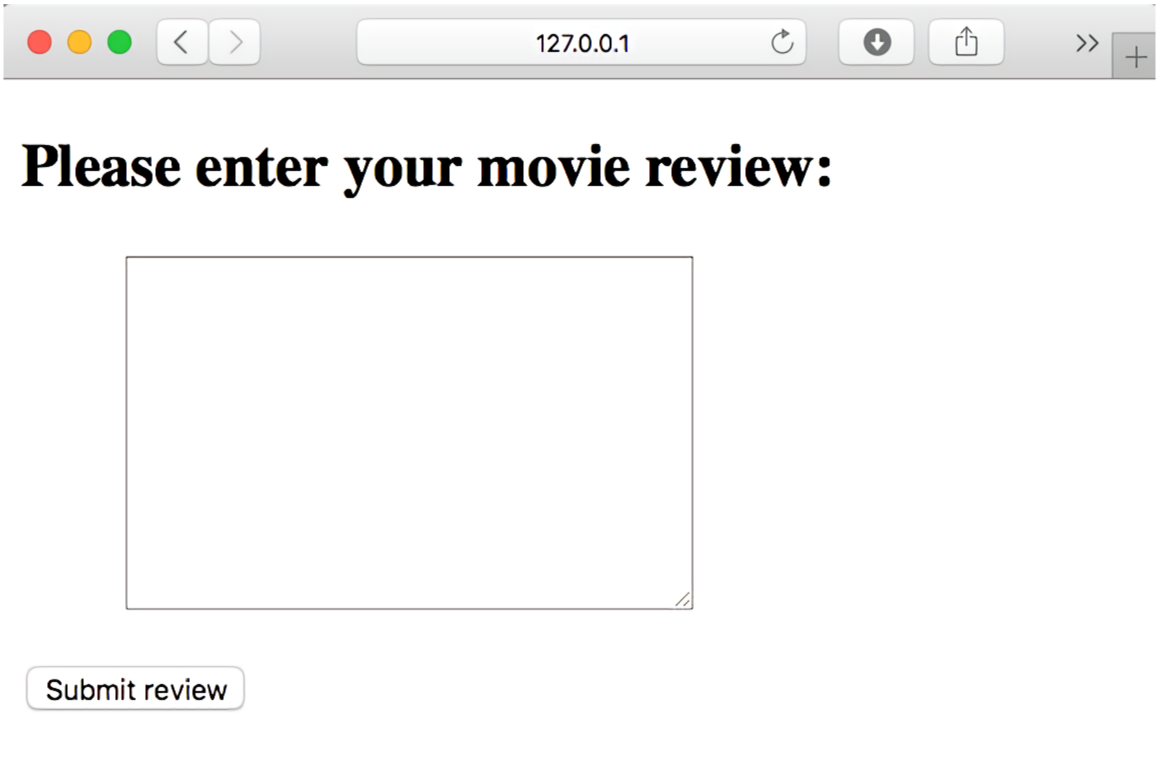



raschkas.pythonanywhere.com/
Please enter your movie review:
I love this movie!
Submit review

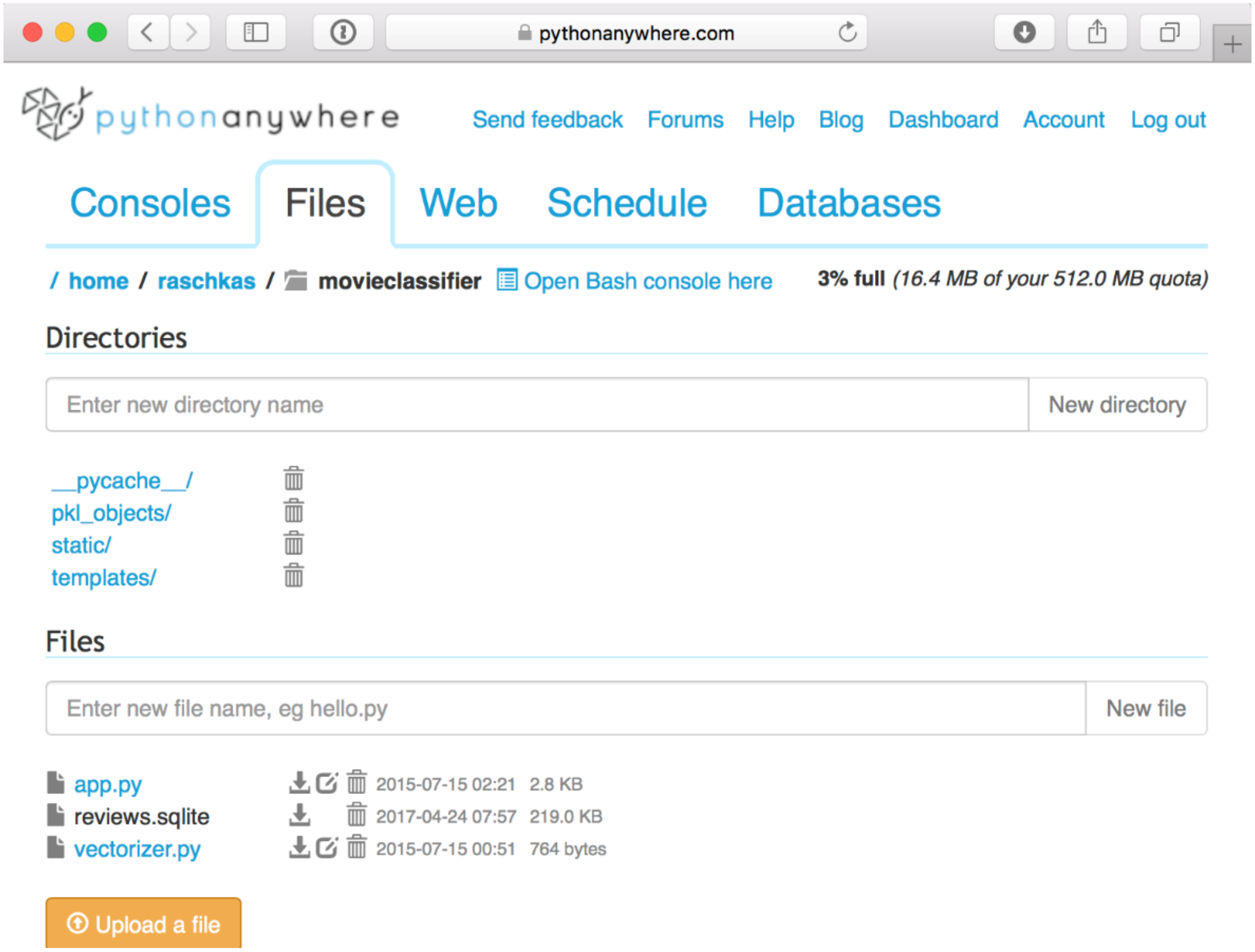

raschkas.pythonanywhere.com/results
Your movie review:
I love this movie!
Prediction:
This movie review is positive (probability: 90.86%).
Correct
Incorrect
Submit another review


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

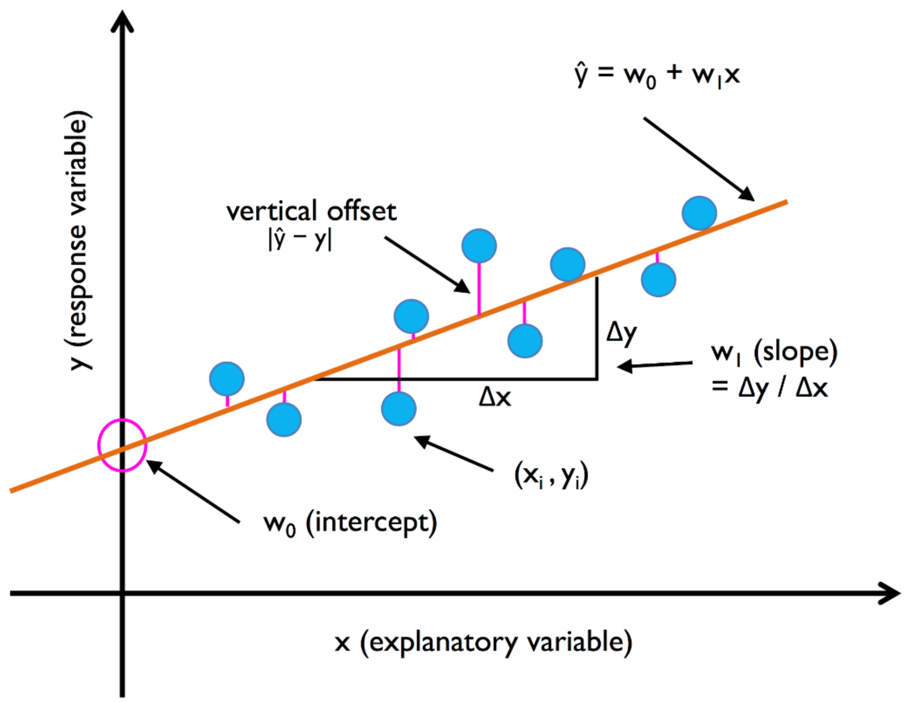

movieclassifier
app.py
pkl_objects
classifier.pkl
stopwords.pkl
reviews.sqlite
static
style.css
templates
_formhelpers.html
results.html
reviewform.html
thanks.html
vectorizer.py

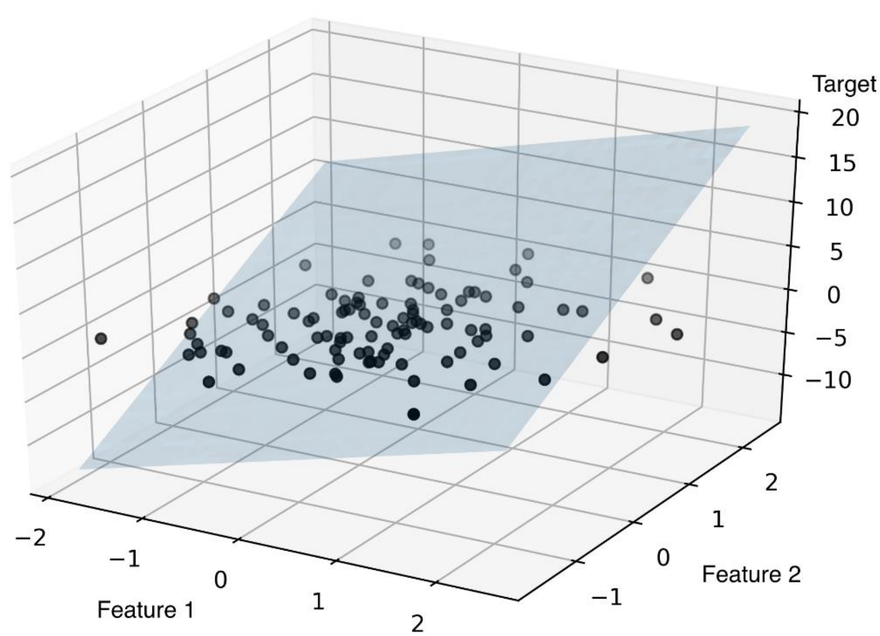



```
        y = int(not(y))
    train(review, y)
    sqlite_entry(db, review, y)
    return render_template('thanks.html')

if __name__ == '__main__':
    app.run(debug=True)
```

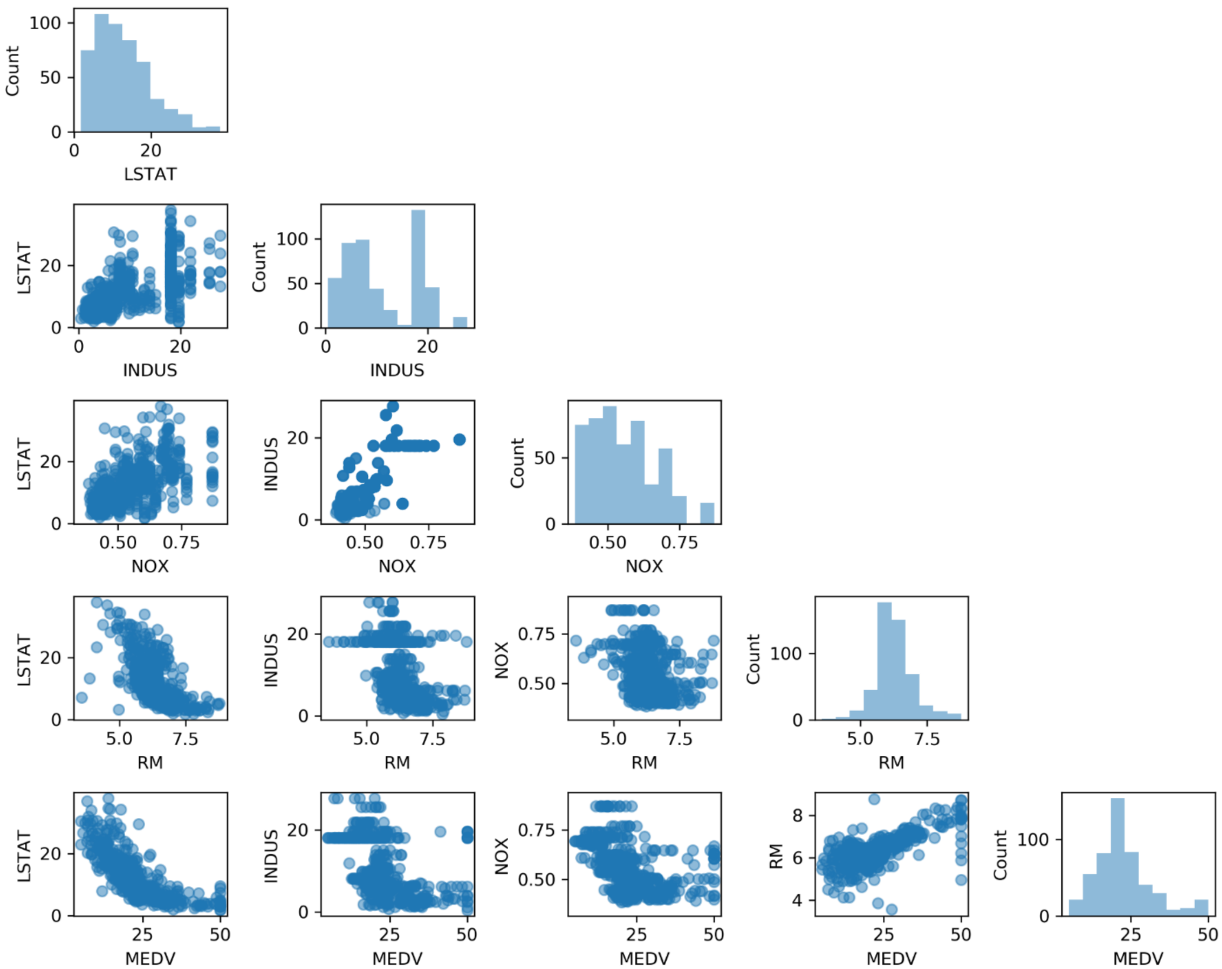

pythonanywhere.com
pythonanywhere
Send feedback
Forums
Help
Blog
Dashboard
Account
Log out
Consoles
Files
Web
Schedule
Databases
/ home / raschkas / movieclassifier
Open Bash console here
3% full (16.4 MB of your 512.0 MB quota)
Directories
Enter new directory name
New directory
__pycache__/
pkl_objects/
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

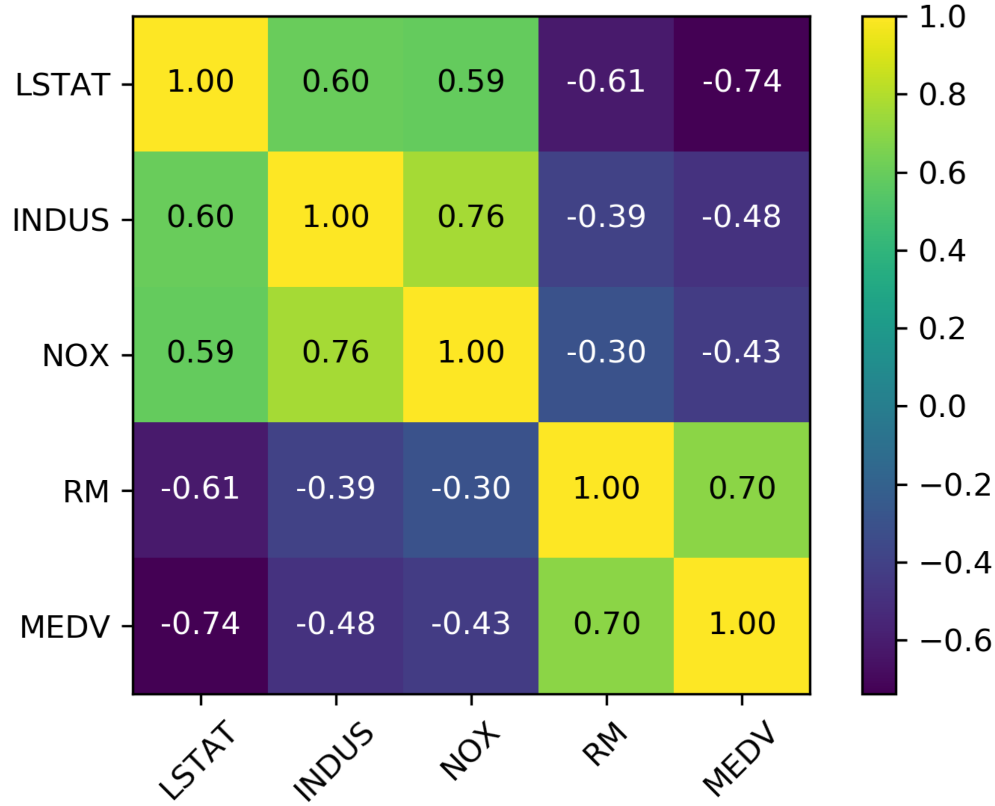
Chapter 10: Predicting Continuous Target Variables with Regression Analysis

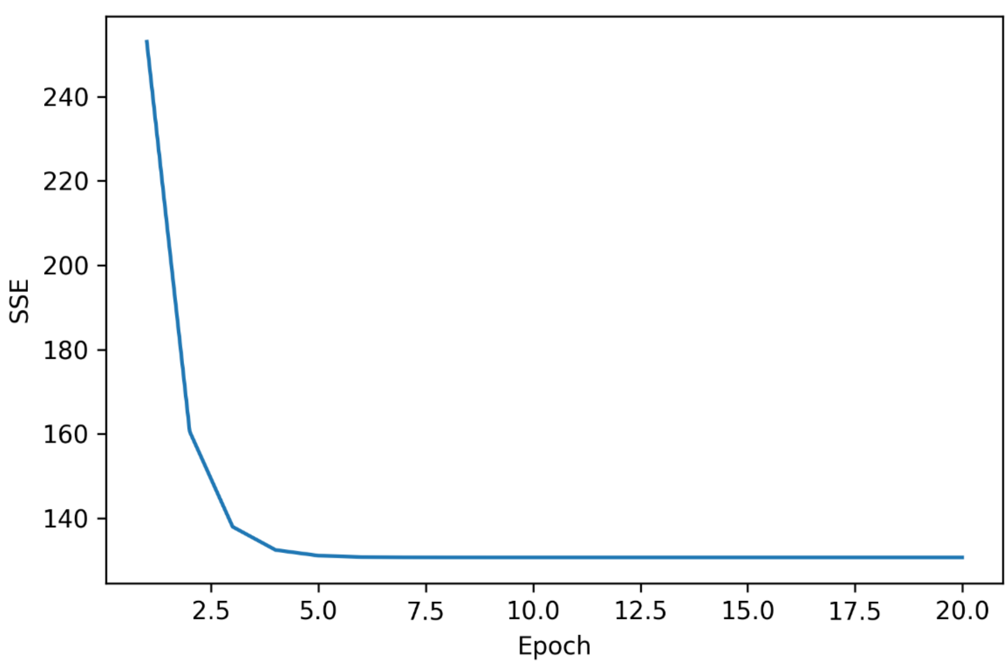


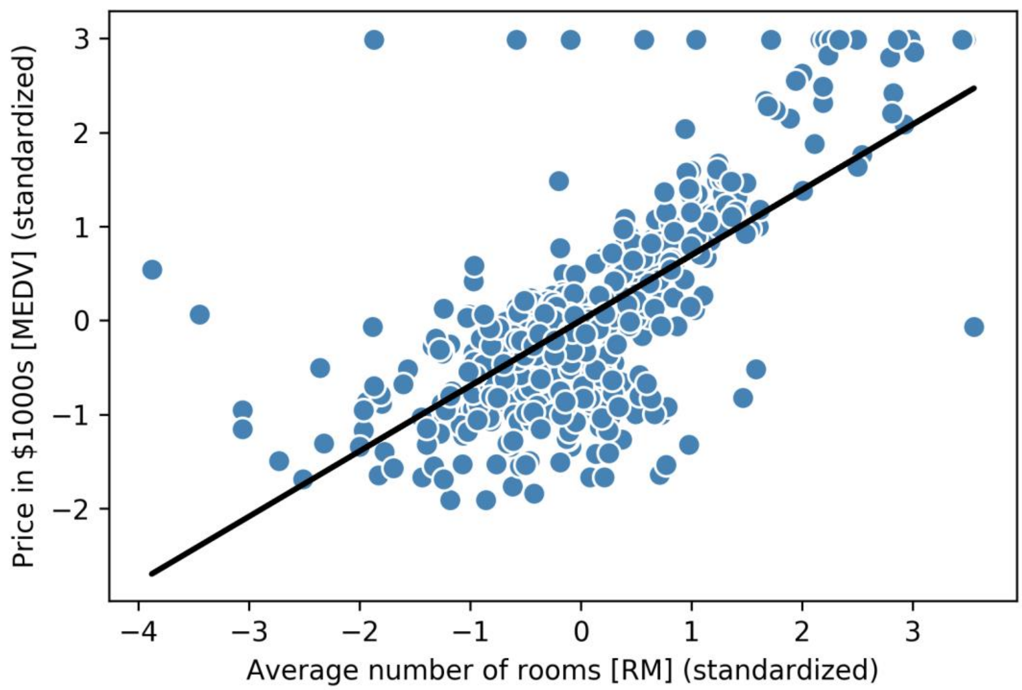


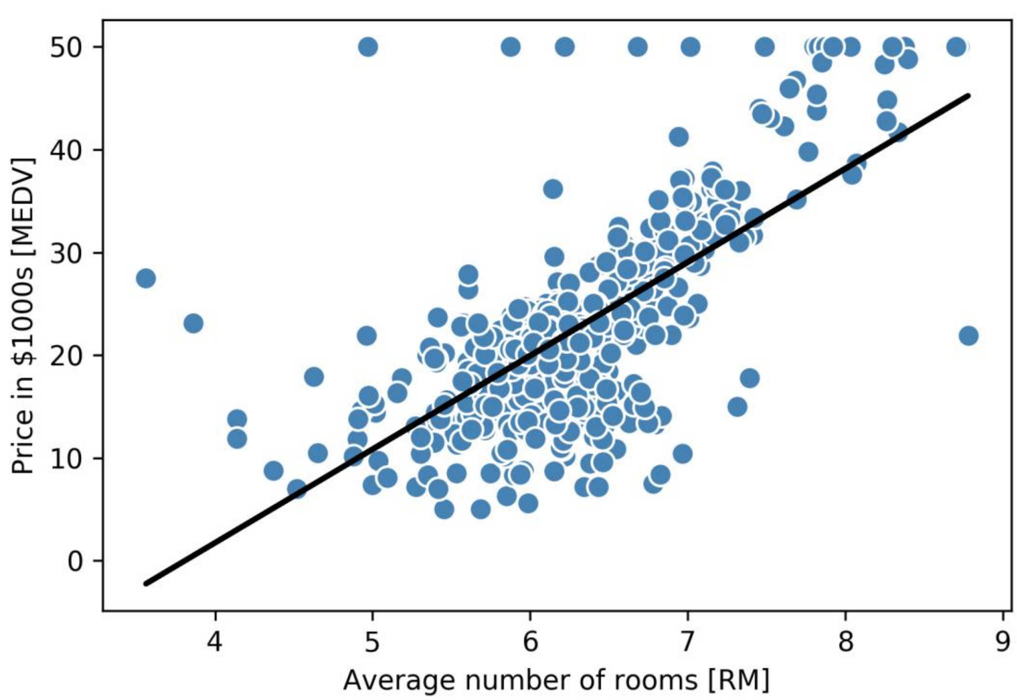
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	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

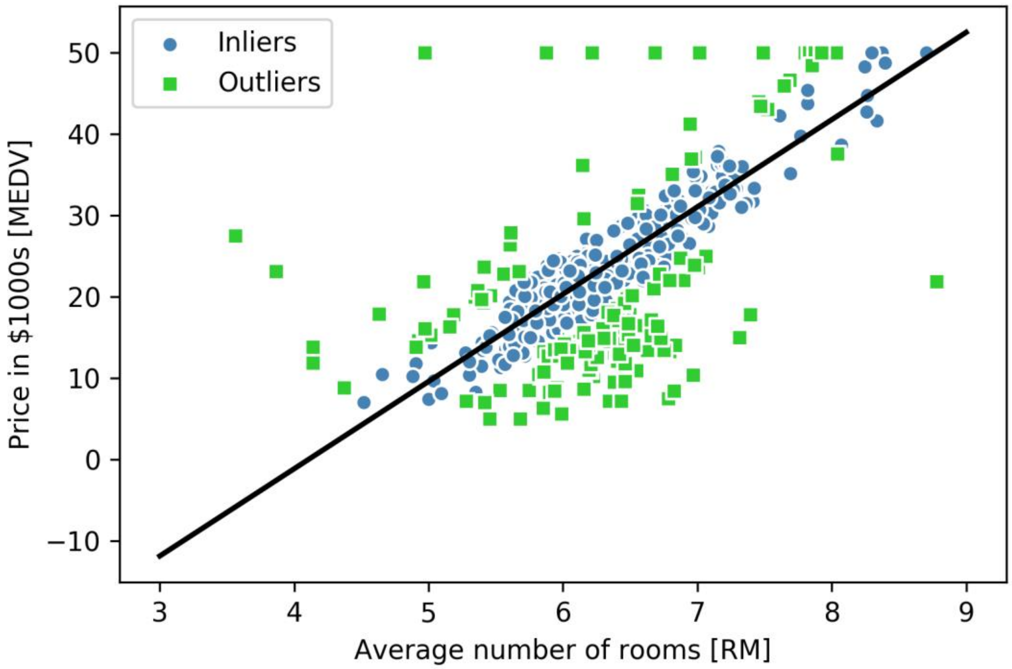


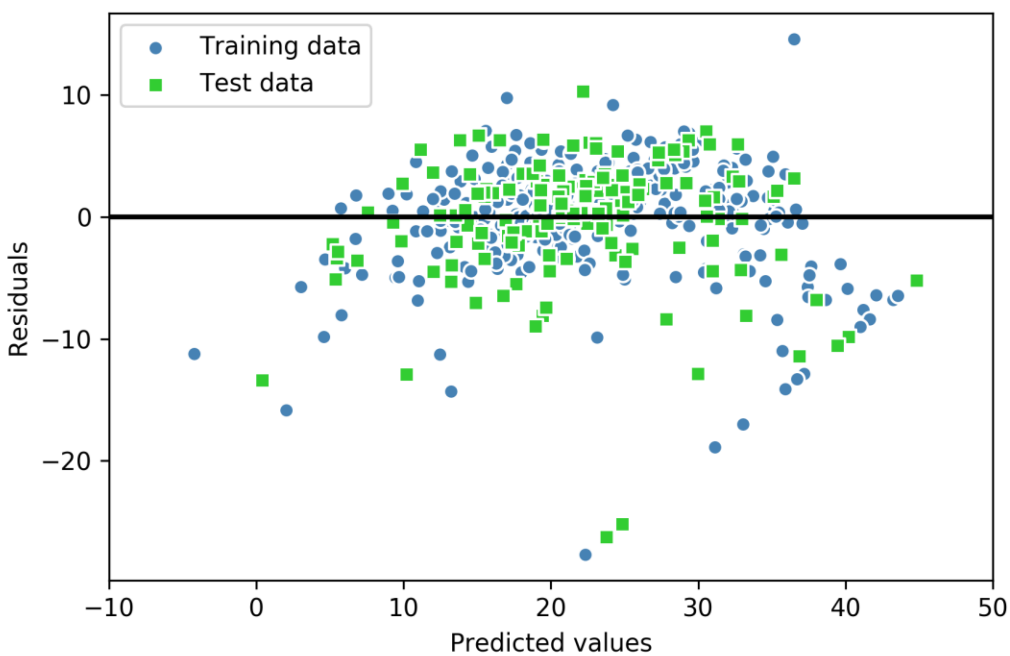

LSTAT
INDUS
NOX
RM
MEDV
1.00
0.60
0.59
-0.61
-0.74
0.60
1.00
0.76
-0.39
-0.48
0.59
0.76
1.00
-0.30
-0.43
-0.61
-0.39
-0.30
1.00
0.70
-0.74
-0.48
-0.43
0.70
1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

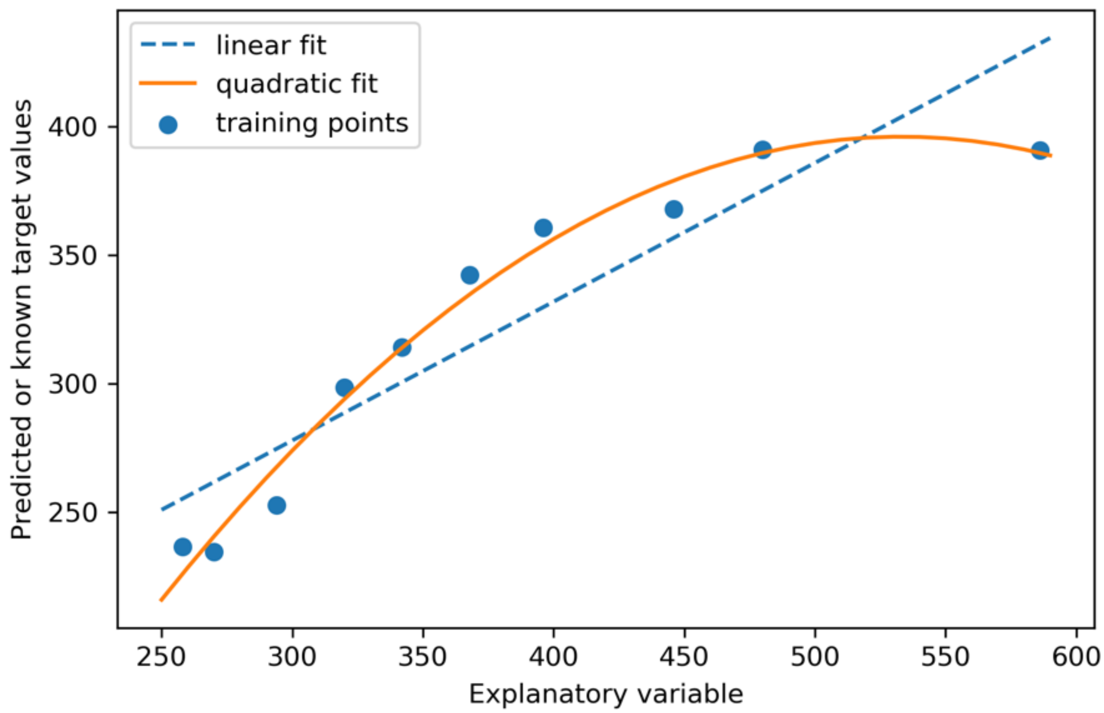

SSE
240
220
200
180
160
140
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epoch

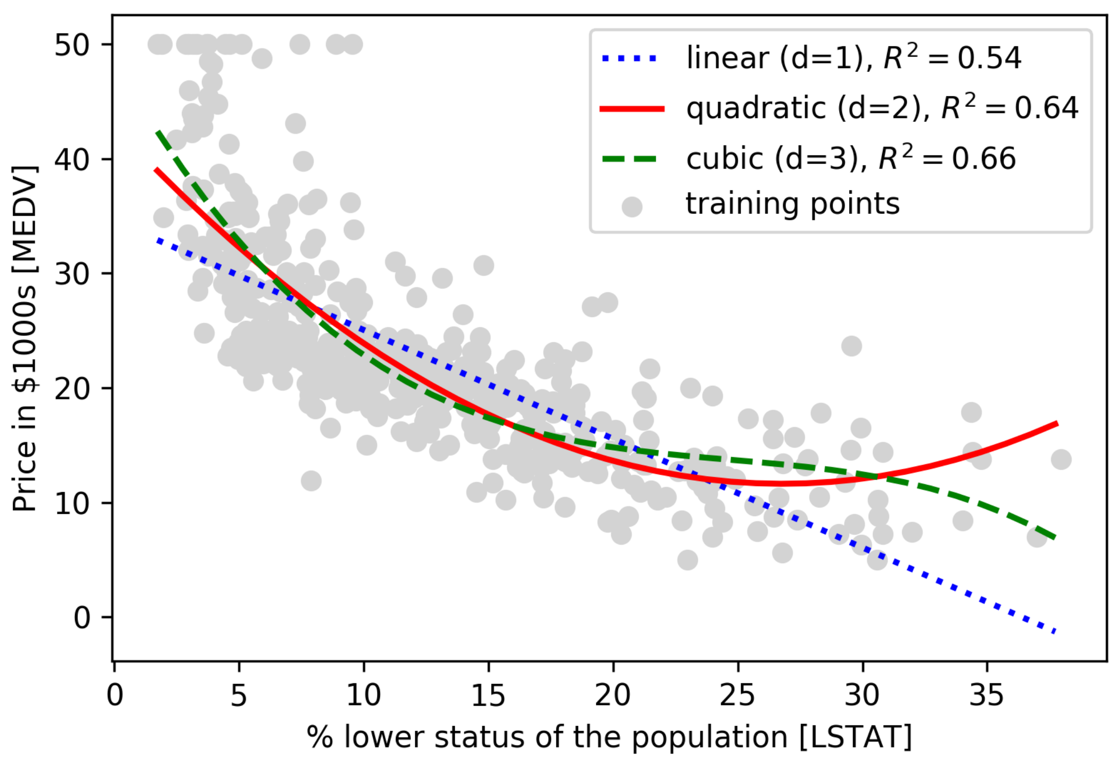

Price in $1000s [MEDV] (standardized)
Average number of rooms [RM] (standardized)

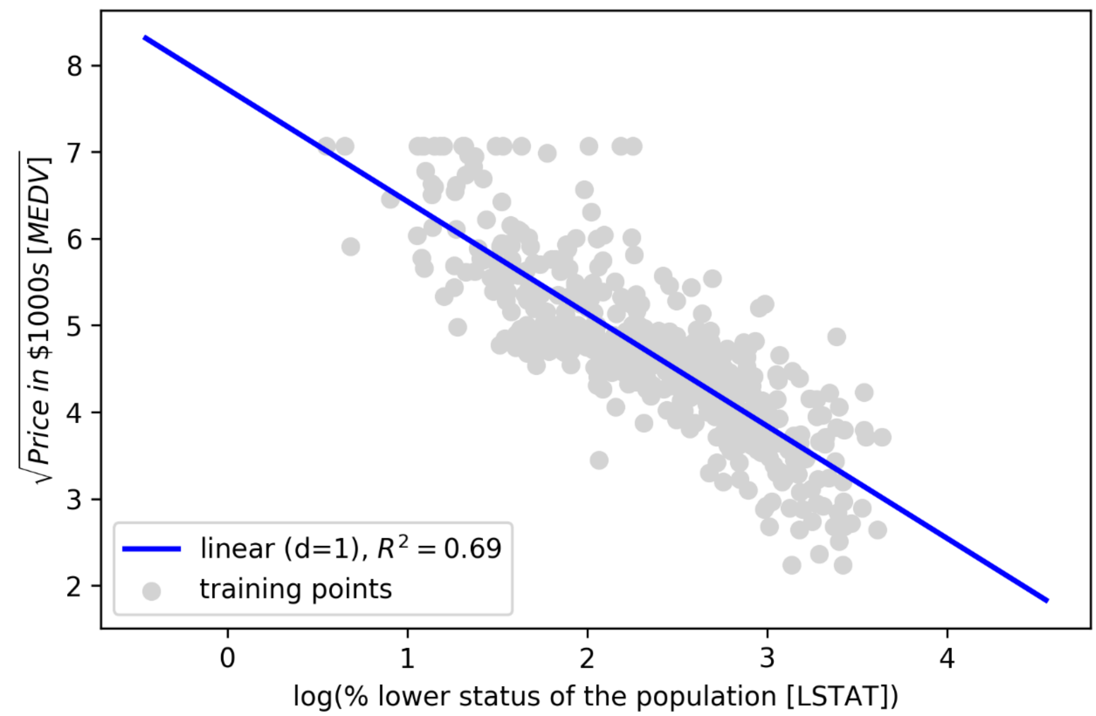

Price in $1000s [MEDV]
Average number of rooms [RM]


