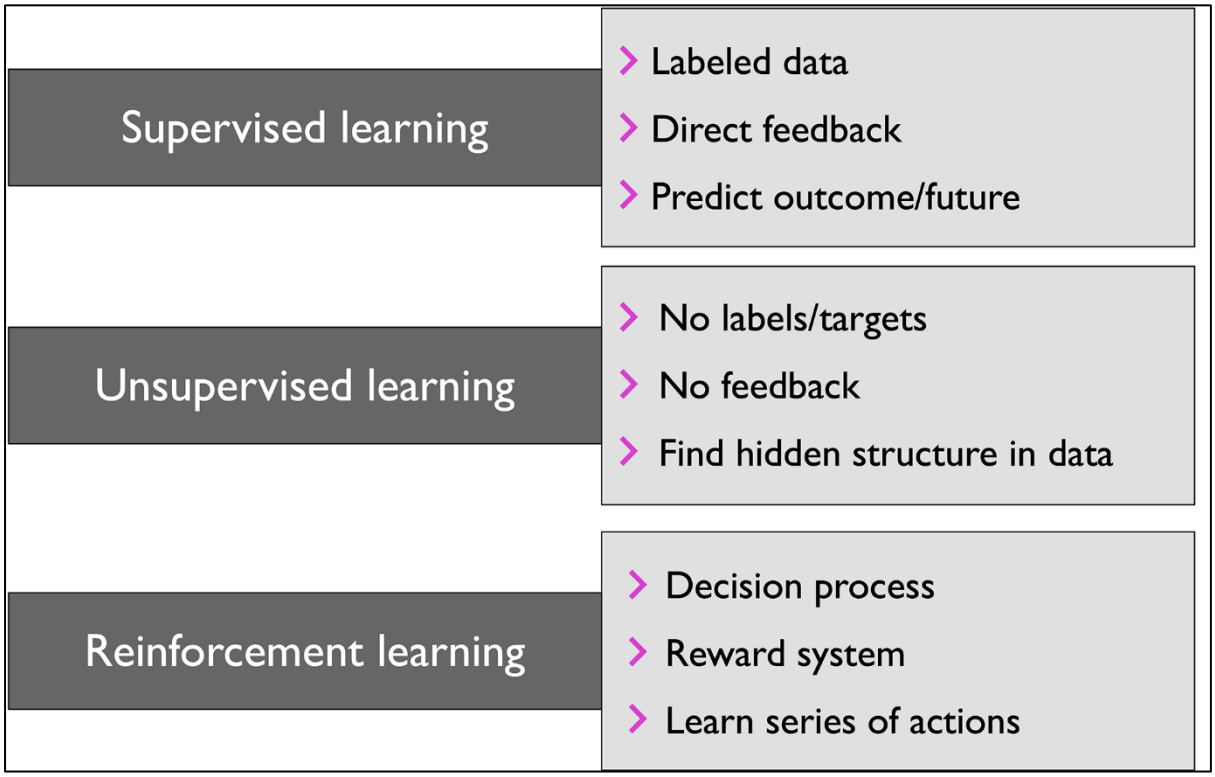
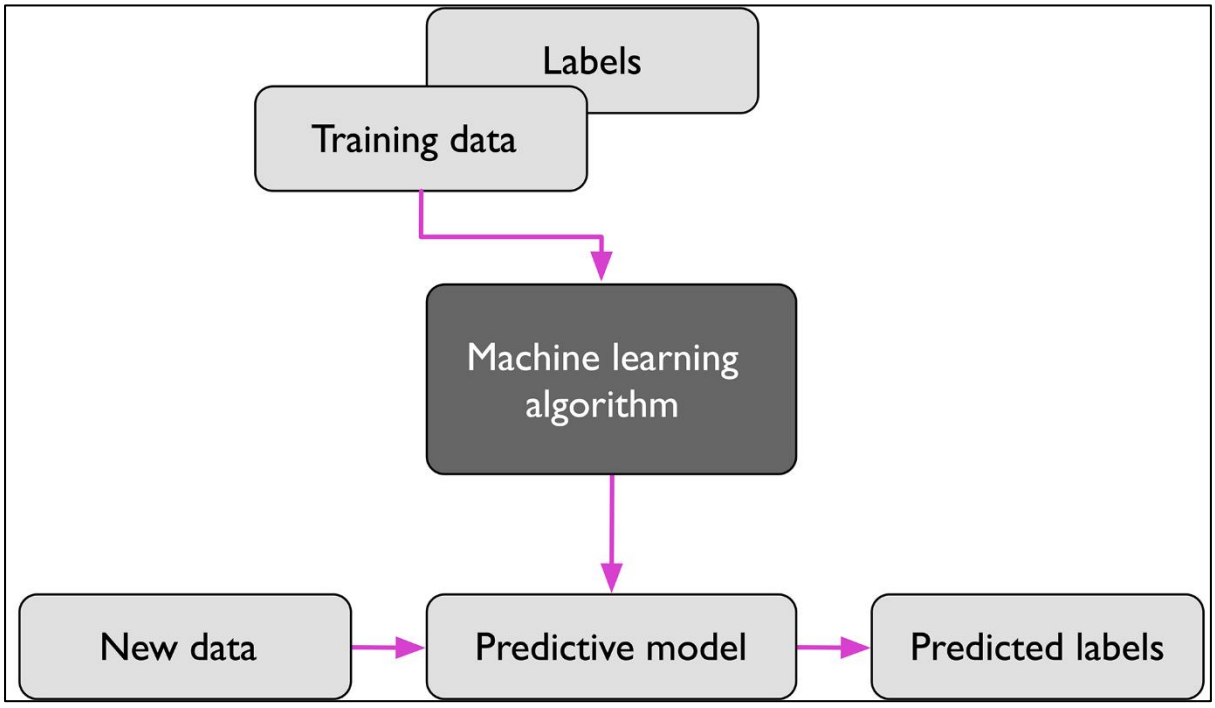
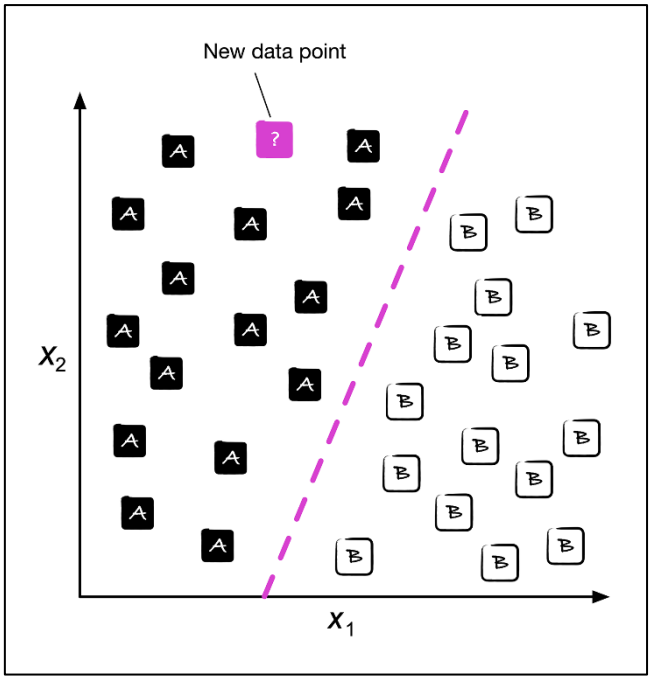
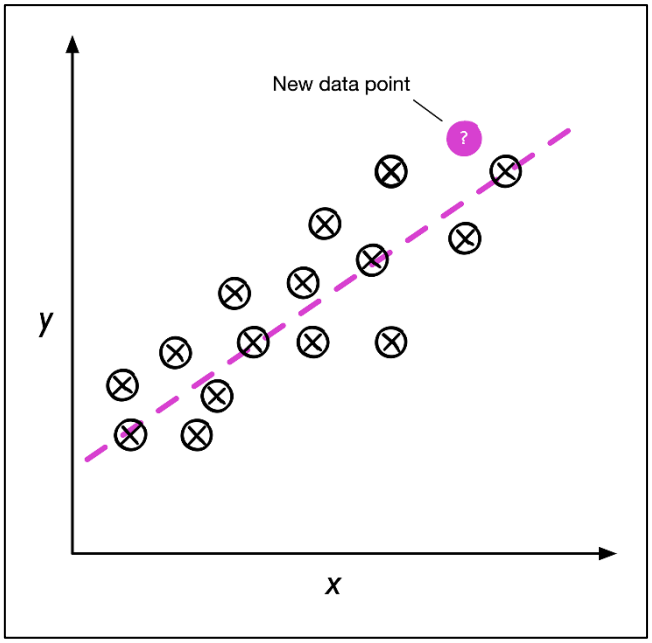
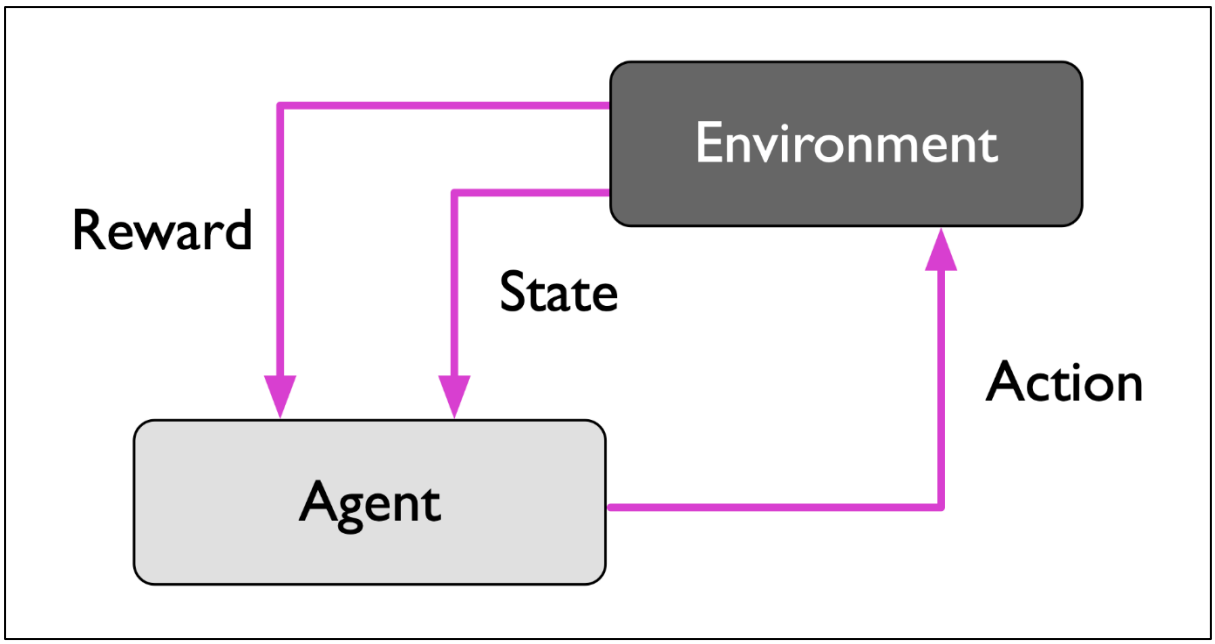
Chapter 1: Giving Computers the Ability to Learn from Data

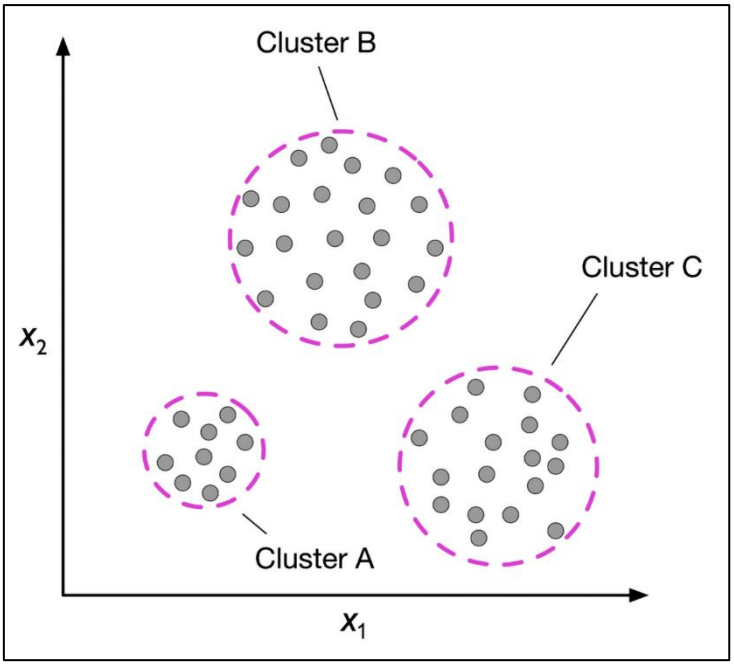


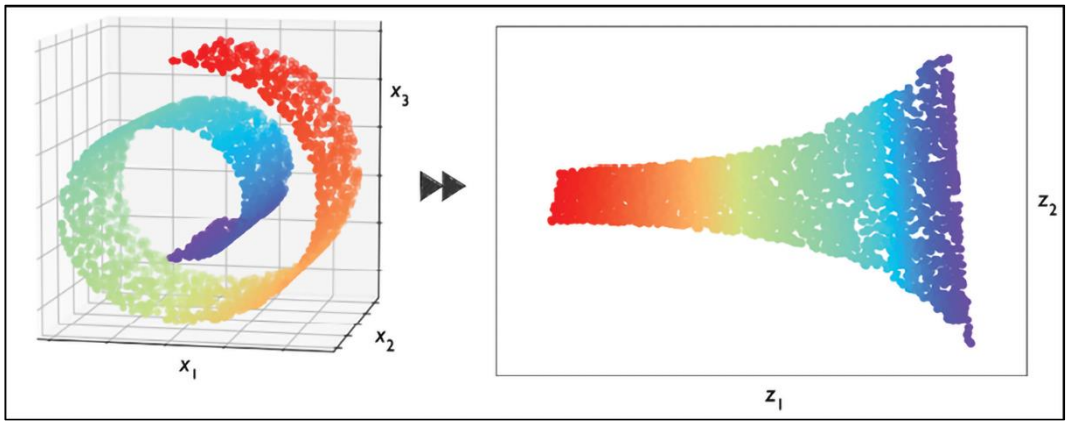



New data point
?
A
B
x_2
x_1


New data point
?
y
x


Environment
Reward
State
Action
Agent


Cluster B
Cluster C
Cluster A
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

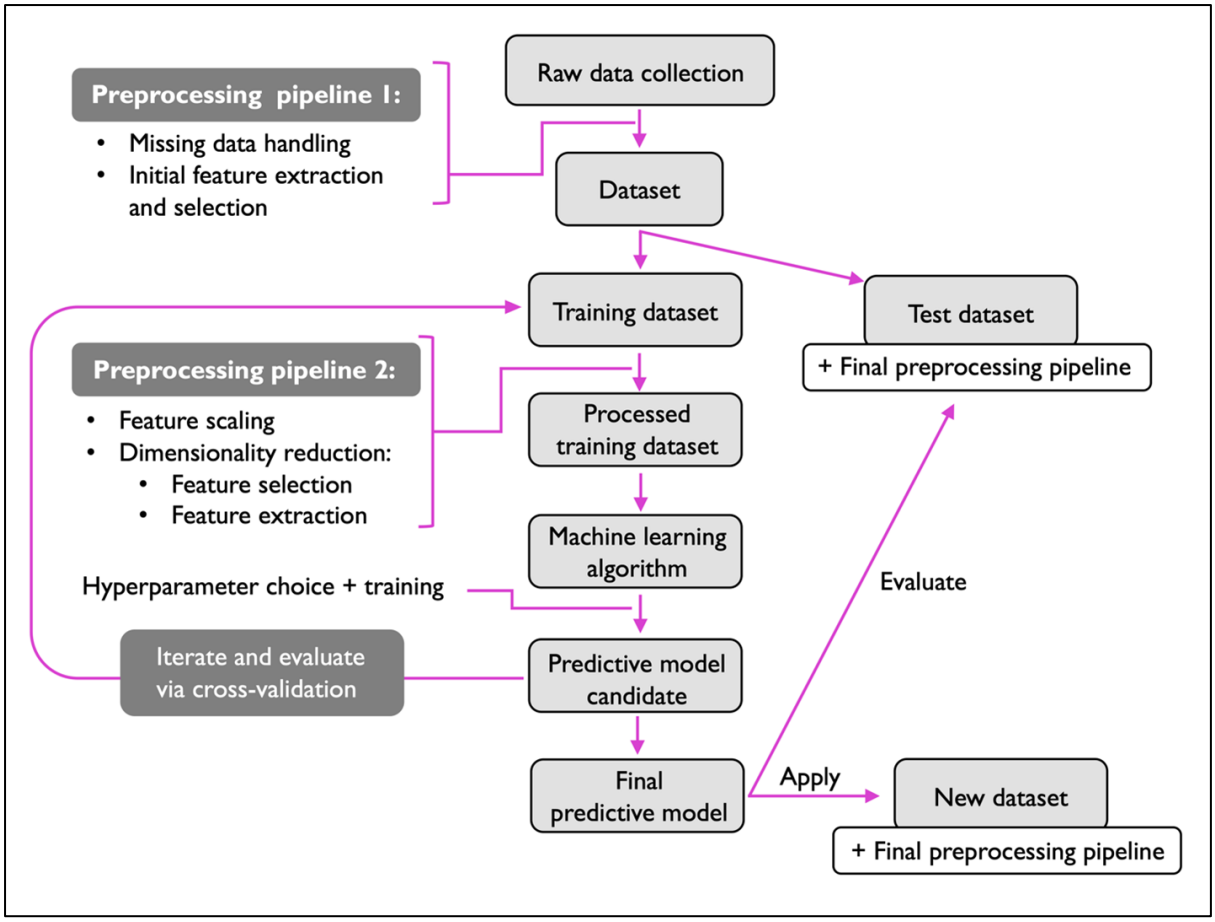
Petal

Length & width

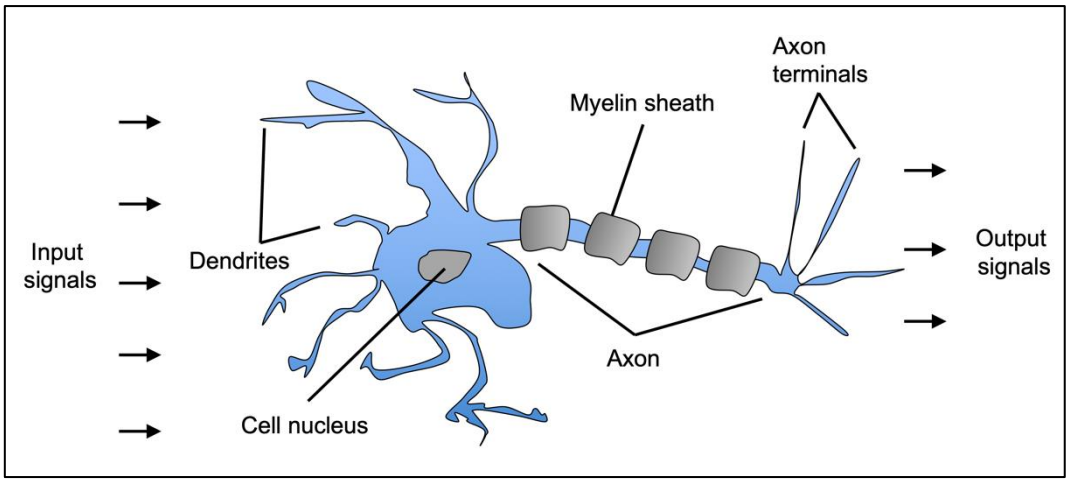
Sepal

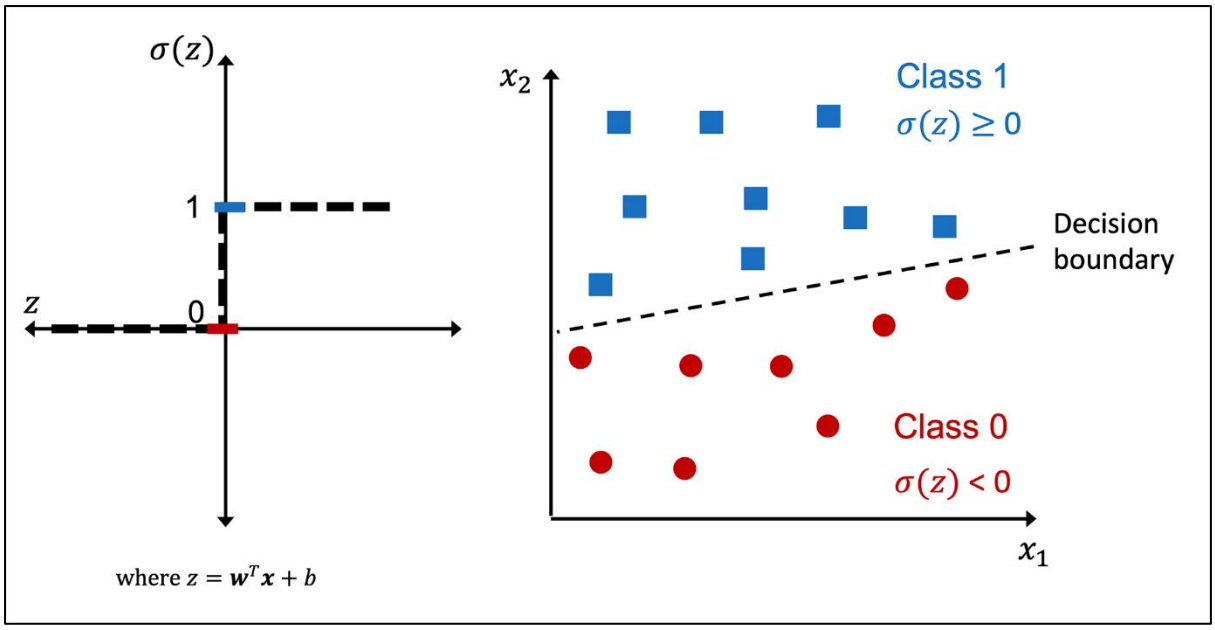
Features
(inputs, attributes, measurements, dimensions)

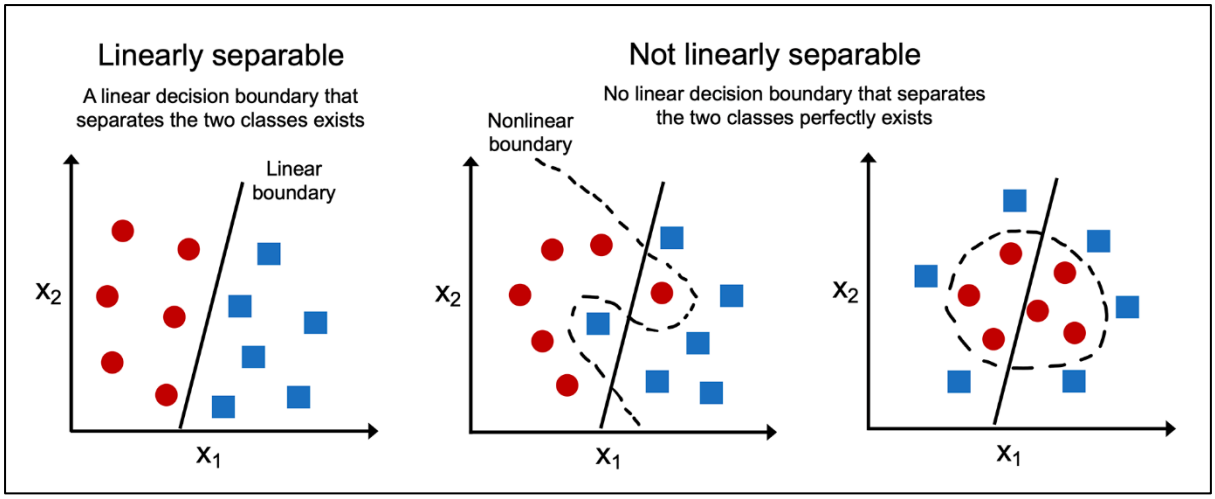
Class labels
(targets, outcomes)

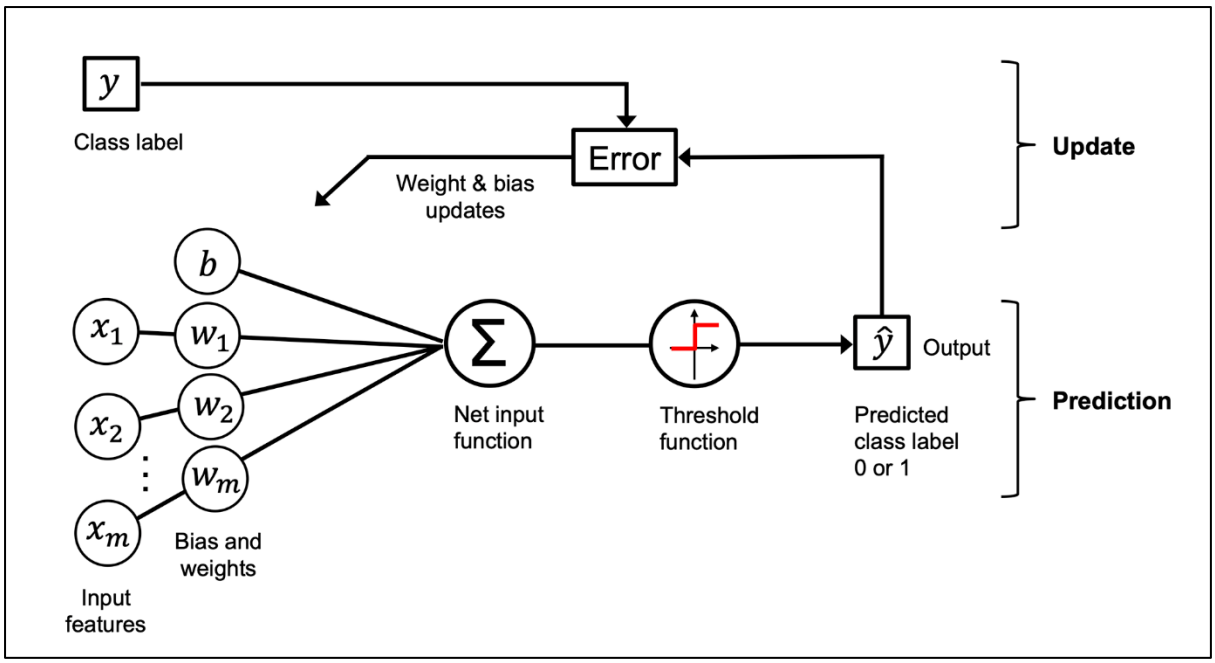

Raw data collection
Preprocessing pipeline 1:
• Missing data handling
• Initial feature extraction and selection
Dataset
Training dataset
Test dataset
+ Final preprocessing pipeline
Preprocessing pipeline 2:
• Feature scaling
• Dimensionality reduction:
• Feature selection
• Feature extraction
Processed training dataset
Machine learning algorithm
Hyperparameter choice + training
Evaluate
Iterate and evaluate via cross-validation
Predictive model candidate
Final predictive model
Apply
New dataset
+ Final preprocessing pipeline

Chapter 2: Training Simple Machine Learning Algorithms for Classification

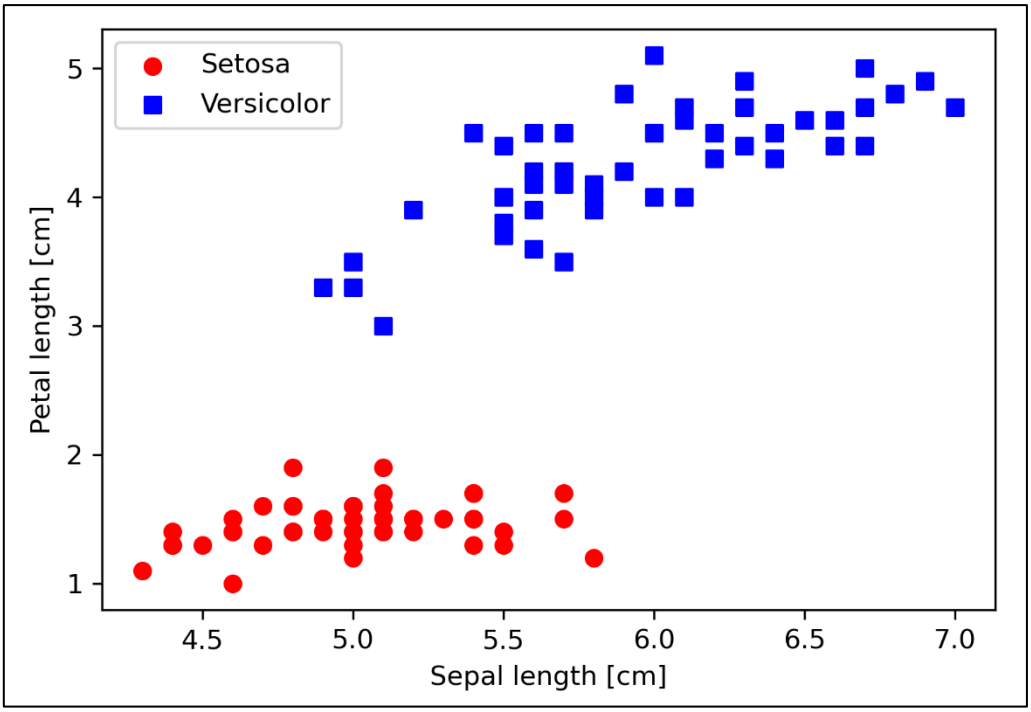


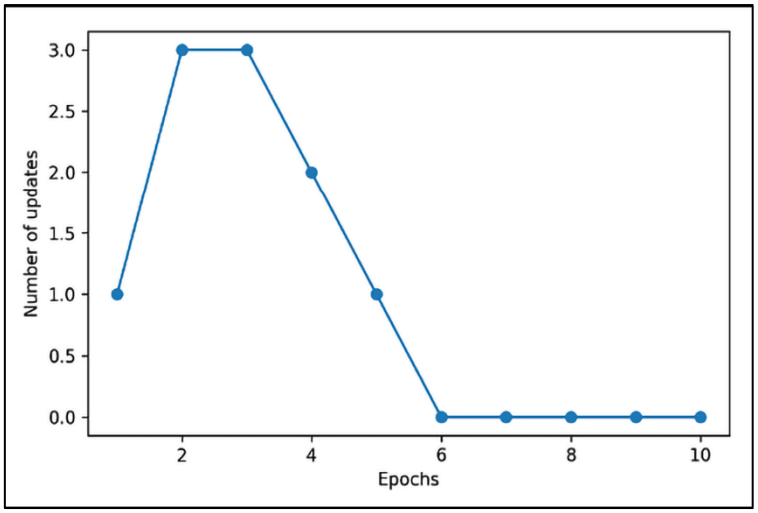


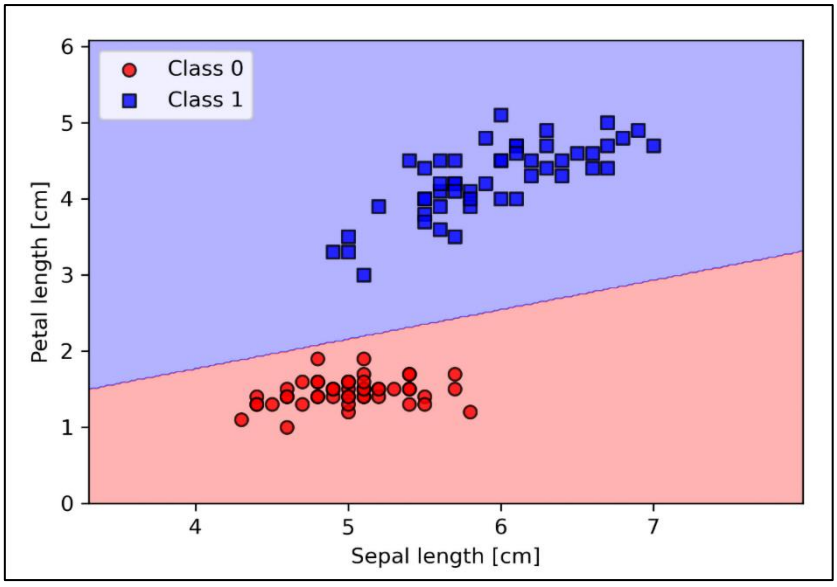

Linearly separable
A linear decision boundary that separates the two classes exists
Linear boundary
x2
x1
Not linearly separable
No linear decision boundary that separates the two classes perfectly exists
Nonlinear boundary
x2
x1
x2
x1

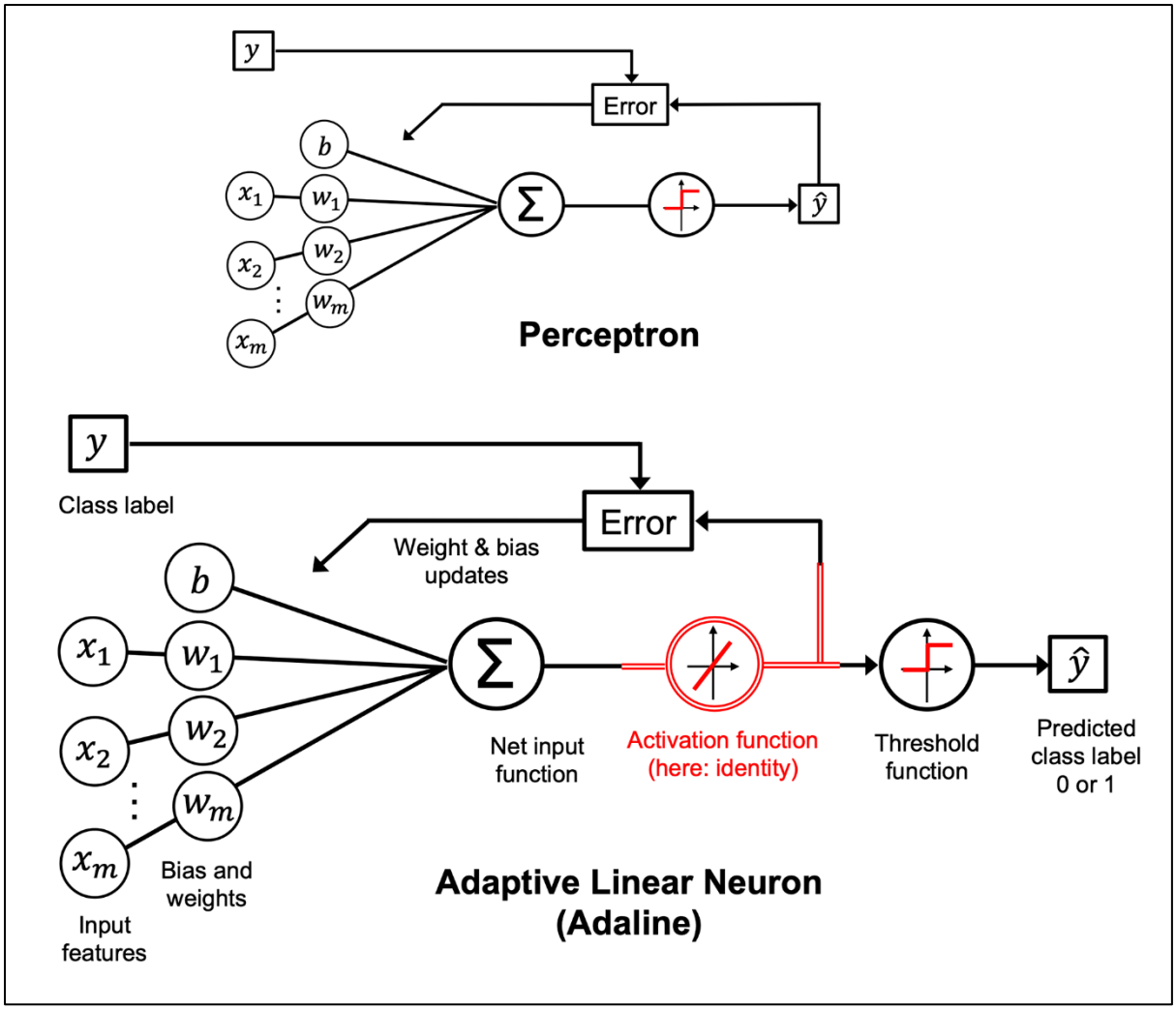

y
Class label
Error
Weight & bias updates
Update
b
x1
w1
x2
w2
xm
wm
Bias and weights
Input features
Σ
Net input function
Threshold function
ŷ
Output
Predicted class label 0 or 1
Prediction

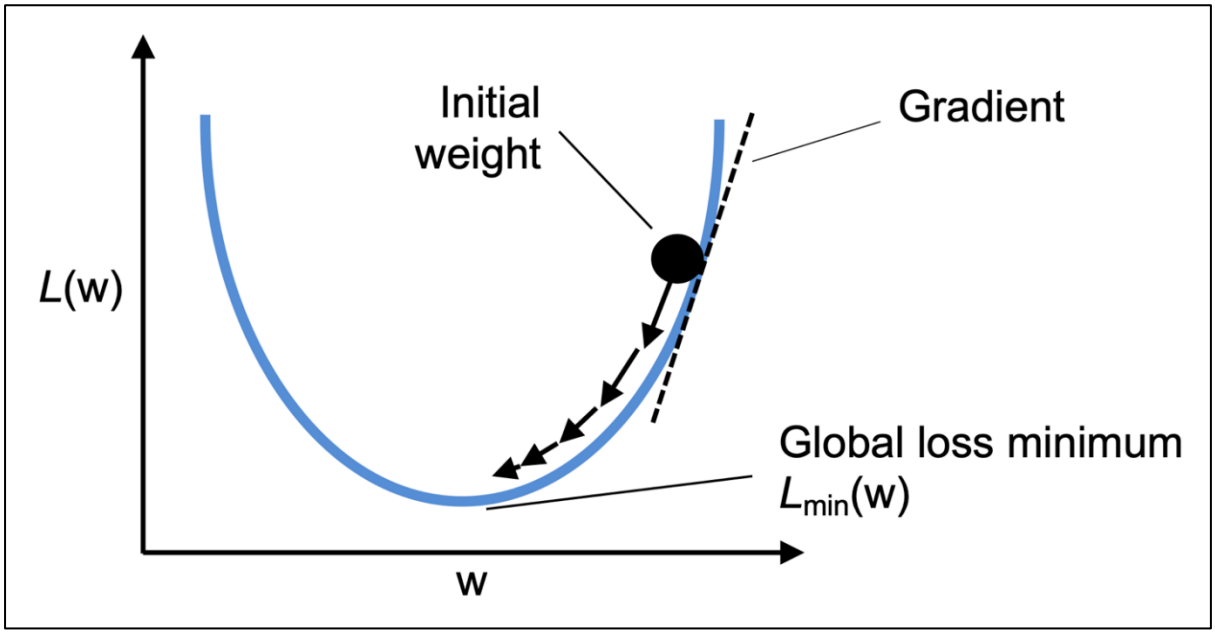
	0	1	2	3	4
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

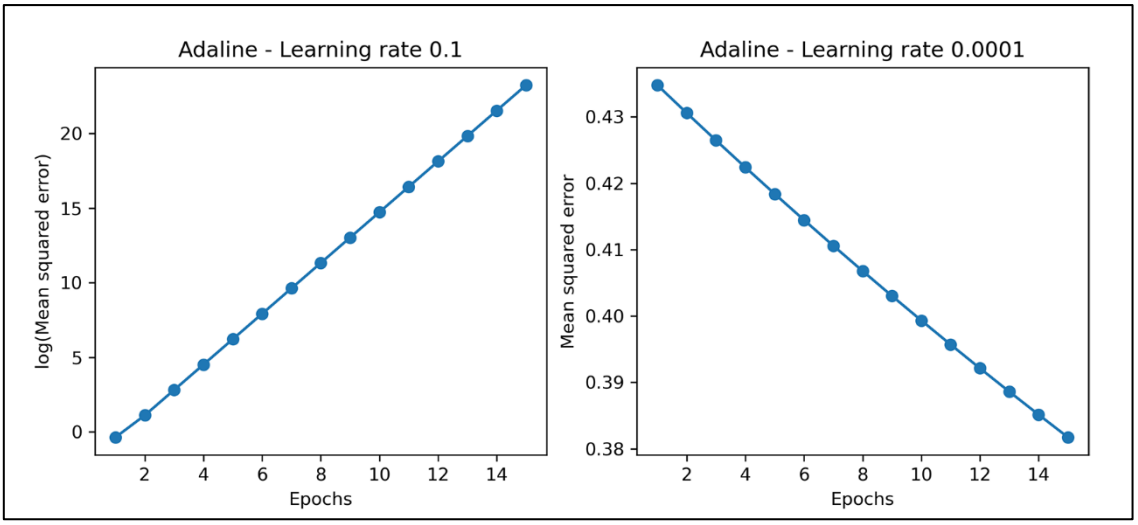


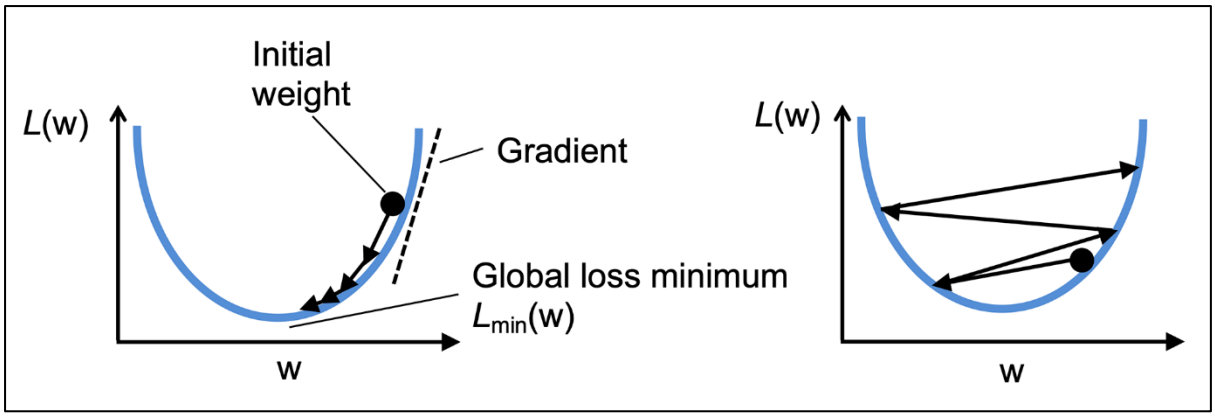

Number of updates
Epochs
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
8
10

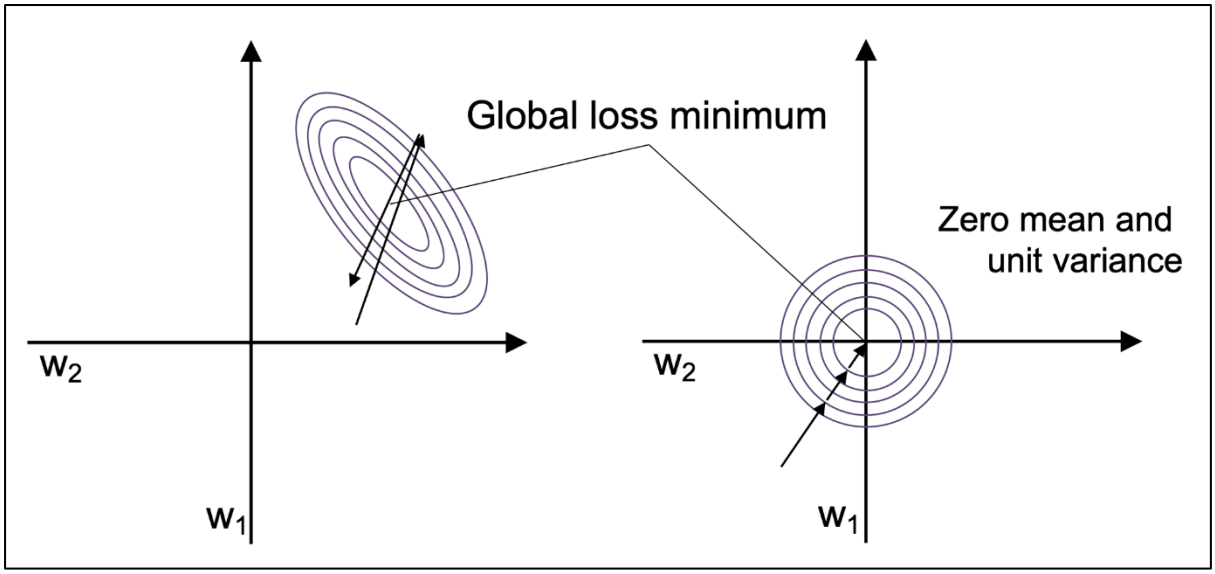

Class 0
Class 1
Petal length [cm]
Sepal length [cm]
0
1
2
3
4
5
6
4
5
6
7

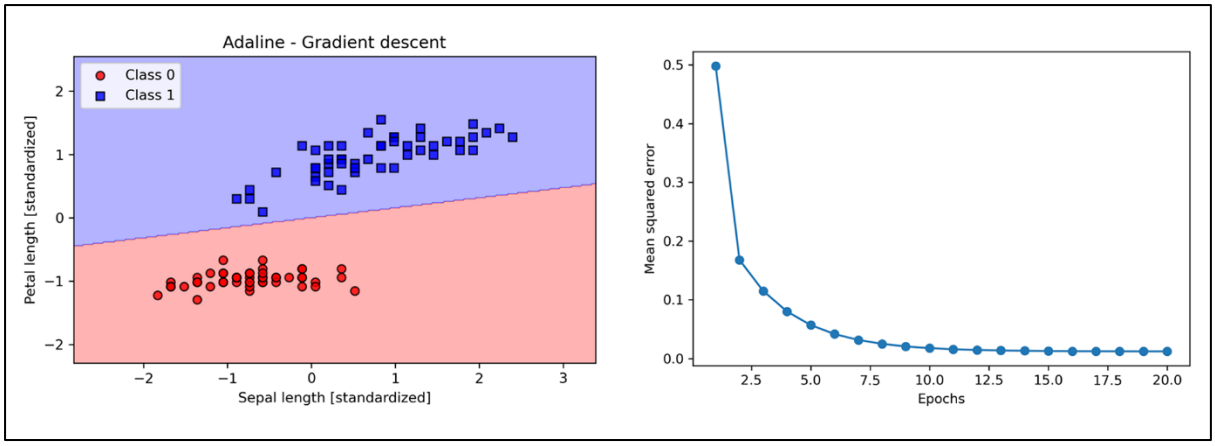

y
Error
b
x1
w1
x2
w2
wm
xm
ŷ
Perceptron
y
Class label
Error
Weight & bias updates
b
x1
w1
x2
w2
wm
xm
Net input function
Activation function (here: identity)
Threshold function
ŷ
Predicted class label 0 or 1
Bias and weights
Input features
Adaptive Linear Neuron (Adaline)

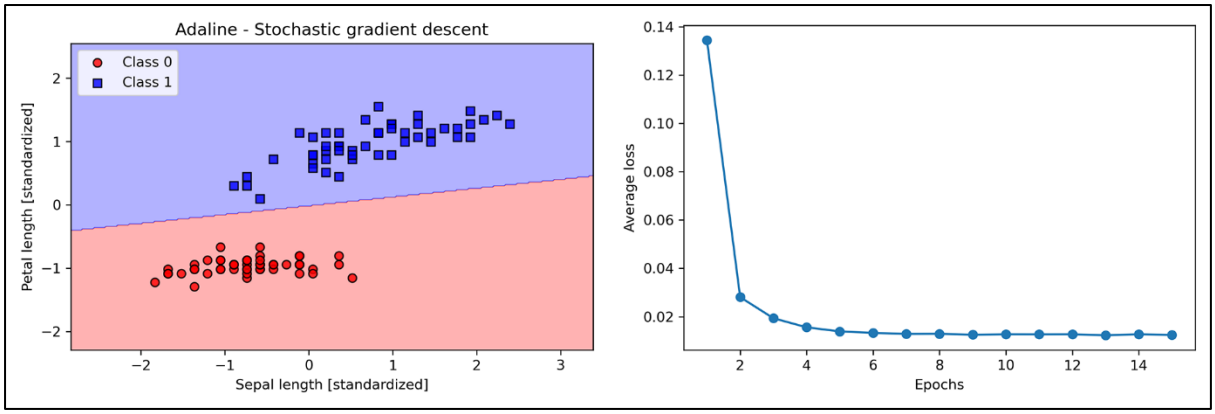

Initial weight
Gradient
L(w)
Global loss minimum
Lmin(w)
w


Adaline - Learning rate 0.1
log(Mean squared error)
Epochs
Adaline - Learning rate 0.0001
Mean squared error
Epochs

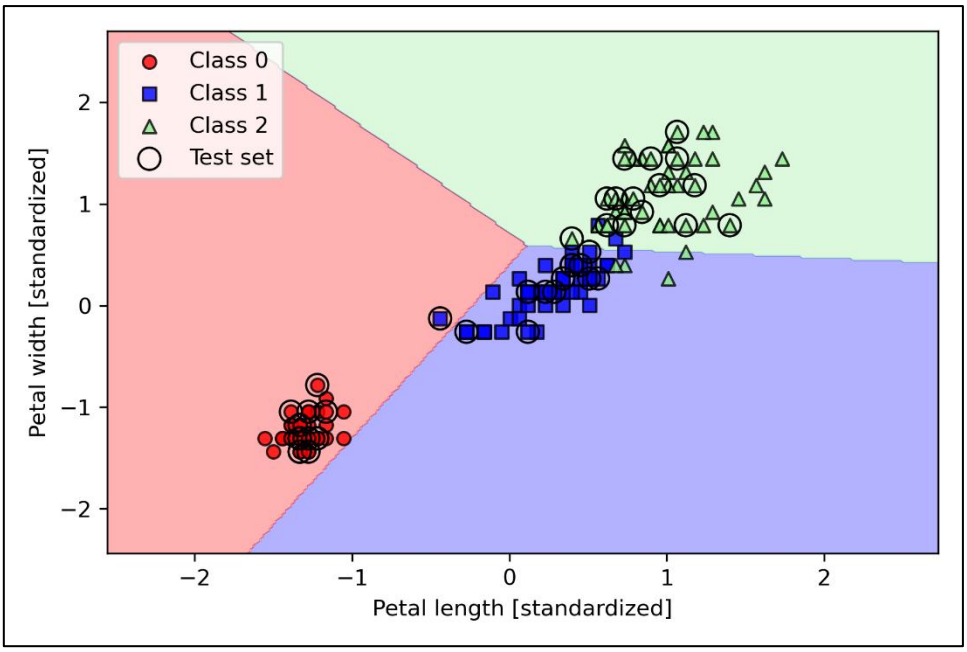

Initial
weight
L(w)
Gradient
Global loss minimum
$L_{min}(w)$
w
L(w)
w

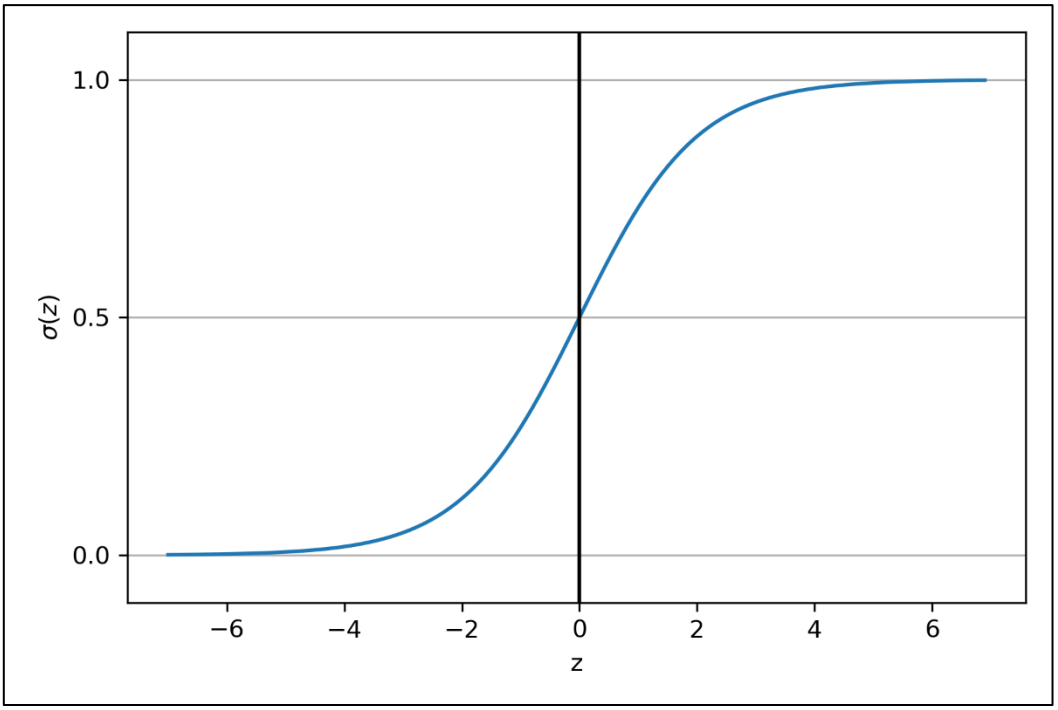

Global loss minimum
Zero mean and
unit variance
w2
w1
w2
w1

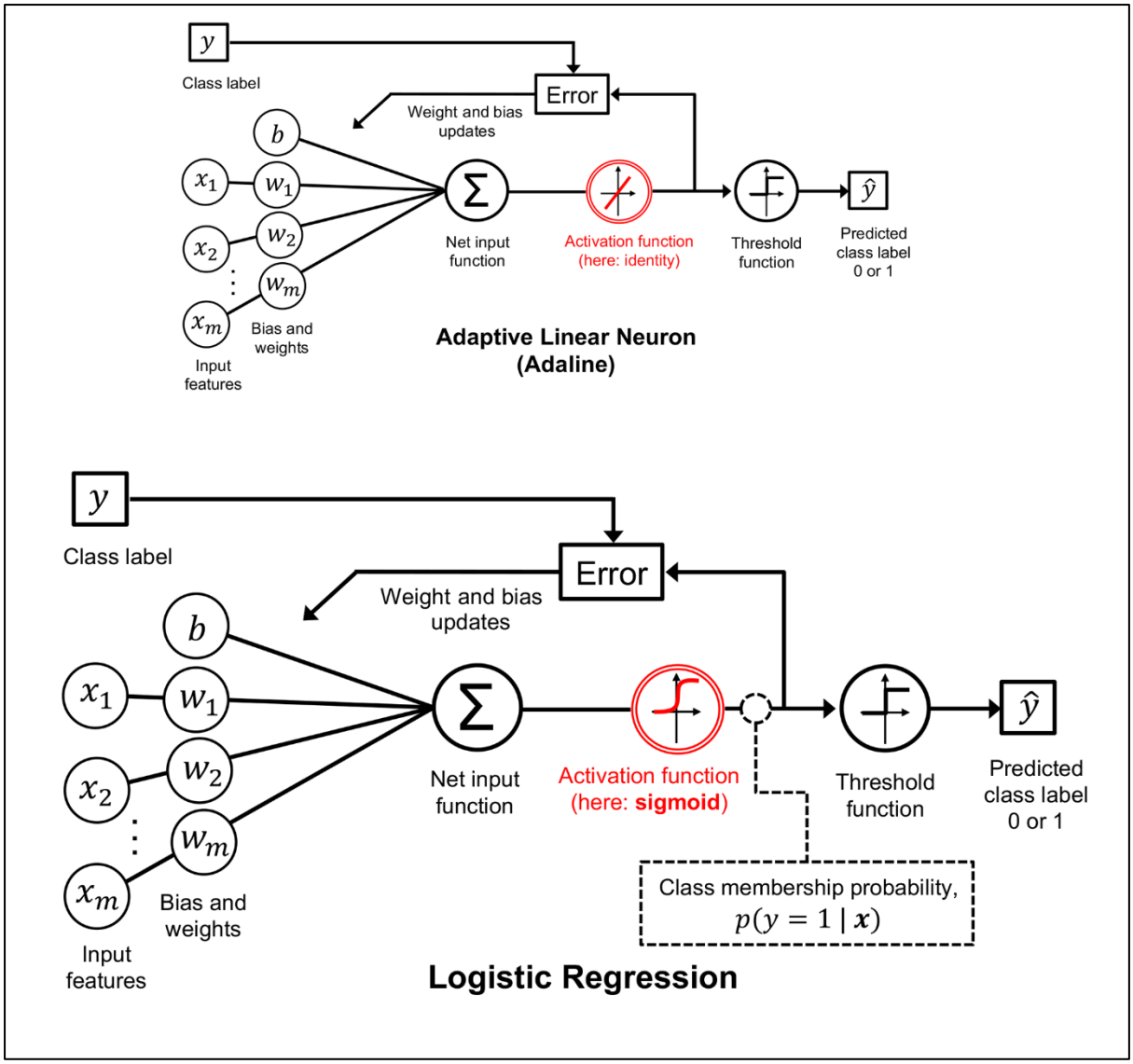

Adaline - Gradient descent
Class 0
Class 1
Petal length [standardized]
Sepal length [standardized]
Mean squared error
Epochs

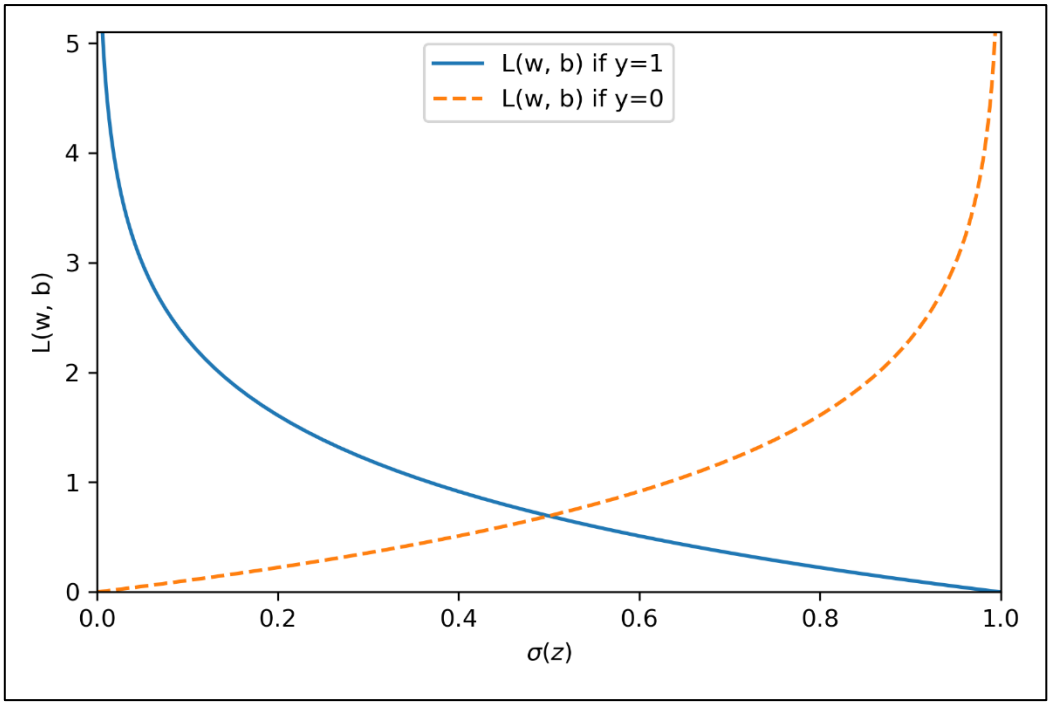

Adaline - Stochastic gradient descent
Class 0
Class 1
Petal length [standardized]
Sepal length [standardized]
Average loss
Epochs

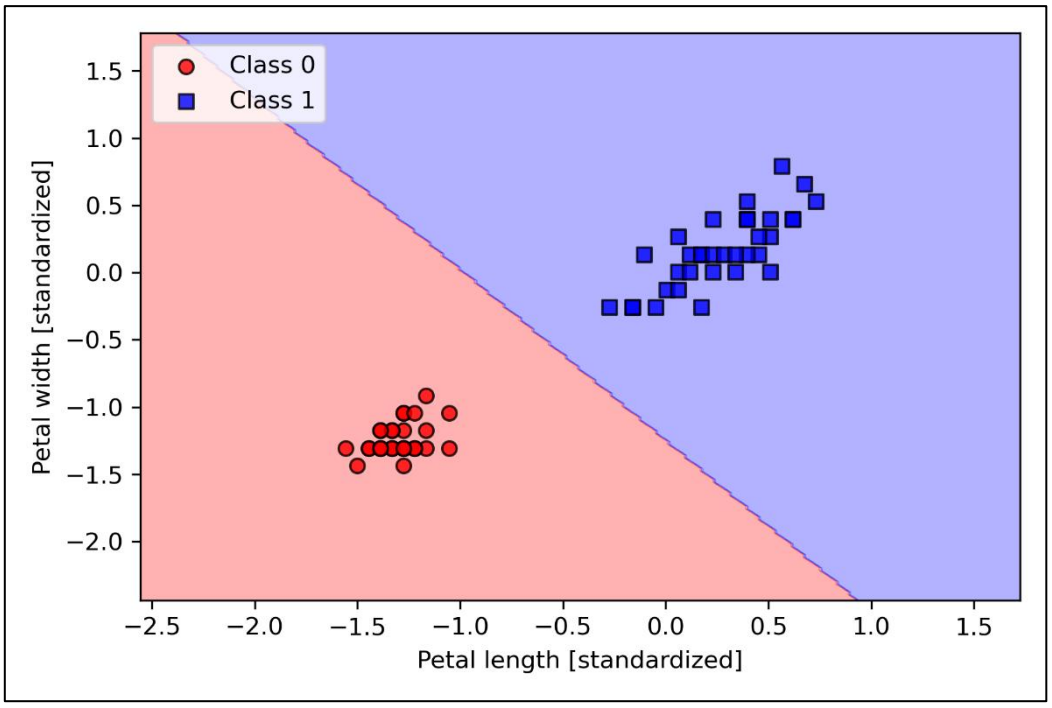
Chapter 3: A Tour of Machine Learning Classifiers Using Scikit-Learn






y
Class label
Error
Weight and bias updates
b
x1
w1
x2
w2
wm
xm
Bias and weights
Input features
Net input function
Activation function (here: identity)
Threshold function
ŷ
Predicted class label 0 or 1
Adaptive Linear Neuron (Adaline)
y
Class label
Error
Weight and bias updates
b
x1
w1
x2
w2
wm
xm
Bias and weights
Input features
Net input function
Activation function (here: sigmoid)
Threshold function
ŷ
Predicted class label 0 or 1
Class membership probability, p(y = 1 | x)
Logistic Regression


L(w, b) if y=1
L(w, b) if y=0
L(w, b)
$\sigma(z)$
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0


Class 0
Class 1
Petal width [standardized]
Petal length [standardized]
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

$$\frac{\partial L}{\partial w_j} = \underbrace{\frac{\partial L}{\partial a}\frac{da}{dz}\frac{\partial z}{\partial w_j}}_{\text{Apply chain rule}} \quad \text{where } a = \sigma(z) = \frac{1}{1+e^{-z}}$$

1) Derive terms separately:

$$\frac{\partial L}{\partial a} = \frac{a-y}{a\cdot(1-a)}$$

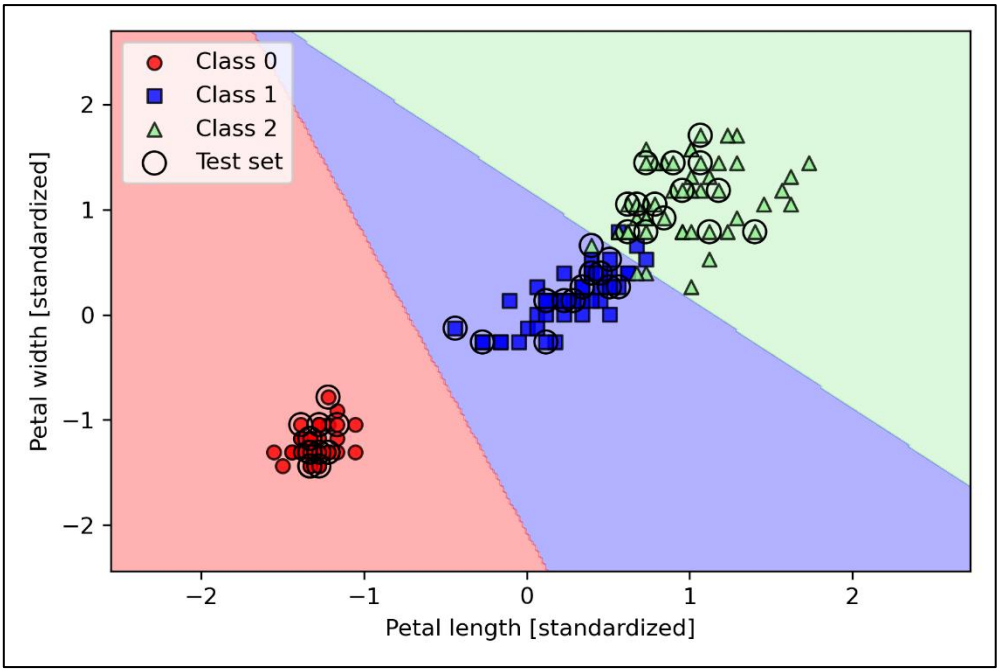
$$\frac{da}{dz} = \frac{e^{-z}}{(1+e^{-z})^2} = a\cdot(1-a)$$

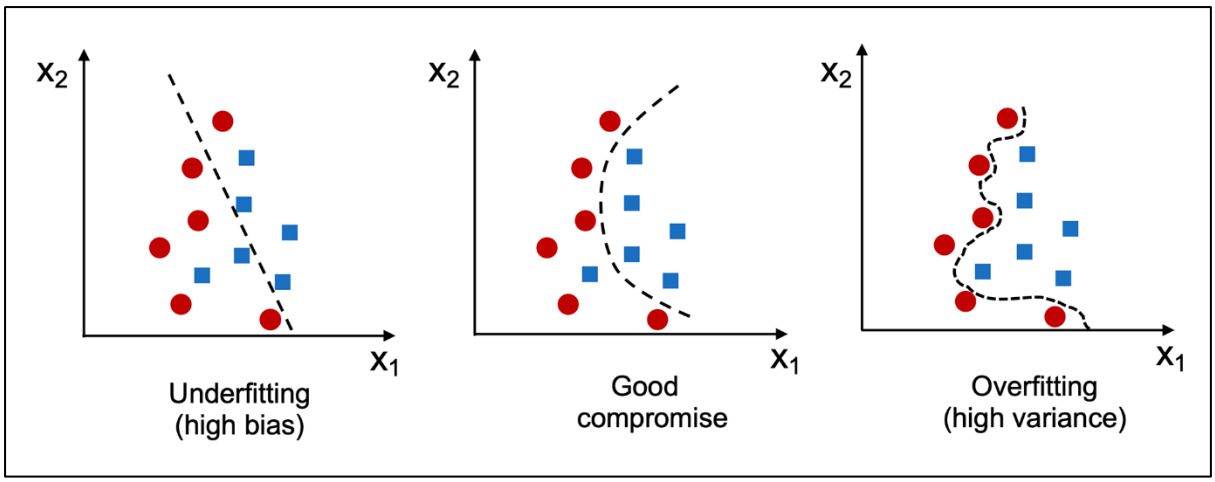
$$\frac{\partial z}{w_j} = x_j$$

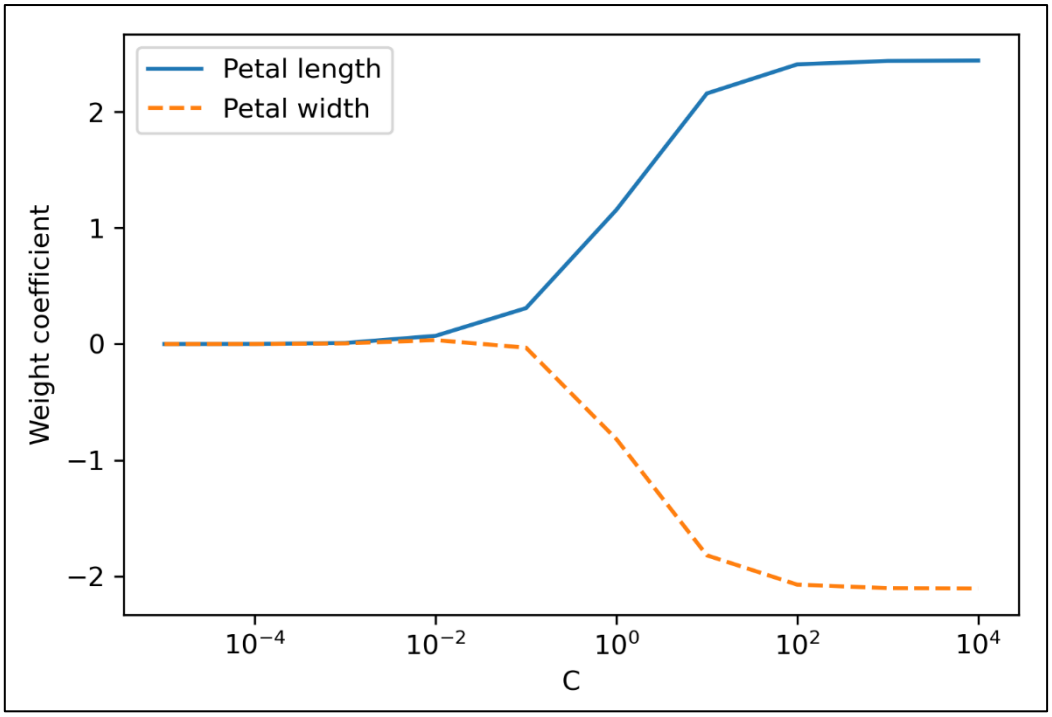
2) Combine via chain rule and simplify:

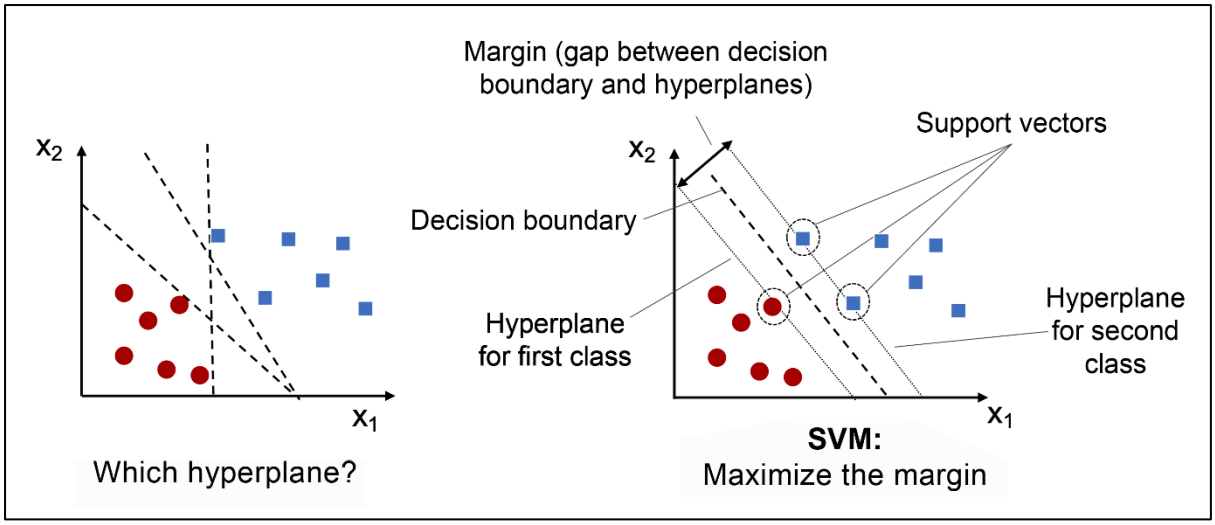
$$\frac{\partial L}{\partial z} = a - y$$