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

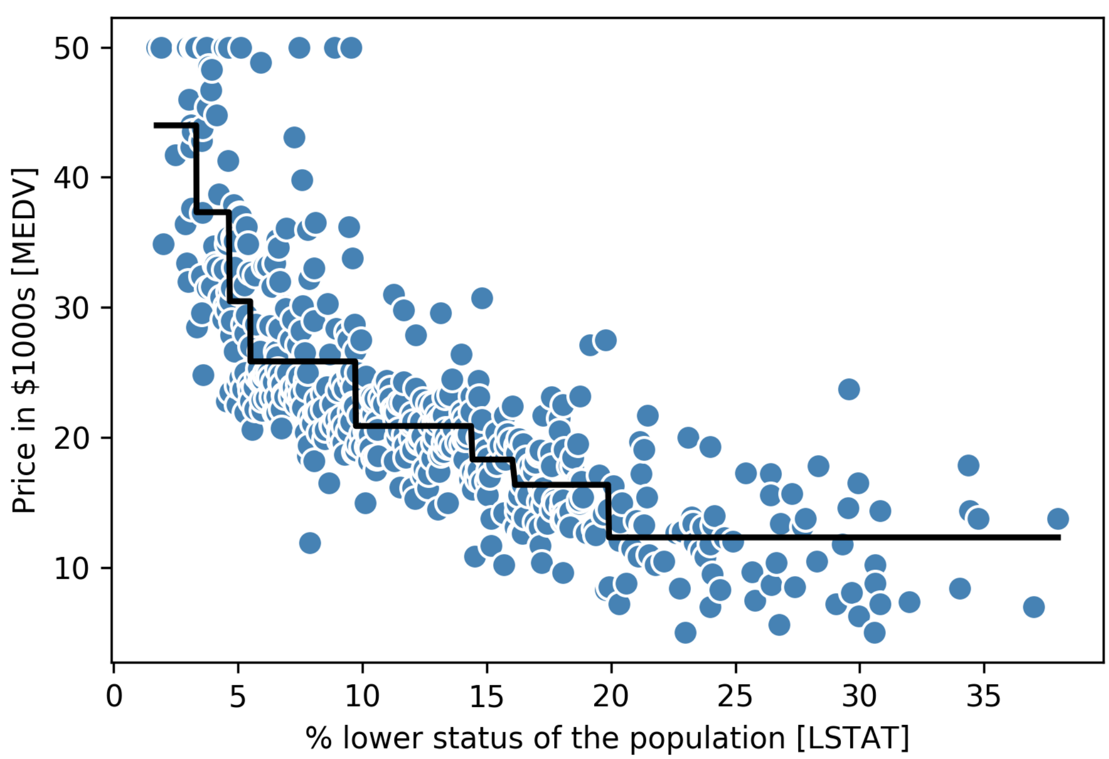

Training data
Test data
Residuals
Predicted values
10
0
−10
−20
−10
0
10
20
30
40
50


linear fit
quadratic fit
training points
Predicted or known target values
Explanatory variable
250
300
350
400
450
500
550
600

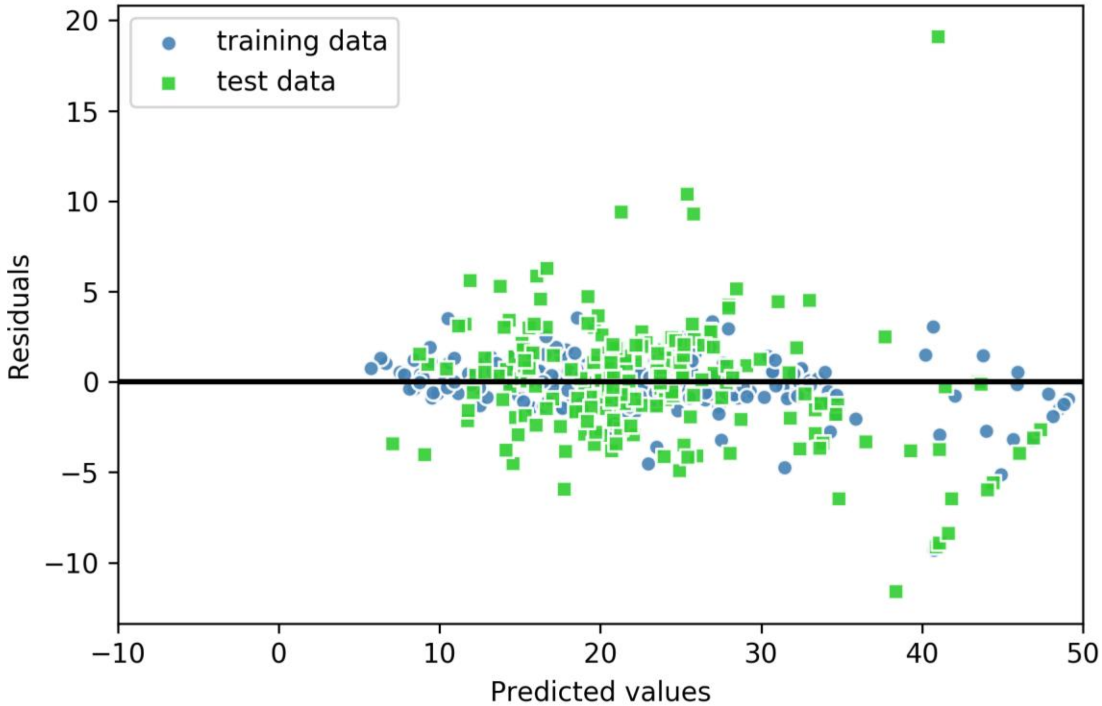

linear (d=1), $R^2 = 0.54$
quadratic (d=2), $R^2 = 0.64$
cubic (d=3), $R^2 = 0.66$
training points
Price in \$1000s [MEDV]
% lower status of the population [LSTAT]
0
5
10
15
20
25
30
35
40
50



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

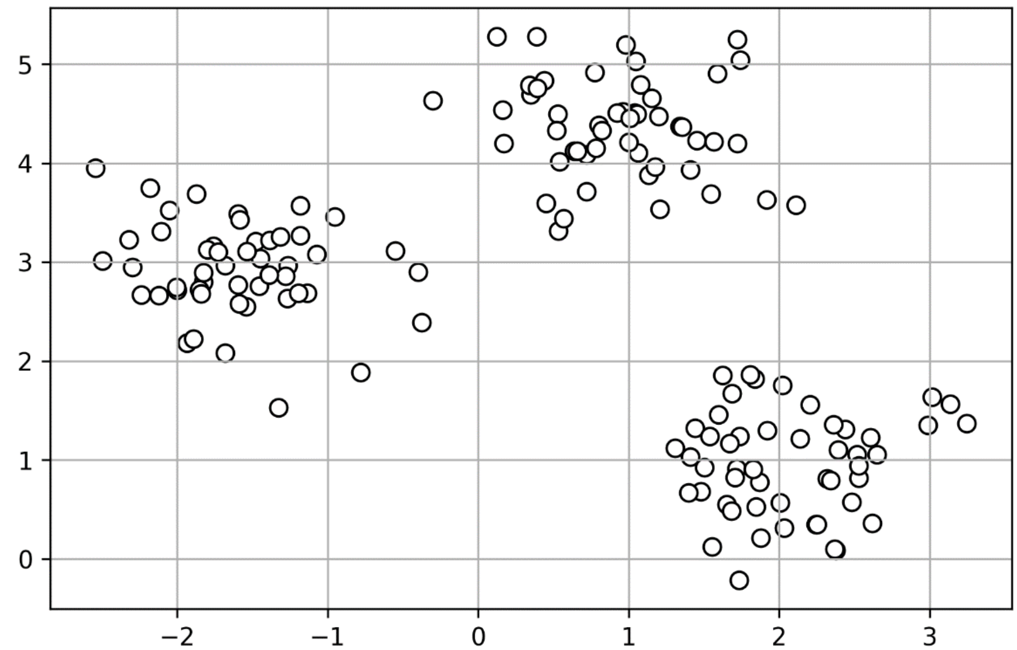


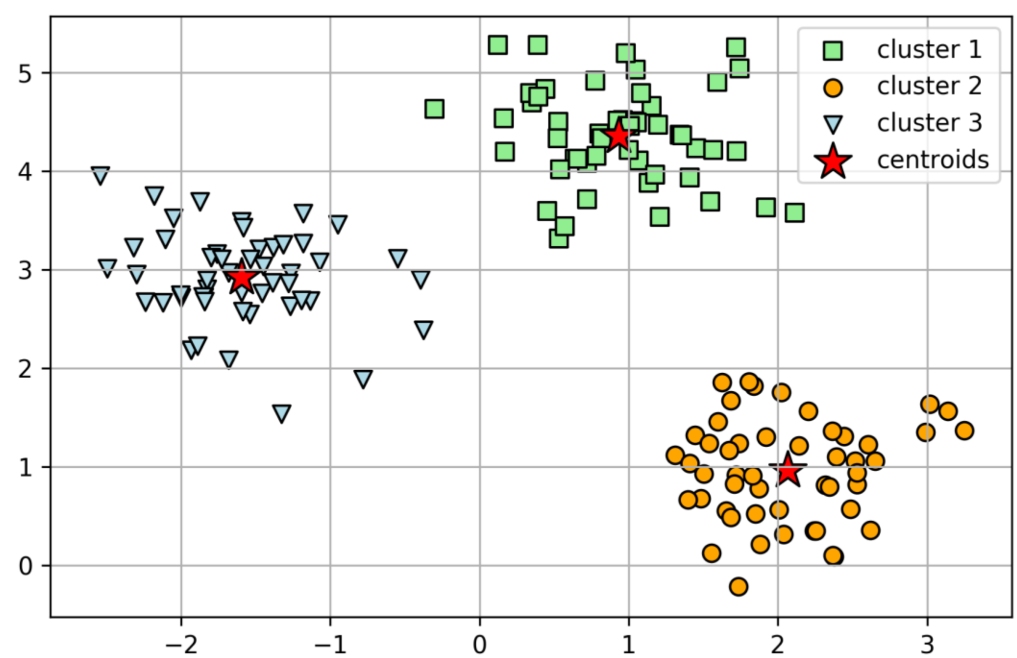
% lower status of the population [LSTAT]
Price in $1000s [MEDV]

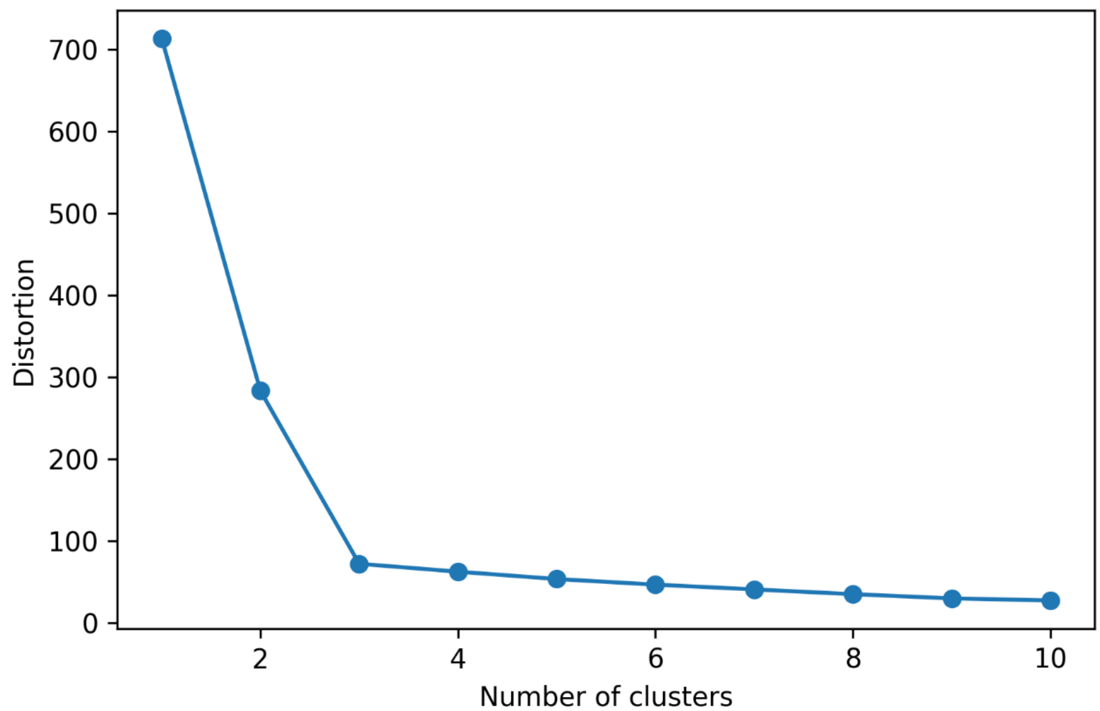


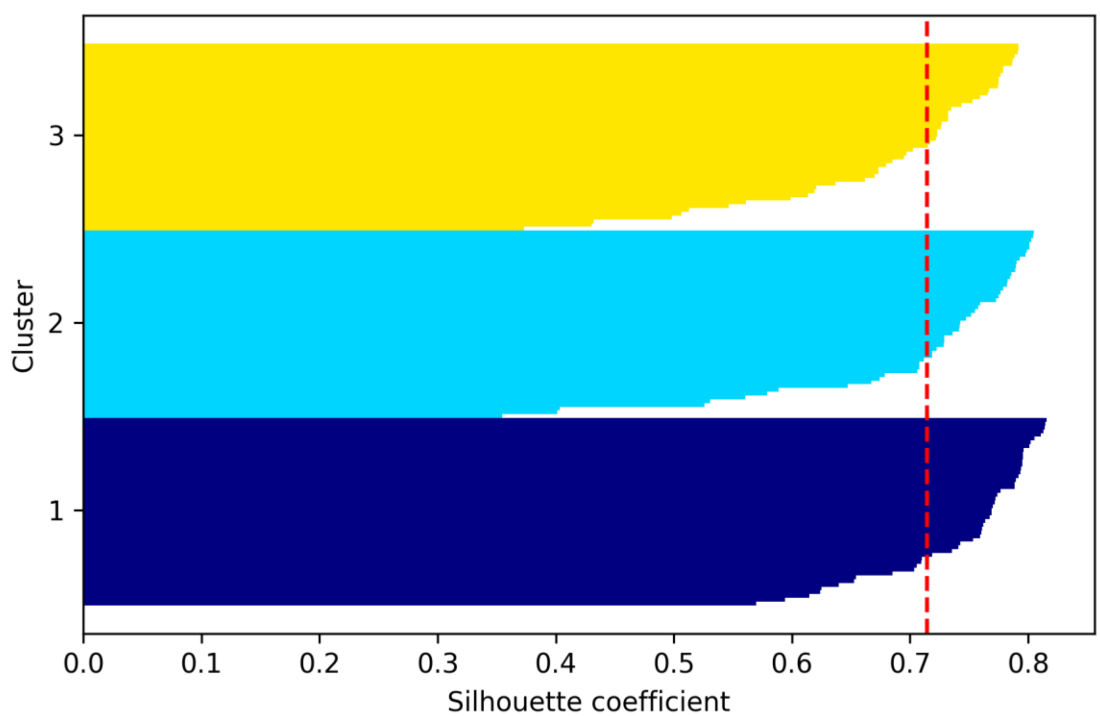
training data
test data
Residuals
Predicted values
20
15
10
5
0
−5
−10
−10
0
10
20
30
40
50

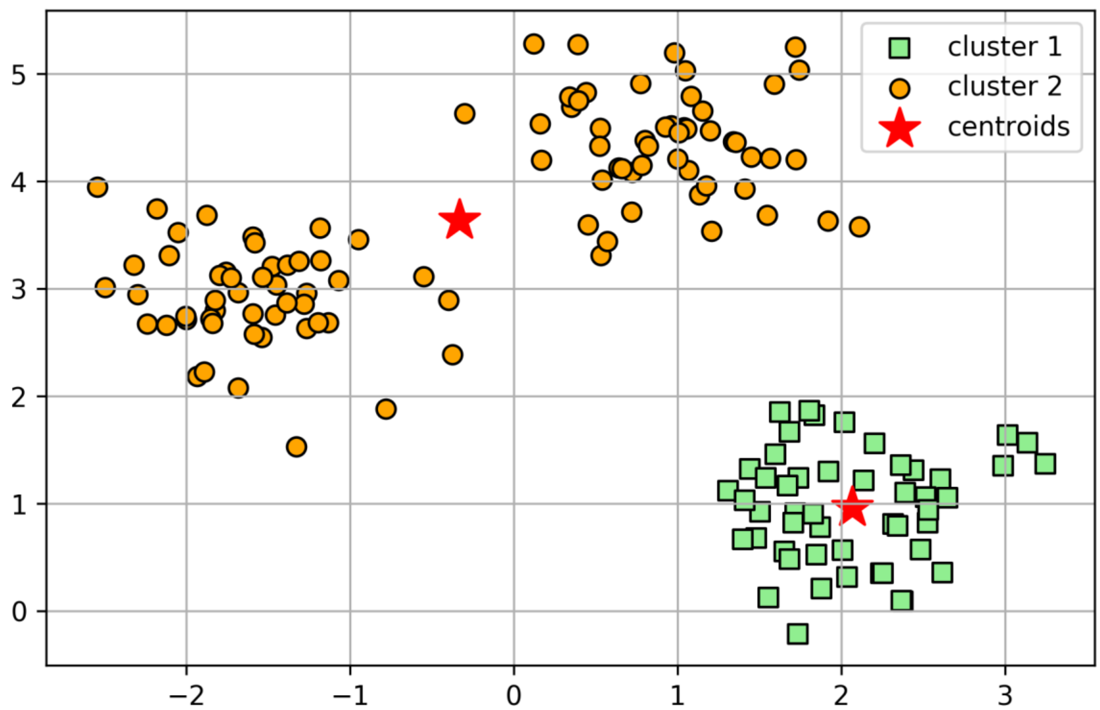
Chapter 11: Working with Unlabeled Data – Clustering Analysis



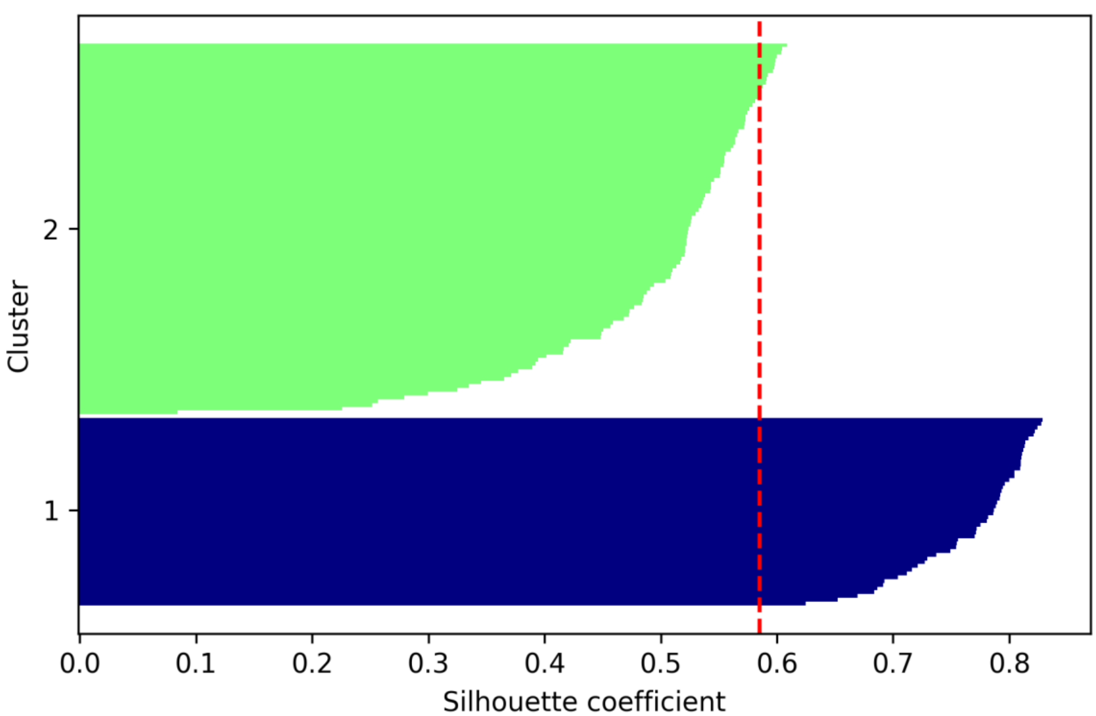



Distortion
Number of clusters
0
100
200
300
400
500
600
700
2
4
6
8
10

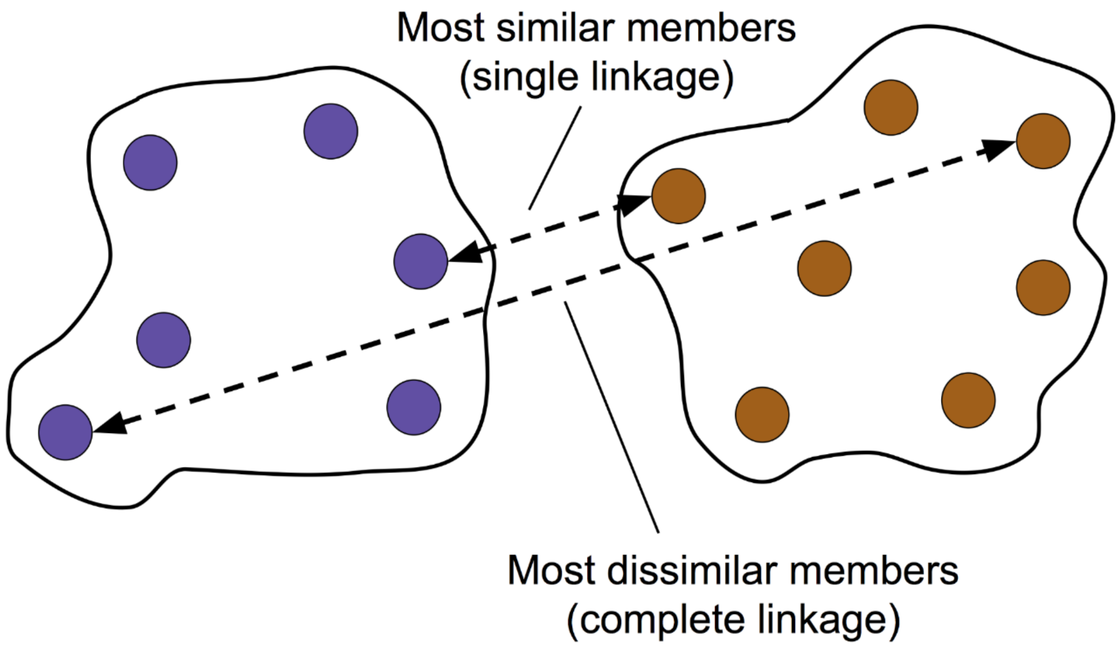

Cluster
Silhouette coefficient
1
2
3
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8



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



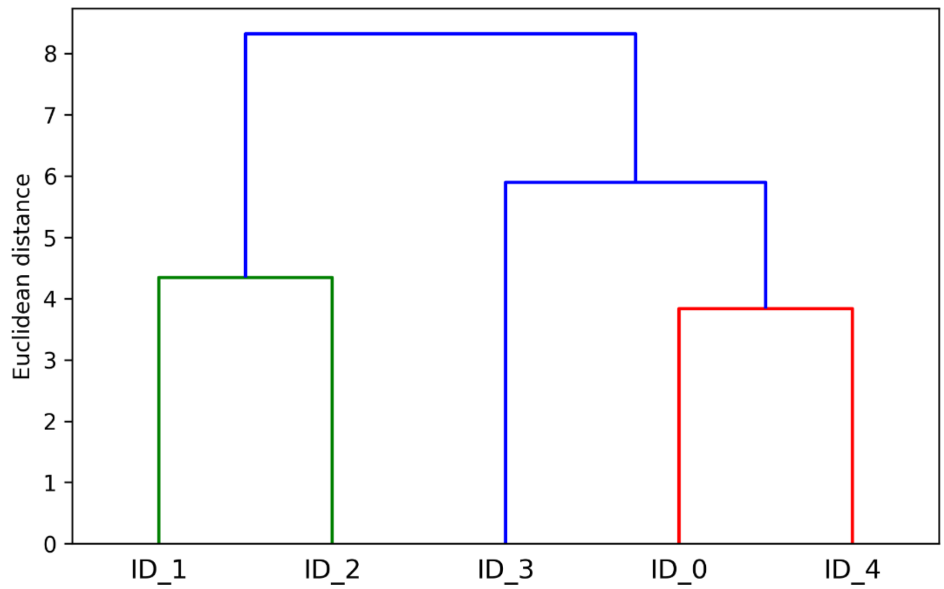
Cluster
2
1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Silhouette coefficient

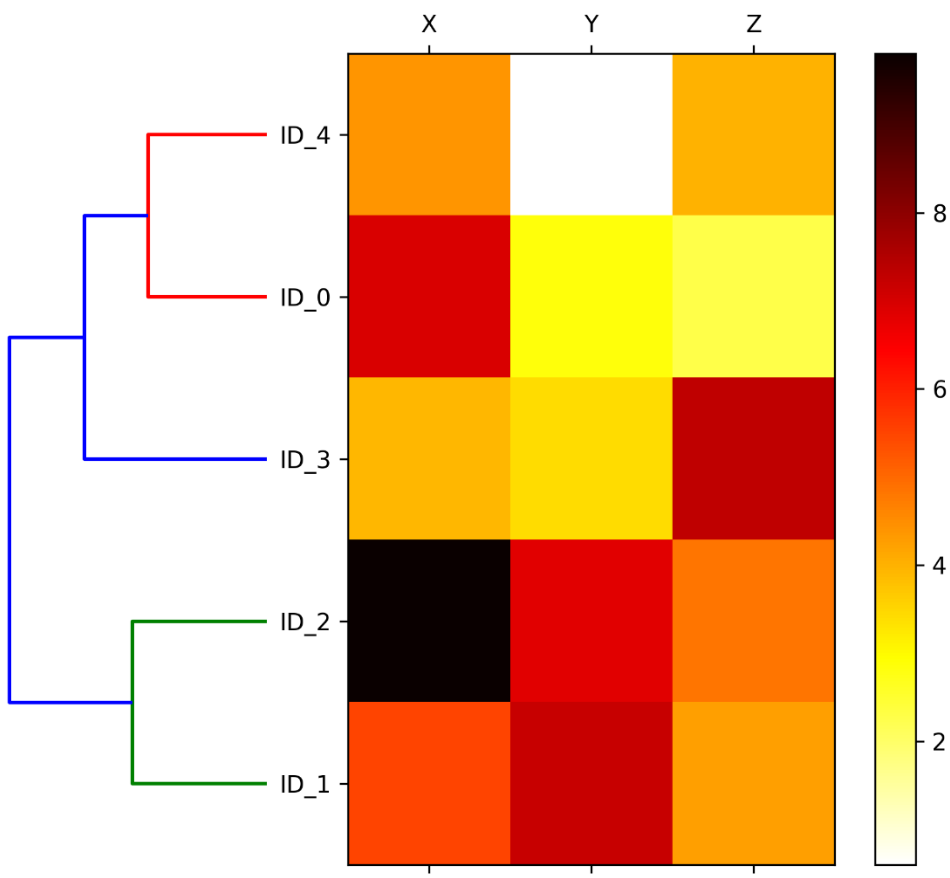


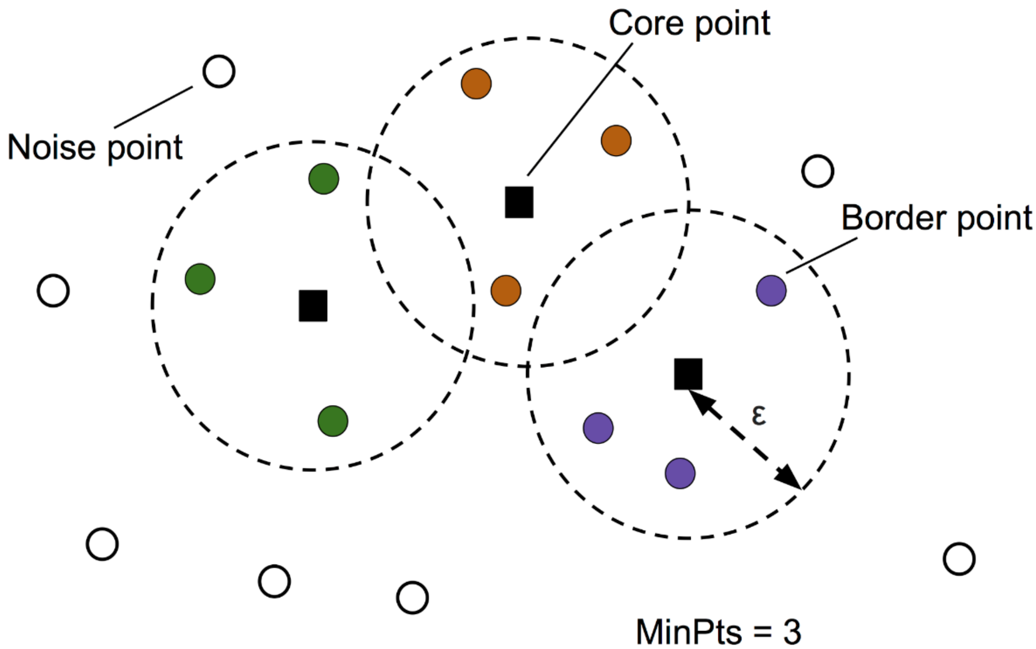
	X	Y	Z
ID_0	6.964692	2.861393	2.268515
ID_1	5.513148	7.194690	4.231065
ID_2	9.807642	6.848297	4.809319
ID_3	3.921175	3.431780	7.290497
ID_4	4.385722	0.596779	3.980443

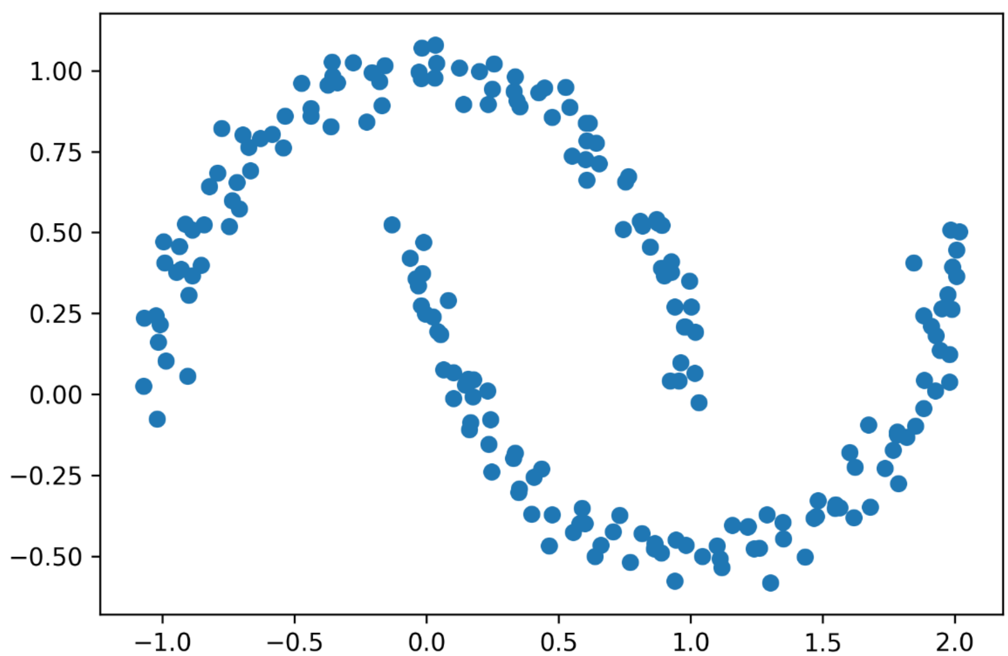
	ID_0	ID_1	ID_2	ID_3	ID_4
ID_0	0.000000	4.973534	5.516653	5.899885	3.835396
ID_1	4.973534	0.000000	4.347073	5.104311	6.698233
ID_2	5.516653	4.347073	0.000000	7.244262	8.316594
ID_3	5.899885	5.104311	7.244262	0.000000	4.382864
ID_4	3.835396	6.698233	8.316594	4.382864	0.000000

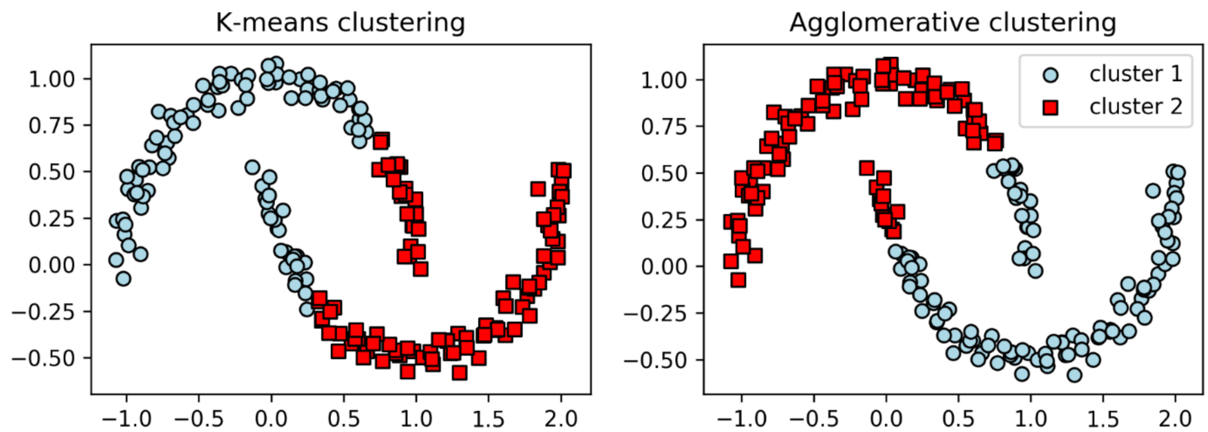
	row label 1	row label 2	distance	no. of items in clust.
cluster 1	0.0	4.0	3.835396	2.0
cluster 2	1.0	2.0	4.347073	2.0
cluster 3	3.0	5.0	5.899885	3.0
cluster 4	6.0	7.0	8.316594	5.0


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

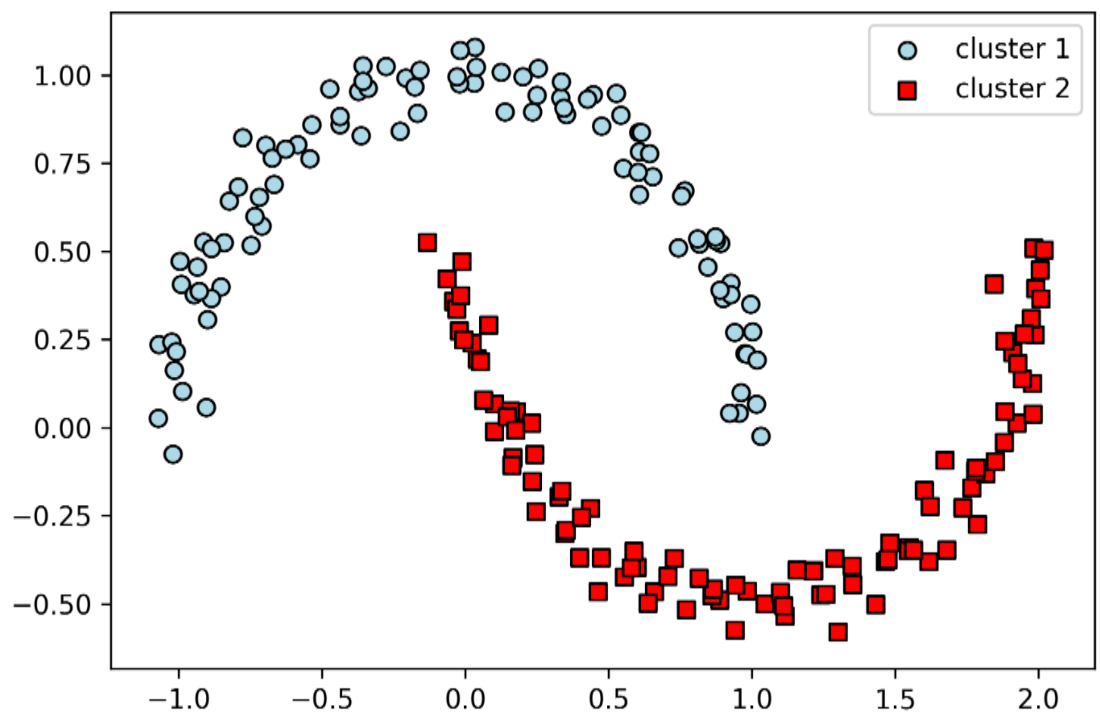

X
Y
Z
ID_4
ID_0
ID_3
ID_2
ID_1
2
4
6
8


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

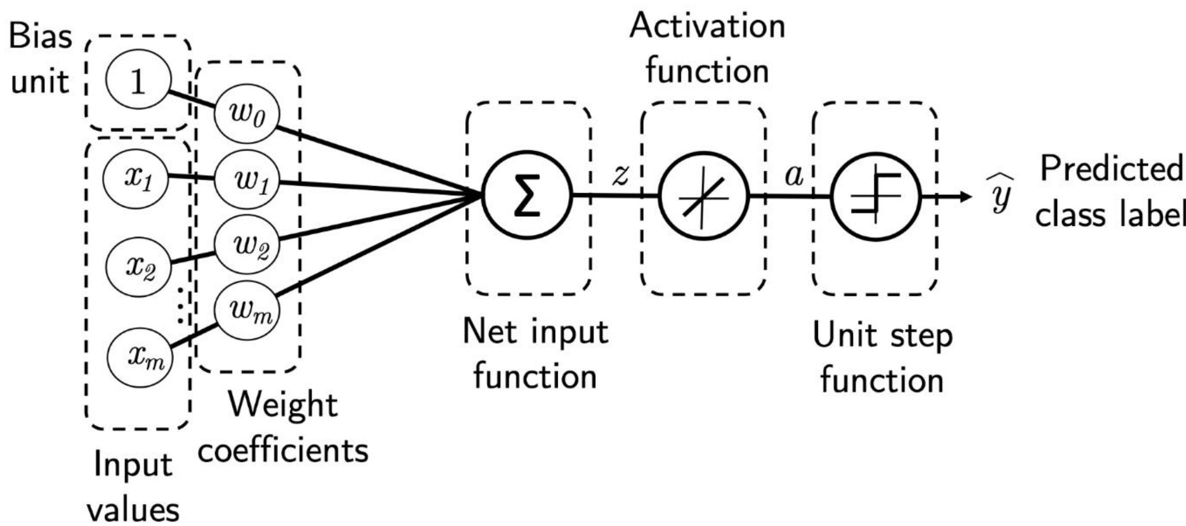


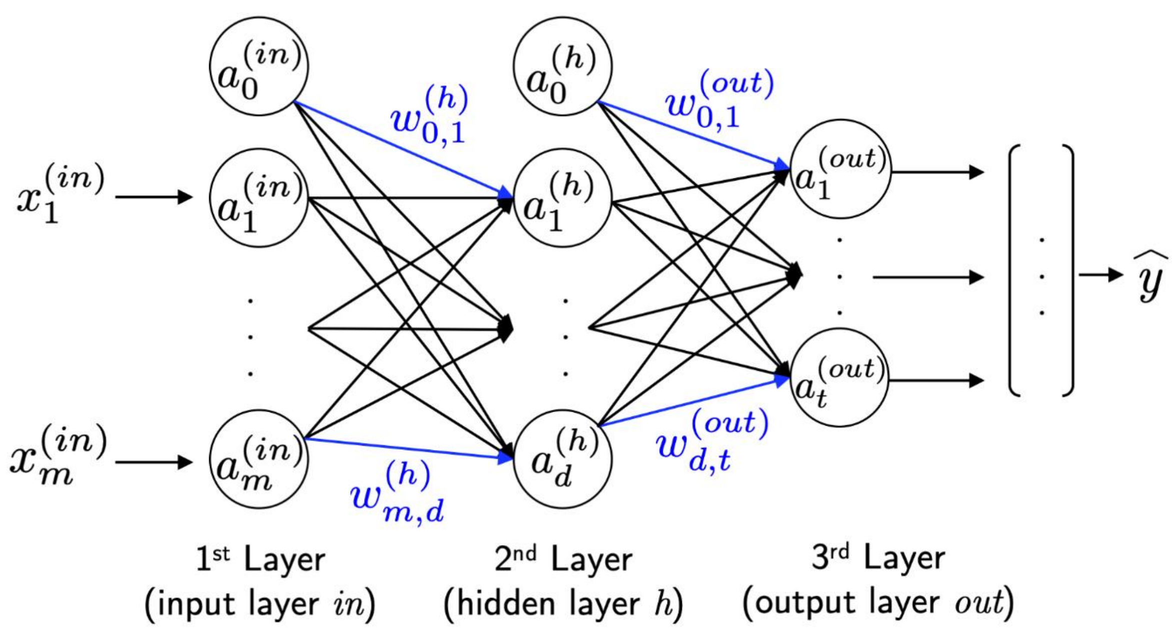
K-means clustering
Agglomerative clustering
cluster 1
cluster 2
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

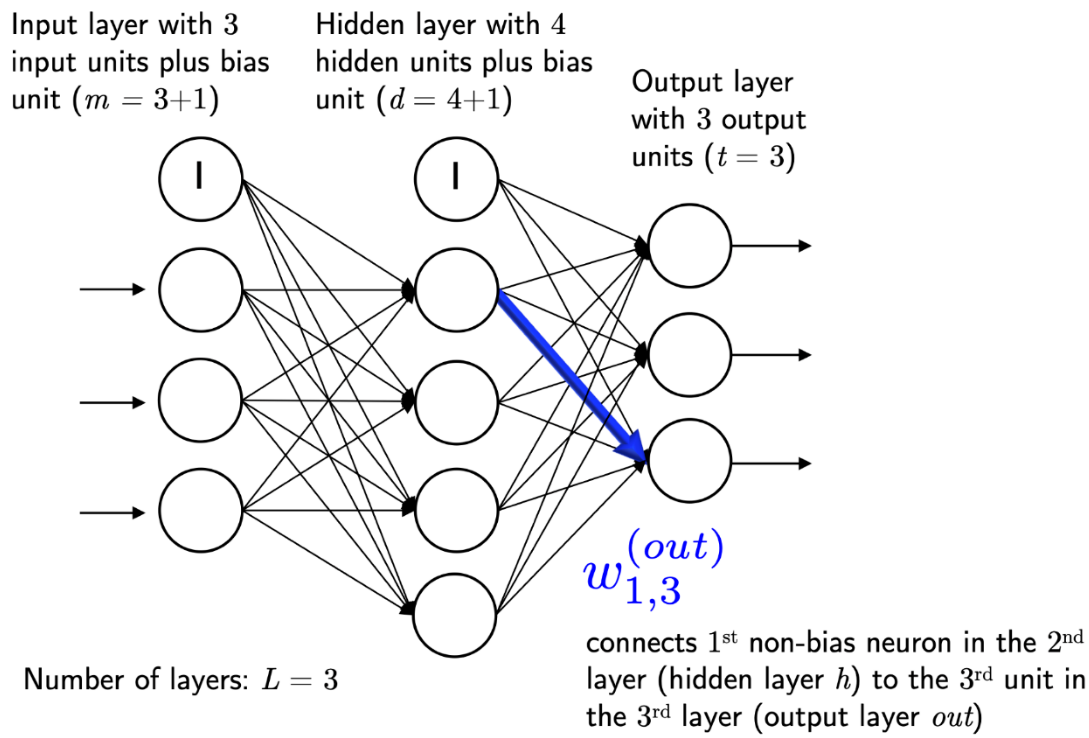


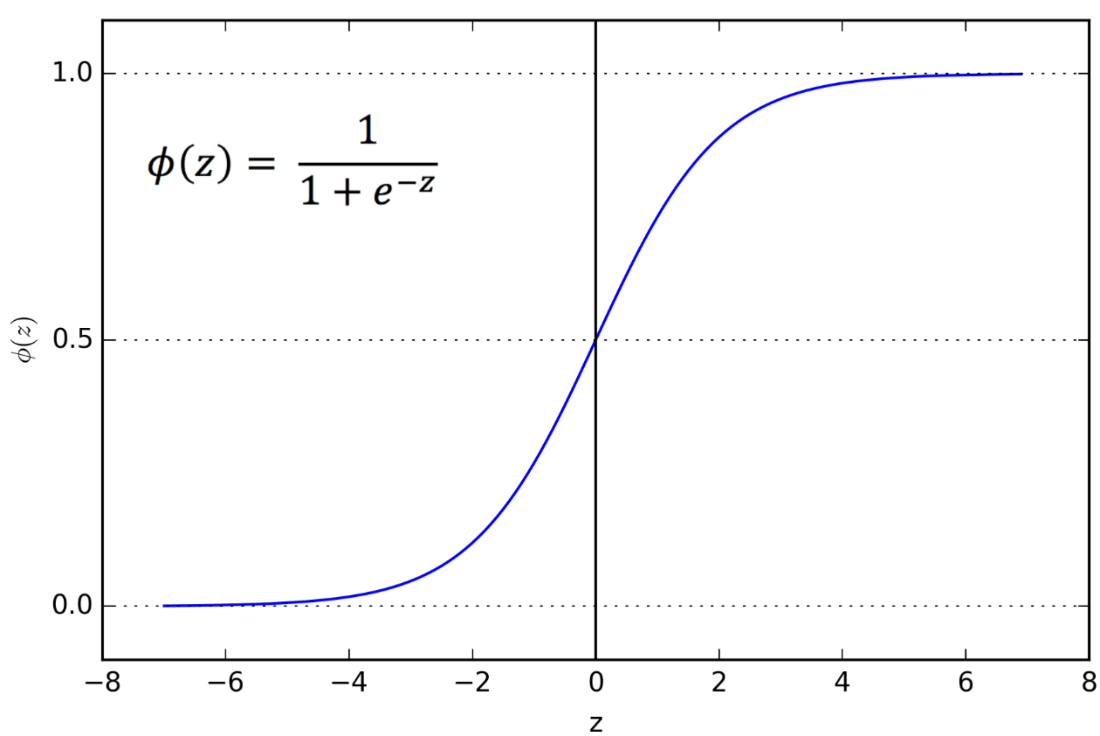
cluster 1
cluster 2
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

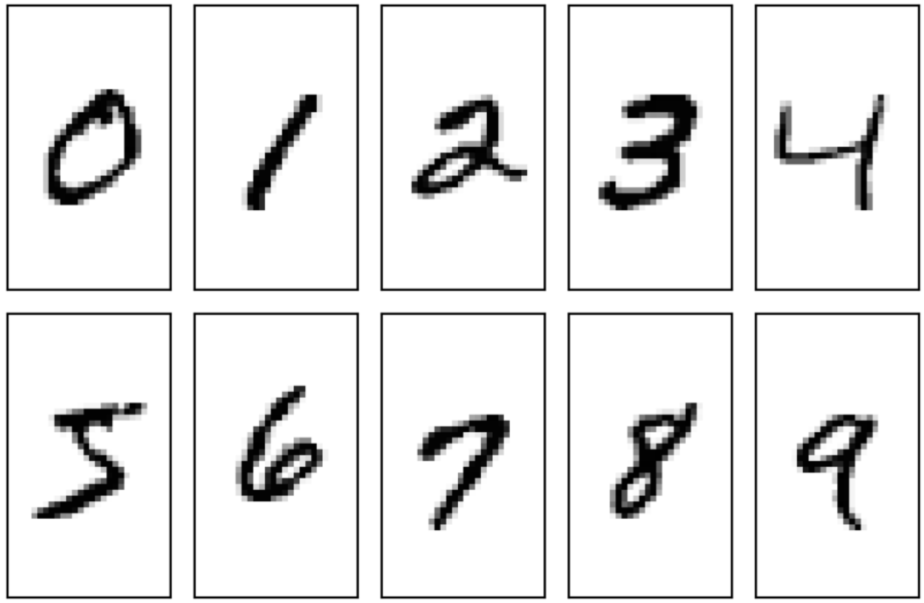
Chapter 12: Implementing a Multilayer Artificial Neural Network from Scratch

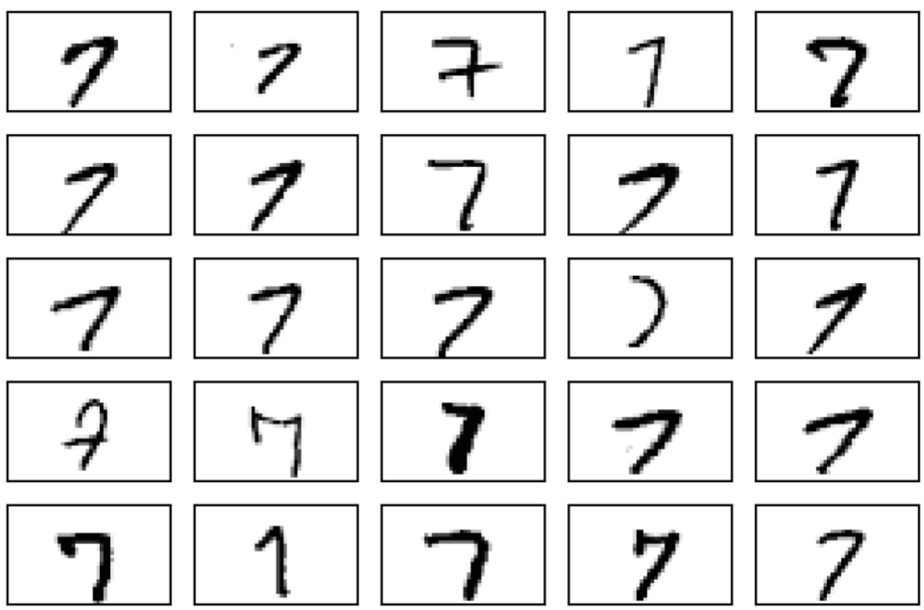


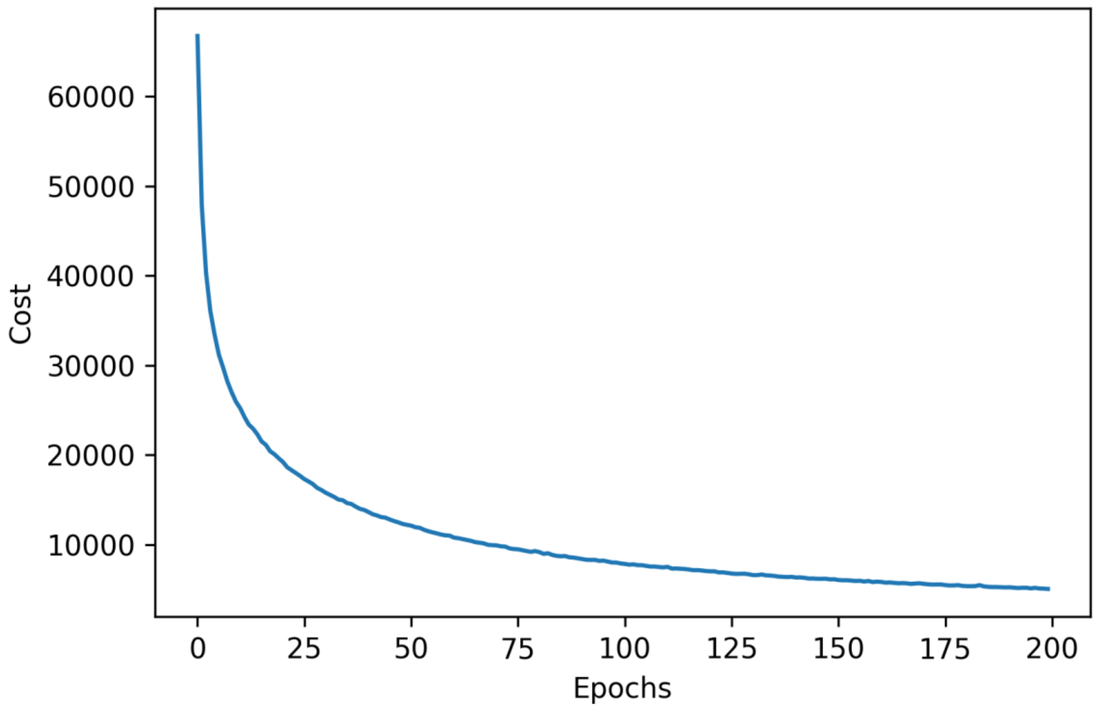


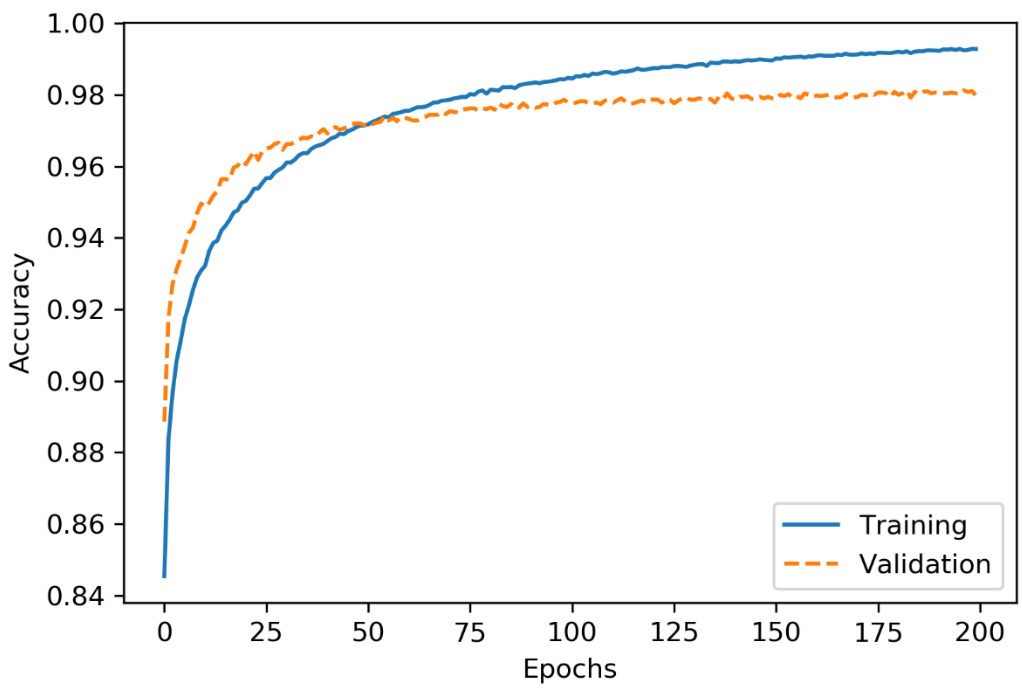

Input layer with 3 input units plus bias unit ($m = 3+1$)
Hidden layer with 4 hidden units plus bias unit ($d = 4+1$)
Output layer with 3 output units ($t = 3$)
I
I
$w_{1,3}^{(out)}$
connects 1st non-bias neuron in the 2nd layer (hidden layer h) to the 3rd unit in the 3rd layer (output layer out)
Number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

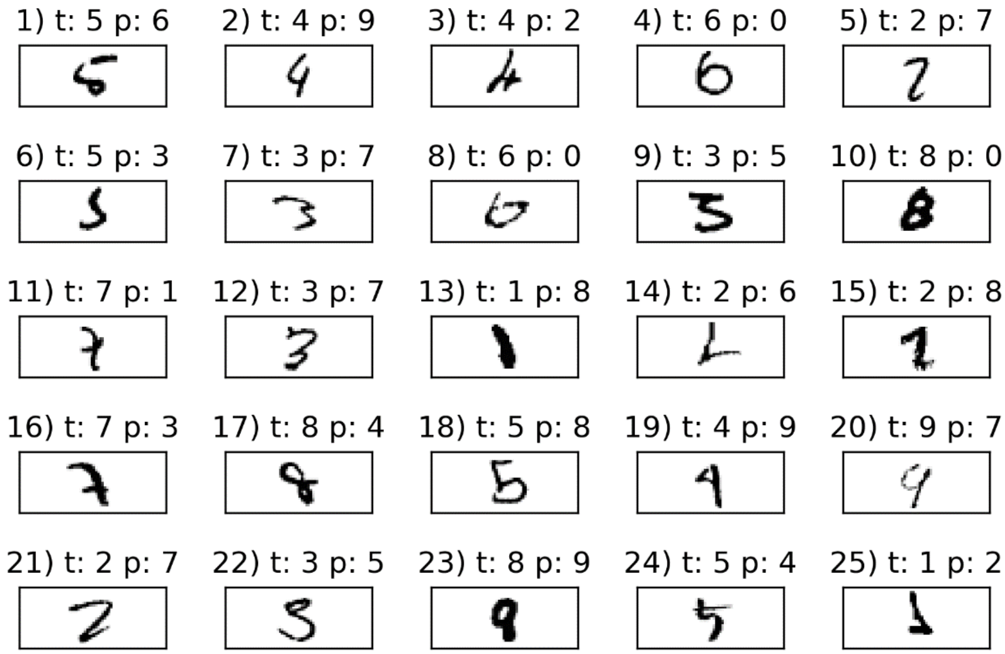

$\phi(z) = \frac{1}{1+e^{-z}}$
1.0
0.5
0.0
$\phi(z)$
−8
−6
−4
−2
0
2
4
6
8
z

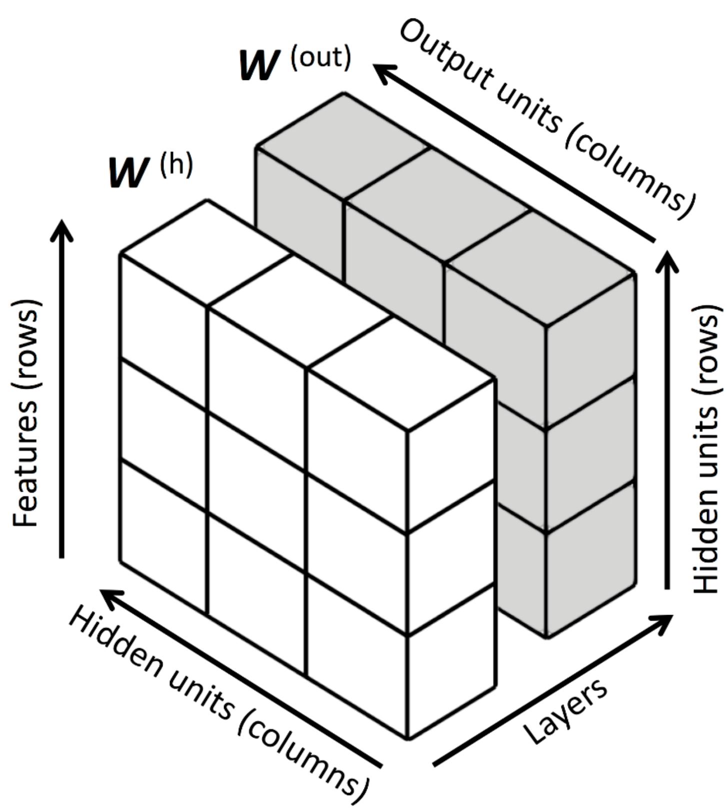


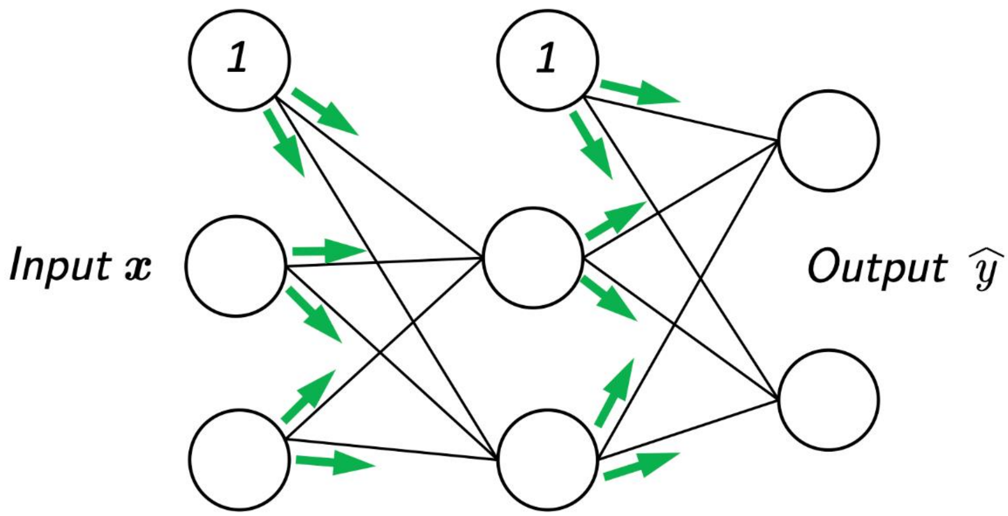



Cost
60000
50000
40000
30000
20000
10000
0
25
50
75
100
125
150
175
200
Epochs

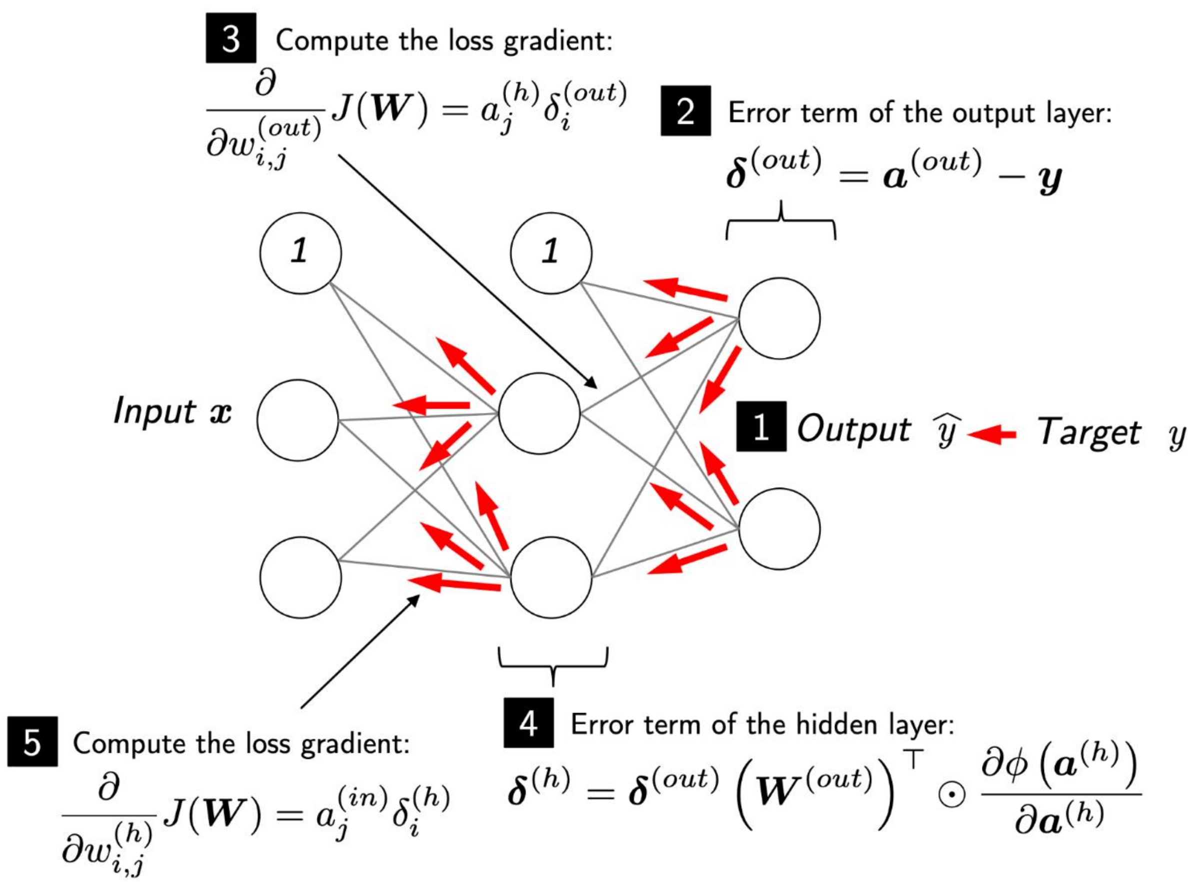

Accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
0
25
50
75
100
125
150
175
200
Epochs
Training
Validation


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

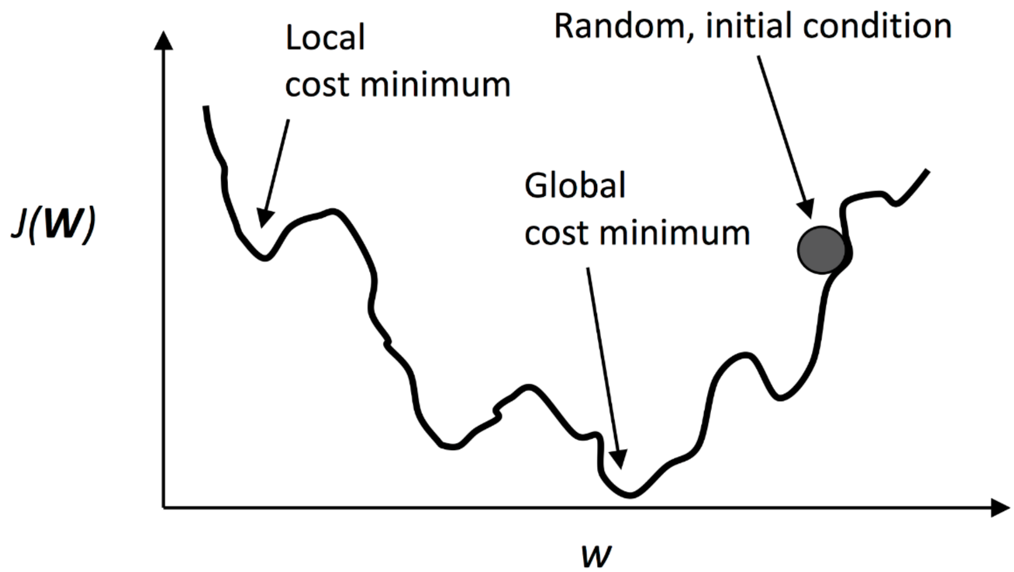

W (out)
Output units (columns)
W (h)
Features (rows)
Hidden units (rows)
Hidden units (columns)
Layers



1
1
Input $\boldsymbol{x}$
Output $\widehat{y}$

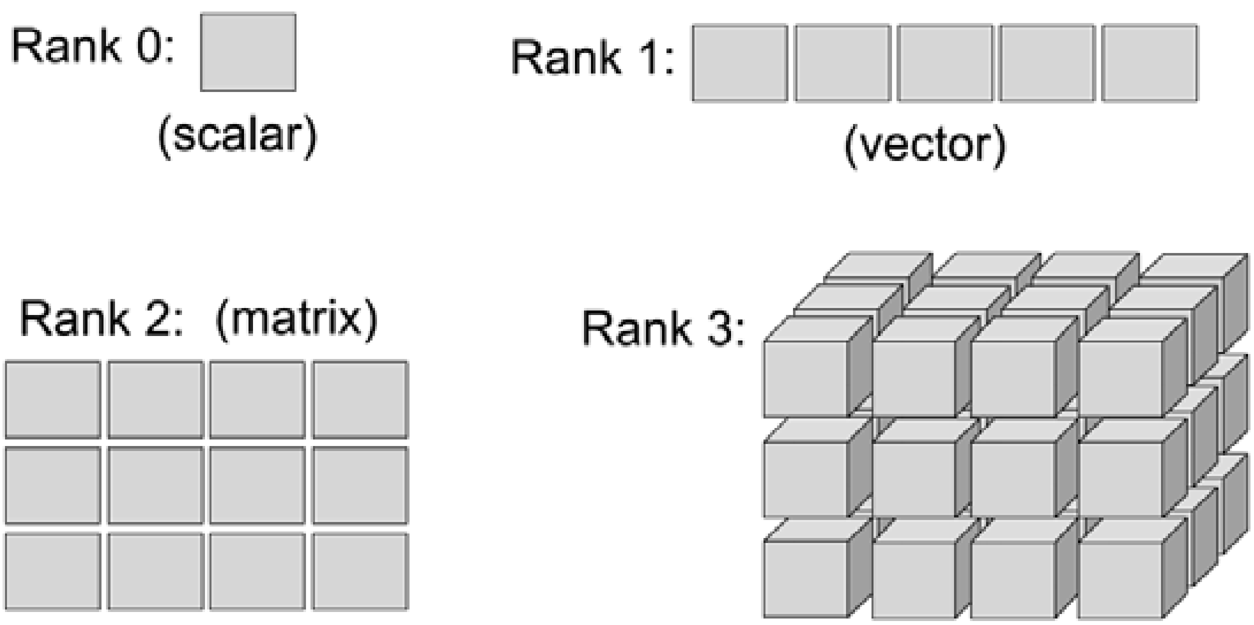


3 Compute the loss gradient:
$\frac{\partial}{\partial w_{i,j}^{(out)}} J(\boldsymbol{W}) = a_j^{(h)} \delta_i^{(out)}$
2 Error term of the output layer:
$\boldsymbol{\delta}^{(out)} = \boldsymbol{a}^{(out)} - \boldsymbol{y}$
1
1
Input $\boldsymbol{x}$
1 Output $\widehat{y}$ Target y
5 Compute the loss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

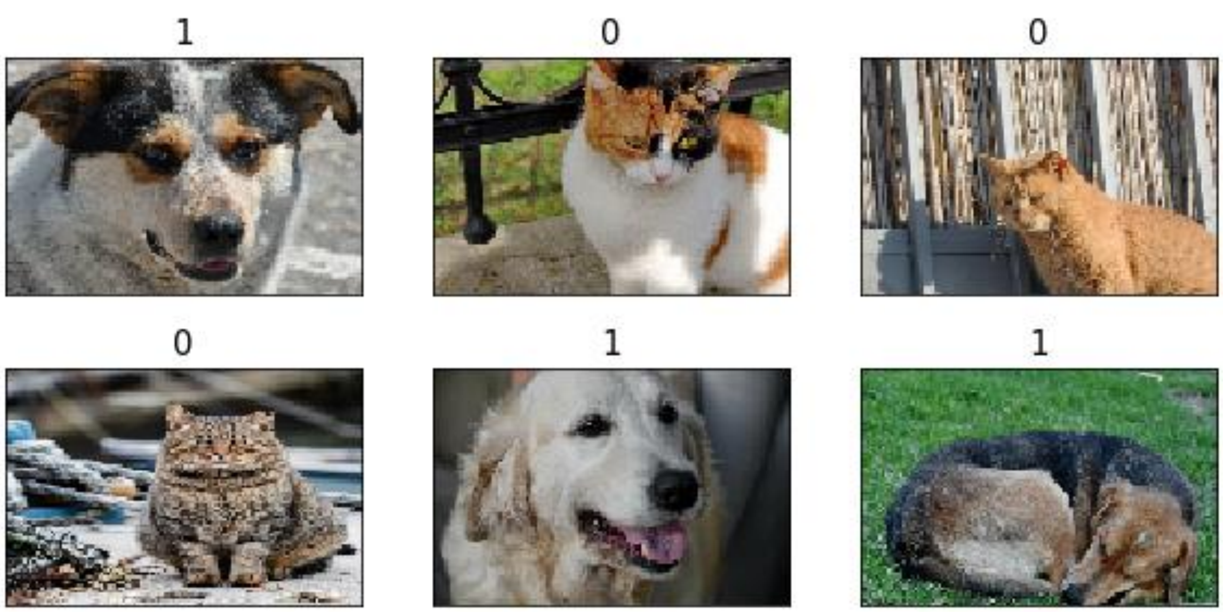

Local
cost minimum
Random, initial condition
Global
cost minimum
J(W)
w

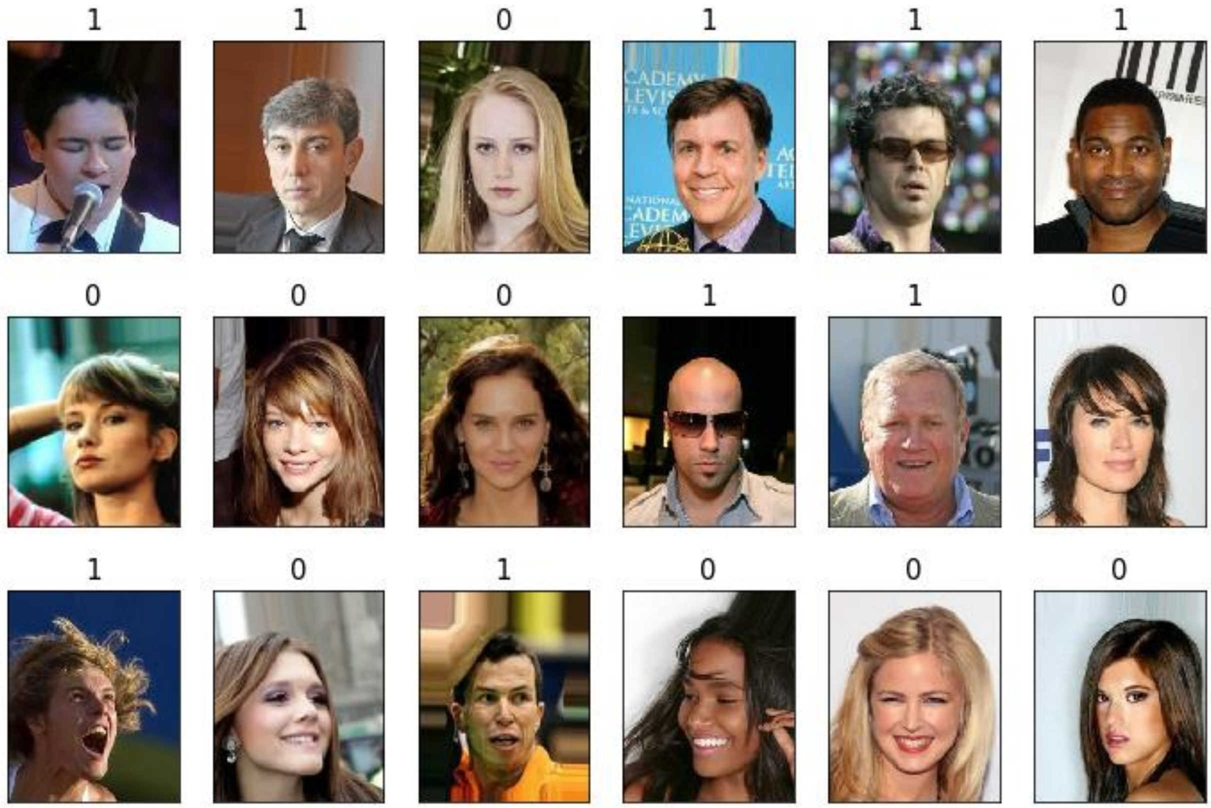
Chapter 13: Parallelizing Neural Network Training with TensorFlow

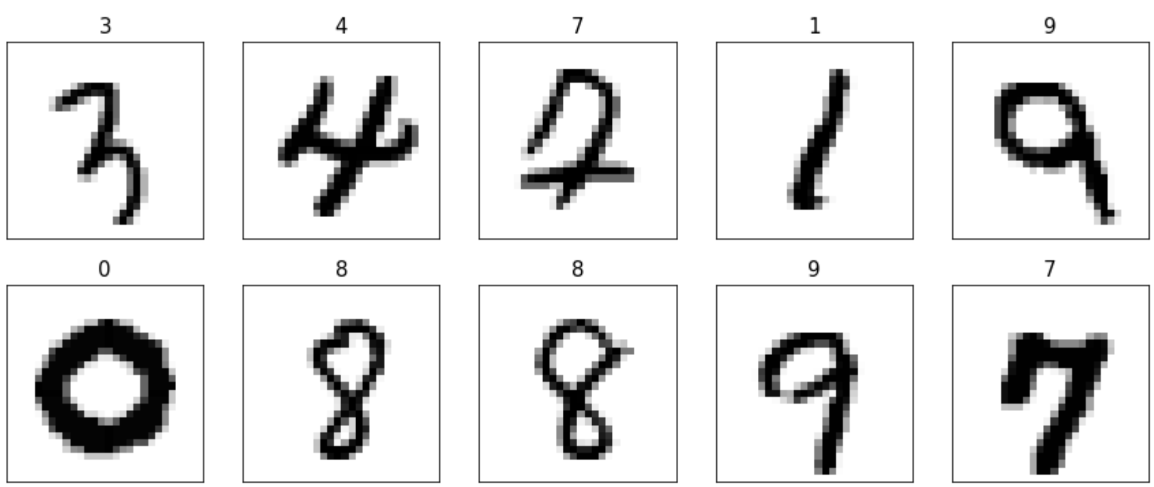
Specifications	Intel® Core™ i9-9960X X-series Processor	NVIDIA GeForce® RTX™ 2080 Ti
Base Clock Frequency	3.1 GHz	1.35 GHz
Cores	16 (32 threads)	4352
Memory Bandwidth	79.47 GB/s	616 GB/s
Floating-Point Calculations	1290 GFLOPS	13400 GFLOPS
Cost	~ $1700.00	~ $1100.00