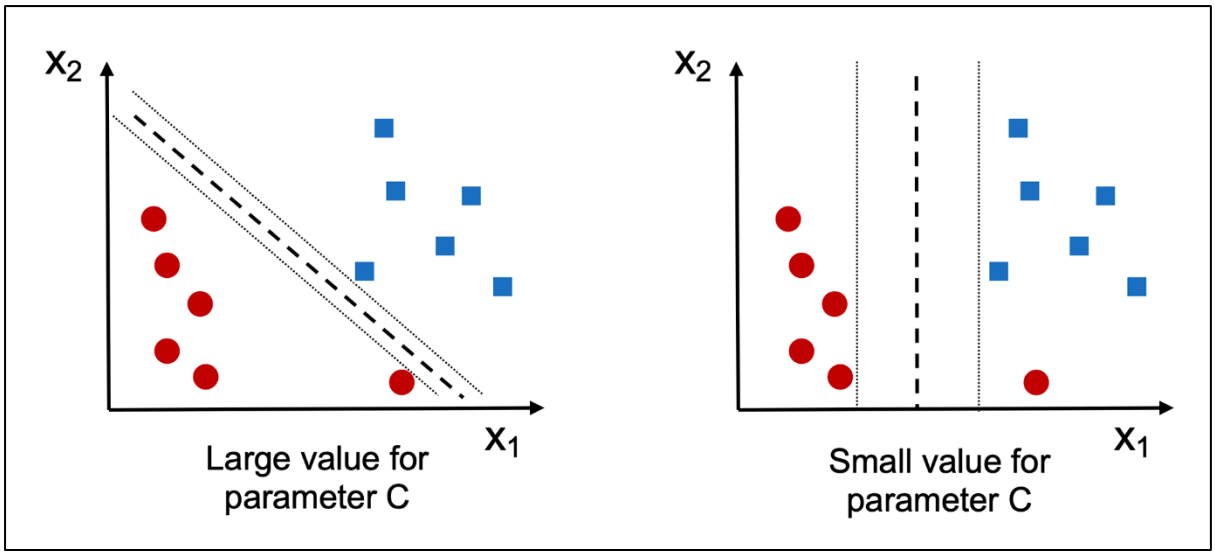
$$\frac{\partial L}{\partial w_j} = (a-y)x_j = -(y-a)x_j$$

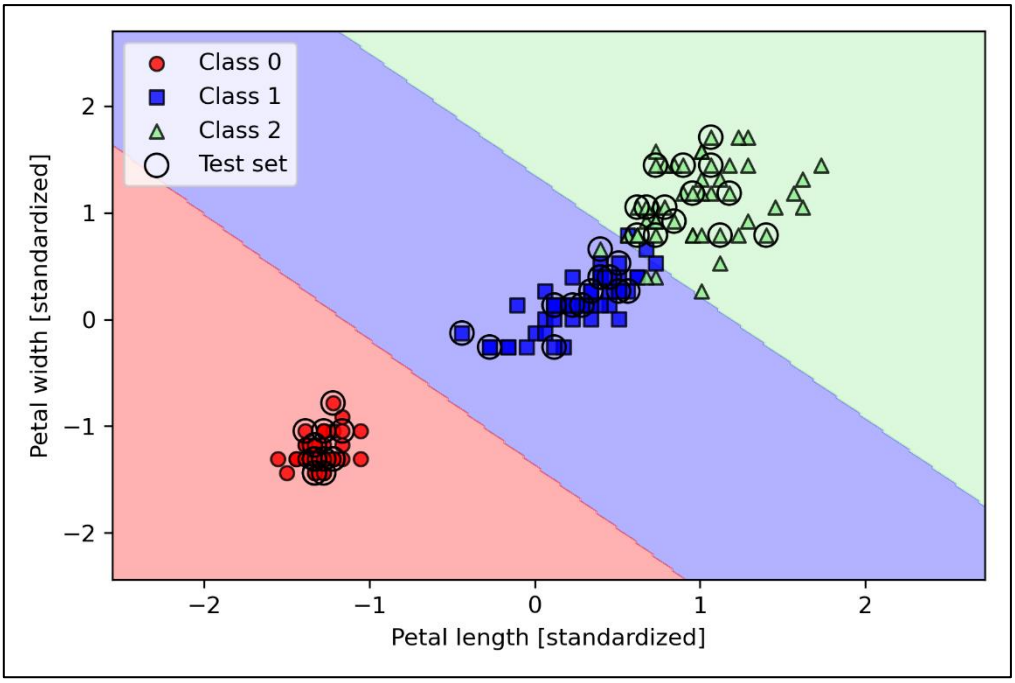


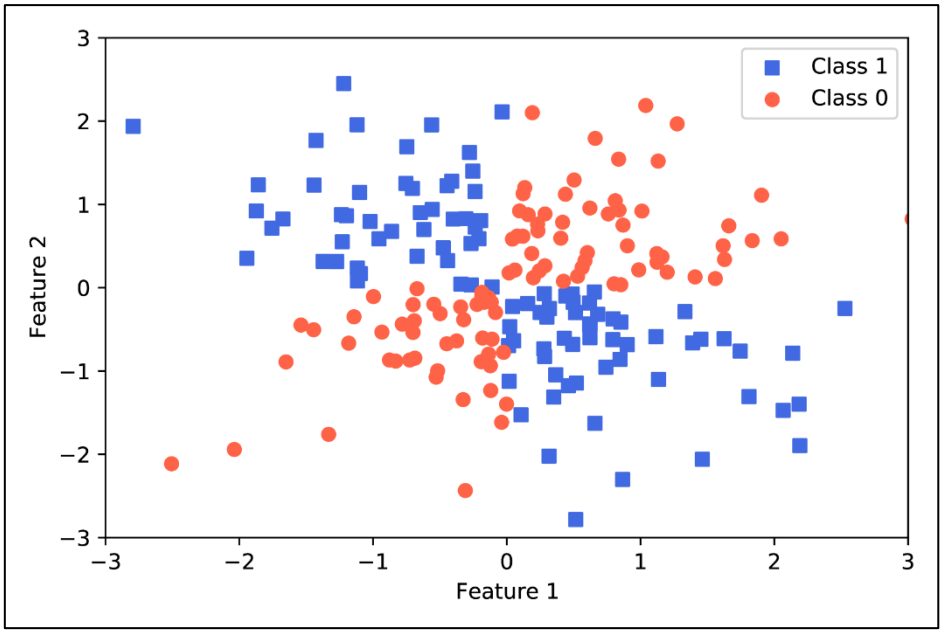

x2
x1
Underfitting
(high bias)
x2
x1
Good
compromise
x2
x1
Overfitting
(high variance)

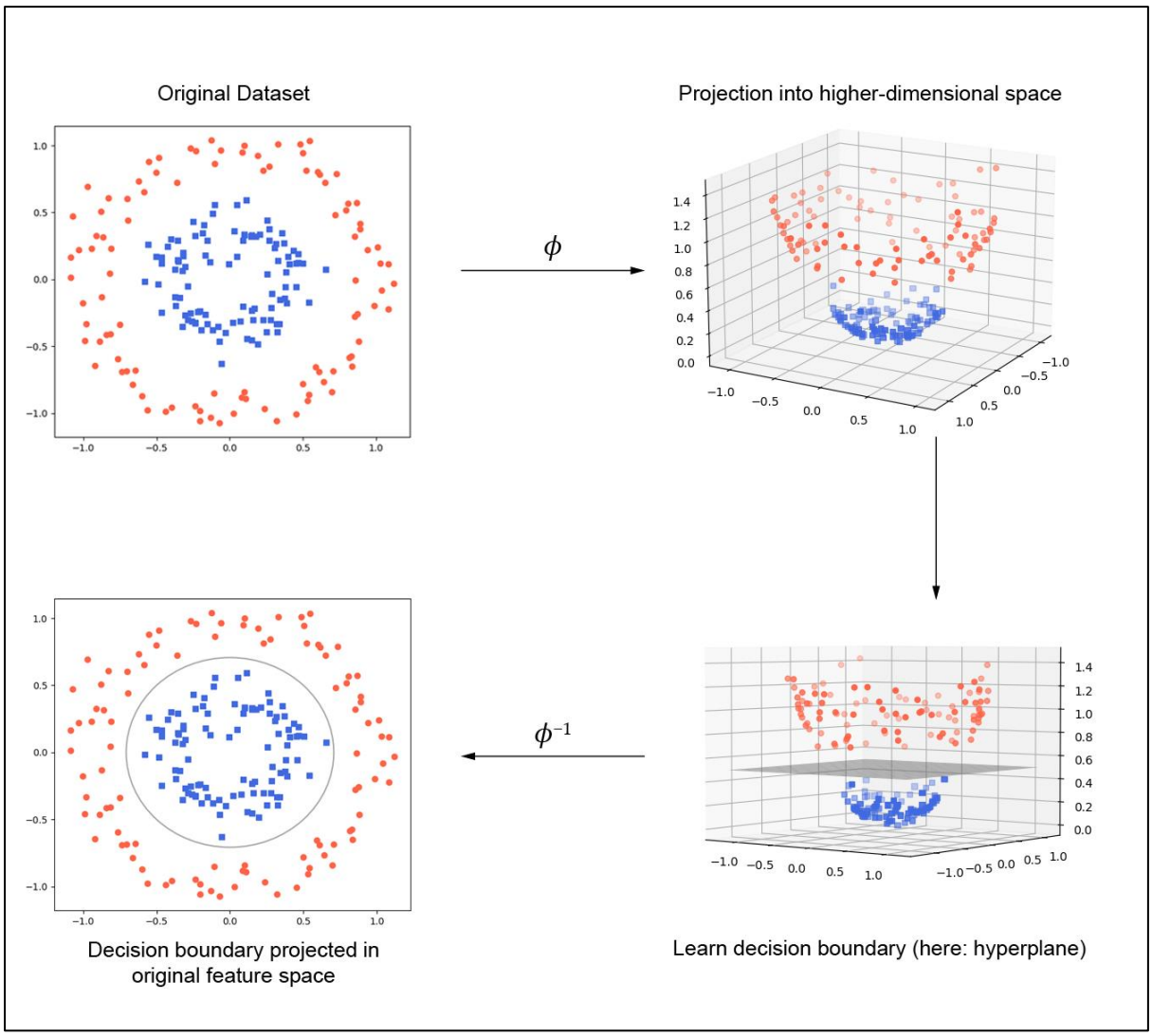

Petal length
Petal width
Weight coefficient
2
1
0
−1
−2
10−4
10−2
100
102
104
C

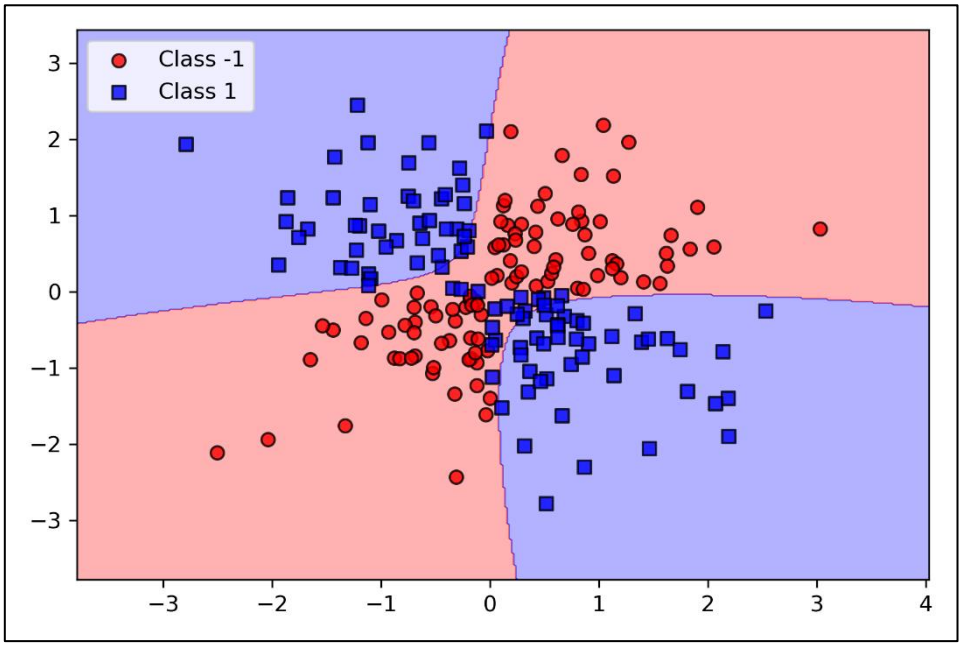

Margin (gap between decision boundary and hyperplanes)
Support vectors
x_2
x_1
Decision boundary
Hyperplane for first class
Hyperplane for second class
Which hyperplane?
SVM:
Maximize the margin

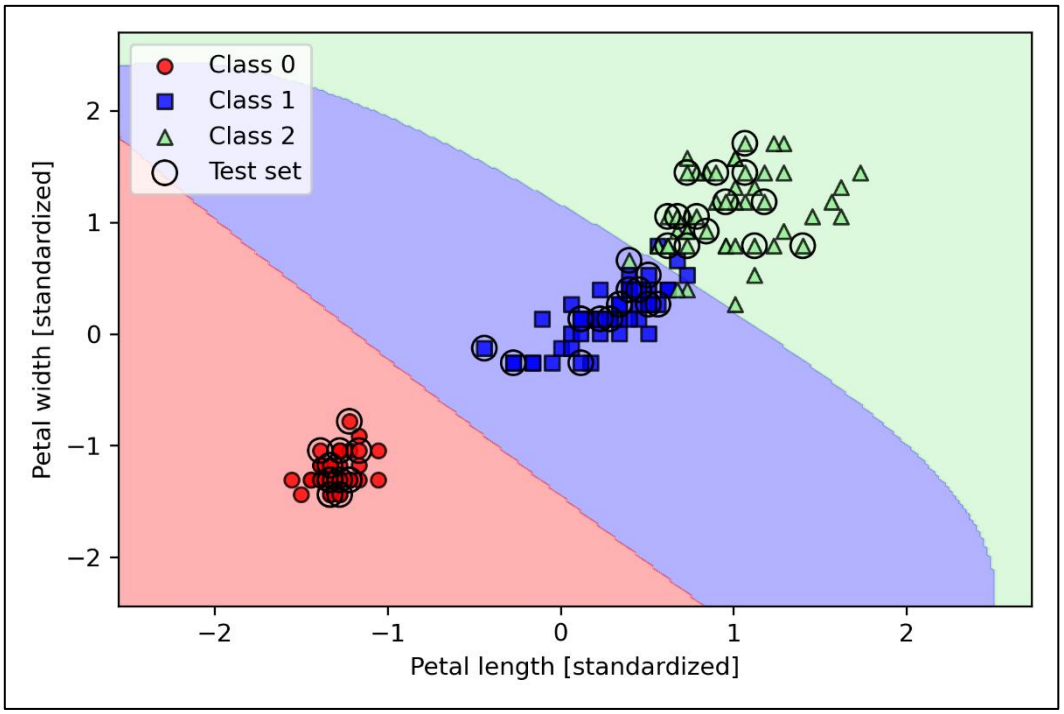

x_2
x_1
Large value for parameter C
x_2
x_1
Small value for parameter C

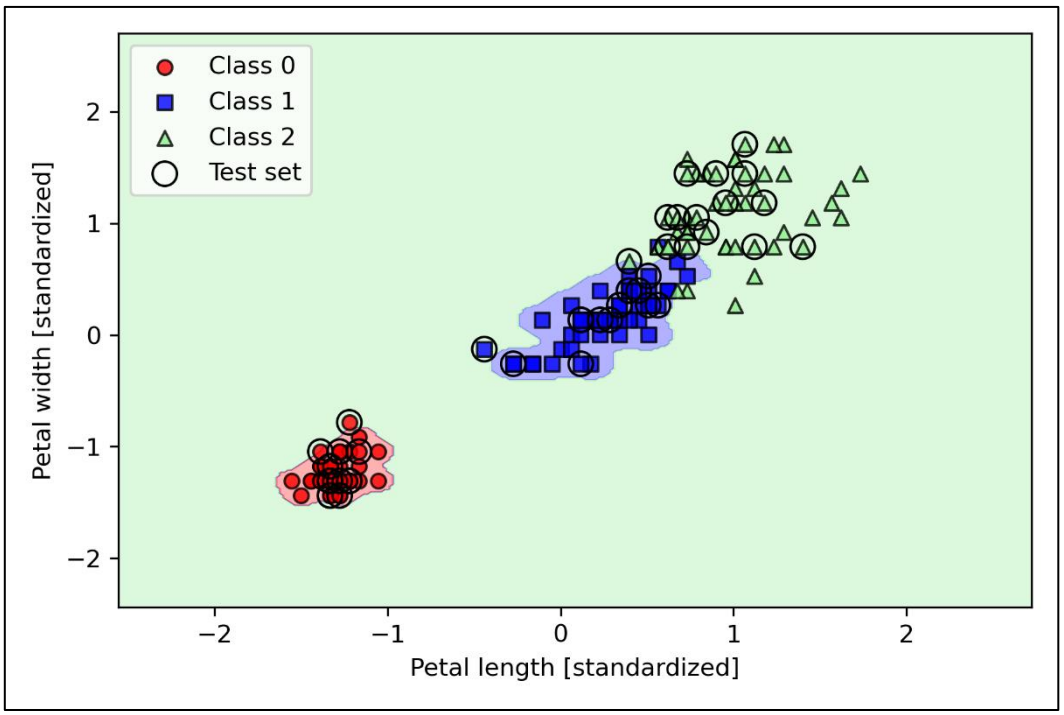

Class 0
Class 1
Class 2
Test set
Petal width [standardized]
Petal length [standardized]
2
1
0
−1
−2
−2
−1
0
1
2

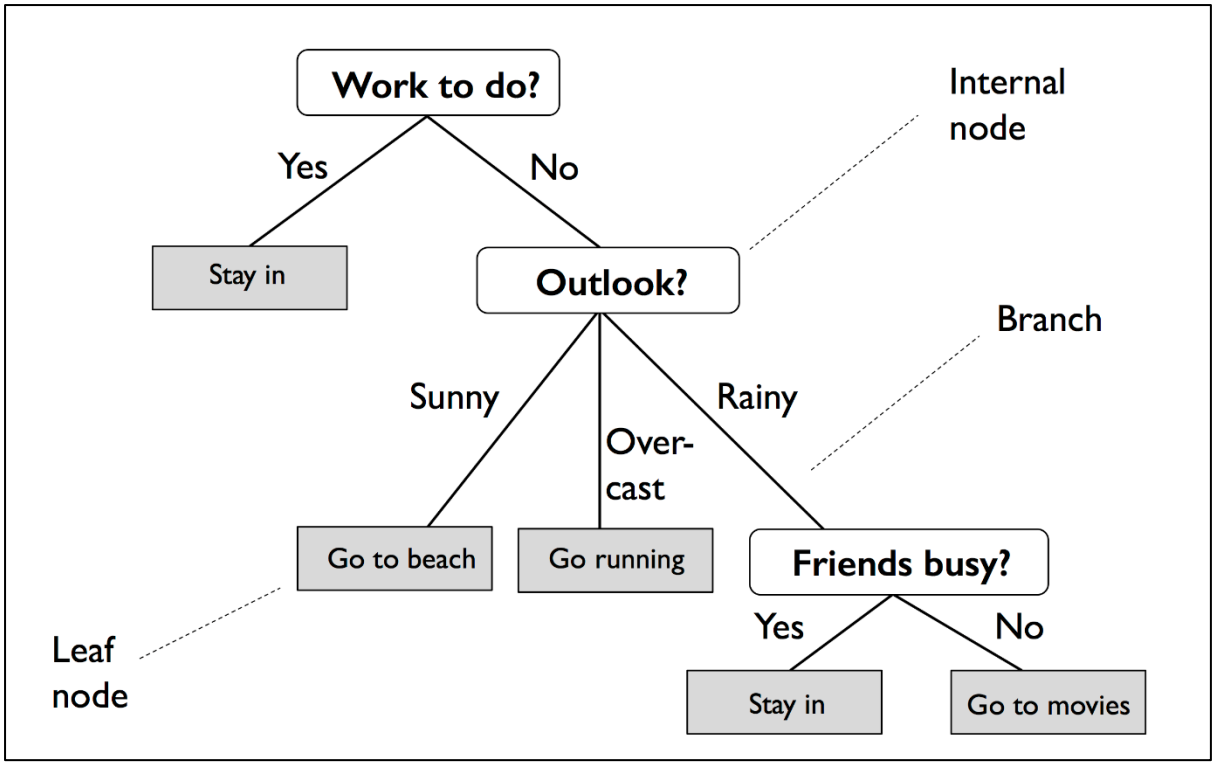

Class 1
Class 0
Feature 2
Feature 1
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

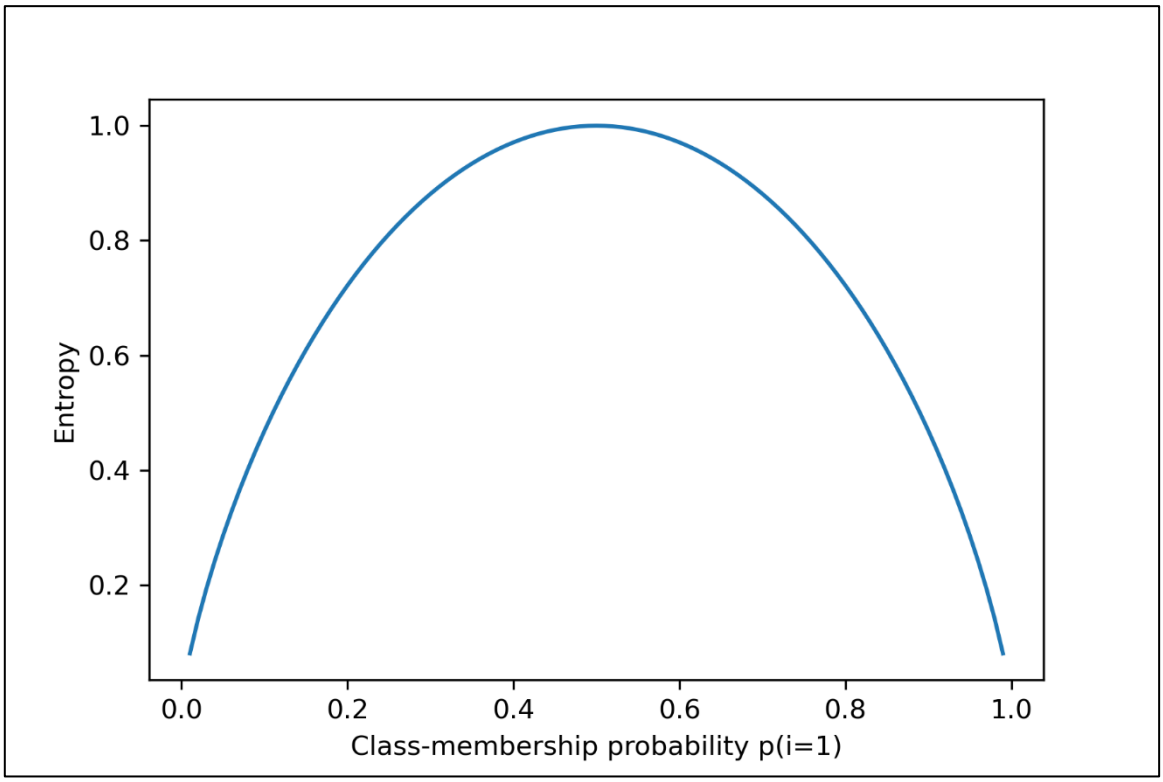

Original Dataset
Projection into higher-dimensional space
ϕ
ϕ^{-1}
Decision boundary projected in original feature space
Learn decision boundary (here: hyperplane)

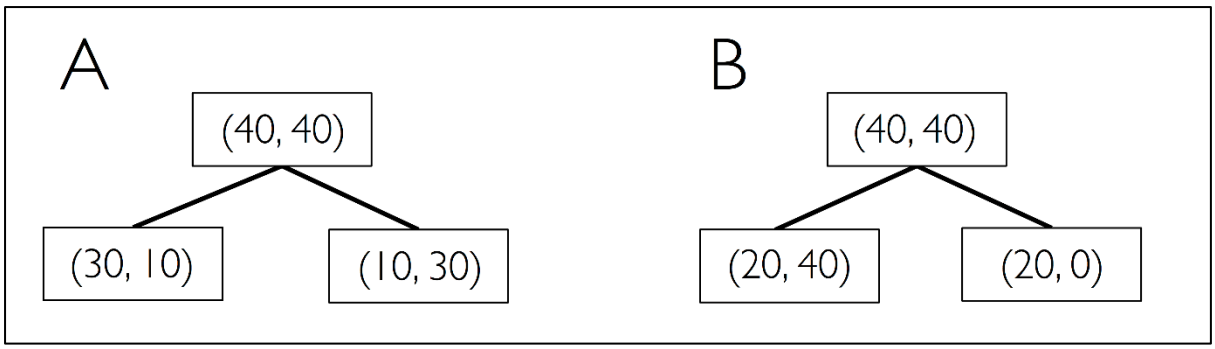

Class -1
Class 1

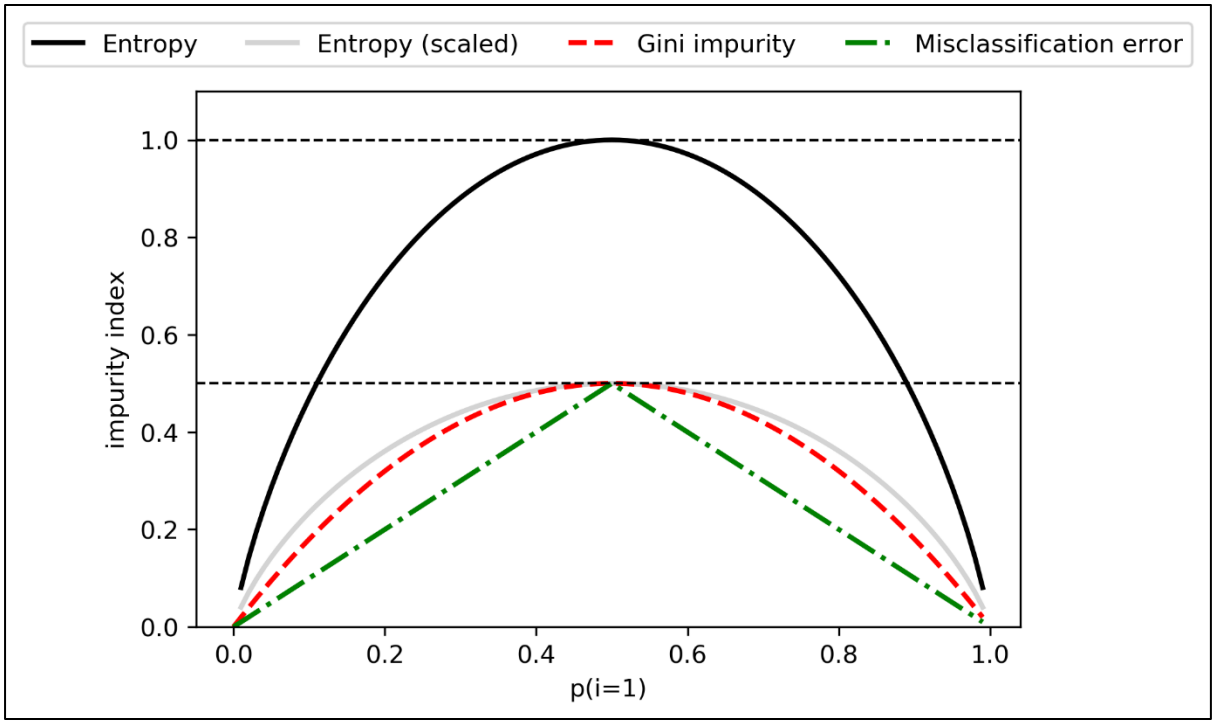

Class 0
Class 1
Class 2
Test set
Petal width [standardized]
Petal length [standardized]
2
1
0
−1
−2
−2
−1
0
1
2

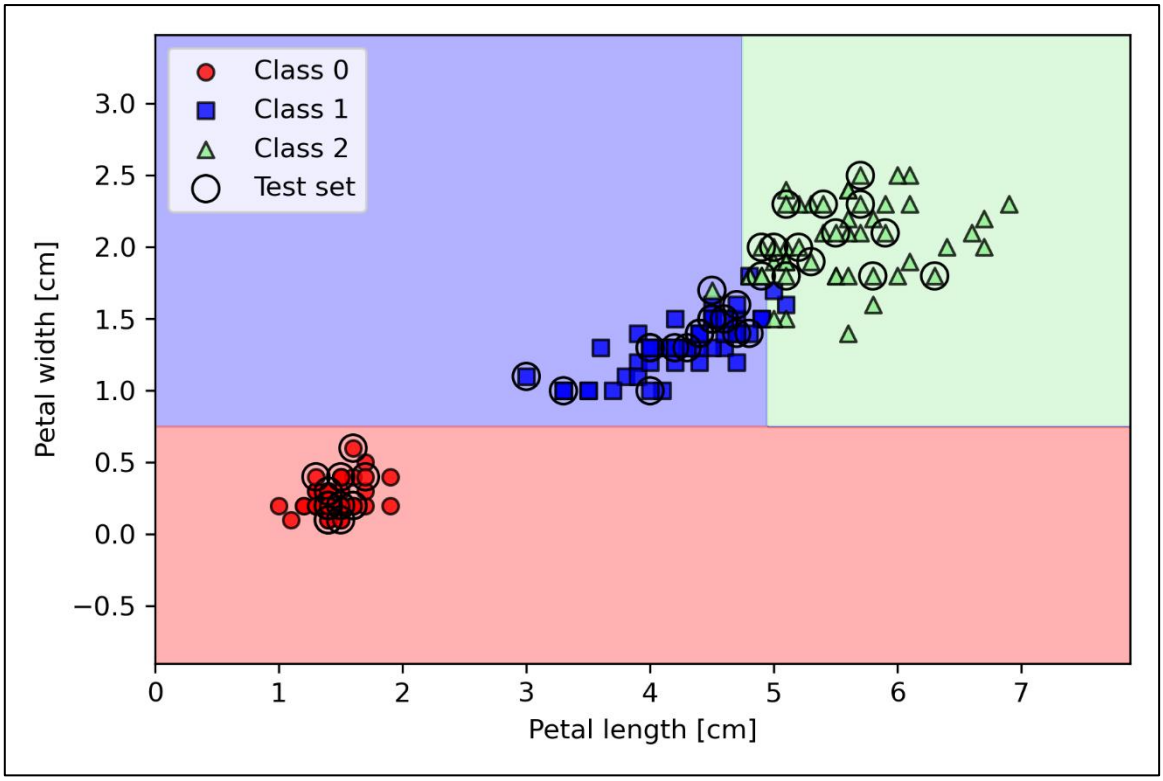

Class 0
Class 1
Class 2
Test set
Petal width [standardized]
Petal length [standardized]
2
1
0
−1
−2
−2
−1
0
1
2

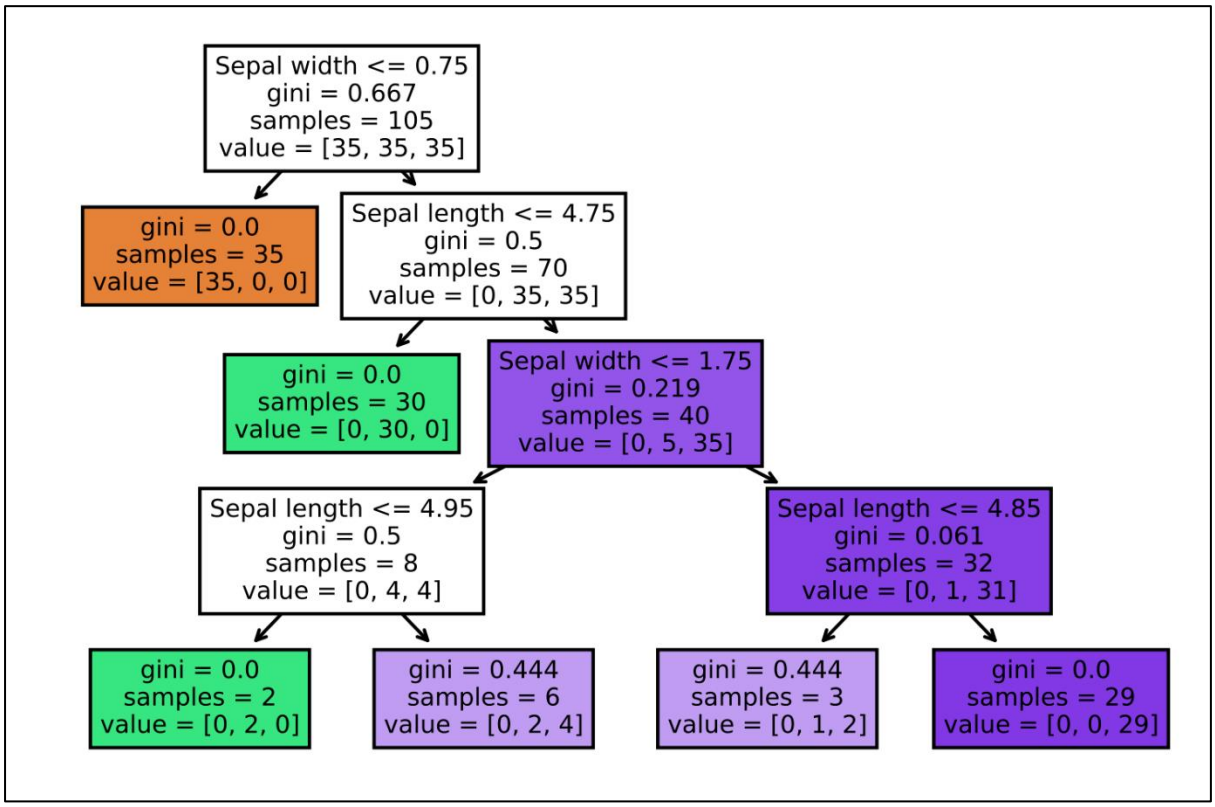

Work to do?
Yes
No
Stay in
Outlook?
Internal node
Branch
Sunny
Over-cast
Rainy
Go to beach
Go running
Friends busy?
Yes
No
Stay in
Go to movies
Leaf node

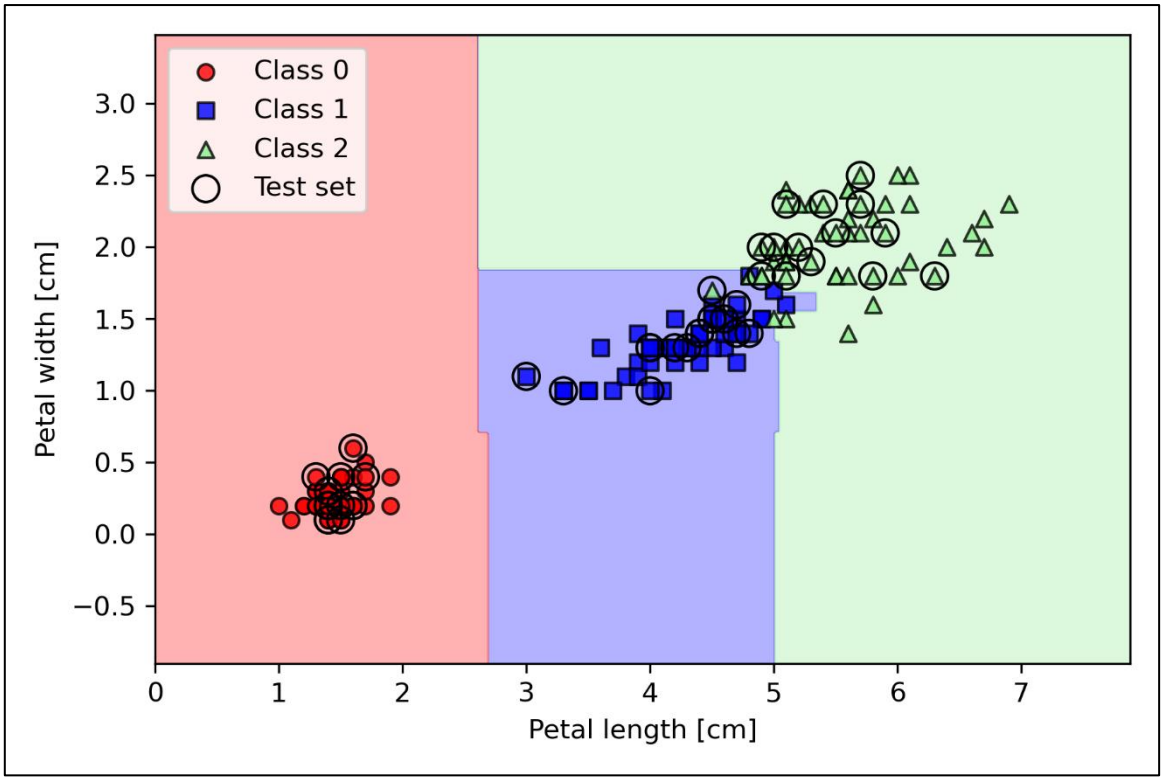

1.0
0.8
0.6
0.4
0.2
Entropy
0.0
0.2
0.4
0.6
0.8
1.0
Class-membership probability p(i=1)

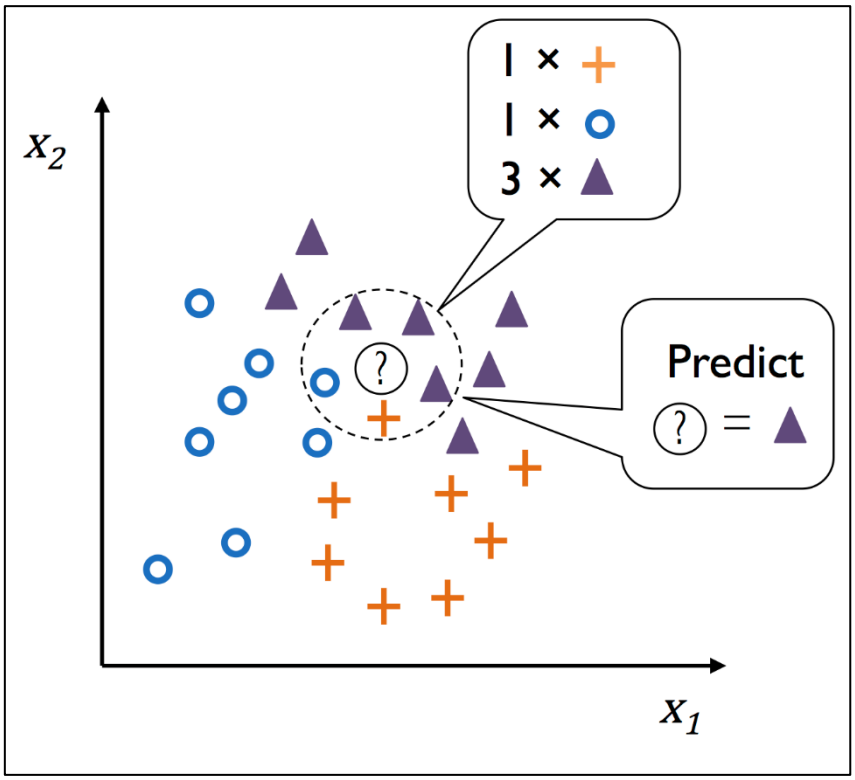

A
(40, 40)
(30, 10)
(10, 30)
B
(40, 40)
(20, 40)
(20, 0)

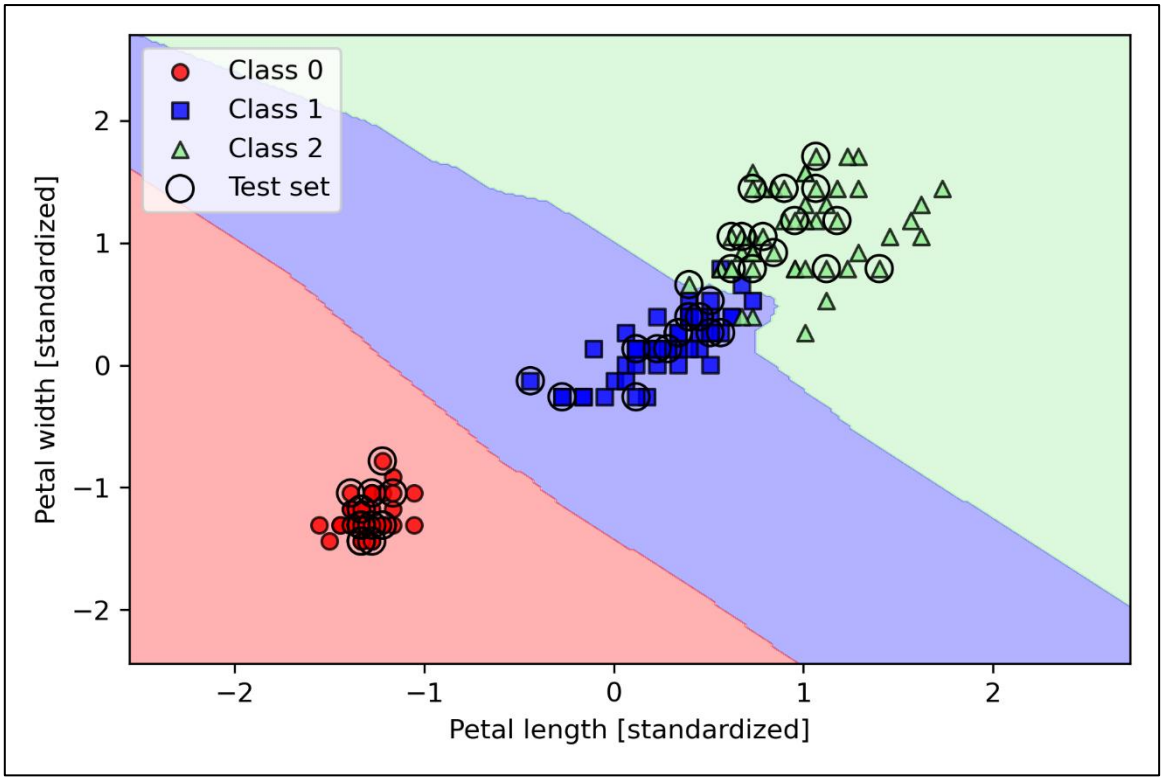

Entropy
Entropy (scaled)
Gini impurity
Misclassification error
impurity index
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
p(i=1)


Class 0
Class 1
Class 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


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

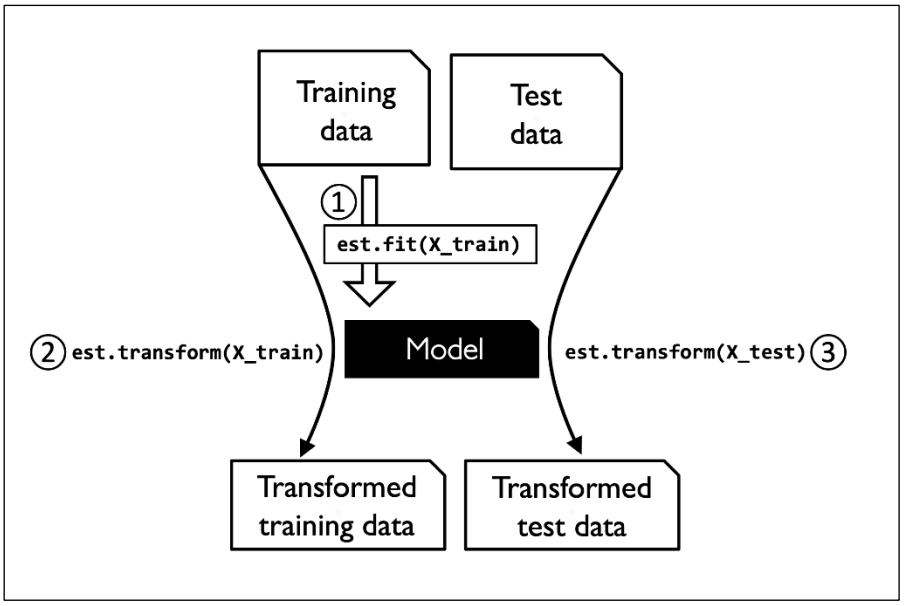

Class 0
Class 1
Class 2
Test set
Petal width [cm]
Petal length [cm]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
1
2
3
4
5
6
7

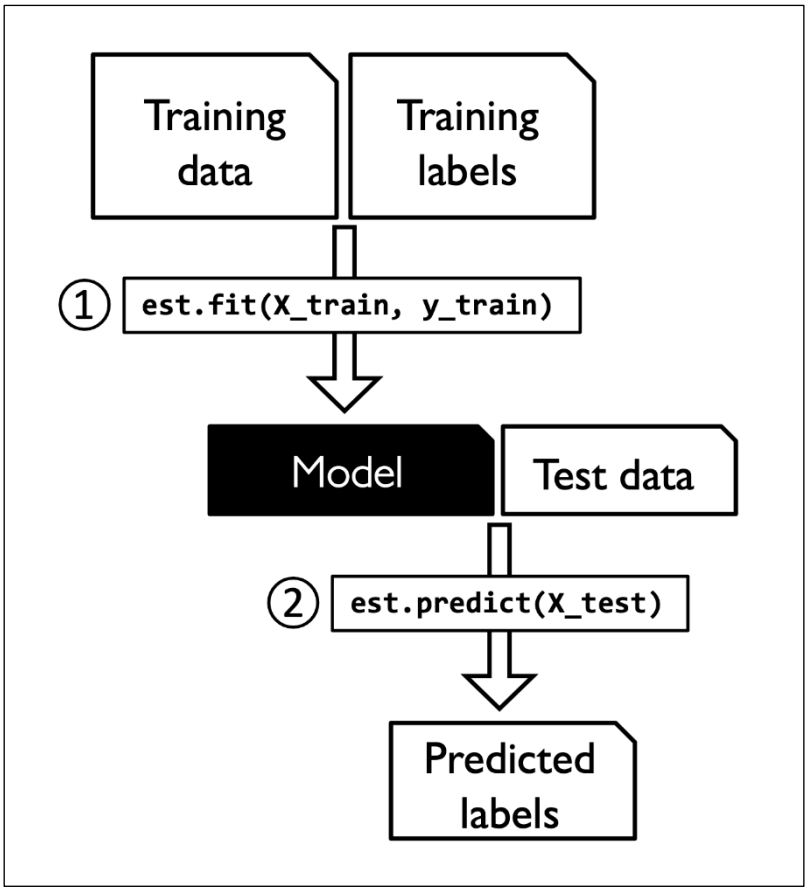

1 ×
1 ×
3 ×
Predict
?
=
x_2
x_1


Class 0
Class 1
Class 2
Test set
Petal width [standardized]
Petal length [standardized]
2
1
0
−1
−2
−2
−1
0
1
2

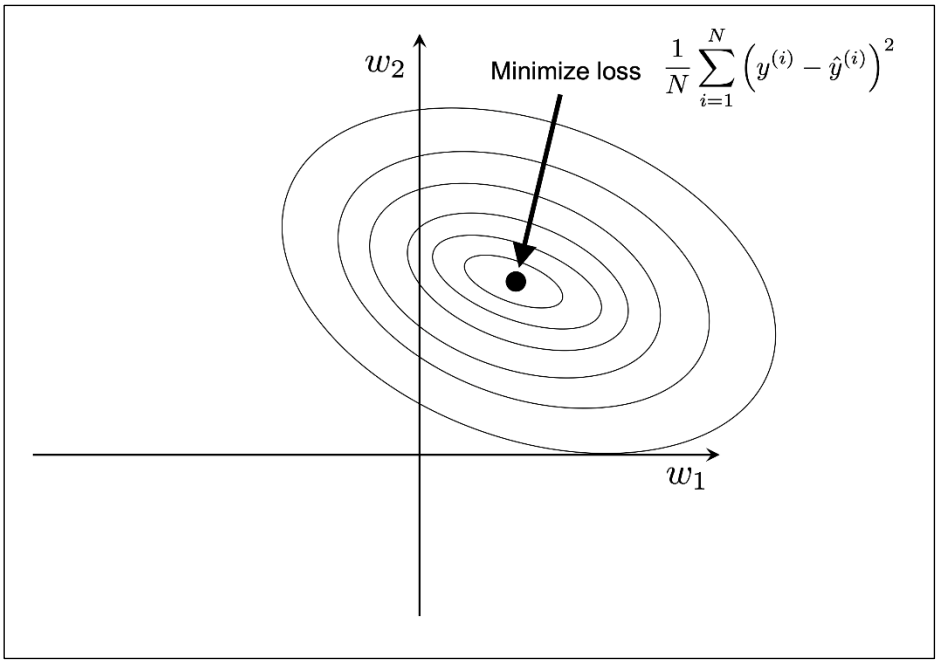
Chapter 4: Building Good Training Datasets – Data Preprocessing

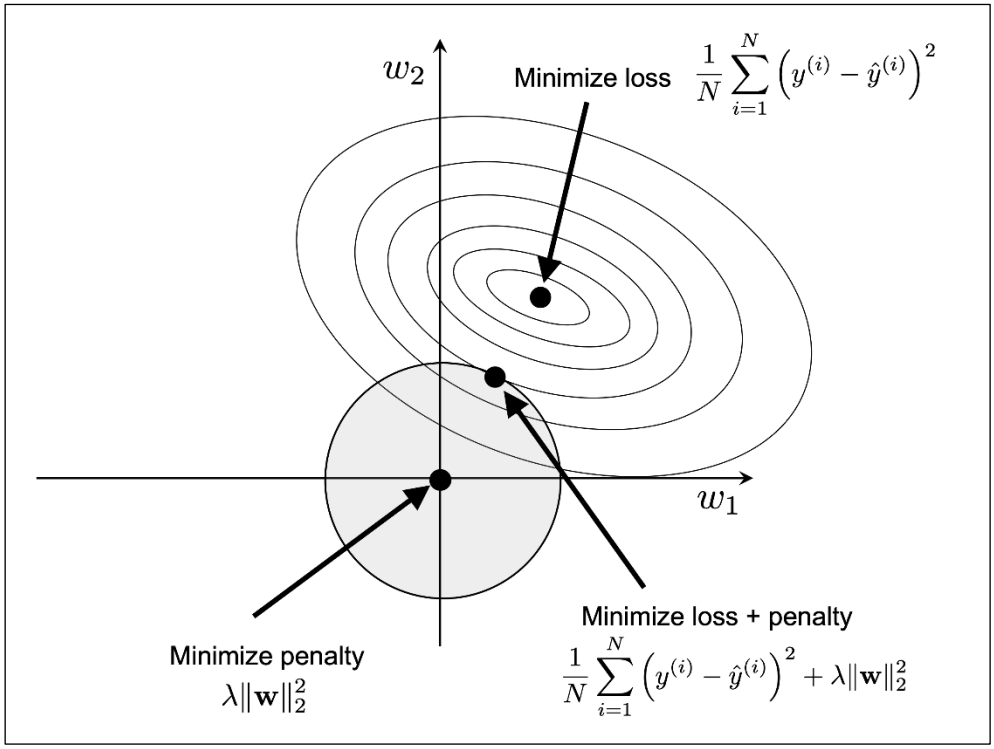
	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	7.5	8.0
2	10.0	11.0	12.0	6.0

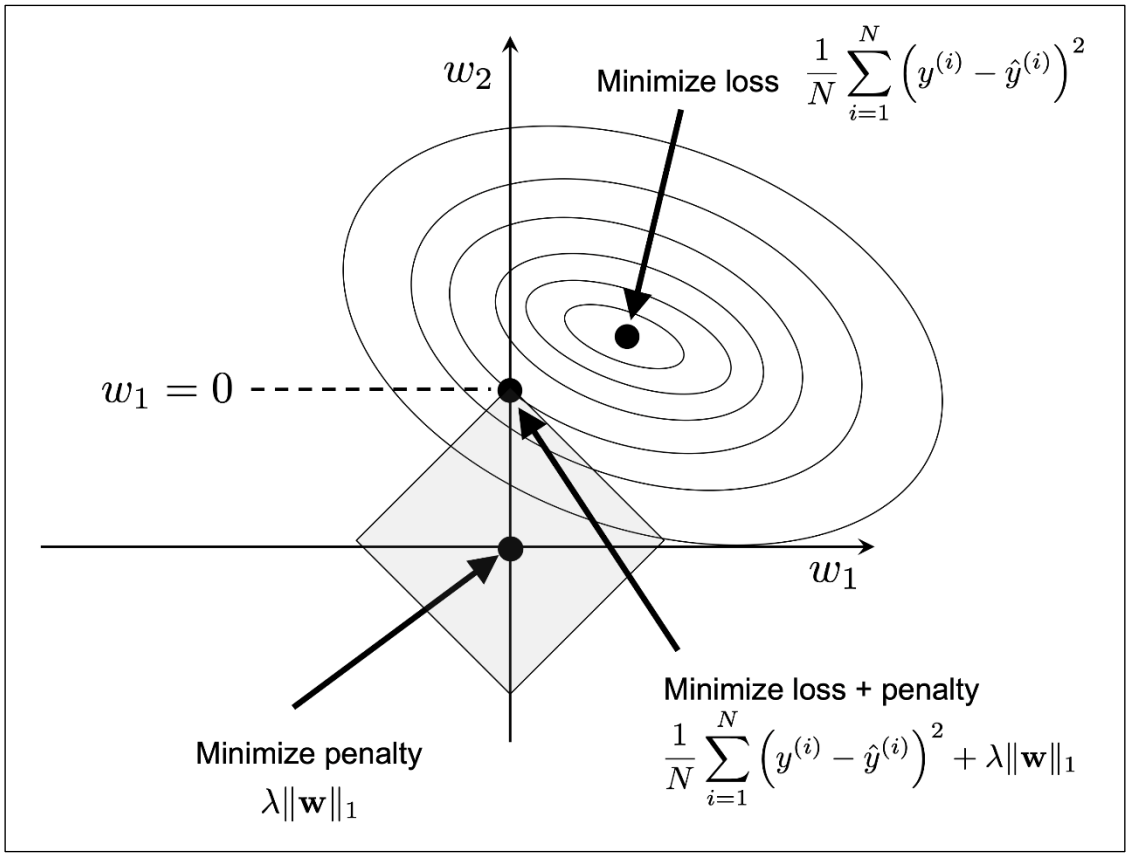


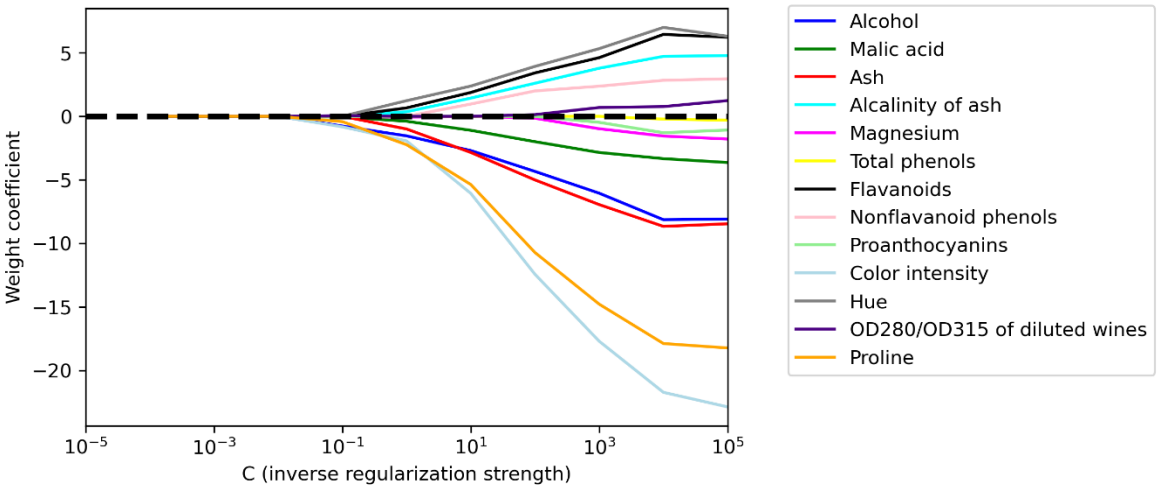


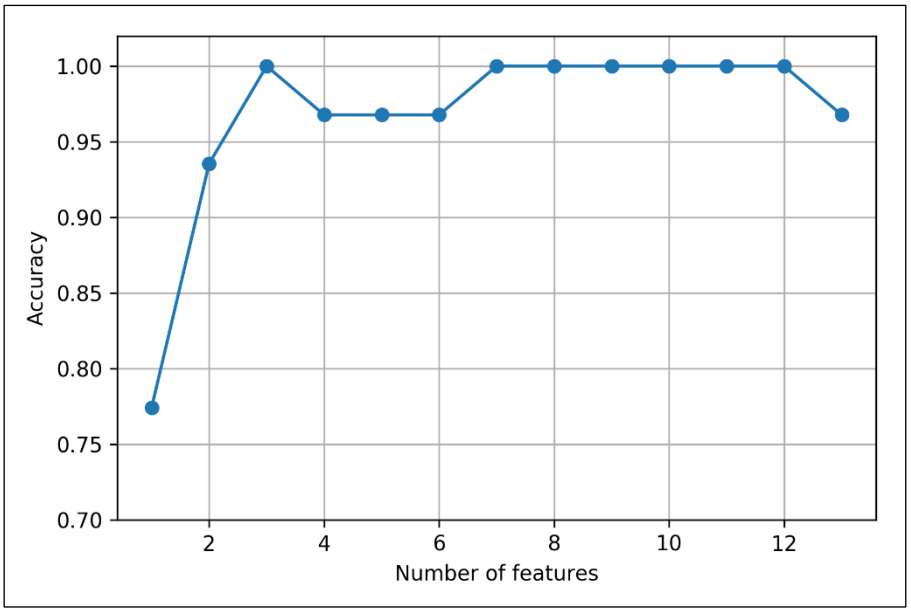
	Class label	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38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

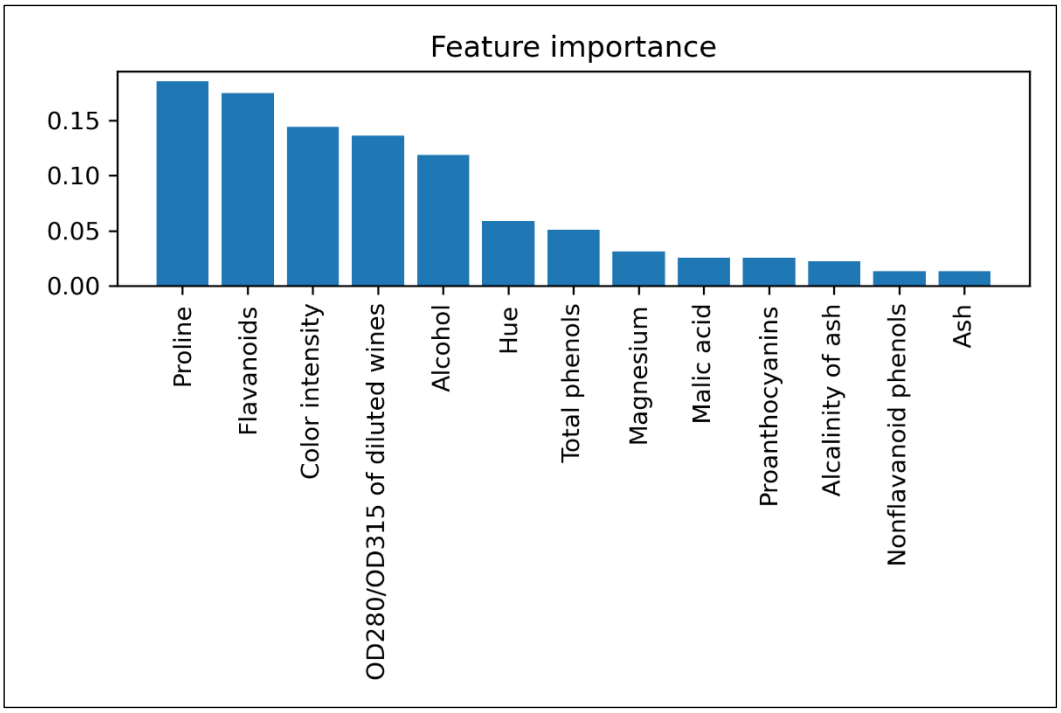

w_2
Minimize loss
$\frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)}-\hat{y}^{(i)}\right)^2$
w_1


w_2
Minimize loss
$\frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)}-\hat{y}^{(i)}\right)^2$
w_1
Minimize penalty
$\lambda\|\mathbf{w}\|_2^2$
Minimize loss + penalty
$\frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)}-\hat{y}^{(i)}\right)^2+\lambda\|\mathbf{w}\|_2^2$

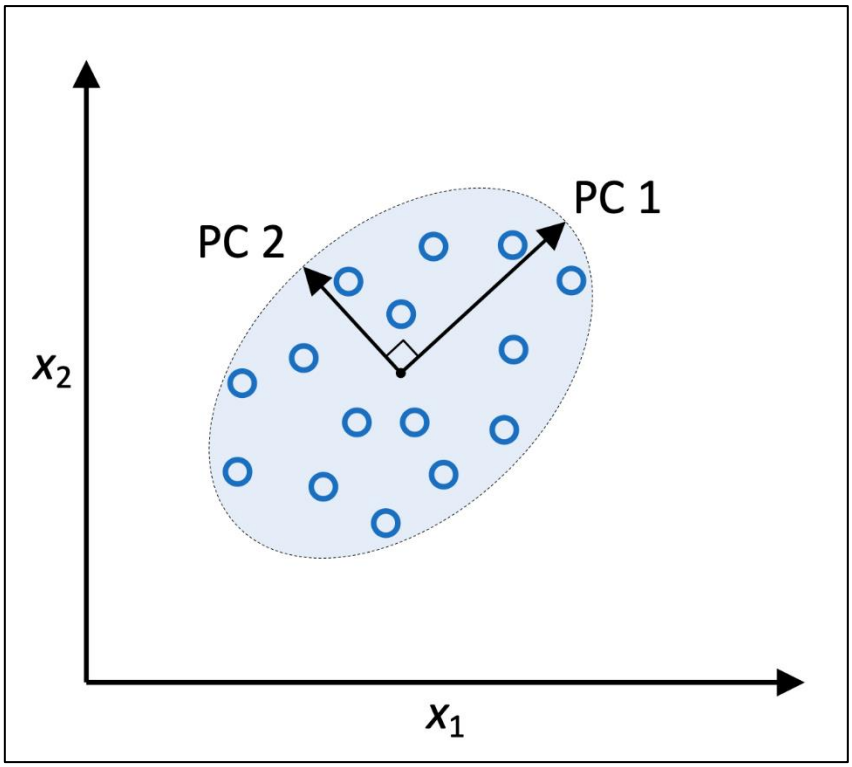

w_2
Minimize loss
$\frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)}-\hat{y}^{(i)}\right)^2$
$w_1 = 0$
w_1
Minimize loss + penalty
$\frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)}-\hat{y}^{(i)}\right)^2 + \lambda\|\mathbf{w}\|_1$
Minimize penalty
$\lambda\|\mathbf{w}\|_1$

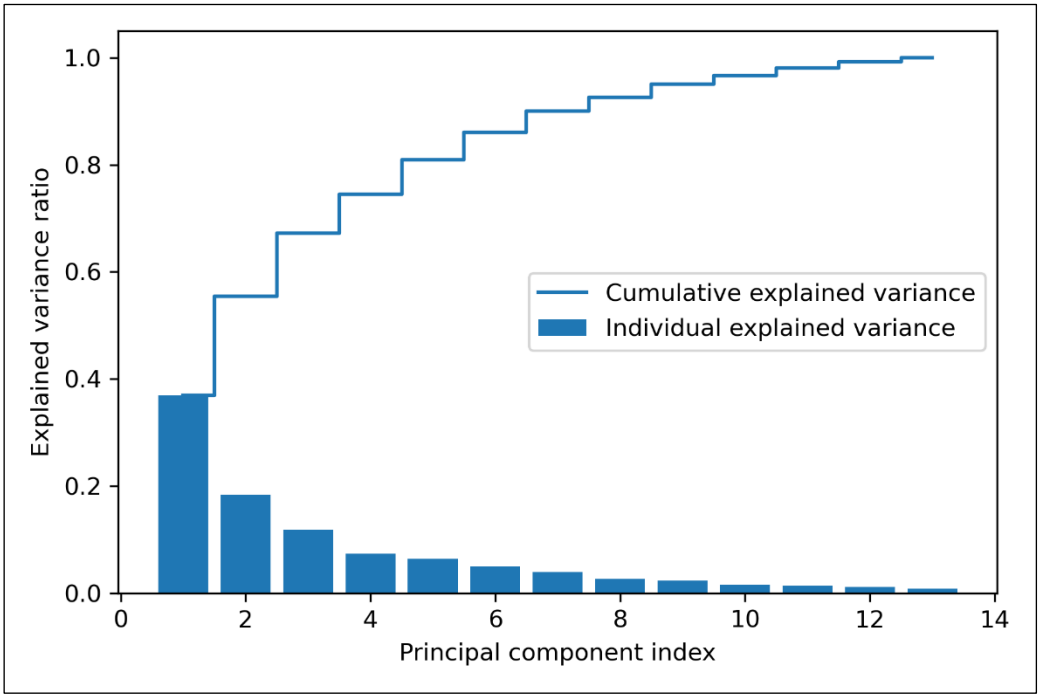

Weight coefficient
5
0
−5
−10
−15
−20
10^{-5}
10^{-3}
10^{-1}
10^{1}
10^{3}
10^{5}
C (inverse regularization strength)
Alcohol
Malic acid
Ash
Alcalinity of ash
Magnesium
Total phenols
Flavanoids
Nonflavanoid phenols
Proanthocyanins
Color intensity
Hue
OD280/OD315 of diluted wines
Proline

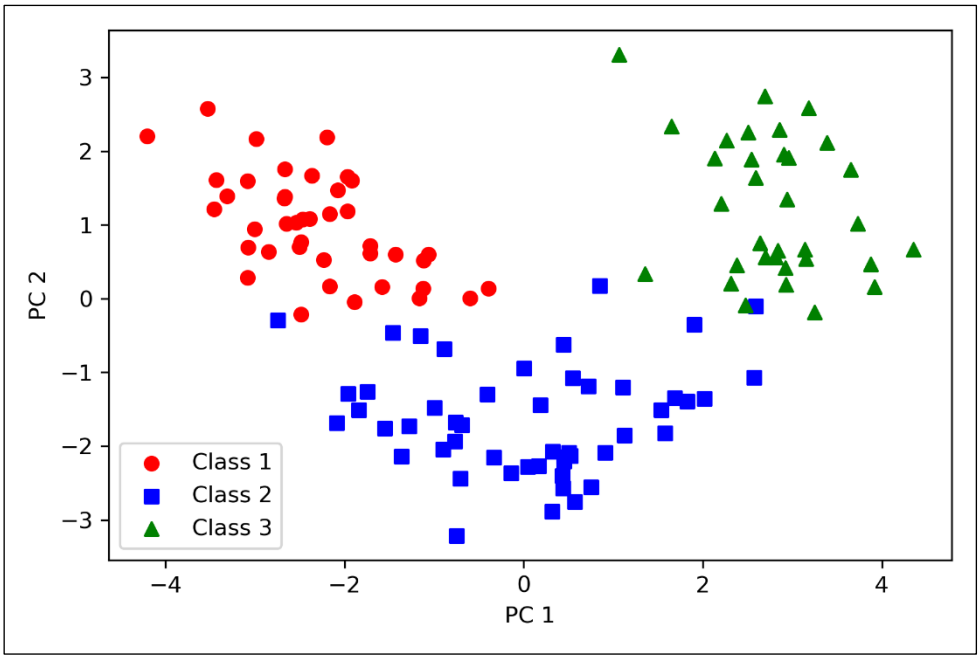

Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
0.70
2
4
6
8
10
12
Number of features

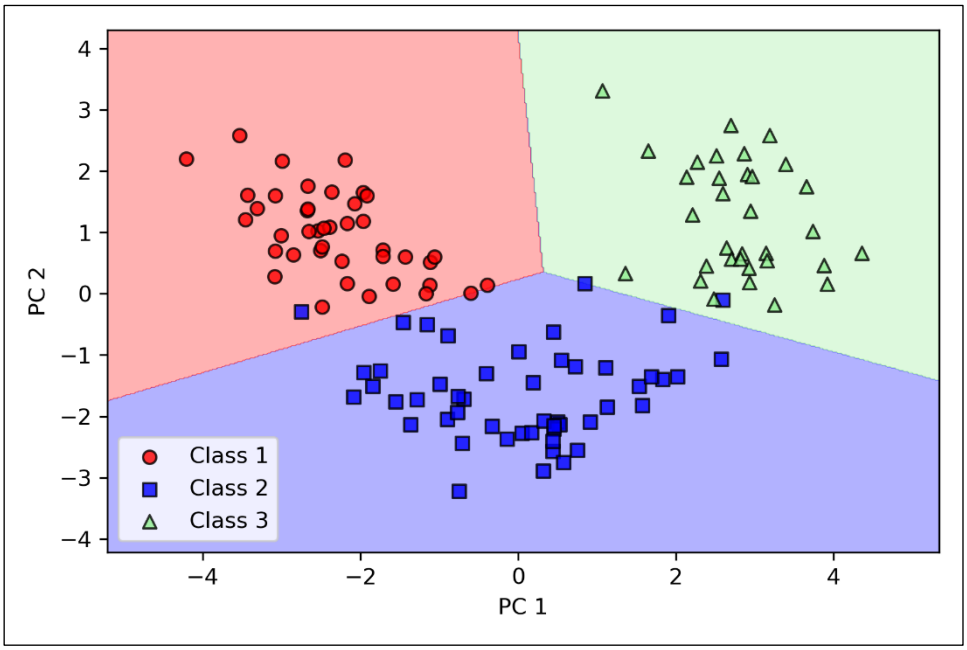

Feature importance
0.15
0.10
0.05
0.00
Proline
Flavanoids
Color intensity
OD280/OD315 of diluted wines
Alcohol
Hue
Total phenols
Magnesium
Malic acid
Proanthocyanins
Alcalinity of ash
Nonflavanoid phenols
Ash

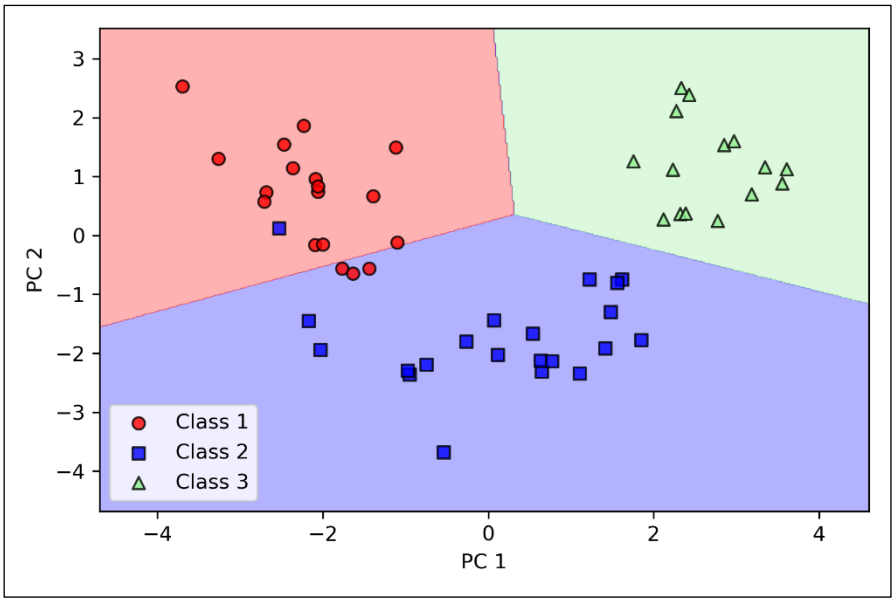
Chapter 5: Compressing Data via Dimensionality Reduction

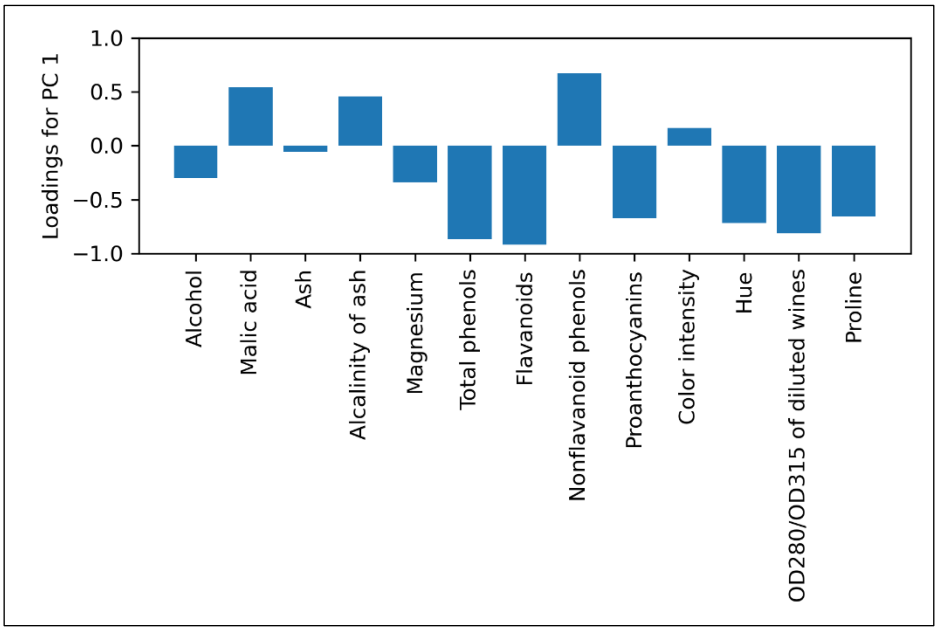


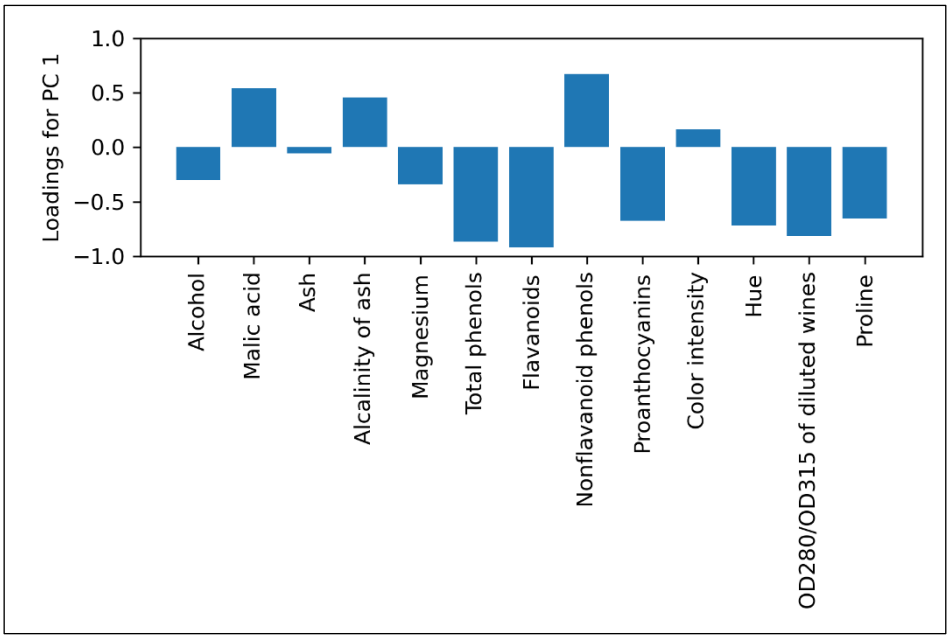


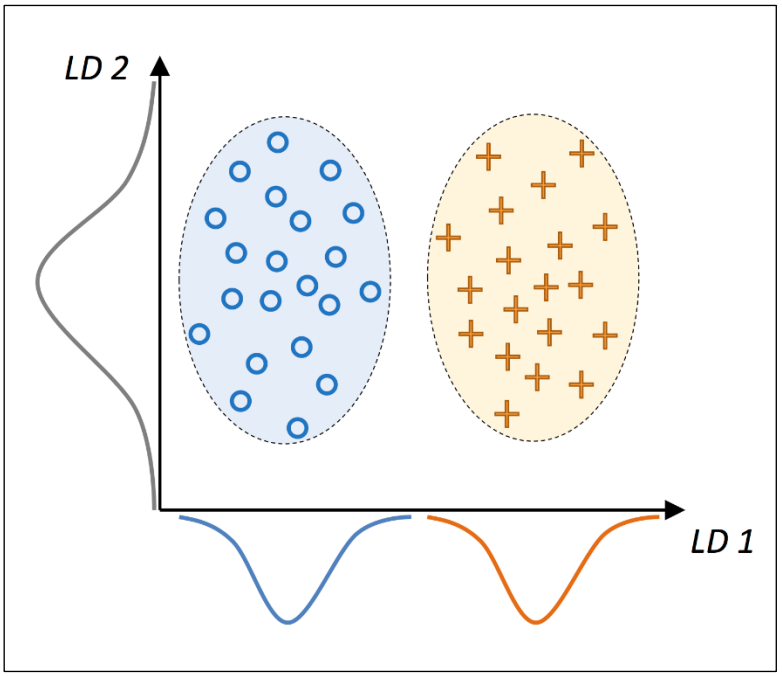

PC 2
PC 1
Class 1
Class 2
Class 3

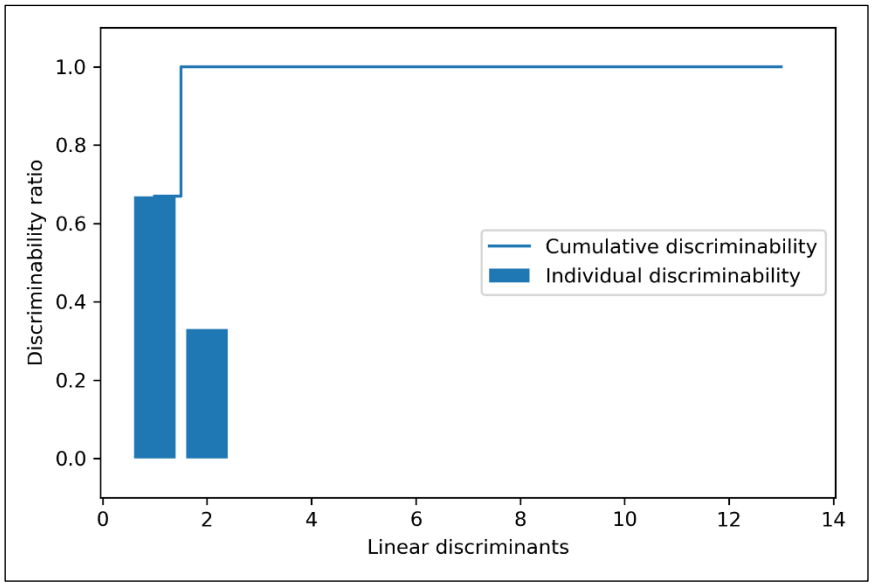

PC 2
PC 1
Class 1
Class 2
Class 3

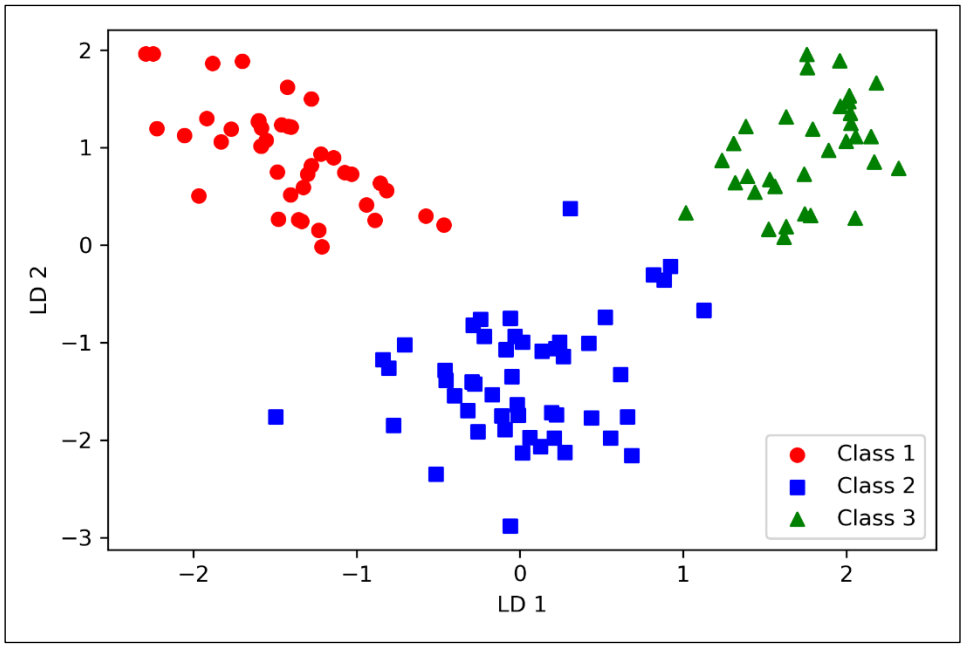

PC 2
PC 1
Class 1
Class 2
Class 3

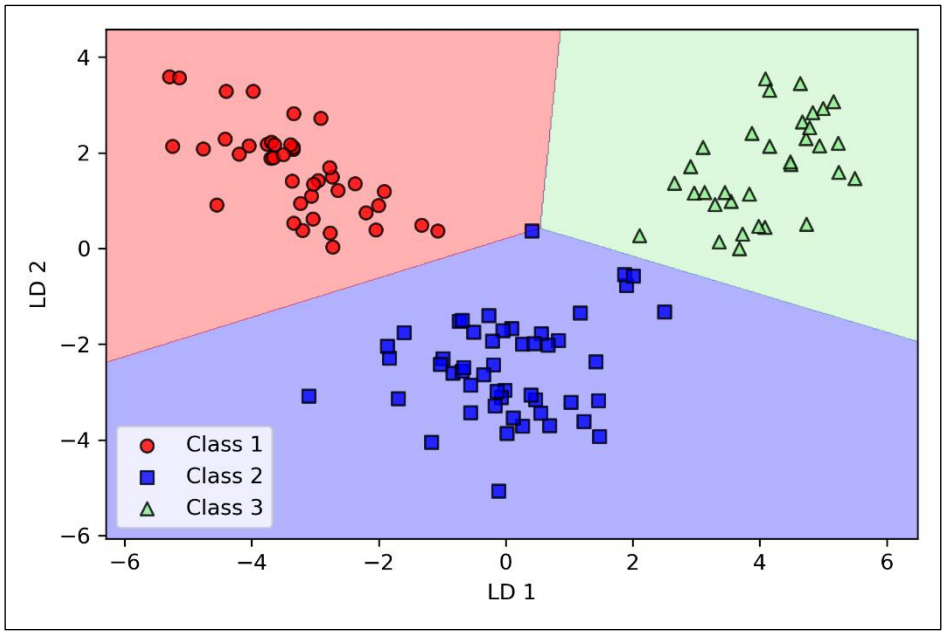

Loadings for PC 1
Alcohol
Malic acid
Ash
Alcalinity of ash
Magnesium
Total phenols
Flavanoids
Nonflavanoid phenols
Proanthocyanins
Color intensity
Hue
OD280/OD315 of diluted wines
Proline

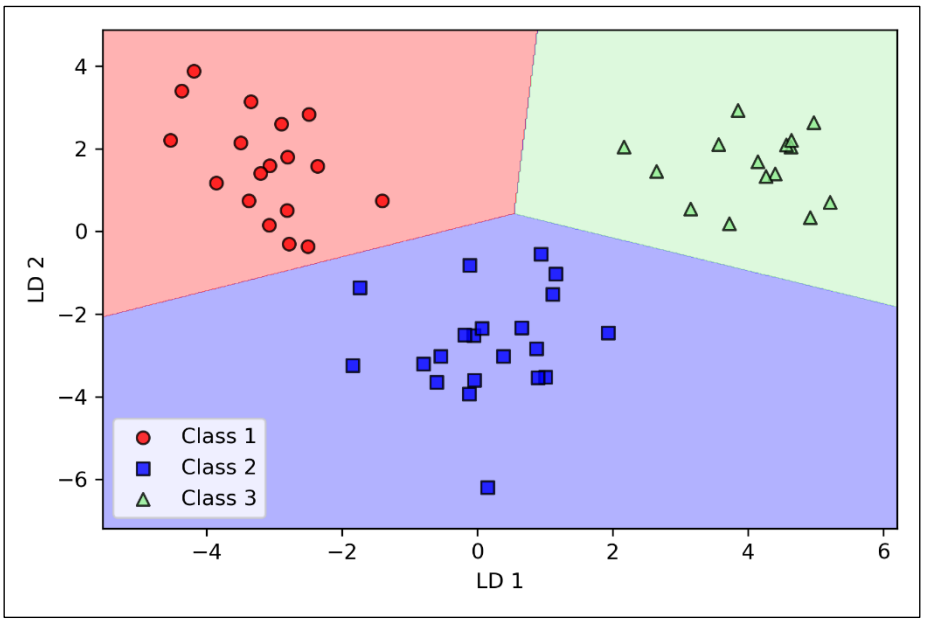

Loadings for PC 1
1.0
0.5
0.0
−0.5
−1.0
Alcohol
Malic acid
Ash
Alcalinity of ash
Magnesium
Total phenols
Flavanoids
Nonflavanoid phenols
Proanthocyanins
Color intensity
Hue
OD280/OD315 of diluted wines
Proline

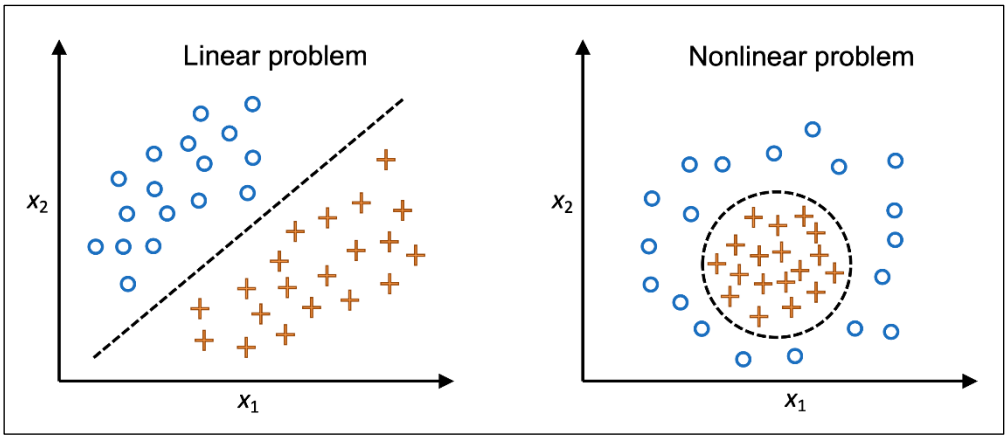

LD 2
LD 1

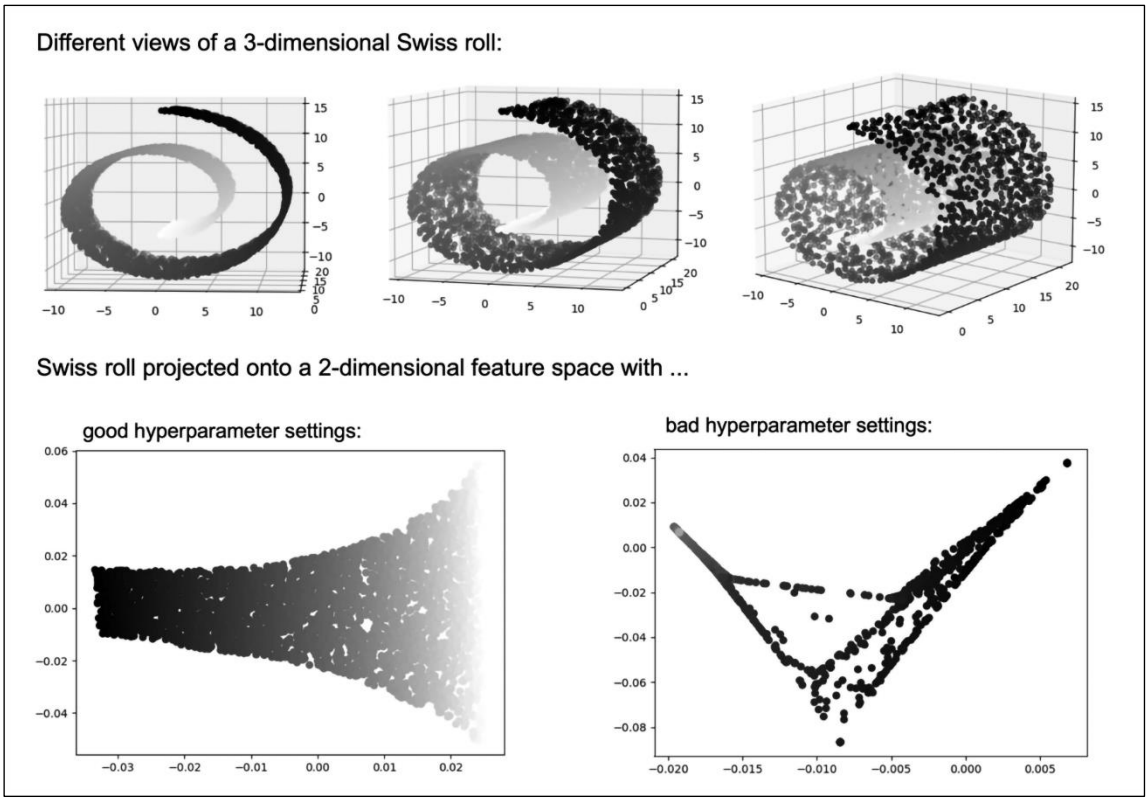

Discriminability ratio
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
12
14
Linear discriminants
Cumulative discriminability
Individual discriminability

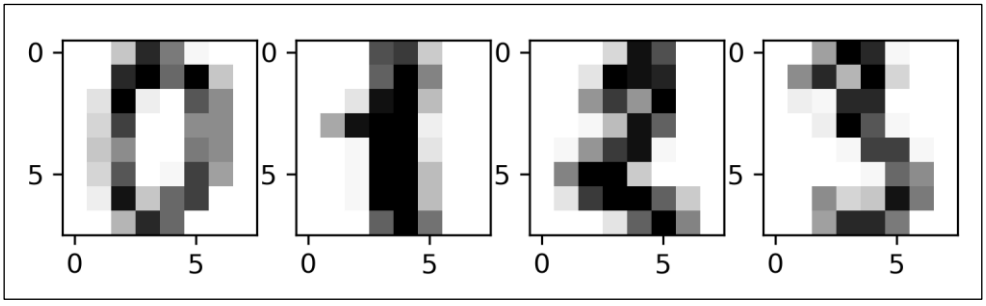

LD 2
2
1
0
−1
−2
−3
−2
−1
0
1
2
LD 1
Class 1
Class 2
Class 3

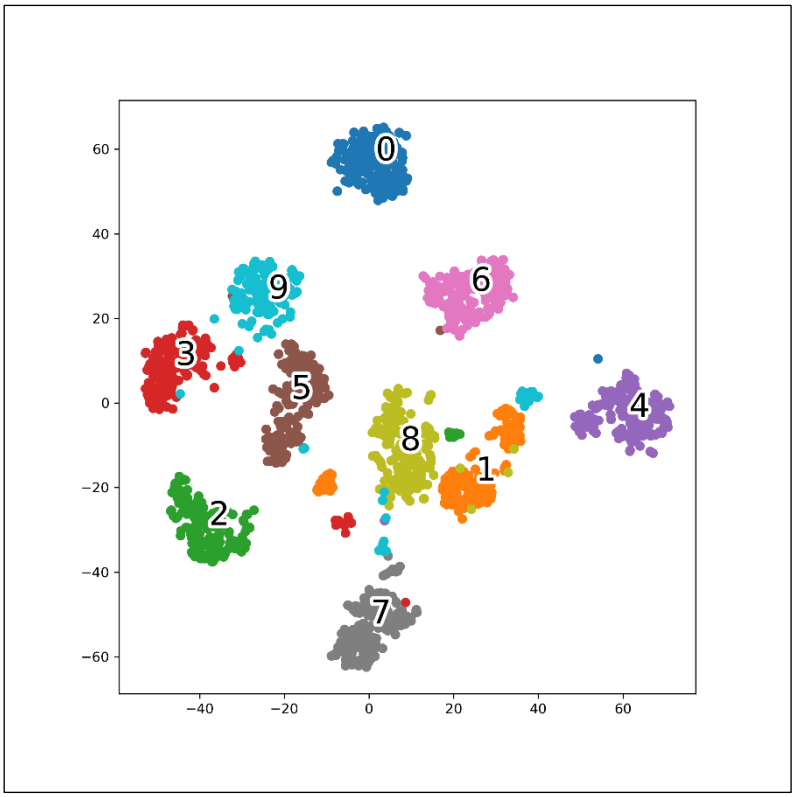

LD 2
LD 1
Class 1
Class 2
Class 3


LD 2
LD 1
Class 1
Class 2
Class 3

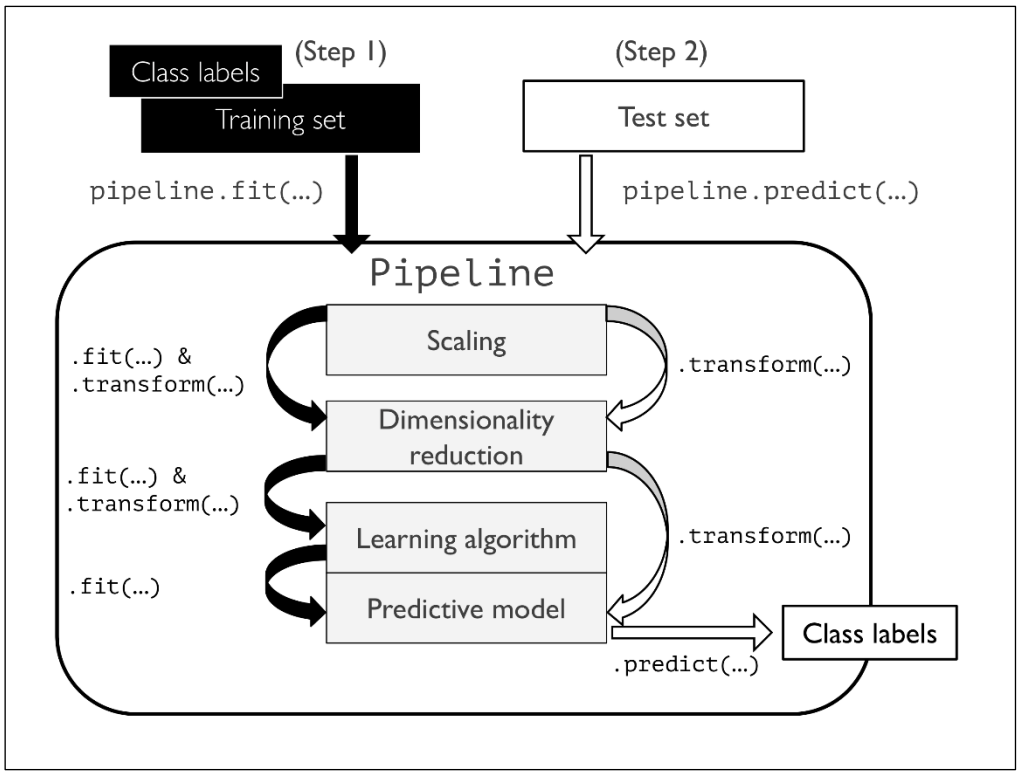

Linear problem
Nonlinear problem
x_2
x_1
x_2
x_1

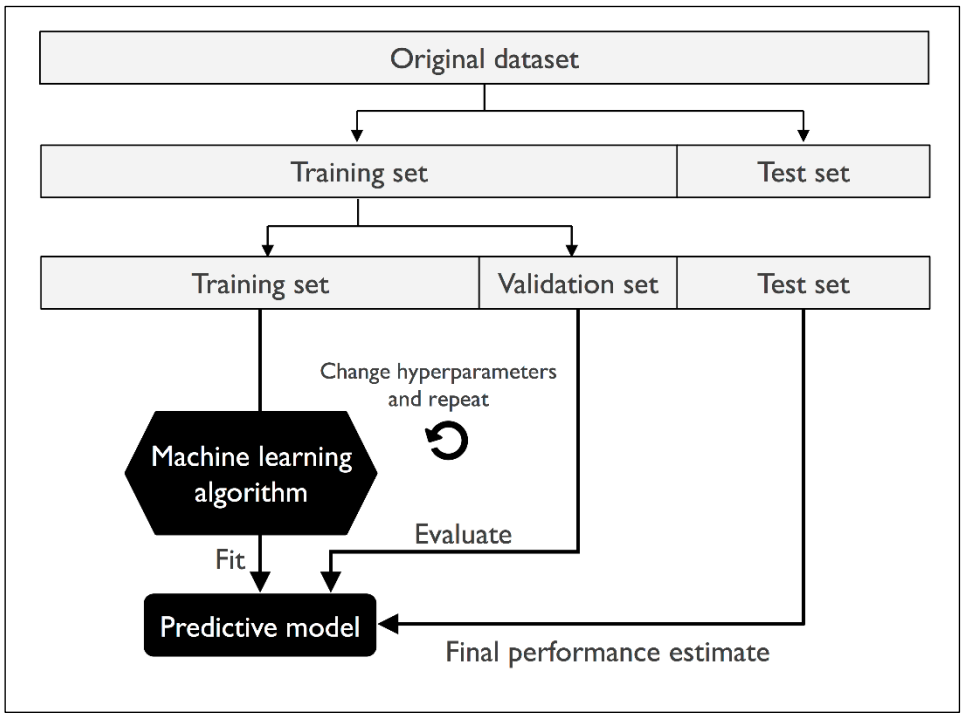

Different views of a 3-dimensional Swiss roll:
Swiss roll projected onto a 2-dimensional feature space with ...
good hyperparameter settings:
bad hyperparameter settings:

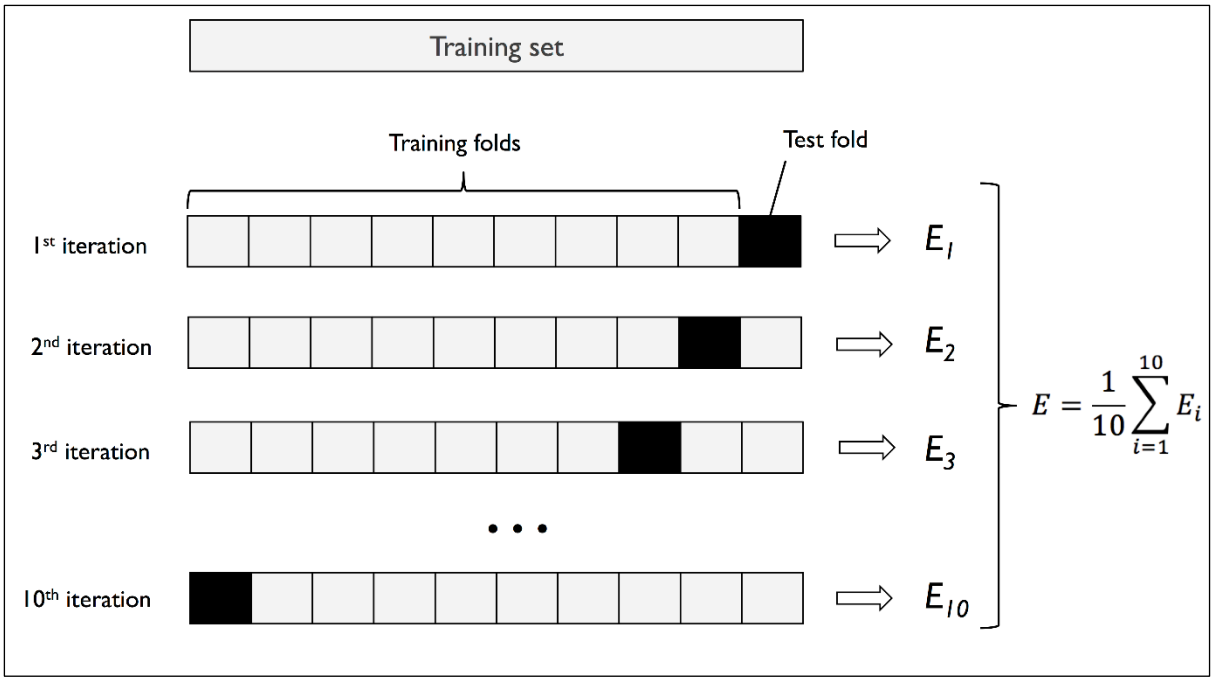


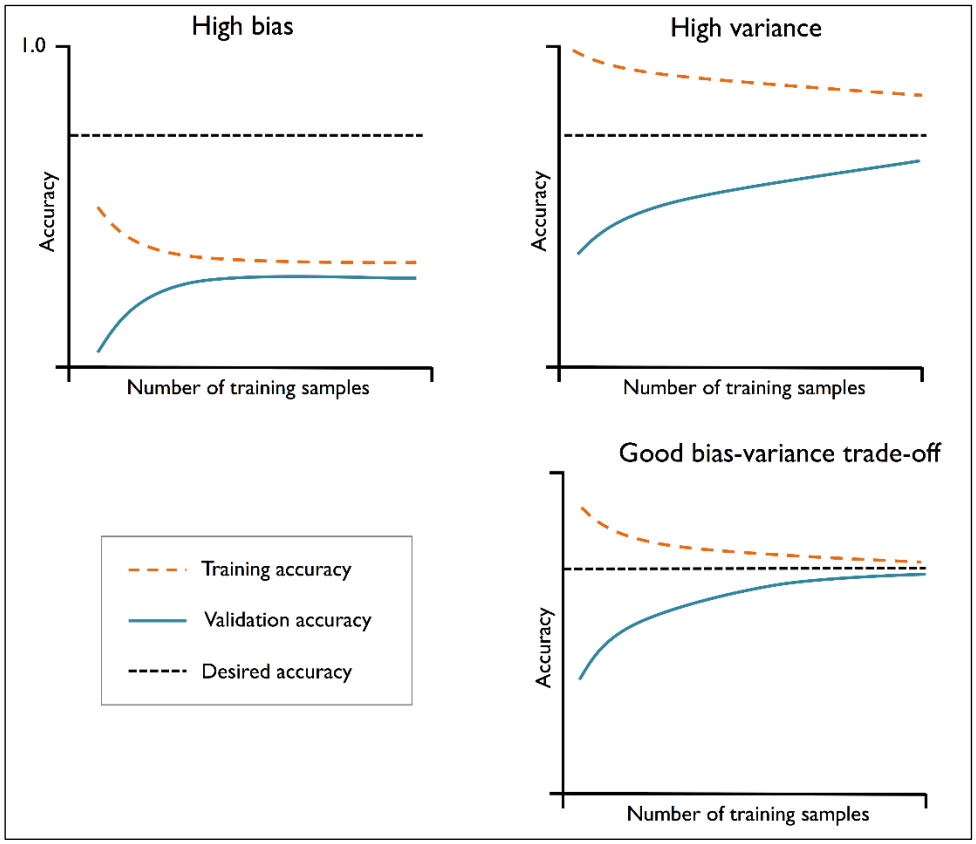

0
1
2
3
4
5
6
7
8
9
60
40
20
0
−20
−40
−60
−40
−20
0
20
40
60

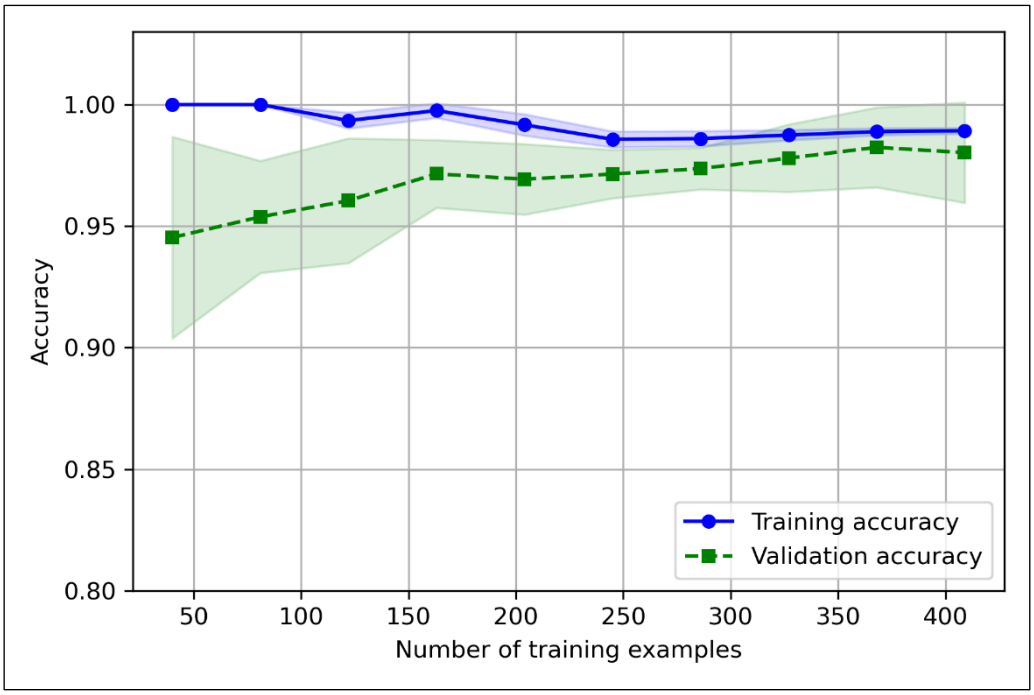
Chapter 6: Learning Best Practices for Model Evaluation and Hyperparameter Tuning

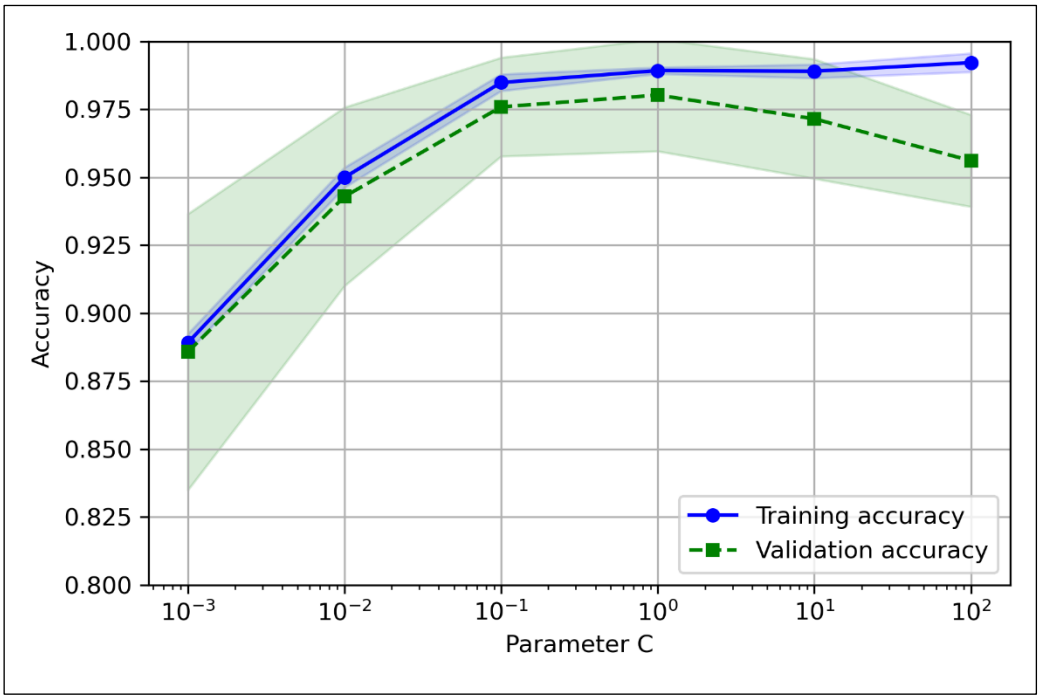


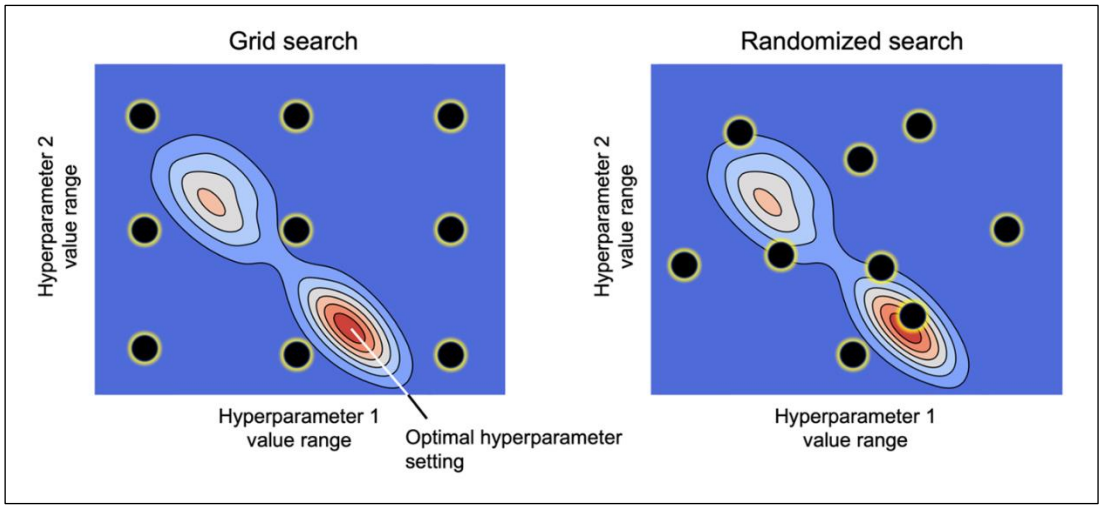


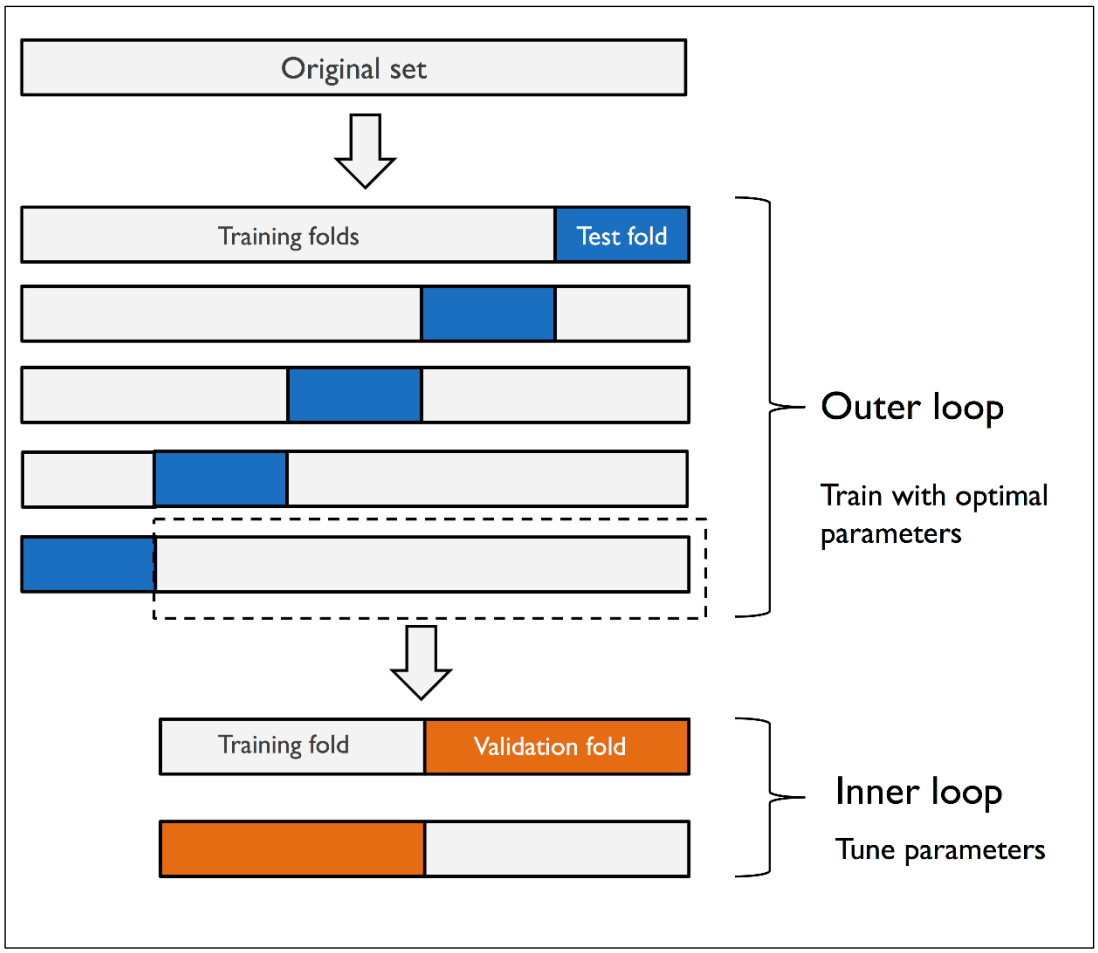

Training set
Training folds
Test fold
1st iteration
2nd iteration
3rd iteration
10th iteration
E_1
E_2
E_3
E_{10}
$E = \frac{1}{10}\sum_{i=1}^{10} E_i$

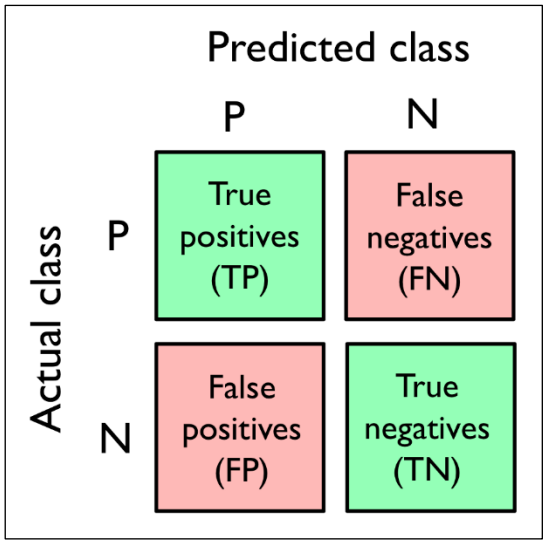

High bias
High variance
Good bias-variance trade-off
1.0
Accuracy
Number of training samples
Training accuracy
Validation accuracy
Desired accuracy

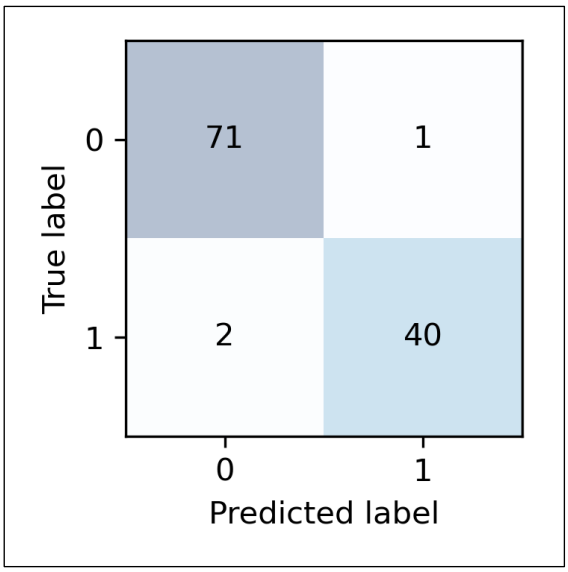

1.00
0.95
0.90
0.85
0.80
Accuracy
50
100
150
200
250
300
350
400
Number of training examples
Training accuracy
Validation accuracy

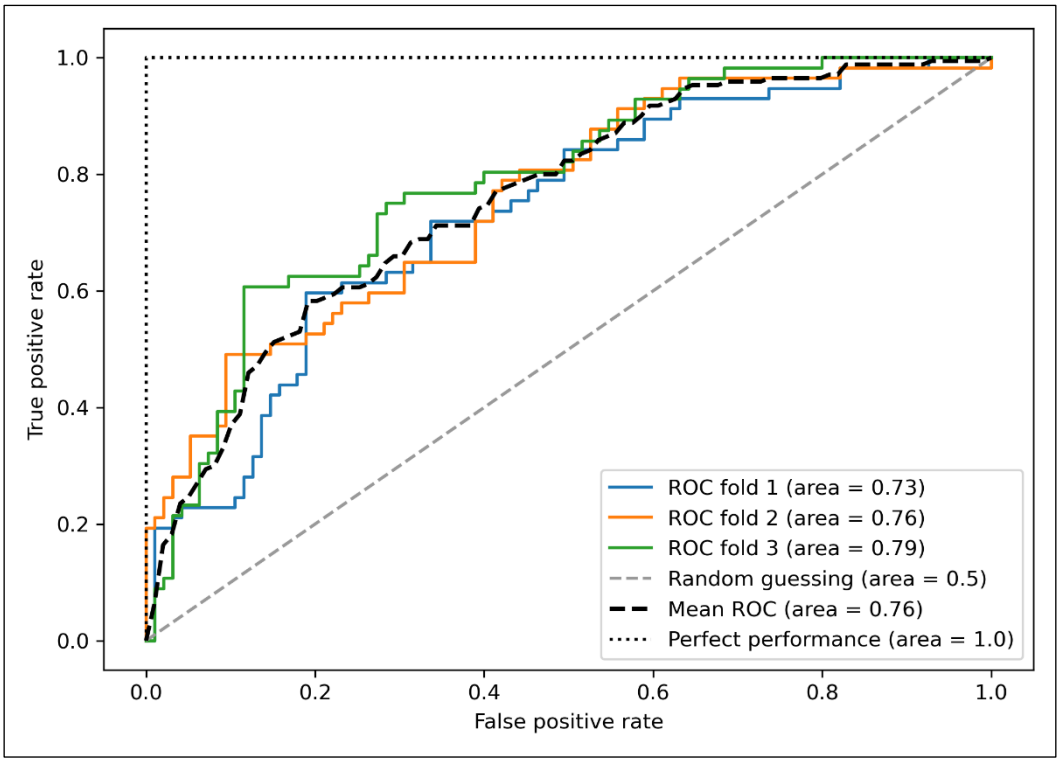

1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
Accuracy
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Parameter C
Training accuracy
Validation accuracy


Grid search
Randomized search
Hyperparameter 2 value range
Hyperparameter 2 value range
Hyperparameter 1 value range
Optimal hyperparameter setting
Hyperparameter 1 value range

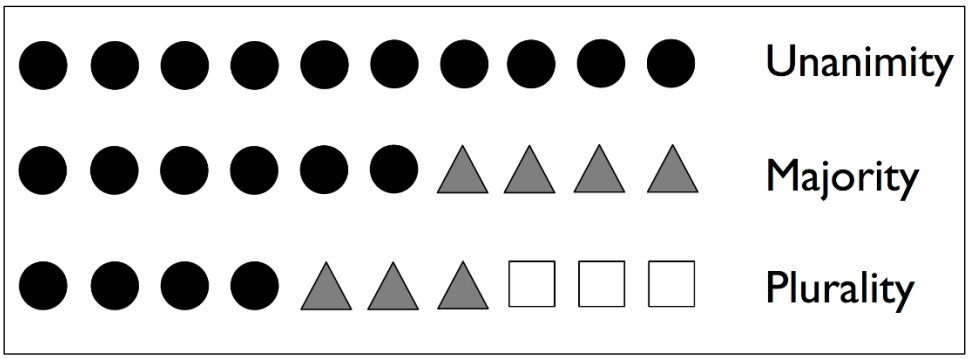

Original set
Training folds
Test fold
Outer loop
Train with optimal parameters
Training fold
Validation fold
Inner loop
Tune parameters

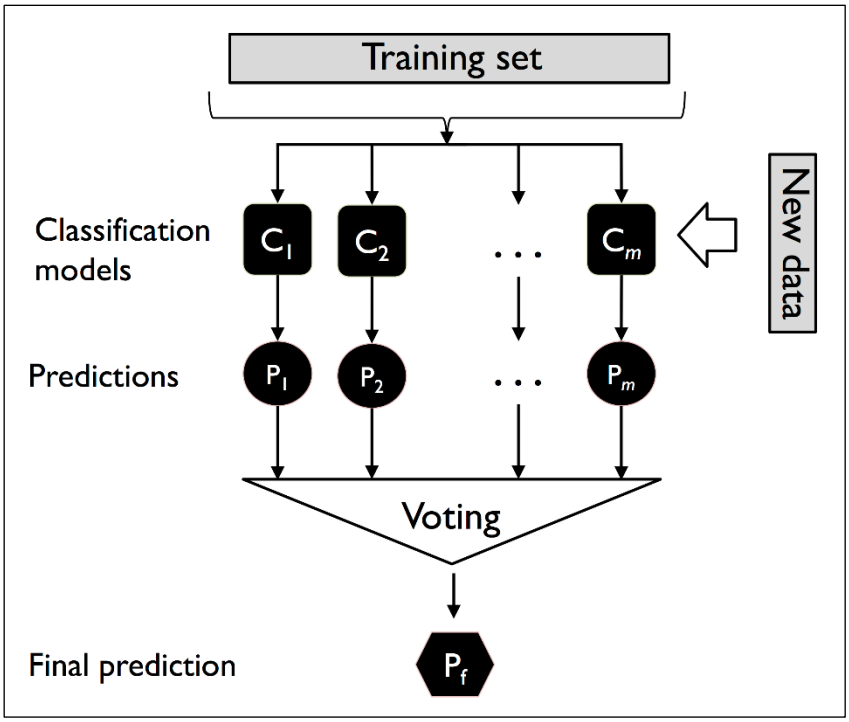

Predicted class
P
N
Actual class
P
N
True positives (TP)
False negatives (FN)
False positives (FP)
True negatives (TN)

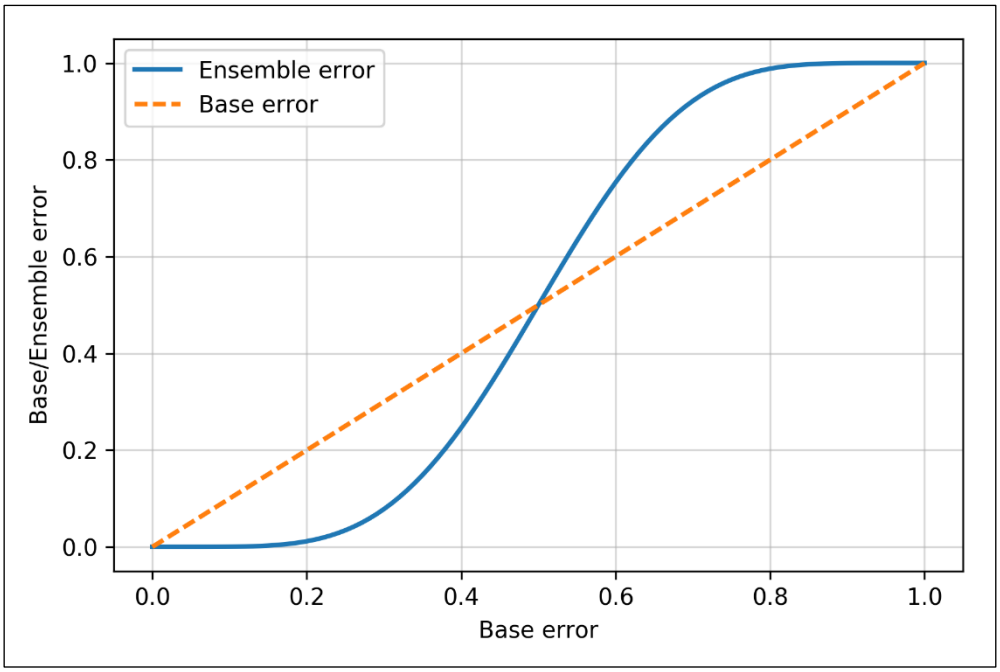

True label
0
1
71
1
2
40
0
1
Predicted label

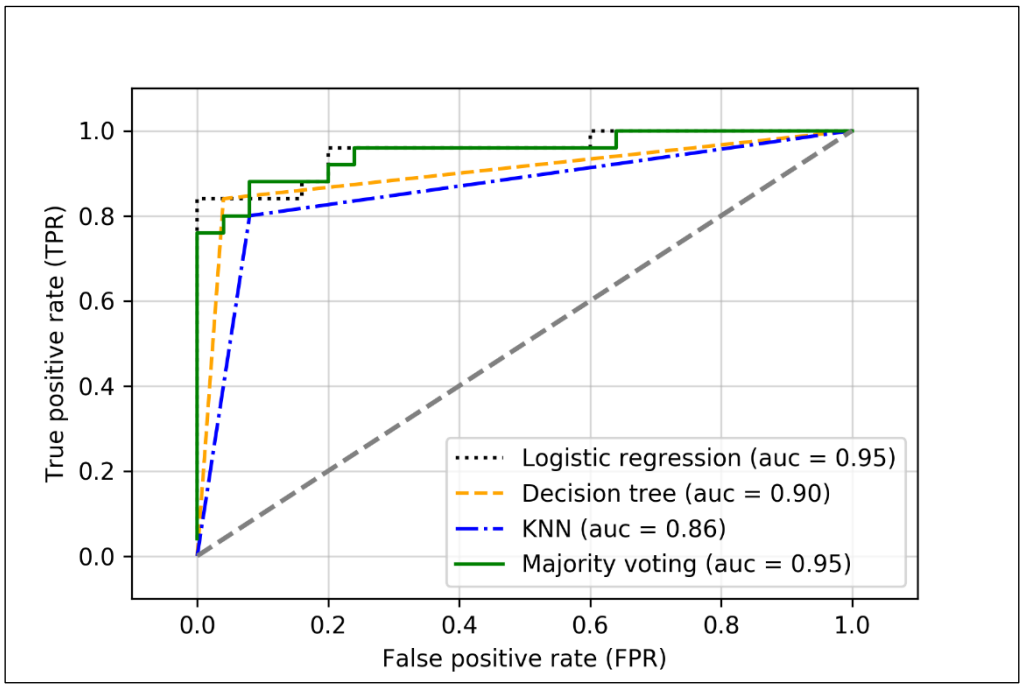

True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC fold 1 (area = 0.73)
ROC fold 2 (area = 0.76)
ROC fold 3 (area = 0.79)
Random guessing (area = 0.5)
Mean ROC (area = 0.76)
Perfect performance (area = 1.0)

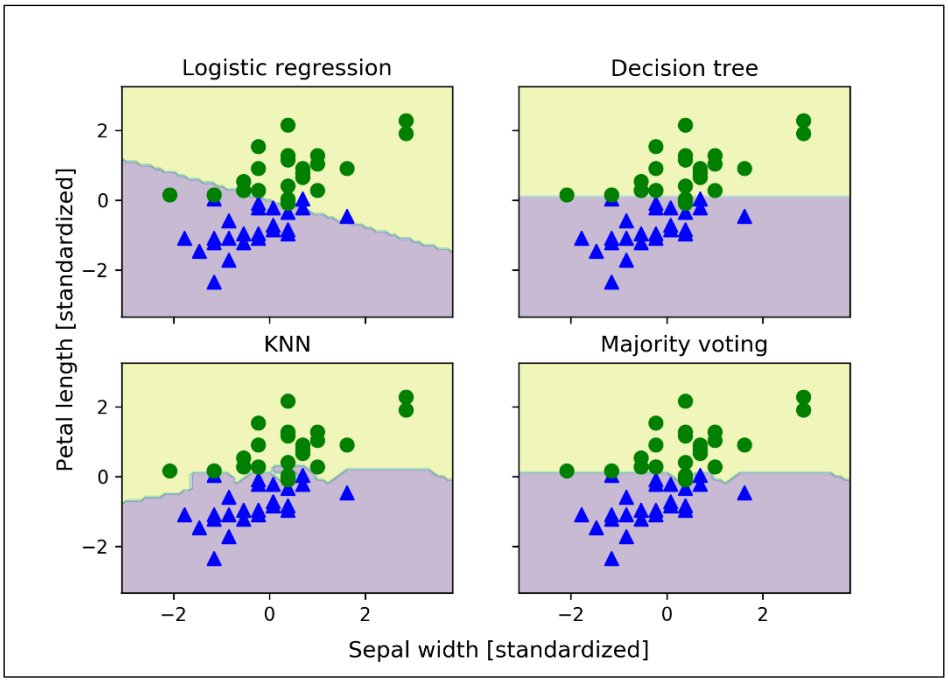
Chapter 7: Combining Different Models for Ensemble Learning

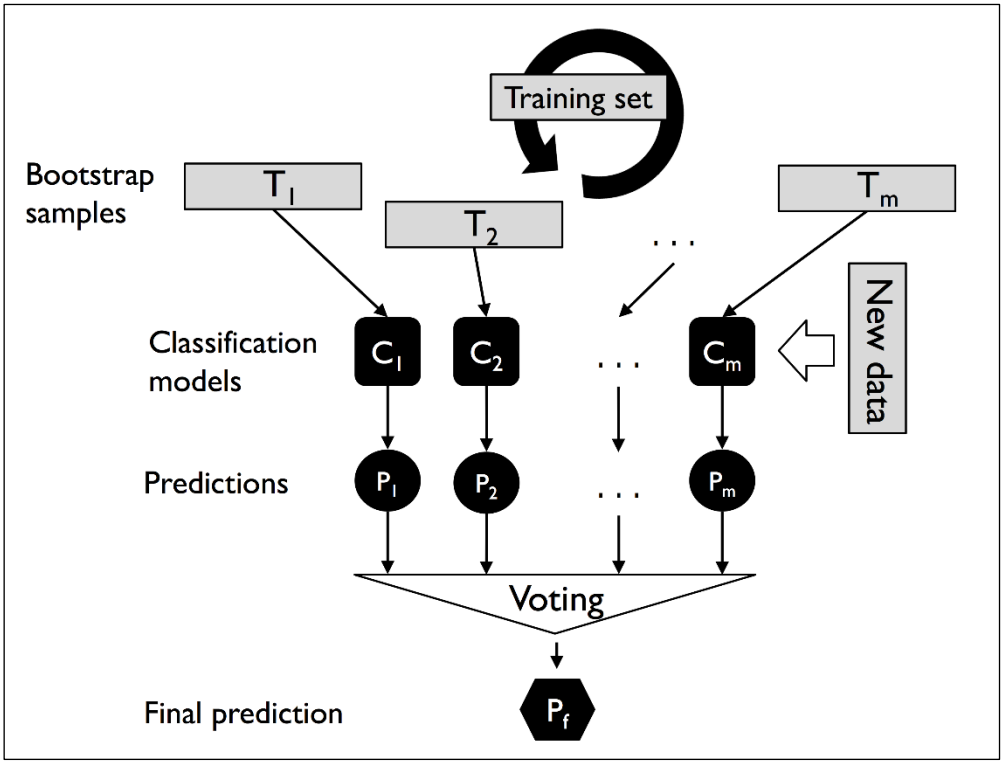


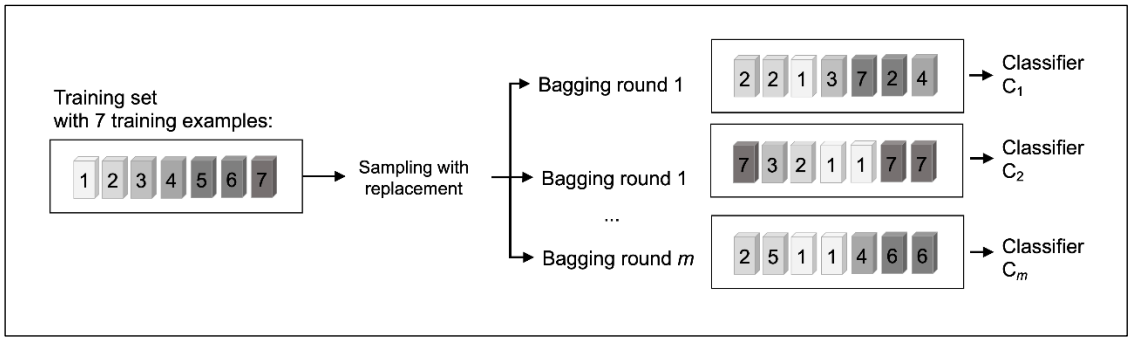


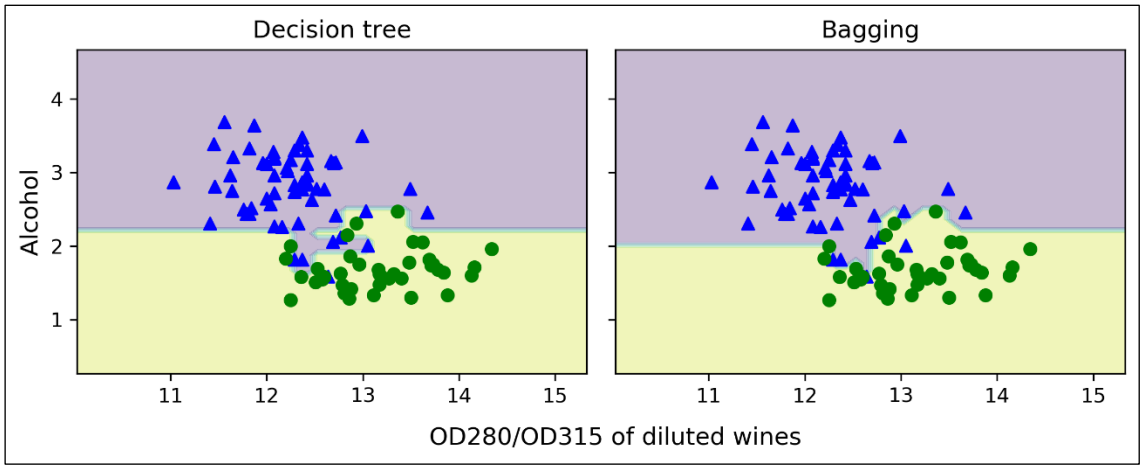

Ensemble error
Base error
Base/Ensemble error
Base error
0.0
0.2
0.4
0.6
0.8
1.0

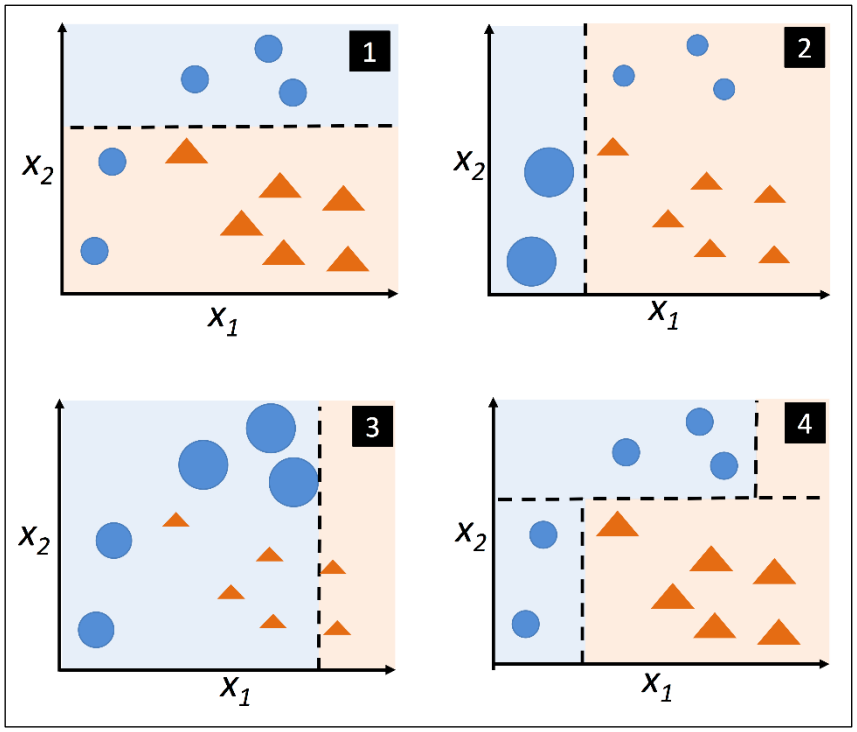

Logistic regression (auc = 0.95)
Decision tree (auc = 0.90)
KNN (auc = 0.86)
Majority voting (auc = 0.95)
True positive rate (TPR)
False positive rate (FPR)
0.0
0.2
0.4
0.6
0.8
1.0


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

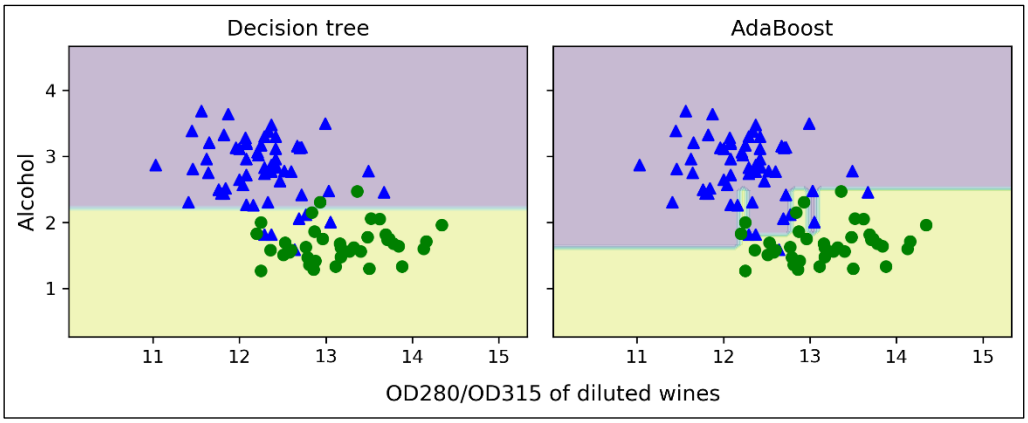

Training set
Bootstrap samples
T_1
T_2
T_m
Classification models
C_1
C_2
C_m
New data
Predictions
P_1
P_2
P_m
Voting
Final prediction
P_f


Training set
with 7 training examples:
1 2 3 4 5 6 7
Sampling with
replacement
Bagging round 1
Bagging round 1
...
Bagging round m
2 2 1 3 7 2 4
7 3 2 1 1 7 7
2 5 1 1 4 6 6
Classifier
C1
Classifier
C2
Classifier
Cm


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

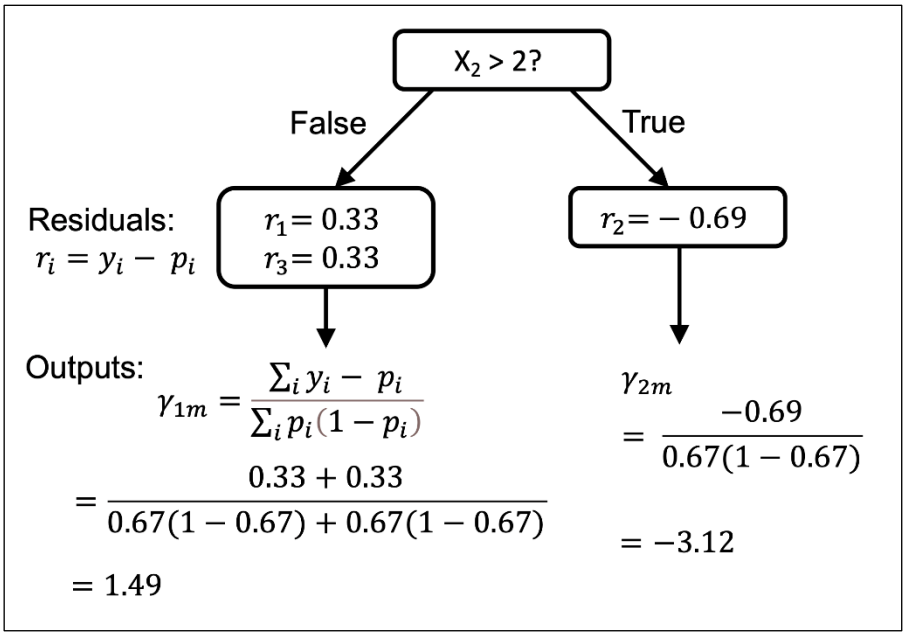

1
2
3
4
x2
x1

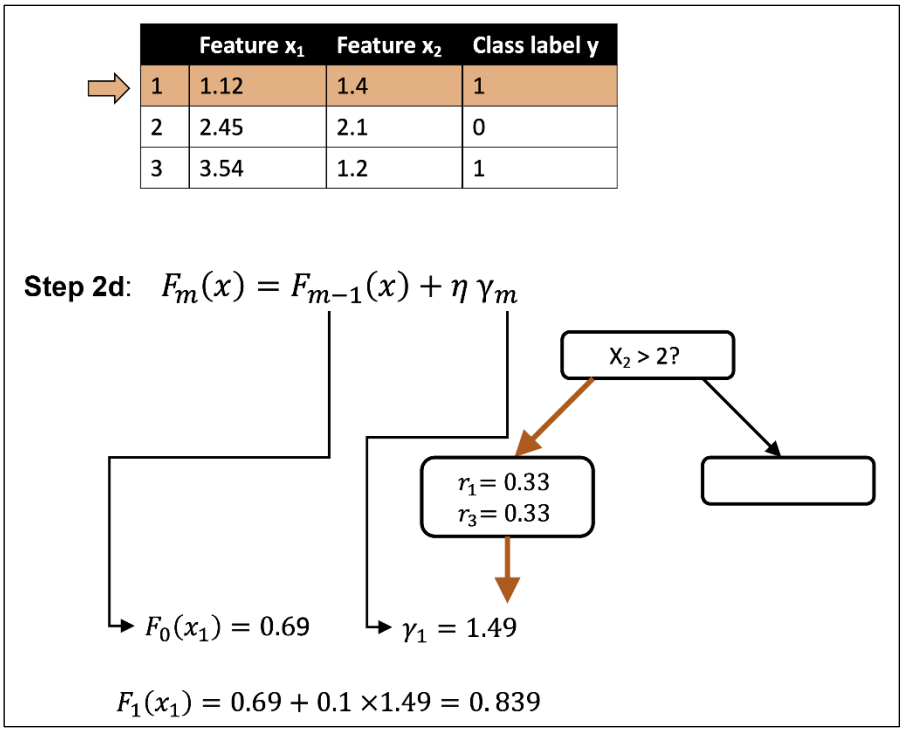
Index	x	y	Weights	$\hat{y}$(x <= 3.0)?	Correct?	Updated weights
1	1.0	1	0.1	1	Yes	0.072
2	2.0	1	0.1	1	Yes	0.072
3	3.0	1	0.1	1	Yes	0.072
4	4.0	-1	0.1	-1	Yes	0.072
5	5.0	-1	0.1	-1	Yes	0.072
6	6.0	-1	0.1	-1	Yes	0.072
7	7.0	1	0.1	-1	No	0.167
8	8.0	1	0.1	-1	No	0.167
9	9.0	1	0.1	-1	No	0.167
10	10.0	-1	0.1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



	Feature x_1	Feature x_2	Class label y
1	1.12	1.4	1
2	2.45	2.1	0
3	3.54	1.2	1

	Feature x_1	Feature x_2	Class label y	Step 1: $\hat{y}$ = log(odds)	Step 2a: $p = \frac{1}{1+e^{-\hat{y}}}$	Step 2a: $r = y - p$
1	1.12	1.4	1	0.69	0.67	0.33
2	2.45	2.1	0	0.69	0.67	-0.67
3	3.54	1.2	1	0.69	0.67	0.33