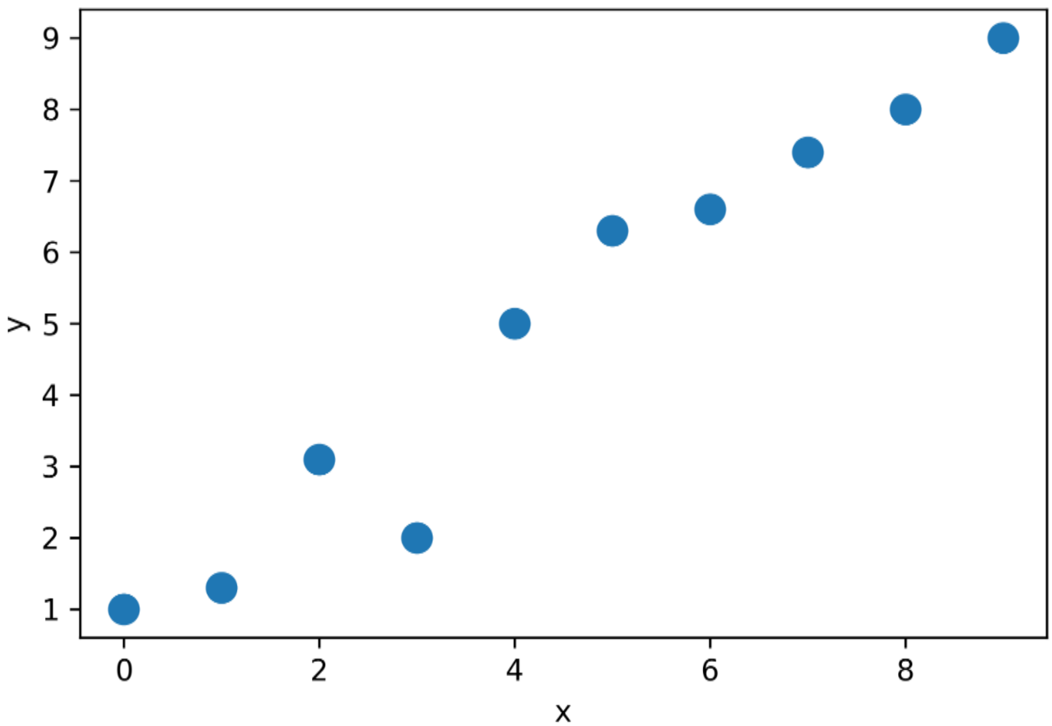



cat_dog_images/cat-01.jpg
cat_dog_images/cat-02.jpg
cat_dog_images/cat-03.jpg
cat_dog_images/dog-01.jpg
cat_dog_images/dog-02.jpg
cat_dog_images/dog-03.jpg

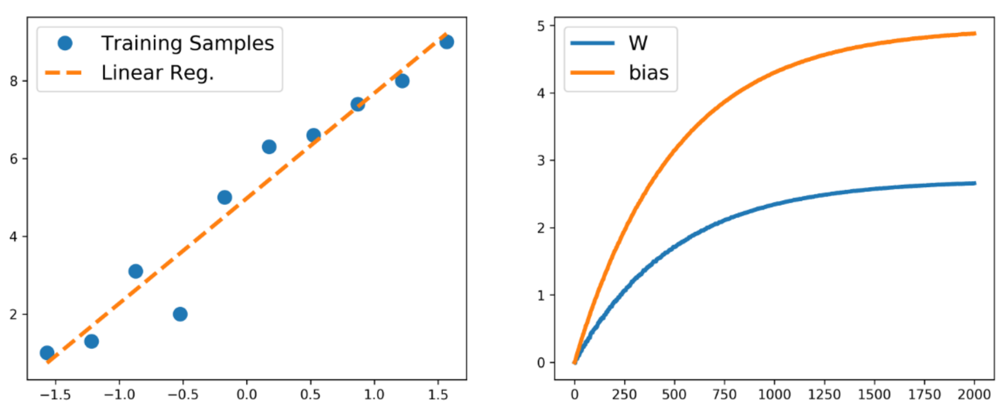

1
0
0
0
1
1


1 1 0 1 1 1
0 0 0 1 1 0
1 0 1 0 0 0

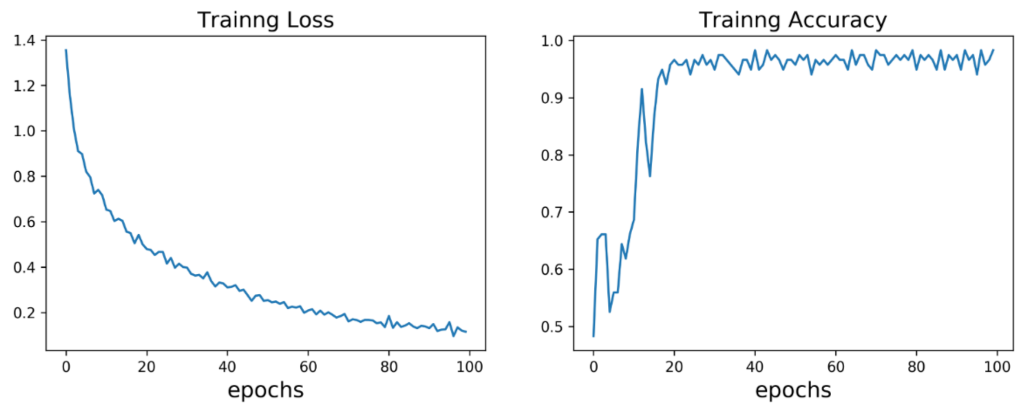

3 4 7 1 9
0 8 8 9 7

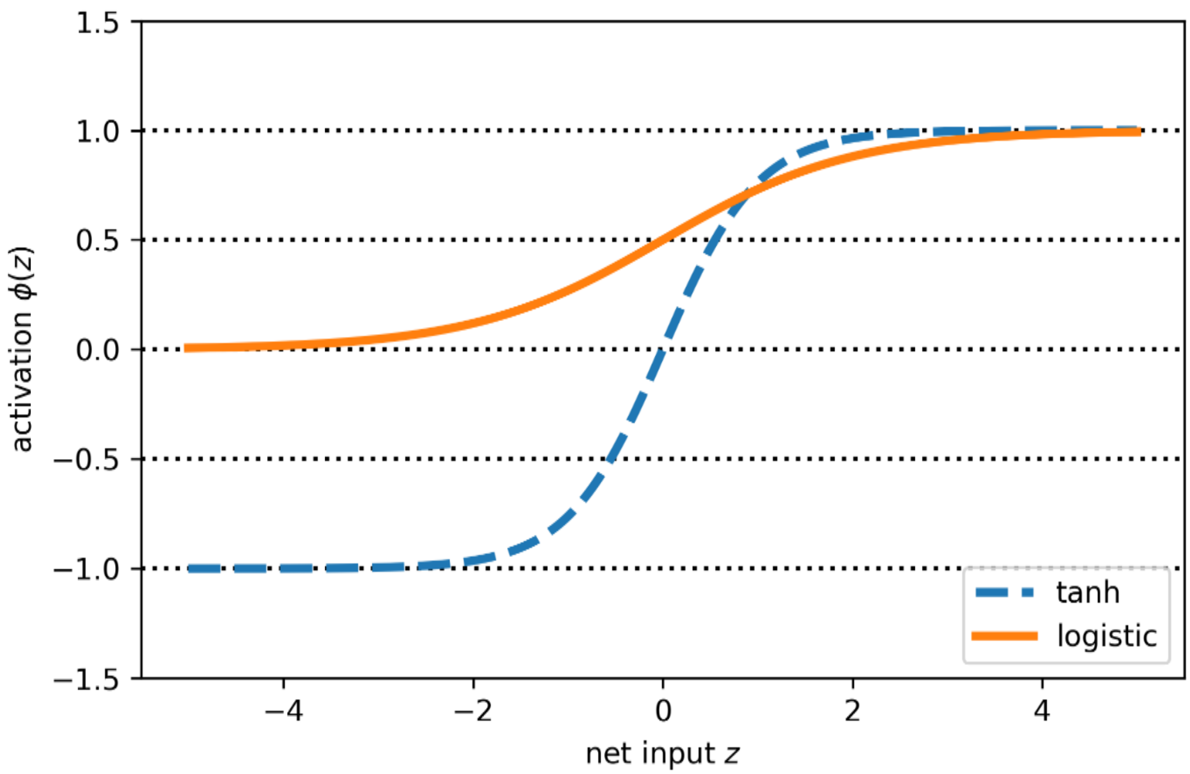


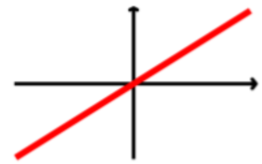
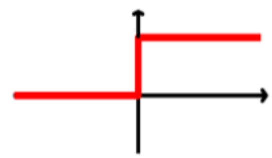
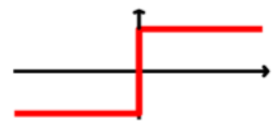
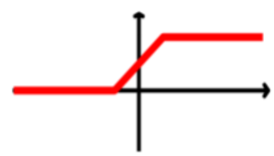
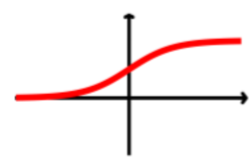
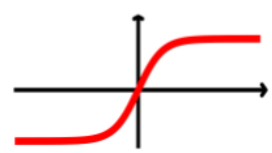
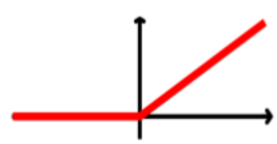
y
x



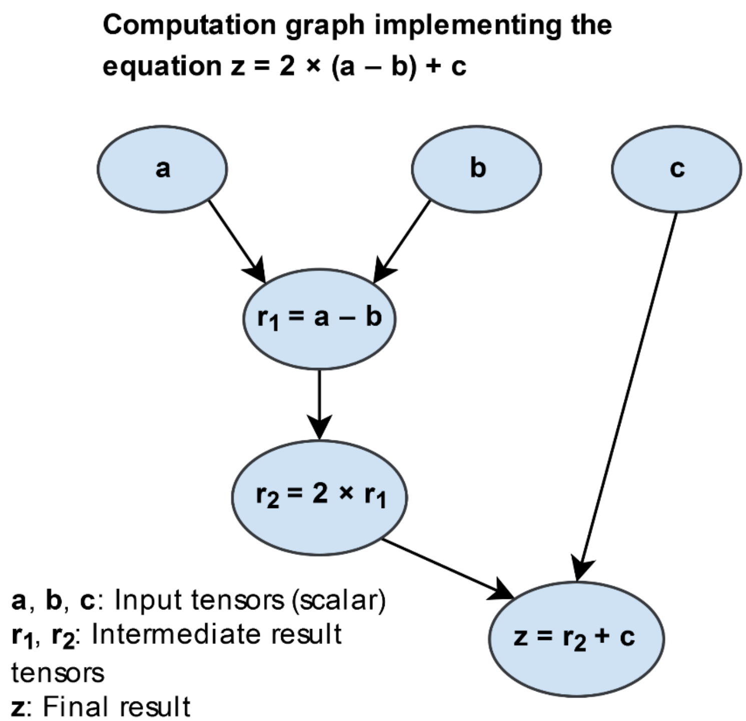
Training Samples
Linear Reg.
W
bias

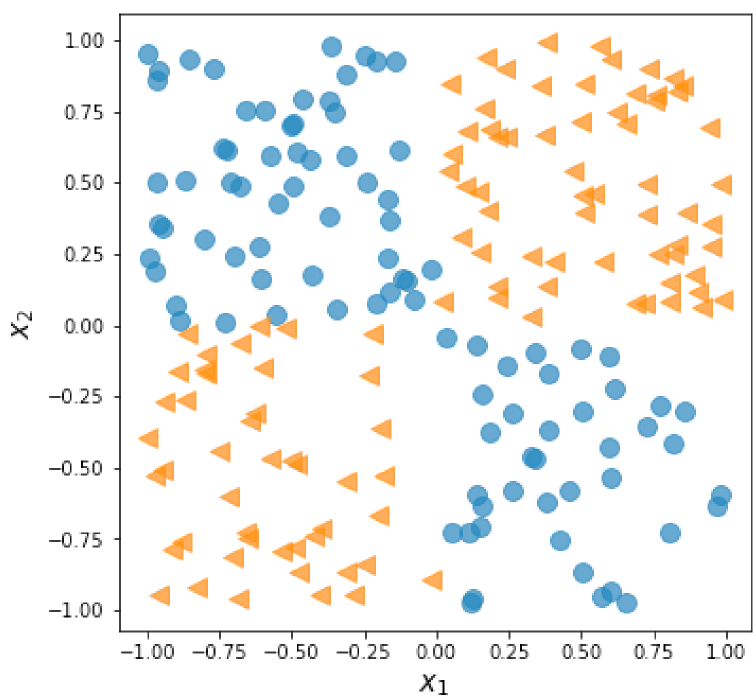

Trainng Loss
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
20
40
60
80
100
epochs
Trainng Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100
epochs

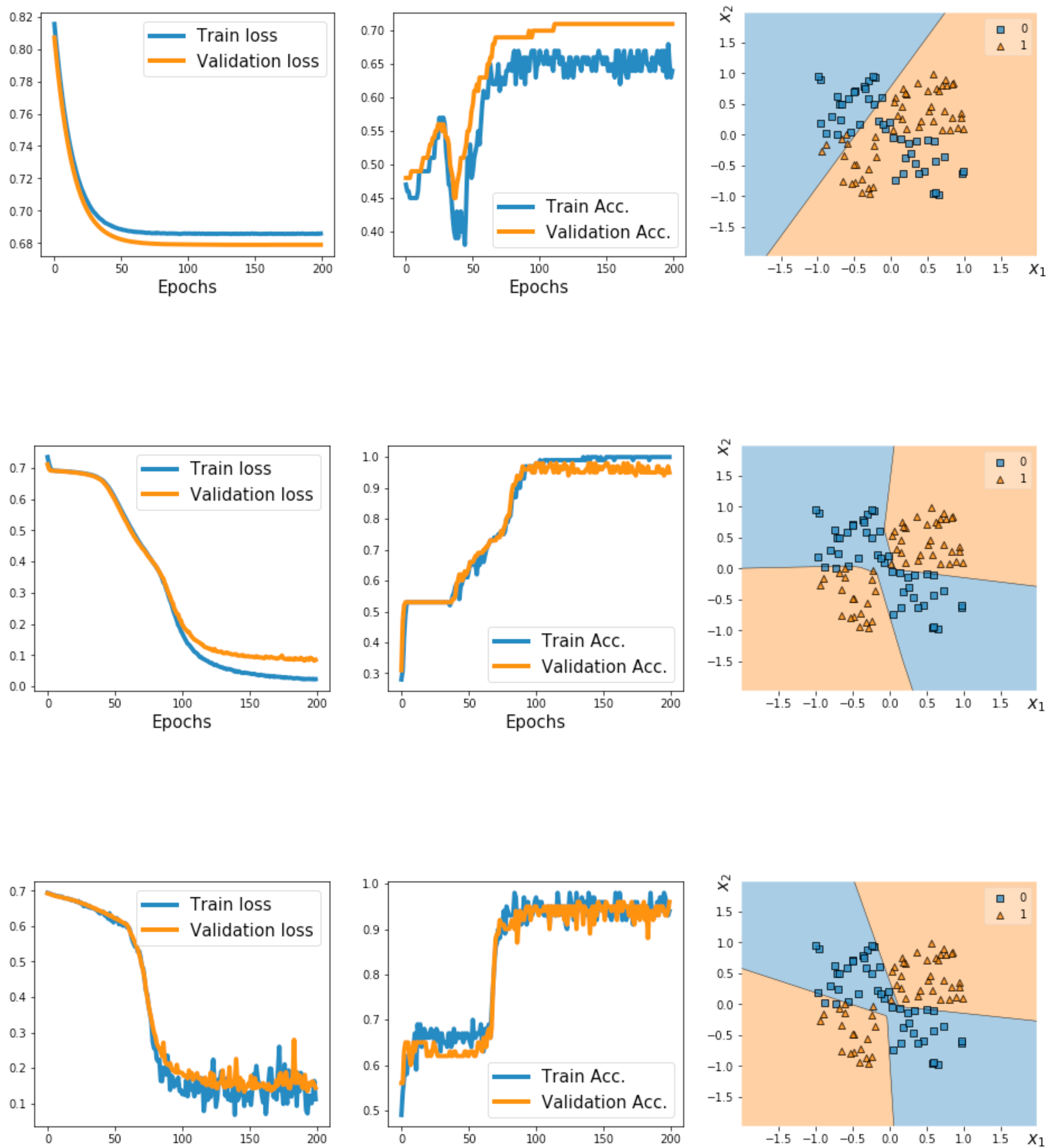

1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
activation $\phi(z)$
−4
−2
0
2
4
net input z
tanh
logistic

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	$\phi(z) = \begin{cases} 0 & z \leq -\frac{1}{2} \\ z + \frac{1}{2} & -\frac{1}{2} \leq z \leq \frac{1}{2} \\ 1 & z \geq \frac{1}{2} \end{cases}$	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	

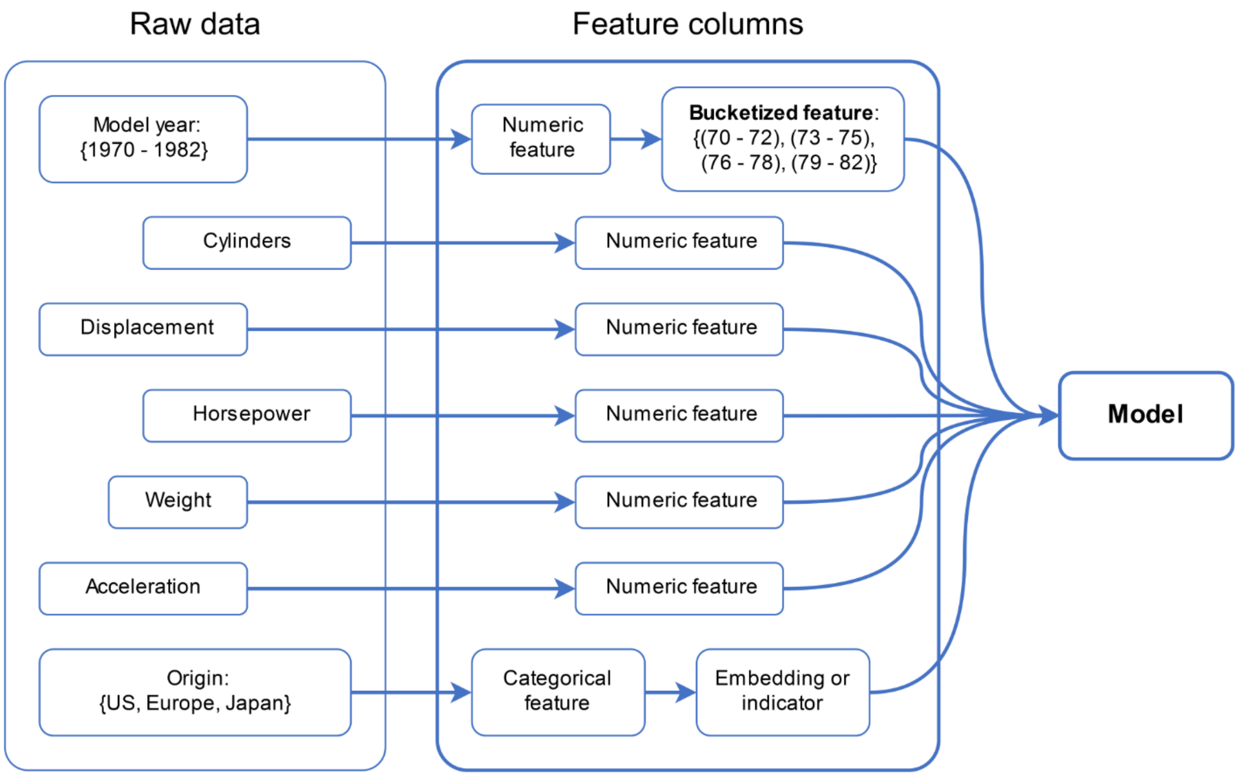
Chapter 14: Going Deeper – The Mechanics of TensorFlow





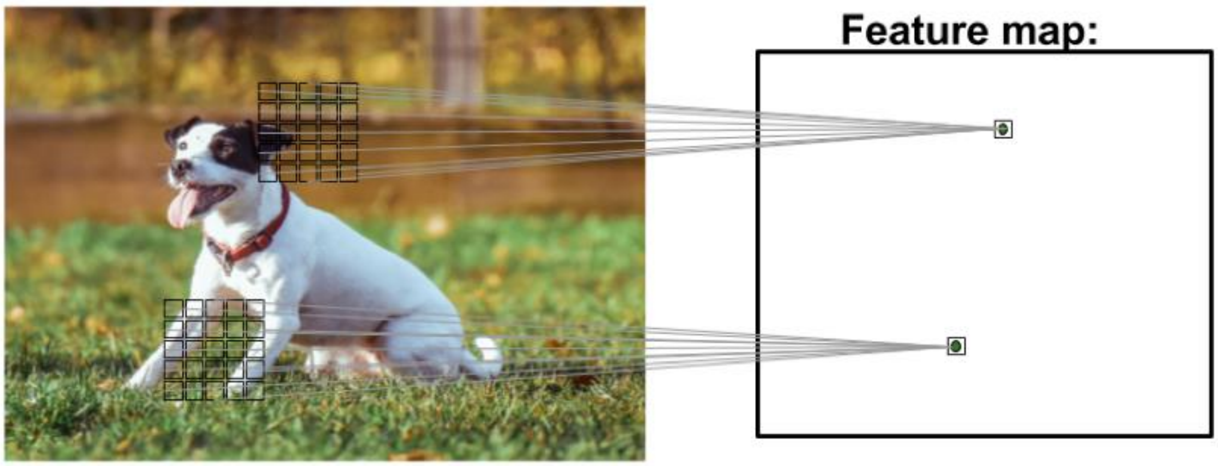


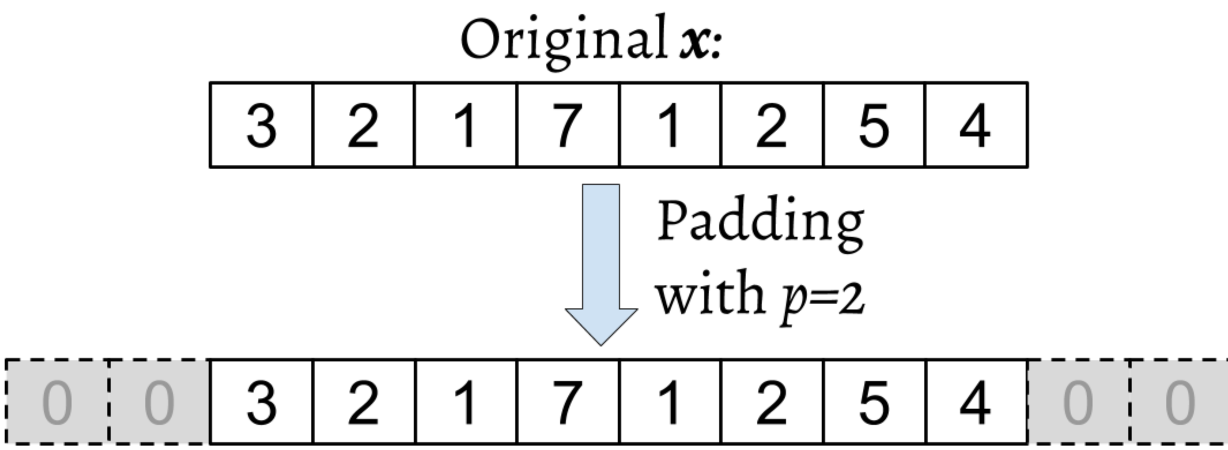
Train loss
Validation loss
Epochs
Train Acc.
Validation Acc.
Epochs
x_2
x_1
0
1
Train loss
Validation loss
Epochs
Train Acc.
Validation Acc.
Epochs
x_2
x_1
0
1
Train loss
Validation loss
Epochs
Train Acc.
Validation Acc.
Epochs
x_2
x_1
0
1

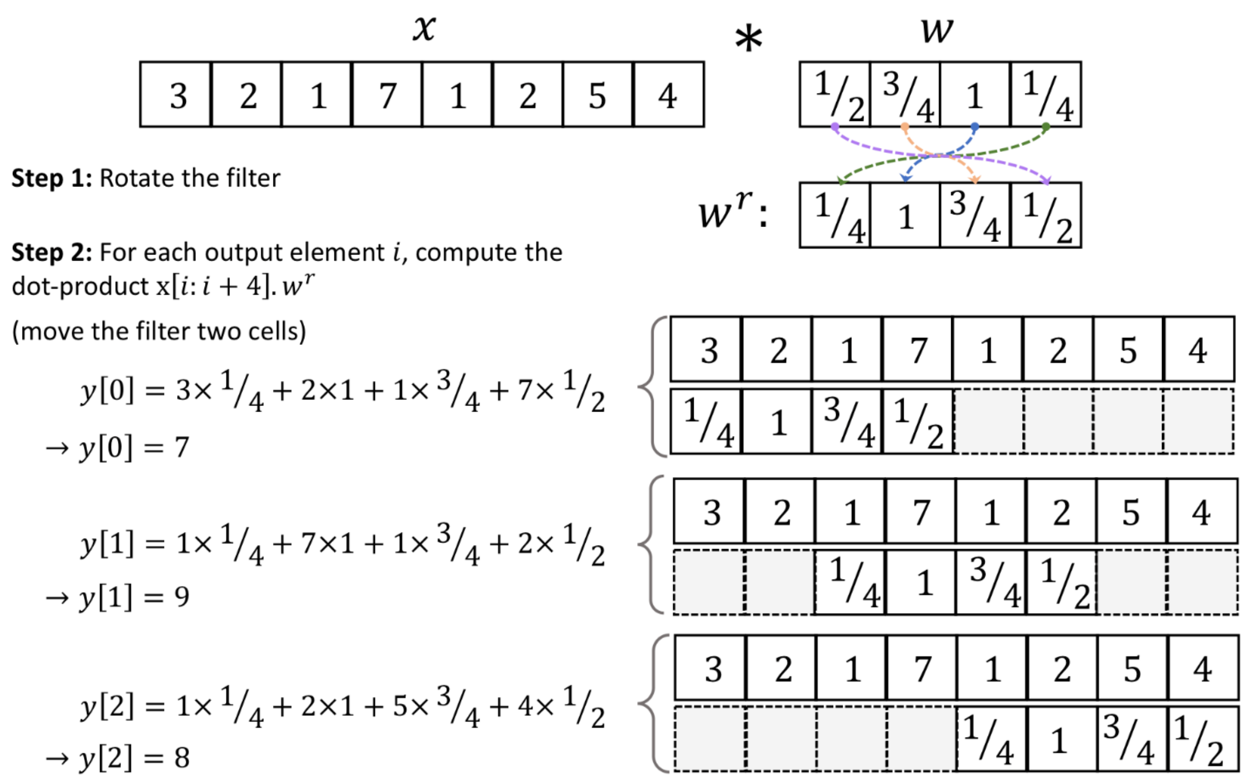


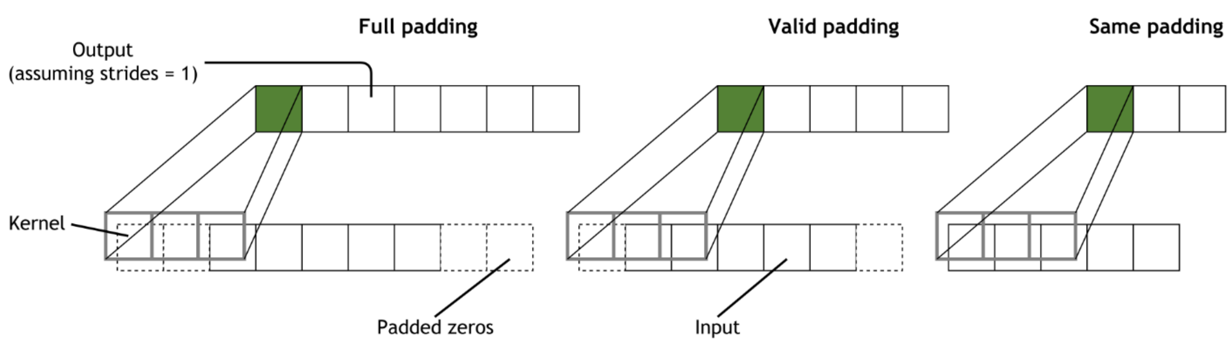
	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	ModelYear	Origin
203	28.0	-0.824303	-0.901020	-0.736562	-0.950031	0.255202	76	3
255	19.4	0.351127	0.413800	-0.340982	0.293190	0.548737	78	1
72	13.0	1.526556	1.144256	0.713897	1.339617	-0.625403	72	1
235	30.5	-0.824303	-0.891280	-1.053025	-1.072585	0.475353	77	1
37	14.0	1.526556	1.563051	1.636916	1.470420	-1.359240	71	1

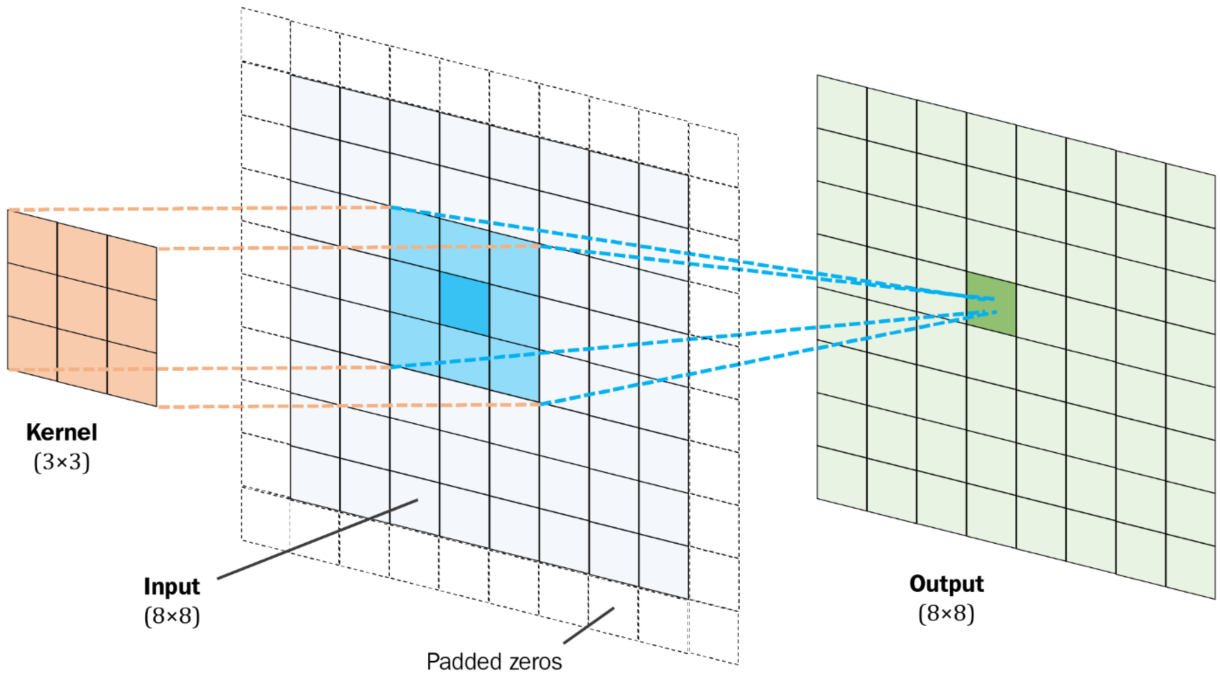
Chapter 15: Classifying Images with Deep Convolutional Neural Networks



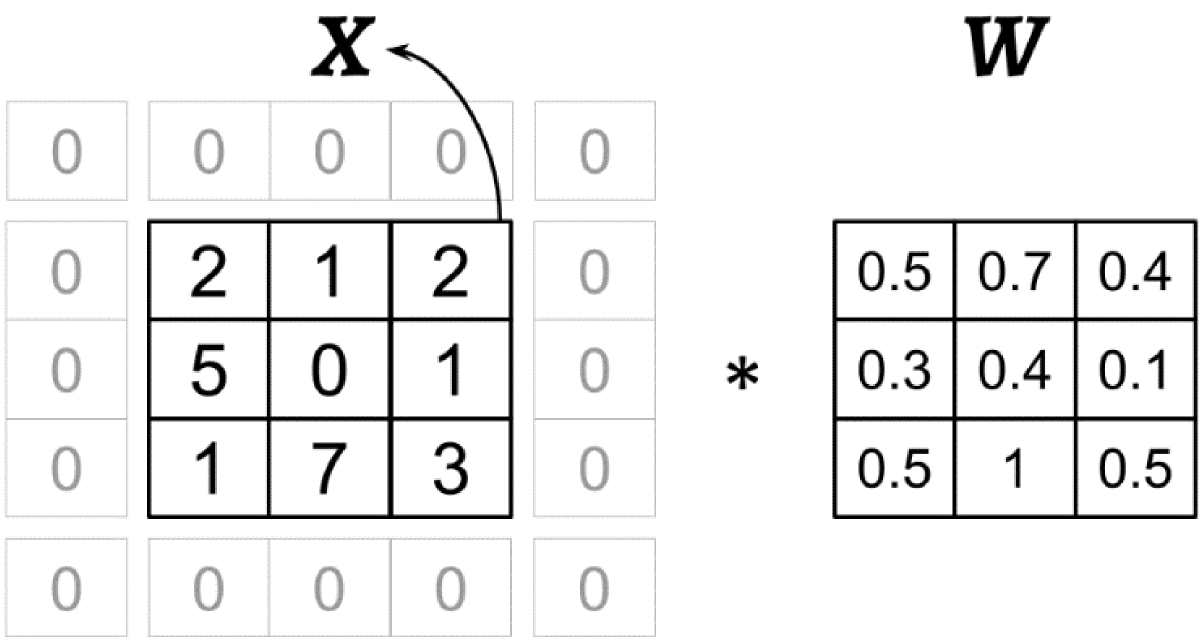



x
3 2 1 7 1 2 5 4
$*$
w
$1/2$ $3/4$ 1 $1/4$
w^r: $1/4$ 1 $3/4$ $1/2$
Step 1: Rotate the filter
Step 2: For each output element i, compute the dot-product $x[i: i+4].w^r$
(move the filter two cells)
$y[0] = 3\times {}^1/_4 + 2\times 1 + 1\times {}^3/_4 + 7\times {}^1/_2$
$\rightarrow y[0] = 7$
$y[1] = 1\times {}^1/_4 + 7\times 1 + 1\times {}^3/_4 + 2\times {}^1/_2$
$\rightarrow y[1] = 9$
$y[2] = 1\times {}^1/_4 + 2\times 1 + 5\times {}^3/_4 + 4\times {}^1/_2$
$\rightarrow y[2] = 8$

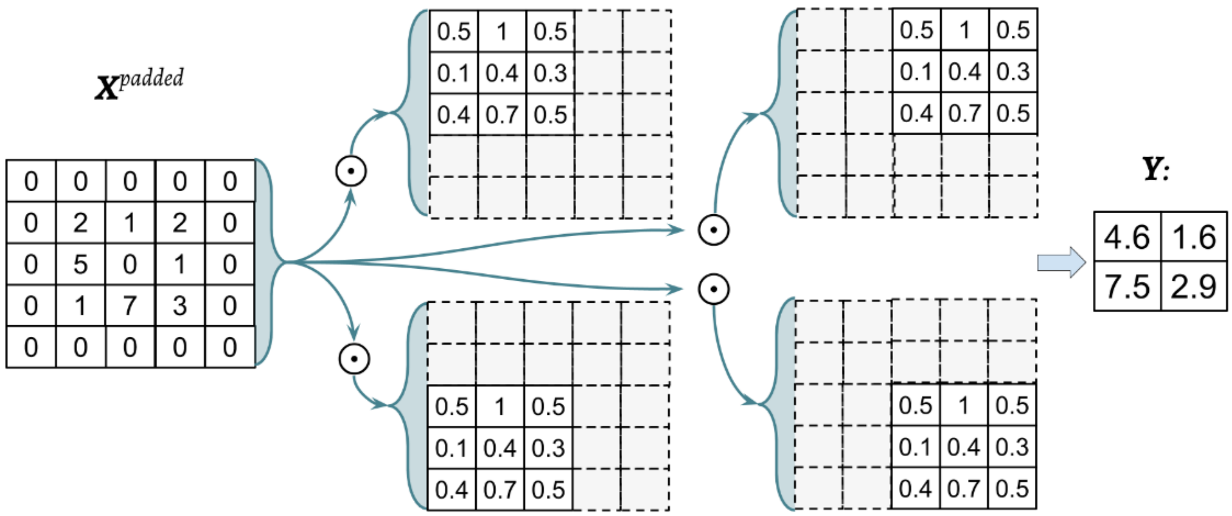

Full padding
Valid padding
Same padding
Output
(assuming strides = 1)
Kernel
Padded zeros
Input