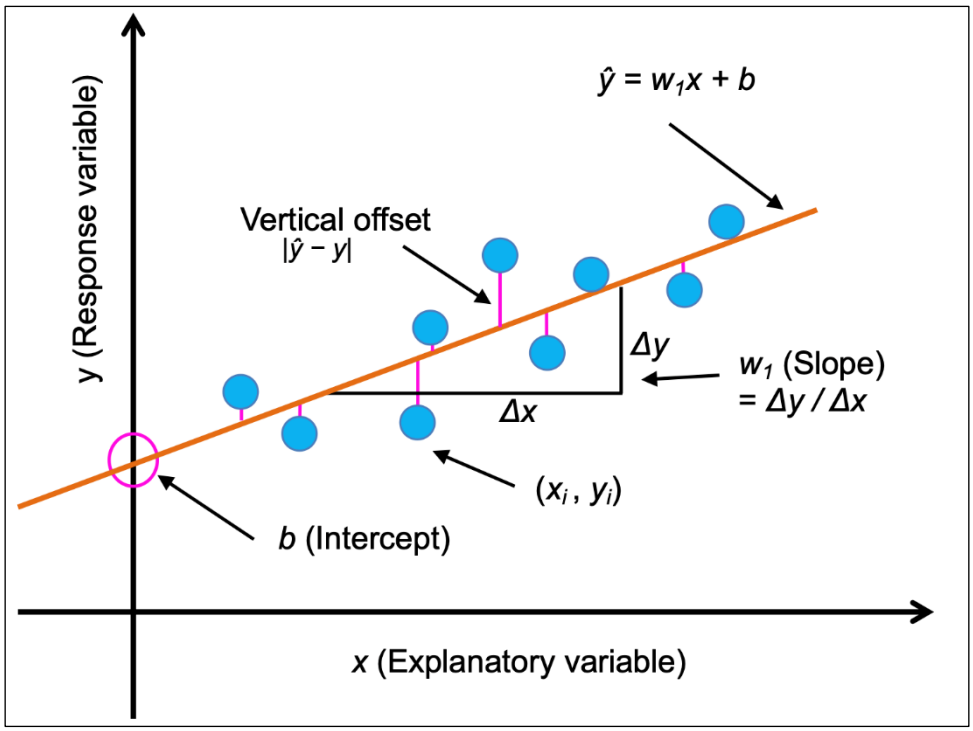
	x_1	x_2	y	Step 1: $F_0(x) = \hat{y}$ = log(odds)	Step 2a: $p = \frac{1}{1+e^{-\hat{y}}}$	Step 2a: $r = y - p$	New log(odds) $\hat{y} = F_1(x)$	Step 2a: p	Step 2a: r
1	1.12	1.4	1	0.69	0.67	0.33	0.839	0.698	0.302
2	2.45	2.1	0	0.69	0.67	-0.67	0.378	0.593	-0.593
3	3.54	1.2	1	0.69	0.67	0.33	0.839	0.698	0.302

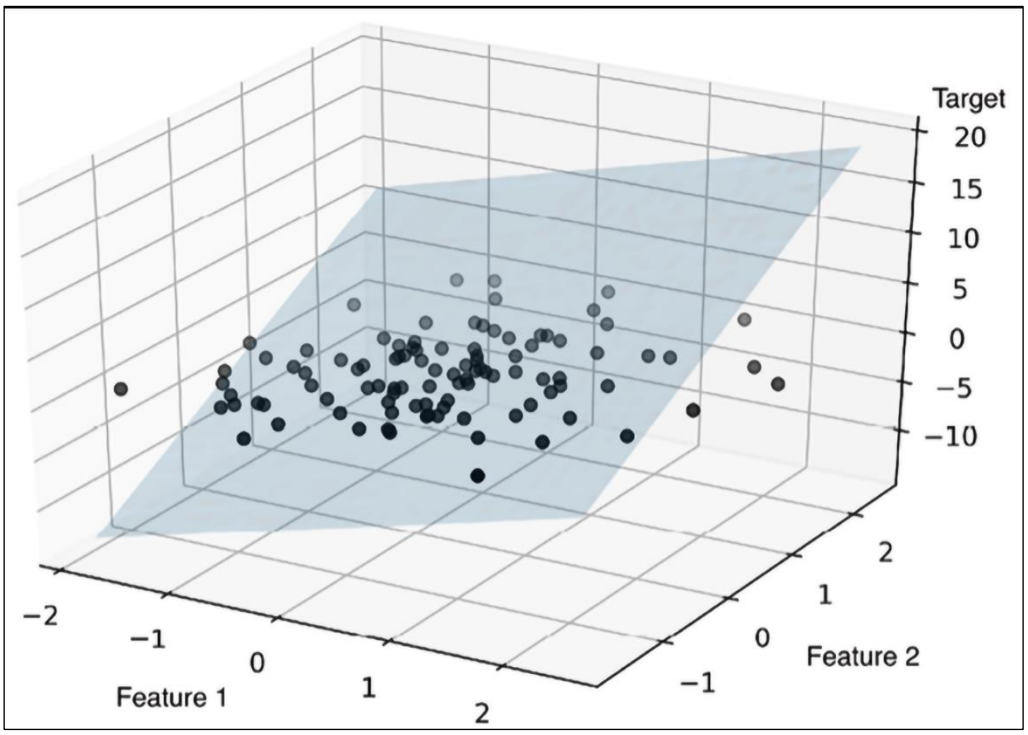
Round $m = 1$ Round $m = 2$

Chapter 8: Applying Machine Learning to Sentiment Analysis

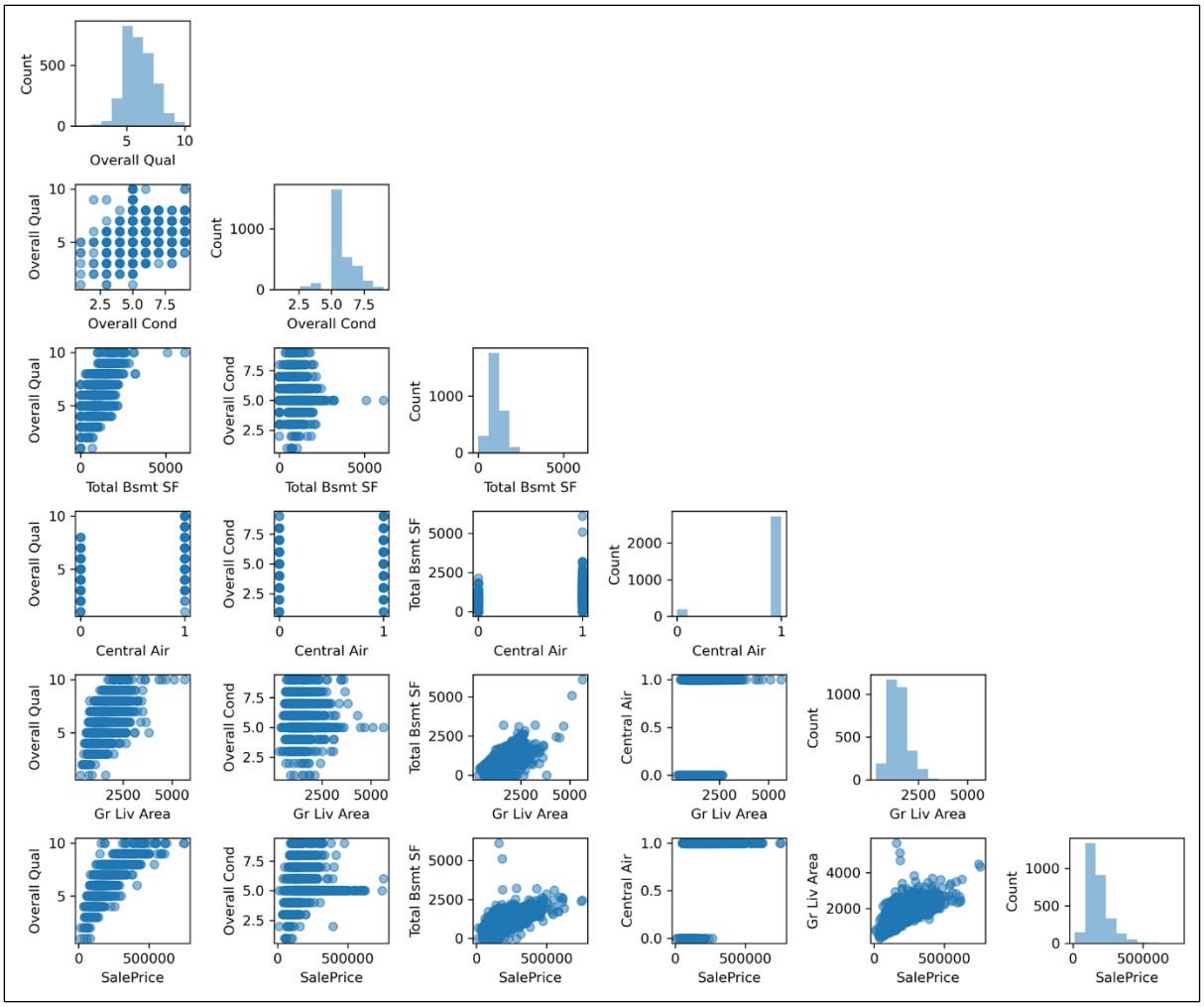
	review	sentiment
0	In 1974, the teenager Martha Moxley (Maggie Gr...	1
1	OK... so... I really like Kris Kristofferson a...	0
2	***SPOILER*** Do not read this, if you think a...	0

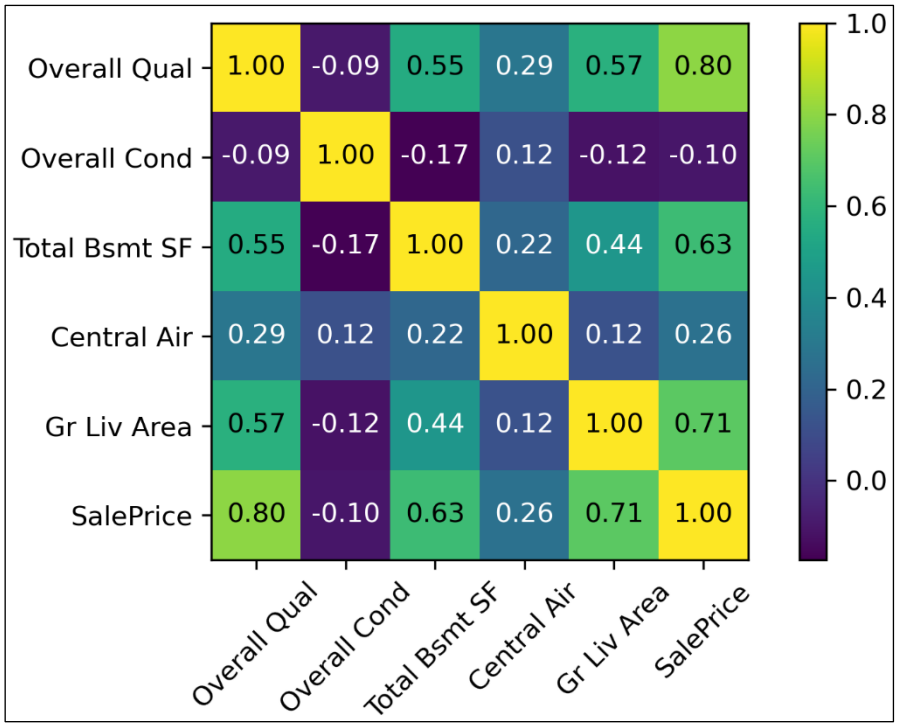
Chapter 9: Predicting Continuous Target Variables with Regression Analysis

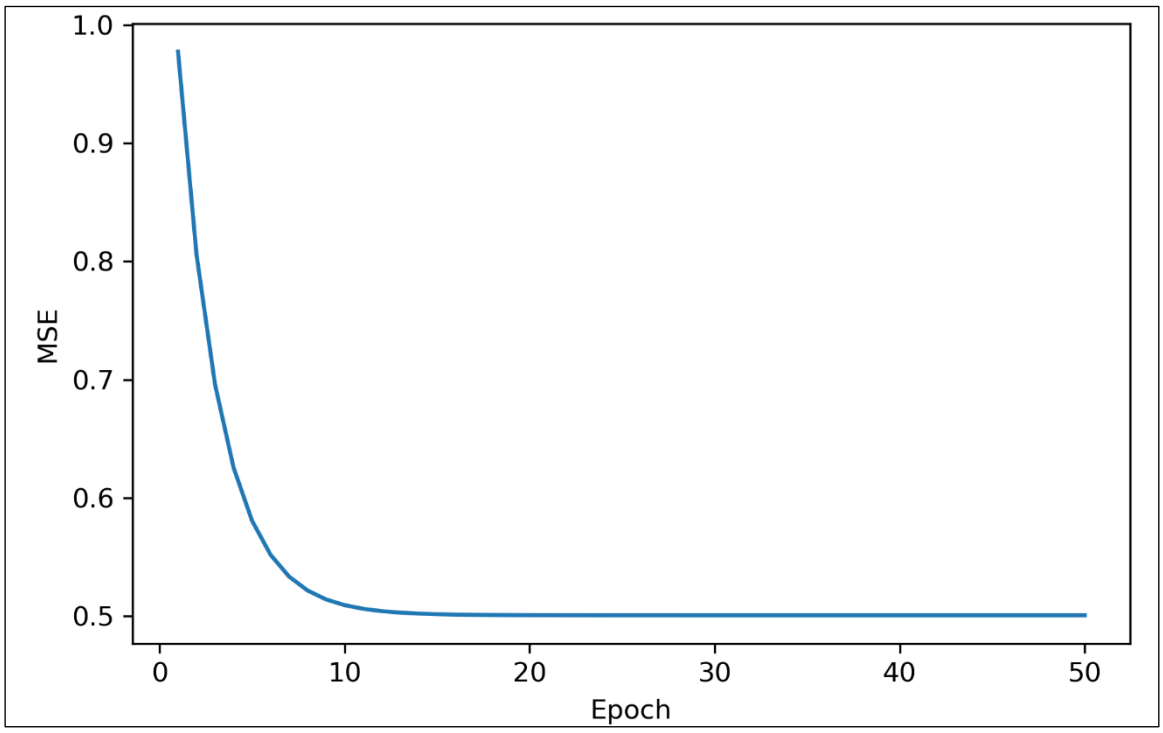


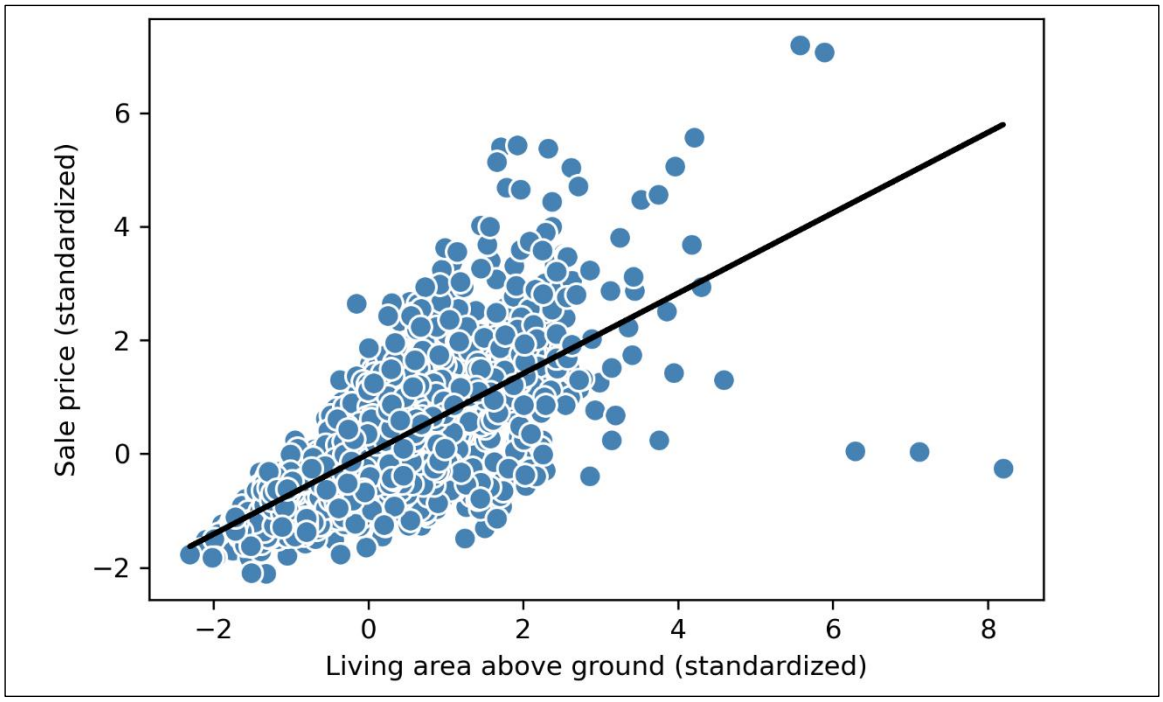


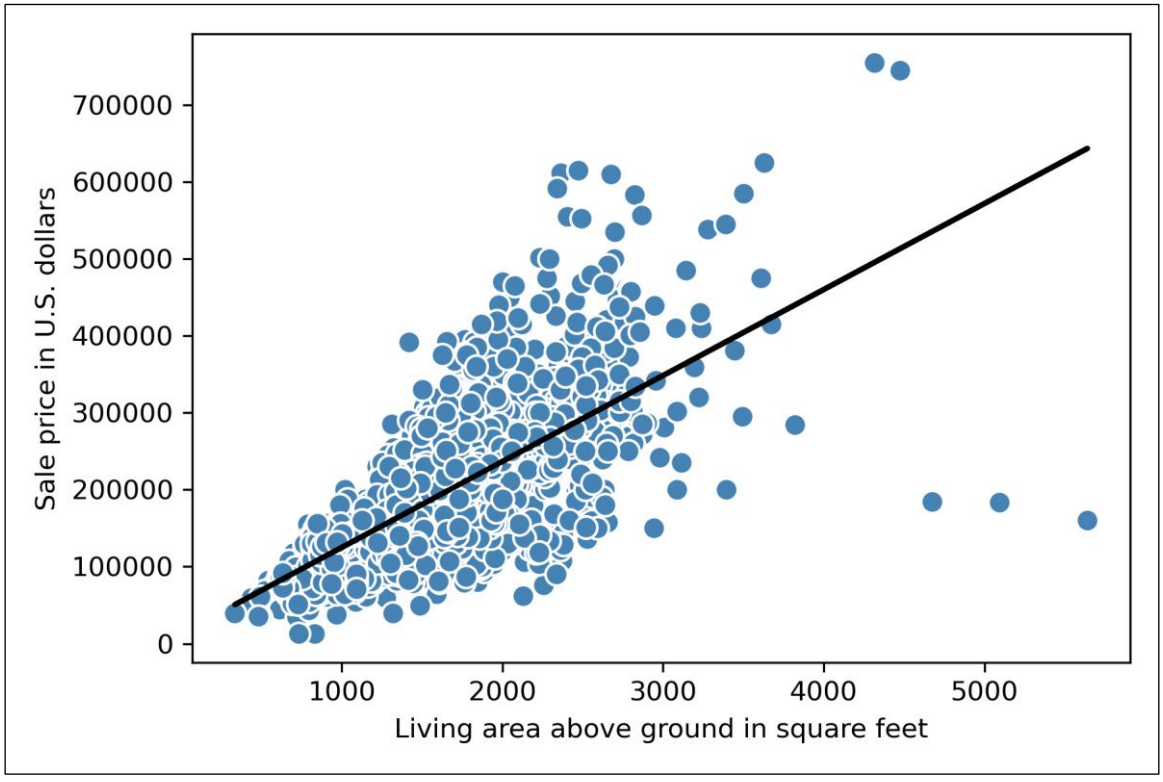
	Overall Qual	Overall Cond	Total Bsmt SF	Central Air	Gr Liv Area	SalePrice
0	6	5	1080.0	Y	1656	215000
1	5	6	882.0	Y	896	105000
2	6	6	1329.0	Y	1329	172000
3	7	5	2110.0	Y	2110	244000
4	5	5	928.0	Y	1629	189900

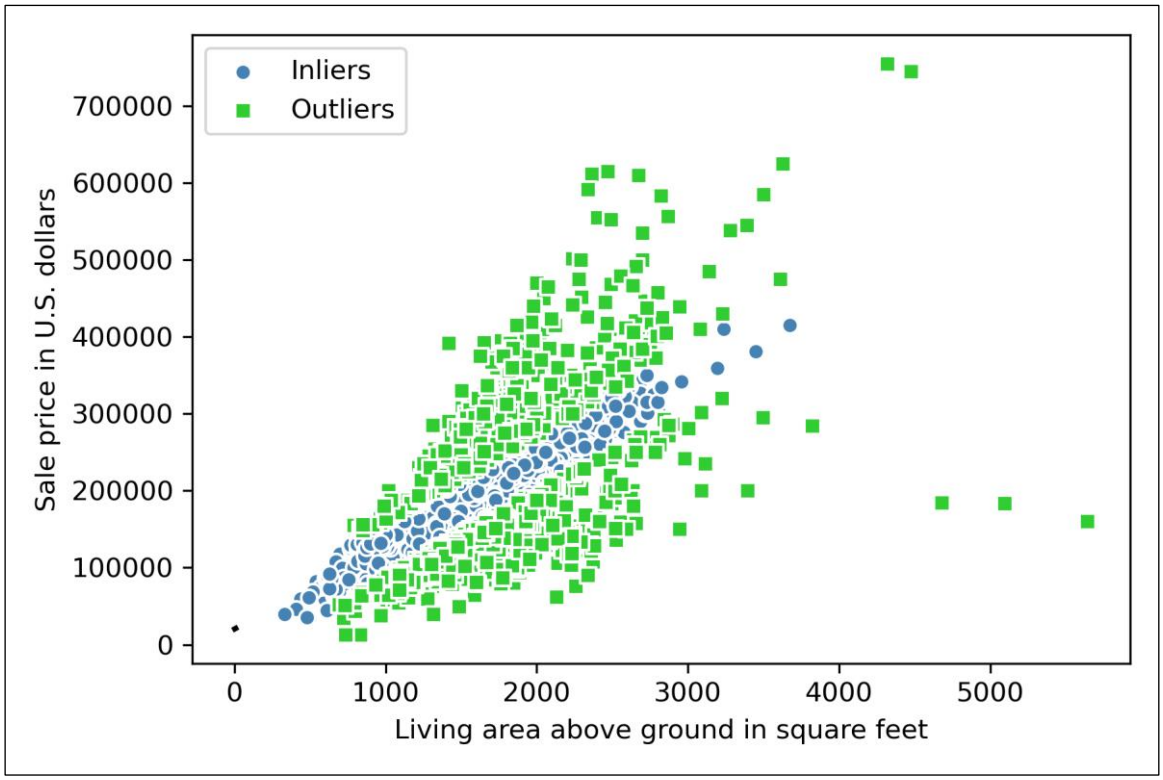


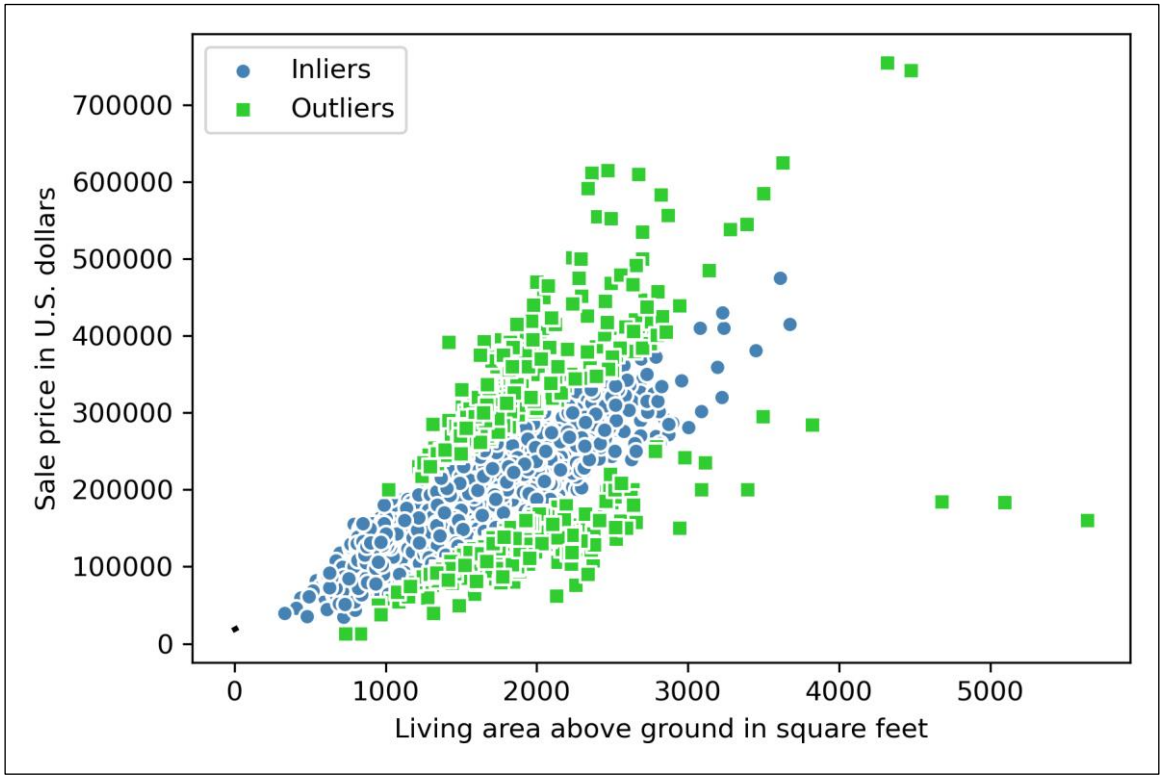

Overall Qual
Overall Cond
Total Bsmt SF
Central Air
Gr Liv Area
SalePrice
1.00 -0.09 0.55 0.29 0.57 0.80
-0.09 1.00 -0.17 0.12 -0.12 -0.10
0.55 -0.17 1.00 0.22 0.44 0.63
0.29 0.12 0.22 1.00 0.12 0.26
0.57 -0.12 0.44 0.12 1.00 0.71
0.80 -0.10 0.63 0.26 0.71 1.00
1.0
0.8
0.6
0.4
0.2
0.0

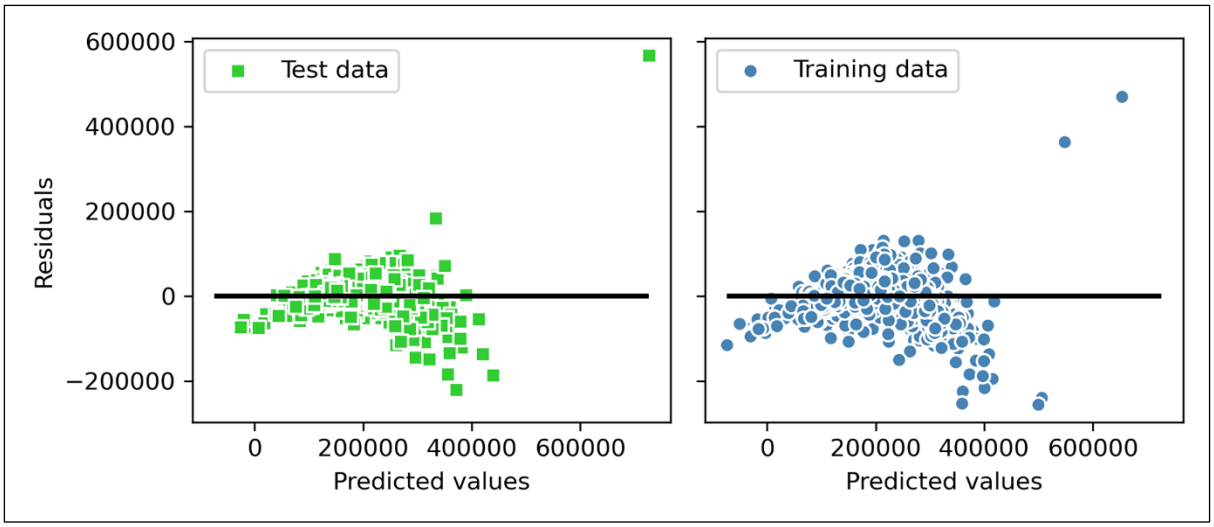

MSE
Epoch
1.0
0.9
0.8
0.7
0.6
0.5
0
10
20
30
40
50

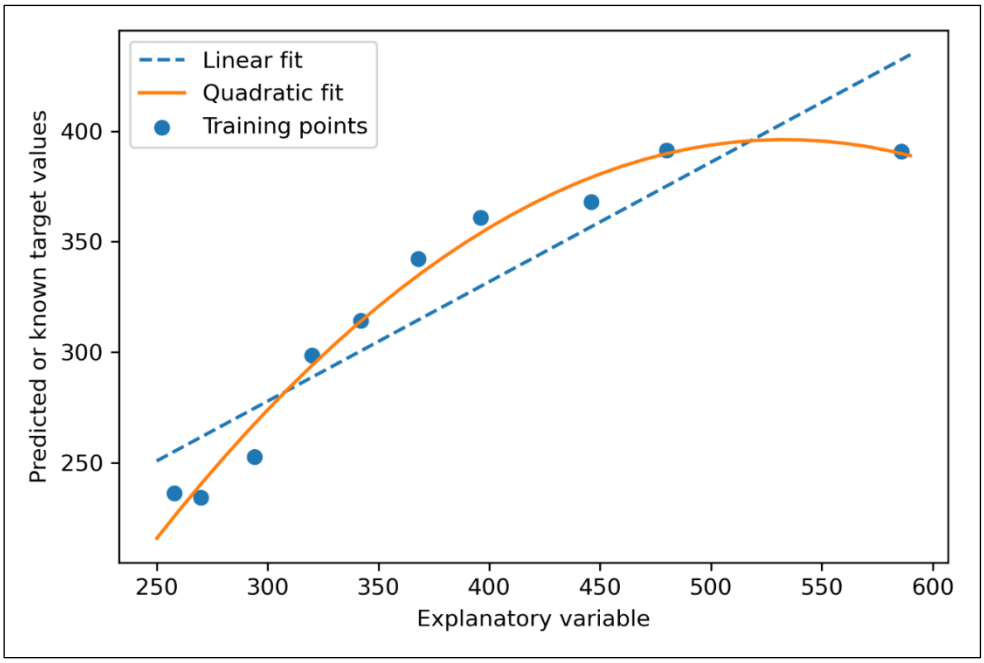

Sale price (standardized)
6
4
2
0
−2
−2
0
2
4
6
8
Living area above ground (standardized)

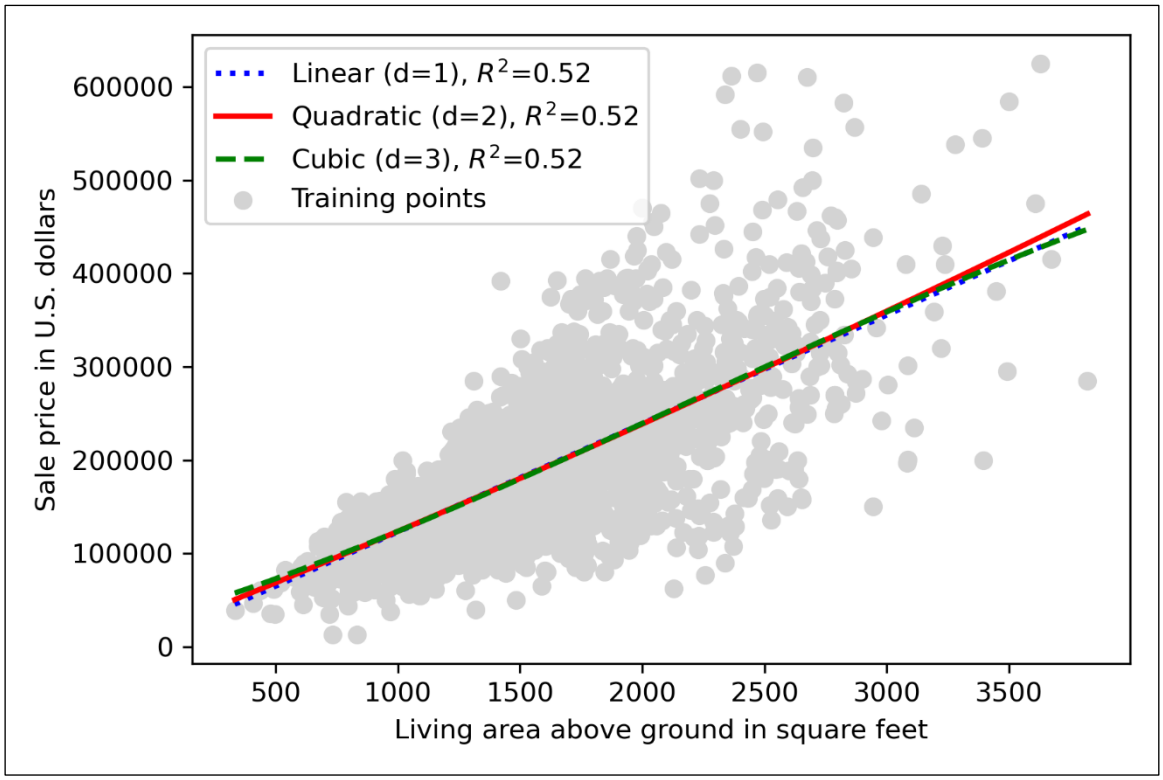

Sale price in U.S. dollars
700000
600000
500000
400000
300000
200000
100000
0
1000
2000
3000
4000
5000
Living area above ground in square feet

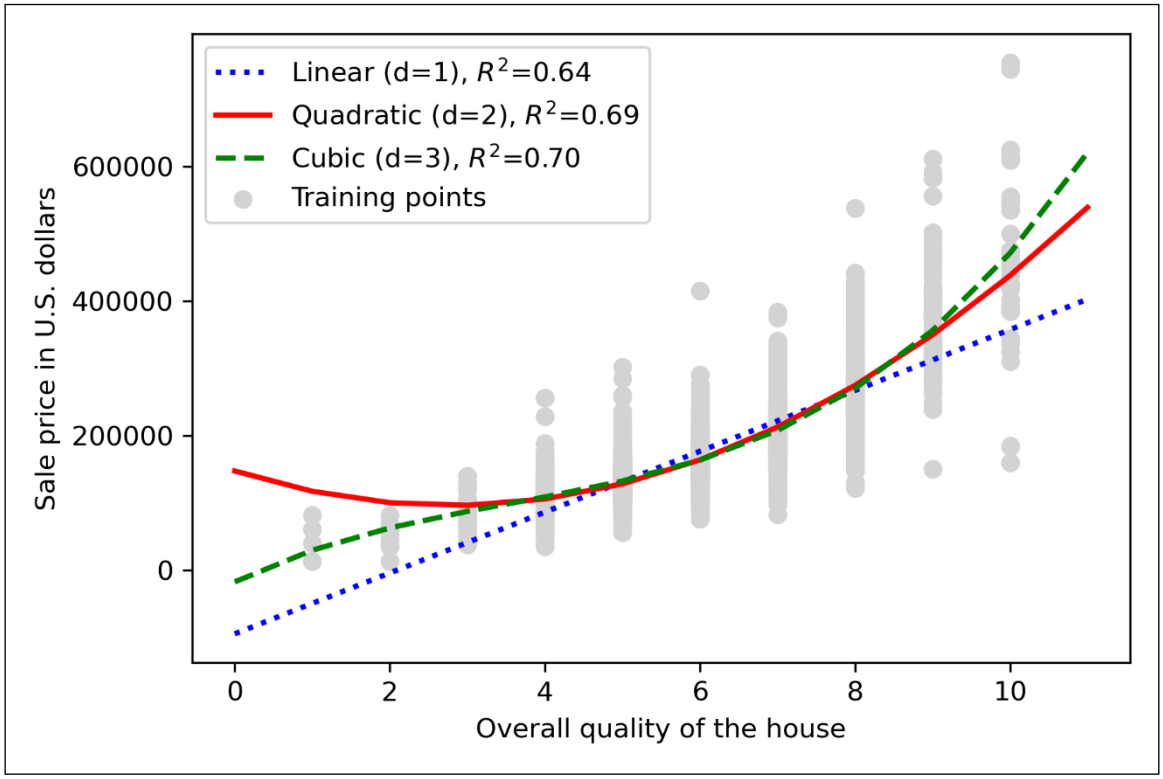

Inliers
Outliers
Sale price in U.S. dollars
700000
600000
500000
400000
300000
200000
100000
0
0
1000
2000
3000
4000
5000
Living area above ground in square feet

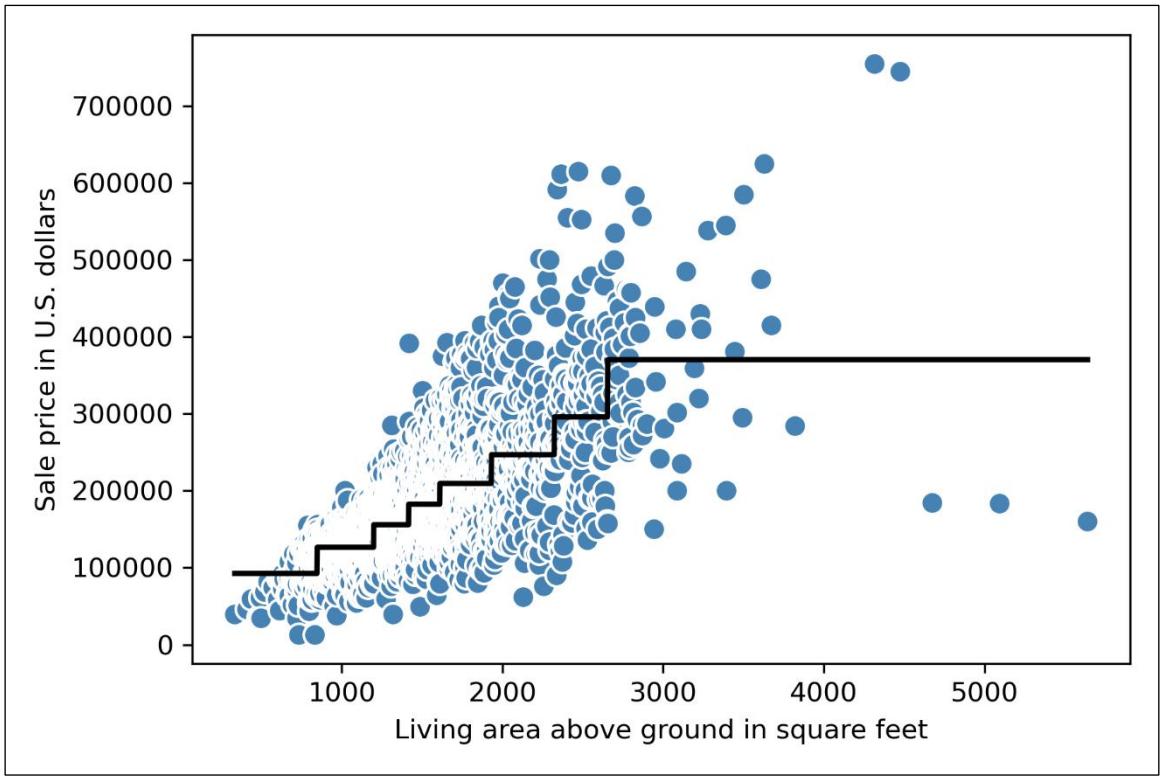

Inliers
Outliers
Sale price in U.S. dollars
700000
600000
500000
400000
300000
200000
100000
0
0
1000
2000
3000
4000
5000
Living area above ground in square feet

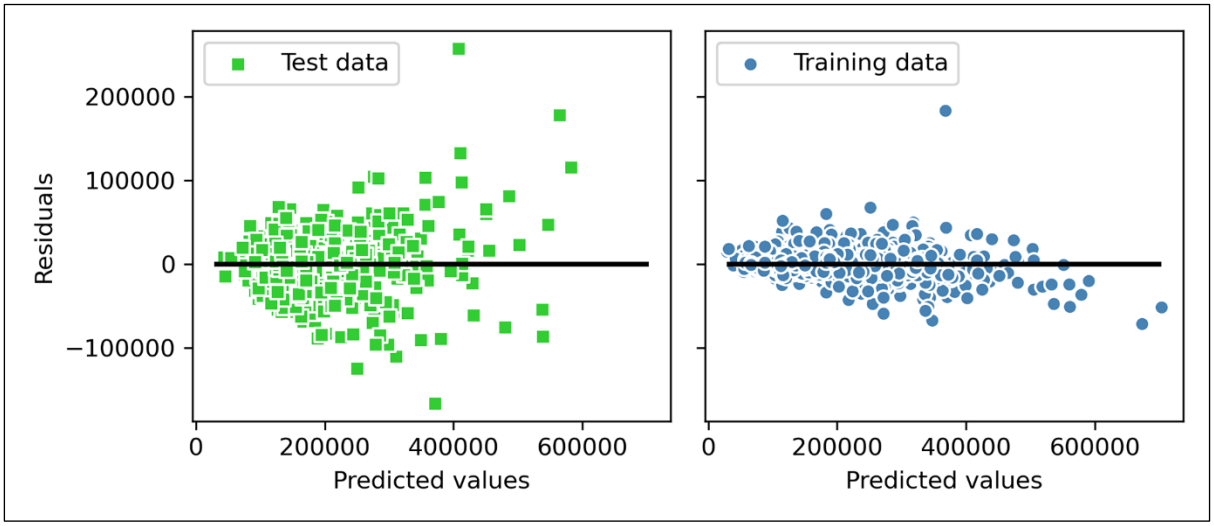

Test data
Training data
Residuals
600000
400000
200000
0
−200000
0
200000
400000
600000
Predicted values
Predicted values


Linear fit
Quadratic fit
Training points
Predicted or known target values
400
350
300
250
250
300
350
400
450
500
550
600
Explanatory variable

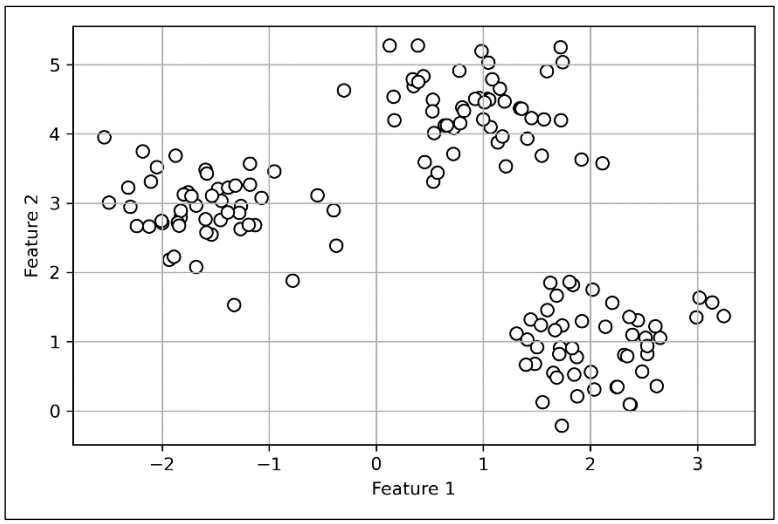

Linear (d=1), R^2=0.52
Quadratic (d=2), R^2=0.52
Cubic (d=3), R^2=0.52
Training points
Sale price in U.S. dollars
0
100000
200000
300000
400000
500000
600000
500
1000
1500
2000
2500
3000
3500
Living area above ground in square feet

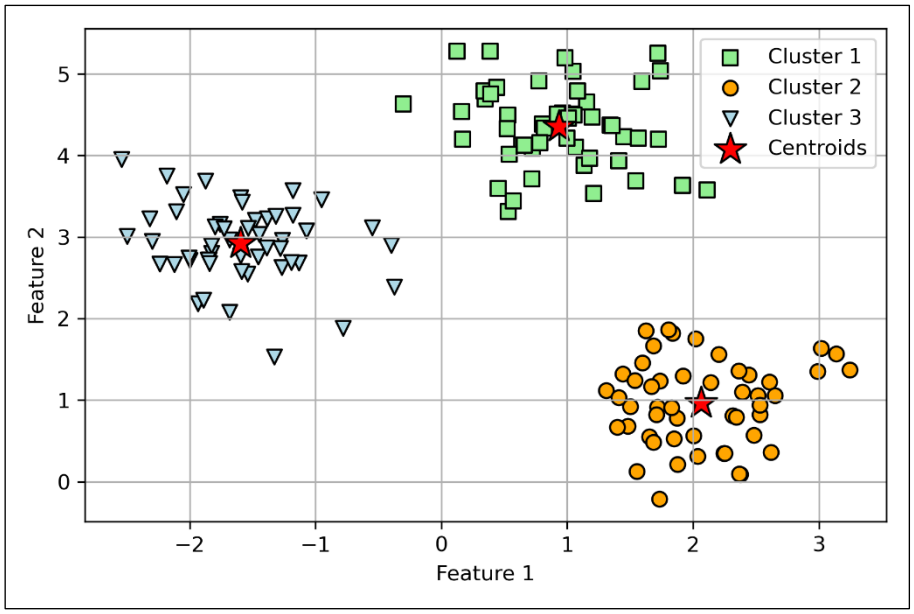

Linear (d=1), R^2=0.64
Quadratic (d=2), R^2=0.69
Cubic (d=3), R^2=0.70
Training points
Sale price in U.S. dollars
0
200000
400000
600000
0
2
4
6
8
10
Overall quality of the house

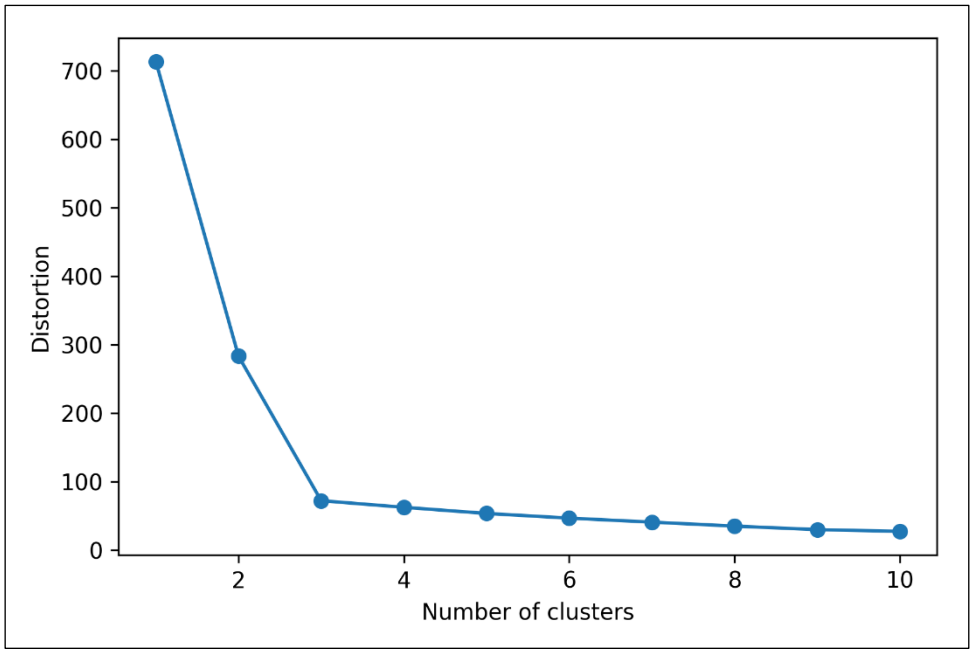

Sale price in U.S. dollars
700000
600000
500000
400000
300000
200000
100000
0
1000
2000
3000
4000
5000
Living area above ground in square feet

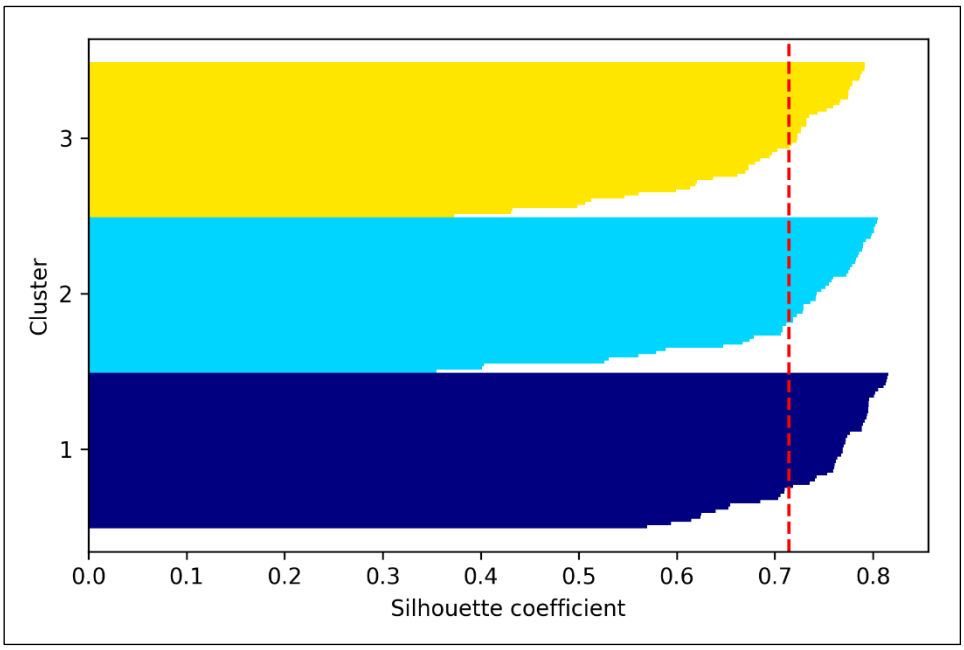

Test data
Training data
Residuals
200000
100000
0
−100000
0
200000
400000
600000
Predicted values
Predicted values

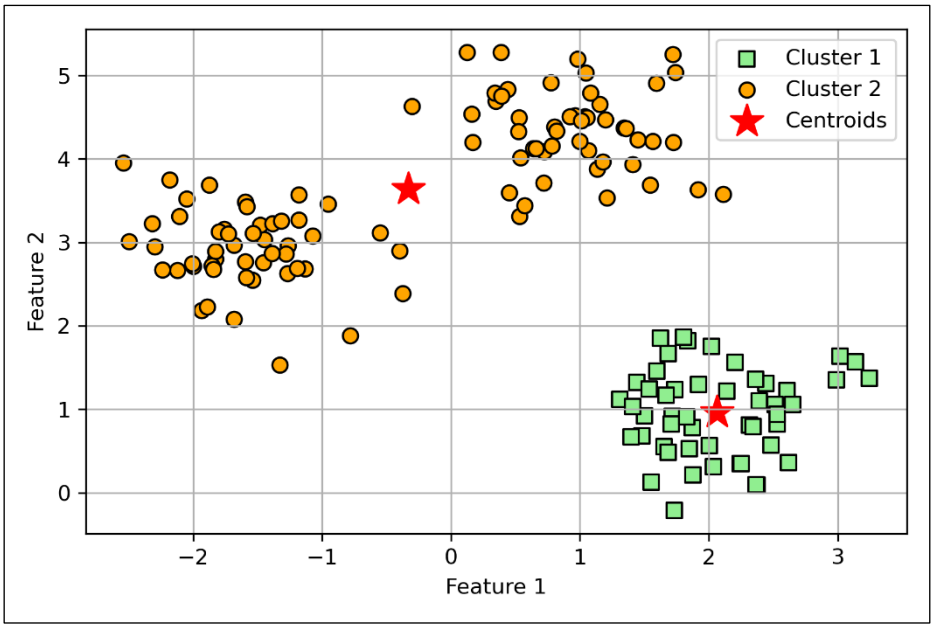
Chapter 10: Working with Unlabeled Data – Clustering Analysis

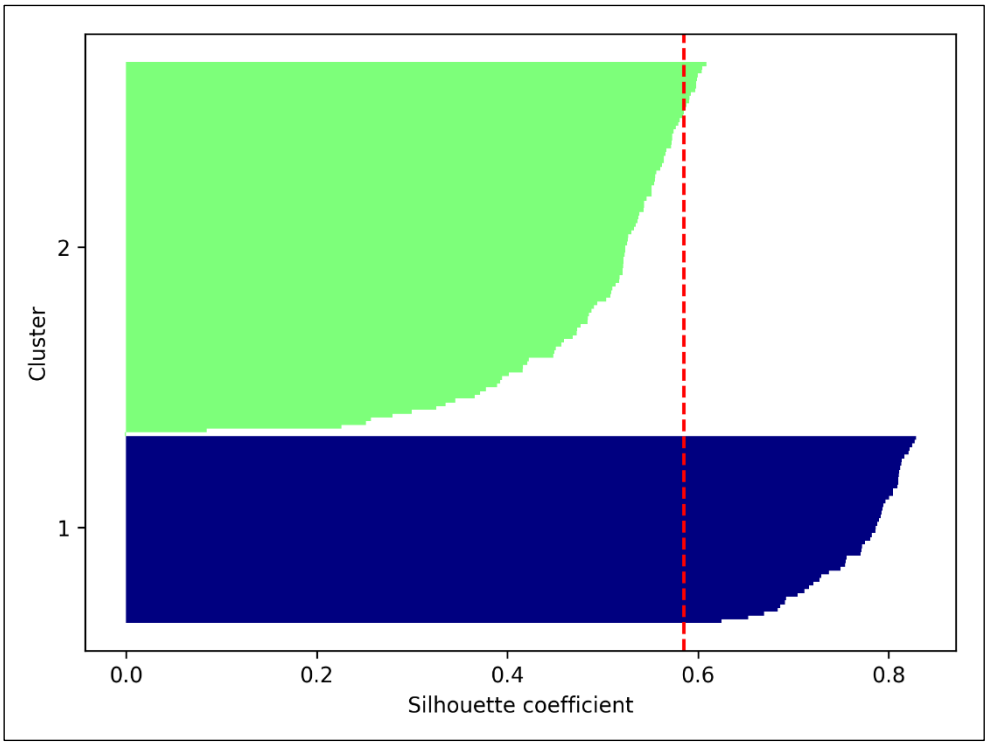


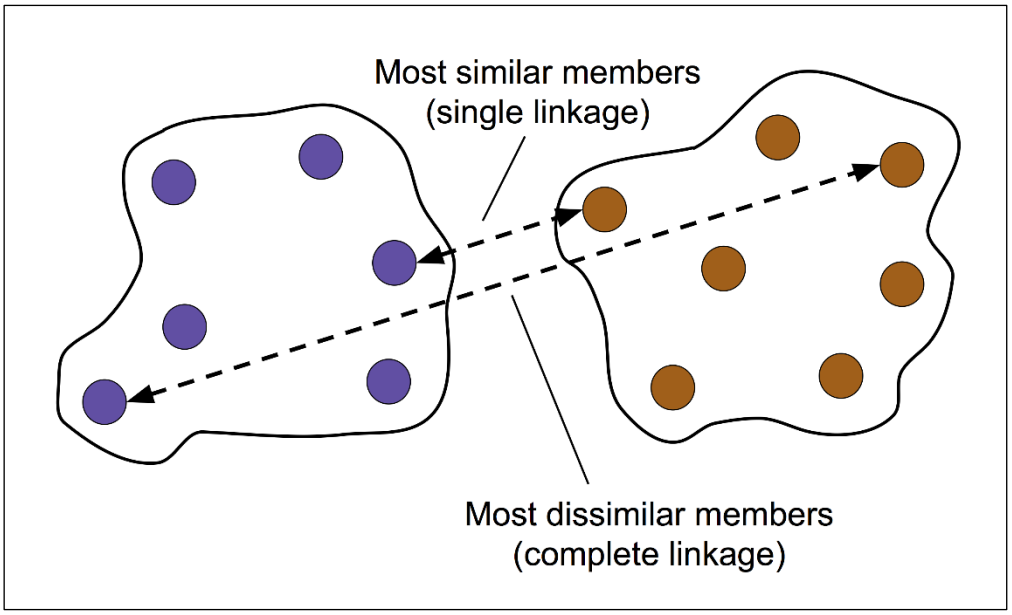



Distortion
700
600
500
400
300
200
100
0
2
4
6
8
10
Number of clusters


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
Feature 2
Feature 1
5
4
3
2
1
0
−2
−1
0
1
2
3

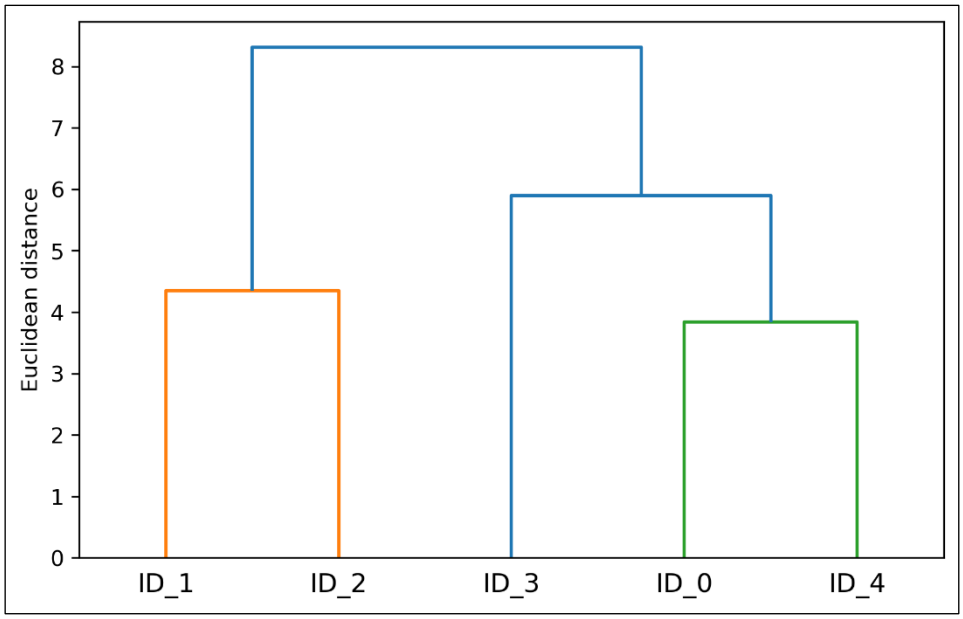

Cluster
2
1
0.0
0.2
0.4
0.6
0.8
Silhouette coefficient

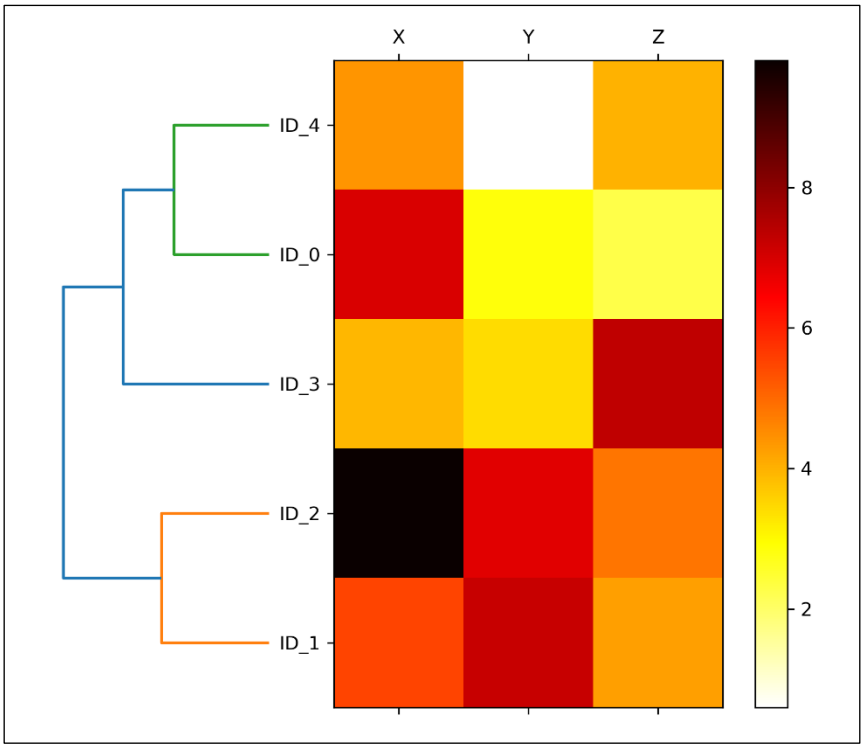


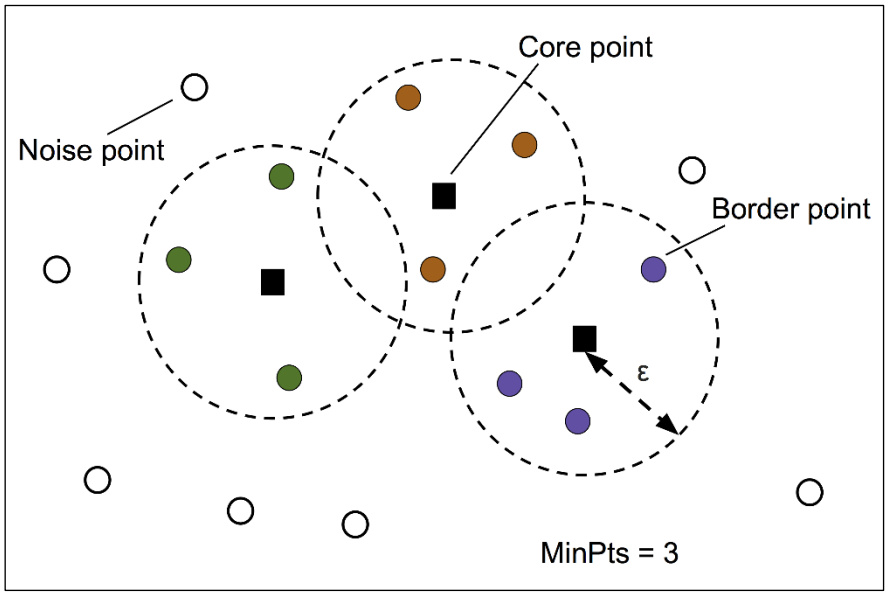
	X	Y	Z
ID_0	6.964692	2.861393	2.268515
ID_1	5.513148	7.194690	4.231065
ID_2	9.807642	6.848297	4.809319
ID_3	3.921175	3.431780	7.290497
ID_4	4.385722	0.596779	3.980443

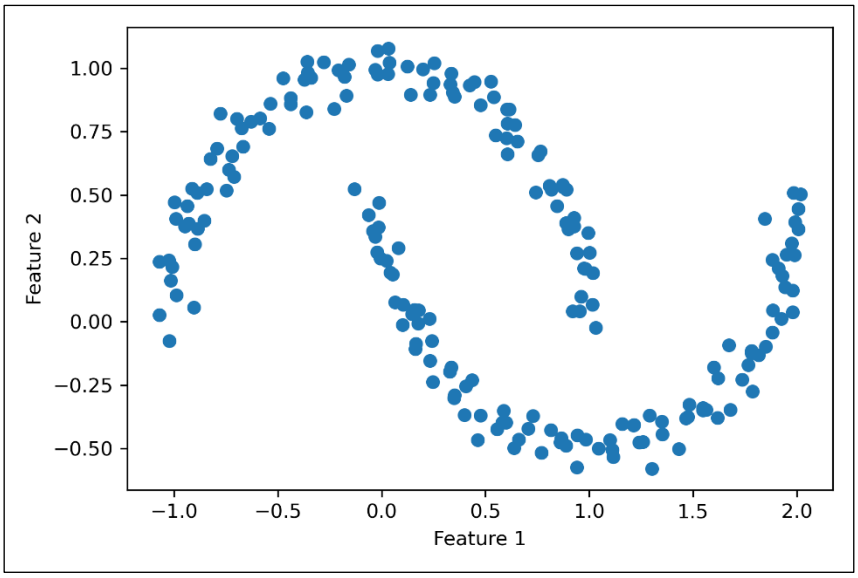
	ID_0	ID_1	ID_2	ID_3	ID_4
ID_0	0.000000	4.973534	5.516653	5.899885	3.835396
ID_1	4.973534	0.000000	4.347073	5.104311	6.698233
ID_2	5.516653	4.347073	0.000000	7.244262	8.316594
ID_3	5.899885	5.104311	7.244262	0.000000	4.382864
ID_4	3.835396	6.698233	8.316594	4.382864	0.000000

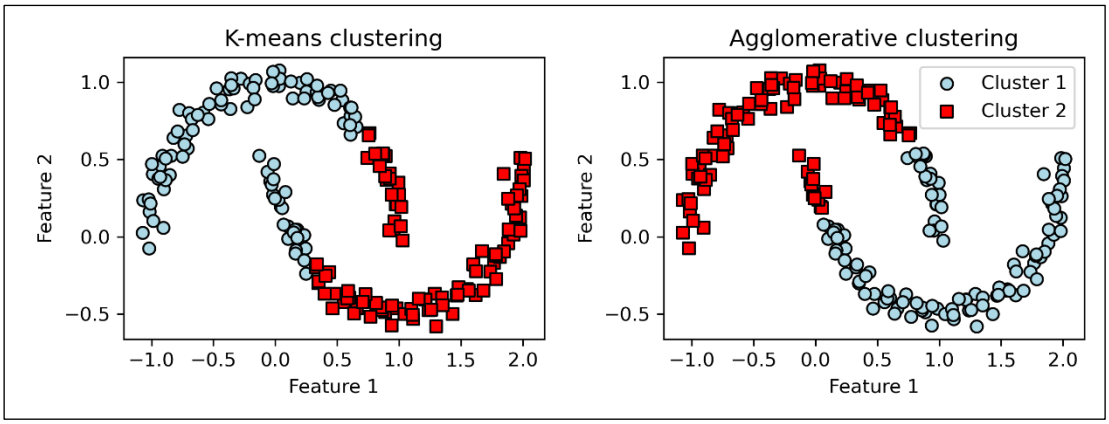
	row label 1	row label 2	distance	no. of items in clust.
cluster 1	0.0	4.0	3.835396	2.0
cluster 2	1.0	2.0	4.347073	2.0
cluster 3	3.0	5.0	5.899885	3.0
cluster 4	6.0	7.0	8.316594	5.0

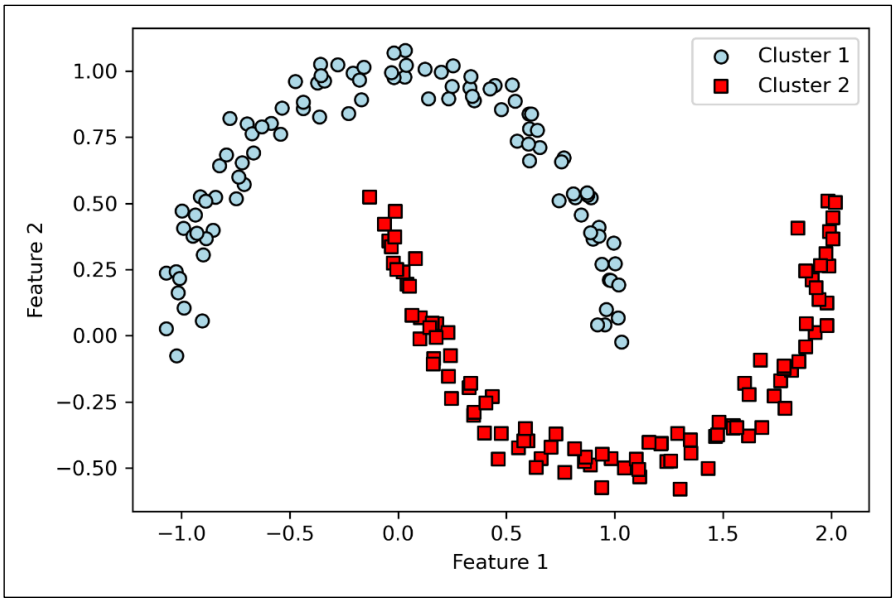



X
Y
Z
ID_4
ID_0
ID_3
ID_2
ID_1
8
6
4
2

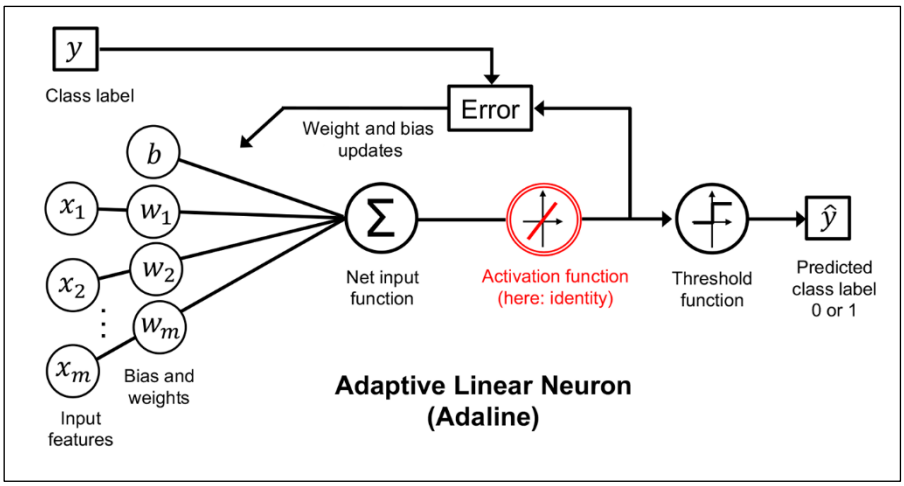

Core point
Noise point
Border point
ε
MinPts = 3

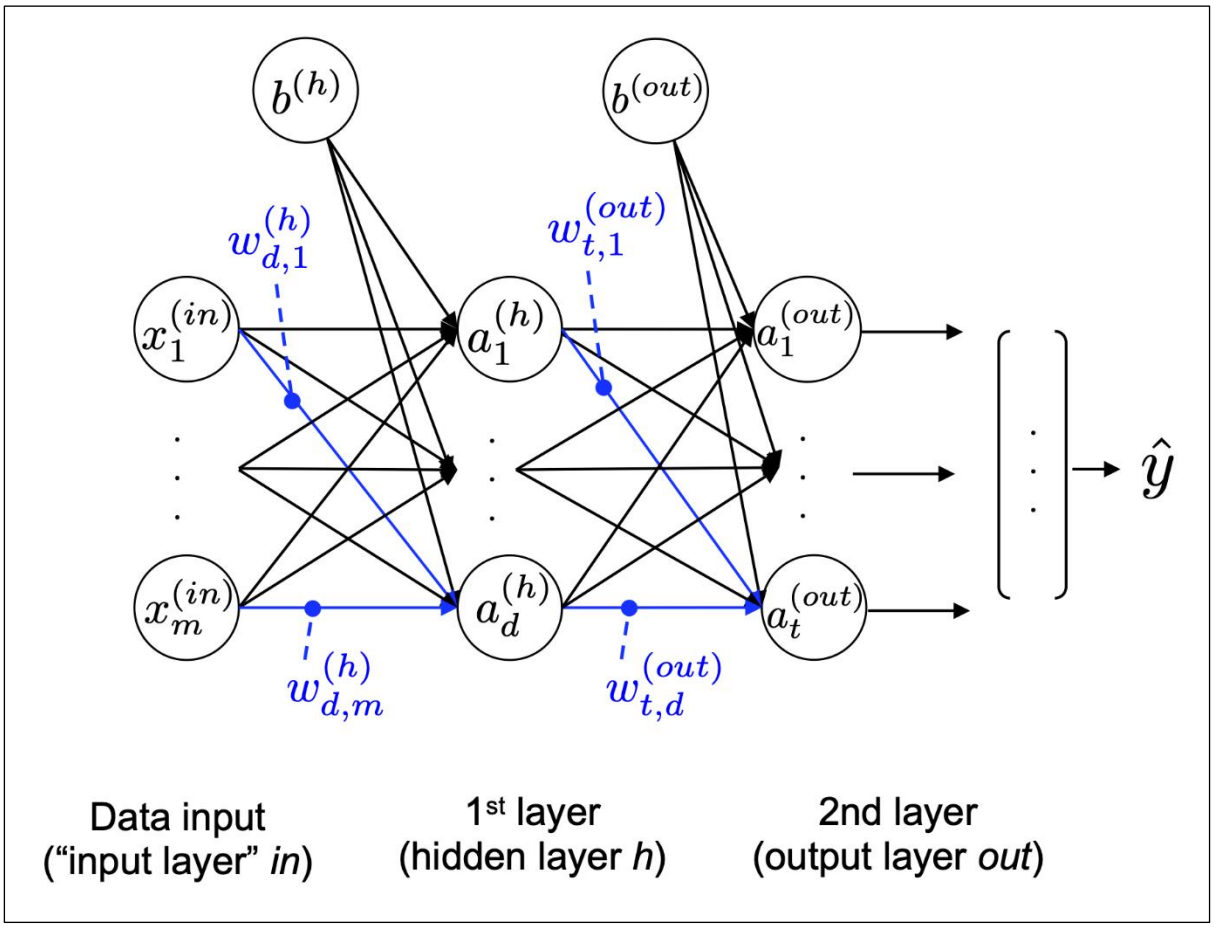

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
Feature 2
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Feature 1