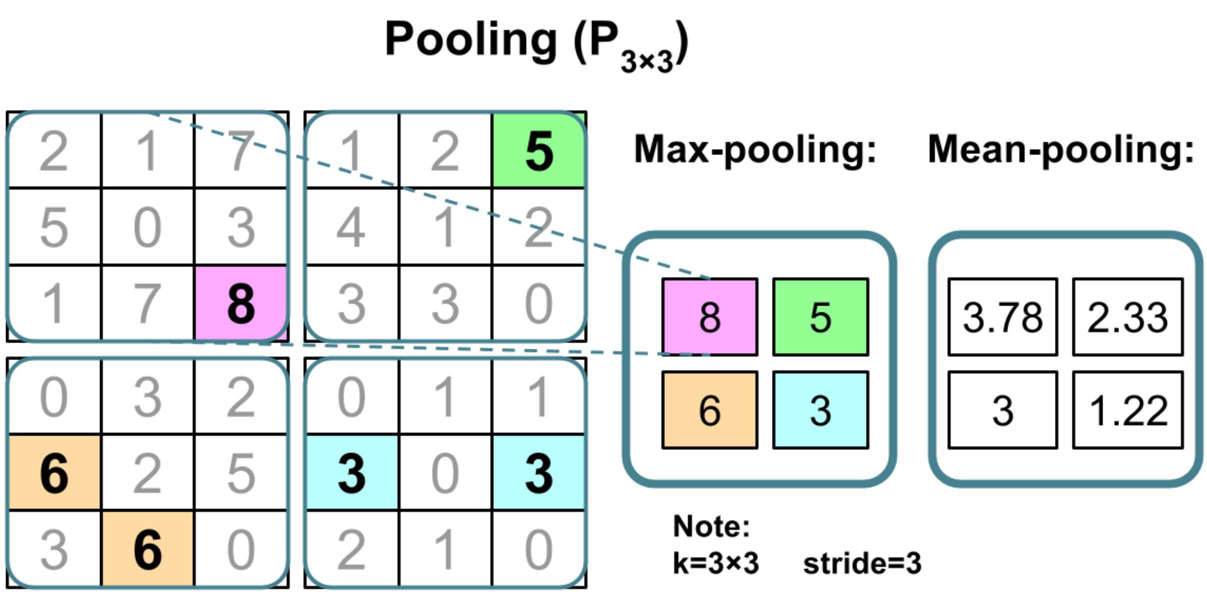
Kernel
(3×3)
Input
(8×8)
Padded zeros
Output
(8×8)



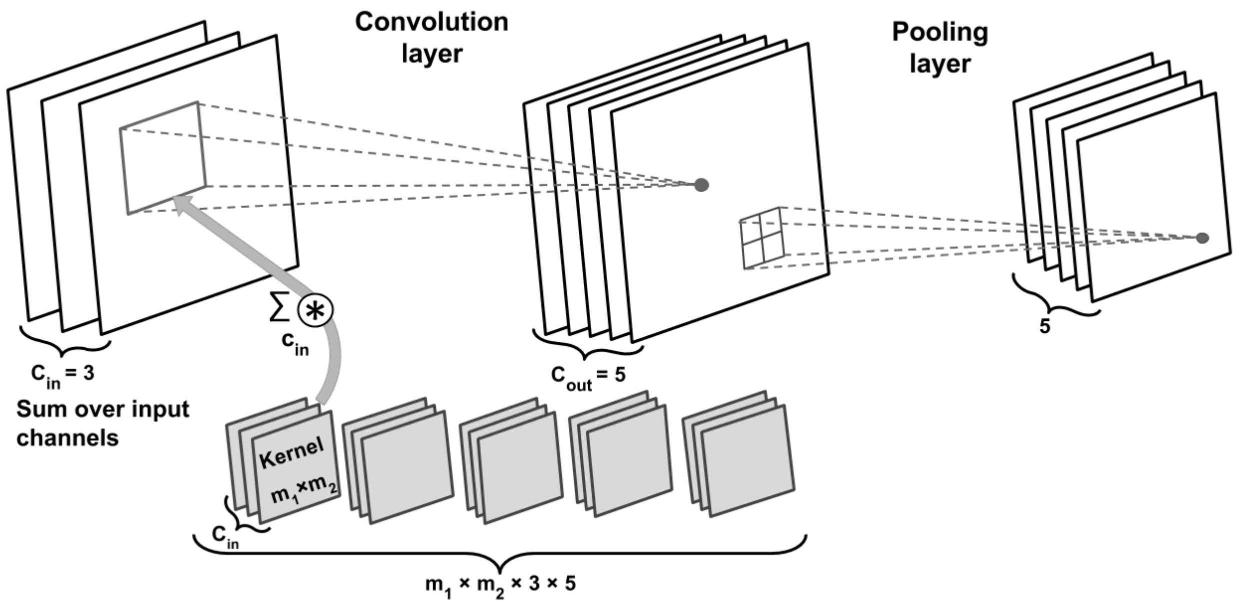
X
W
0 0 0 0 0
0 2 1 2 0
0 5 0 1 0
0 1 7 3 0
0 0 0 0 0
*
0.5 0.7 0.4
0.3 0.4 0.1
0.5 1 0.5

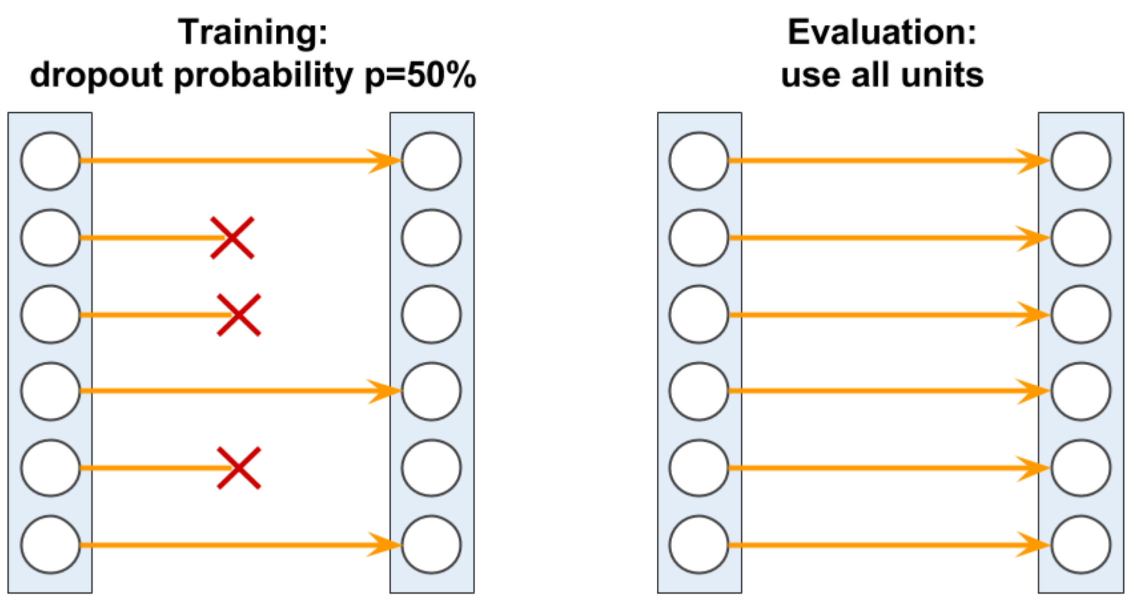


X^padded
0 0 0 0 0
0 2 1 2 0
0 5 0 1 0
0 1 7 3 0
0 0 0 0 0
0.5 1 0.5
0.1 0.4 0.3
0.4 0.7 0.5
Y:
4.6 1.6
7.5 2.9

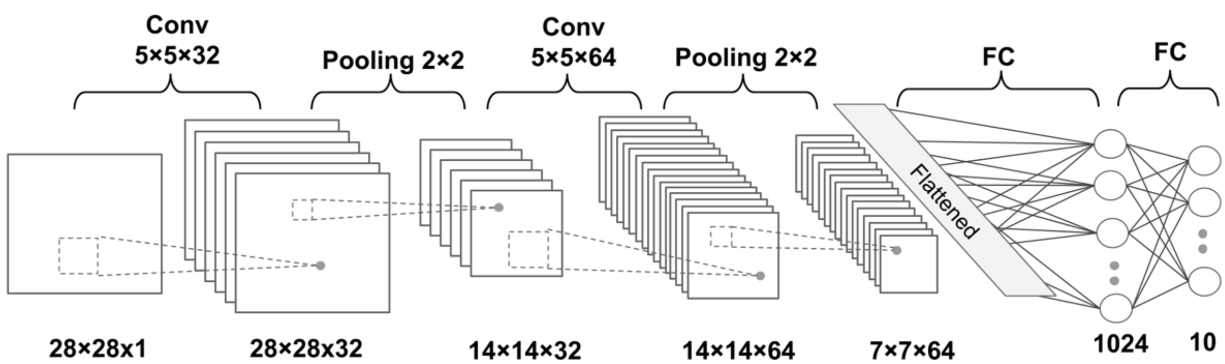


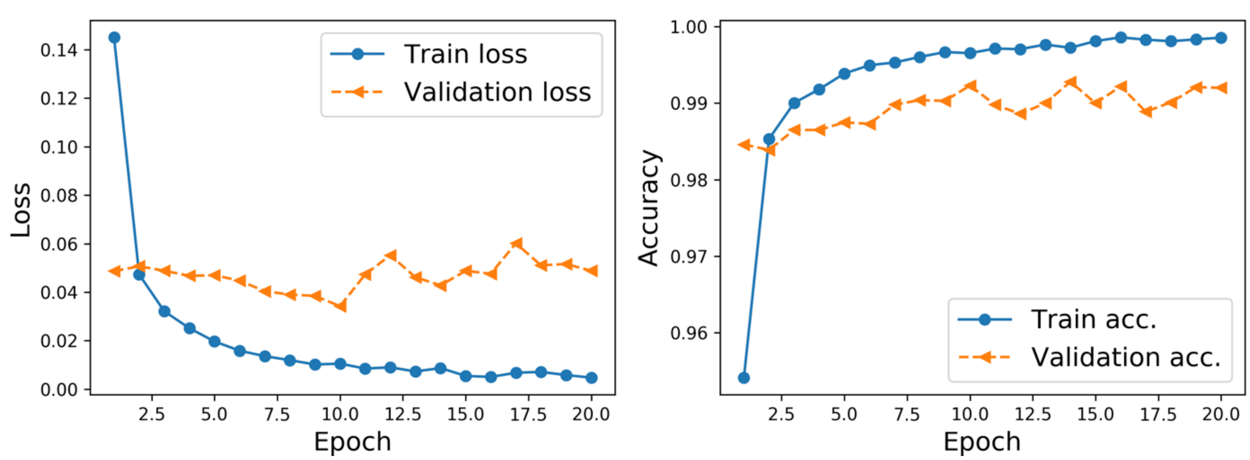
Pooling (P3×3)
2 1 7 1 2 5
5 0 3 4 1 2
1 7 8 3 3 0
0 3 2 0 1 1
6 2 5 3 0 3
3 6 0 2 1 0
Max-pooling:
8 5
6 3
Mean-pooling:
3.78 2.33
3 1.22
Note:
k=3×3 stride=3

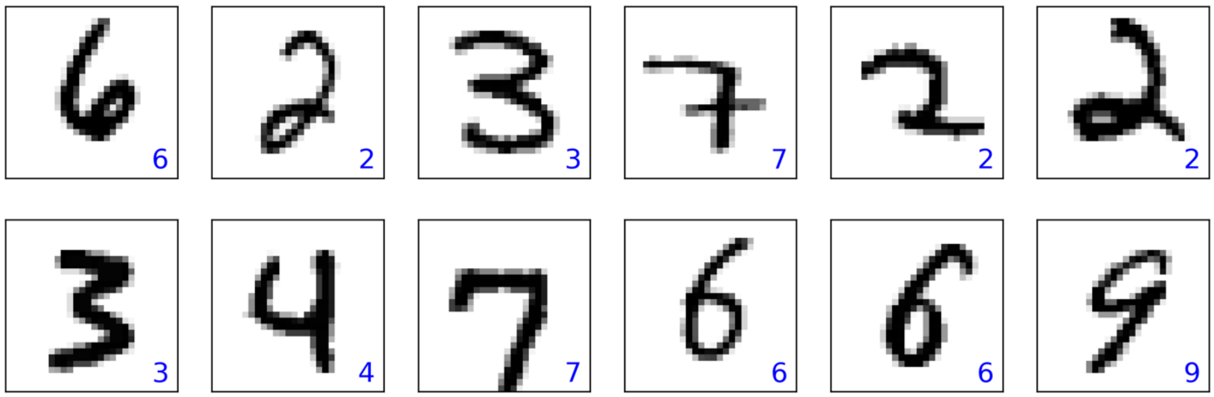

Convolution layer
Pooling layer
C_{in} = 3
Sum over input channels
$\sum$ ⊛ c_{in}
C_{out} = 5
5
Kernel $m_1 \times m_2$
C_{in}
$m_1 \times m_2 \times 3 \times 5$

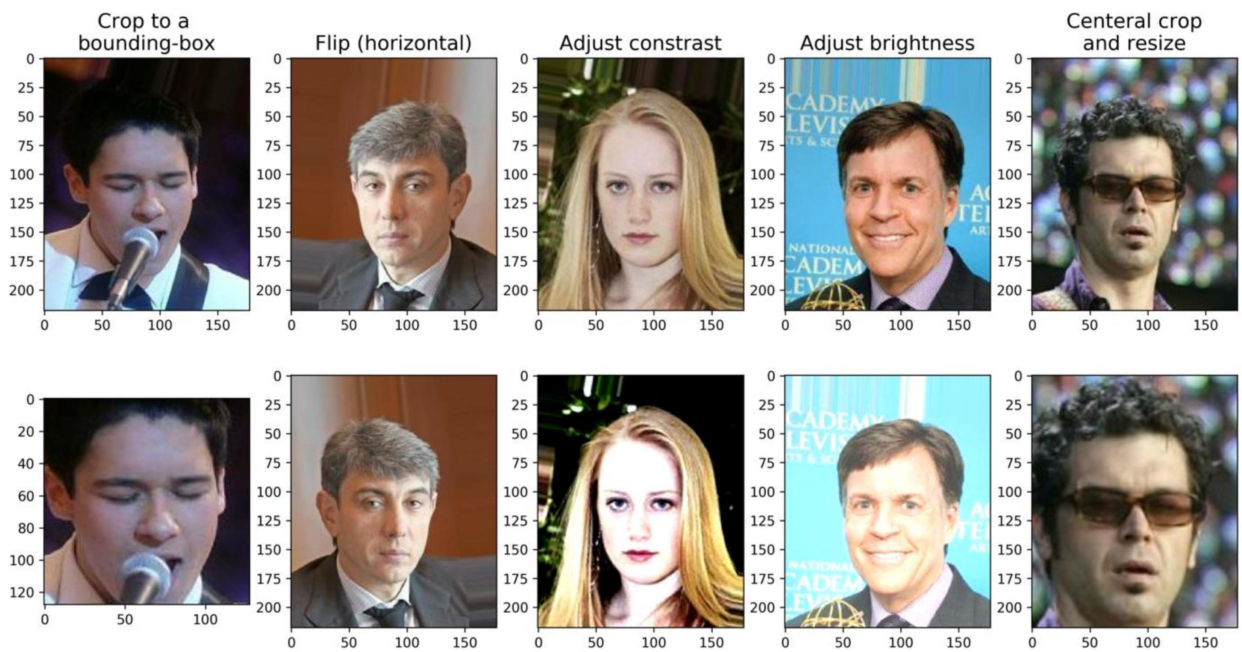

Training:
dropout probability p=50%
Evaluation:
use all units

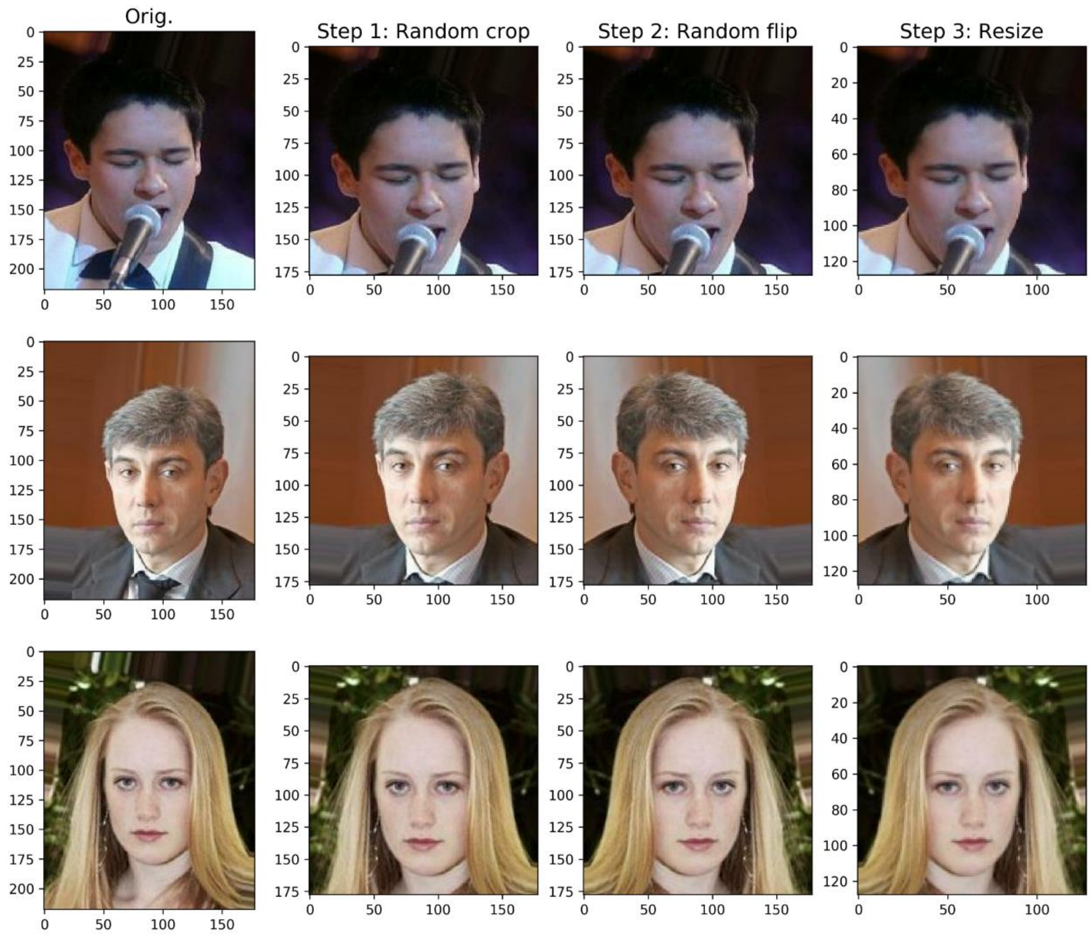
Loss function	Usage	Examples: Using probabilities *from_logits=False*	Examples: Using logits *from_logits=True*
`BinaryCrossentropy`	Binary classification	**y_true:** 1 **y_pred:** 0.69	**y_true:** 1 **y_pred:** 0.8
`CategoricalCrossentropy`	Multiclass classification	**y_true:** 0, 0, 1 **y_pred:** 0.30, 0.15, 0.55	**y_true:** 0, 0, 1 **y_pred:** 1.5, 0.8, 2.1
`Sparse CategoricalCrossentropy`	Multiclass classification	**y_true:** 2 **y_pred:** 0.30, 0.15, 0.55	**y_true:** 2 **y_pred:** 1.5, 0.8, 2.1

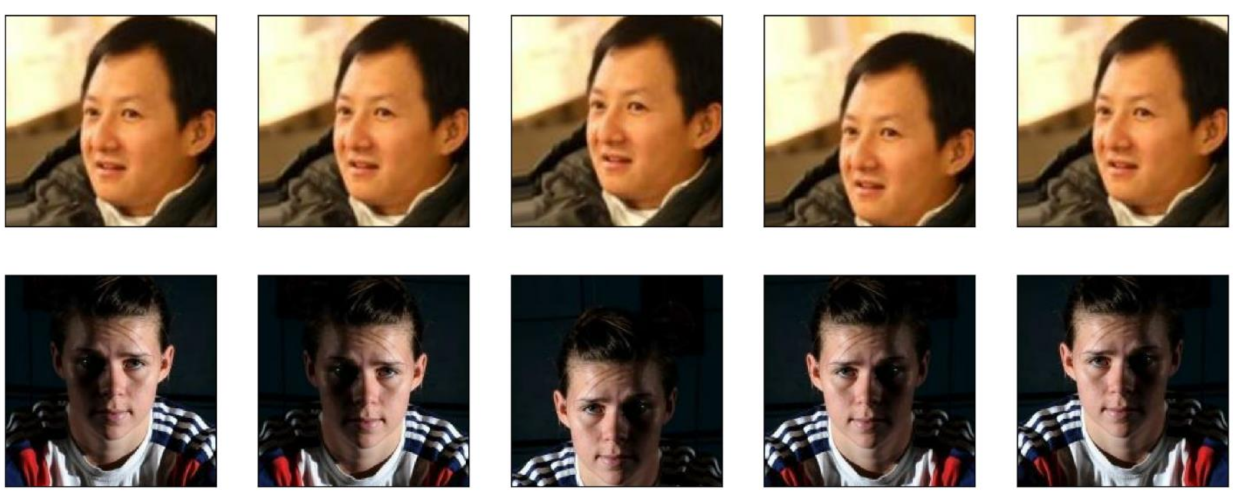


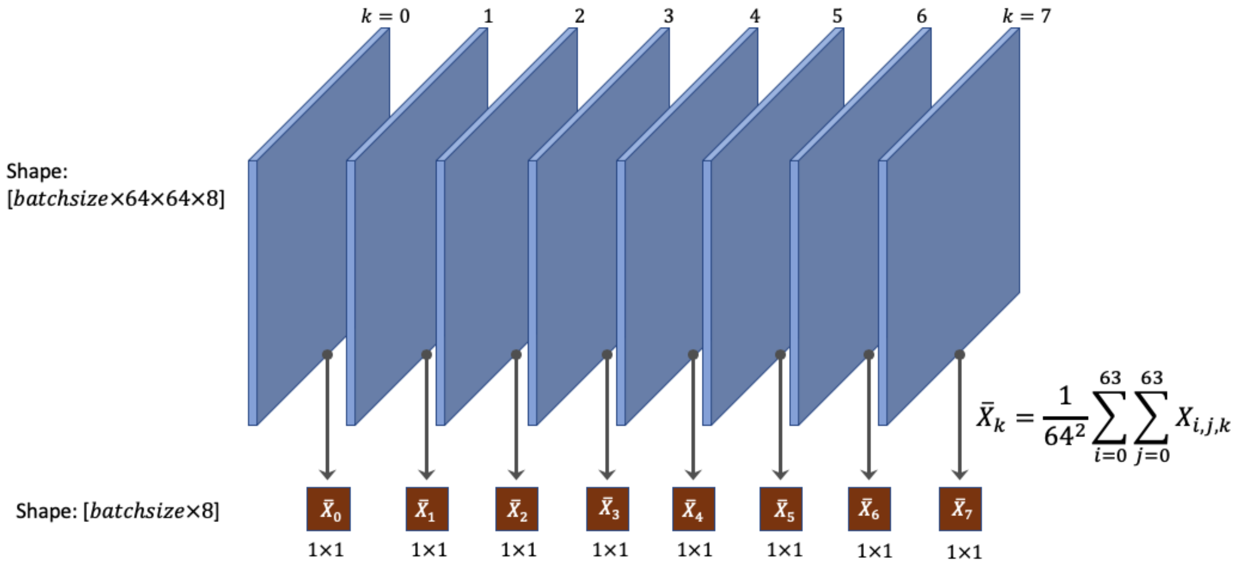



6
2
3
7
2
2
3
4
7
6
6
9

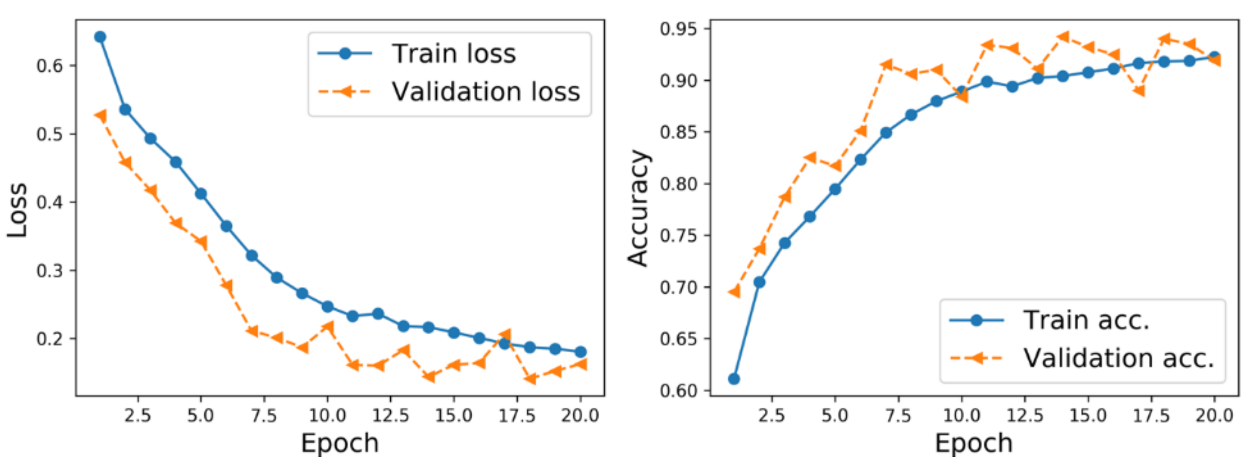

Crop to a bounding-box
Flip (horizontal)
Adjust constrast
Adjust brightness
Centeral crop and resize


Orig.
Step 1: Random crop
Step 2: Random flip
Step 3: Resize





$k = 0$
1
2
3
4
5
6
$k = 7$
Shape: [$batchsize$×64×64×8]
$\bar{X}_k = \frac{1}{64^2}\sum_{i=0}^{63}\sum_{j=0}^{63} X_{i,j,k}$
Shape: [$batchsize$×8]
$\bar{X}_0$ $\bar{X}_1$ $\bar{X}_2$ $\bar{X}_3$ $\bar{X}_4$ $\bar{X}_5$ $\bar{X}_6$ $\bar{X}_7$
1×1 1×1 1×1 1×1 1×1 1×1 1×1 1×1



Train loss
Validation loss
Loss
Epoch
Train acc.
Validation acc.
Accuracy
Epoch


GT: Male
Pr(Male)=80%


GT: Female
Pr(Male)=0%


GT: Male
Pr(Male)=100%


GT: Female
Pr(Male)=89%


GT: Male
Pr(Male)=89%


GT: Male
Pr(Male)=99%

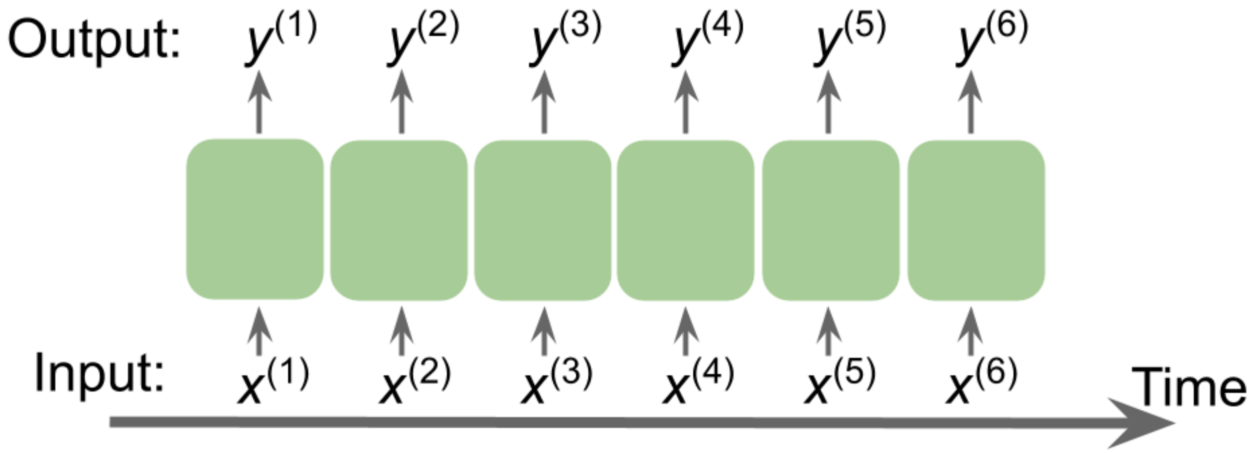

GT: Female
Pr(Male)=0%

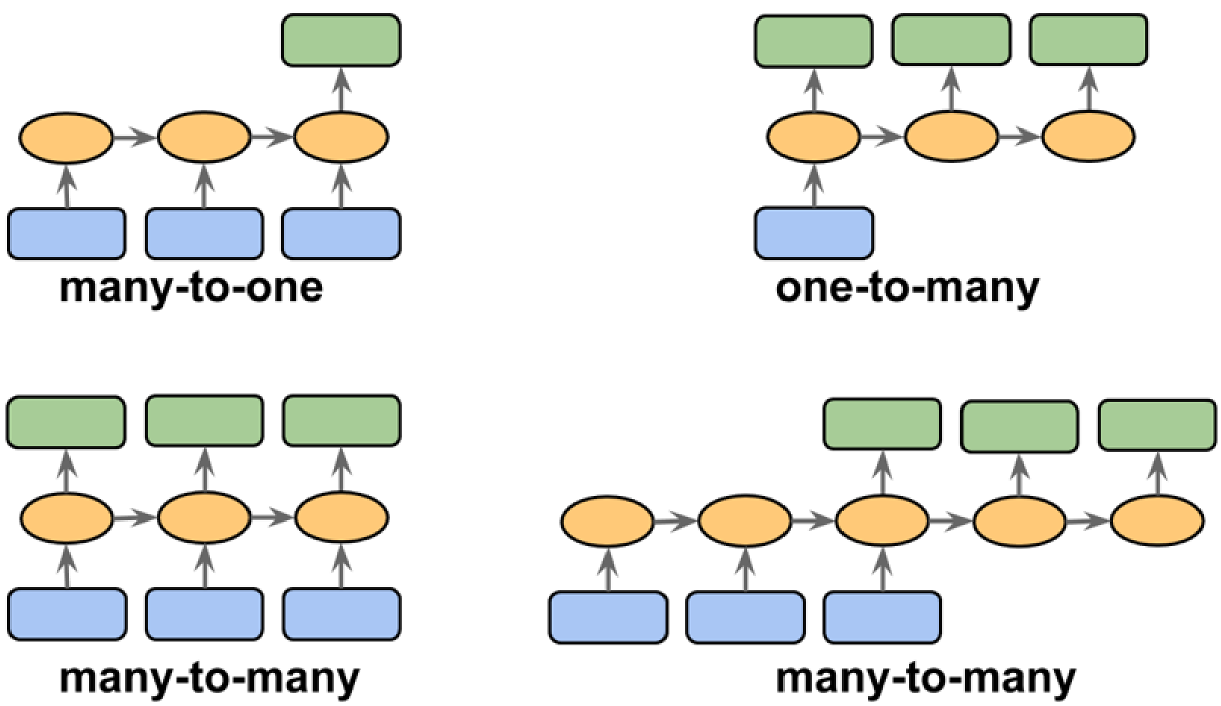

GT: Female
Pr(Male)=0%

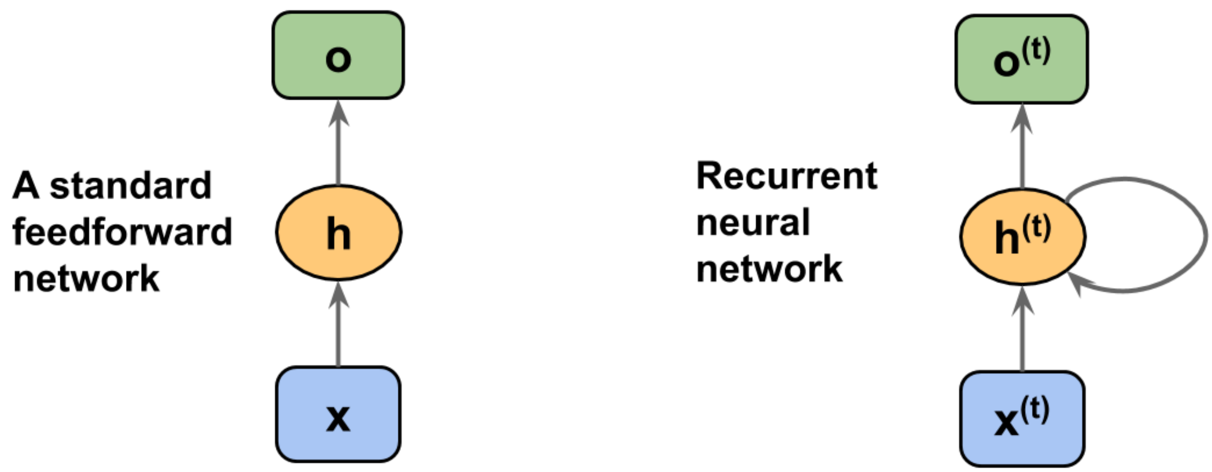

GT: Male
Pr(Male)=99%

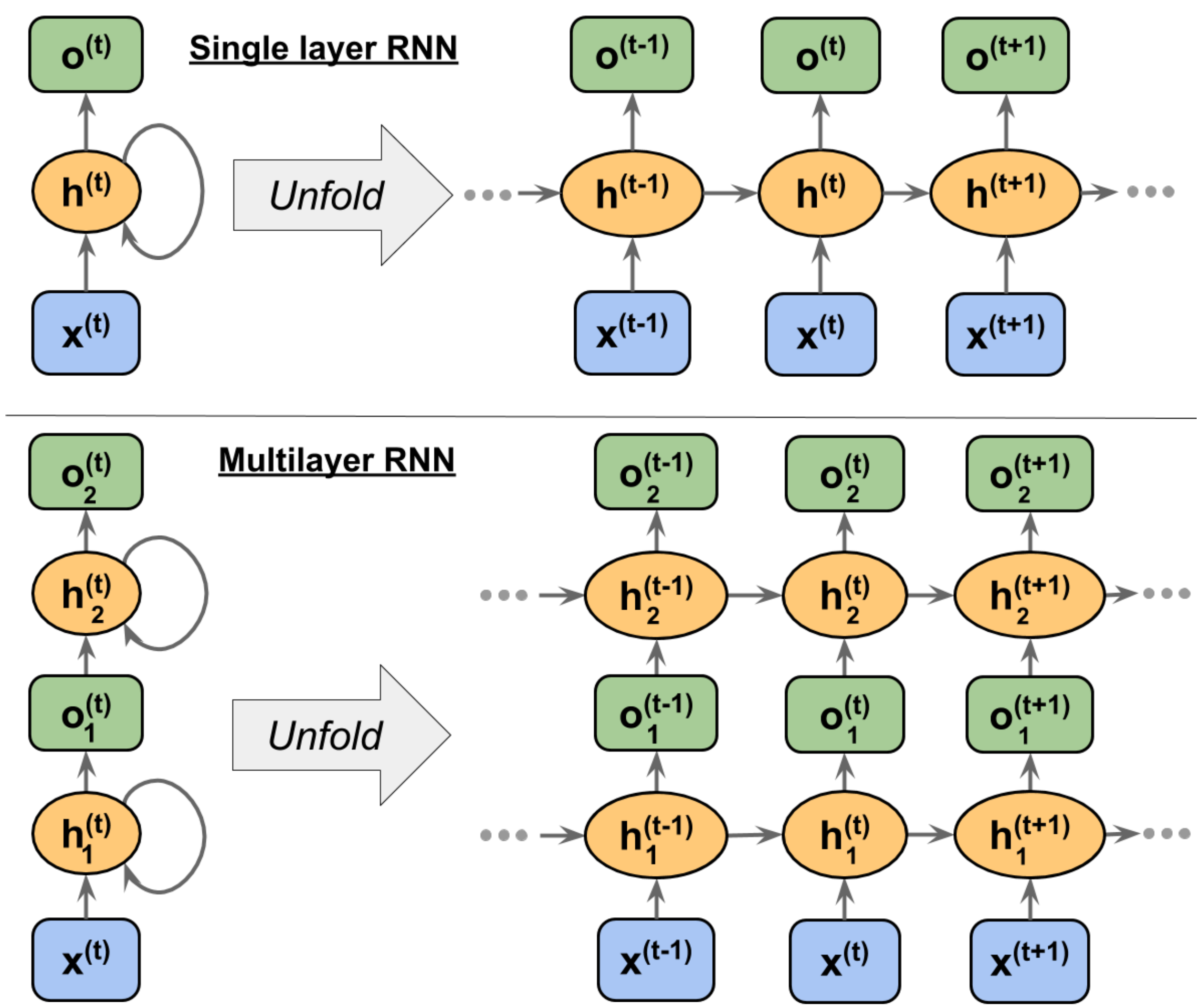

GT: Male
Pr(Male)=100%

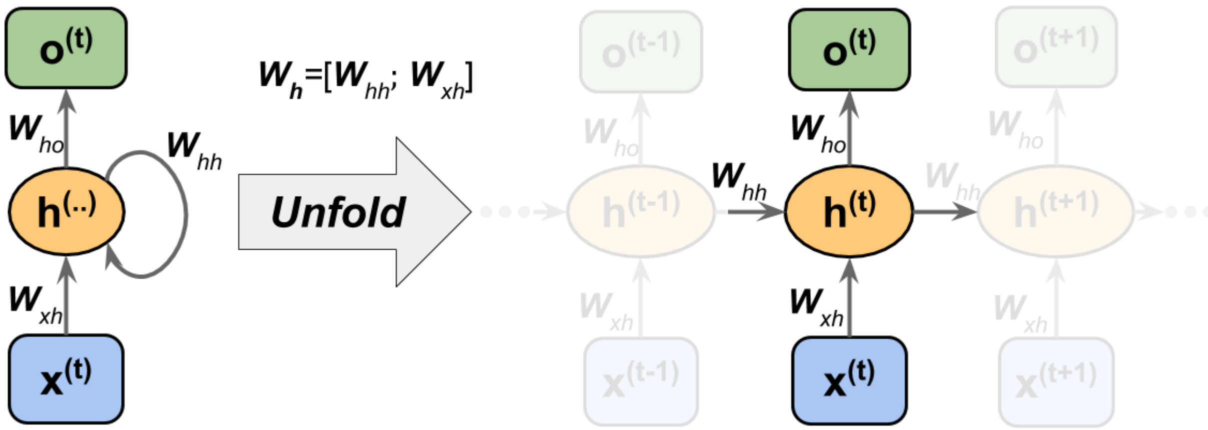
Chapter 16: Modeling Sequential Data Using Recurrent Neural Networks