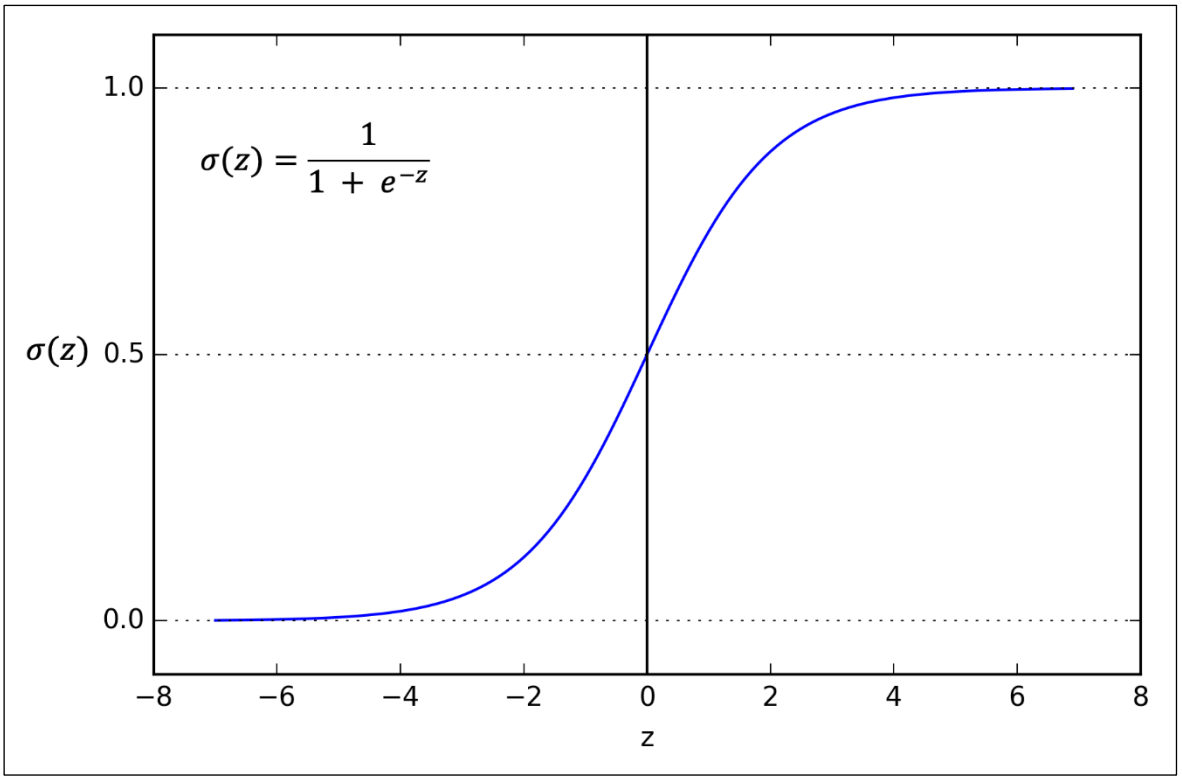

K-means clustering
Agglomerative clustering
1.0
0.5
0.0
−0.5
Feature 2
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Feature 1
Cluster 1
Cluster 2

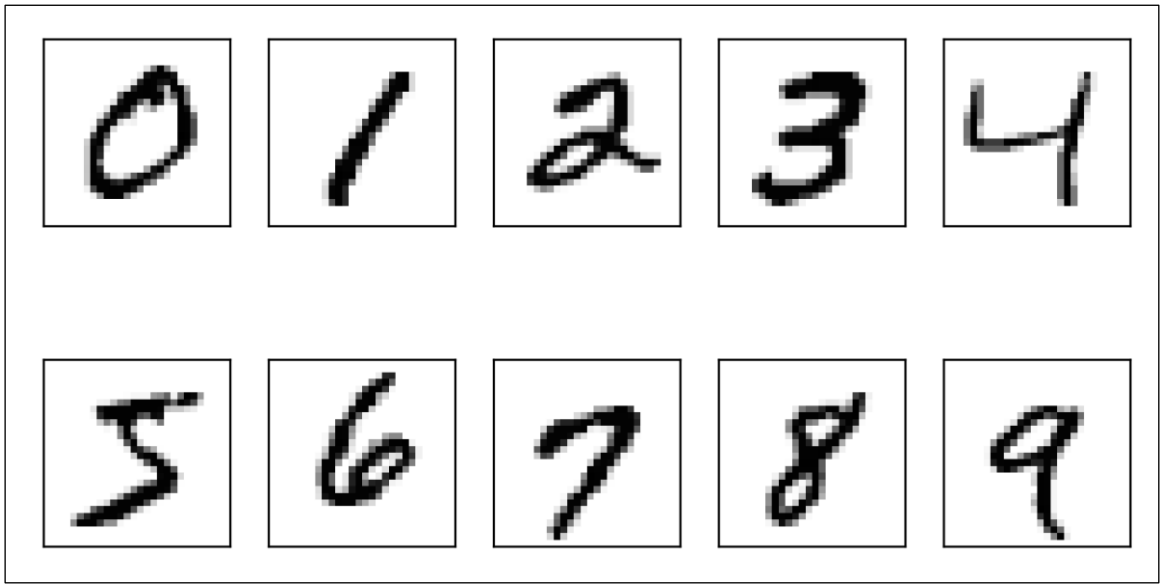

Cluster 1
Cluster 2
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
Feature 2
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Feature 1

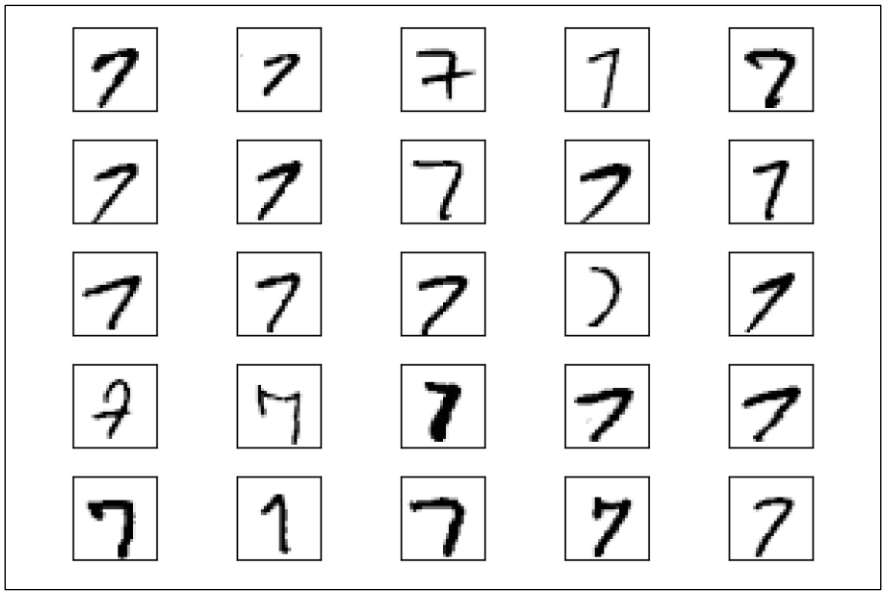
Chapter 11: Implementing a Multilayer Artificial Neural Network from Scratch

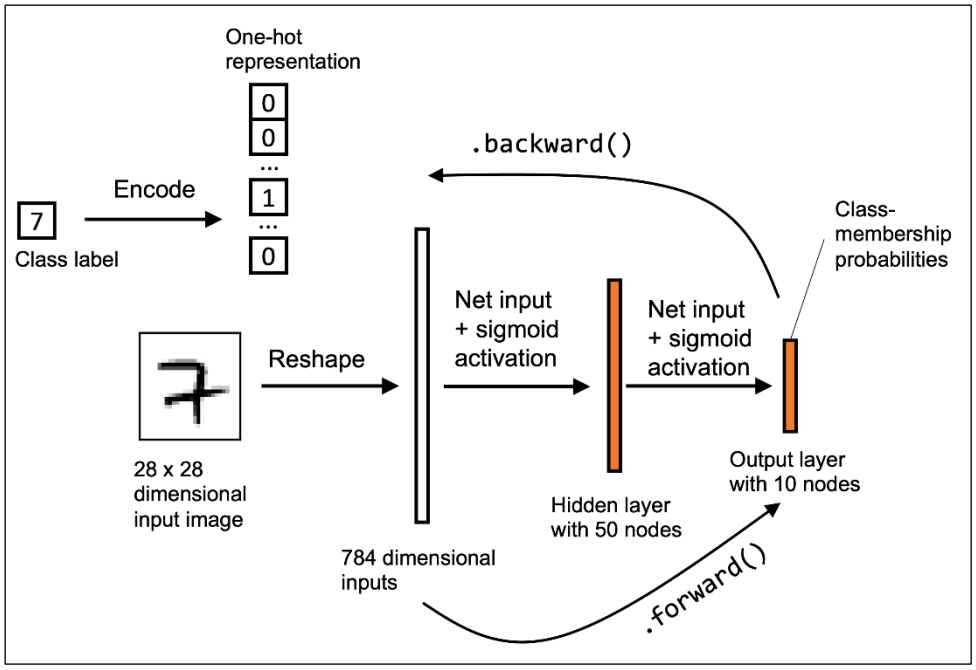


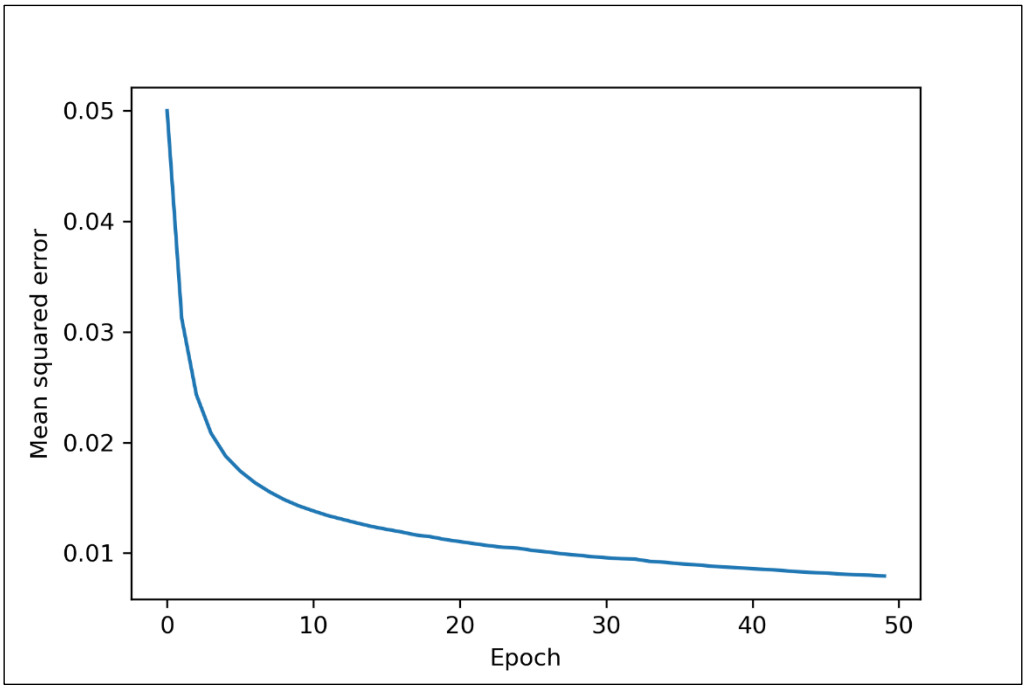


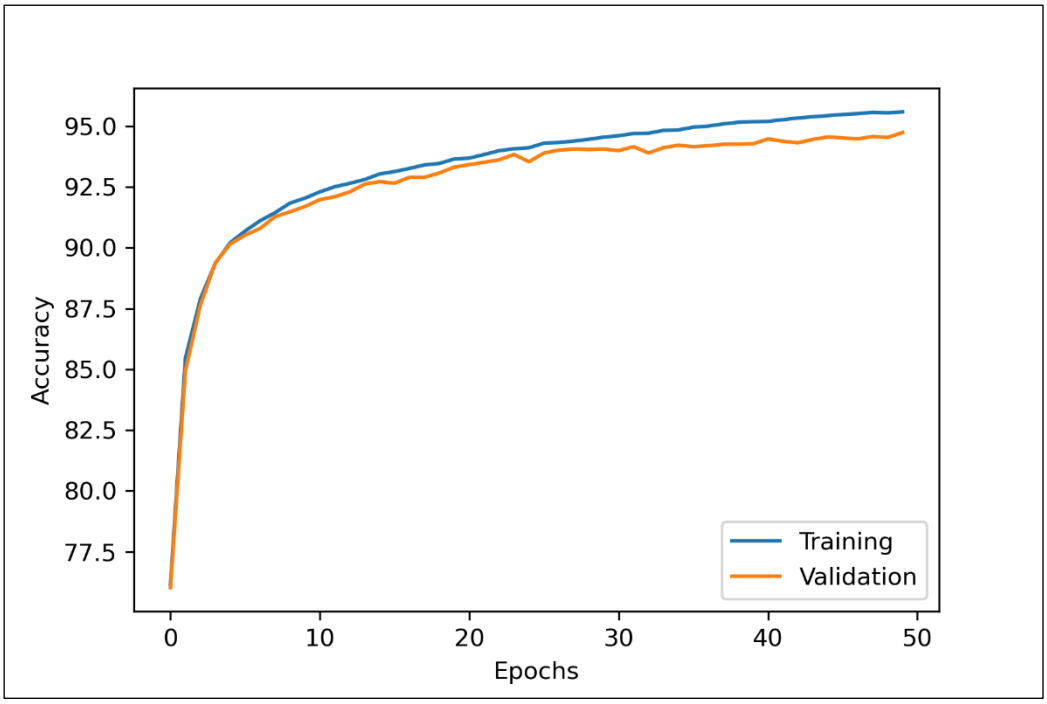

$\sigma(z) = \frac{1}{1 + e^{-z}}$
1.0
0.5
0.0
$\sigma(z)$
−8
−6
−4
−2
0
2
4
6
8
z



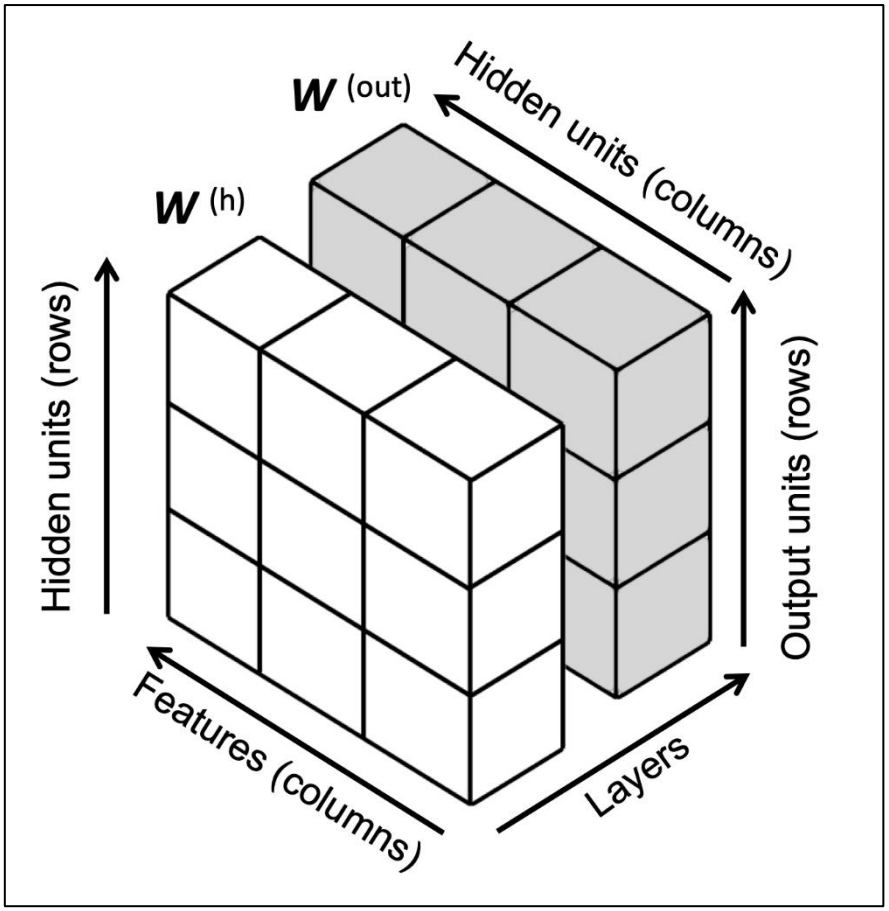


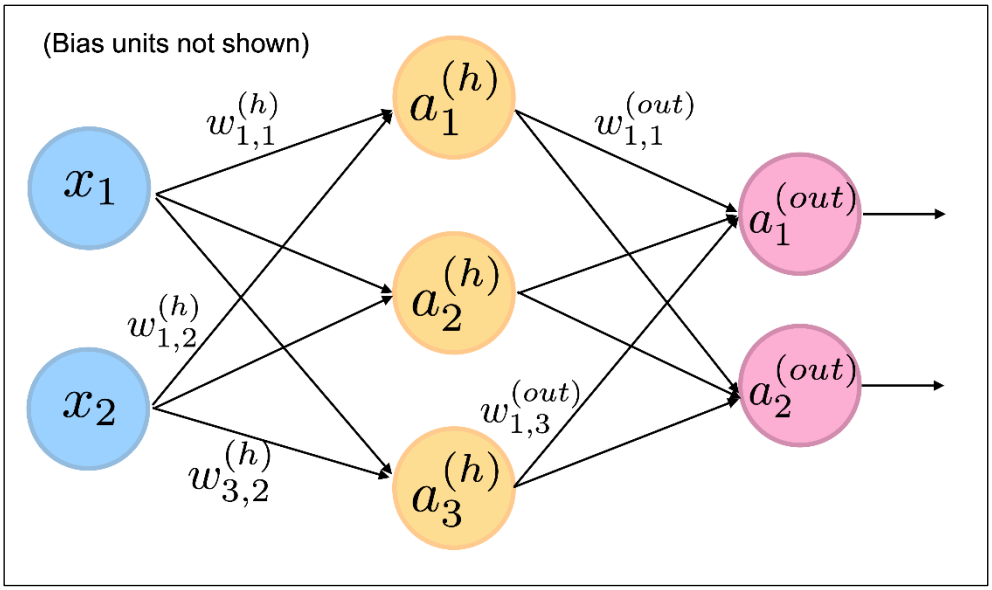

One-hot representation
0
0
...
1
...
0
Encode
7
Class label
.backward()
Class-membership probabilities
Net input + sigmoid activation
Net input + sigmoid activation
Reshape
28 x 28 dimensional input image
784 dimensional inputs
Hidden layer with 50 nodes
Output layer with 10 nodes
.forward()

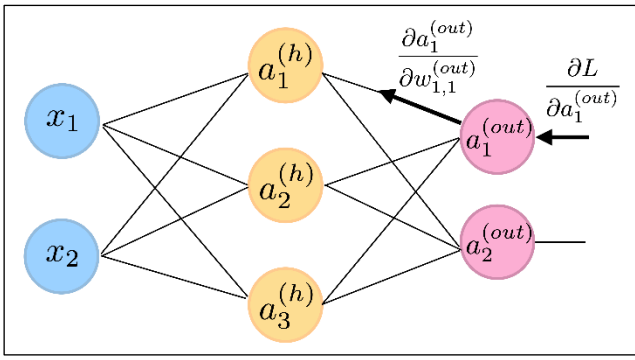

0.05
0.04
0.03
0.02
0.01
Mean squared error
0
10
20
30
40
50
Epoch


95.0
92.5
90.0
87.5
85.0
82.5
80.0
77.5
Accuracy
0
10
20
30
40
50
Epochs
Training
Validation


1) True: 5
Predicted: 0
2) True: 3
Predicted: 7
3) True: 1
Predicted: 9
4) True: 2
Predicted: 7
5) True: 4
Predicted: 8
6) True: 5
Predicted: 9
7) True: 5
Predicted: 3
8) True: 8
Predicted: 5
9) True: 5
Predicted: 4
10) True: 5
Predicted: 7
11) True: 5
Predicted: 0
12) True: 9
Predicted: 3
13) True: 5
Predicted: 3
14) True: 4
Predicted: 6
15) True: 9
Predicted: 0
16) True: 2
Predicted: 4
17) True: 9
Predicted: 5
18) True: 8
Predicted: 5
19) True: 7
Predicted: 2
20) True: 7
Predicted: 2
21) True: 8
Predicted: 1
22) True: 6
Predicted: 5
23) True: 9
Predicted: 4
24) True: 2
Predicted: 8
25) True: 3
Predicted: 2

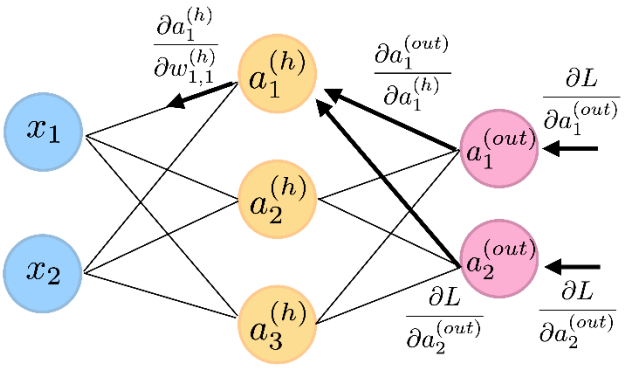

W (out)
Hidden units (columns)
W (h)
Hidden units (rows)
Output units (rows)
Features (columns)
Layers


(Bias units not shown)
x_1
x_2
$a_1^{(h)}$
$a_2^{(h)}$
$a_3^{(h)}$
$a_1^{(out)}$
$a_2^{(out)}$
$w_{1,1}^{(h)}$
$w_{1,2}^{(h)}$
$w_{3,2}^{(h)}$
$w_{1,1}^{(out)}$
$w_{1,3}^{(out)}$



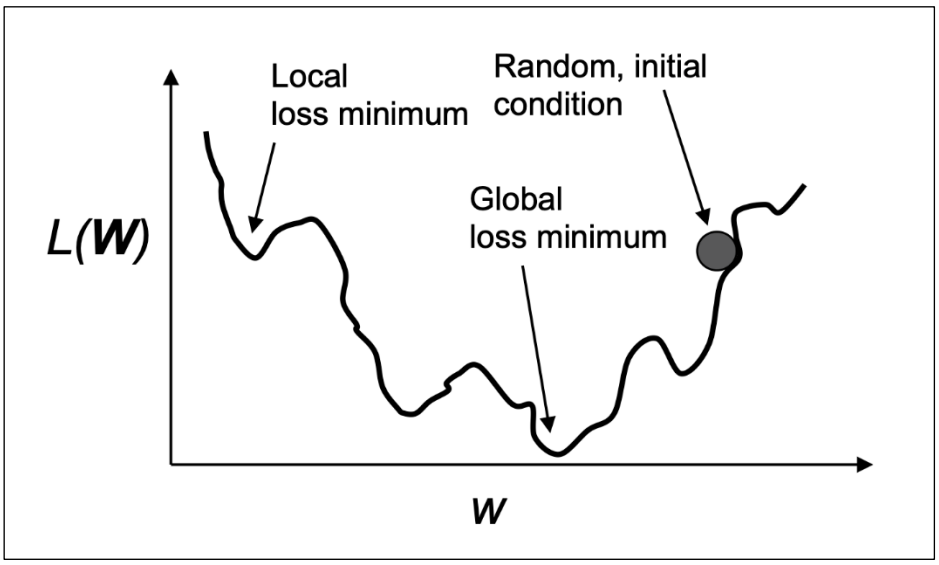
Gradient for output layer weight:

$$\frac{\partial L}{\partial w_{1,1}^{(out)}} = \frac{\partial L}{\partial a_1^{(out)}} \cdot \frac{\partial a_1^{(out)}}{\partial w_{1,1}^{(out)}}$$



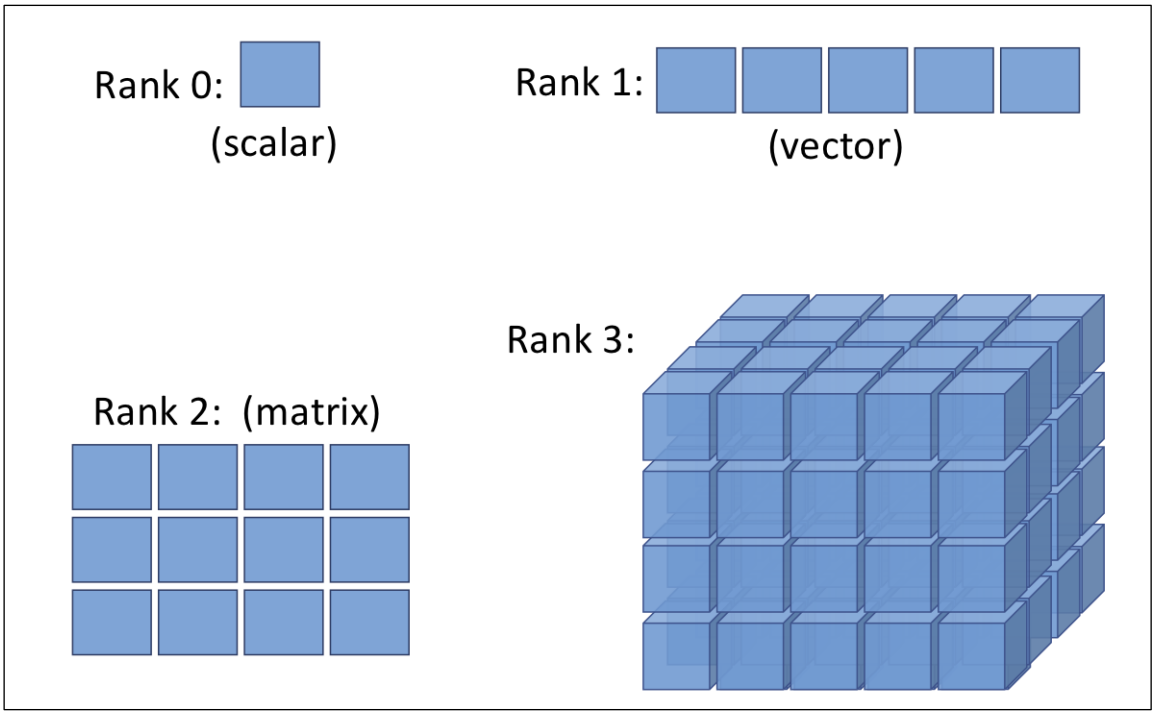
Gradient for hidden layer weight:

$$\frac{\partial L}{\partial w_{1,1}^{(h)}} = \frac{\partial L}{\partial a_1^{(out)}} \cdot \frac{\partial a_1^{(out)}}{\partial a_1^{(h)}} \cdot \frac{\partial a_1^{(h)}}{\partial w_{1,1}^{(h)}} + \frac{\partial L}{\partial a_2^{(out)}} \cdot \frac{\partial a_2^{(out)}}{\partial a_1^{(h)}} \cdot \frac{\partial a_1^{(h)}}{\partial w_{1,1}^{(h)}}$$

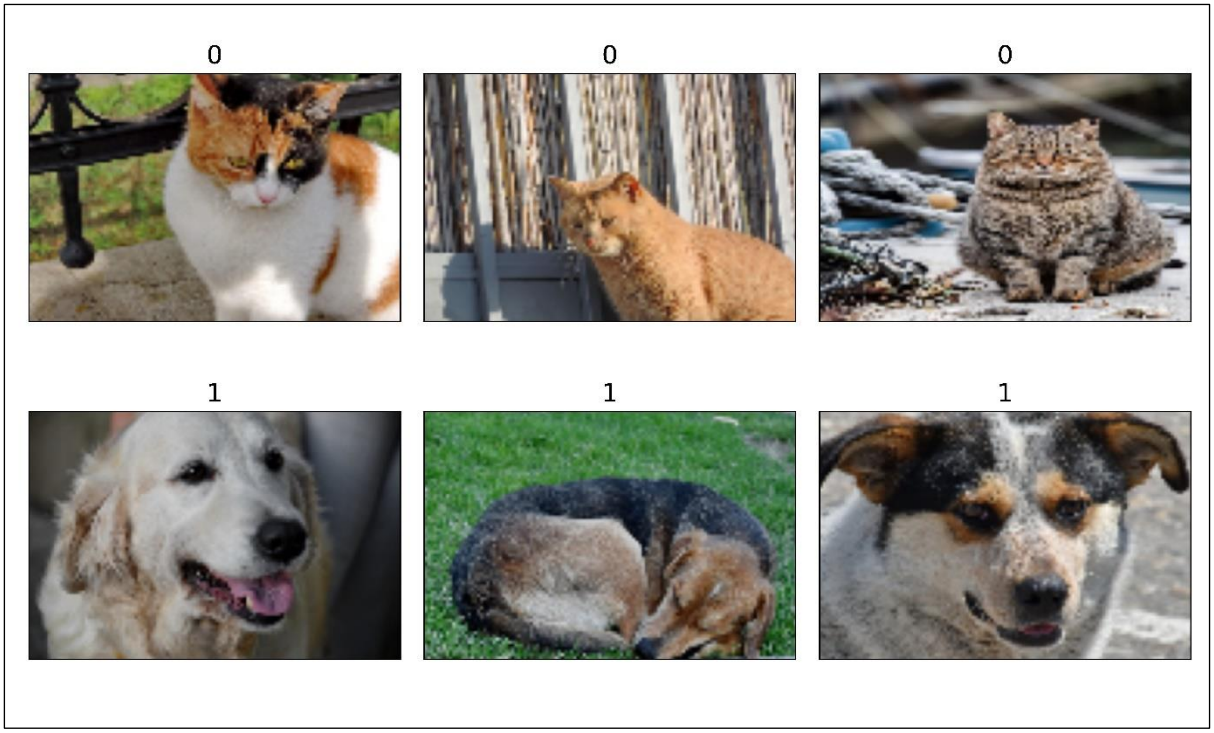


Chapter 12: Parallelizing Neural Network Training with PyTorch

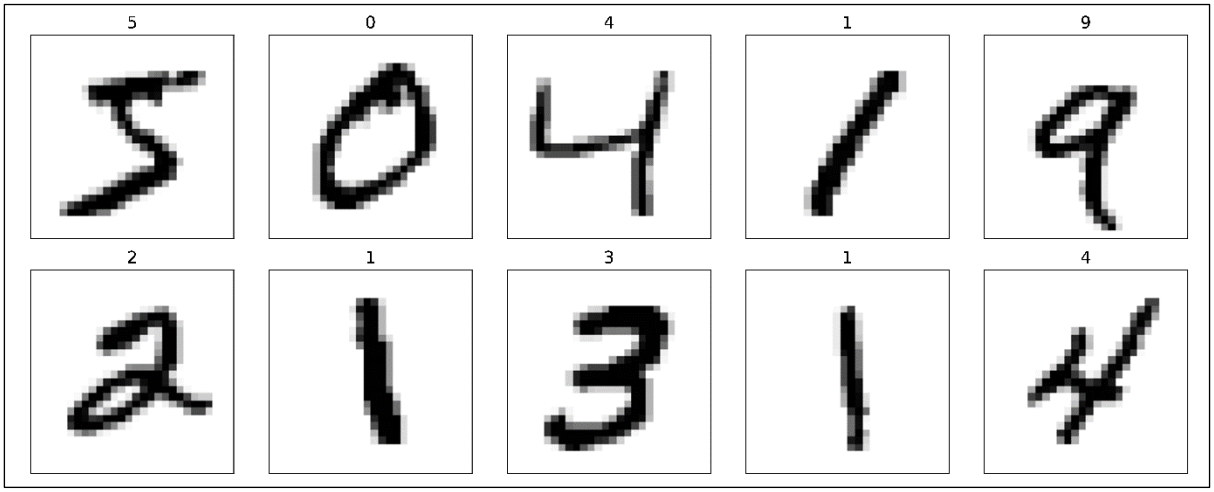
Specifications	Intel® Core™ i9-11900KB Processor	NVIDIA GeForce® RTX™ 3080 Ti
Base Clock Frequency	3.3 GHz	1.37 GHz
Cores	16 (32 threads)	10240
Memory Bandwidth	45.8 GB/s	912.1 GB/s
Floating-Point Calculations	742 GFLOPS	34.10 TFLOPS
Cost	~ $540.00	~ $1200.00

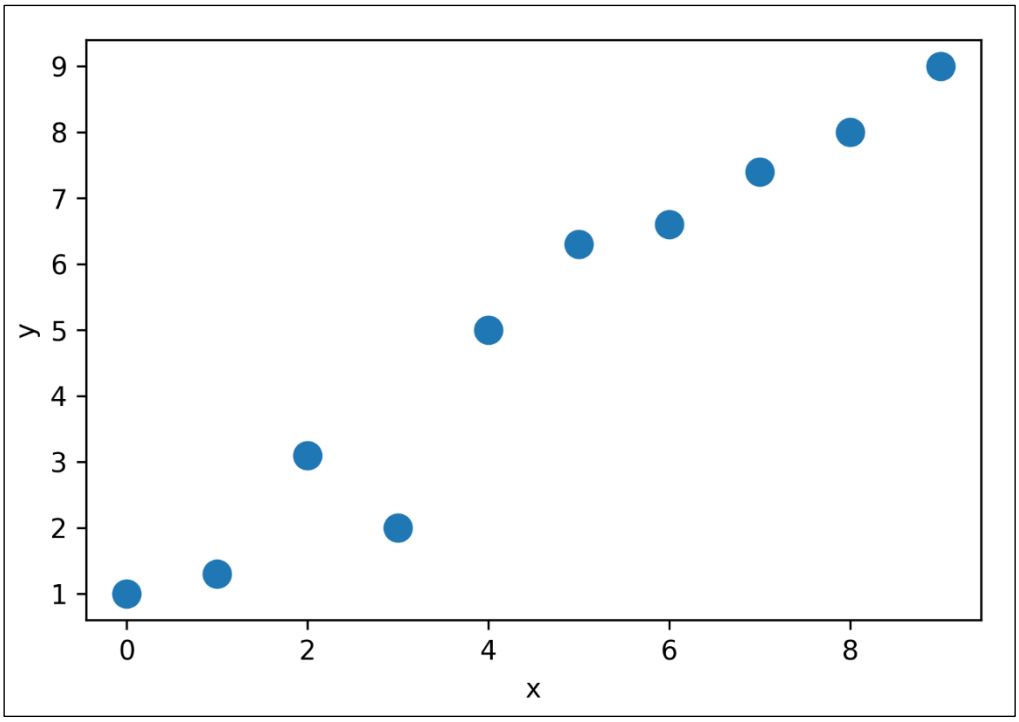


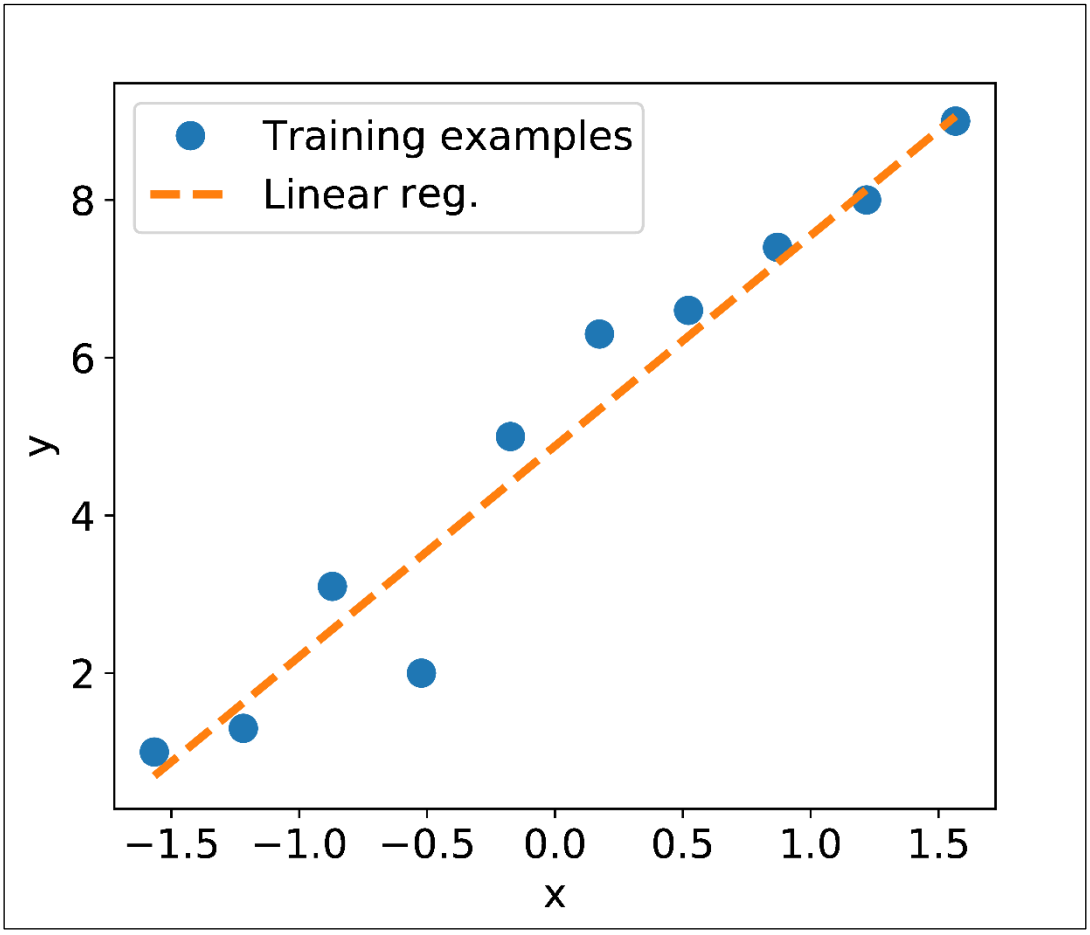

cat-01.jpg
cat-02.jpg
cat-03.jpg
dog-01.jpg
dog-02.jpg
dog-03.jpg

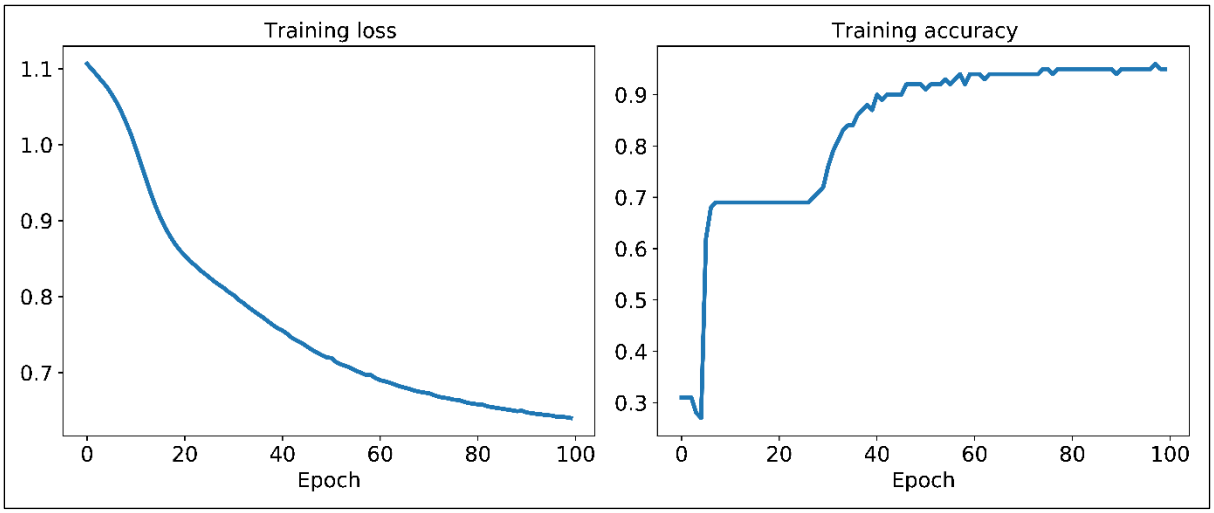

0
0
0
1
1
1

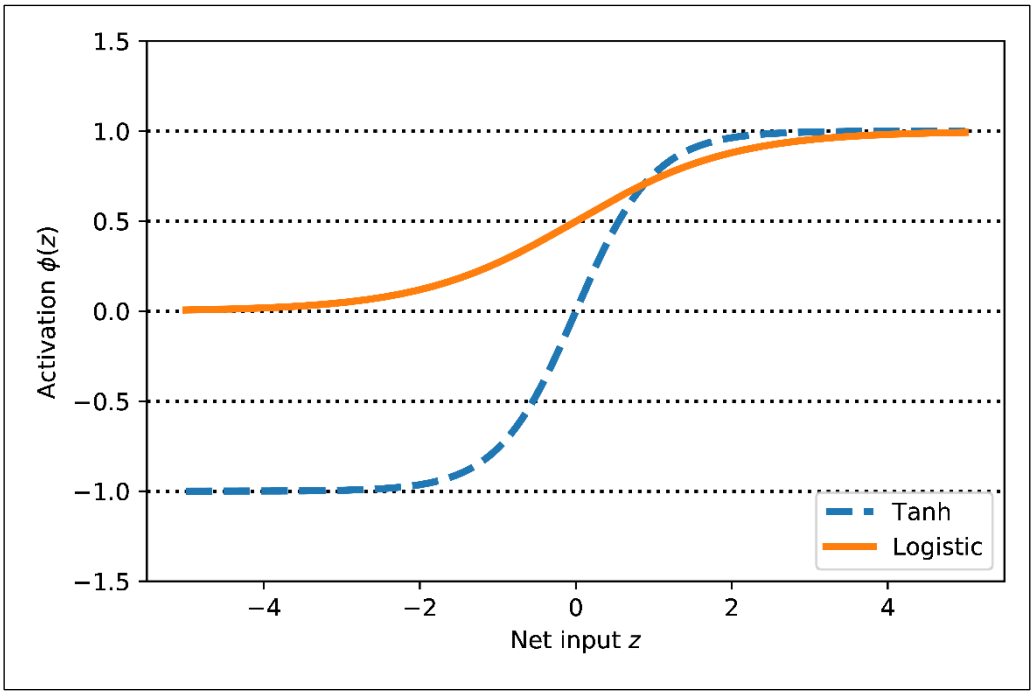

1 1 0 0 0 0
0 0 1 0 1 1
1 1 0 1 1 1

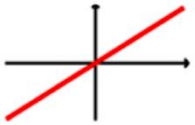
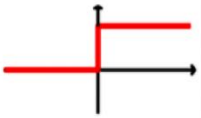
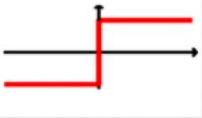
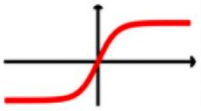
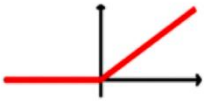

5 0 4 1 9
2 1 3 1 4


9
8
7
6
5
4
3
2
1
y
0
2
4
6
8
x

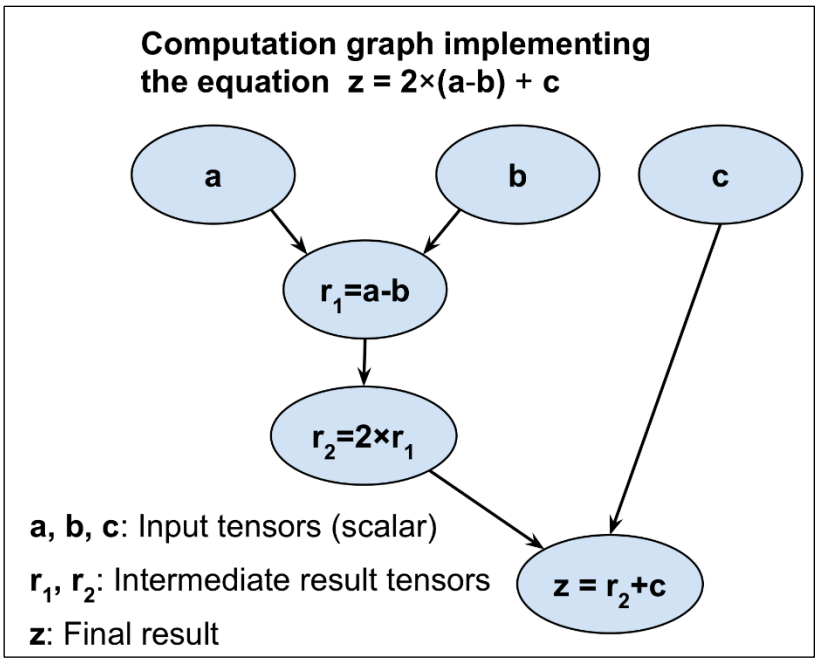

Training examples
Linear reg.
8
6
4
2
y
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
x

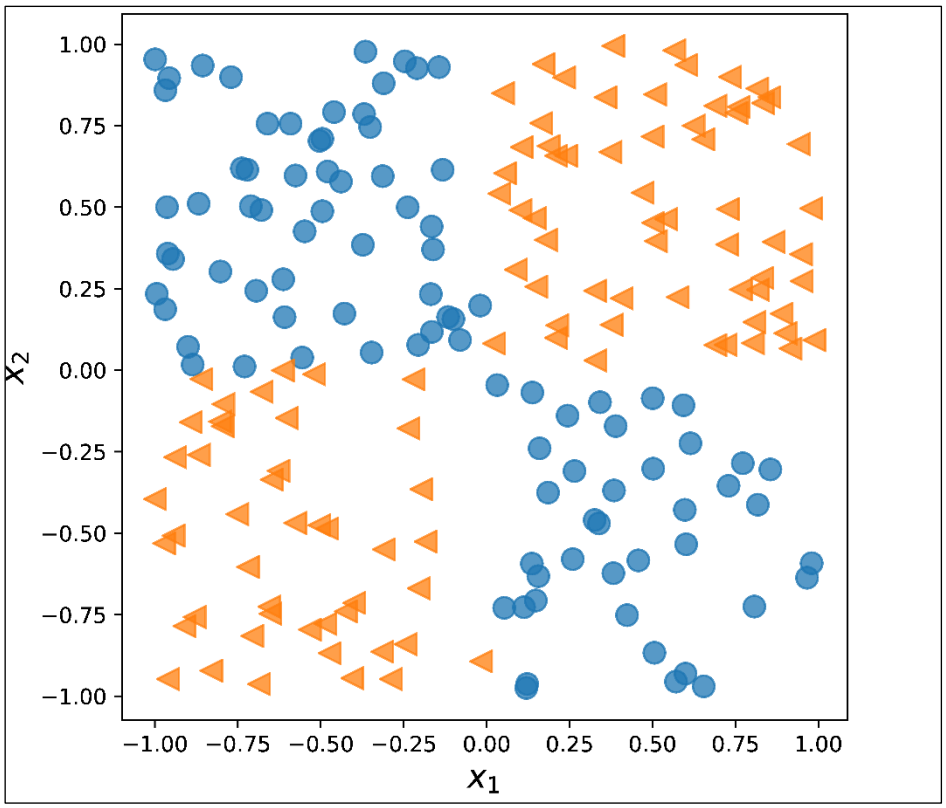

Training loss
1.1
1.0
0.9
0.8
0.7
0
20
40
60
80
100
Epoch
Training accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100
Epoch

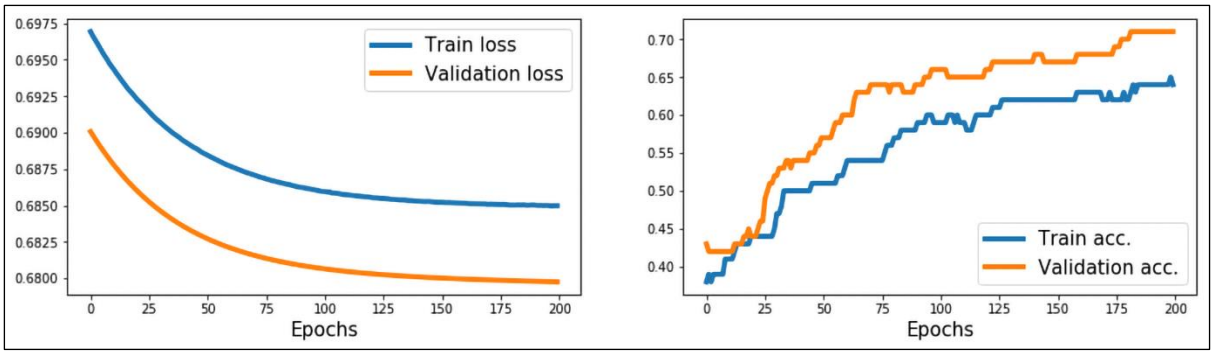

1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
Activation $\phi(z)$
−4
−2
0
2
4
Net input z
Tanh
Logistic

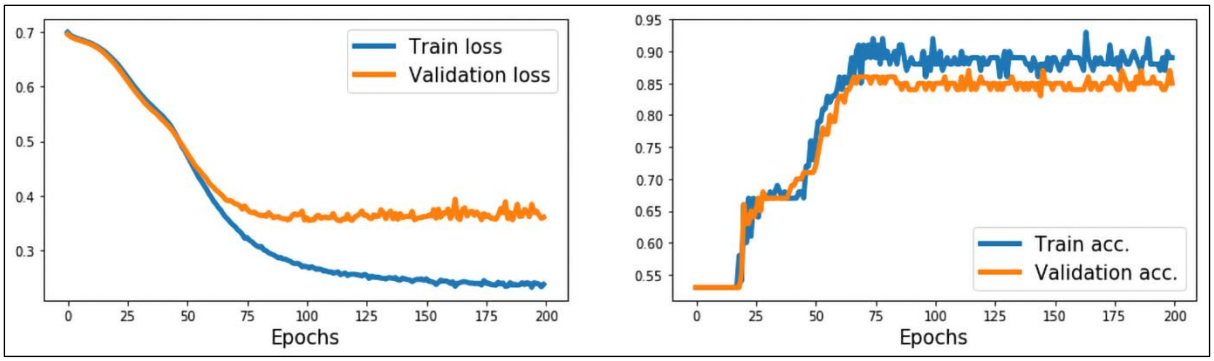
Activation function	Equation	Example	1D graph
Linear	$\sigma(z) = z$	Adaline, linear regression	
Unit step (Heaviside function)	$\sigma(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\sigma(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Piece-wise linear	$\sigma(z) = \begin{cases} 0 & z \leq -\frac{1}{2} \\ z + \frac{1}{2} & -\frac{1}{2} \leq z \leq \frac{1}{2} \\ 1 & z \geq \frac{1}{2} \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\sigma(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, multilayer NN	
Hyperbolic tangent (tanh)	$\sigma(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multilayer NN, RNNs	
ReLU	$\sigma(z) = \begin{cases} 0 & z < 0 \\ z & z > 0 \end{cases}$	Multilayer NN, CNNs	

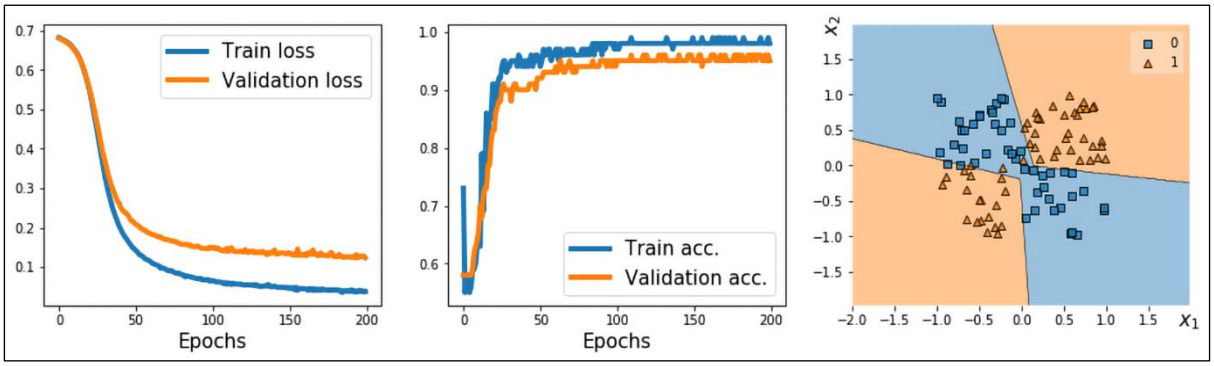
Chapter 13: Going Deeper – The Mechanics of PyTorch

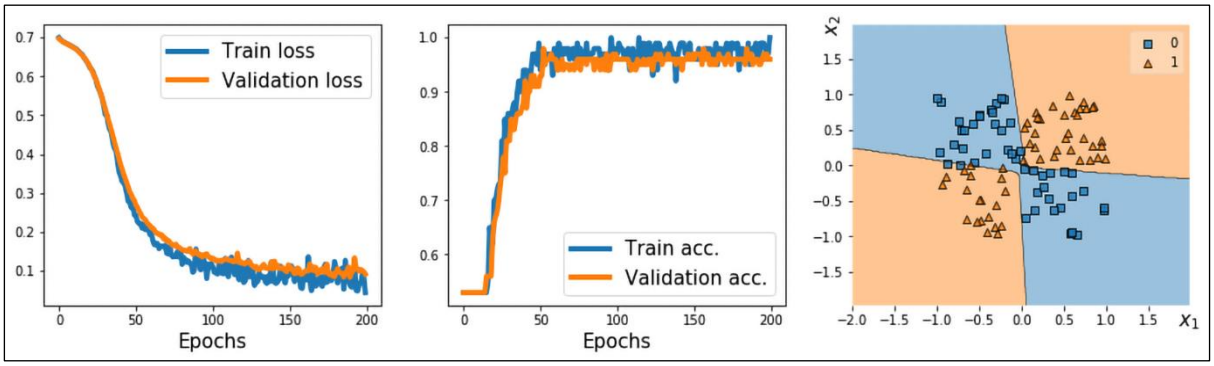


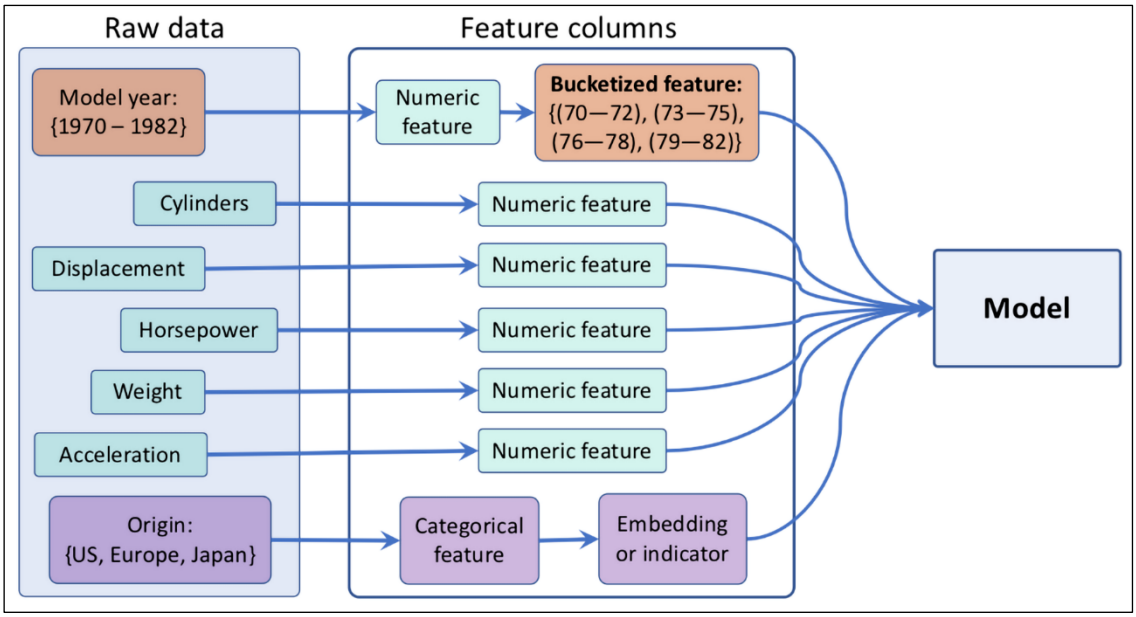



Train loss
Validation loss
Epochs
Train acc.
Validation acc.
Epochs

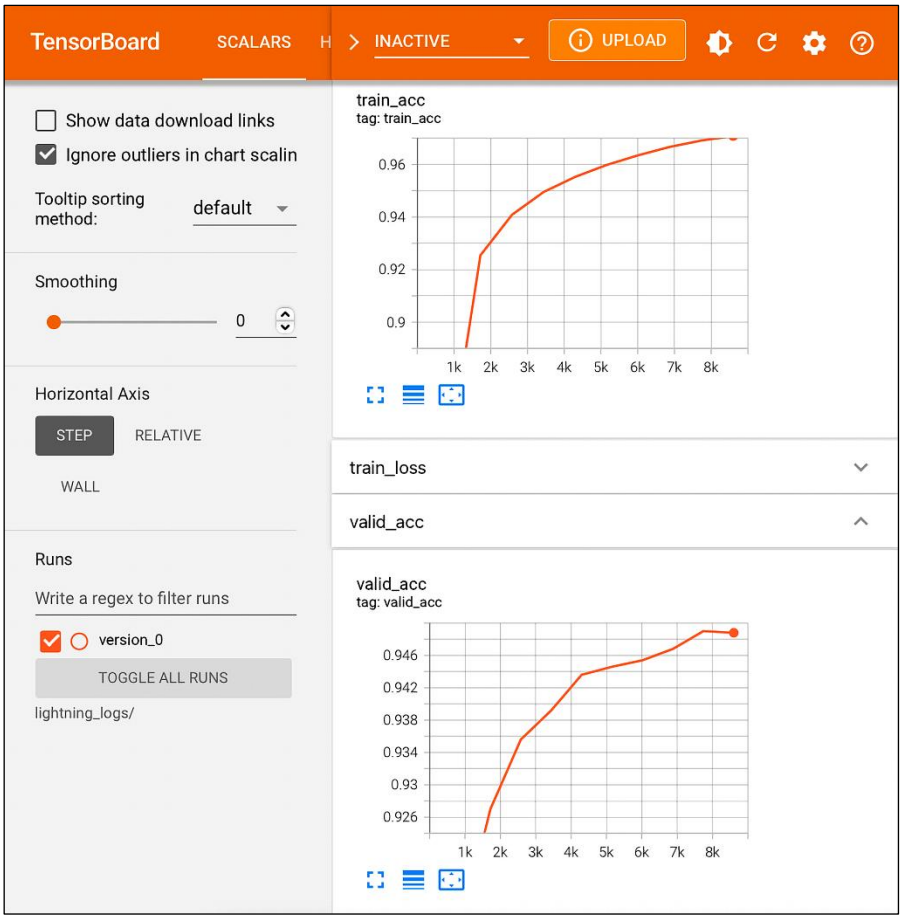

Train loss
Validation loss
Epochs
Train acc.
Validation acc.
Epochs

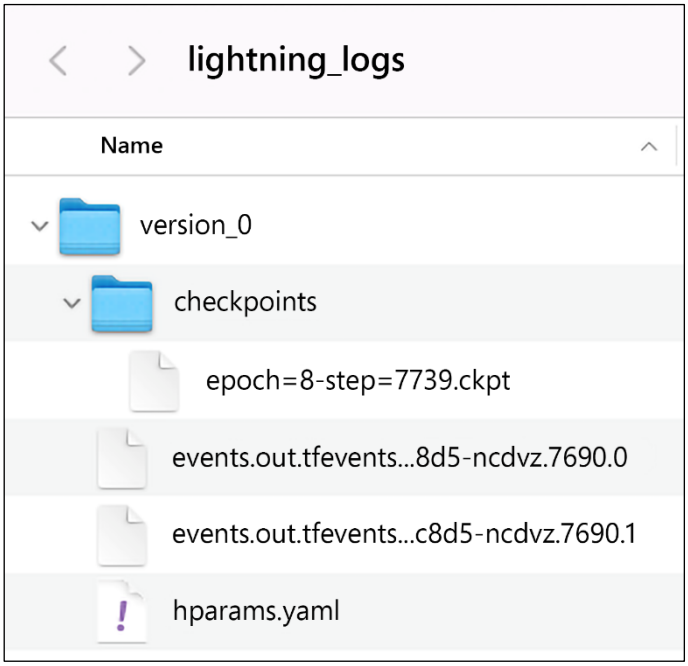

Train loss
Validation loss
Epochs
Train acc.
Validation acc.
Epochs
0
1
x_1
x_2

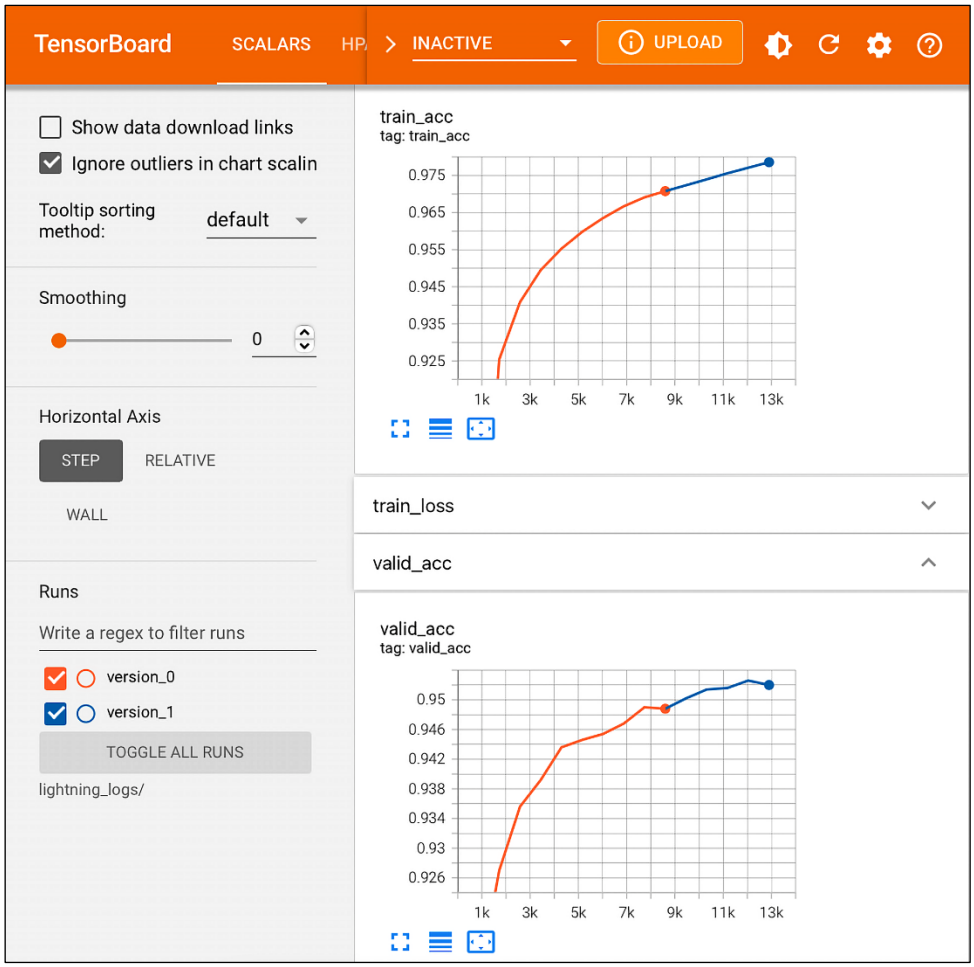

Train loss
Validation loss
Epochs
Train acc.
Validation acc.
Epochs
0
1
x_1
x_2



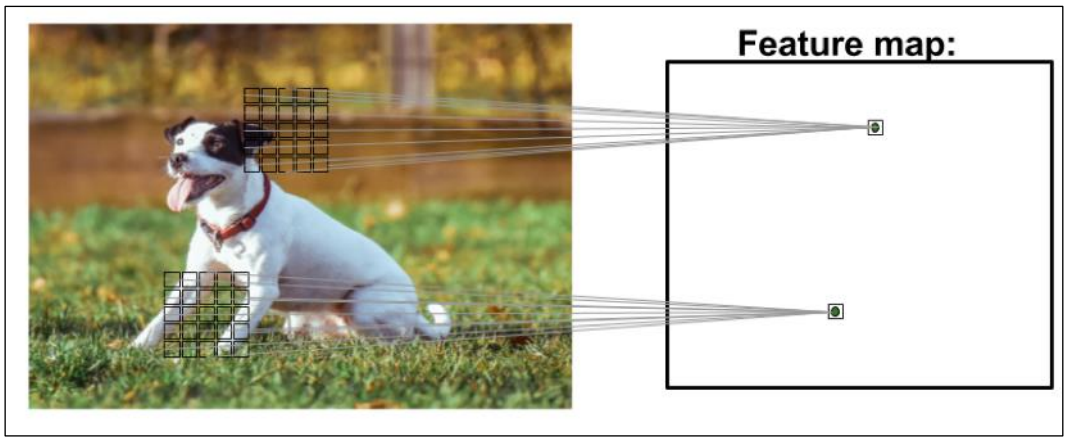
	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	ModelYear	Origin
203	28.0	-0.824303	-0.901020	-0.736562	-0.950031	0.255202	76	3
255	19.4	0.351127	0.413800	-0.340982	0.293190	0.548737	78	1
72	13.0	1.526556	1.144256	0.713897	1.339617	-0.625403	72	1
235	30.5	-0.824303	-0.891280	-1.053025	-1.072585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

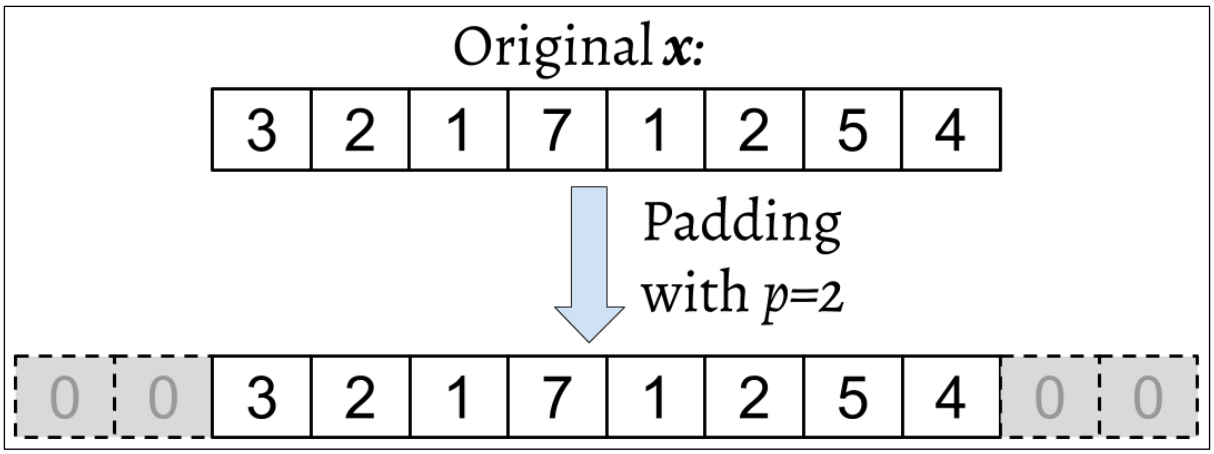

TensorBoard
SCALARS
INACTIVE
UPLOAD
Show data download links
Ignore outliers in chart scalin
Tooltip sorting method:
default
Smoothing
0
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
version_0
TOGGLE ALL RUNS
lightning_logs/
train_acc
tag: train_acc
0.96
0.94
0.92
0.9
1k
2k
3k
4k
5k
6k
7k
8k
train_loss
valid_acc
valid_acc
tag: valid_acc
0.946
0.942
0.938
0.934
0.93
0.926

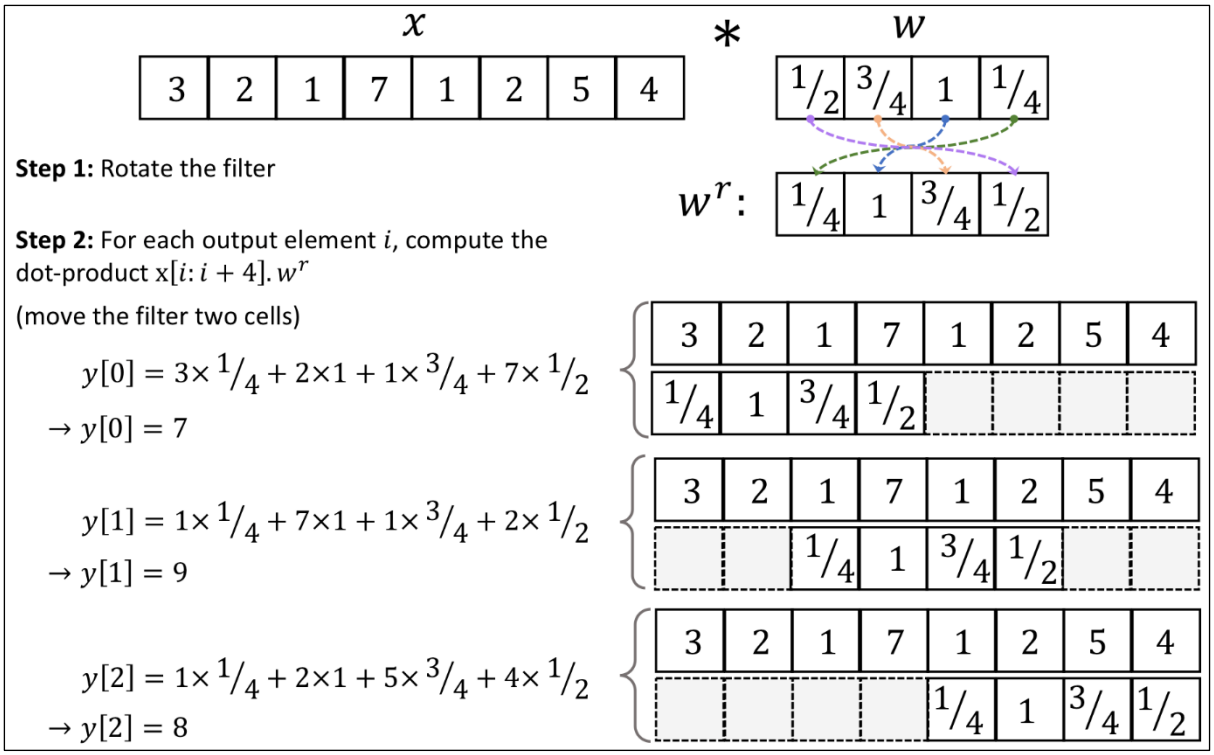

lightning_logs
Name
version_0
checkpoints
epoch=8-step=7739.ckpt
events.out.tfevents...8d5-ncdvz.7690.0
events.out.tfevents...c8d5-ncdvz.7690.1
hparams.yaml

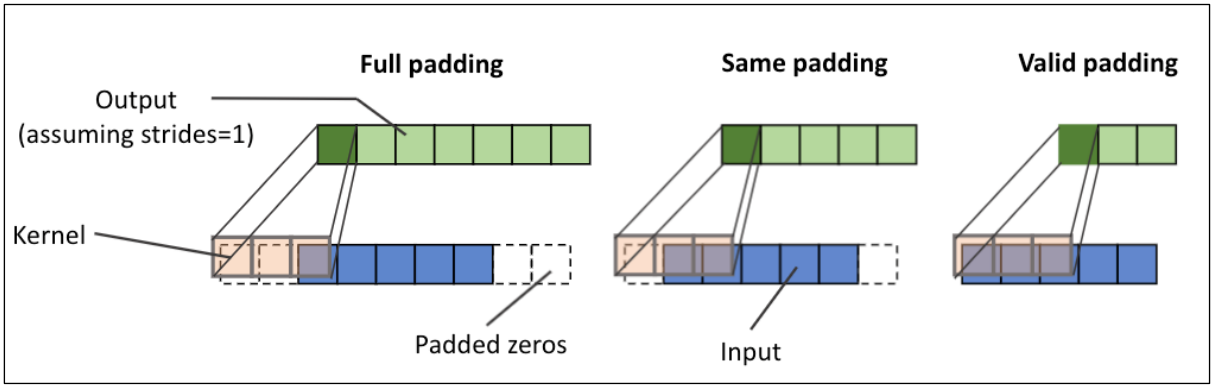

TensorBoard
SCALARS
HP
INACTIVE
UPLOAD
Show data download links
Ignore outliers in chart scalin
Tooltip sorting method:
default
Smoothing
0
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
version_0
version_1
TOGGLE ALL RUNS
lightning_logs/
train_acc
tag: train_acc
0.975
0.965
0.955
0.945
0.935
0.925
1k
3k
5k
7k
9k
11k
13k
train_loss
valid_acc
valid_acc
tag: valid_acc
0.95
0.946
0.942
0.938
0.934
0.93
0.926
1k
3k
5k
7k
9k
11k
13k

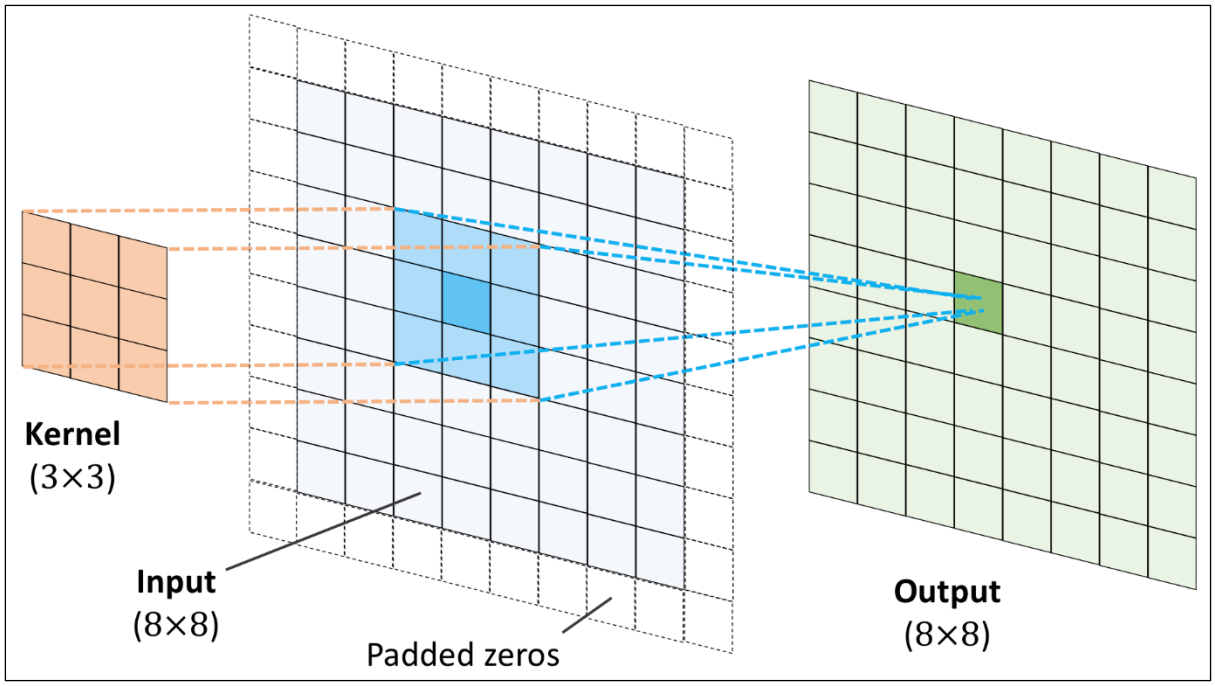
Chapter 14: Classifying Images with Deep Convolutional Neural Networks

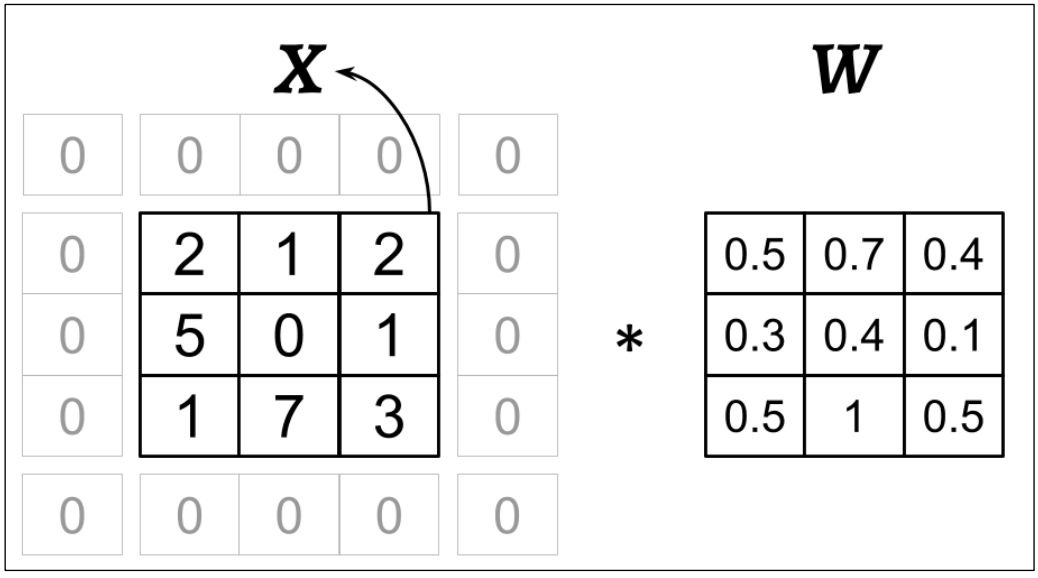


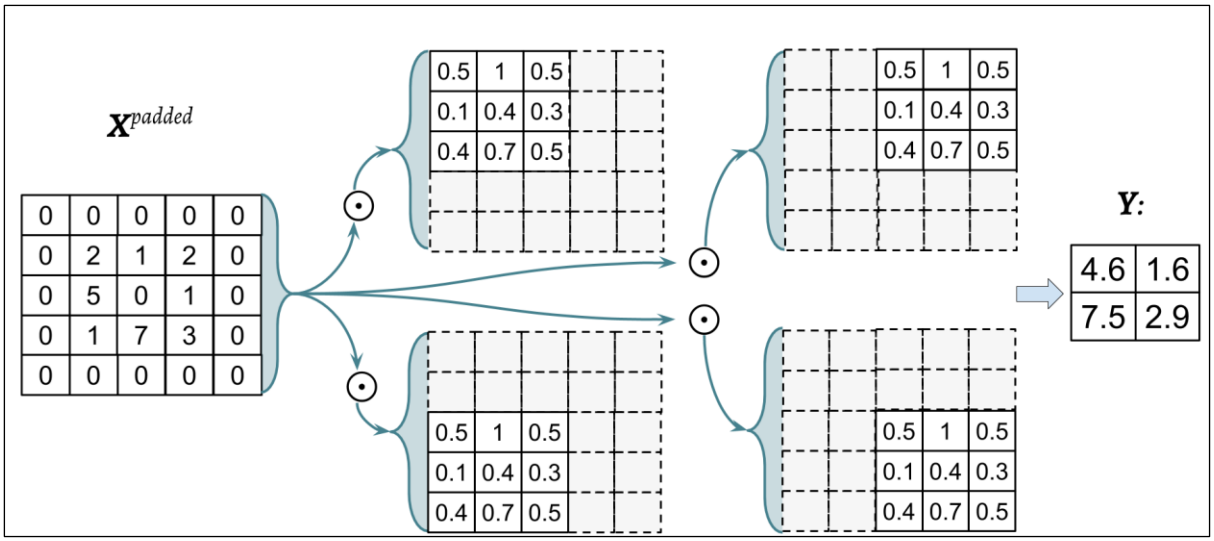


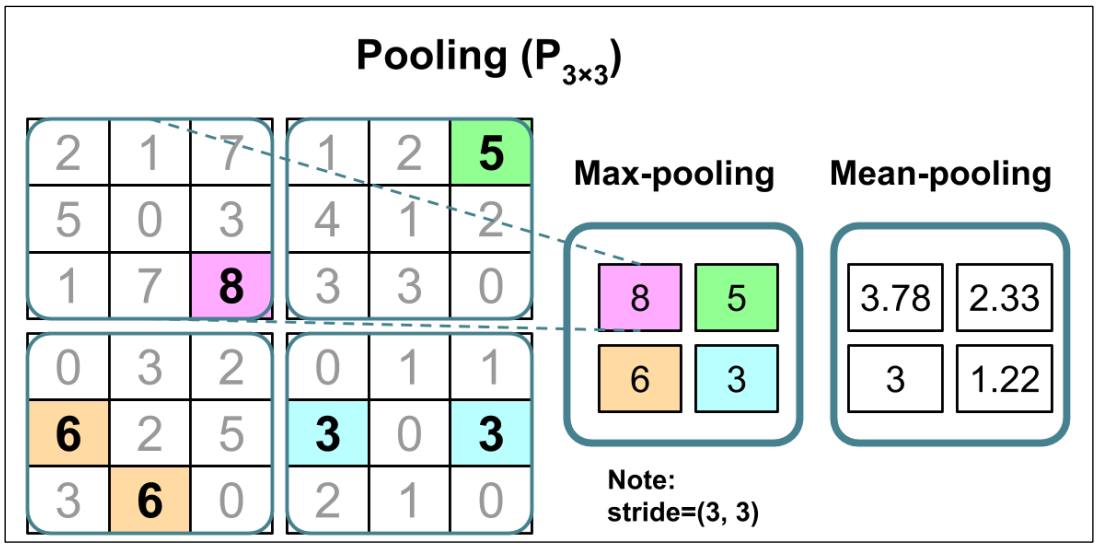

x
3 2 1 7 1 2 5 4
$*$
w
$1/2$ $3/4$ 1 $1/4$
w^r:
$1/4$ 1 $3/4$ $1/2$
Step 1: Rotate the filter
Step 2: For each output element i, compute the dot-product $x[i:i+4].w^r$
(move the filter two cells)
$y[0] = 3\times 1/4 + 2\times 1 + 1\times 3/4 + 7\times 1/2$
$\rightarrow y[0] = 7$
$y[1] = 1\times 1/4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

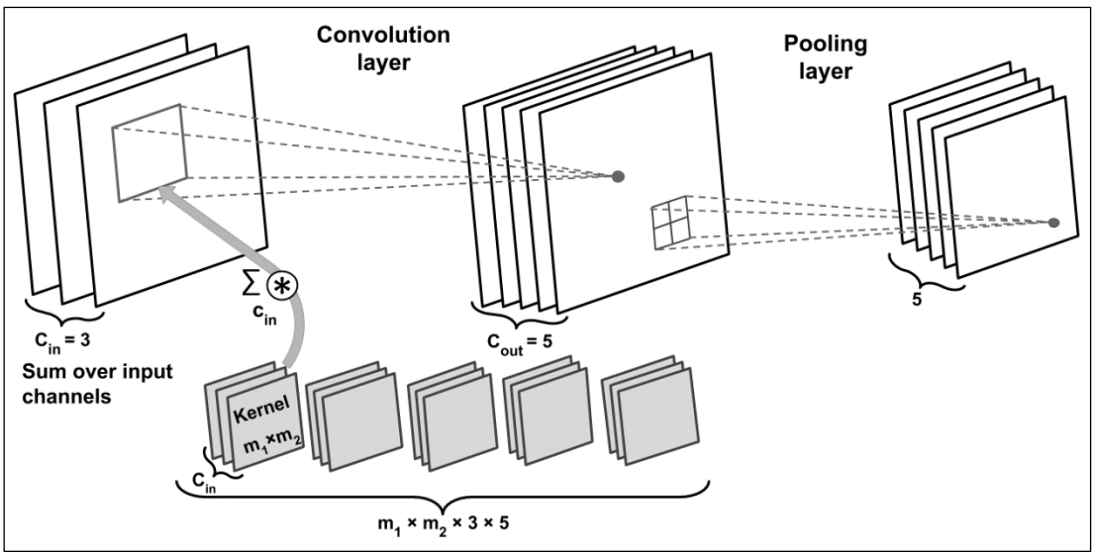

Full padding
Same padding
Valid padding
Output
(assuming strides=1)
Kernel
Padded zeros
Input

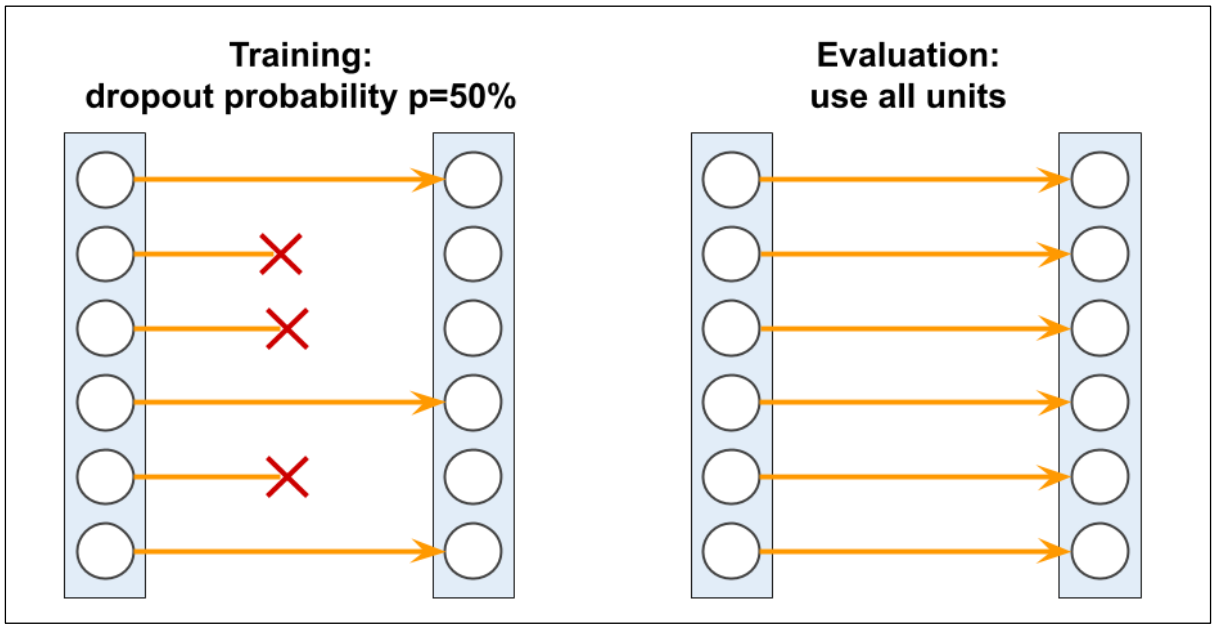

Kernel
(3×3)
Input
(8×8)
Padded zeros
Output
(8×8)


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

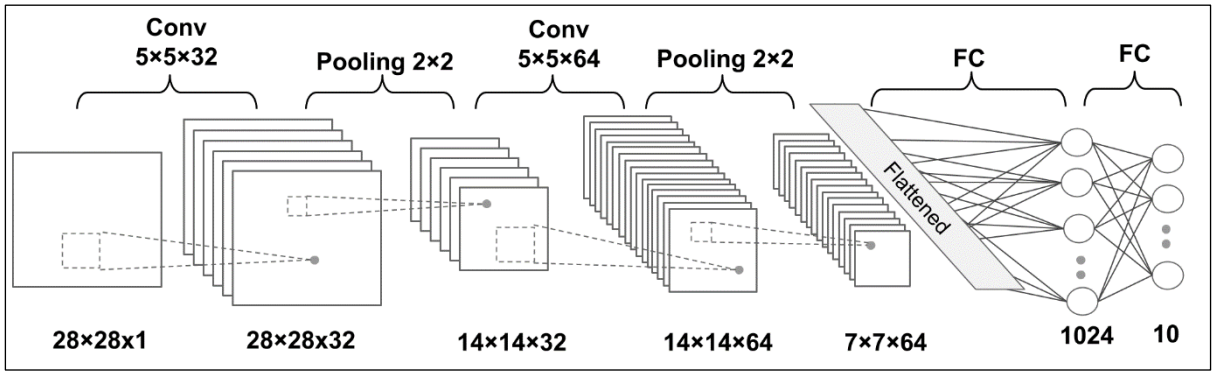

X^{padded}
0 0 0 0 0
0 2 1 2 0
0 5 0 1 0
0 1 7 3 0
0 0 0 0 0
0.5 1 0.5
0.1 0.4 0.3
0.4 0.7 0.5
Y:
4.6 1.6
7.5 2.9

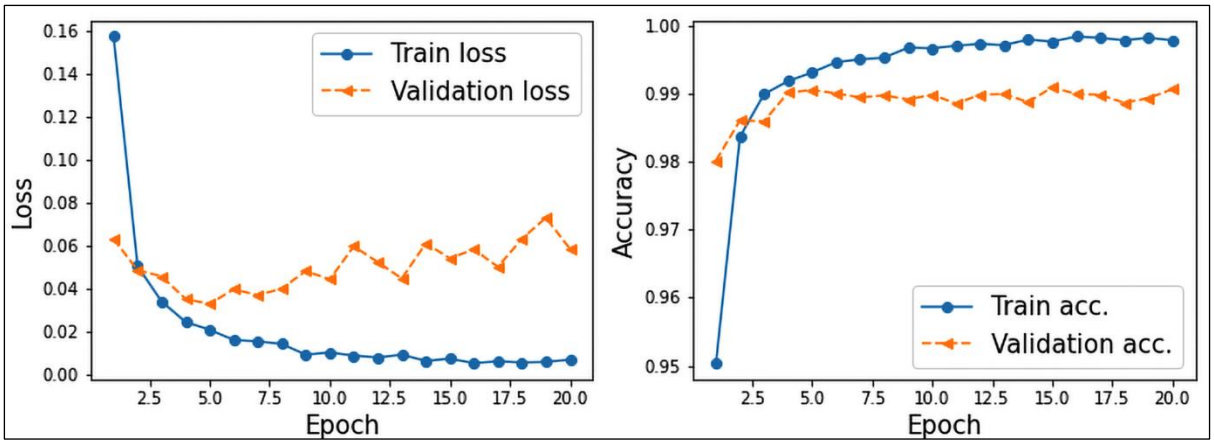

Pooling ($P_{3\times3}$)
Max-pooling
Mean-pooling
8 5
6 3
3.78 2.33
3 1.22
Note:
stride=(3, 3)

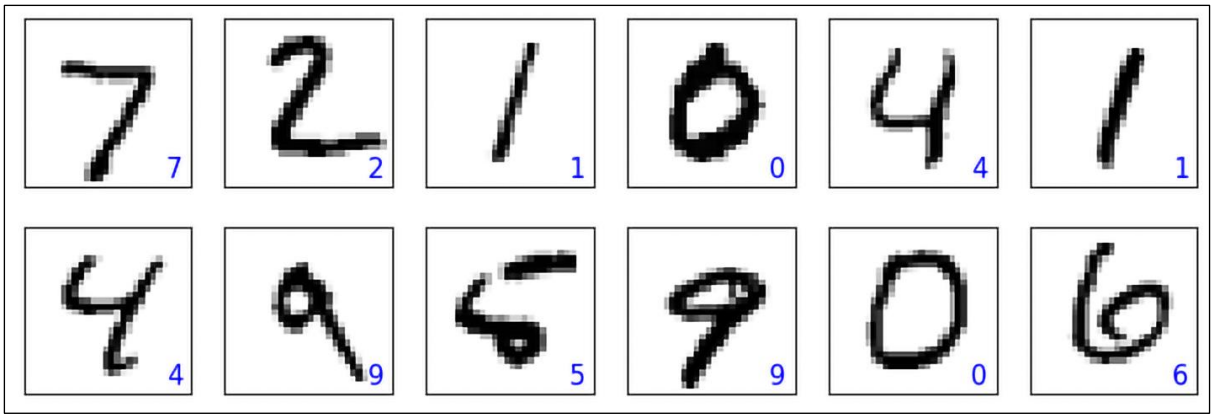

Convolution layer
Pooling layer
$C_{in} = 3$
Sum over input channels
$C_{out} = 5$
5
Kernel $m_1 \times m_2$
C_{in}
$m_1 \times m_2 \times 3 \times 5$

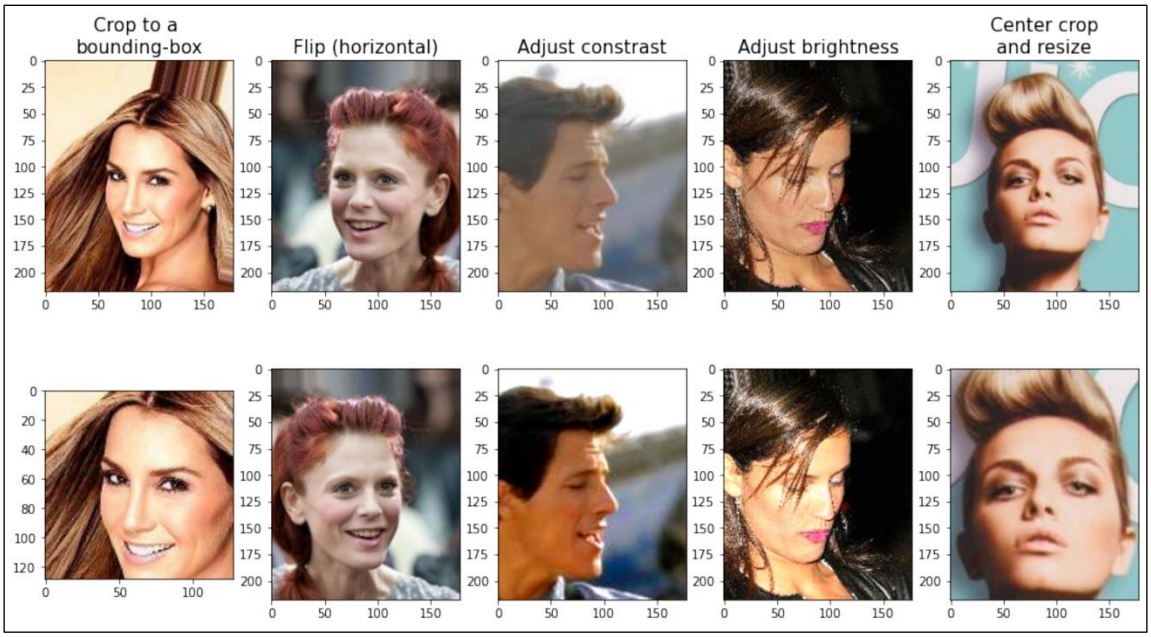


Loss function	Usage	Example *Using probabilities*	Example *Using logits*
BCELoss or BCEWithLogitsLoss	Binary classification	*BCELoss* y_true: 1 y_pred: 0.8	*BCEWithLogitsLoss* y_true: 1 y_pred: 0.8
NLLLoss or CrossEntropyLoss	Multiclass classification	*NLLLoss* y_true: 2 y_pred: 0.30 0.15 0.55	*CrossEntropyLoss* y_true: 2 y_pred: 1.5 0.8 2.1



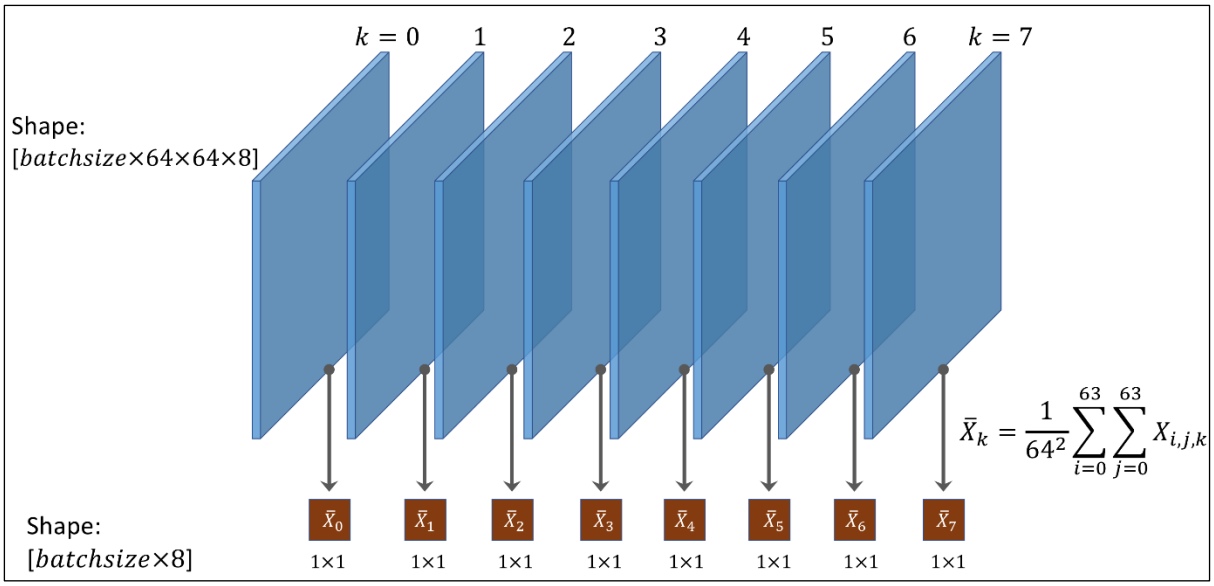

Train loss
Validation loss
Loss
Epoch
Train acc.
Validation acc.
Accuracy
Epoch

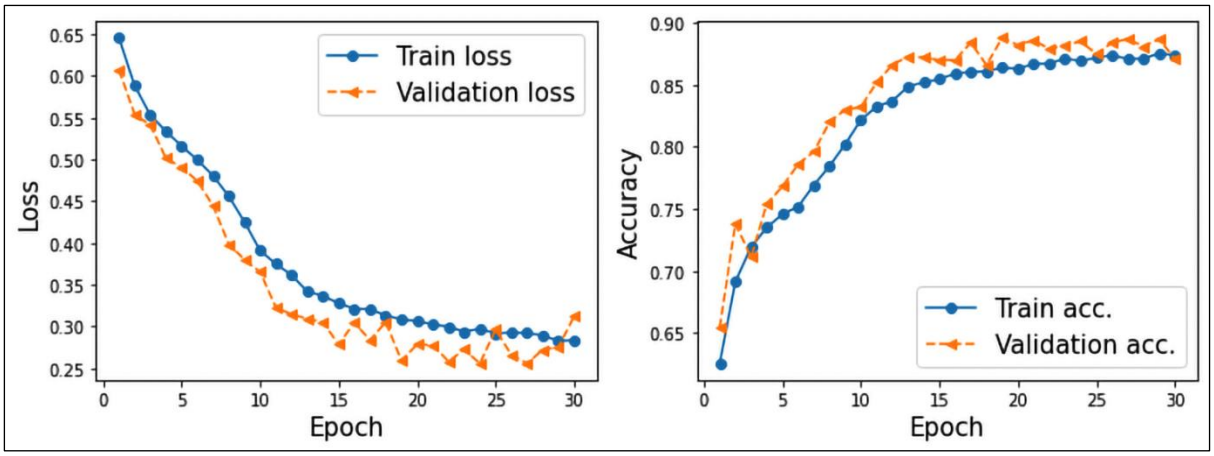


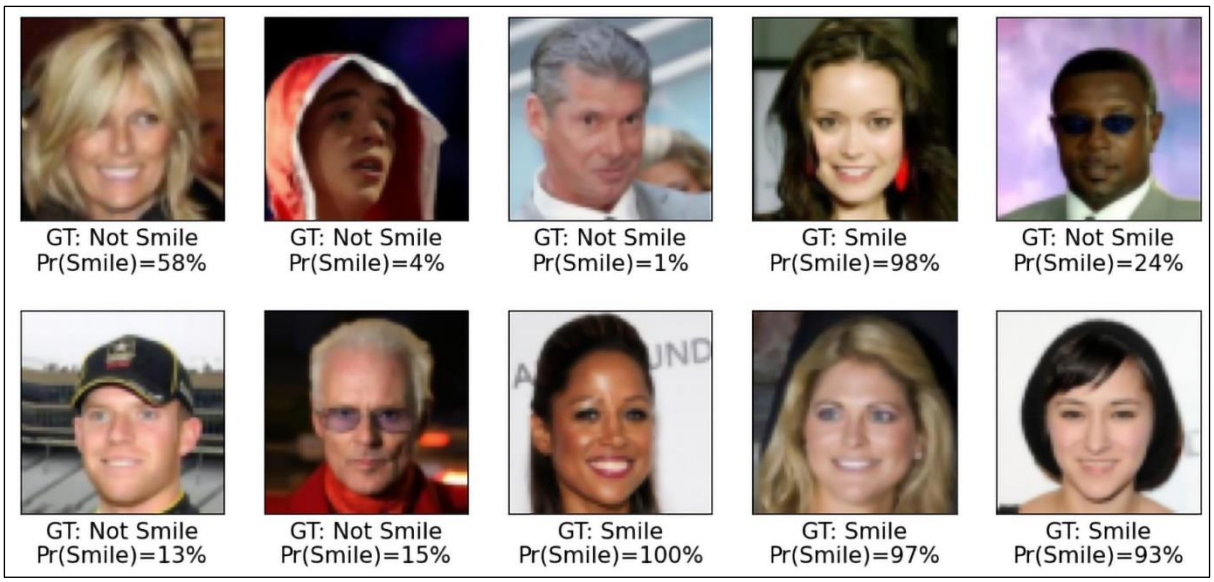

Crop to a bounding-box
Flip (horizontal)
Adjust constrast
Adjust brightness
Center crop and resize


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

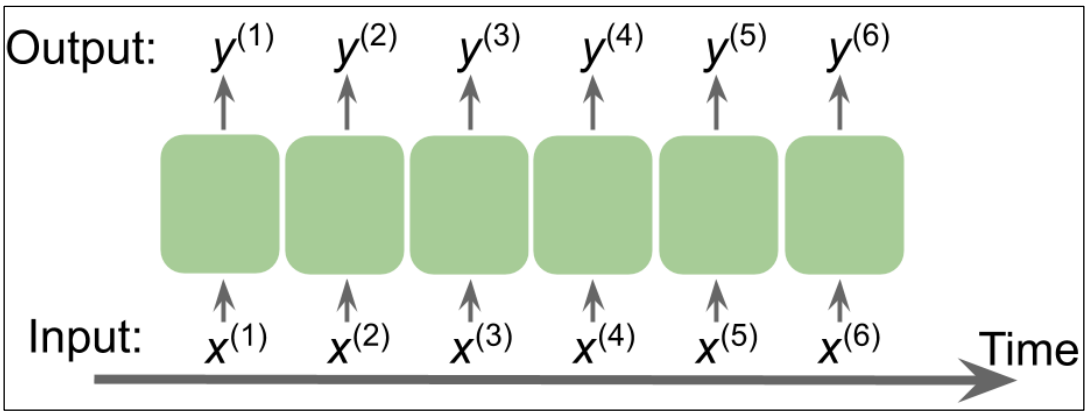

Epoch 0:
Epoch 1:
Epoch 2:
Epoch 3:
Epoch 4:

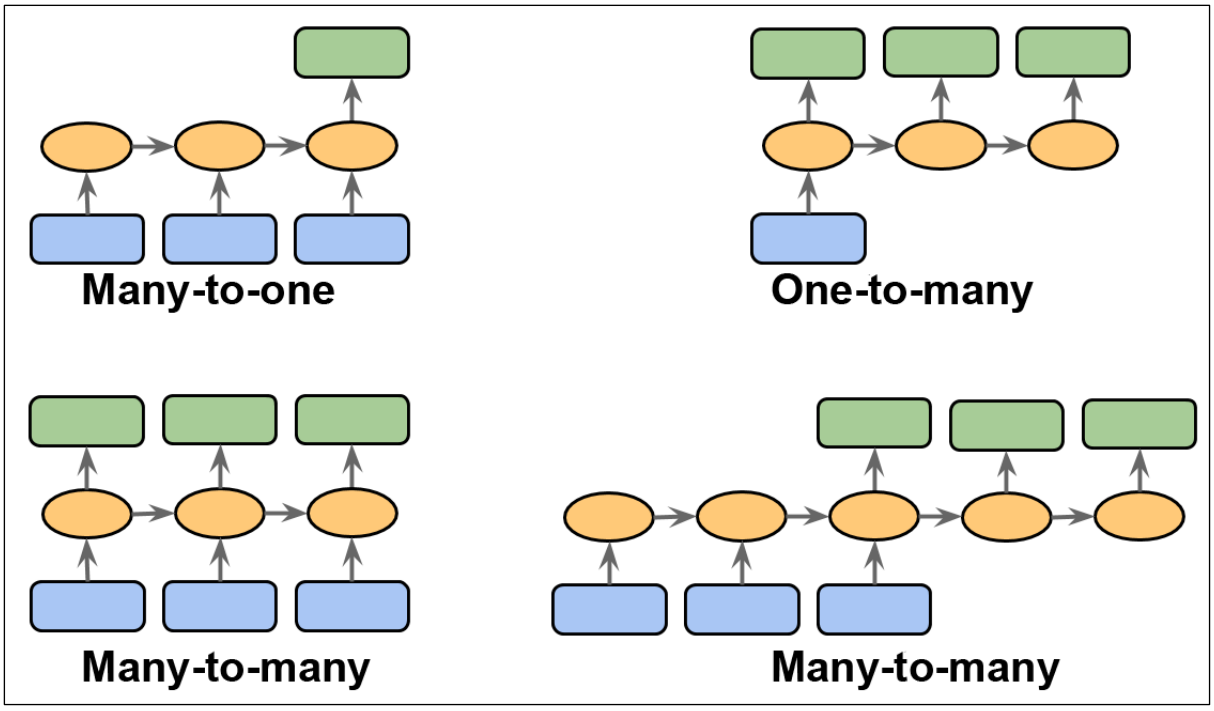

$k = 0$ 1 2 3 4 5 6 $k = 7$
Shape:
$[batchsize \times 64 \times 64 \times 8]$
$\bar{X}_k = \frac{1}{64^2} \sum_{i=0}^{63} \sum_{j=0}^{63} X_{i,j,k}$
Shape:
$[batchsize \times 8]$
$\bar{X}_0$ $\bar{X}_1$ $\bar{X}_2$ $\bar{X}_3$ $\bar{X}_4$ $\bar{X}_5$ $\bar{X}_6$ $\bar{X}_7$
1×1 1×1 1×1 1×1 1×1 1×1 1×1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

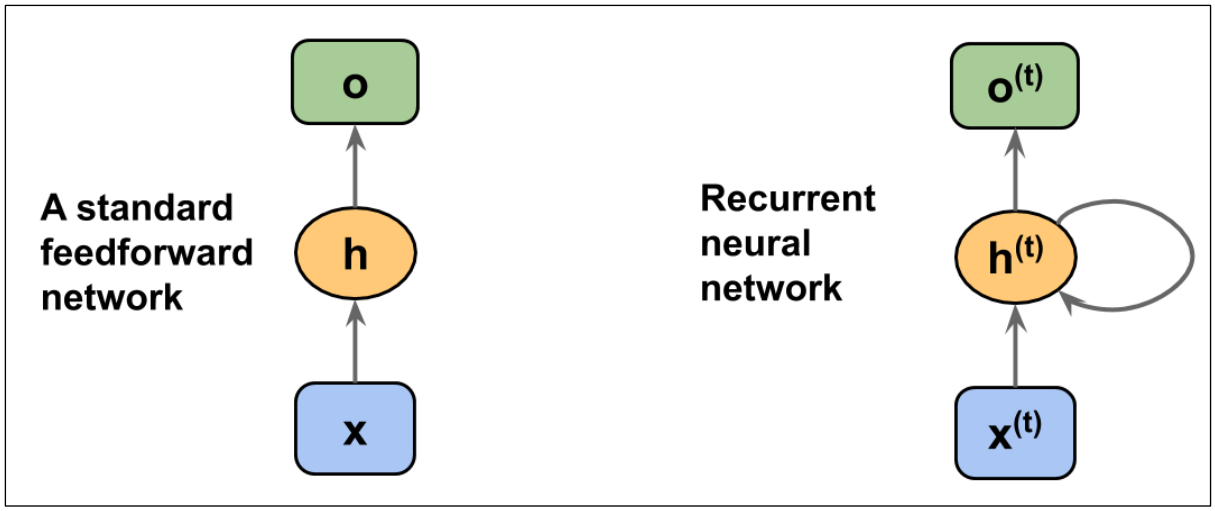

Train loss
Validation loss
Loss
Epoch
Train acc.
Validation acc.
Accuracy
Epoch

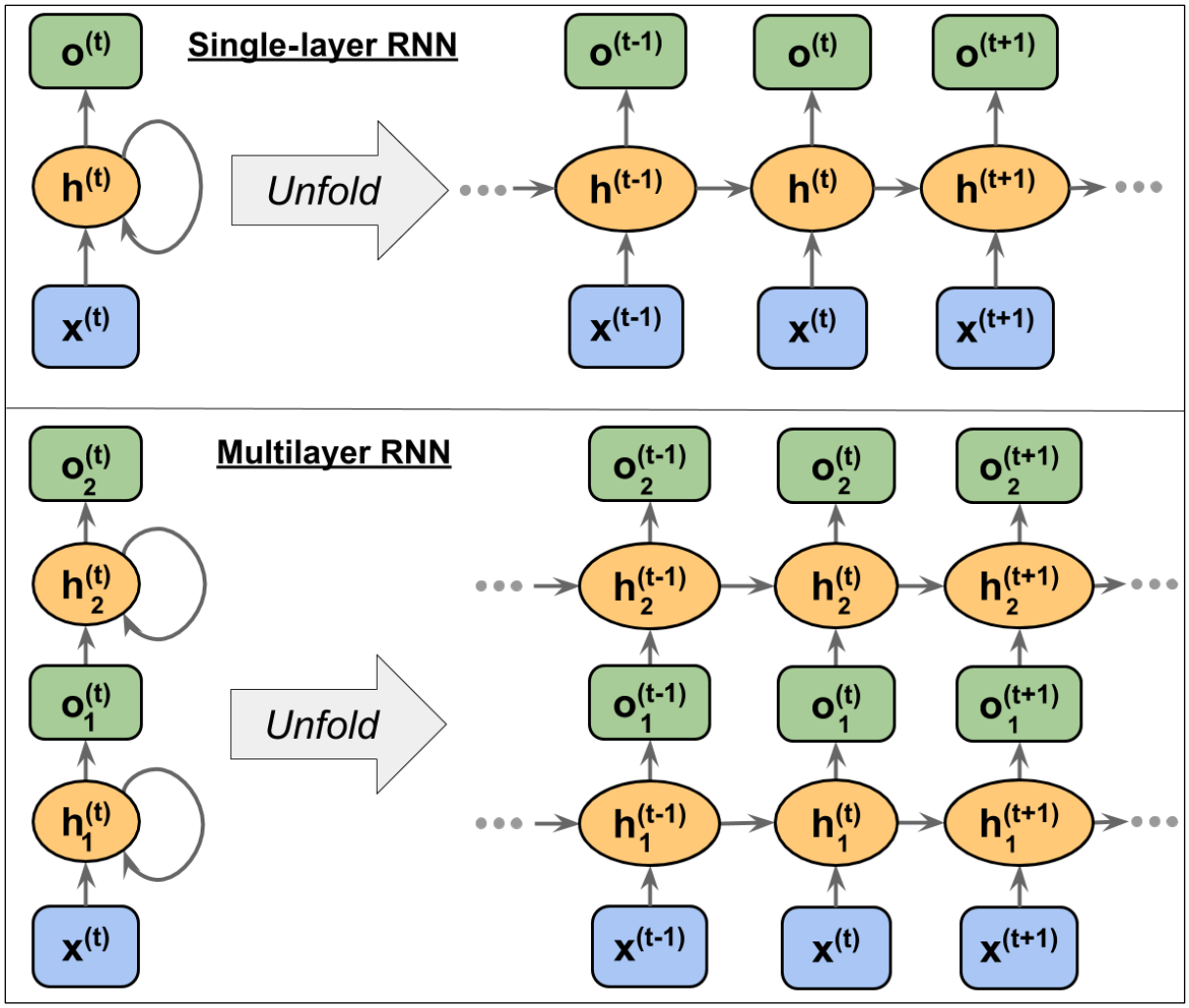

GT: Not Smile
Pr(Smile)=58%
GT: Not Smile
Pr(Smile)=4%
GT: Not Smile
Pr(Smile)=1%
GT: Smile
Pr(Smile)=98%
GT: Not Smile
Pr(Smile)=24%
GT: Not Smile
Pr(Smile)=13%
GT: Not Smile
Pr(Smile)=15%
GT: Smile
Pr(Smile)=100%
GT: Smile
Pr(Smile)=97%
GT: Smile
Pr(Smile)=93%

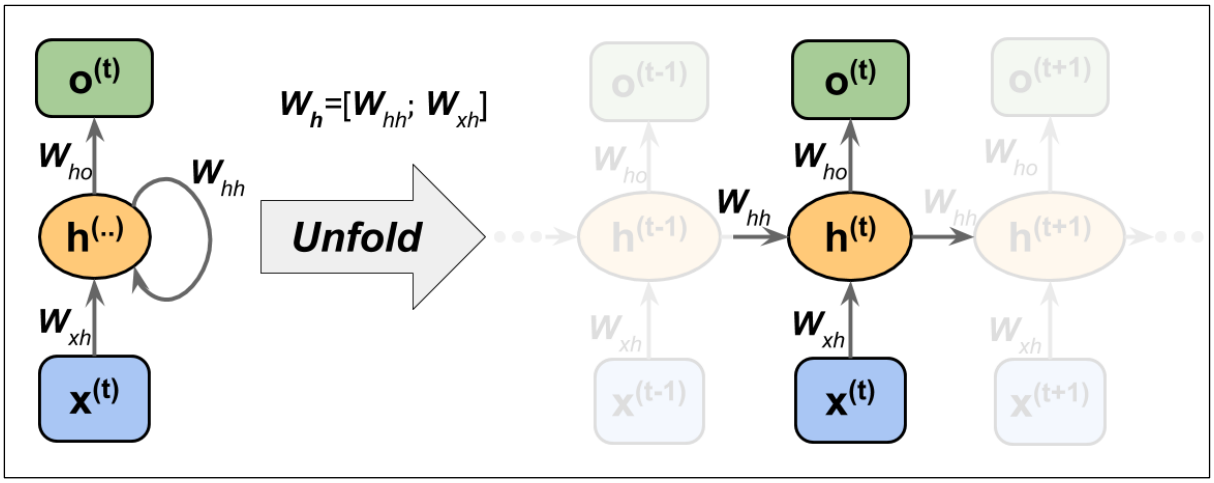
Chapter 15: Modeling Sequential Data Using Recurrent Neural Networks

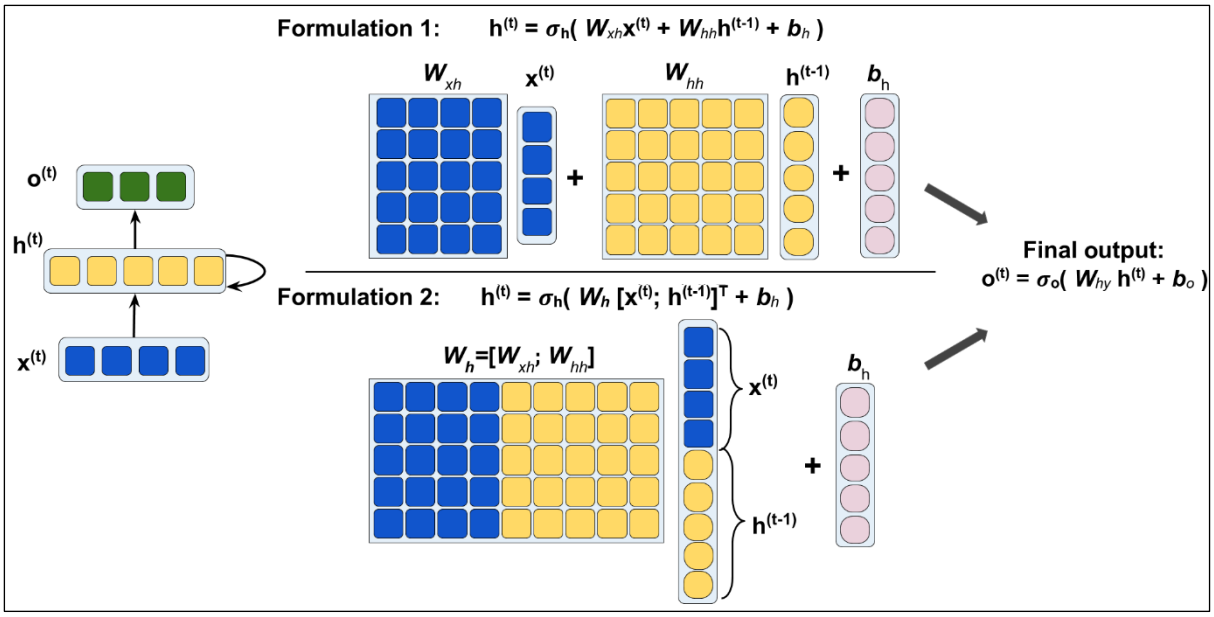


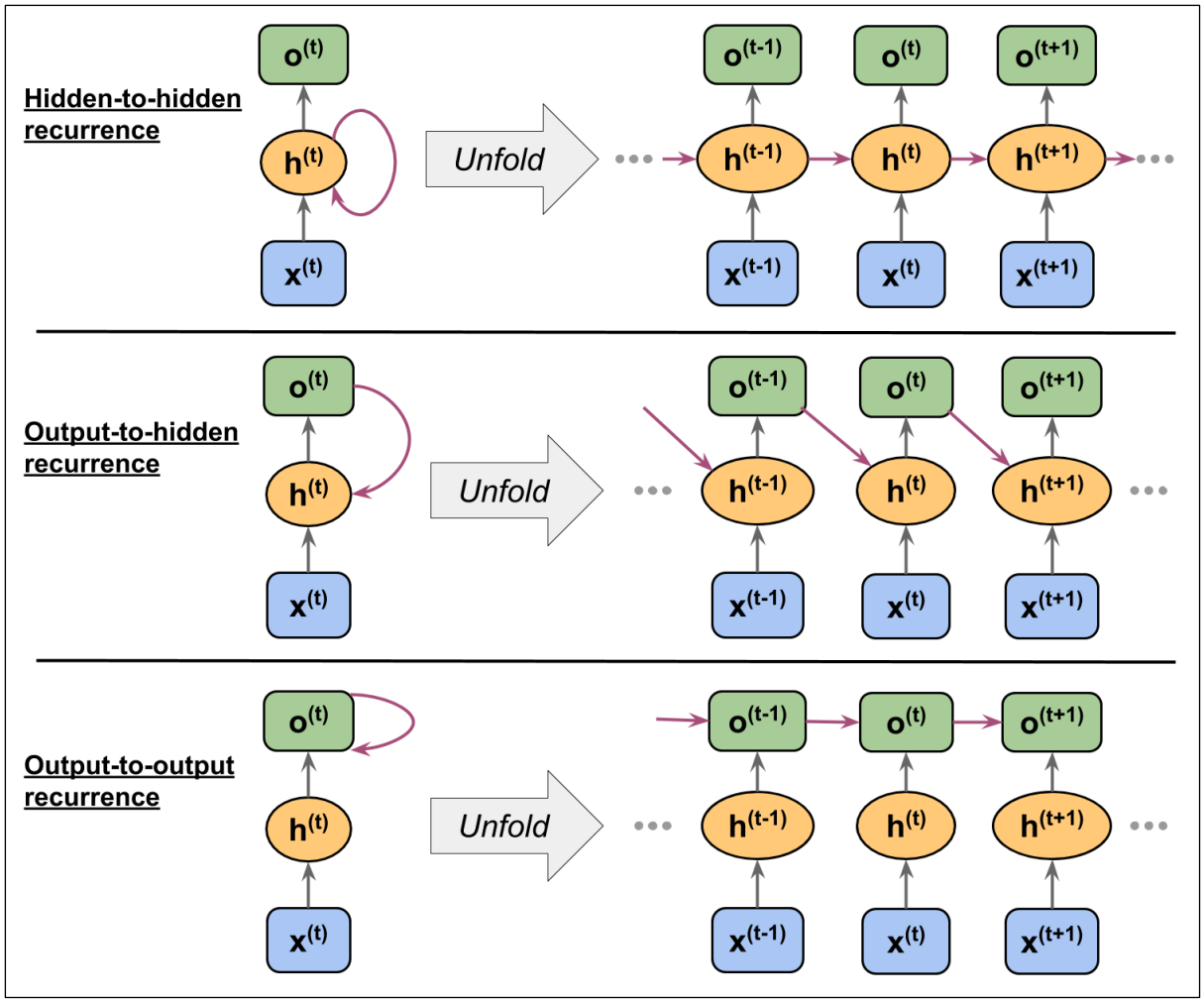


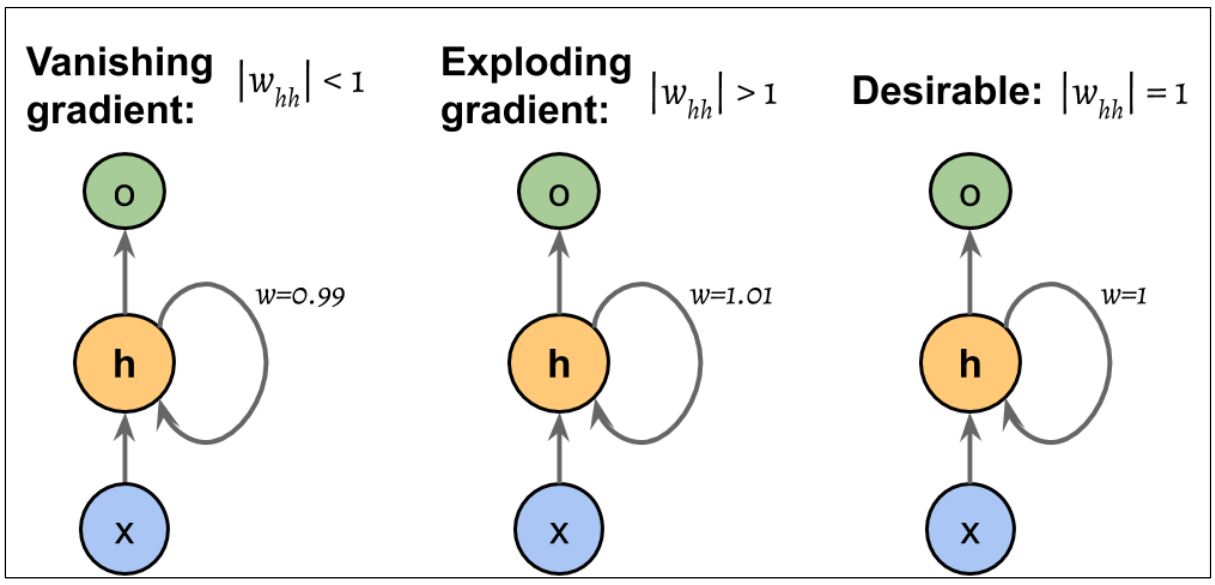

A standard feedforward network
o
h
x
Recurrent neural network
o(t)
h(t)
x(t)

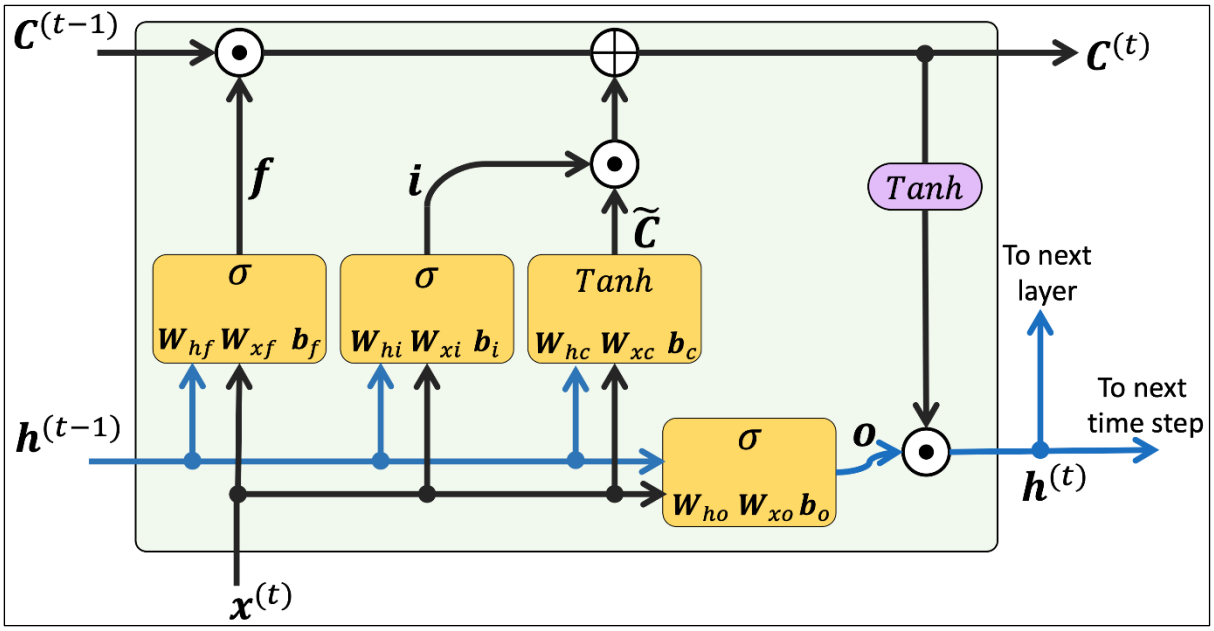

Single-layer RNN
Unfold
Multilayer RNN
Unfold

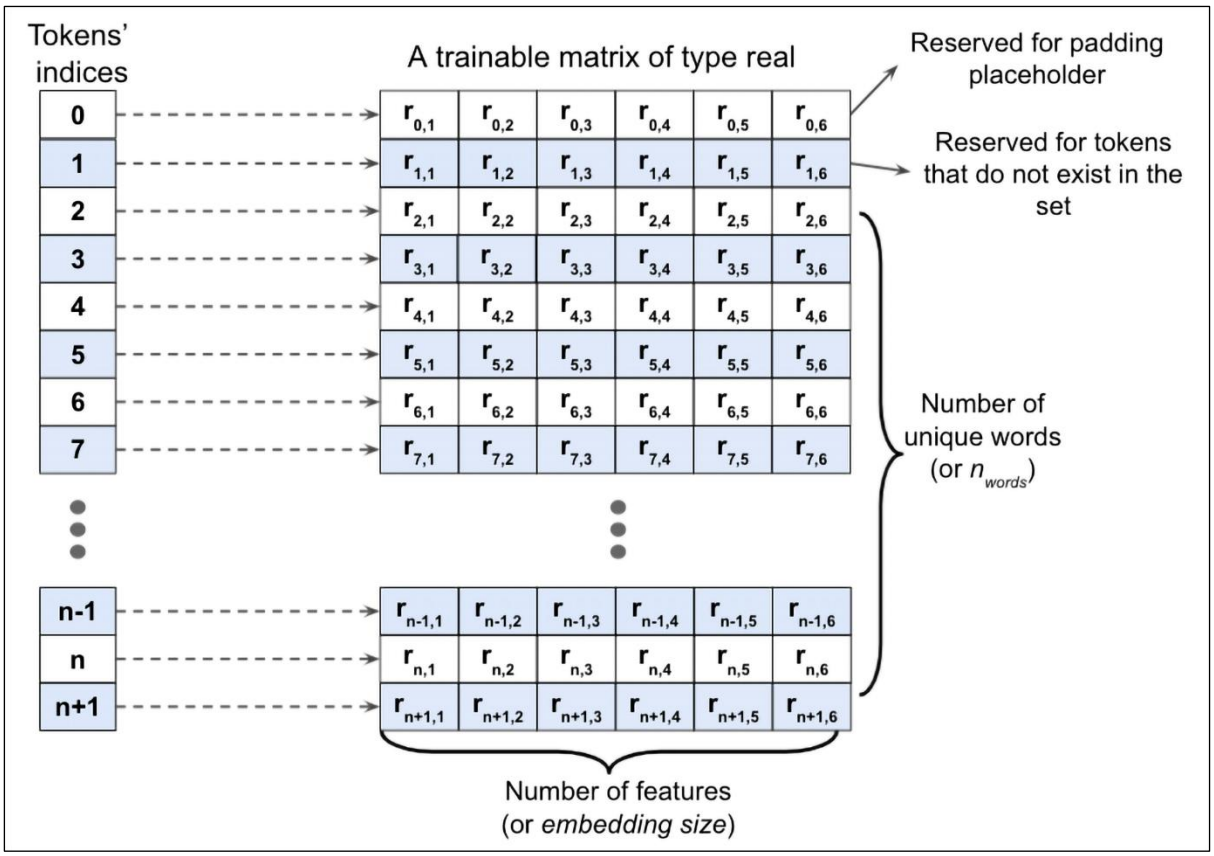

o(t)
W_ho
W_hh
h(..)
W_xh
x(t)
W_h=[W_hh; W_xh]
Unfold
o(t-1)
o(t)
o(t+1)
W_ho
W_hh
h(t-1)
h(t)
h(t+1)
W_xh
x(t-1)
x(t)
x(t+1)

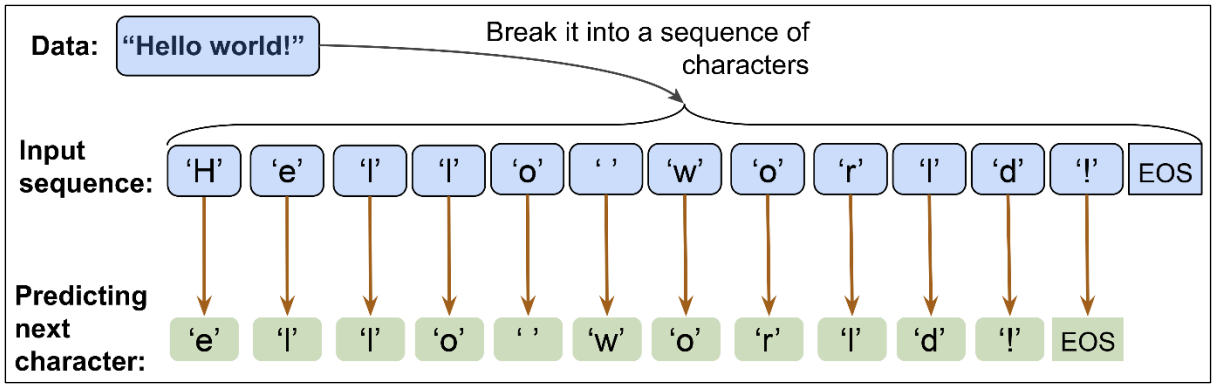

Formulation 1: h(t) = σh(Wxh x(t) + Whh h(t-1) + bh)
Wxh
x(t)
Whh
h(t-1)
bh
o(t)
h(t)
x(t)
Final output:
o(t) = σo(Why h(t) + bo)
Formulation 2: h(t) = σh(Wh [x(t); h(t-1)]T + bh)
Wh=[Wxh; Whh]
x(t)
h(t-1)
bh

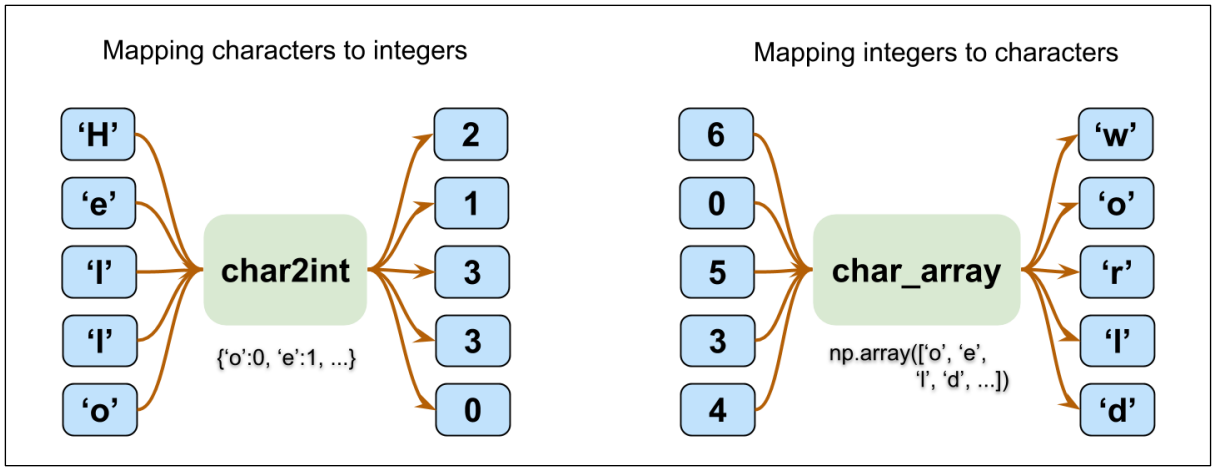

Hidden-to-hidden recurrence
Unfold
Output-to-hidden recurrence
Unfold
Output-to-output recurrence
Unfold

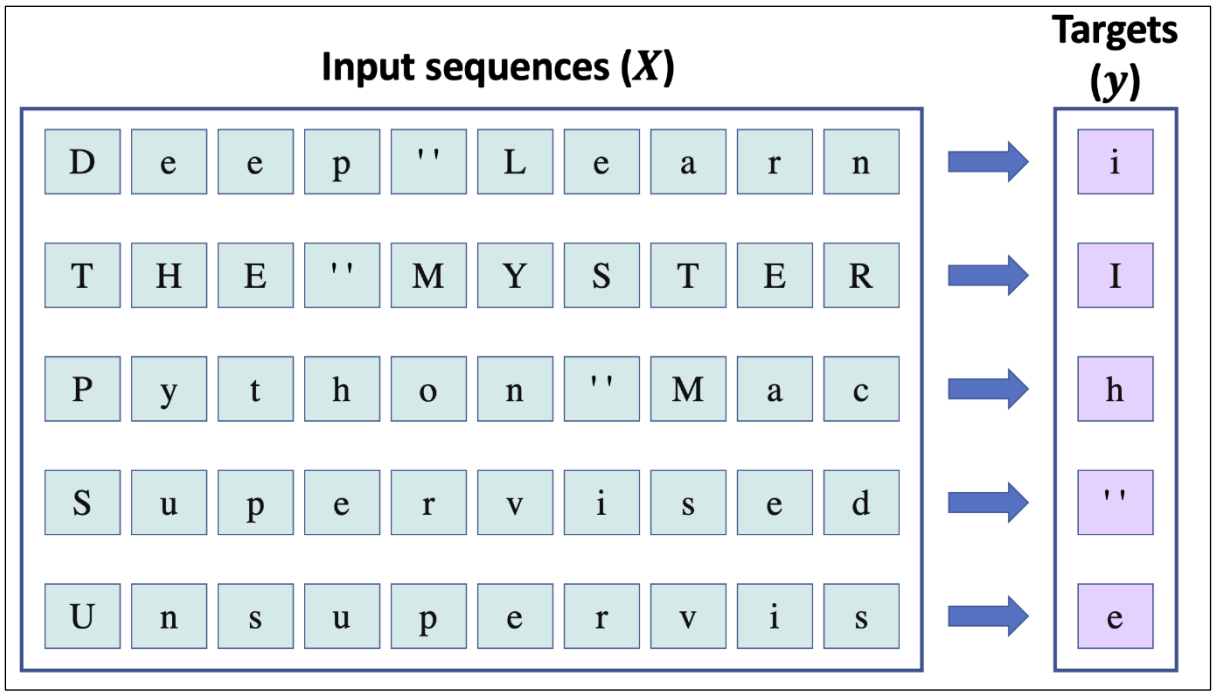

Vanishing gradient: $|w_{hh}| < 1$
w=0.99
Exploding gradient: $|w_{hh}| > 1$
w=1.01
Desirable: $|w_{hh}| = 1$
w=1

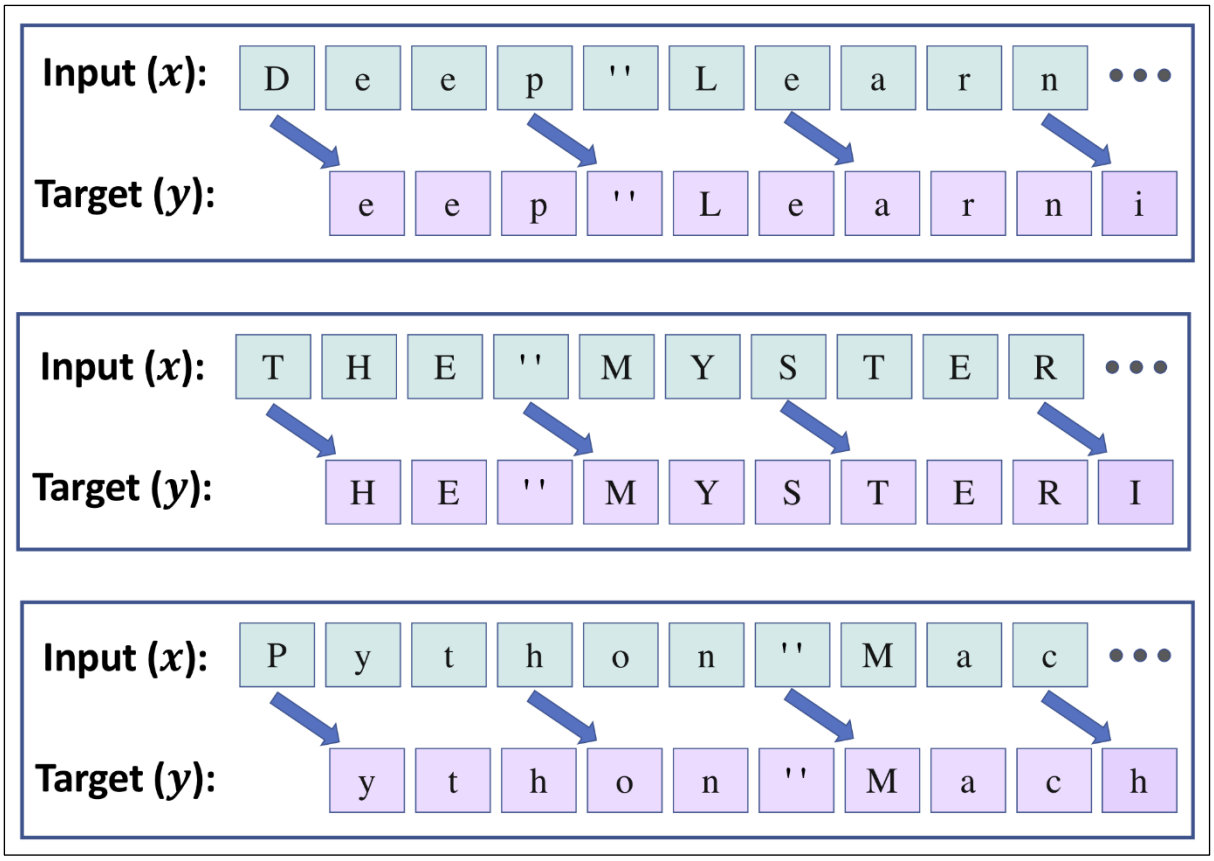

$C^{(t-1)}$
$C^{(t)}$
f
i
$\tilde{C}$
$Tanh$
σ
W_{hf} W_{xf} b_f
σ
W_{hi} W_{xi} b_i
$Tanh$
W_{hc} W_{xc} b_c
To next layer
To next time step
$h^{(t-1)}$
σ
W_{ho} W_{xo} b_o
o
$h^{(t)}$
$x^{(t)}$


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

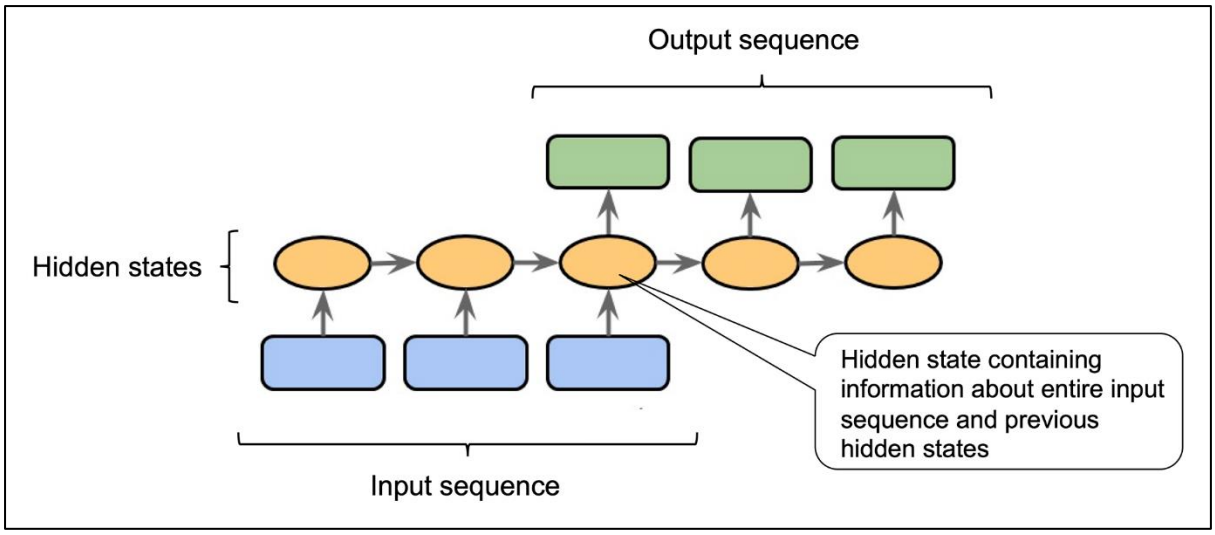

Data:
"Hello world!"
Break it into a sequence of characters
Input sequence:
'H' 'e' 'l' 'l' 'o' ' ' 'w' 'o' 'r' 'l' 'd' '!' EOS
Predicting next character:
'e' 'l' 'l' 'o' ' ' 'w' 'o' 'r' 'l' 'd' '!' EOS

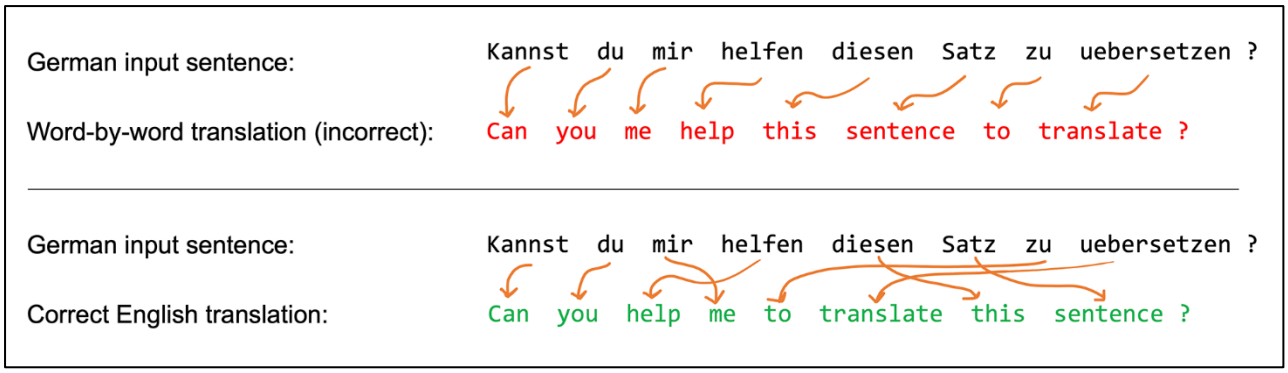

Mapping characters to integers
'H'
'e'
'l'
'l'
'o'
char2int
{'o':0, 'e':1, ...}
2
1
3
3
0
Mapping integers to characters
6
0
5
3
4
char_array
np.array(['o', 'e', 'l', 'd', ...])
'w'
'o'
'r'
'l'
'd'