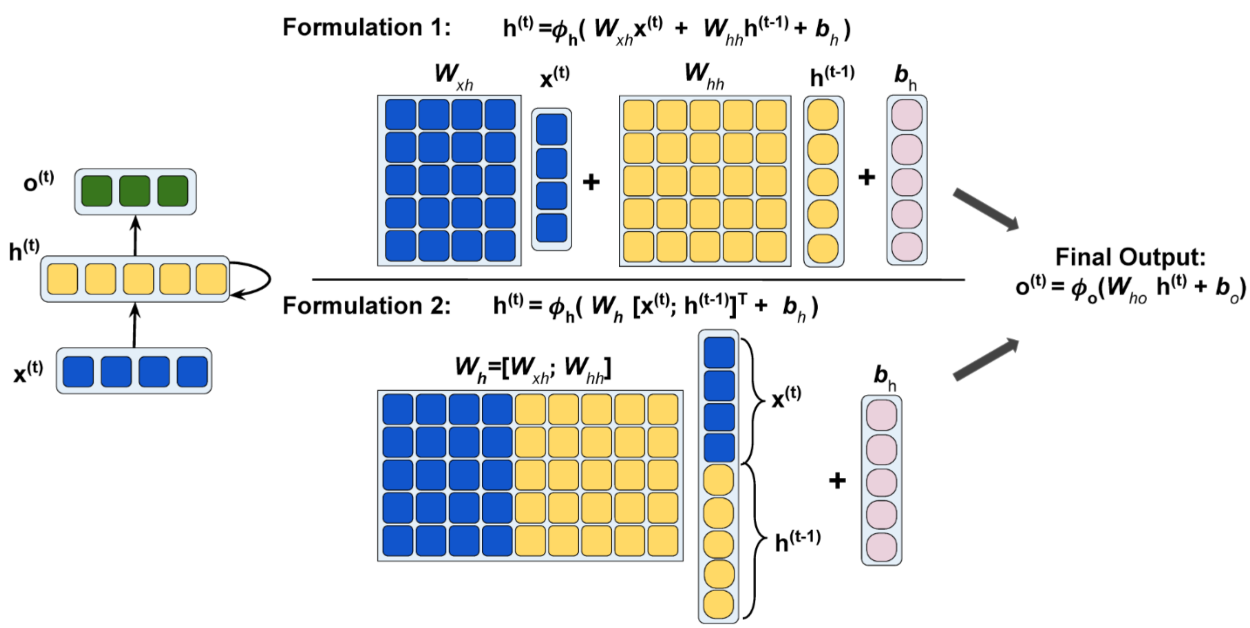



o
A standard feedforward network
h
x
$o^{(t)}$
Recurrent neural network
$h^{(t)}$
$x^{(t)}$

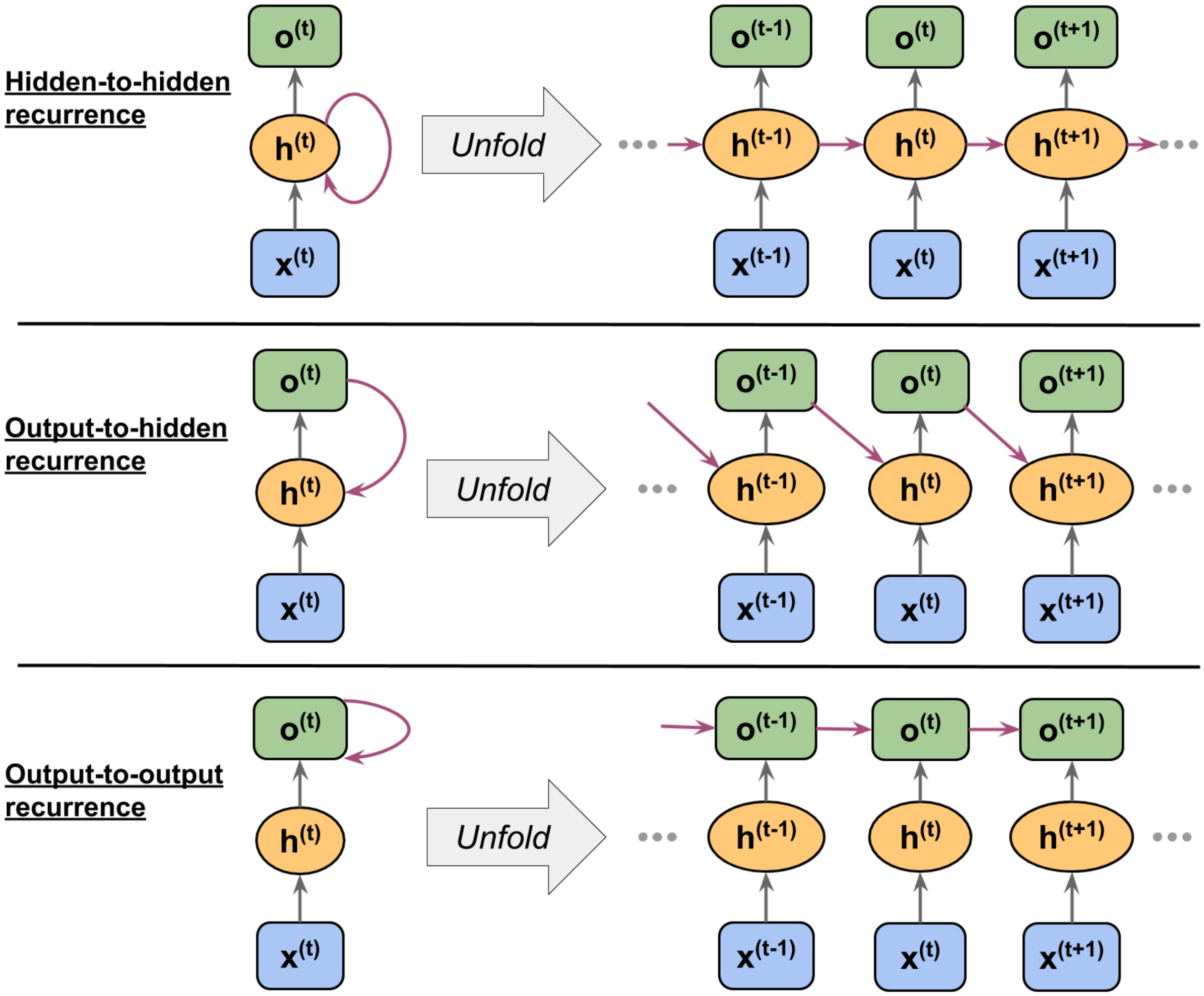

Single layer RNN
$o^{(t)}$
$h^{(t)}$
$x^{(t)}$
Unfold
$o^{(t-1)}$
$o^{(t)}$
$o^{(t+1)}$
$h^{(t-1)}$
$h^{(t)}$
$h^{(t+1)}$
$x^{(t-1)}$
$x^{(t)}$
$x^{(t+1)}$
Multilayer RNN
$o_2^{(t)}$
$h_2^{(t)}$
$o_1^{(t)}$
$h_1^{(t)}$
$x^{(t)}$
Unfold
$o_2^{(t-1)}$
$o_2^{(t)}$
$o_2^{(t+1)}$
$h_2^{(t-1)}$
$h_2^{(t)}$
$h_2^{(t+1)}$
$o_1^{(t-1)}$
$o_1^{(t)}$
$o_1^{(t+1)}$
$h_1^{(t-1)}$
$h_1^{(t)}$
$h_1^{(t+1)}$
$x^{(t-1)}$
$x^{(t)}$
$x^{(t+1)}$

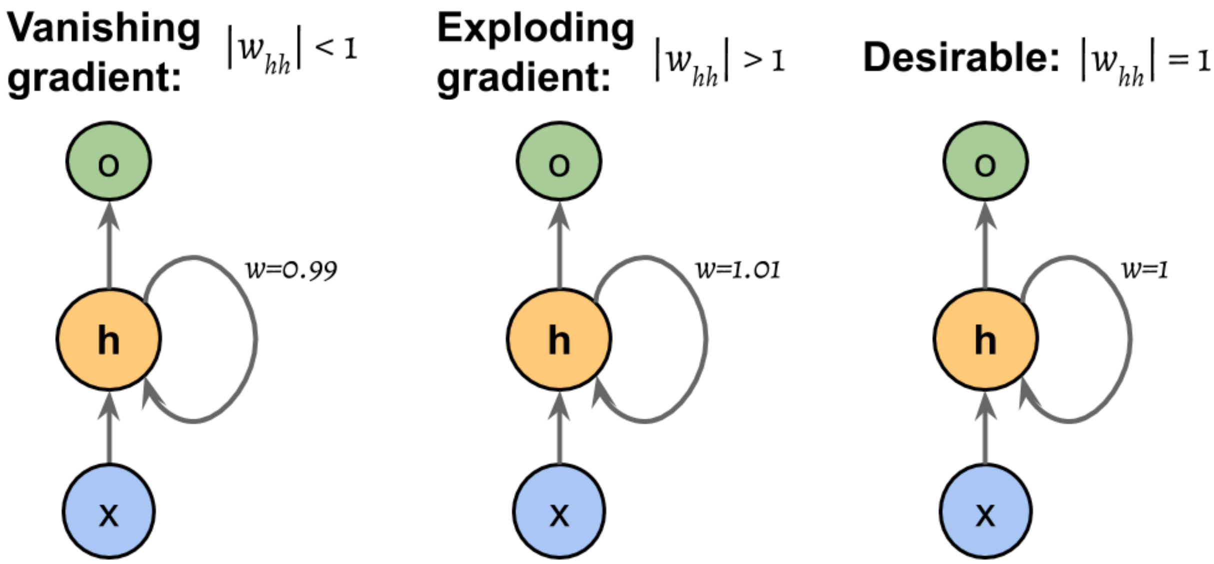


o(t)
Who
h(..)
Whh
Wxh
x(t)
Wh=[Whh; Wxh]
Unfold
o(t-1)
h(t-1)
x(t-1)
h(t)
o(t+1)
h(t+1)
x(t+1)

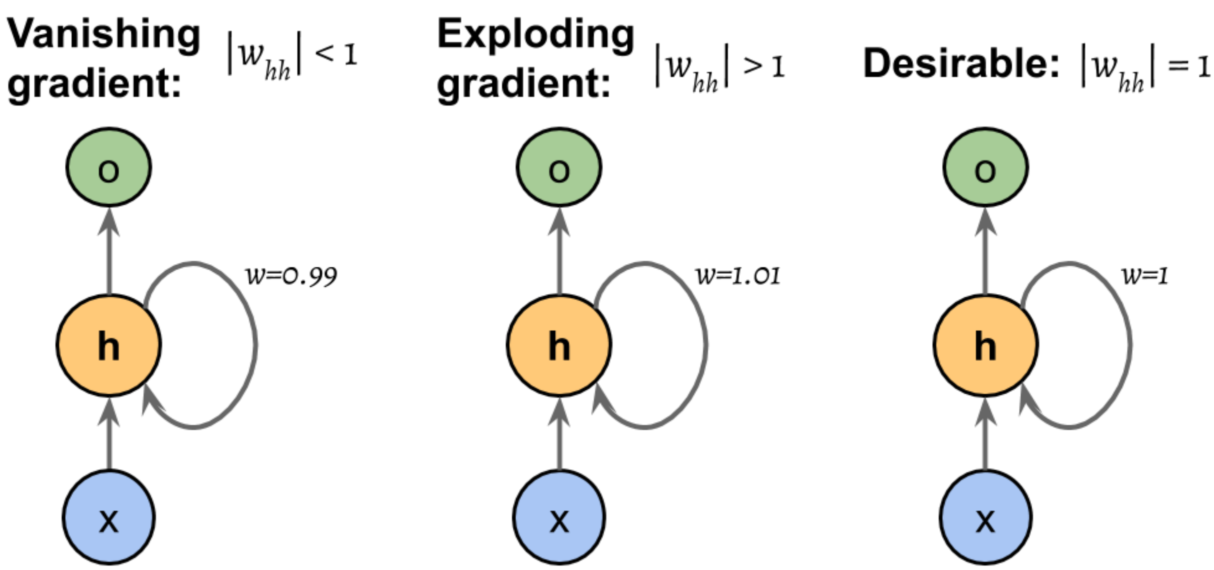


o(t)
h(t)
x(t)
Formulation 1: h(t) =φh(Wxhx(t) + Whhh(t-1) + bh)
Wxh
x(t)
Whh
h(t-1)
bh
+
Formulation 2: h(t) = φh(Wh [x(t); h(t-1)]T + bh)
Wh=[Wxh; Whh]
x(t)
h(t-1)
bh
+
Final Output:
o(t) = φo(Who h(t) + bo)

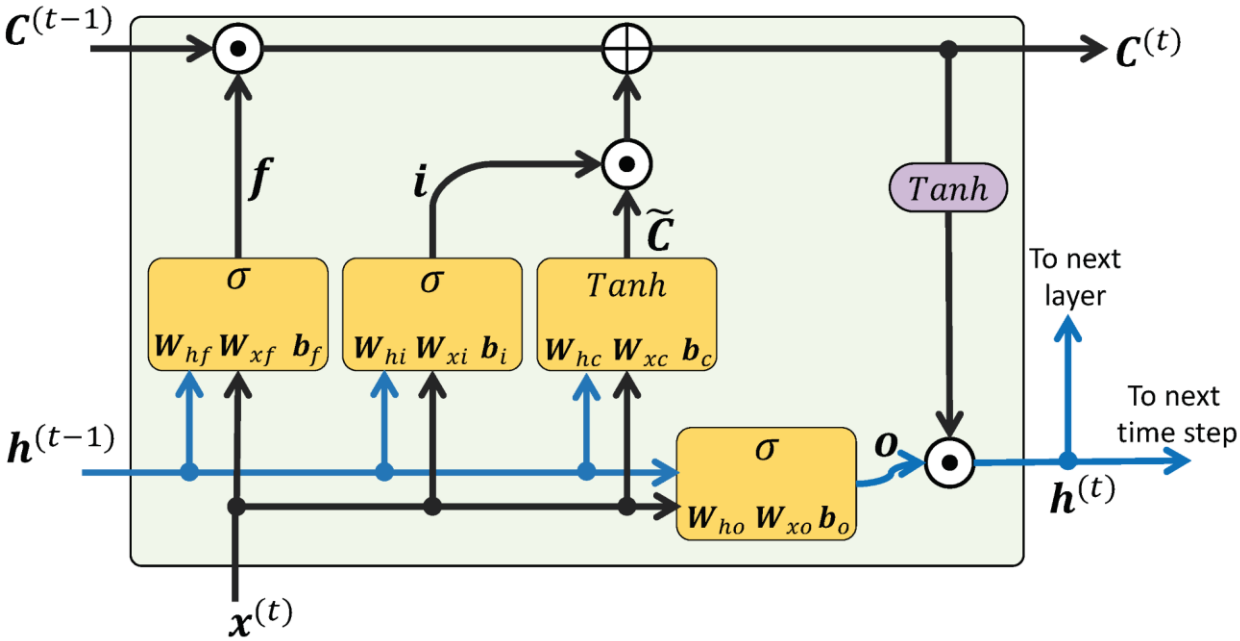

Hidden-to-hidden recurrence
o(t)
h(t)
x(t)
Unfold
o(t-1)
o(t)
o(t+1)
h(t-1)
h(t)
h(t+1)
x(t-1)
x(t)
x(t+1)
Output-to-hidden recurrence
o(t)
h(t)
x(t)
Unfold
o(t-1)
o(t)
o(t+1)
h(t-1)
h(t)
h(t+1)
x(t-1)
x(t)
x(t+1)
Output-to-output recurrence
o(t)
h(t)
x(t)
Unfold
o(t-1)
o(t)
o(t+1)
h(t-1)
h(t)
h(t+1)
x(t-1)
x(t)
x(t+1)

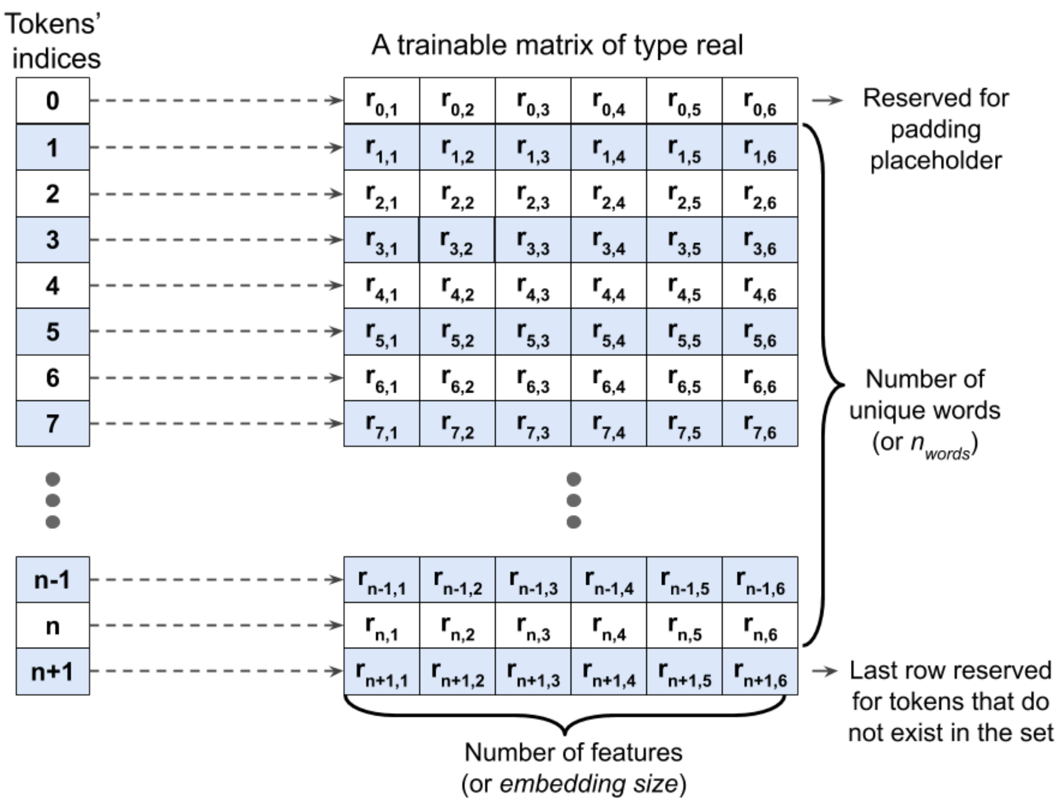


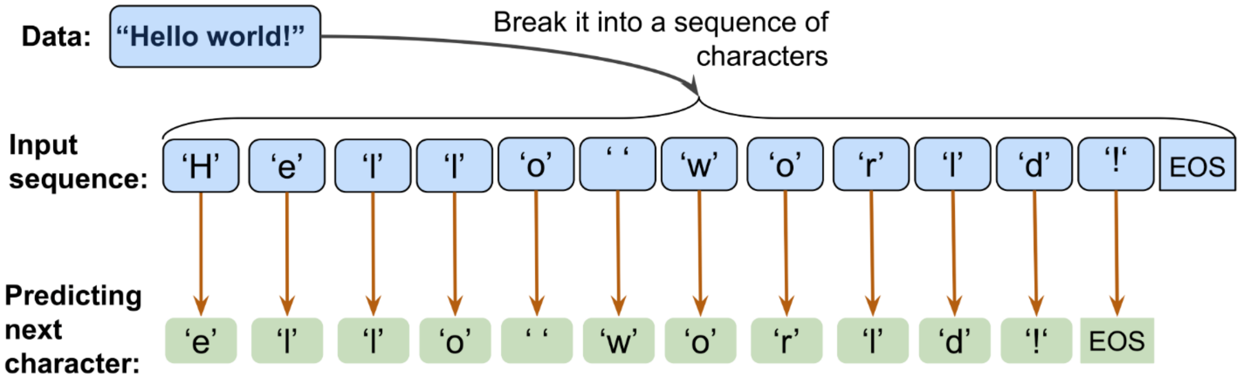
Vanishing gradient: $|w_{hh}| < 1$
Exploding gradient: $|w_{hh}| > 1$
Desirable: $|w_{hh}| = 1$
o
h
x
w=0.99
w=1.01
w=1

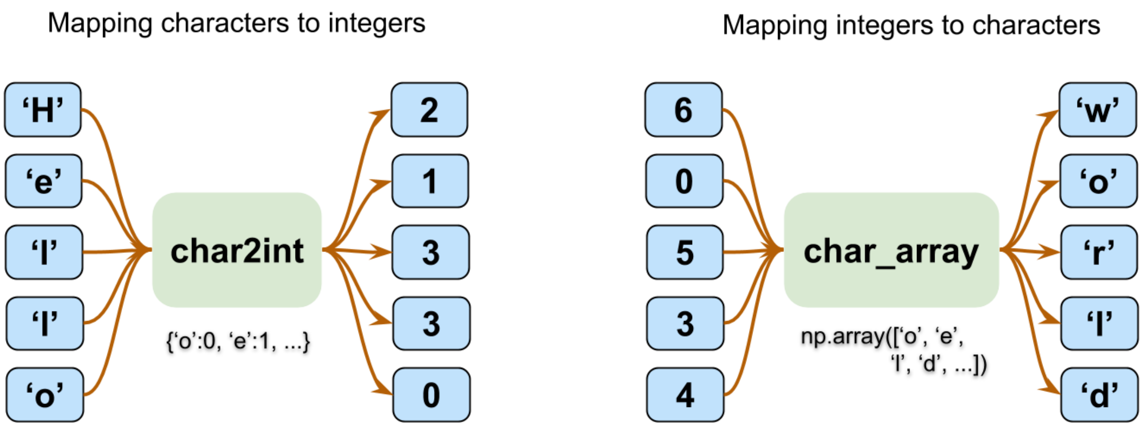


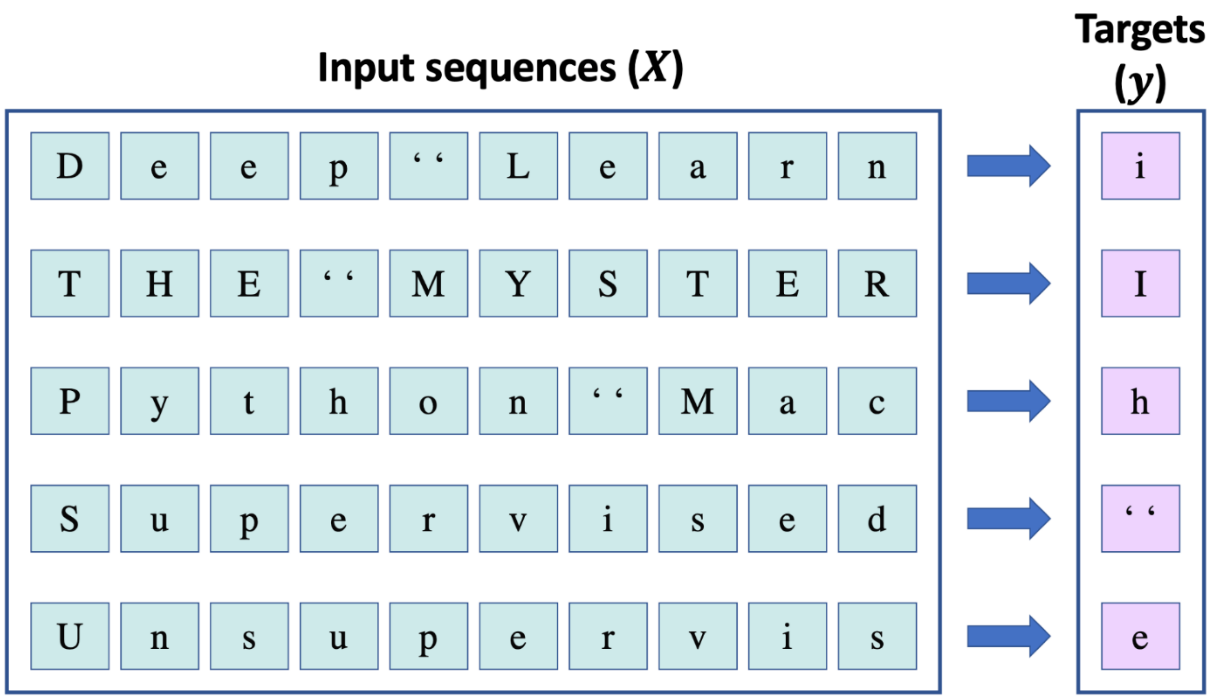
Vanishing gradient: $|w_{hh}| < 1$
Exploding gradient: $|w_{hh}| > 1$
Desirable: $|w_{hh}| = 1$
o
h
x
w=0.99
w=1.01
w=1

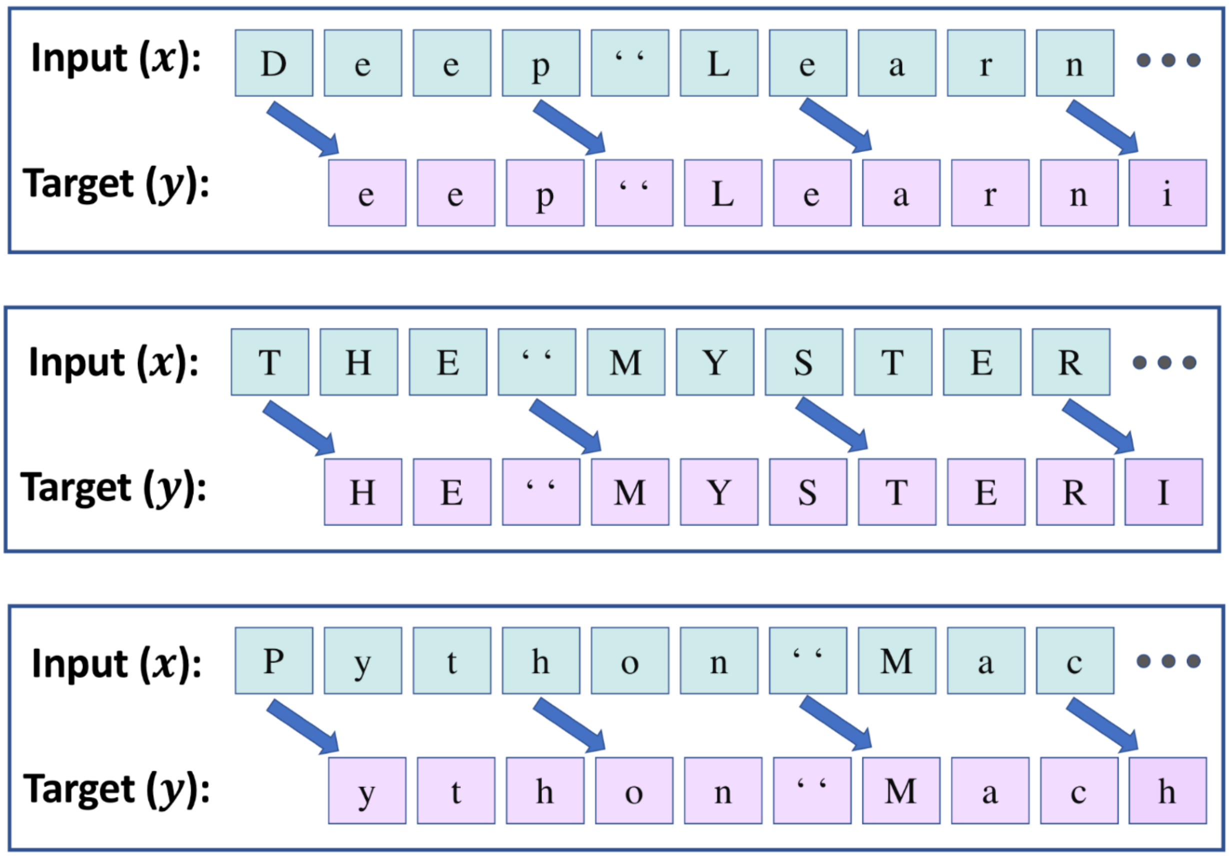

$C^{(t-1)}$
$C^{(t)}$
f
i
$\tilde{C}$
$Tanh$
σ
W_{hf} W_{xf} b_f
W_{hi} W_{xi} b_i
W_{hc} W_{xc} b_c
W_{ho} W_{xo} b_o
o
To next layer
To next time step
$h^{(t-1)}$
$h^{(t)}$
$x^{(t)}$

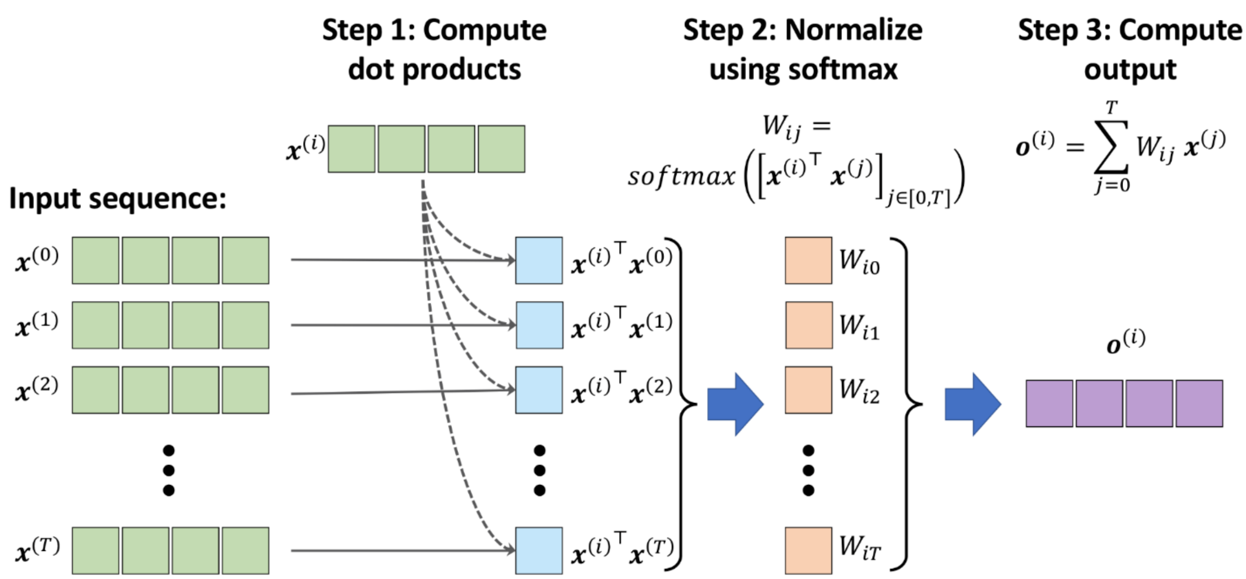

Tokens' indices
A trainable matrix of type real
0
1
2
3
4
5
6
7
n-1
n
n+1
Reserved for padding placeholder
Number of unique words (or n_{words})
Last row reserved for tokens that do not exist in the set
Number of features (or *embedding size*)

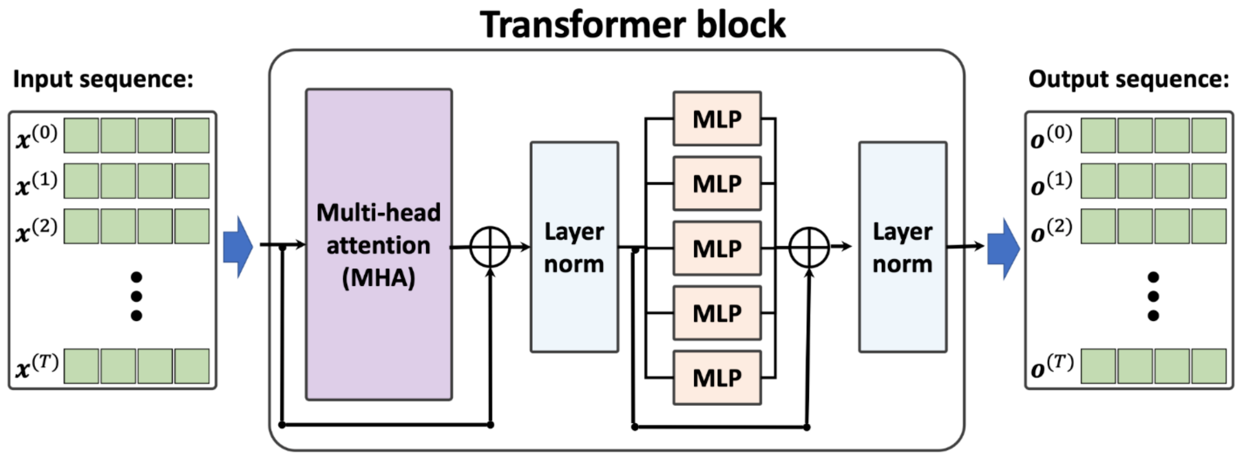

Data:
“Hello world!”
Break it into a sequence of characters
Input sequence:
‘H’ ‘e’ ‘l’ ‘l’ ‘o’ ‘ ‘ ‘w’ ‘o’ ‘r’ ‘l’ ‘d’ ‘!‘ EOS
Predicting next character:
‘e’ ‘l’ ‘l’ ‘o’ ‘ ‘ ‘w’ ‘o’ ‘r’ ‘l’ ‘d’ ‘!‘ EOS


Mapping characters to integers
‘H’ ‘e’ ‘l’ ‘l’ ‘o’
char2int
{‘o’:0, ‘e’:1, ...}
2 1 3 3 0
Mapping integers to characters
6 0 5 3 4
char_array
np.array([‘o’, ‘e’, ‘l’, ‘d’, ...])
‘w’ ‘o’ ‘r’ ‘l’ ‘d’

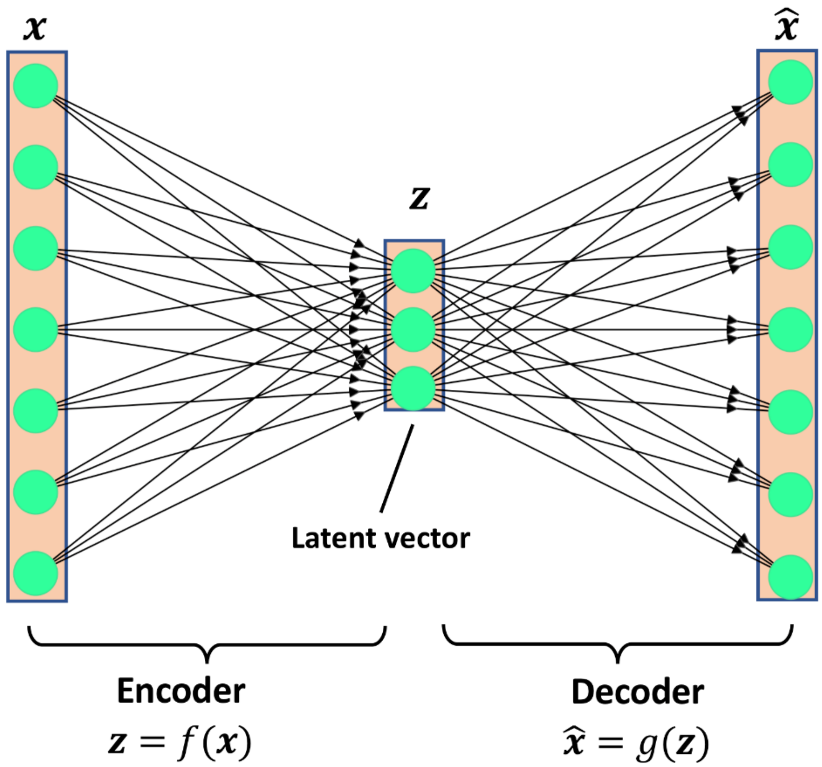

Input sequences (X)
Targets (y)
D e e p ‘ ‘ L e a r n
i
T H E ‘ ‘ M Y S T E R
I
P y t h o n ‘ ‘ M a c
h
S u p e r v i s e d
‘ ‘
U n s u p e r v i s
e