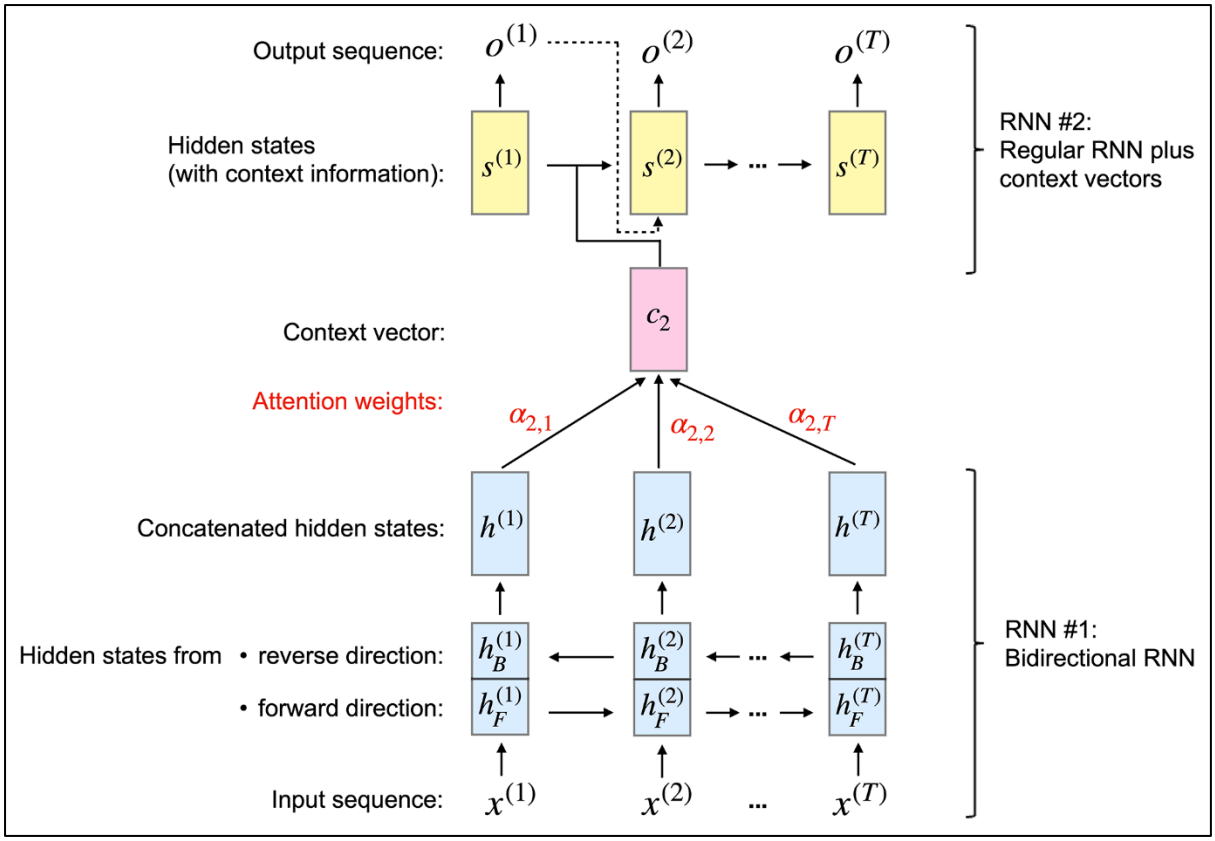

Input sequences (X)
Targets (y)
D e e p ' ' L e a r n i
T H E ' ' M Y S T E R I
P y t h o n ' ' M a c h
S u p e r v i s e d ' '
U n s u p e r v i s e

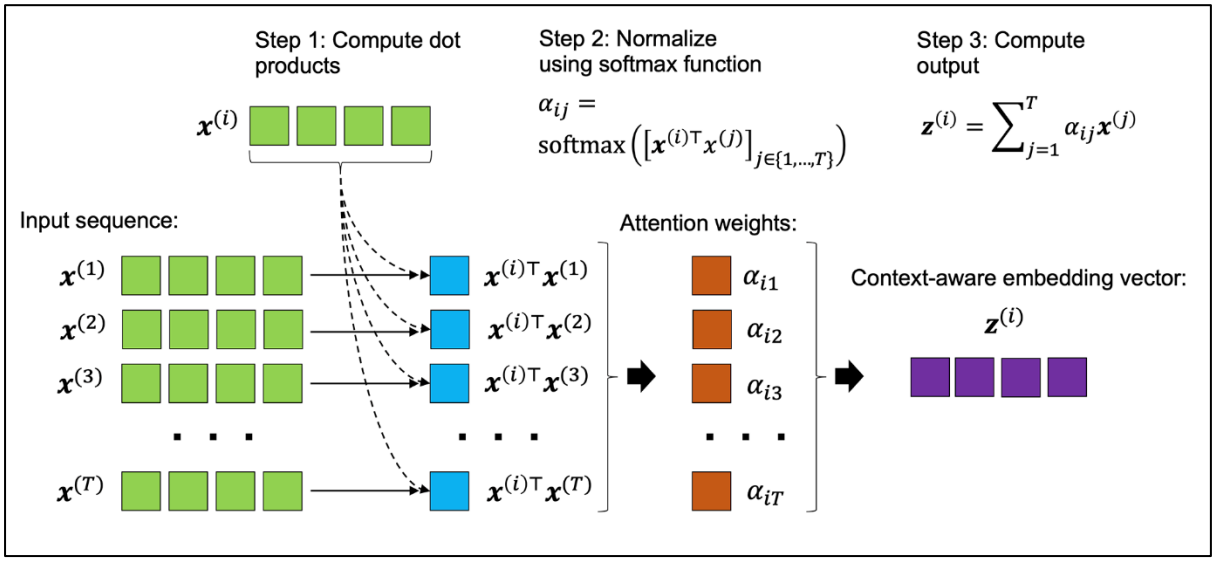

Input (x):
D e e p '' L e a r n • • •
Target (y):
e e p '' L e a r n i
Input (x):
T H E '' M Y S T E R • • •
Target (y):
H E '' M Y S T E R I
Input (x):
P y t h o n '' M a c • • •
Target (y):
y t h o n '' M a c h

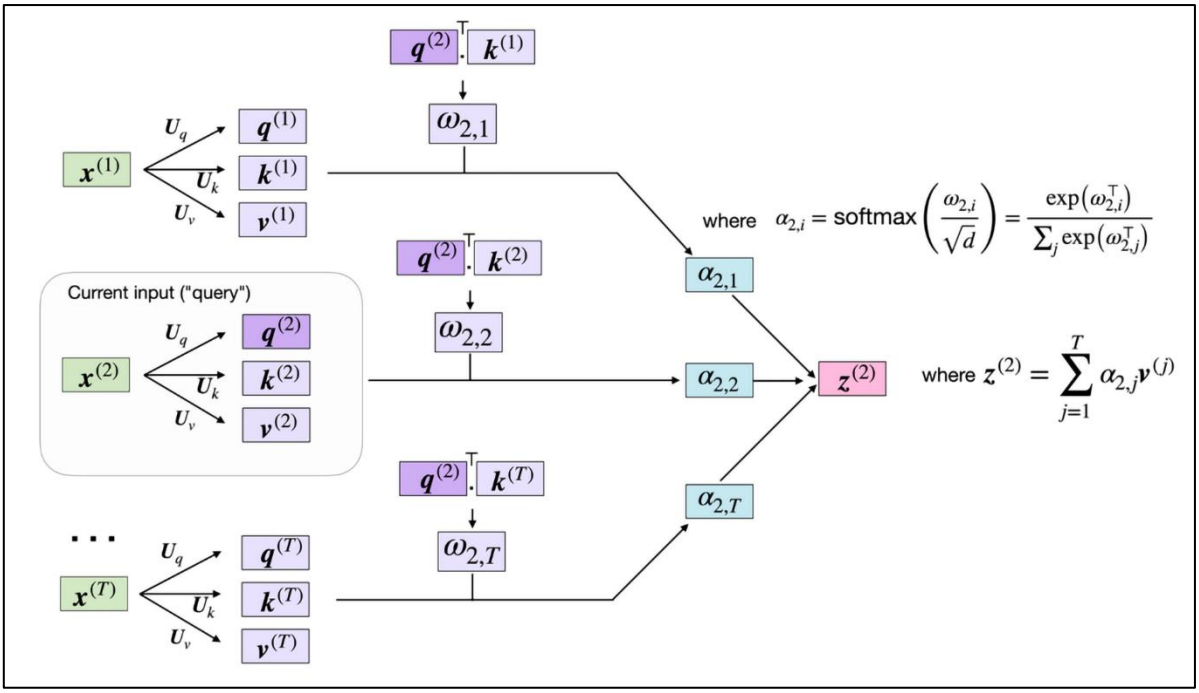
Chapter 16: Transformers – Improving Natural Language Processing with Attention Mechanisms

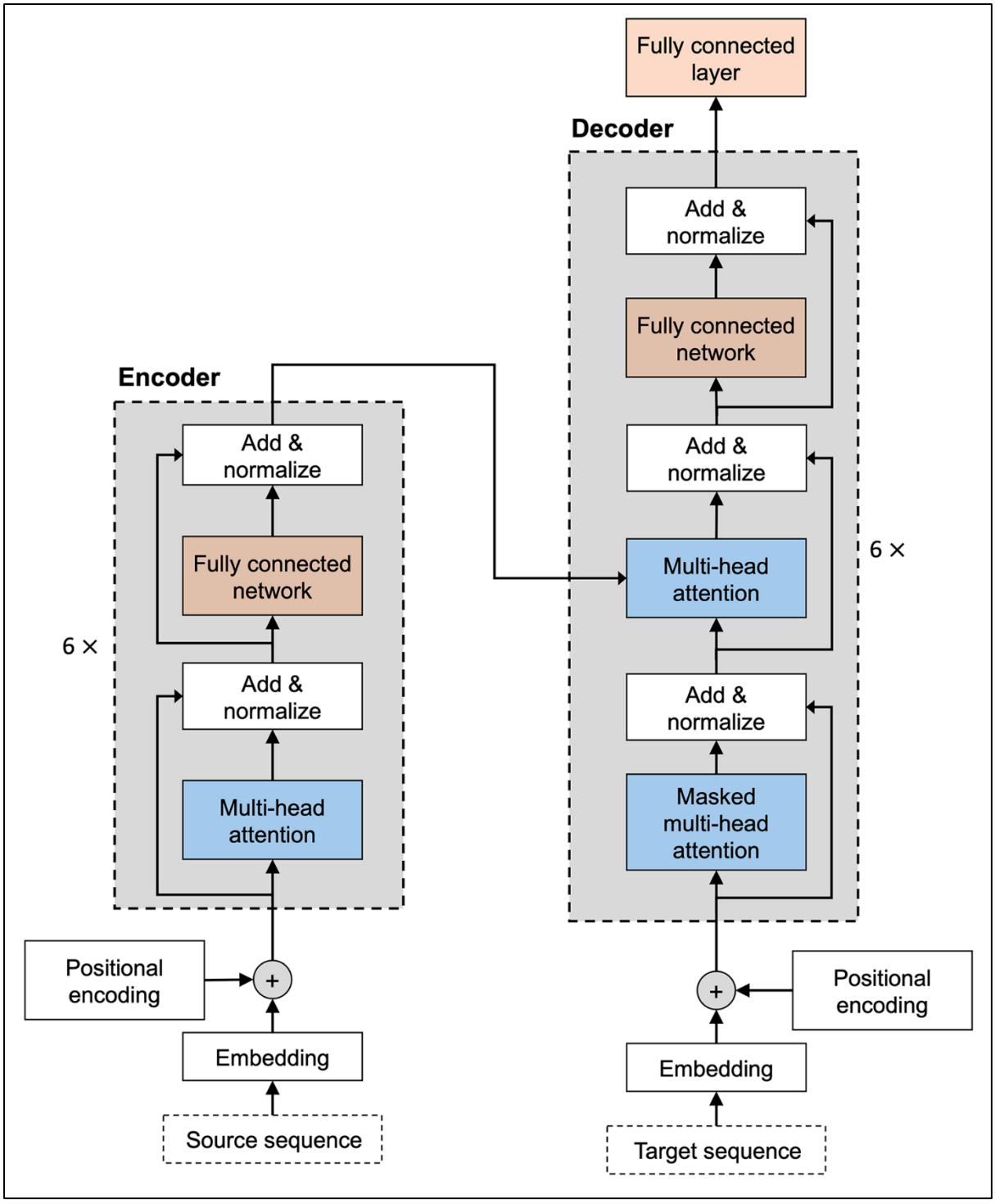


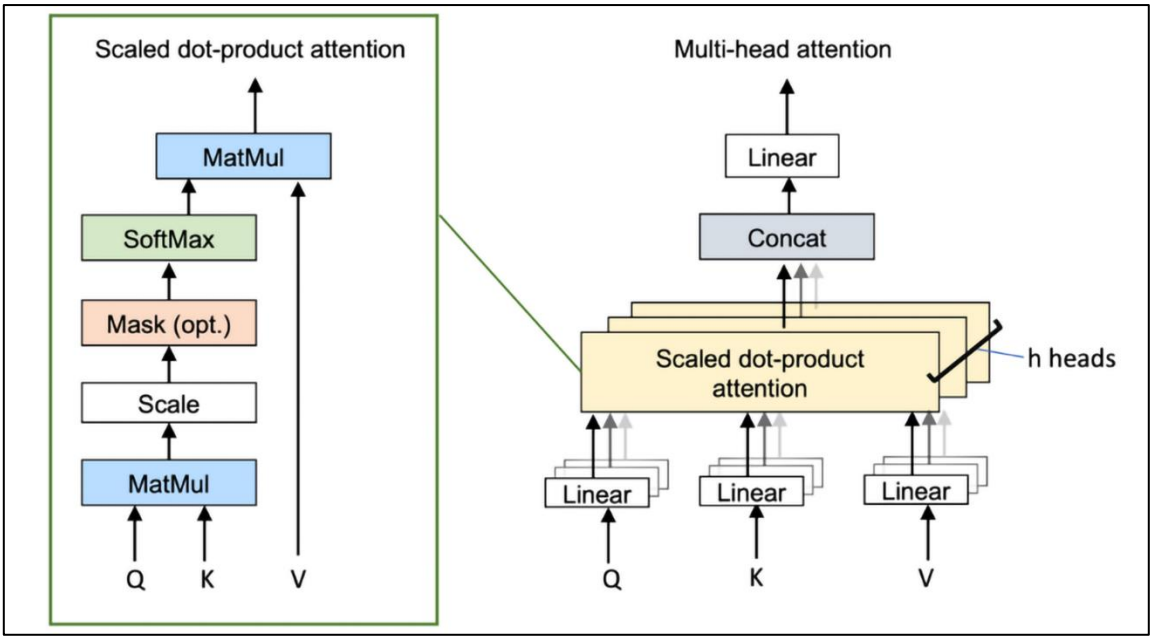


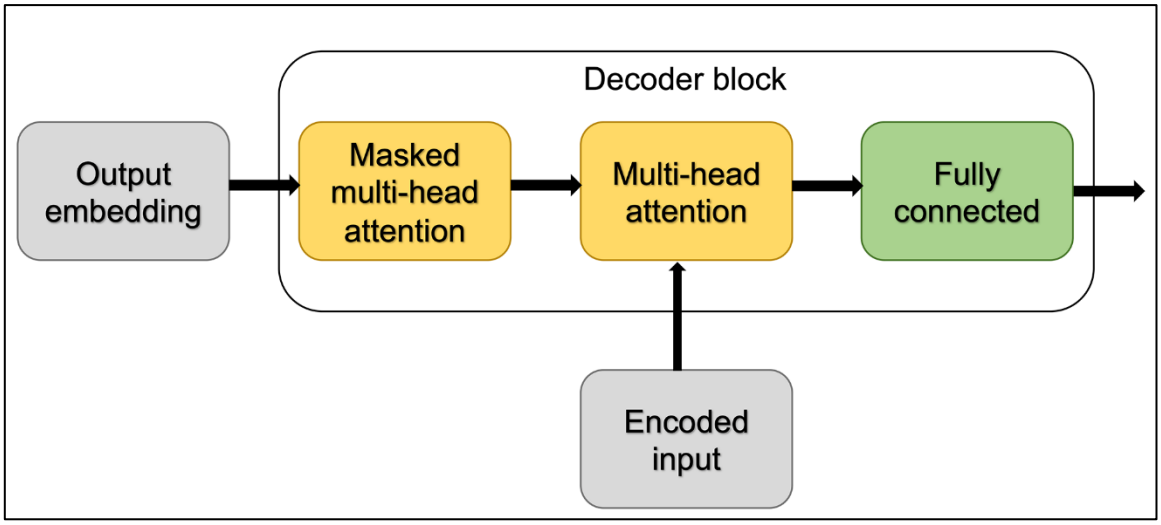

Output sequence:
$o^{(1)}$
$o^{(2)}$
$o^{(T)}$
Hidden states
(with context information):
$s^{(1)}$
$s^{(2)}$
$s^{(T)}$
RNN #2:
Regular RNN plus
context vectors
Context vector:
c_2
Attention weights:
$\alpha_{2,1}$
$\alpha_{2,2}$
$\alpha_{2,T}$
Concatenated hidden states:
$h^{(1)}$
$h^{(2)}$
$h^{(T)}$
Hidden states from • reverse direction:
• forward direction:
$h_B^{(1)}$
$h_B^{(2)}$
$h_B^{(T)}$
$h_F^{(1)}$
$h_F^{(2)}$
$h_F^{(T)}$
RNN #1:
Bidirectional RNN
Input sequence:
$x^{(1)}$
$x^{(2)}$
$x^{(T)}$

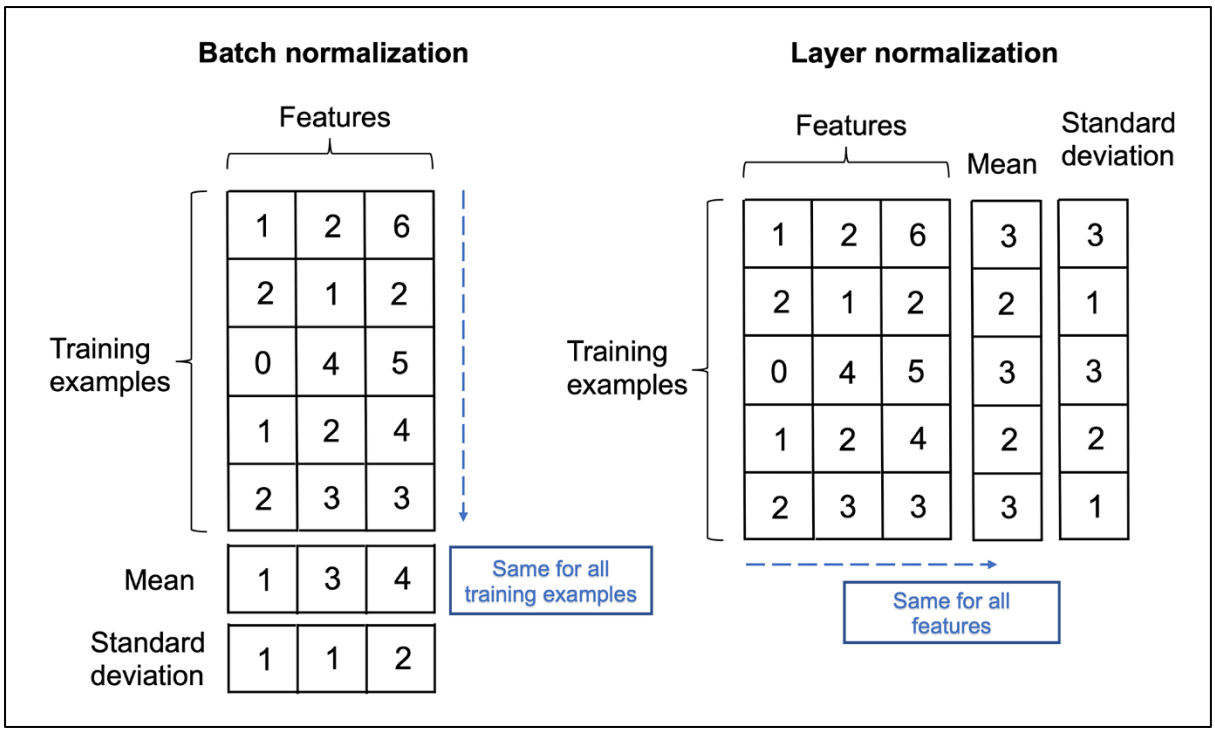

Step 1: Compute dot products
Step 2: Normalize using softmax function
$\alpha_{ij} = \text{softmax}\left(\left[x^{(i)\top}x^{(j)}\right]_{j\in\{1,\ldots,T\}}\right)$
Step 3: Compute output
$z^{(i)} = \sum_{j=1}^{T} \alpha_{ij} x^{(j)}$
$x^{(i)}$
Input sequence:
$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(T)}$
$x^{(i)\top}x^{(1)}$
$x^{(i)\top}x^{(2)}$
$x^{(i)\top}x^{(3)}$
$x^{(i)\top}x^{(T)}$
Attention weights:
α_{i1}
α_{i2}
α_{i3}
α_{iT}
Context-aware embedding vector:
$z^{(i)}$

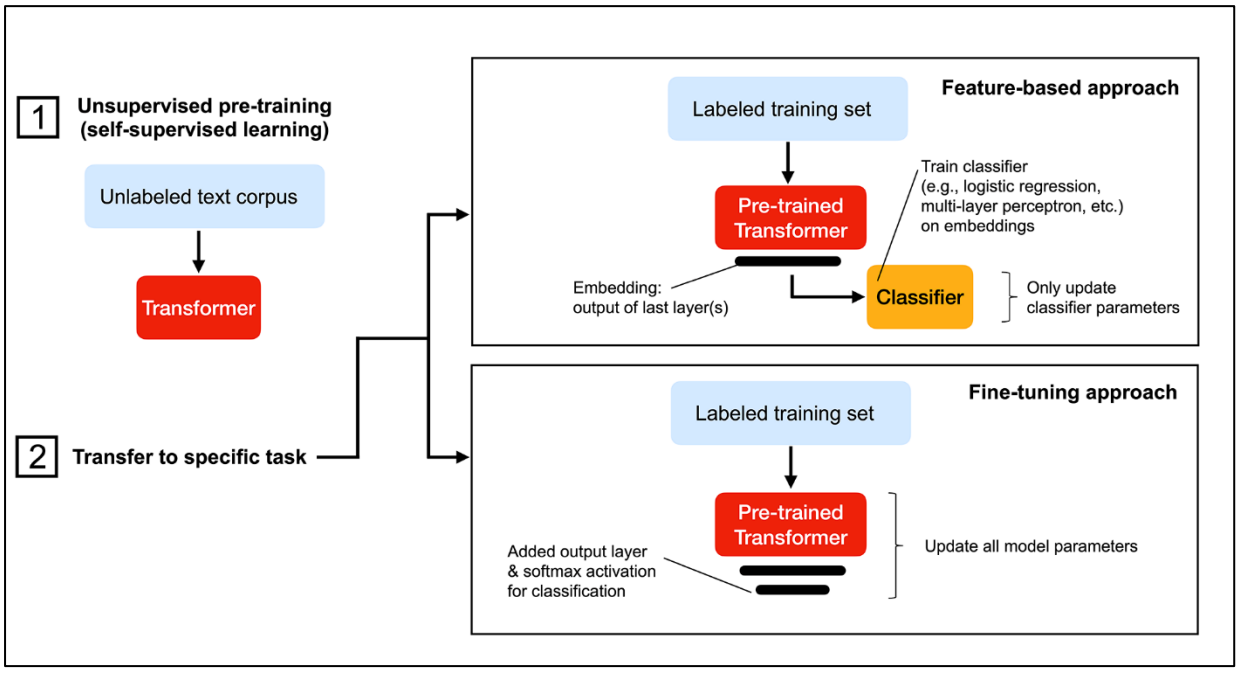

Current input ("query")
where $\alpha_{2,i} = \text{softmax}\left(\frac{\omega_{2,i}}{\sqrt{d}}\right) = \frac{\exp(\omega_{2,i}^{\top})}{\sum_j \exp(\omega_{2,j}^{\top})}$
where $z^{(2)} = \sum_{j=1}^{T} \alpha_{2,j} v^{(j)}$

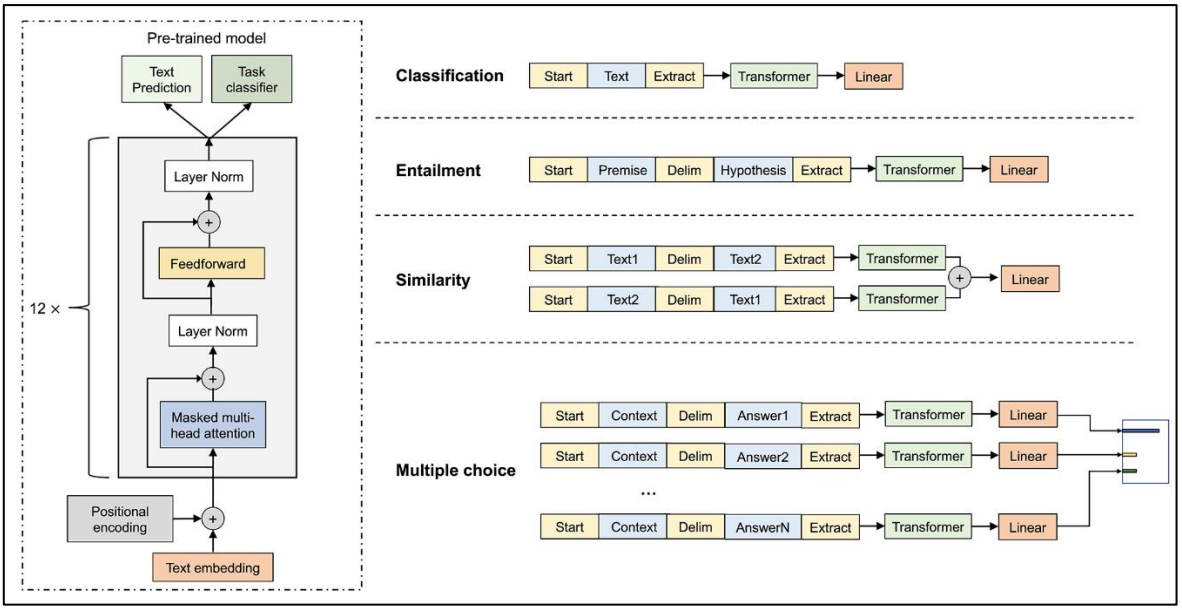

Fully connected layer
Decoder
Add & normalize
Fully connected network
Add & normalize
Multi-head attention
Add & normalize
Masked multi-head attention
6 ×
Encoder
Add & normalize
Fully connected network
Add & normalize
Multi-head attention
6 ×
Positional encoding
+
Embedding
Source sequence
+
Positional encoding
Embedding
Target sequence


Scaled dot-product attention
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V
Multi-head attention
Linear
Concat
Scaled dot-product attention
h heads
Linear
Linear
Linear
Q
K
V


Decoder block
Output embedding
Masked multi-head attention
Multi-head attention
Fully connected
Encoded input


Batch normalization
Features
Training examples
1 2 6
2 1 2
0 4 5
1 2 4
2 3 3
Mean
1 3 4
Standard deviation
1 1 2
Same for all training examples
Layer normalization
Features
Mean
Standard deviation
Training examples
1 2 6 3 3
2 1 2 2 1
0 4 5 3 3
1 2 4 2 2
2 3 3 3 1
Same for all features


1 Unsupervised pre-training (self-supervised learning)
Unlabeled text corpus
Transformer
2 Transfer to specific task
Feature-based approach
Labeled training set
Pre-trained Transformer
Train classifier (e.g., logistic regression, multi-layer perceptron, etc.) on embeddings
Embedding: output of last layer(s)
Classifier
Only update classifier parameters
Fine-tuning approach
Labeled training set
Pre-trained Transformer
Added output layer & softmax activation for classification
Update all model parameters



The three settings we explore for in-context learning

1) Zero-shot

The model is only given a natural language description of the task. No weight updates are performed.

```
Translate English to French:
cheese =>
```

Task description

Prompt

2) One-shot

In addition to the task description, the model is also given a simple example of a task. No weight updates are performed.

```
Translate English to French:
sea otter => loutre de mer
cheese =>
```

Task description

Example

Prompt

3) Few-shot

In addition to the task description, the model is also given a simple example of a task. No gradient updates are performed.

```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush giraffe => giraffe peluche
cheese =>
```

Task description

Examples

Prompt

Traditional fine-tuning (not for GPT-3)

Fine-tuning

The model is trained using a large corpus of example tasks.

```
sea otter => loutre de mer
```

Example #1

Gradient update

```
peppermint => menthe poivrée
```

Example #2

Gradient update

...

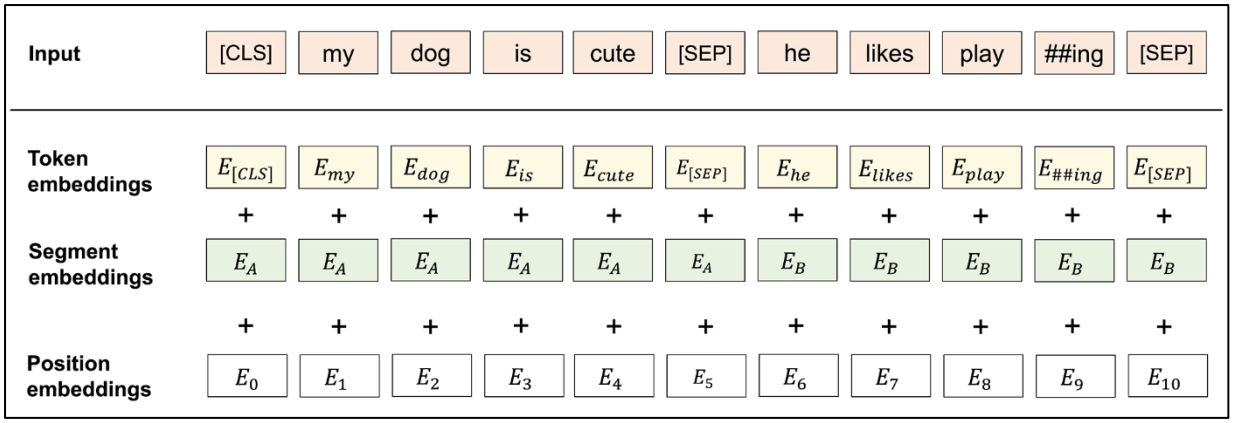
```
plush giraffe => giraffe peluche
```

Example #N

Gradient update

```
cheese =>
```

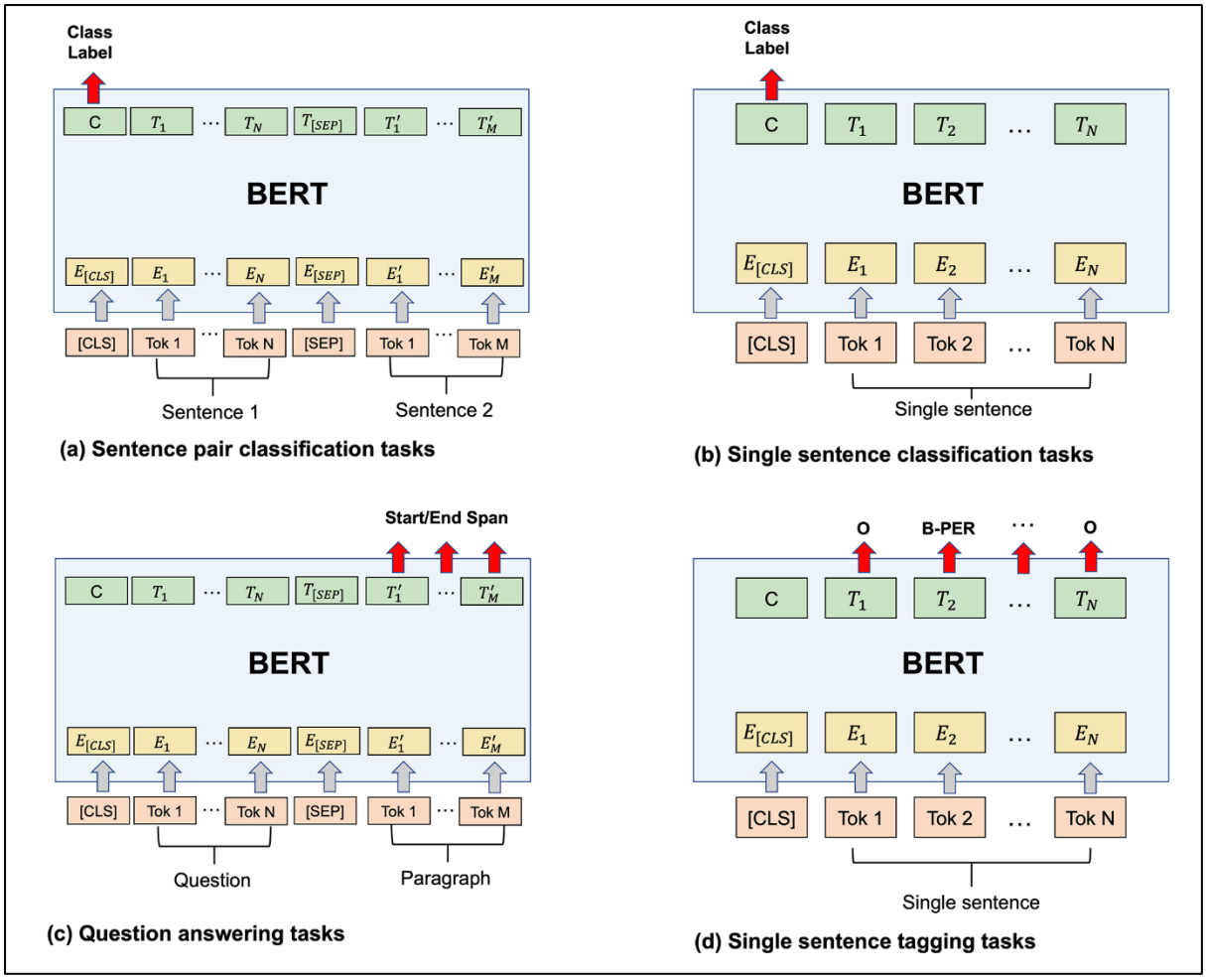
Prompt

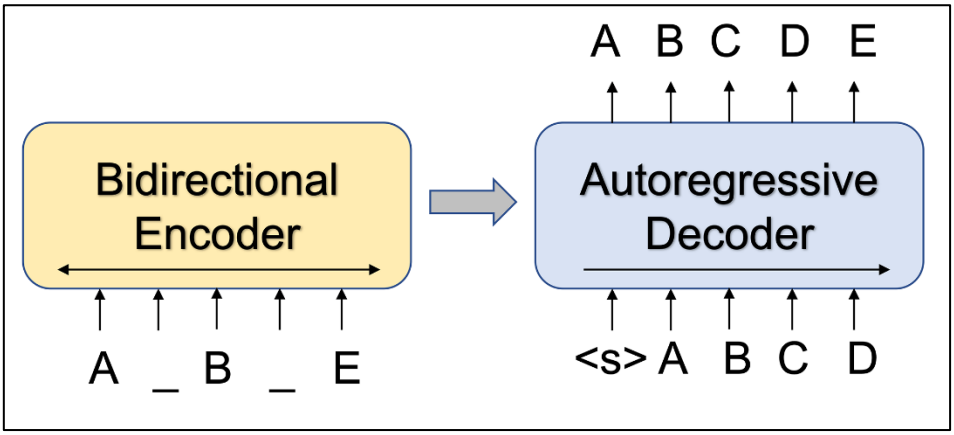


Input sentence: A quick brown fox jumps over a lazy dog.

Output sentence:

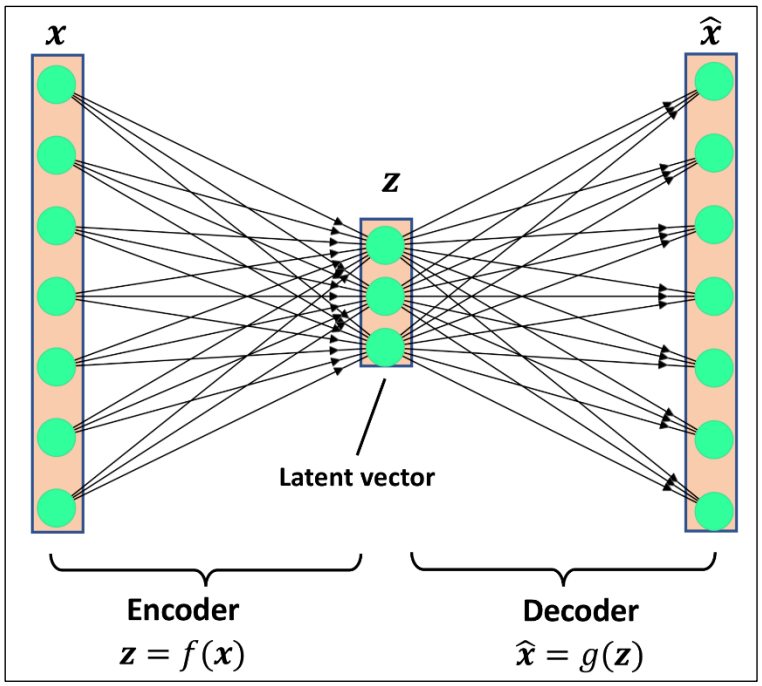
- 80% Mask token: replace fox with [MASK]
- 10% Random token: replace fox with coffee
- 10% Unchanged: keep fox

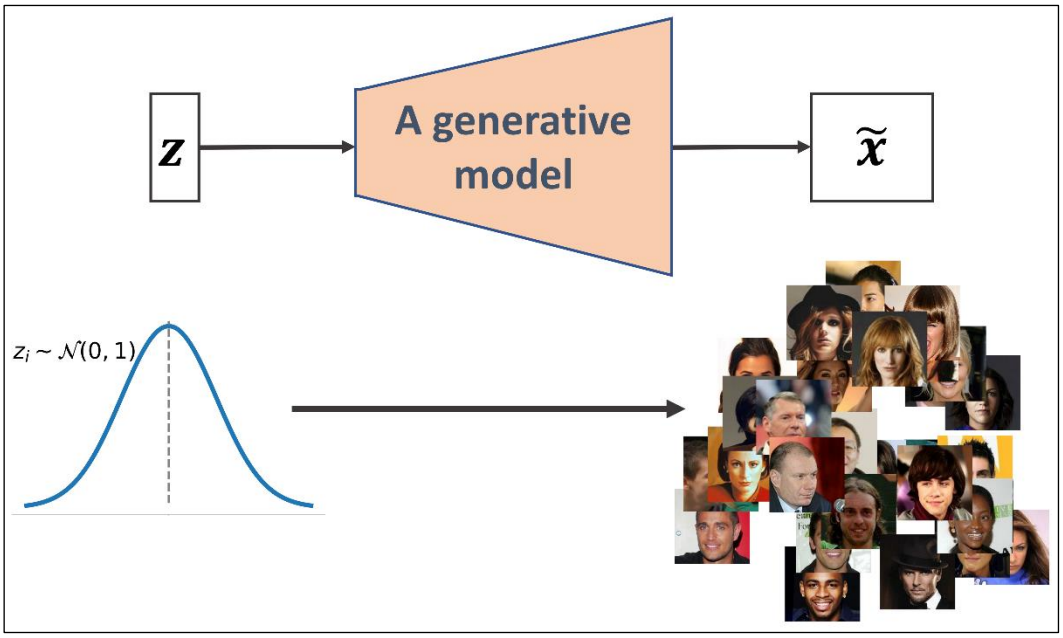

Class Label
C
T_1
T_N
$T_{[SEP]}$
T'_1
T'_M
BERT
$E_{[CLS]}$
E_1
E_N
$E_{[SEP]}$
E'_1
E'_M
[CLS]
Tok 1
Tok N
[SEP]
Tok 1
Tok M
Sentence 1
Sentence 2
(a) Sentence pair classification tasks
Class Label
C
T_1
T_2
T_N
BERT
$E_{[CLS]}$
E_1
E_2
E_N
[CLS]
Tok 1
Tok 2
Tok N
Single sentence
(b) Single sentence classification tasks
Start/End Span
C
T_1
T_N
$T_{[SEP]}$
T'_1
T'_M
BERT
$E_{[CLS]}$
E_1
E_N
$E_{[SEP]}$
E'_1
E'_M
[CLS]
Tok 1
Tok N
[SEP]
Tok 1
Tok M
Question
Paragraph
(c) Question answering tasks
O
B-PER
O
C
T_1
T_2
T_N
BERT
$E_{[CLS]}$
E_1
E_2
E_N
[CLS]
Tok 1
Tok 2
Tok N
Single sentence
(d) Single sentence tagging tasks

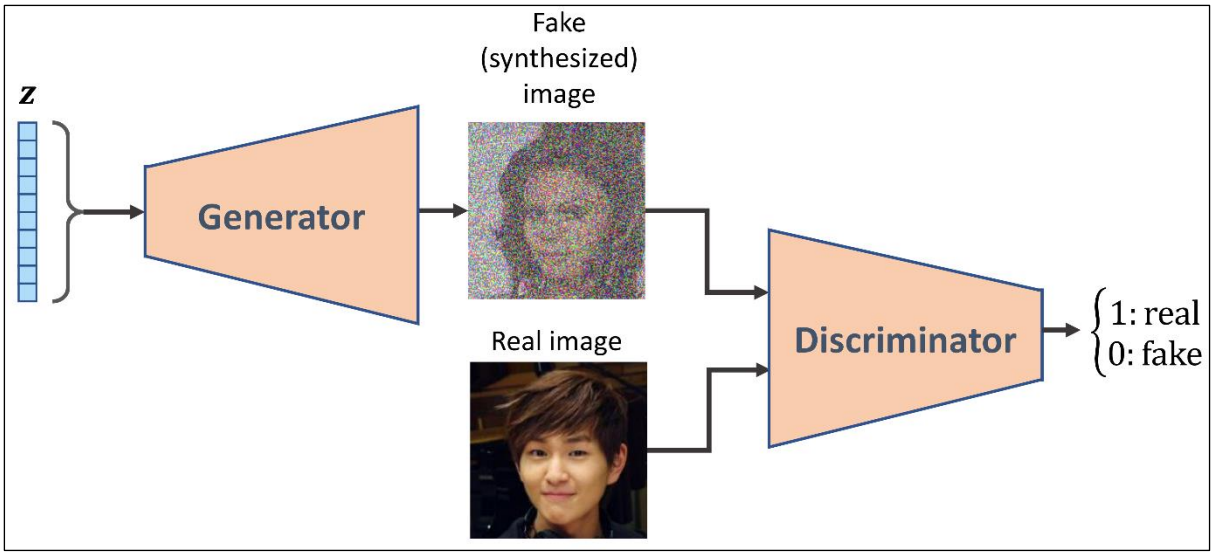

A B C D E
Bidirectional Encoder
Autoregressive Decoder
A _ B _ E
<s> A B C D

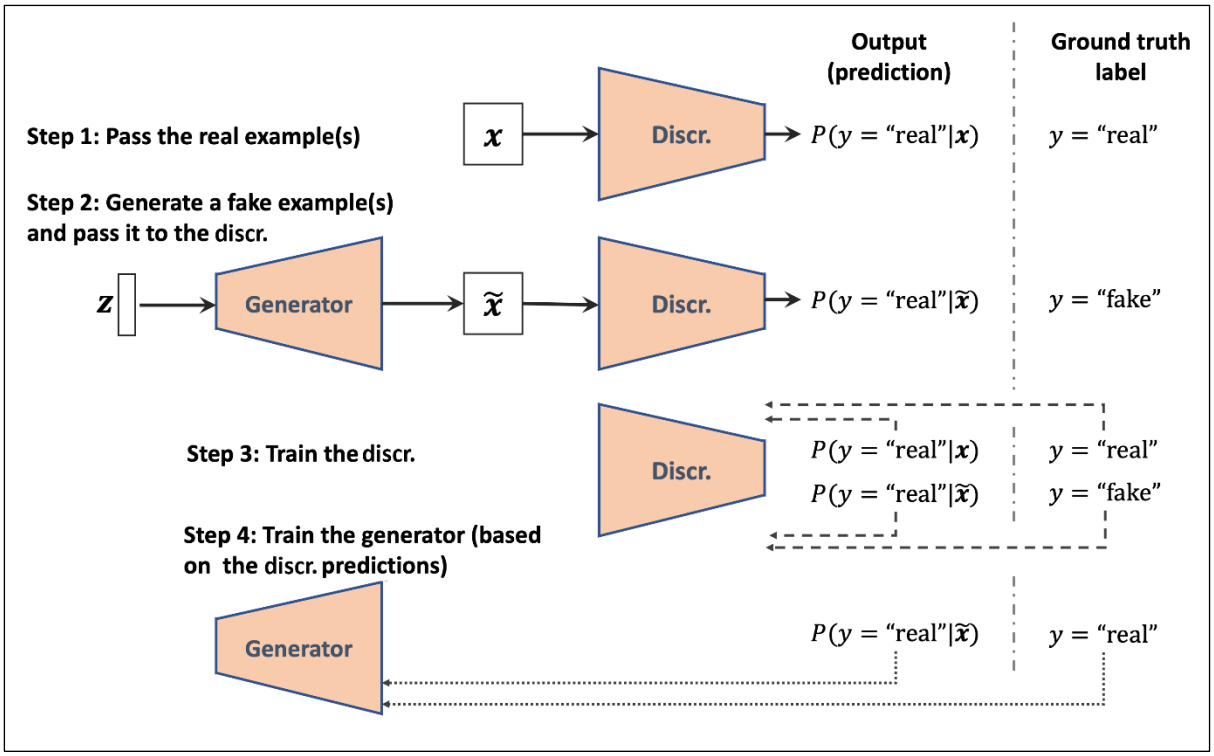
	review	sentiment
0	In 1974, the teenager Martha Moxley (Maggie Gr...	1
1	OK... so... I really like Kris Kristofferson a...	0
2	***SPOILER*** Do not read this, if you think a...	0

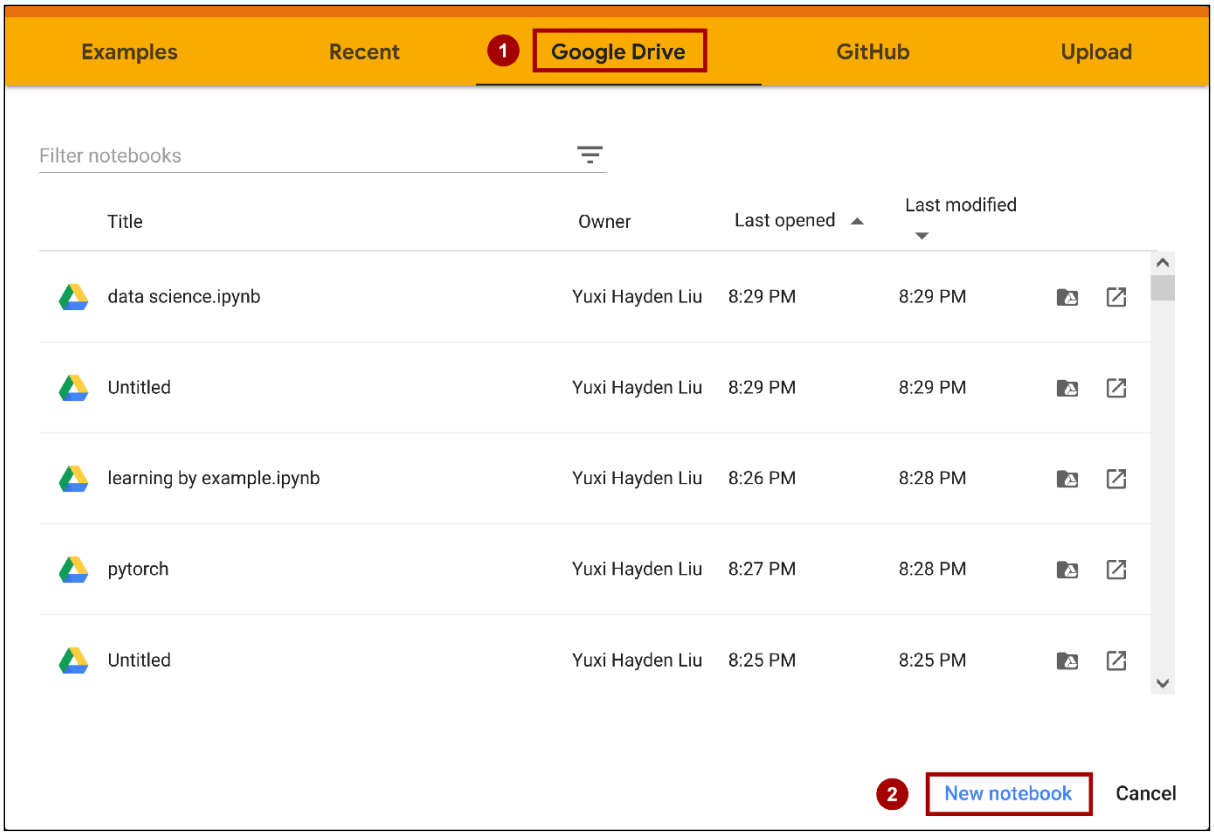
Chapter 17: Generative Adversarial Networks for Synthesizing New Data

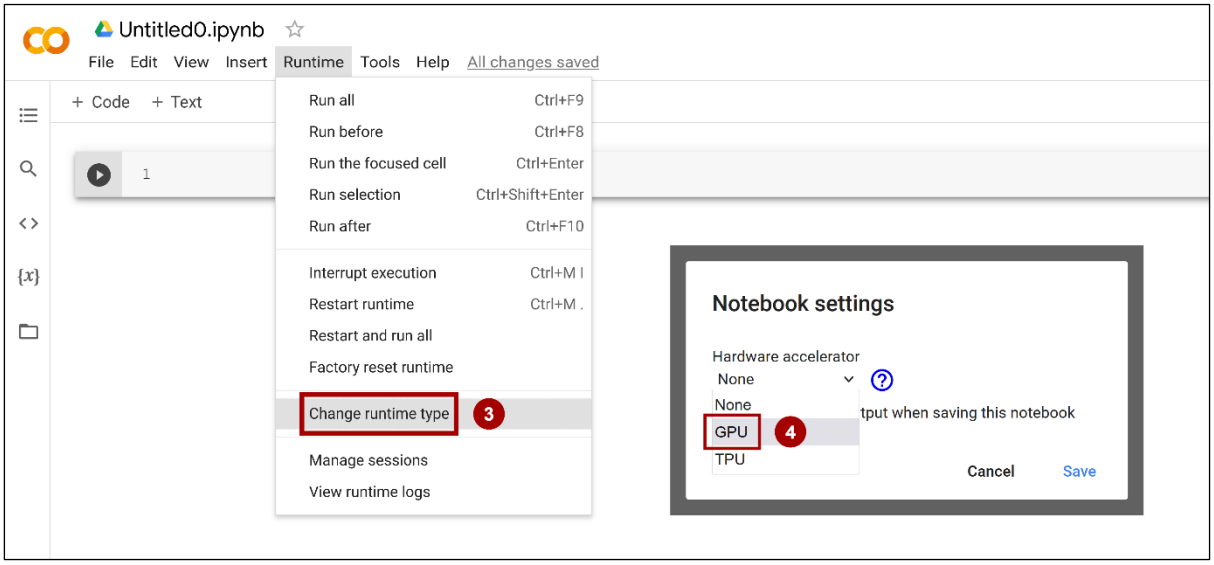


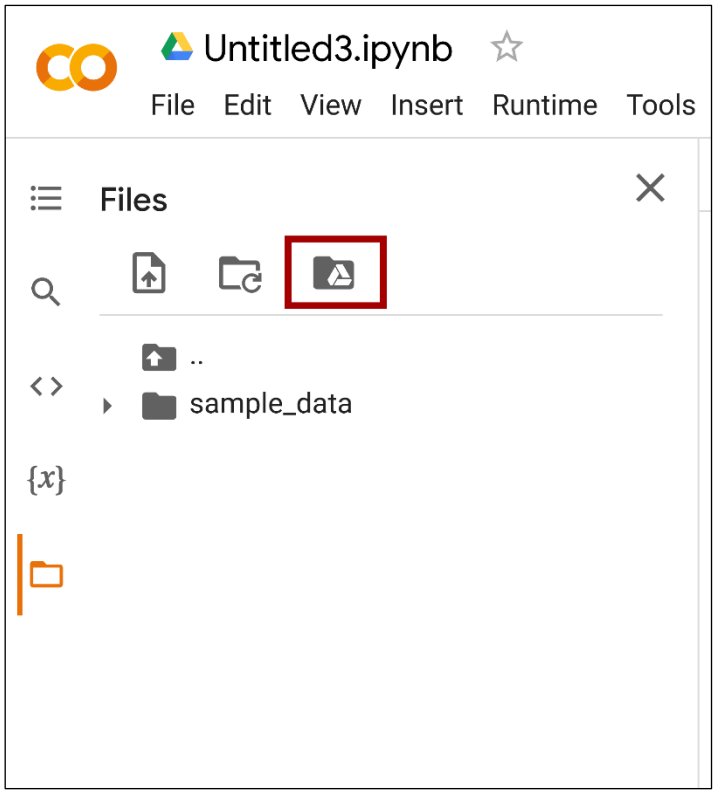


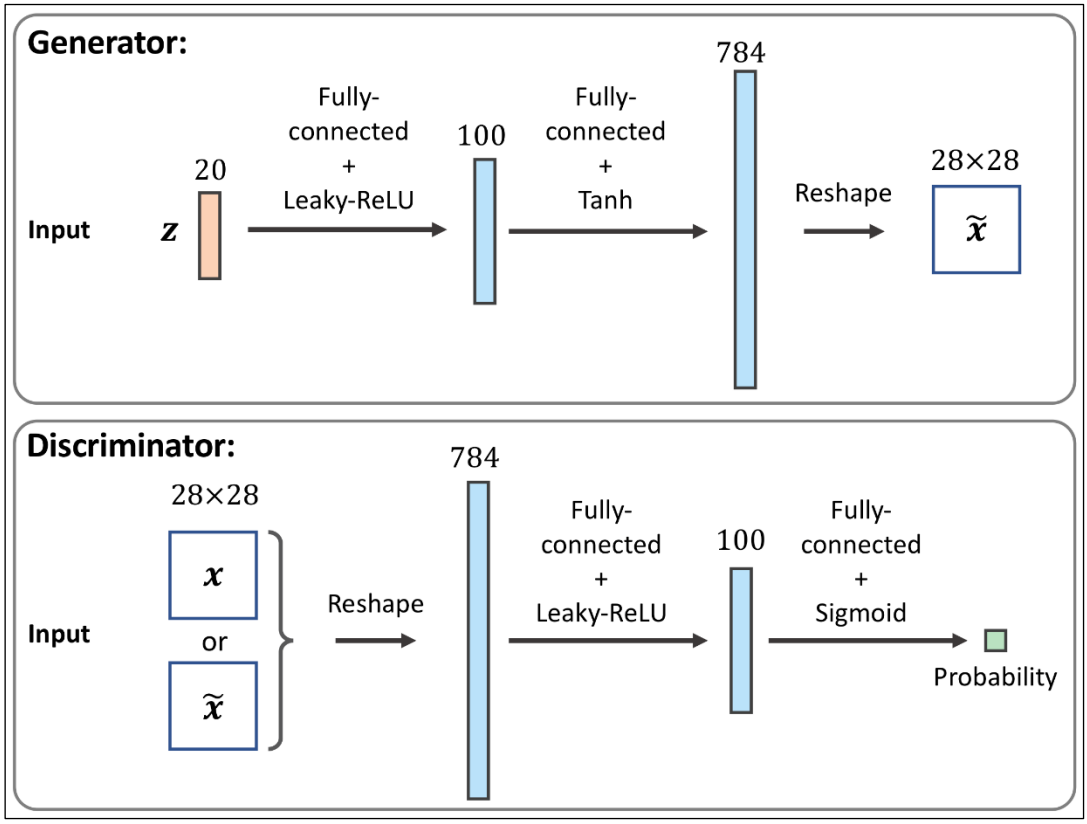

z
Generator
Fake
(synthesized)
image
Real image
Discriminator
1: real
0: fake

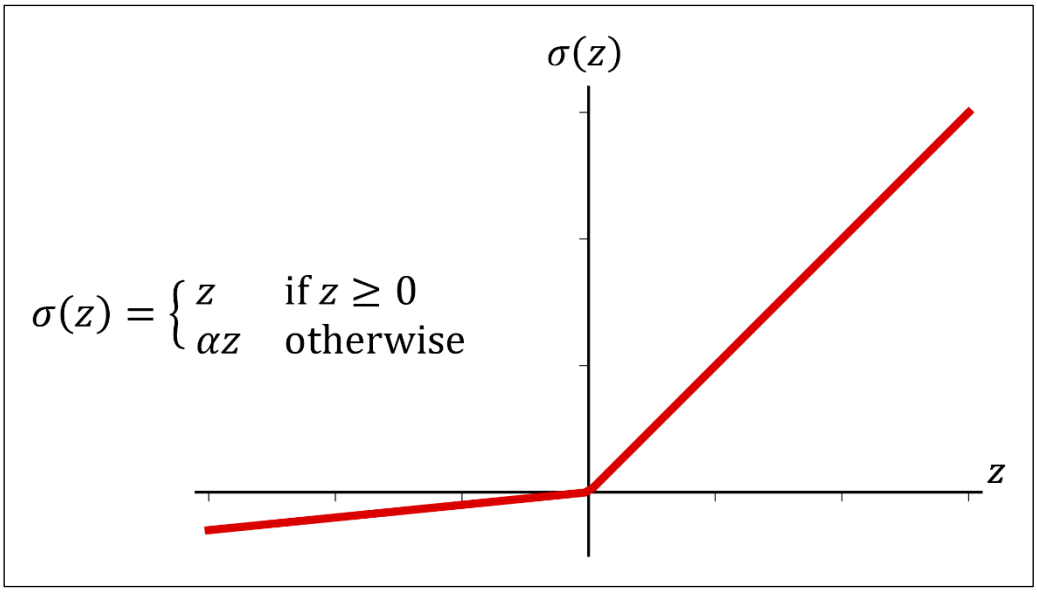

Output
(prediction)
Ground truth
label
Step 1: Pass the real example(s)
x
Discr.
P(y = "real"|x)
y = "real"
Step 2: Generate a fake example(s)
and pass it to the discr.
z
Generator
x̃
Discr.
P(y = "real"|x̃)
y = "fake"
Step 3: Train the discr.
Discr.
P(y = "real"|x)
y = "real"
P(y = "real"|x̃)
y = "fake"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

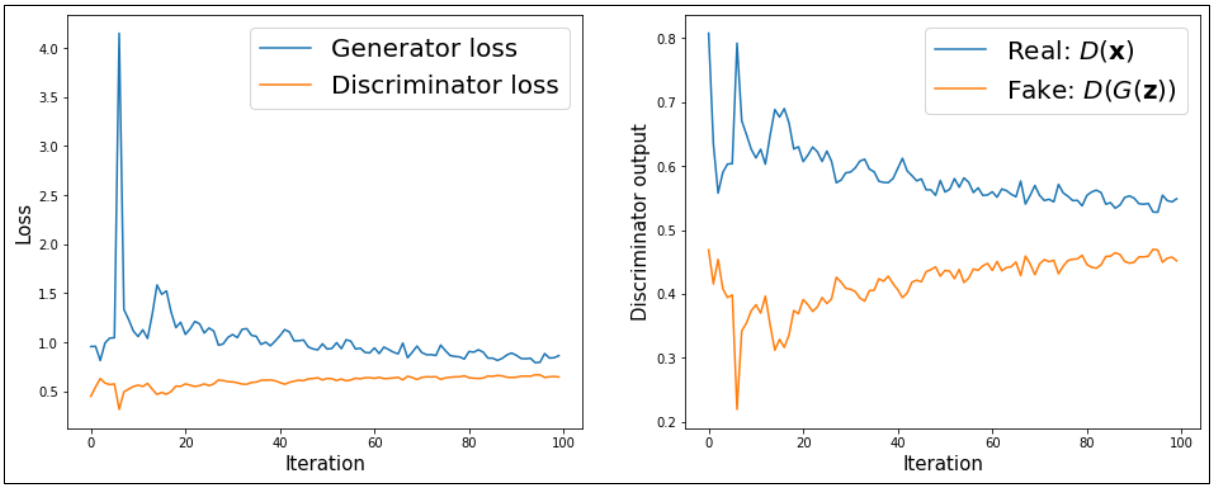

Examples
Recent
1
Google Drive
GitHub
Upload
Filter notebooks
Title
Owner
Last opened
Last modified
data science.ipynb
Yuxi Hayden Liu
8:29 PM
8:29 PM
Untitled
Yuxi Hayden Liu
8:29 PM
8:29 PM
learning by example.ipynb
Yuxi Hayden Liu
8:26 PM
8:28 PM
pytorch
Yuxi Hayden Liu
8:27 PM
8:28 PM
Untitled
Yuxi Hayden Liu
8:25 PM
8:25 PM
2
New notebook
Cancel

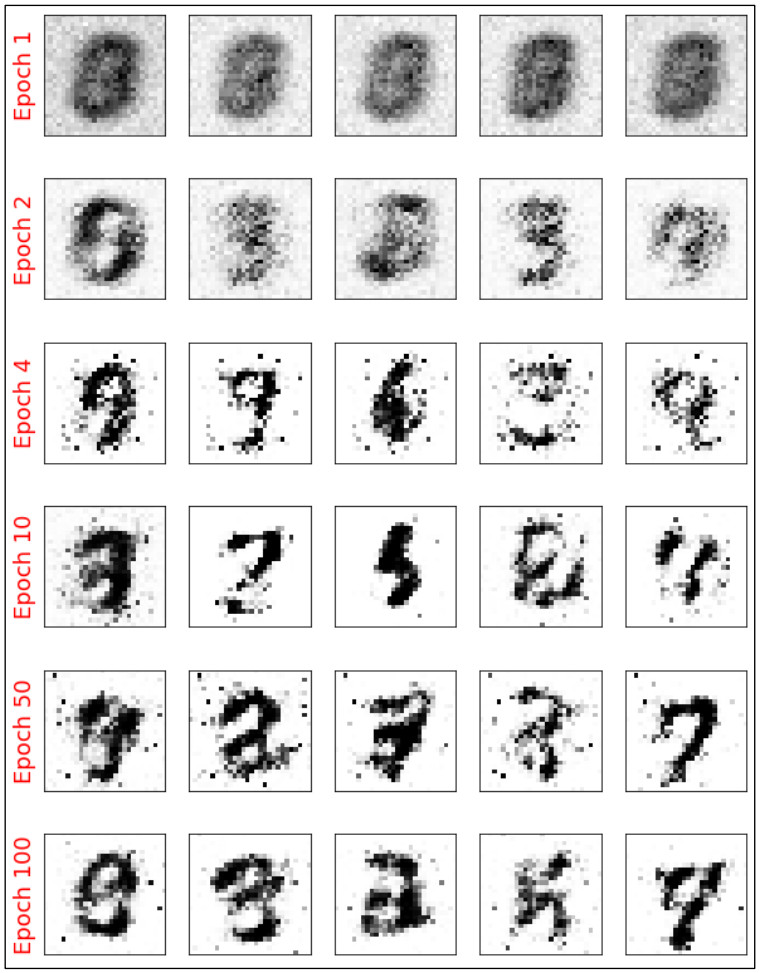

Untitled0.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
All changes saved
+ Code
+ Text
Run all
Ctrl+F9
Run before
Ctrl+F8
Run the focused cell
Ctrl+Enter
Run selection
Ctrl+Shift+Enter
Run after
Ctrl+F10
Interrupt execution
Ctrl+M I
Restart runtime
Ctrl+M .
Restart and run all
Factory reset runtime
Change runtime type
3
Manage sessions
View runtime logs
Notebook settings
Hardware accelerator
None
None
GPU
4
TPU
tput when saving this notebook
Cancel
Save

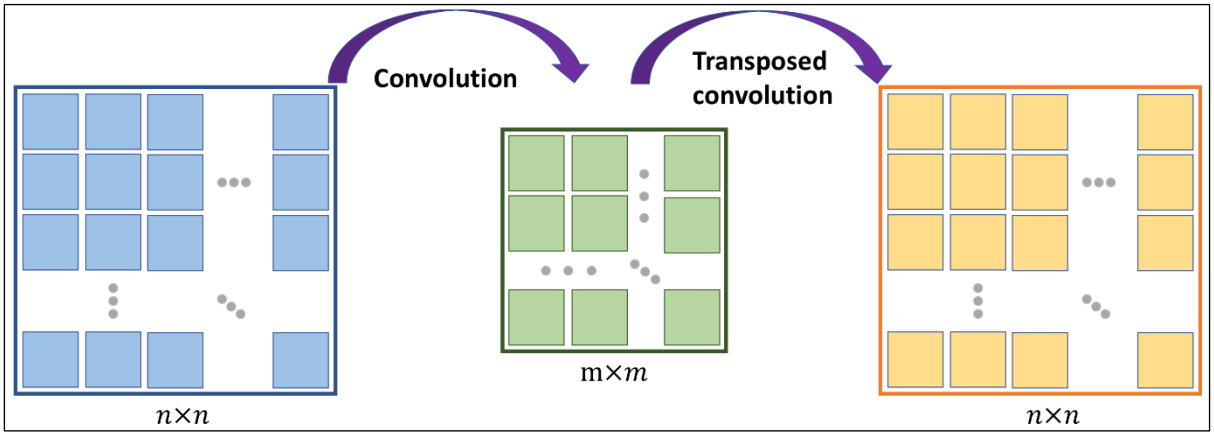

Untitled3.ipynb
File Edit View Insert Runtime Tools
Files
..
sample_data

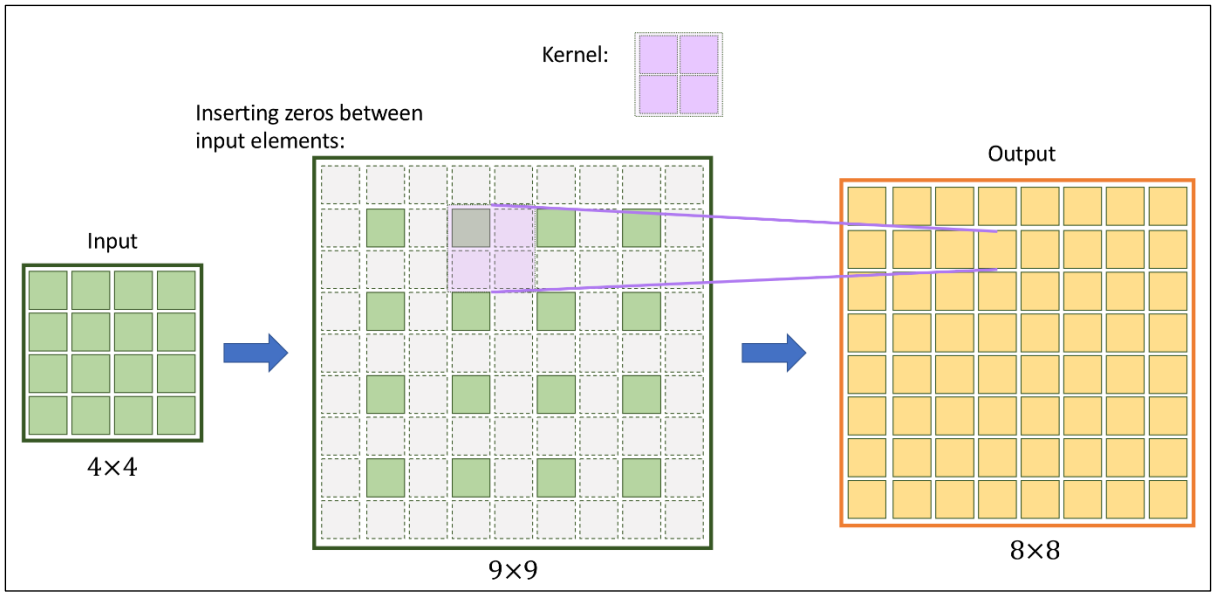

Generator:
Input
z
20
Fully-connected + Leaky-ReLU
100
Fully-connected + Tanh
784
Reshape
28×28
Discriminator:
Input
28×28
or
Reshape
784
Fully-connected + Leaky-ReLU
100
Fully-connected + Sigmoid
Probability

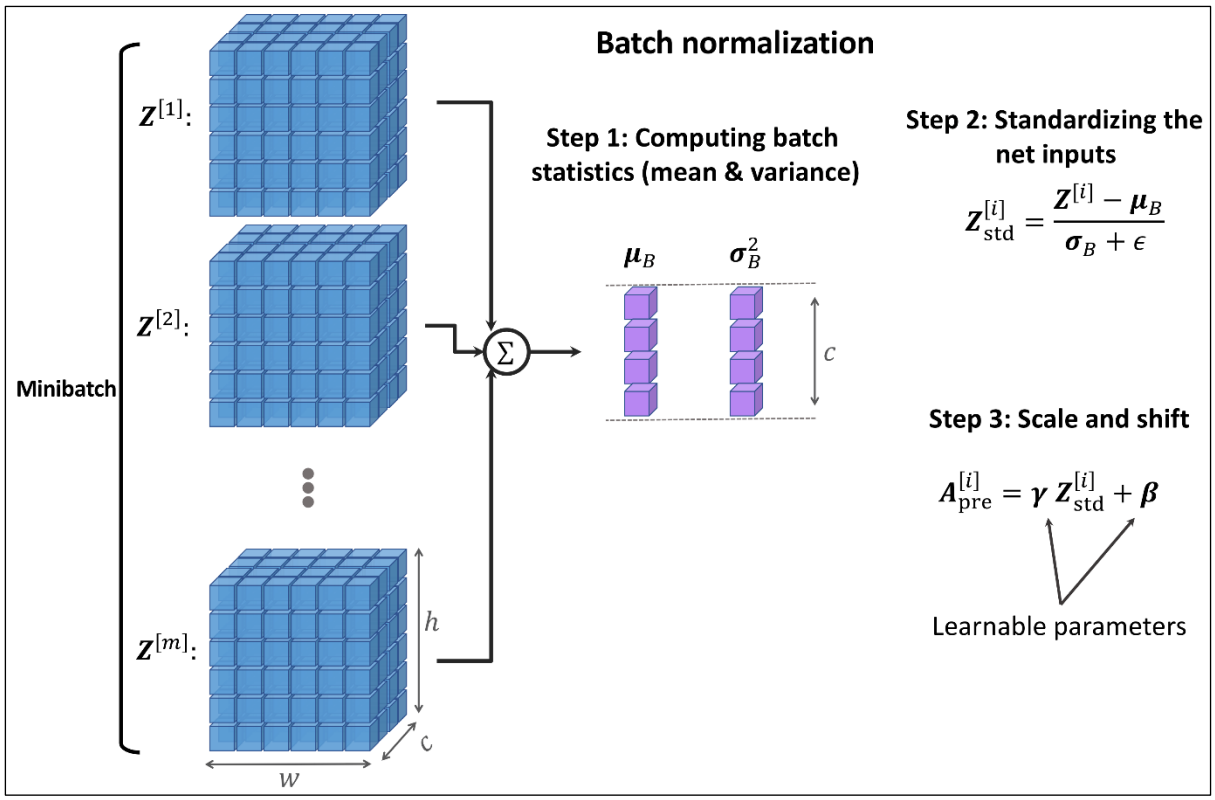

$\sigma(z) = \begin{cases} z & \text{if } z \geq 0 \\ \alpha z & \text{otherwise} \end{cases}$
$\sigma(z)$
z