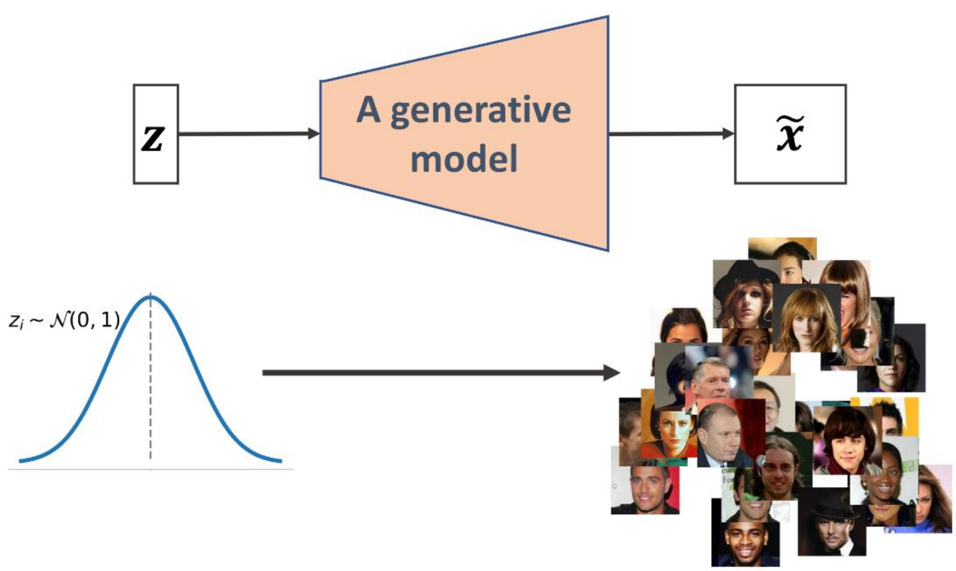

Input (x):
D e e p ‘ ‘ L e a r n • • •
Target (y):
e e p ‘ ‘ L e a r n i
Input (x):
T H E ‘ ‘ M Y S T E R • • •
Target (y):
H E ‘ ‘ M Y S T E R I
Input (x):
P y t h o n ‘ ‘ M a c • • •
Target (y):
y t h o n ‘ ‘ M a c h

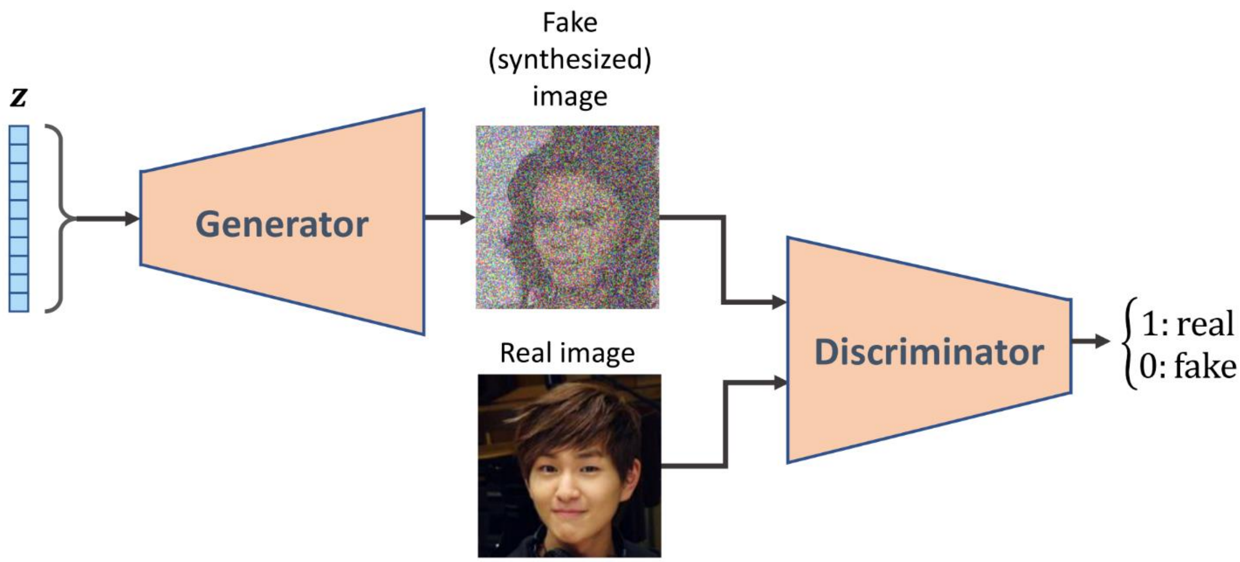

Step 1: Compute dot products
Step 2: Normalize using softmax
Step 3: Compute output
Input sequence:
$x^{(i)}$
$x^{(0)}$
$x^{(1)}$
$x^{(2)}$
$x^{(T)}$
$x^{(i)\top} x^{(0)}$
$x^{(i)\top} x^{(1)}$
$x^{(i)\top} x^{(2)}$
$x^{(i)\top} x^{(T)}$
$W_{ij} = softmax\left(\left[x^{(i)\top} x^{(j)}\right]_{j\in[0,T]}\right)$
W_{i0}
W_{i1}
W_{i2}
W_{iT}
$o^{(i)} = \sum_{j=0}^{T} W_{ij}\, x^{(j)}$
$o^{(i)}$

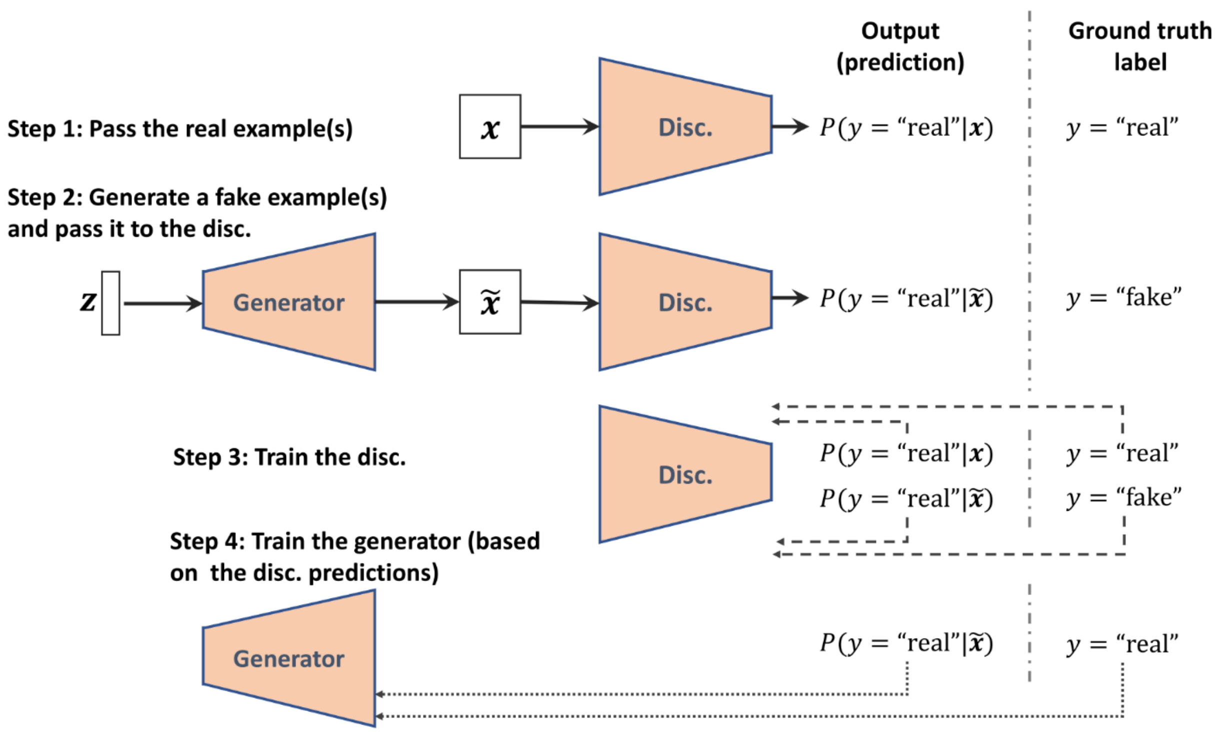

Transformer block
Input sequence:
$x^{(0)}$
$x^{(1)}$
$x^{(2)}$
$x^{(T)}$
Multi-head attention (MHA)
Layer norm
MLP
MLP
MLP
MLP
MLP
Layer norm
Output sequence:
$o^{(0)}$
$o^{(1)}$
$o^{(2)}$
$o^{(T)}$

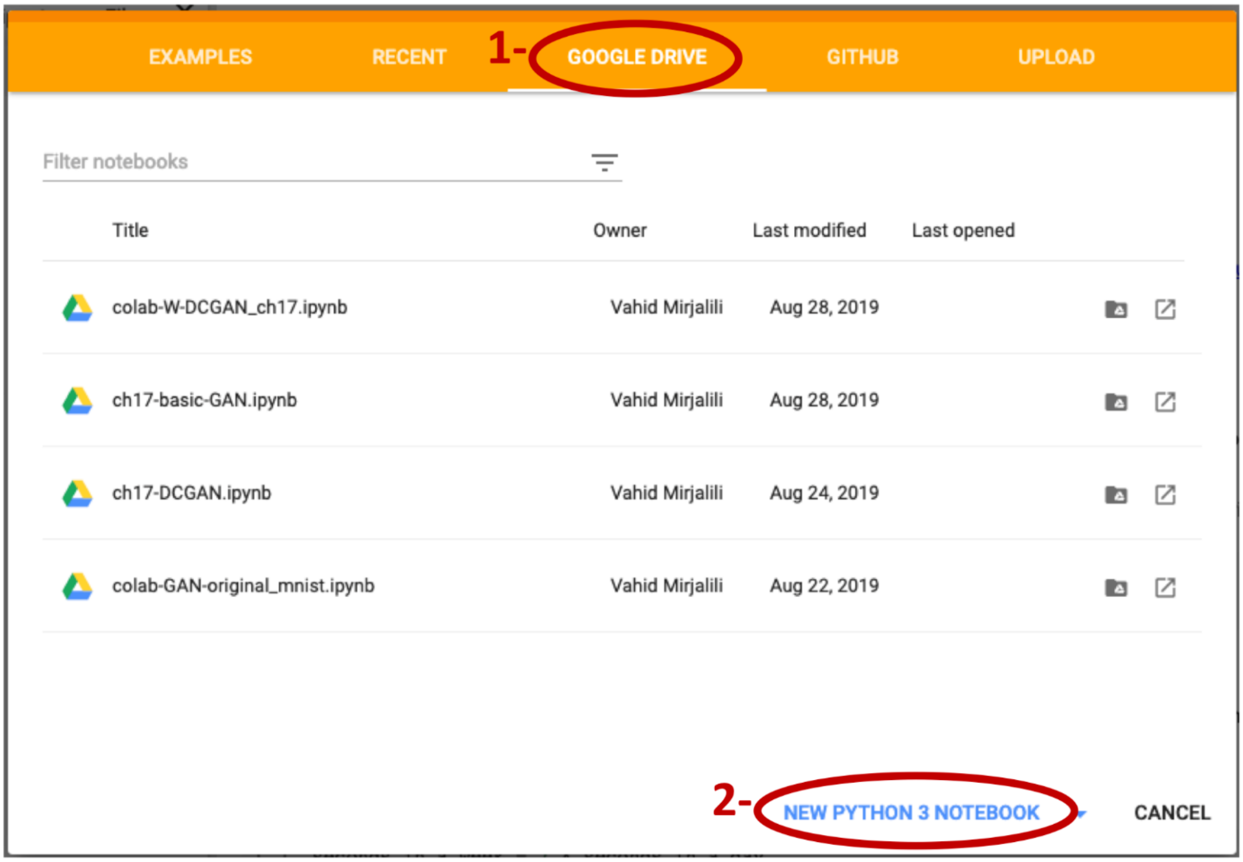
Chapter 17: Generative Adversarial Networks for Synthesizing New Data

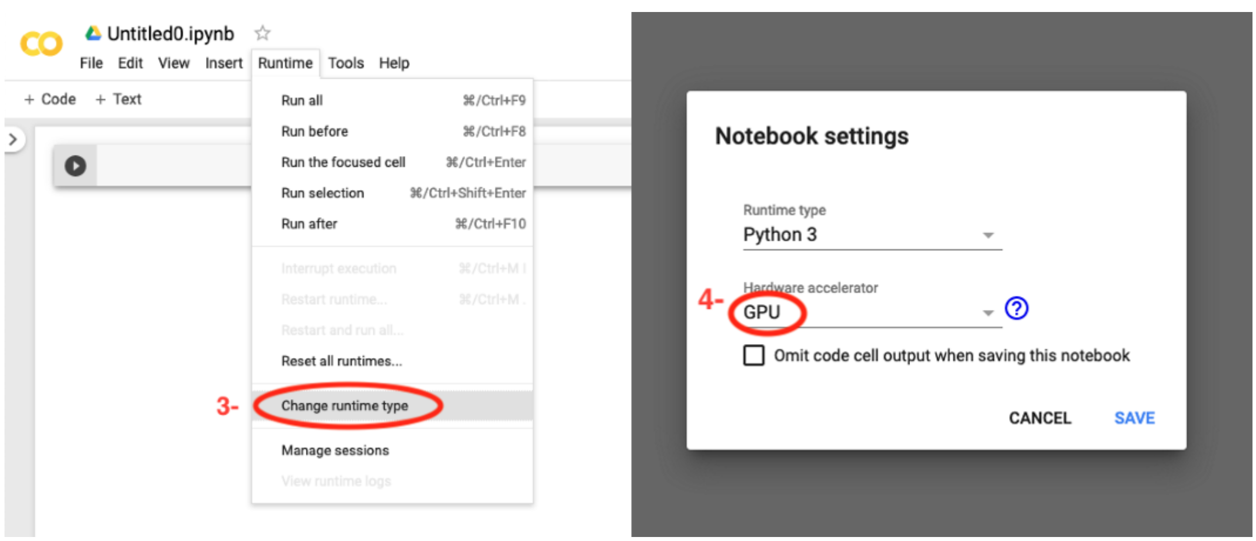


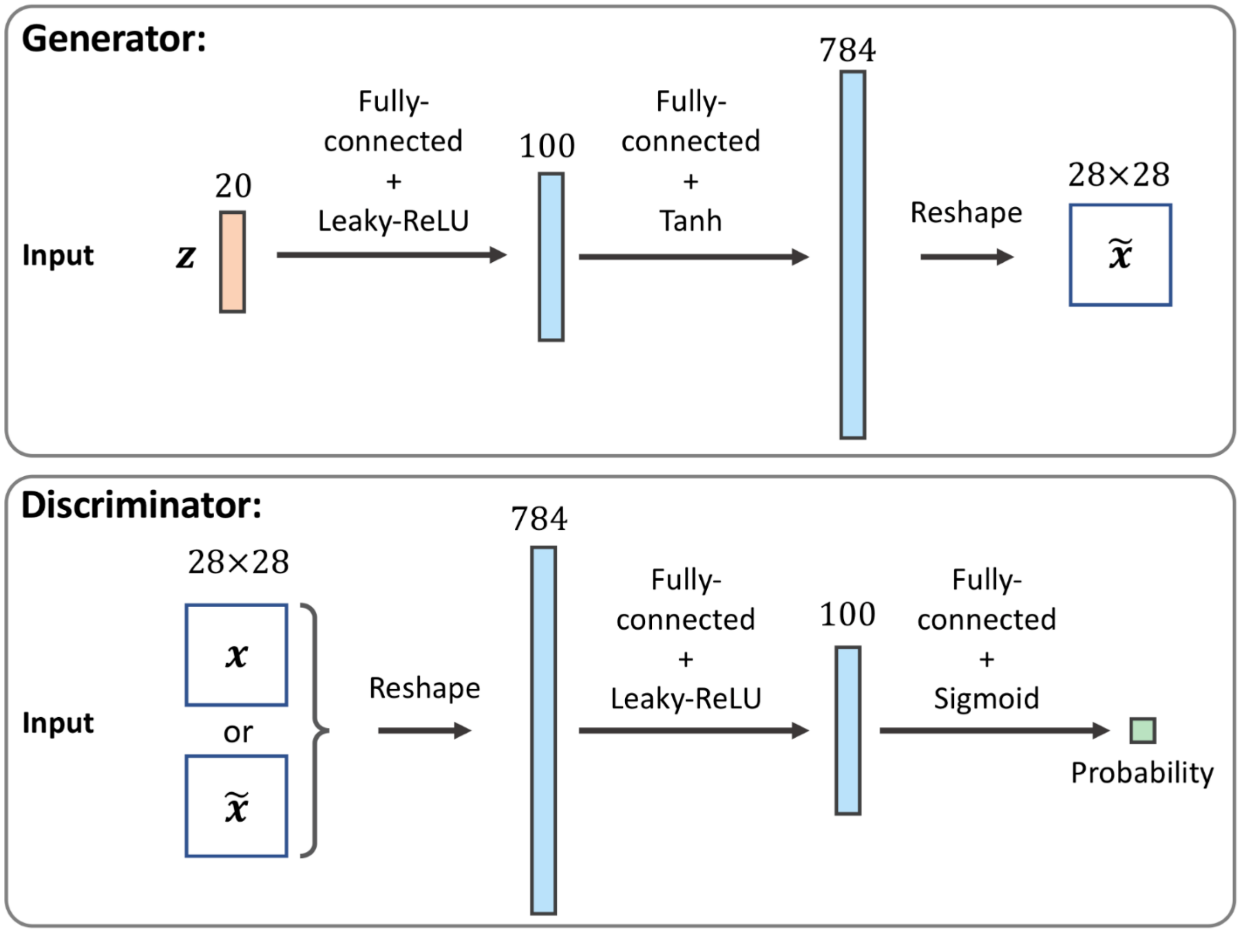



z
Generator
Fake
(synthesized)
image
Real image
Discriminator
1: real
0: fake

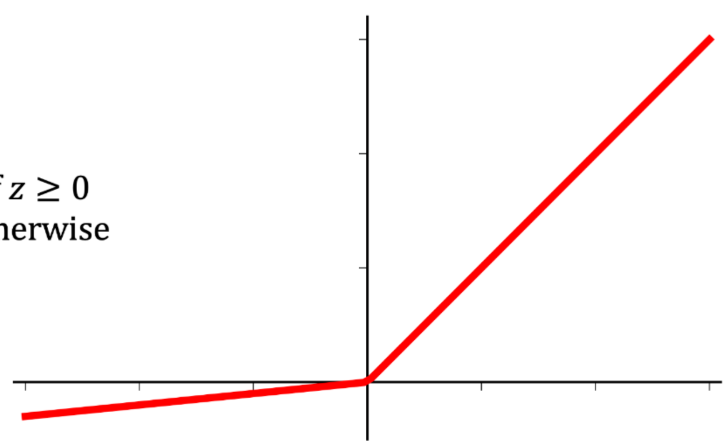

Output
(prediction)
Ground truth
label
Step 1: Pass the real example(s)
x
Disc.
P(y = "real"|x)
y = "real"
Step 2: Generate a fake example(s)
and pass it to the disc.
z
Generator
x̃
Disc.
P(y = "real"|x̃)
y = "fake"
Step 3: Train the disc.
Disc.
P(y = "real"|x)
y = "real"
P(y = "real"|x̃)
y = "fake"
Step 4: Train the generator (based
on the disc. predictions)
Generator
P(y = "real"|x̃)
y = "real"

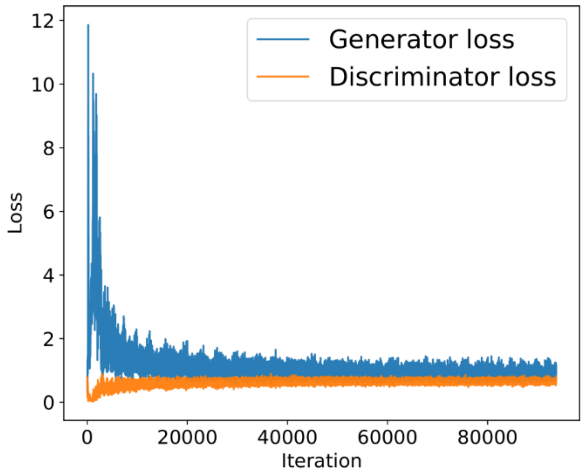

EXAMPLES
RECENT
1-
GOOGLE DRIVE
GITHUB
UPLOAD
Filter notebooks
Title
Owner
Last modified
Last opened
colab-W-DCGAN_ch17.ipynb
Vahid Mirjalili
Aug 28, 2019
ch17-basic-GAN.ipynb
Vahid Mirjalili
Aug 28, 2019
ch17-DCGAN.ipynb
Vahid Mirjalili
Aug 24, 2019
colab-GAN-original_mnist.ipynb
Vahid Mirjalili
Aug 22, 2019
2-
NEW PYTHON 3 NOTEBOOK
CANCEL


Untitled0.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
+ Code
+ Text
Run all
⌘/Ctrl+F9
Run before
⌘/Ctrl+F8
Run the focused cell
⌘/Ctrl+Enter
Run selection
⌘/Ctrl+Shift+Enter
Run after
⌘/Ctrl+F10
Interrupt execution
Restart runtime...
Restart and run all...
Reset all runtimes...
3-
Change runtime type
Manage sessions
View runtime logs
Notebook settings
Runtime type
Python 3
Hardware accelerator
4-
GPU
Omit code cell output when saving this notebook
CANCEL
SAVE

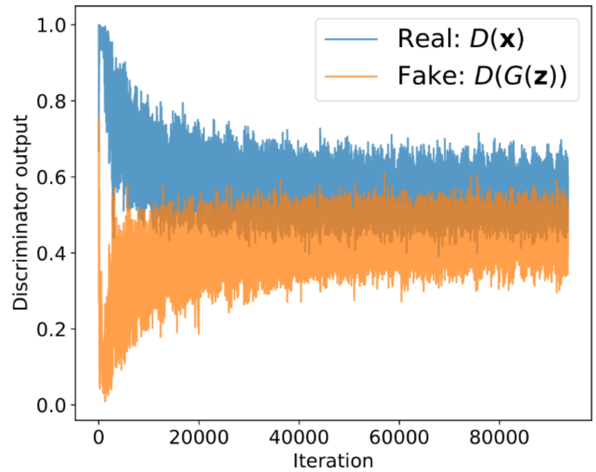


$$\phi(z) = \begin{cases} z & \text{if } z \geq 0 \\ \alpha z & \text{otherwise} \end{cases}$$

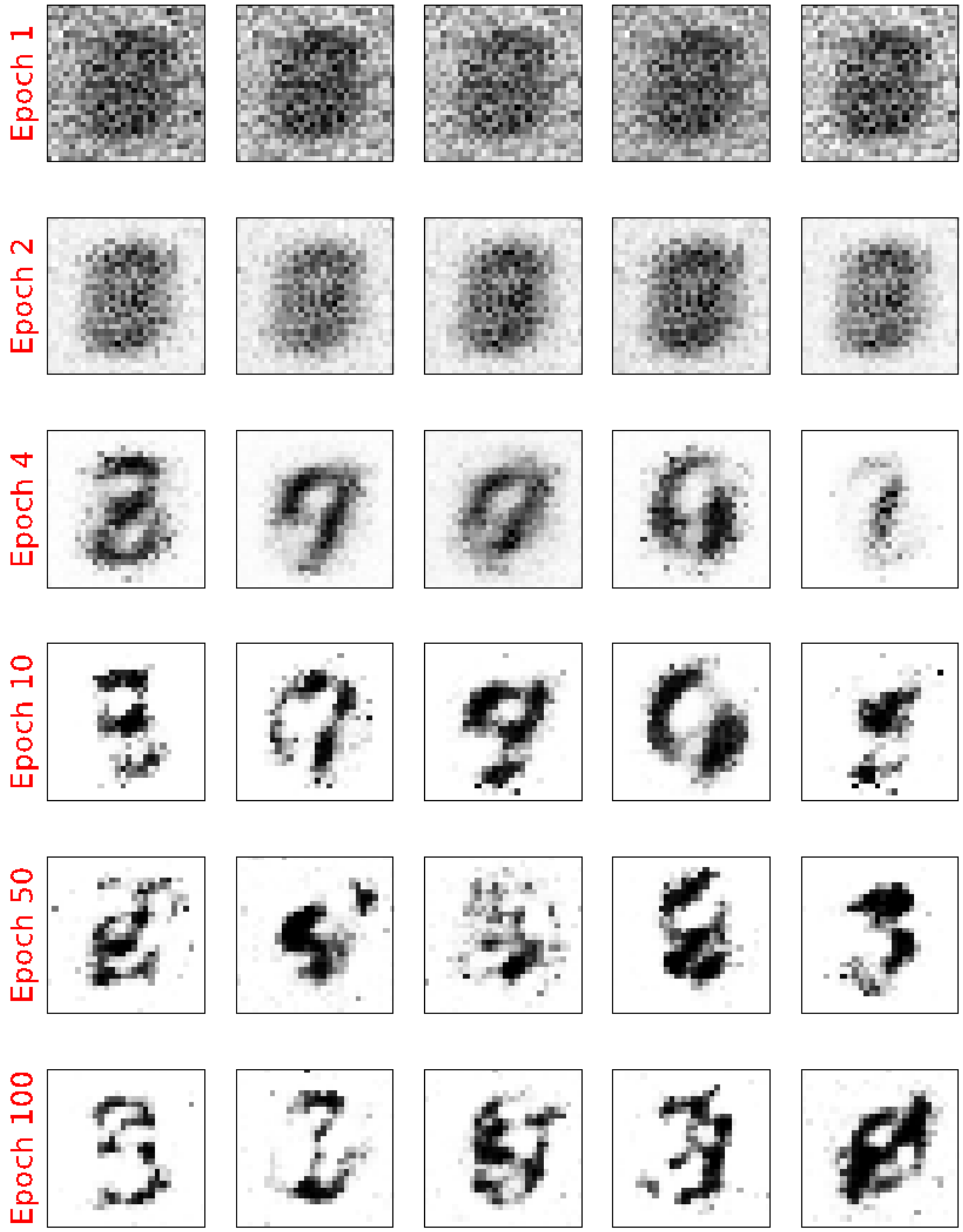


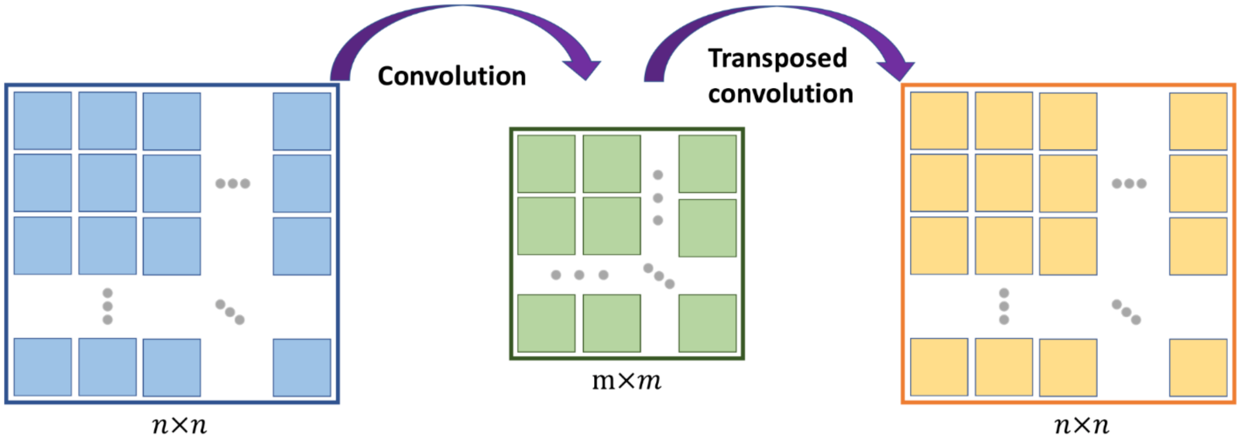


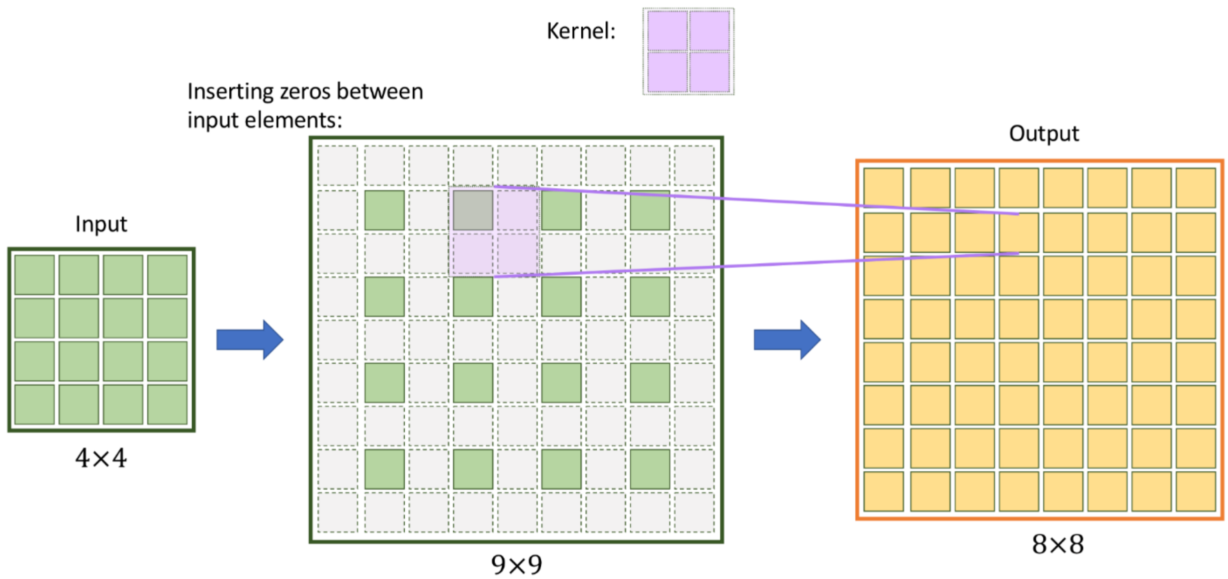
Generator loss
Discriminator loss
Loss
Iteration
12
10
8
6
4
2
0
0
20000
40000
60000
80000

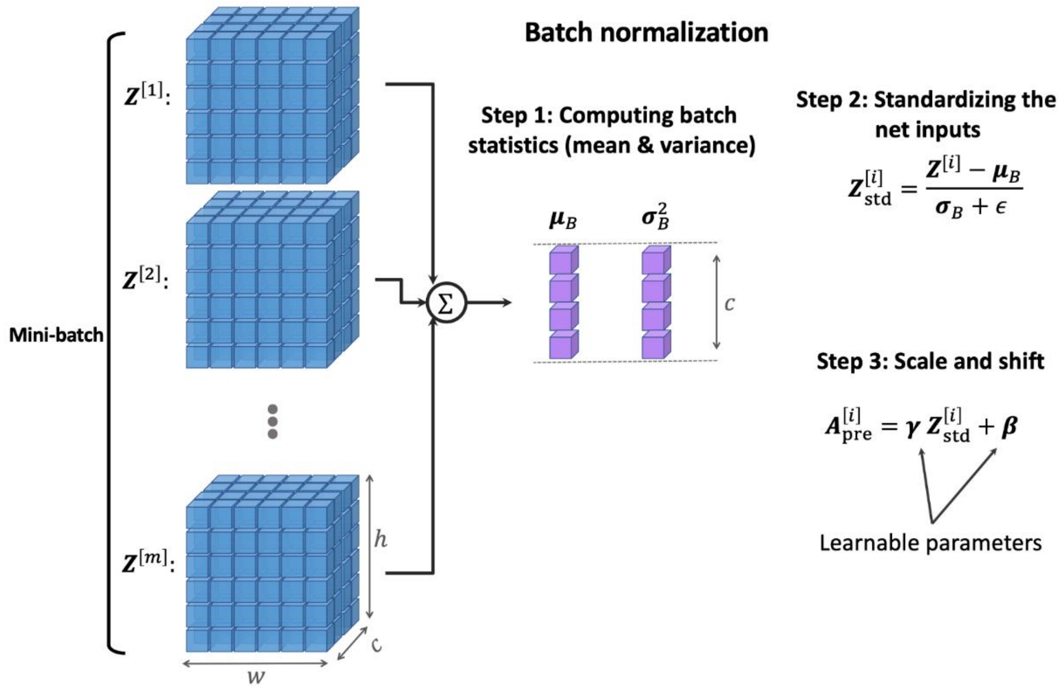


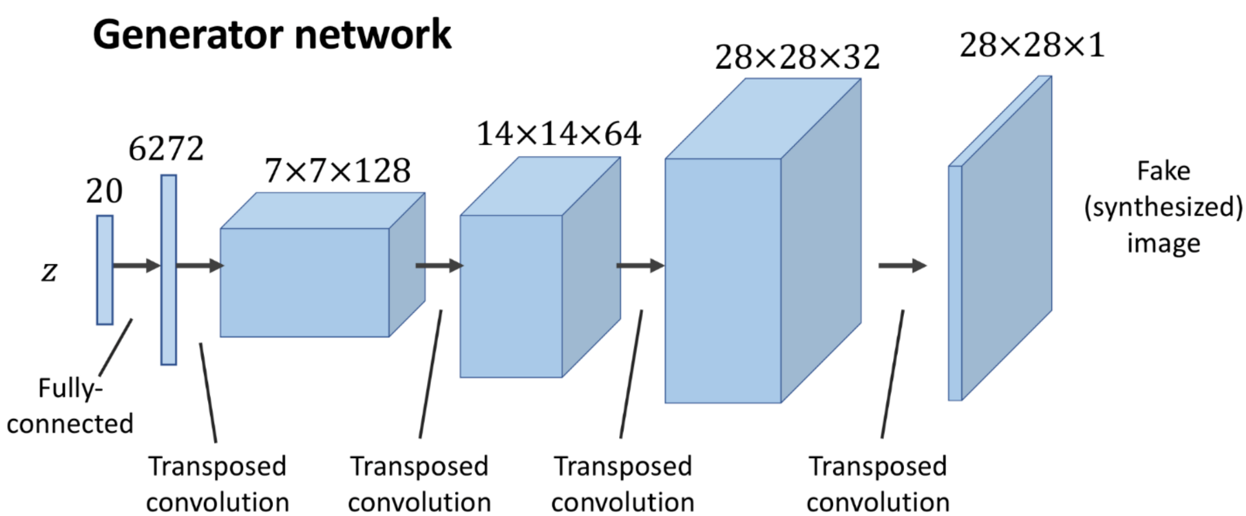
Real: $D(\mathbf{x})$
Fake: $D(G(\mathbf{z}))$
Discriminator output
Iteration
1.0
0.8
0.6
0.4
0.2
0.0
0
20000
40000
60000
80000


Epoch 1
Epoch 2
Epoch 4
Epoch 10
Epoch 50
Epoch 100

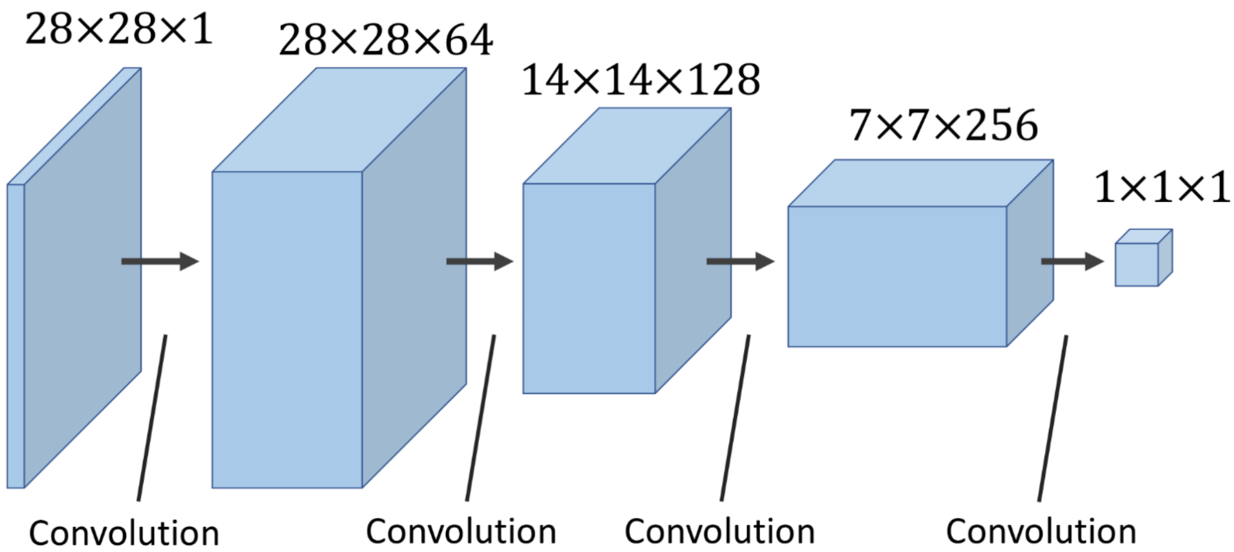

Convolution
Transposed convolution
n×n
m×m
n×n


Kernel:
Inserting zeros between input elements:
Input
Output
4×4
9×9
8×8

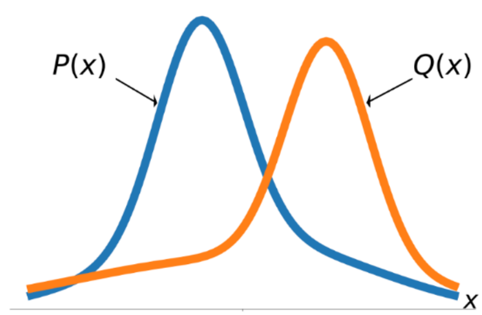

Batch normalization
Mini-batch
$Z^{[1]}$:
$Z^{[2]}$:
$Z^{[m]}$:
h
c
w
Step 1: Computing batch statistics (mean & variance)
μ_B
σ_B^2
c
Step 2: Standardizing the net inputs
$Z_{std}^{[i]} = \frac{Z^{[i]} - \mu_B}{\sigma_B + \epsilon}$
Step 3: Scale and shift
$A_{pre}^{[i]} = \gamma Z_{std}^{[i]} + \beta$
Learnable parameters

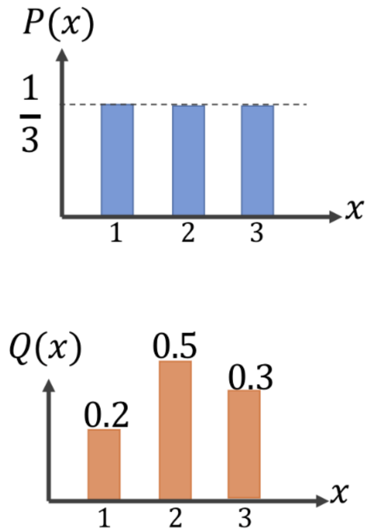

Generator network
z
20
6272
7×7×128
14×14×64
28×28×32
28×28×1
Fake (synthesized) image
Fully-connected
Transposed convolution
Transposed convolution
Transposed convolution
Transposed convolution