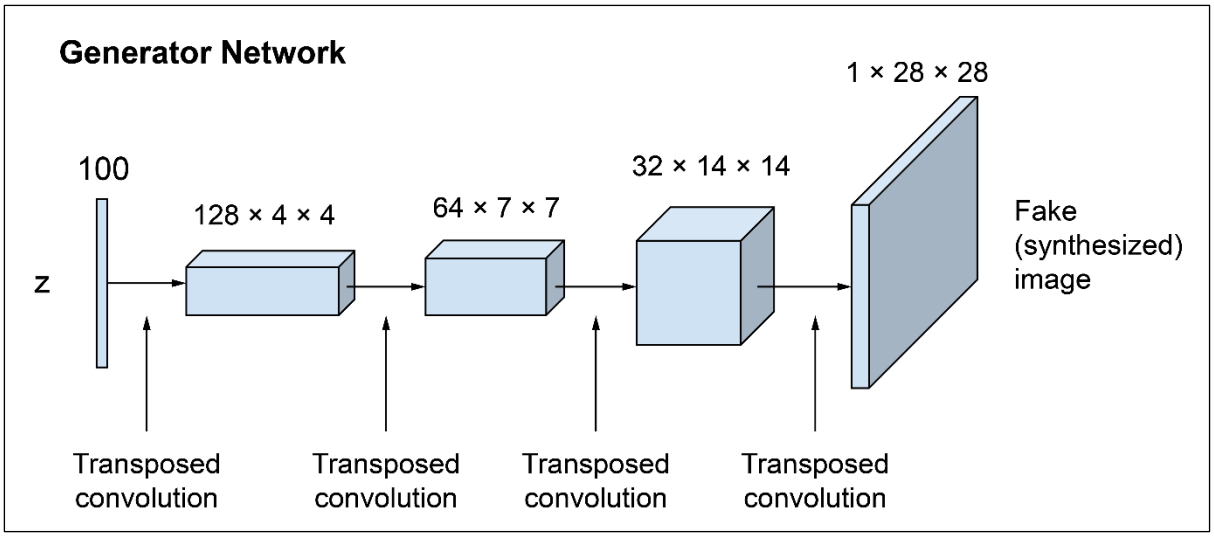

Generator loss
Discriminator loss
Loss
Iteration
Real: $D(\mathbf{x})$
Fake: $D(G(\mathbf{z}))$
Discriminator output
Iteration

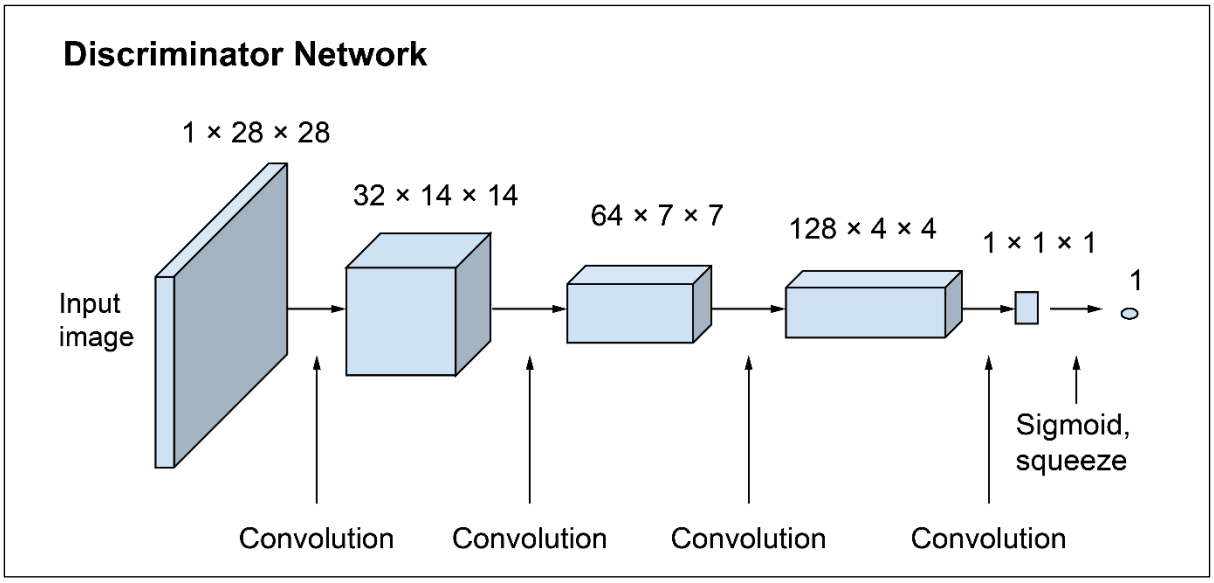

Epoch 1
Epoch 2
Epoch 4
Epoch 10
Epoch 50
Epoch 100

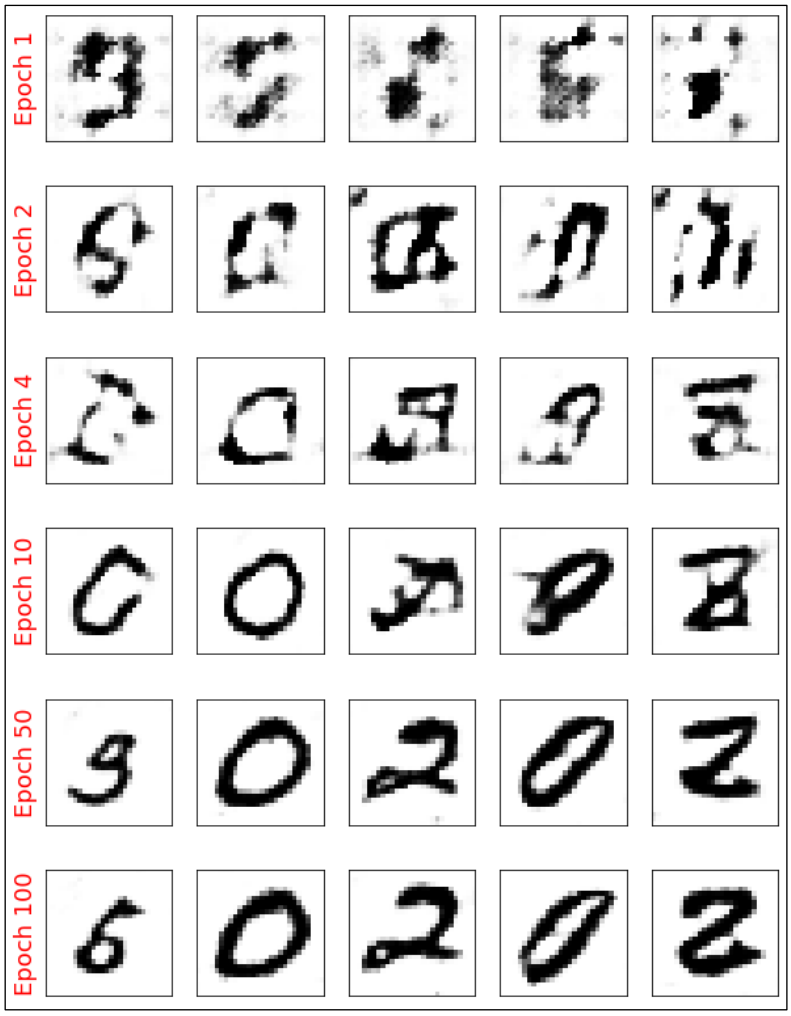

Convolution
Transposed convolution
n×n
m×m
n×n


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

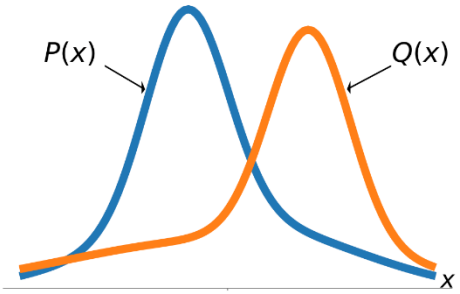

Batch normalization
Minibatch
$Z^{[1]}$:
$Z^{[2]}$:
$Z^{[m]}$:
h
w
c
Step 1: Computing batch statistics (mean & variance)
μ_B
σ_B^2
c
Step 2: Standardizing the net inputs
$Z_{\text{std}}^{[i]} = \frac{Z^{[i]} - \mu_B}{\sigma_B + \epsilon}$
Step 3: Scale and shift
$A_{\text{pre}}^{[i]} = \gamma Z_{\text{std}}^{[i]}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

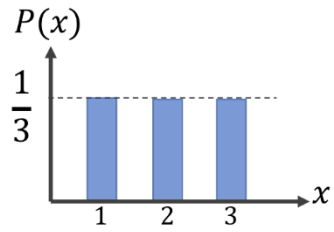

Generator Network
100
z
128 × 4 × 4
64 × 7 × 7
32 × 14 × 14
1 × 28 × 28
Fake
(synthesized)
image
Transposed
convolution
Transposed
convolution
Transposed
convolution
Transposed
convolution

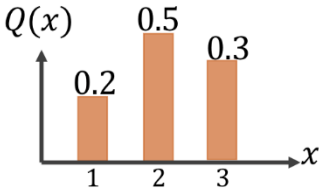

Discriminator Network
1 × 28 × 28
Input
image
32 × 14 × 14
64 × 7 × 7
128 × 4 × 4
1 × 1 × 1
1
Sigmoid,
squeeze
Convolution
Convolution
Convolution
Convolution



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







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:

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

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

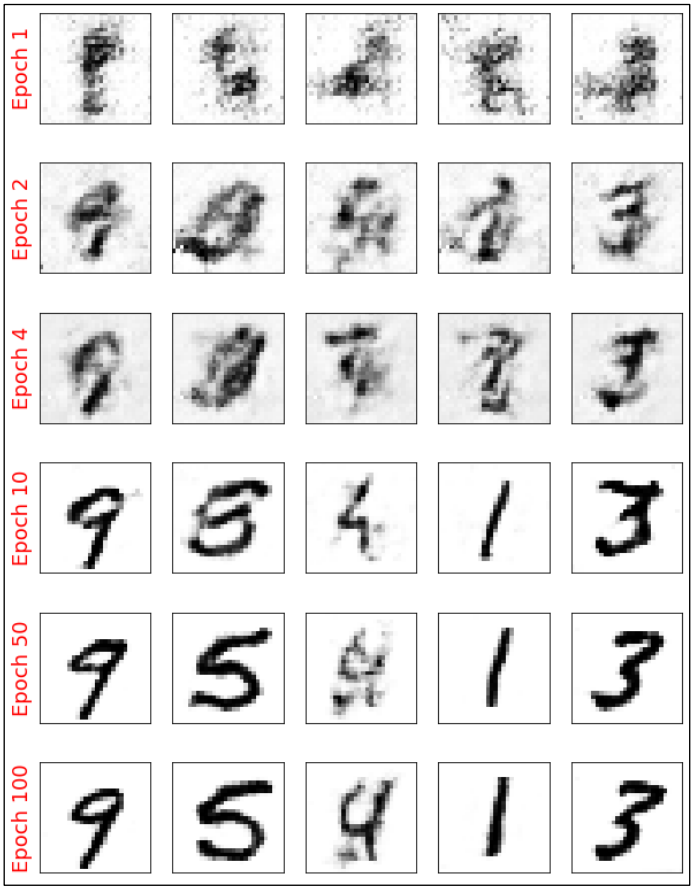
JS divergence:

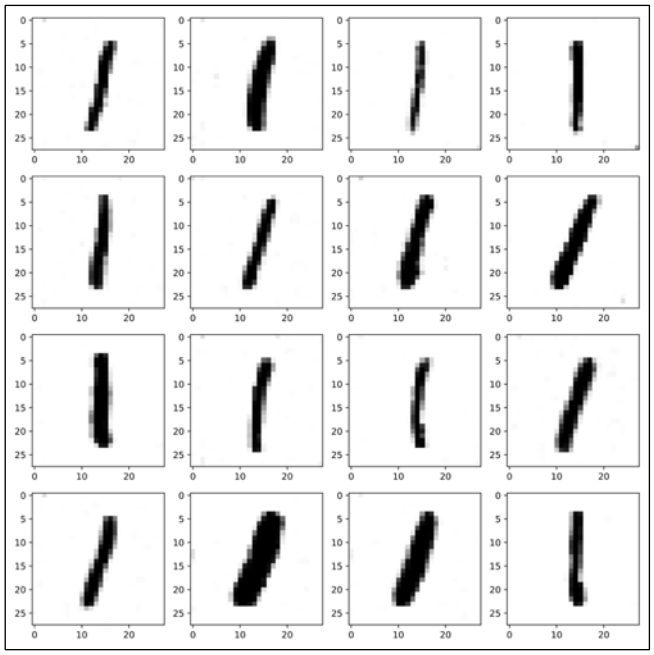
$$P_m \rightarrow \left[\frac{0.33 + 0.2}{2}, \frac{0.33 + 0.5}{2}, \frac{0.33 + 0.3}{2}\right] = [0.26, 0.42, 0.32]$$

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

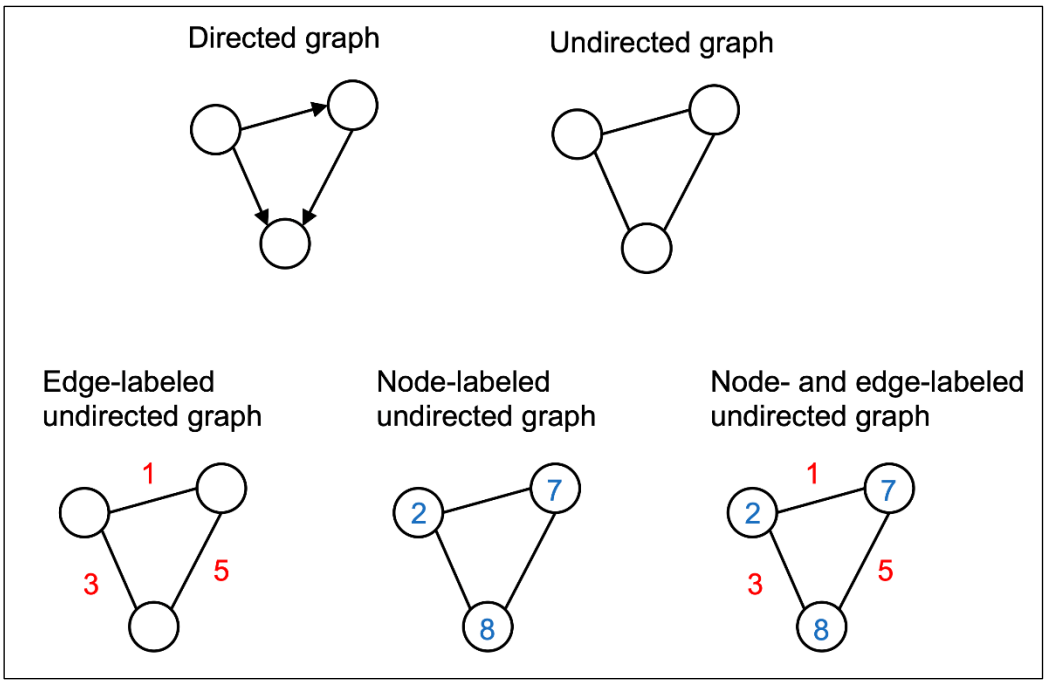
EM distance:

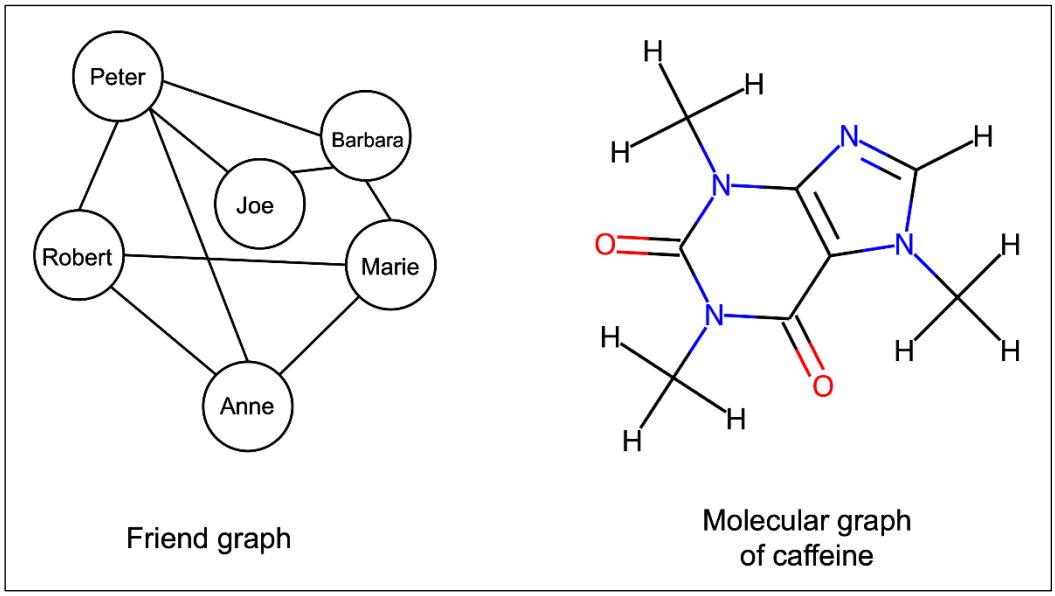
$$EM(P,Q) = (0.33 - 0.2) + (0.33 - 0.3) = 0.16$$

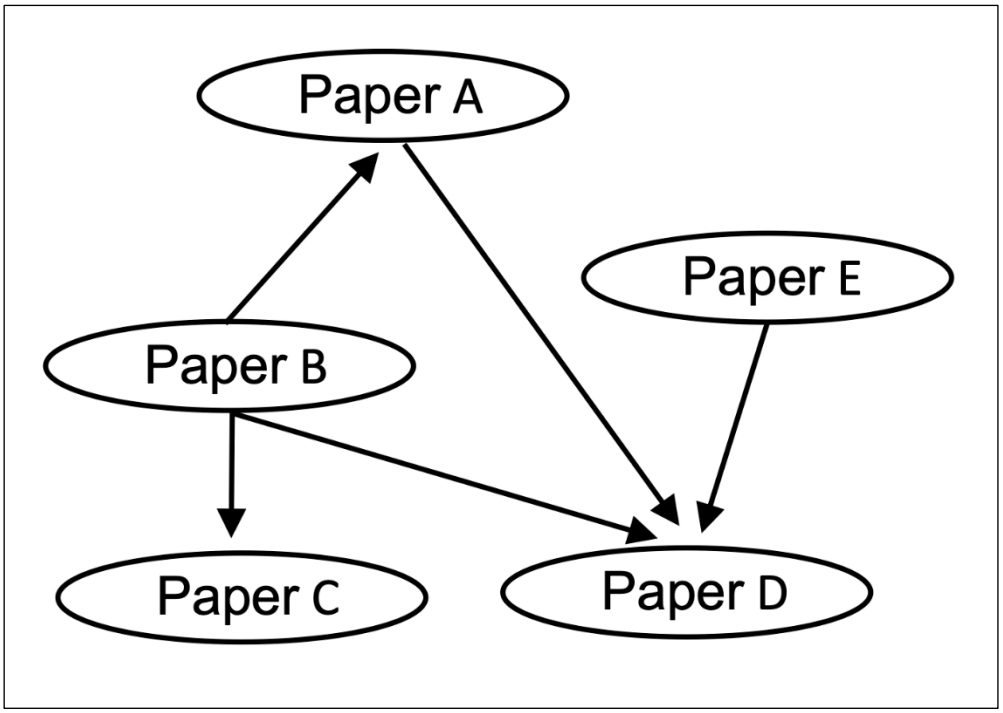


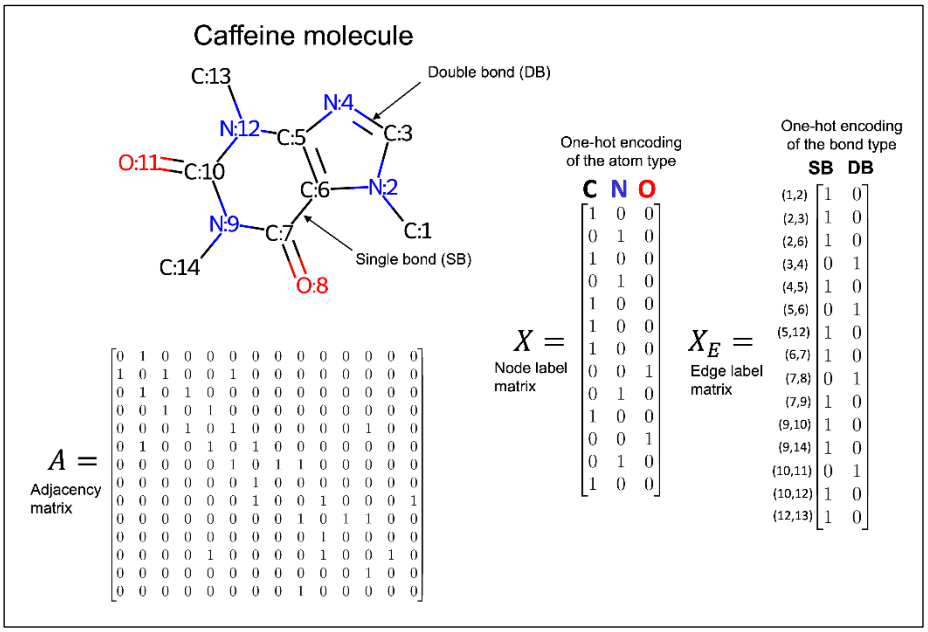


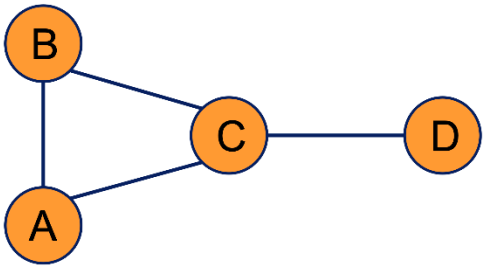
Chapter 18: Graph Neural Networks for Capturing Dependencies in Graph Structured Data






Paper A
Paper E
Paper B
Paper C
Paper D


Caffeine molecule
C:13
N:4
Double bond (DB)
N:12
C:5
C:3
O:11
C:10
C:6
N:2
N:9
C:7
C:1
C:14
Single bond (SB)
O:8
$A =$
Adjacency matrix
$X =$
Node label matrix
One-hot encoding of the atom type
C N O
$X_E =$
Edge label matrix
One-hot encoding of the bond type
SB DB
(1,2)
(2,3)
(2,6)
(3,4)
(4,5)
(5,6)
(5,12)
(6,7)
(7,8)
(7,9)
(9,10)
(9,14)
(10,11)
(10,12)
(12,13)



Adjacency matrix 1:

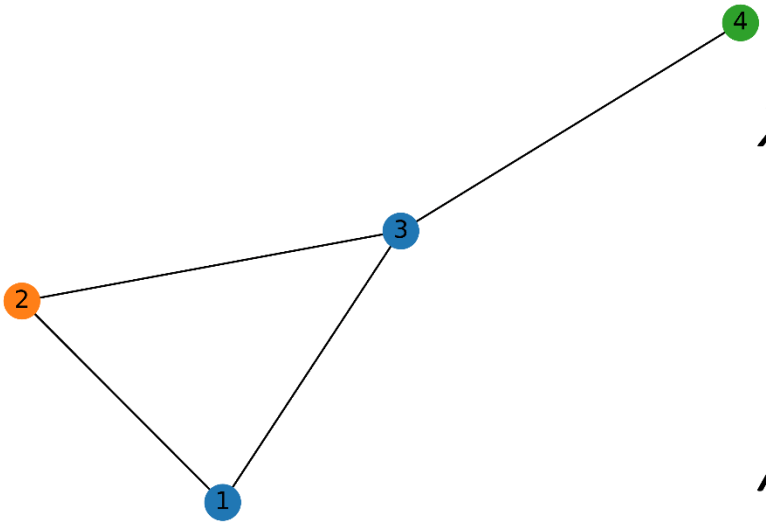
$$\begin{array}{c} \\ \mathbf{D} \\ \mathbf{B} \\ \mathbf{A} \\ \mathbf{C} \end{array}\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Adjacency matrix 2:

$$\begin{array}{c} \\ \mathbf{A} \\ \mathbf{C} \\ \mathbf{D} \\ \mathbf{B} \end{array}\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

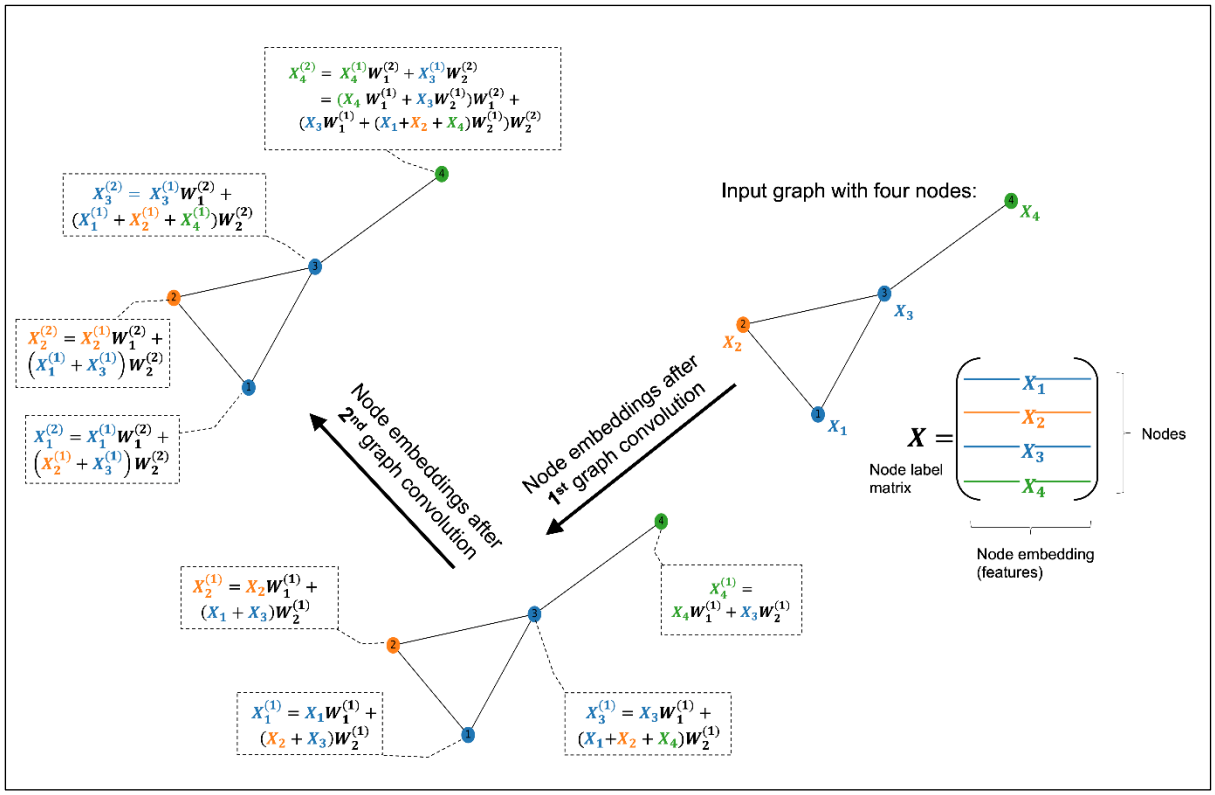
Adjacency matrix 3:

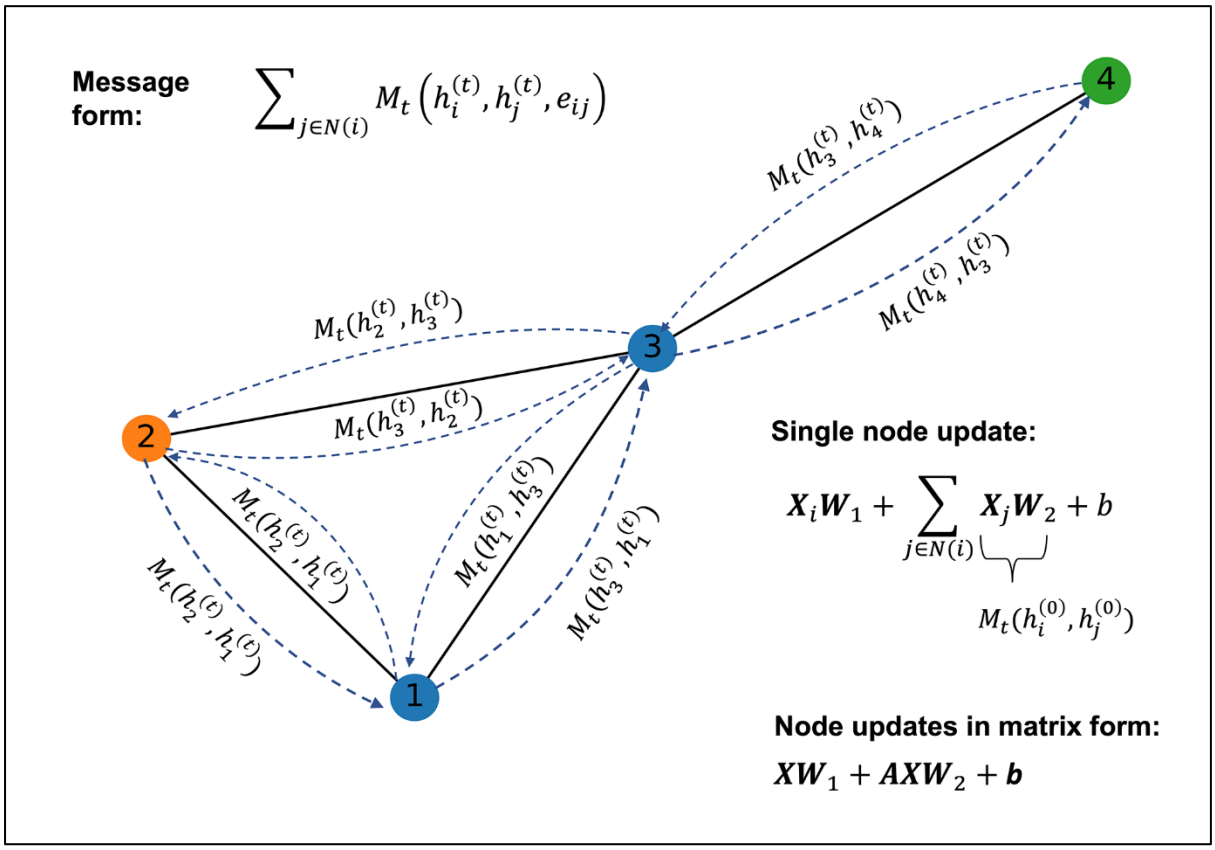
$$\begin{array}{c} \\ \mathbf{A} \\ \mathbf{B} \\ \mathbf{C} \\ \mathbf{D} \end{array}\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

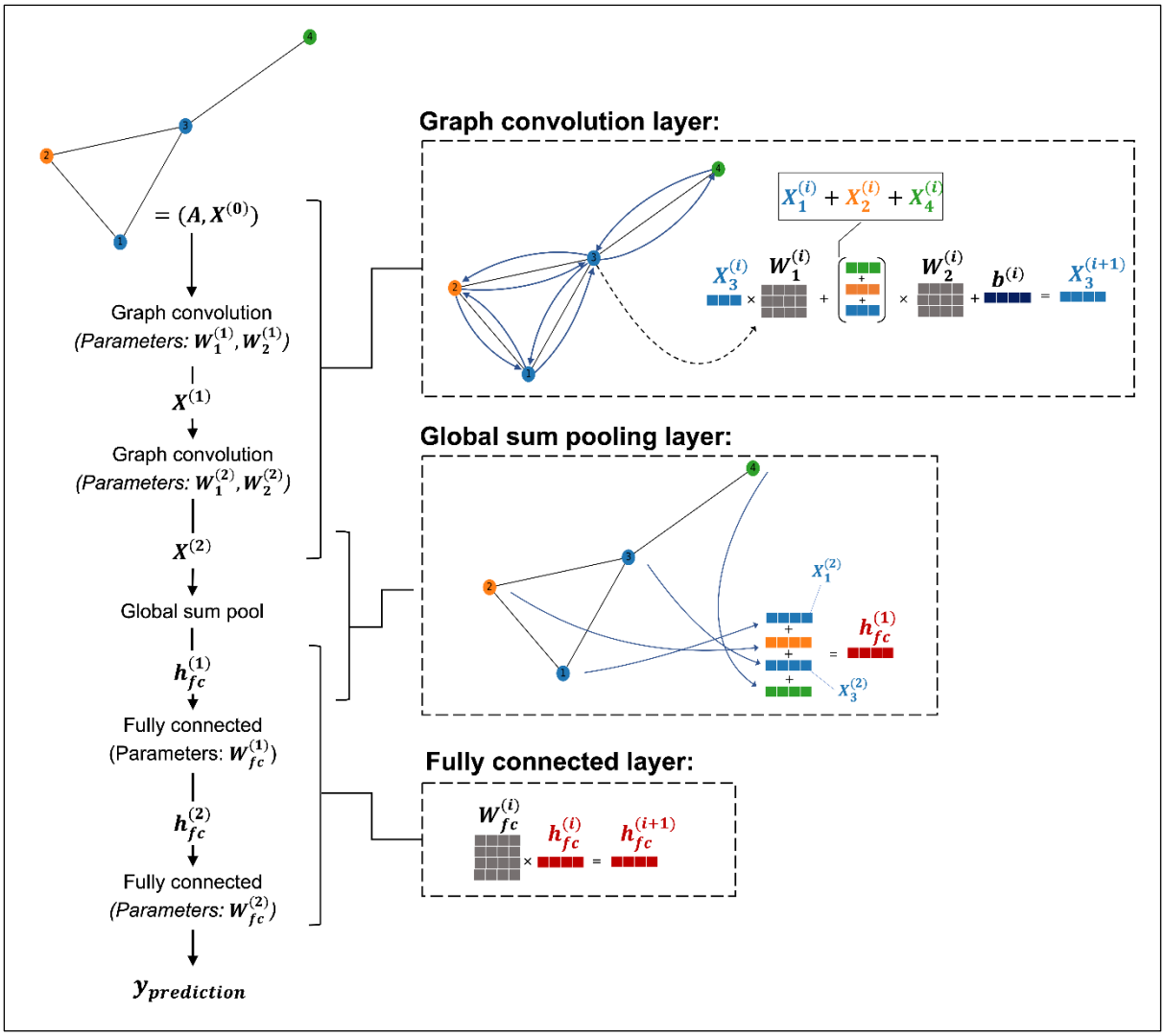


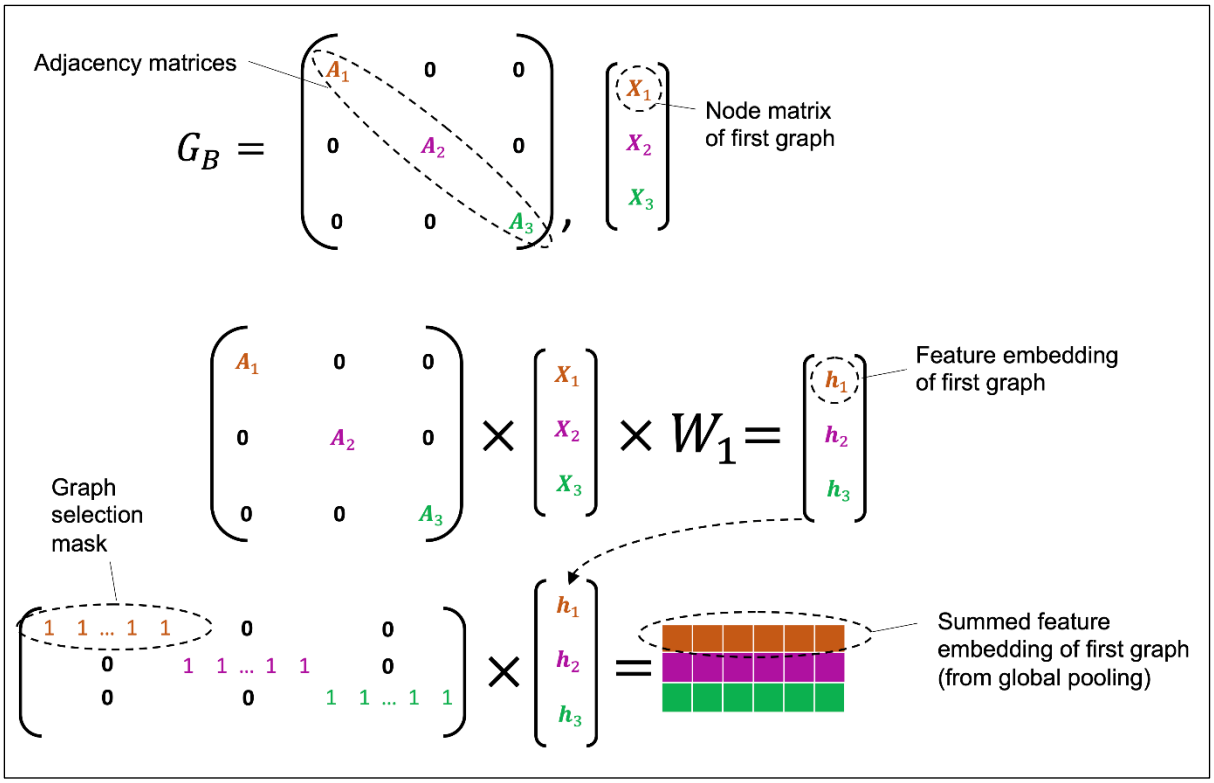
$$X = \begin{array}{c} \mathbf{G}\ \ \mathbf{B}\ \ \mathbf{O} \\ \end{array}\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

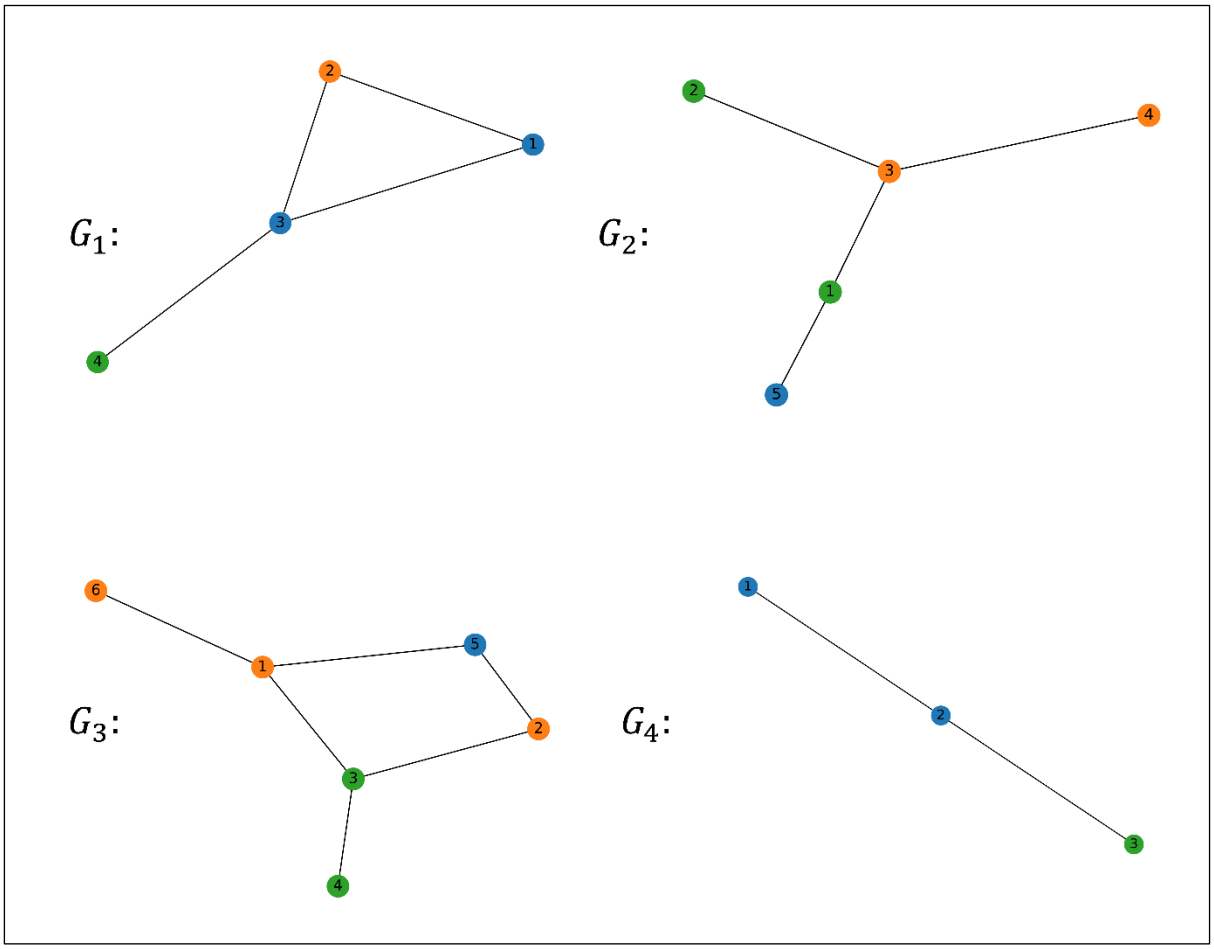
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

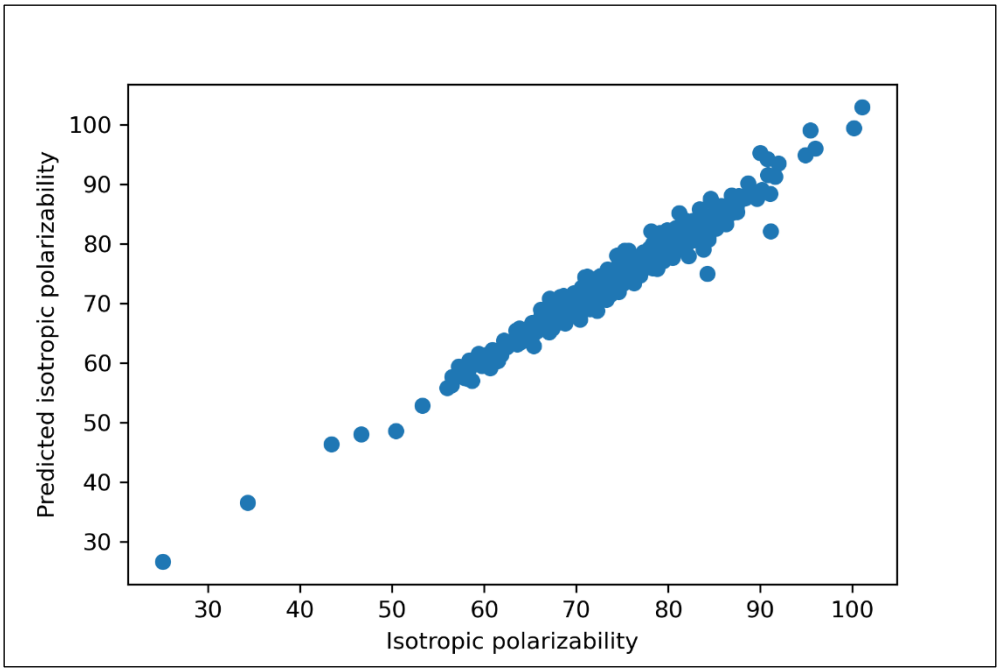

Input graph with four nodes:
Node embeddings after 1st graph convolution
Node embeddings after 2nd graph convolution
Nodes
Node label matrix
Node embedding (features)

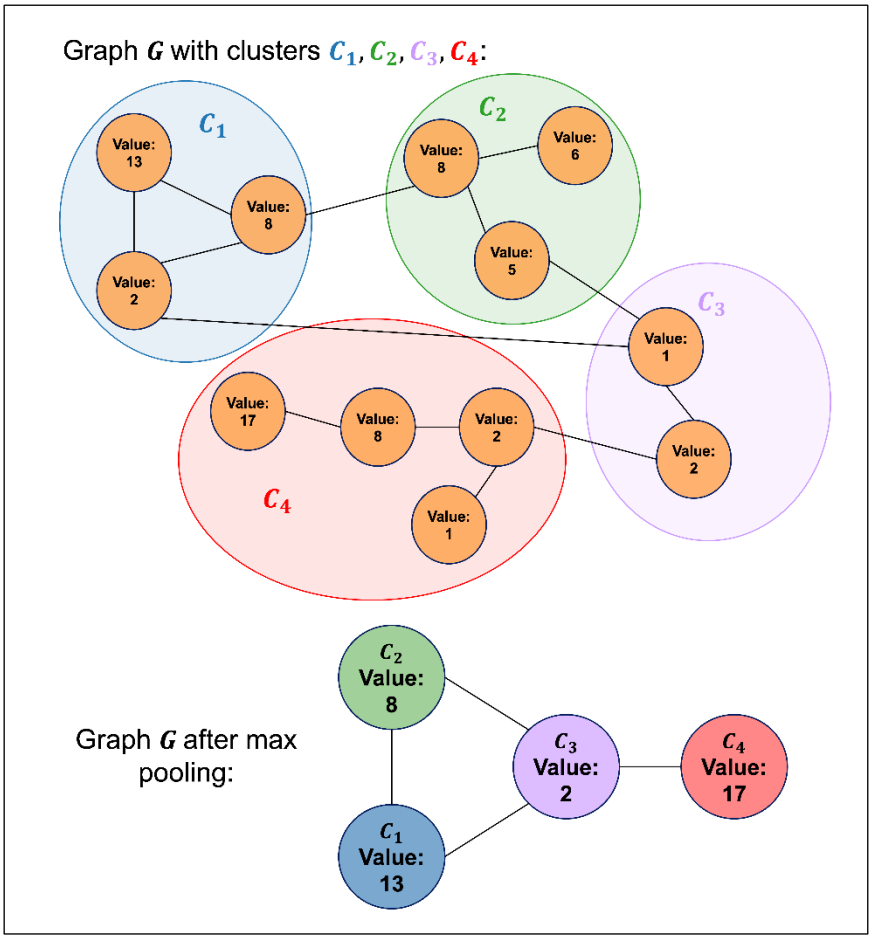

Message form:
Single node update:
Node updates in matrix form:


$= (A, X^{(0)})$
Graph convolution
(Parameters: $W_1^{(1)}, W_2^{(1)}$)
$X^{(1)}$
Graph convolution
(Parameters: $W_1^{(2)}, W_2^{(2)}$)
$X^{(2)}$
Global sum pool
$h_{fc}^{(1)}$
Fully connected
(Parameters: $W_{fc}^{(1)}$)
$h_{fc}^{(2)}$
Fully connected
(Parameters: $W_{fc}^{(2)}$)
$y_{prediction}$
Graph convolution layer:
$X_1^{(i)} + X_2^{(i)} + X_4^{(i)}$
$X_3^{(i)} \times W_1^{(i)} + (\quad) \times W_2^{(i)} + b^{(i)} = X_3^{(i+1)}$
Global sum pooling layer:
$X_1^{(2)}$
$X_3^{(2)}$
$h_{fc}^{(1)}$
Fully connected layer:
$W_{fc}^{(i)} \times h_{fc}^{(i)} = h_{fc}^{(i+1)}$

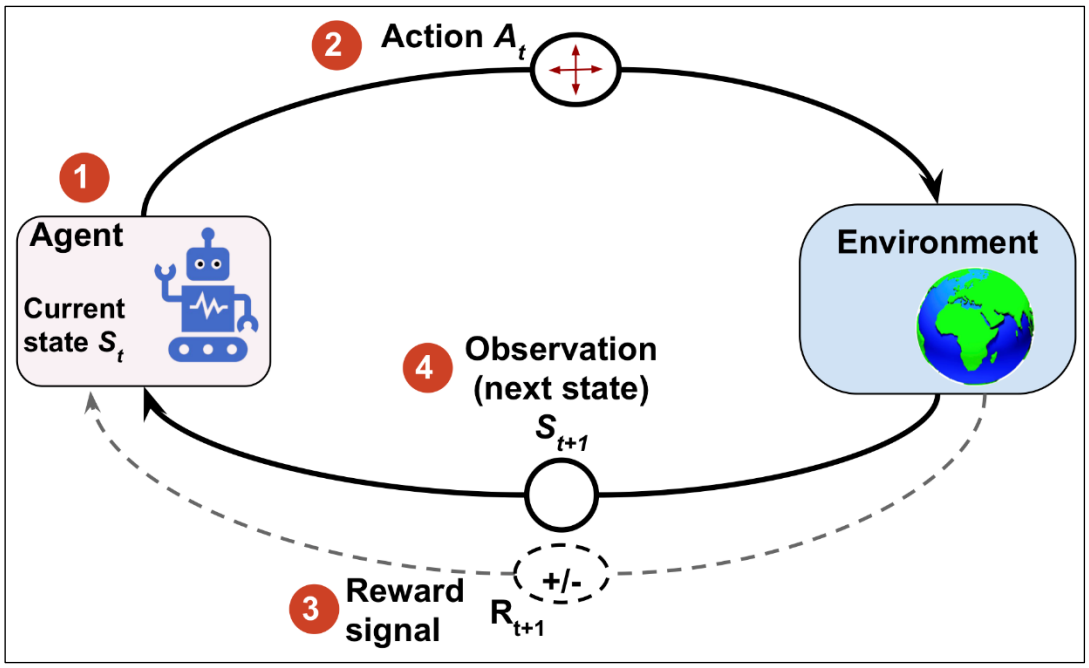

Adjacency matrices
$G_B =$
A_1 0 0
0 A_2 0
0 0 A_3
,
X_1
X_2
X_3
Node matrix of first graph
A_1 0 0
0 A_2 0
0 0 A_3
× X_1 X_2 X_3 × W_1 =
h_1
h_2
h_3
Feature embedding of first graph
Graph selection mask
1 1 ... 1 1 0 0
0 1 1 ... 1 1 0
0 0 1 1 ... 1 1
× h_1 h_2 h_3 =
Summed feature embedding of first graph (from global pooling)

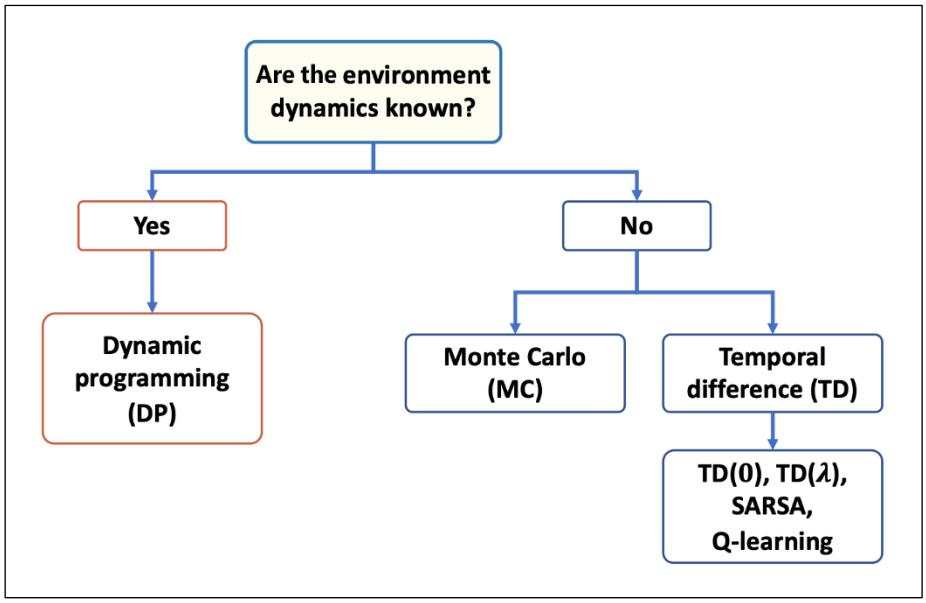

G_1:
G_2:
G_3:
G_4:


Predicted isotropic polarizability
Isotropic polarizability
100
90
80
70
60
50
40
30
30
40
50
60
70
80
90
100

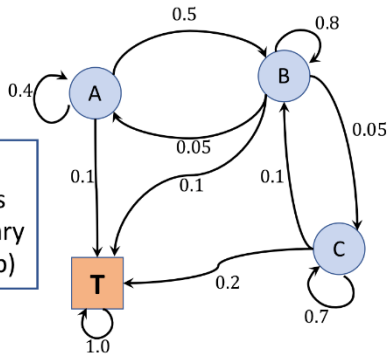

Graph G with clusters C_1, C_2, C_3, C_4:
C_1
Value: 13
Value: 8
Value: 2
C_2
Value: 8
Value: 6
Value: 5
C_3
Value: 1
Value: 2
C_4
Value: 17
Value: 8
Value: 2
Value: 1
Graph G after max pooling:
C_2 Value: 8
C_3 Value: 2
C_4 Value: 17
C_1 Value: 13

Chapter 19: Reinforcement Learning for Decision Making in Complex Environments



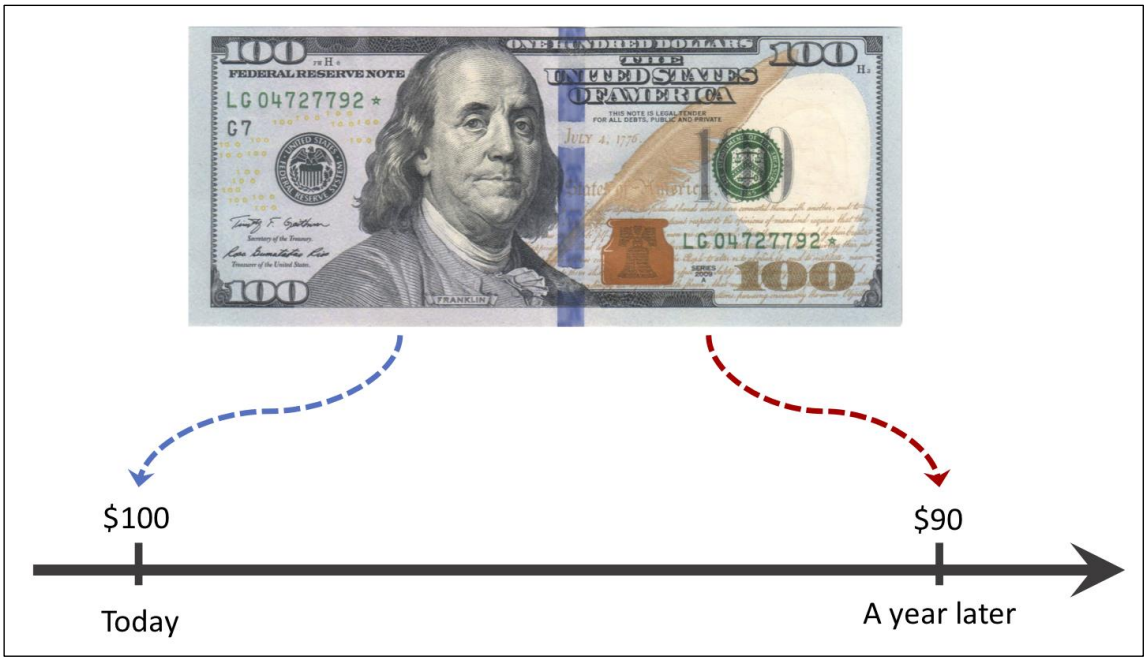


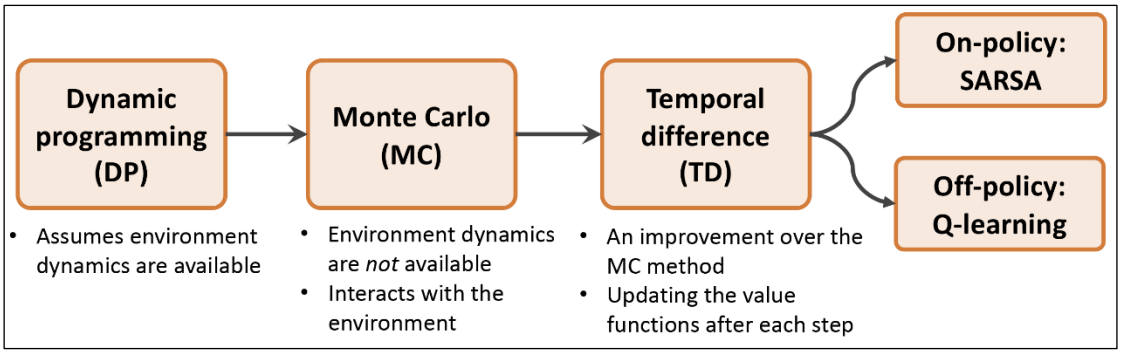
A: studying at home
B: playing video games
C: studying at the library
T: terminal state (sleep)

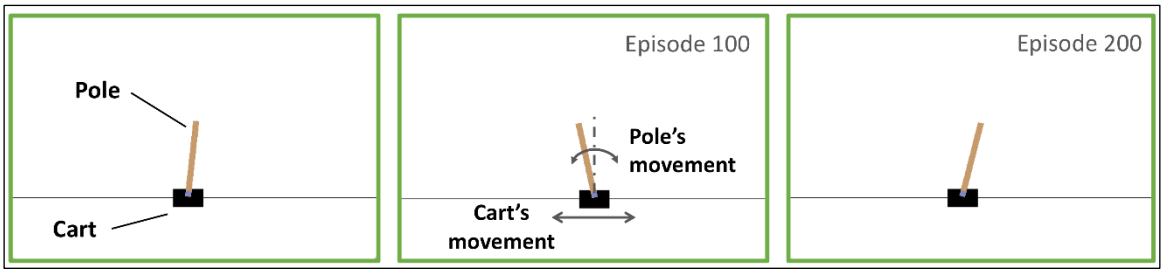


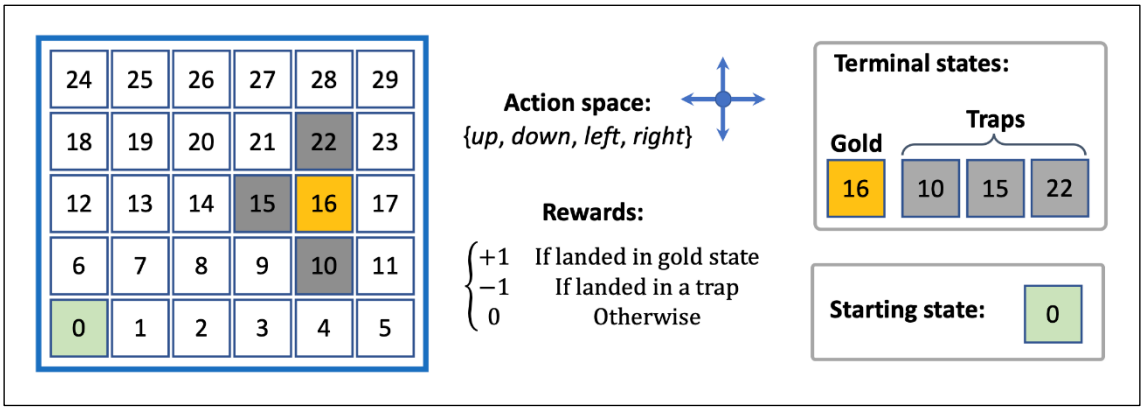
Transition probabilities:

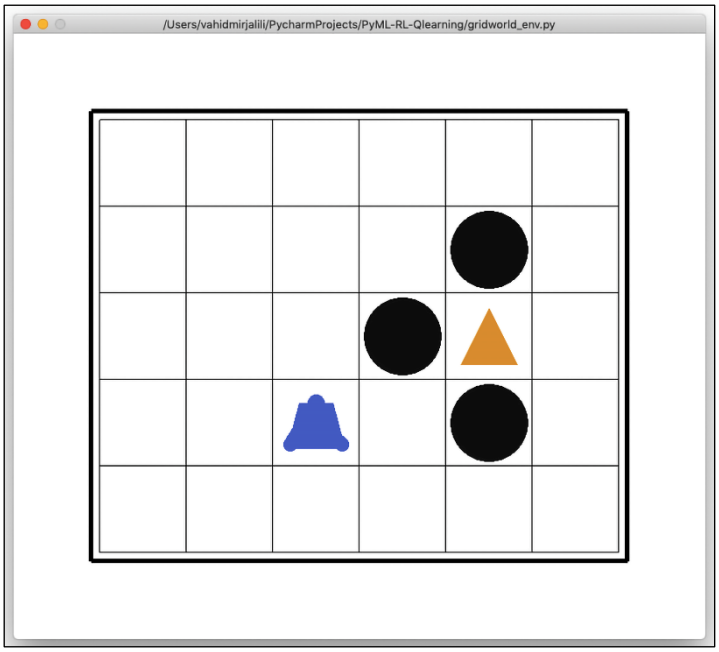
	A	*B*	*C*	*T*
A	0.4	0.5	0.0	0.1
B	0.05	0.8	0.05	0.1
C	0.0	0.1	0.7	0.2
T	0.0	0.0	0.0	1.0

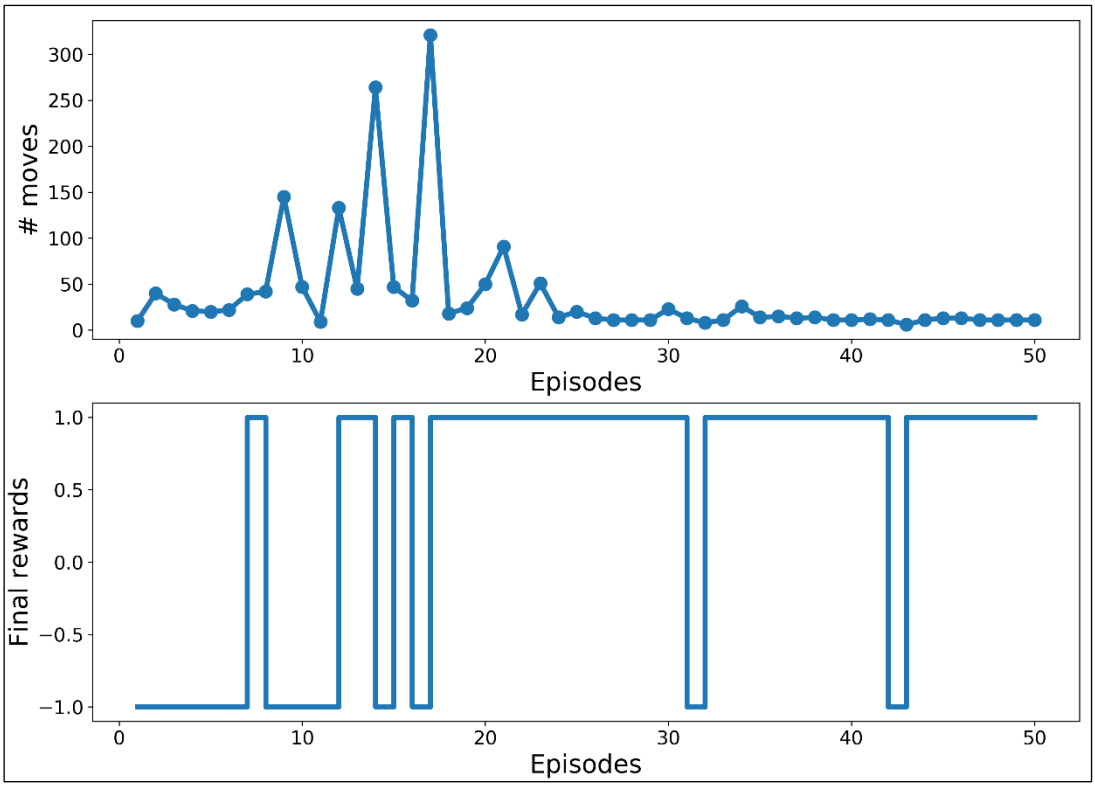


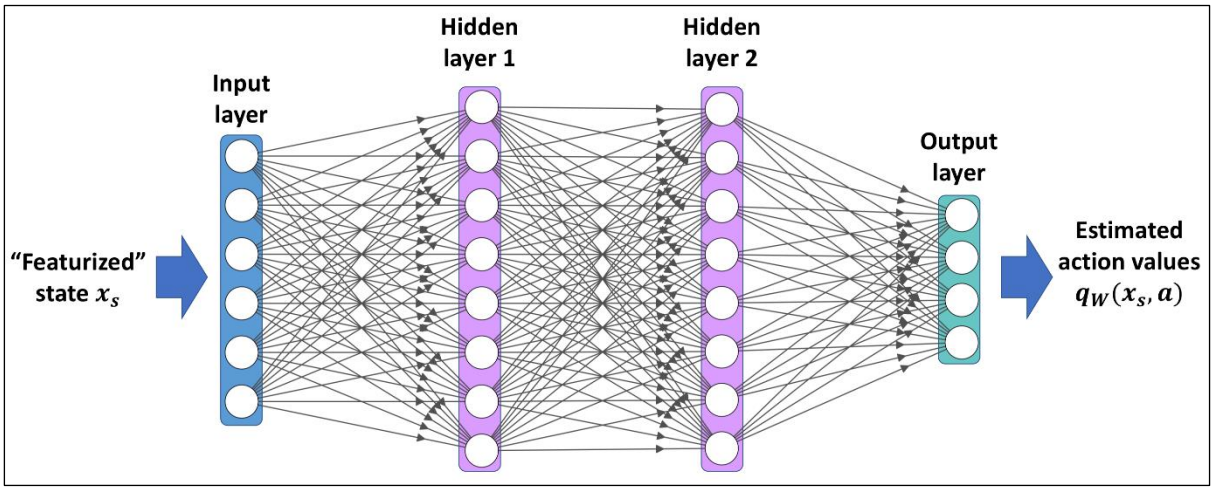


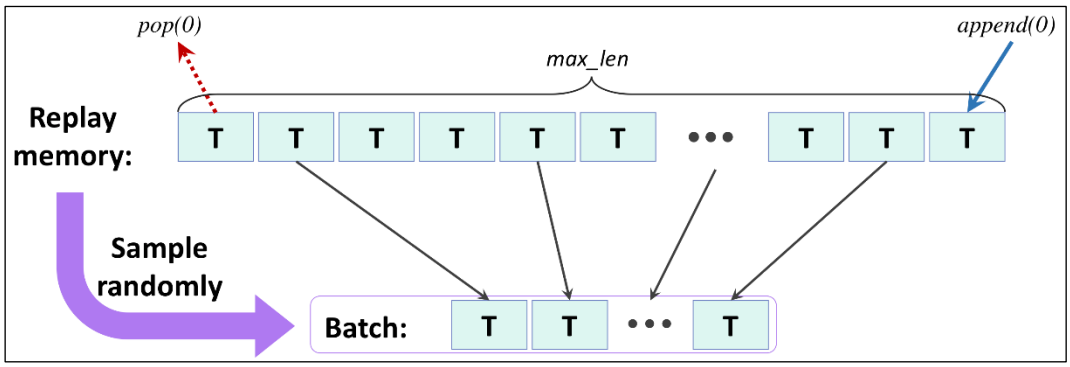

Pole
Cart
Episode 100
Pole's
movement
Cart's
movement
Episode 200

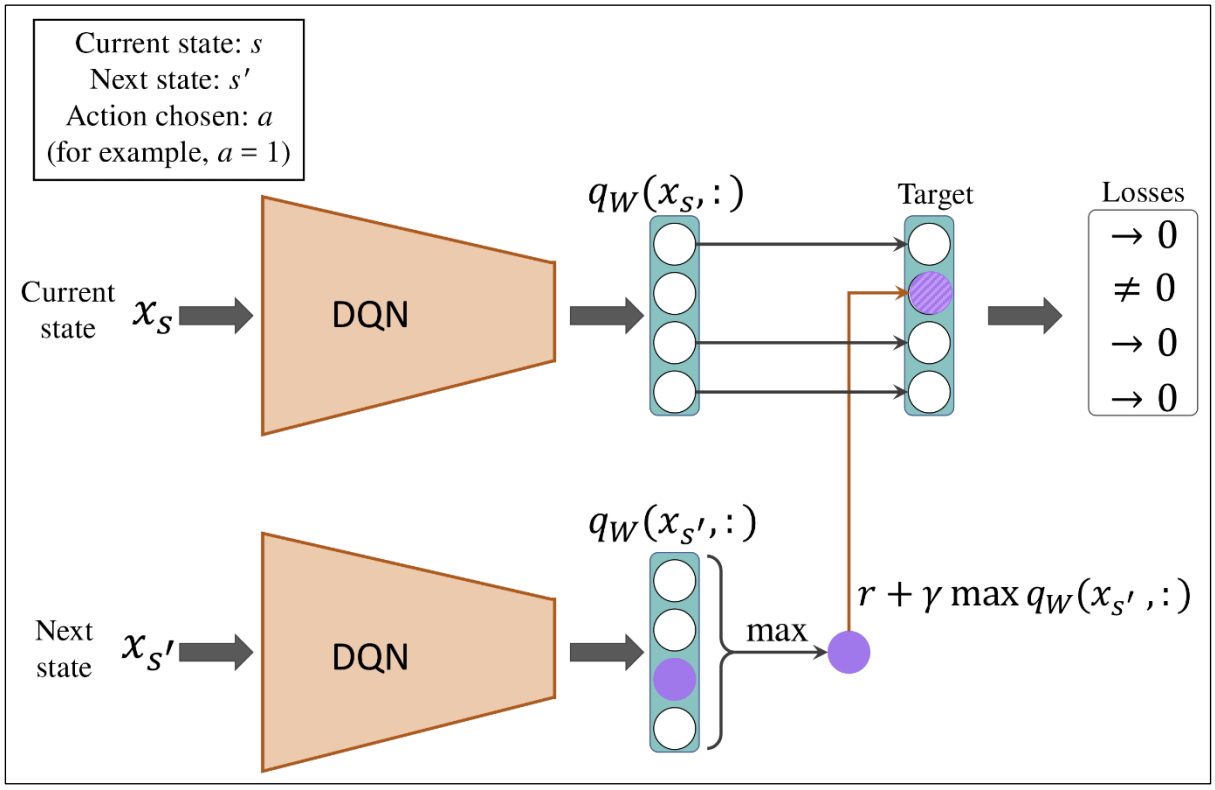

24 25 26 27 28 29
18 19 20 21 22 23
12 13 14 15 16 17
6 7 8 9 10 11
0 1 2 3 4 5
Action space:
{up, down, left, right}
Rewards:
+1 If landed in gold state
−1 If landed in a trap
0 Otherwise
Terminal states:
Gold
Traps
16
10
15
22
Starting state:
0

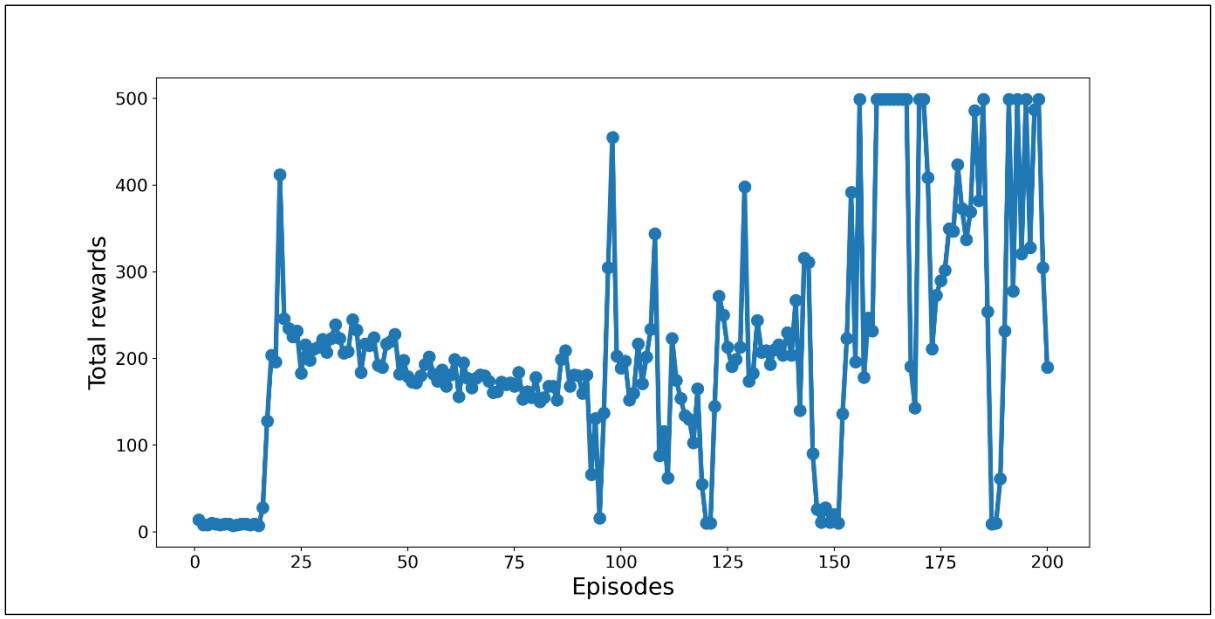

/Users/vahidmirjalili/PycharmProjects/PyML-RL-Qlearning/gridworld_env.py


moves
Episodes
Final rewards
Episodes


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


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


Current state: s
Next state: s'
Action chosen: a
(for example, $a = 1$)
Current state
x_s
DQN
$q_W(x_s, :)$
Target
Losses
$\rightarrow 0$
$\neq 0$
$\rightarrow 0$
$\rightarrow 0$
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