Discriminator network



Measures	Formulation
Total variation (TV)	$TV(P,Q) = \sup_x \lvert P(x) - Q(x) \rvert$
Kullback-Leibler (KL) divergence	$KL(P\|\|Q) = \int P(x) \log \frac{P(x)}{Q(x)} dx$
Jensen-Shannon (JS) divergence	$JS(P,Q) = \frac{1}{2}\left(KL\left(P\|\|\frac{P+Q}{2}\right) + KL\left(Q\|\|\frac{P+Q}{2}\right)\right)$
Earth mover's (EM) distance	$EM(P,Q) = \inf_{\gamma \in \Pi(P,Q)} E_{(u,v)\in\gamma}(\|\|u - v\|\|)$





Total variation:

$$TV(P, Q) = \sup_x \left\{ \left|\frac{1}{3} - 0.2\right|, \left|\frac{1}{3} - 0.5\right|, \left|\frac{1}{3} - 0.3\right| \right\} = 0.167$$

KL divergence:

$$KL(P||Q) = 0.33 \log\left(\frac{0.33}{0.2}\right) + 0.33 \log\left(\frac{0.33}{0.5}\right) + 0.33 \log\left(\frac{0.33}{0.3}\right) = 0.101$$

$$KL(Q||P) = 0.2 \log\left(\frac{0.2}{0.33}\right) + 0.5 \log\left(\frac{0.5}{0.33}\right) + 0.33 \log\left(\frac{0.3}{0.33}\right) = 0.099$$

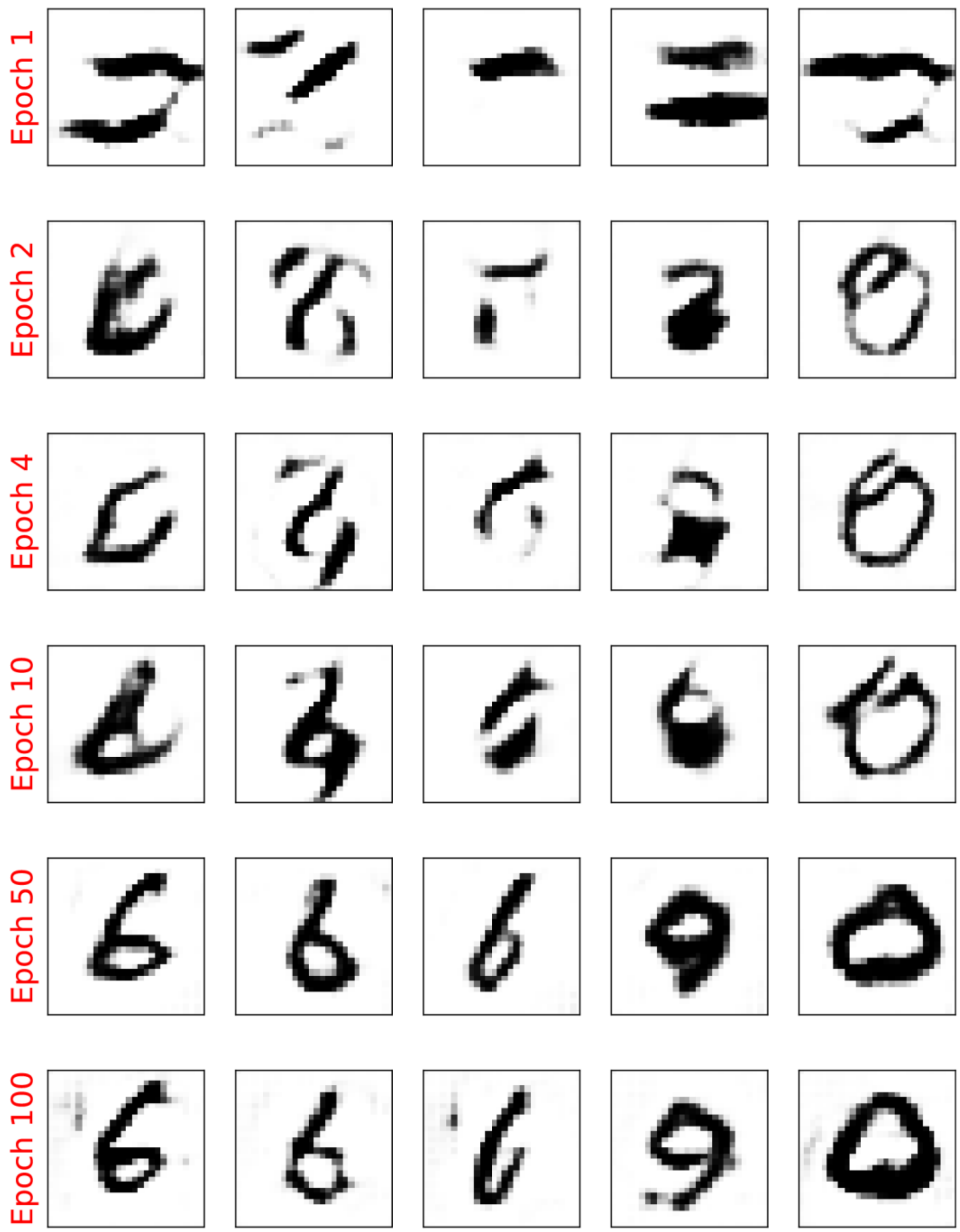
JS divergence:

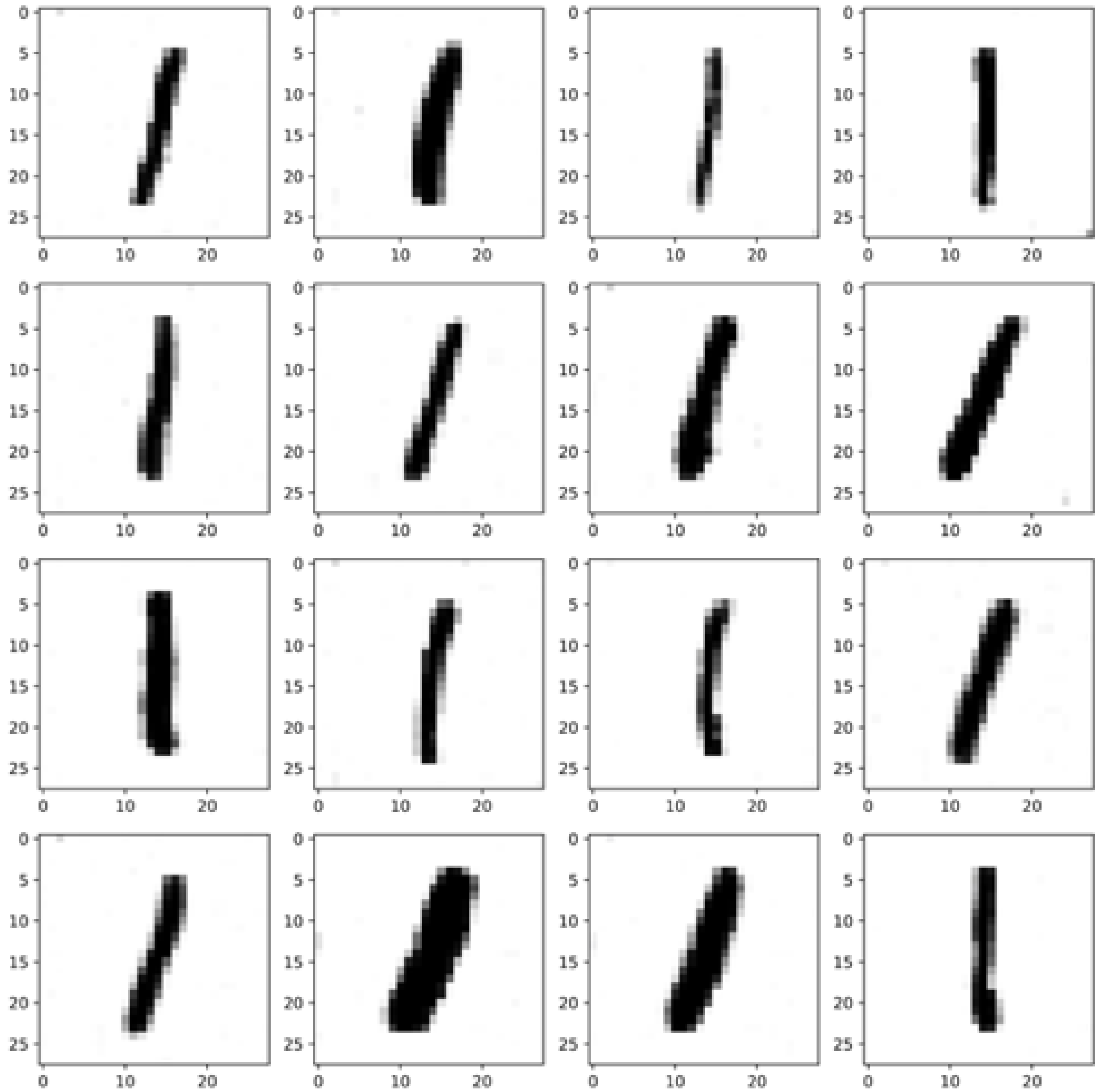
$$P_m \rightarrow \left[\frac{0.33 + 0.2}{2}, \frac{0.33 + 0.5}{2}, \frac{0.33 + 0.3}{2}\right] = [0.26, 0.42, 0.32]$$

$$\left.\begin{matrix} KL(P||P_m) = 0.0246 \\ KL(Q||P_m) = 0.0246 \end{matrix}\right\} \rightarrow JS(P||Q) = 0.0248$$

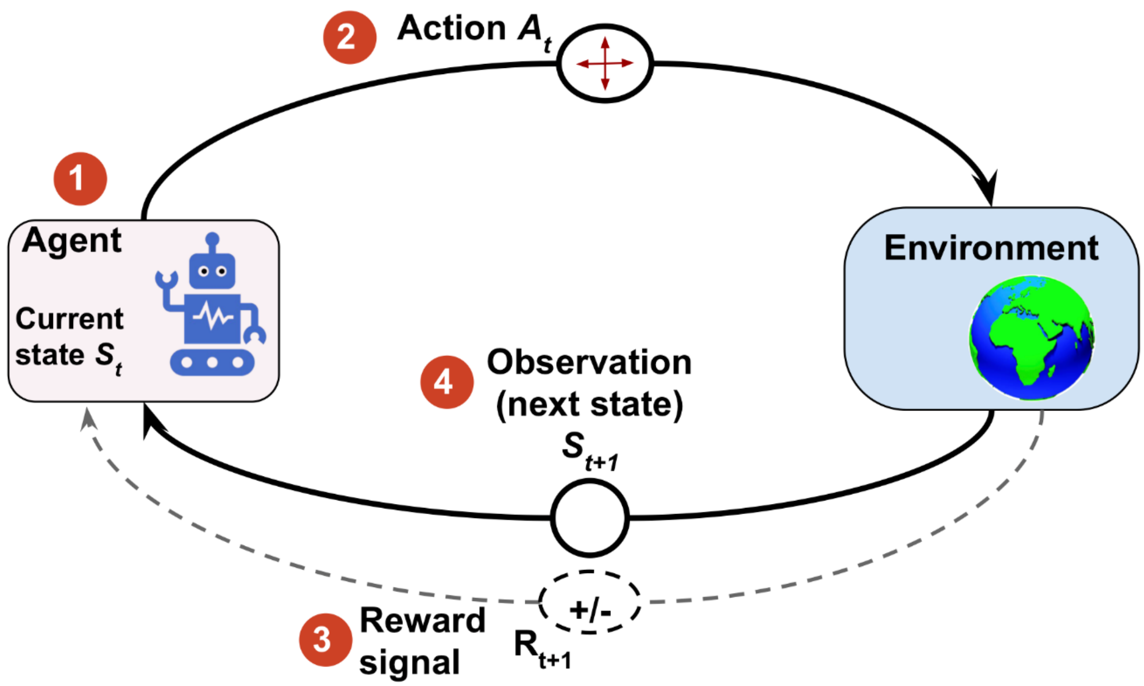
EM distance:

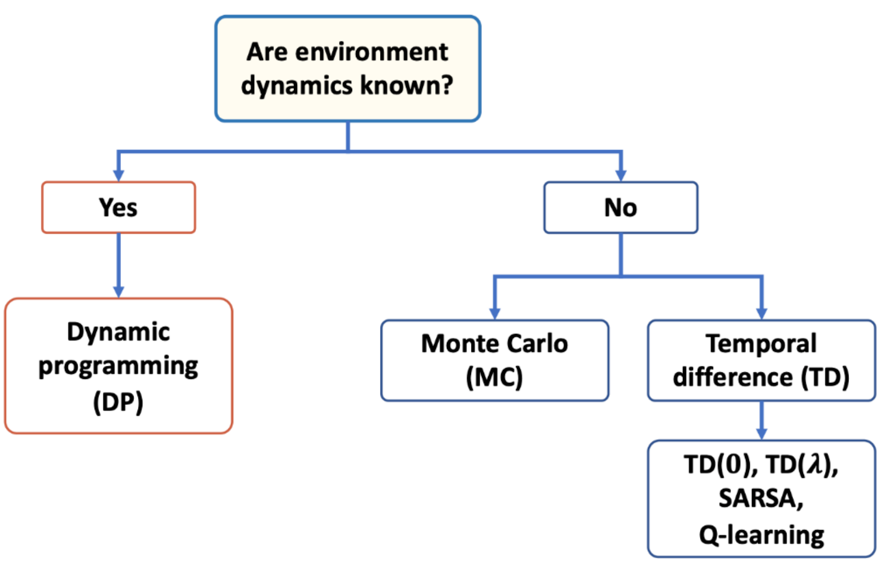
$$EM(P, Q) = (0.33 - 0.2) + (0.33 - 0.3) = 0.16$$

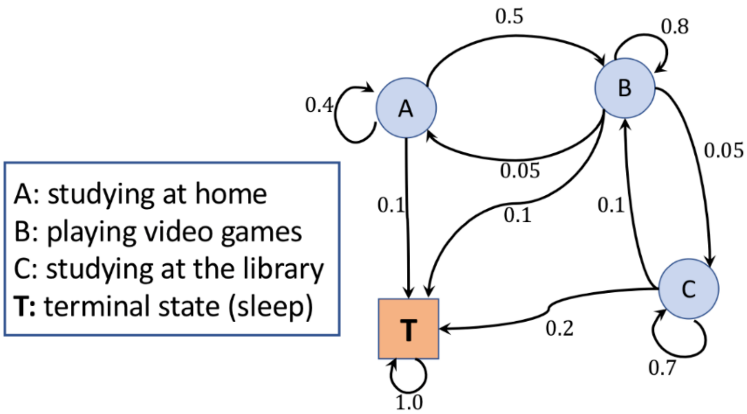

Epoch 1
Epoch 2
Epoch 4
Epoch 10
Epoch 50
Epoch 100



Chapter 18: Reinforcement Learning for Decision Making in Complex Environments

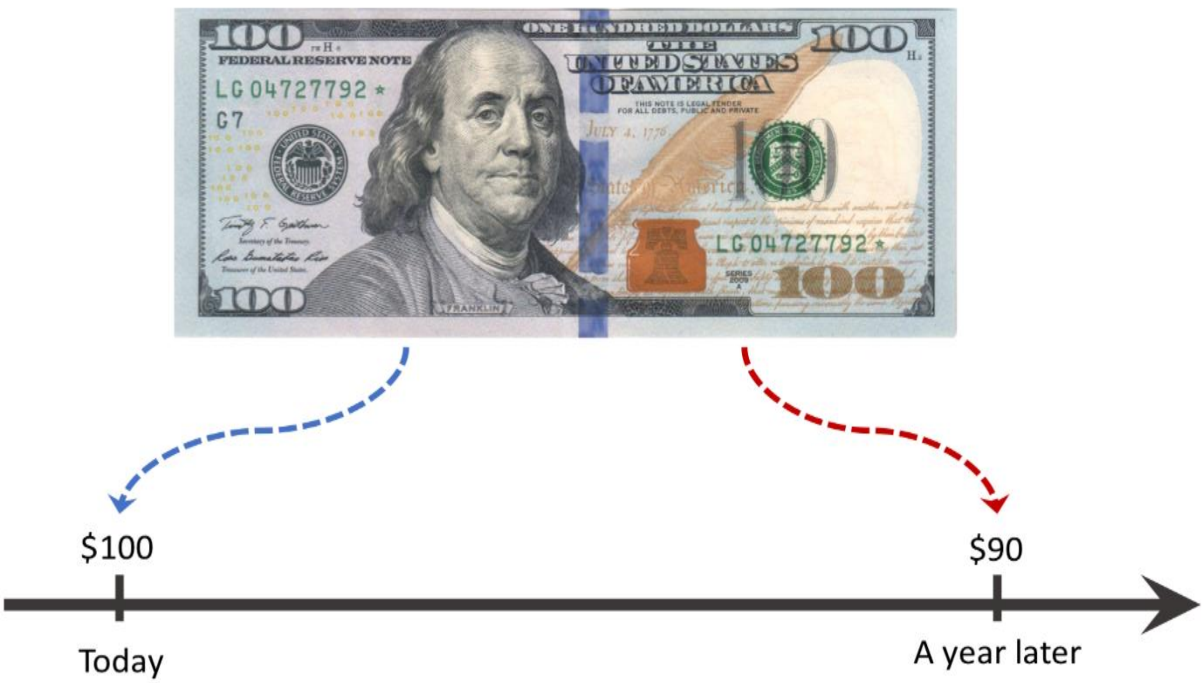


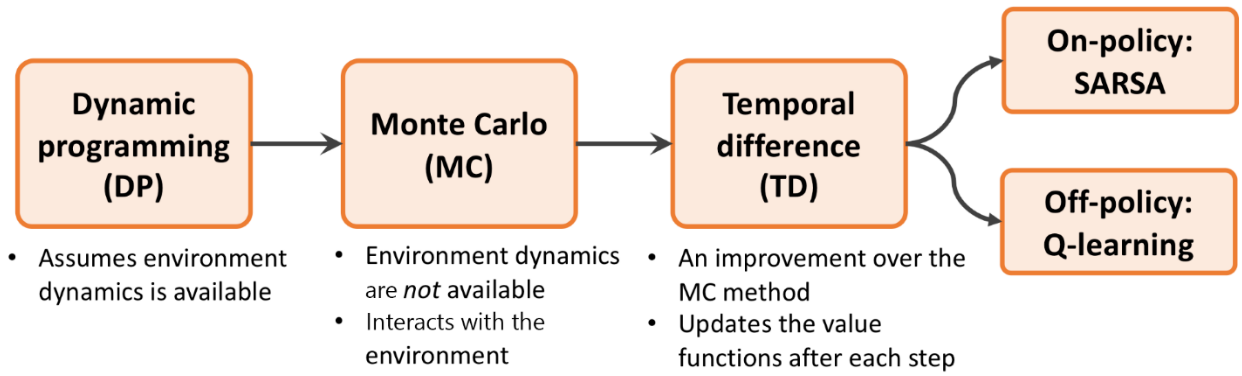


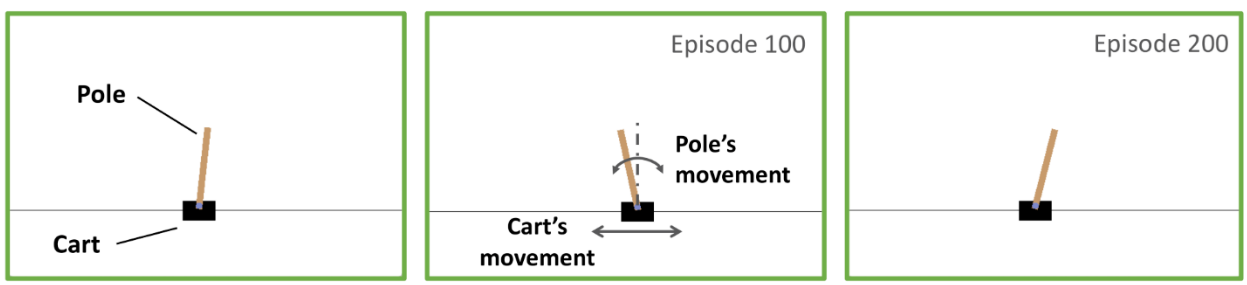


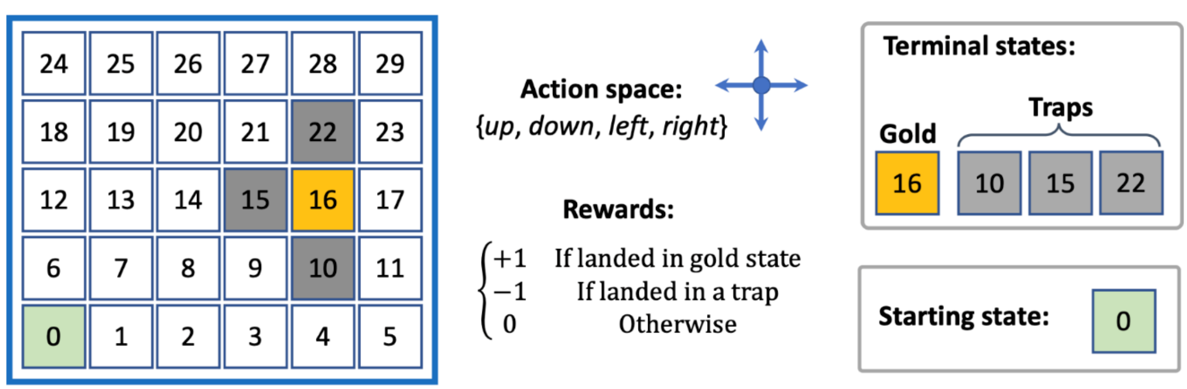
Transition probabilities:

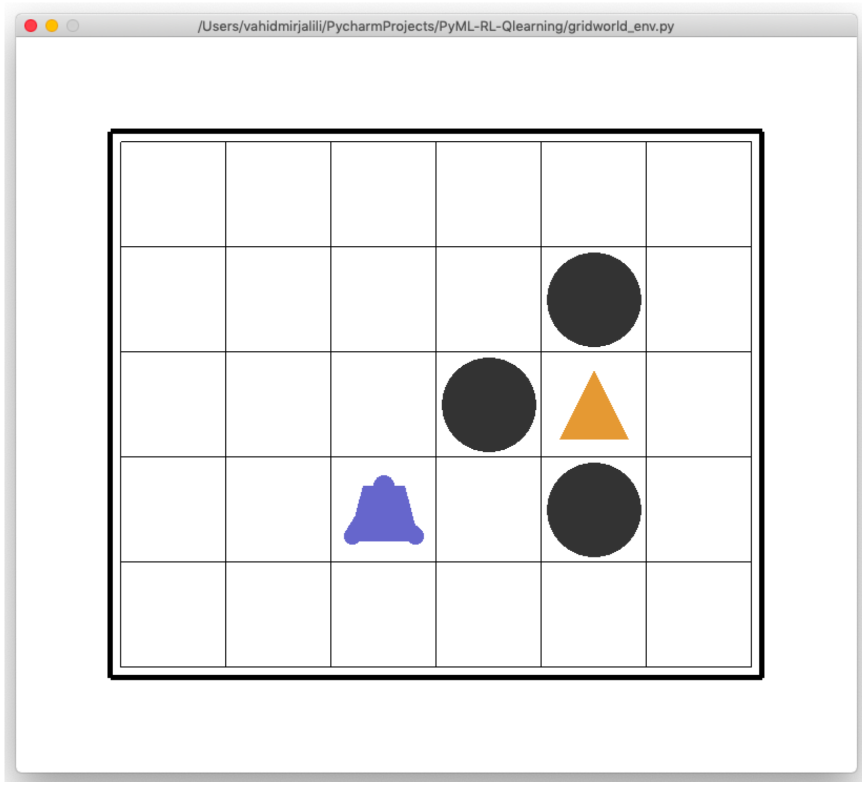
	A	*B*	*C*	*T*
A	**0.4**	**0.5**	**0.0**	**0.1**
B	**0.05**	**0.8**	**0.05**	**0.1**
C	**0.0**	**0.1**	**0.7**	**0.2**
T	**0.0**	**0.0**	**0.0**	**1.0**

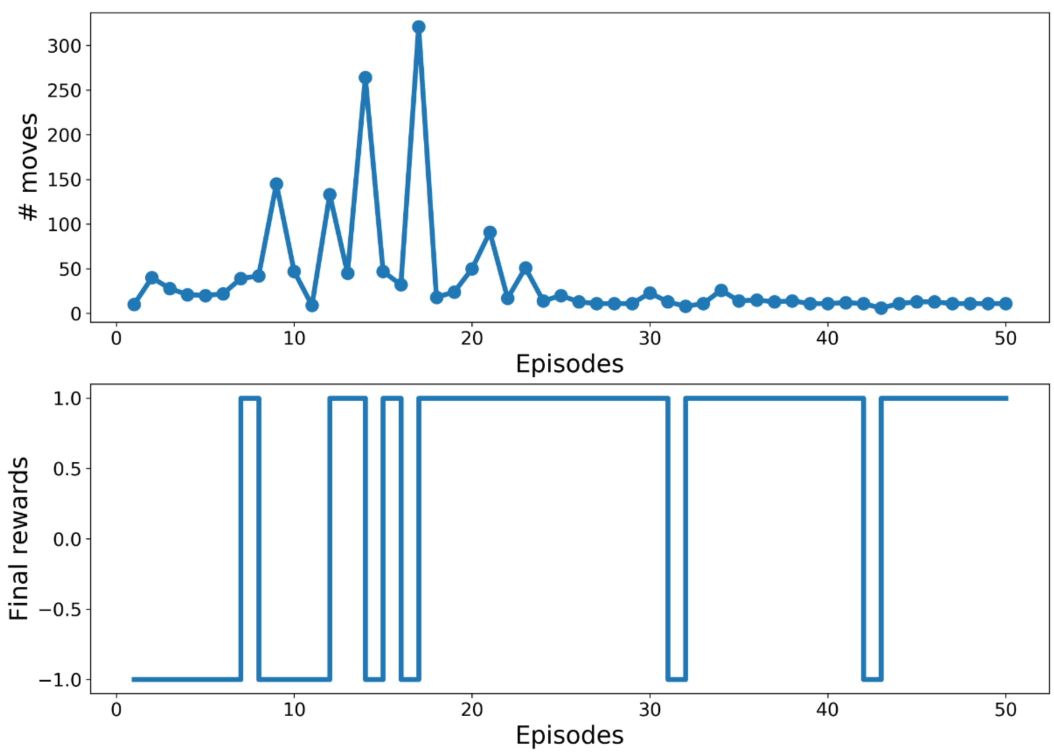



Dynamic programming (DP)
Monte Carlo (MC)
Temporal difference (TD)
On-policy: SARSA
Off-policy: Q-learning
• Assumes environment dynamics is available
• Environment dynamics are not available
• Interacts with the environment
• An improvement over the MC method
• Updates the value functions after each step

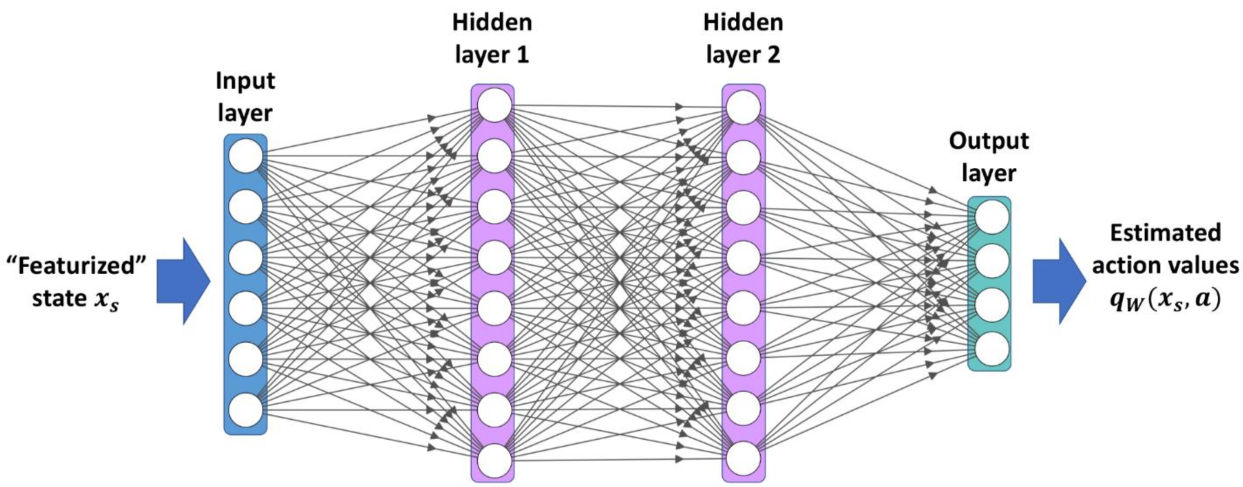

Pole
Cart
Episode 100
Pole's movement
Cart's movement
Episode 200


24 25 26 27 28 29
18 19 20 21 22 23
12 13 14 15 16 17
6 7 8 9 10 11
0 1 2 3 4 5
Action space:
{up, down, left, right}
Rewards:
+1 If landed in gold state
−1 If landed in a trap
0 Otherwise
Terminal states:
Gold
16
Traps
10 15 22
Starting state:
0

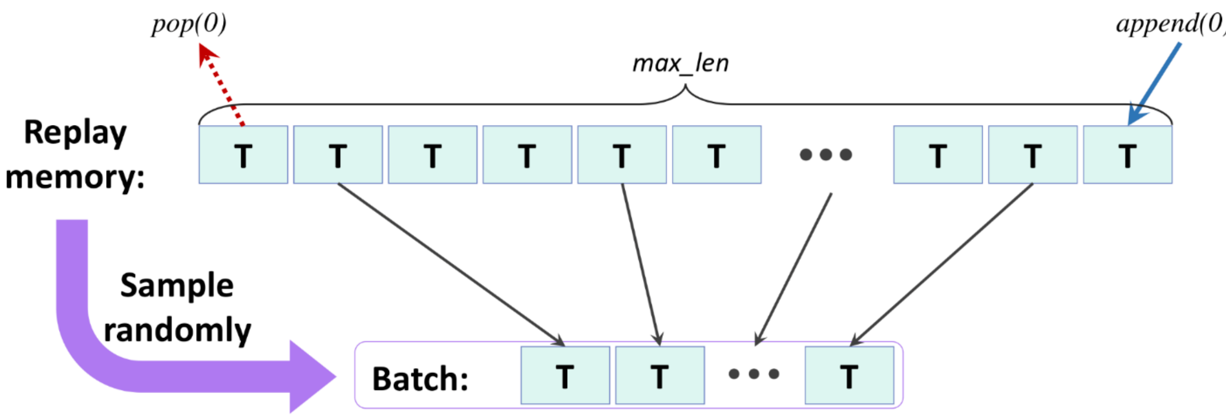

/Users/vahidmirjalili/PycharmProjects/PyML-RL-Qlearning/gridworld_env.py



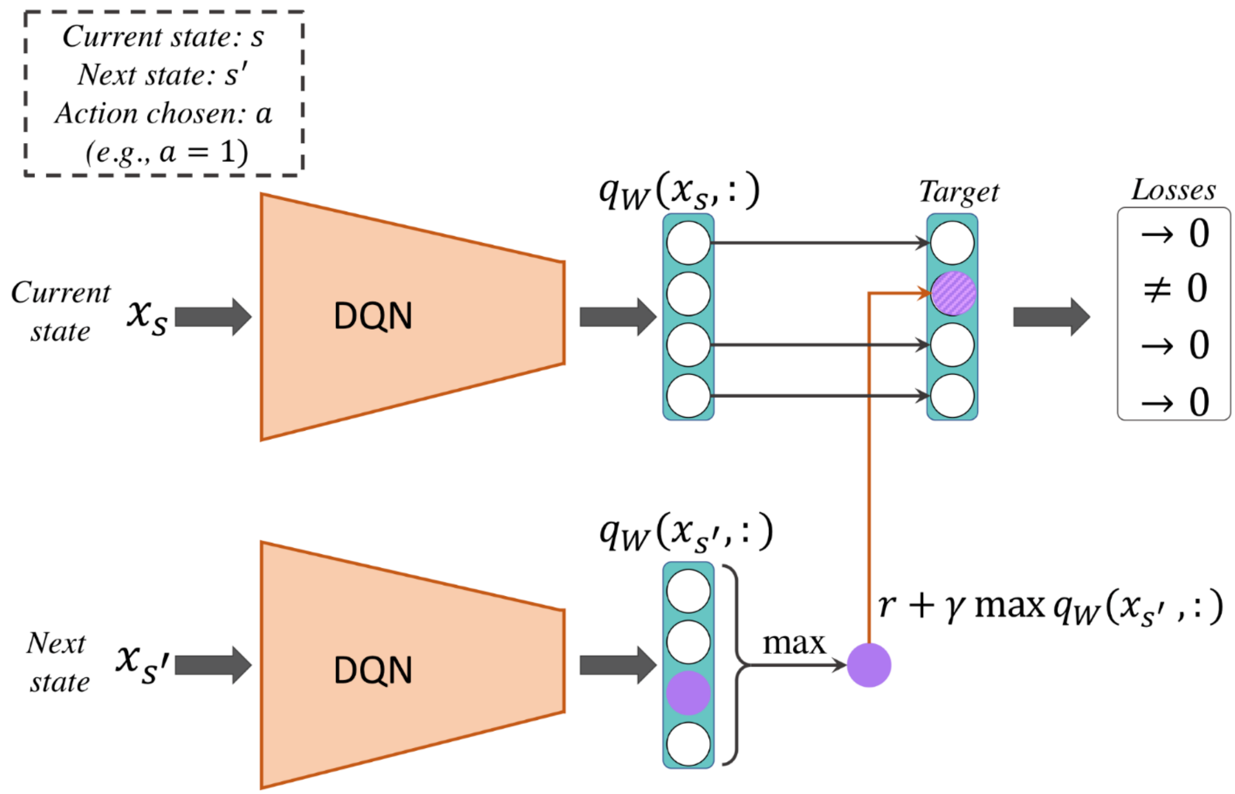
moves
300
250
200
150
100
50
0
0
10
20
30
40
50
Episodes
Final rewards
1.0
0.5
0.0
−0.5
−1.0
0
10
20
30
40
50
Episodes



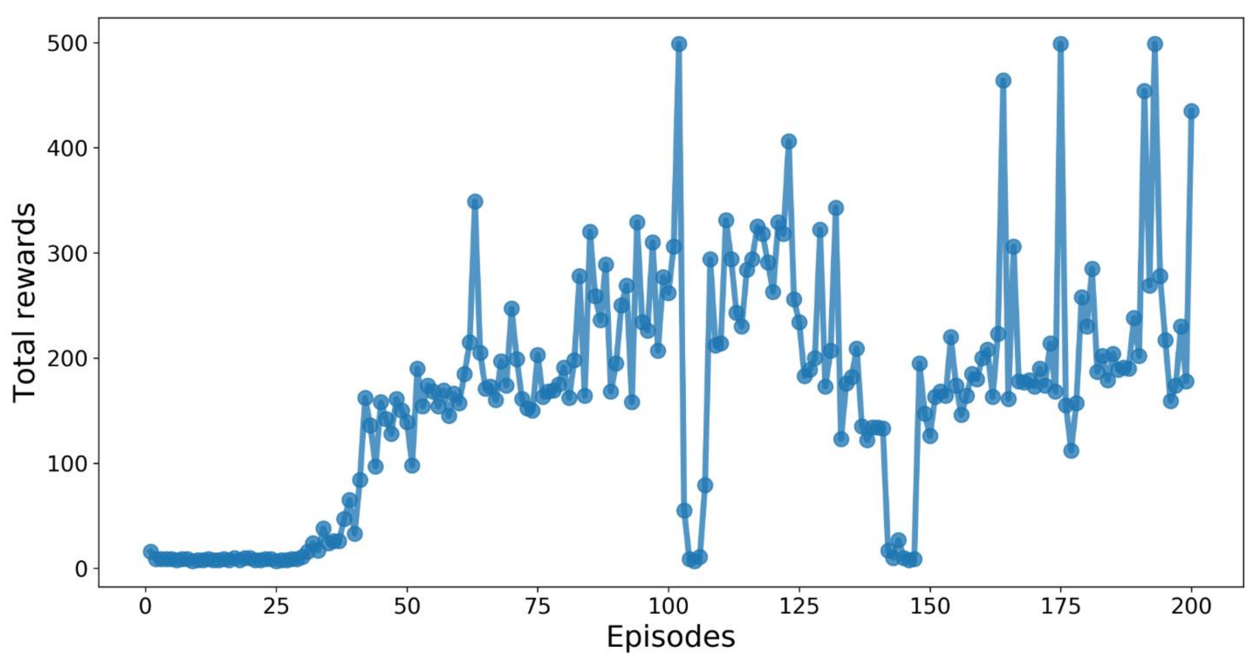
"Featurized" state x_s
Input layer
Hidden layer 1
Hidden layer 2
Output layer
Estimated action values $q_W(x_s, a)$



pop(0)
max_len
append(0)
Replay memory:
T
Sample randomly
Batch:



Current state: s
Next state: s′
Action chosen: a
(e.g., a = 1)
Current state
x_s
DQN
$q_W(x_s, :)$
Target
Losses
→ 0
≠ 0
→ 0
→ 0
Next state
$x_{s'}$
DQN
$q_W(x_{s'}, :)$
max
$r + \gamma \max q_W(x_{s'}, :)$



Total rewards
Episodes
0
100
200
300
400
500
0
25
50
75
100
125
150
175
200