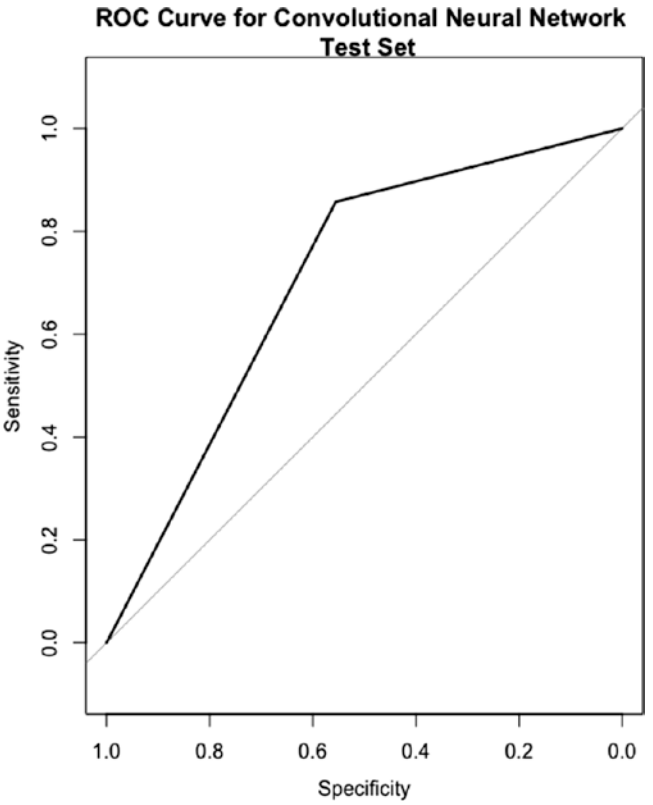

Figure 11-13. *ROC plot for CNN over test data*

The model when predicting against the test data has an AUC of 0.7063. Roughly speaking, the performance here is considerably similar, although as expected we do notice a drop-off in performance from the training to test set. That said, it's unlikely even in this instance that there is any indication of overfitting. However, you might still be inclined to improve the performance of these models should you want to put something like this into production. Ideally, when classifying images, we would like to have models that perform with at least 90% accuracy. Although the image classification case here is rather benign, there are cases in which incorrect classifications can lose considerable amounts of money per observation or cause incorrect diagnoses that therefore cause patients to receive improper care. With this in mind, how would you proceed from this point?

The most logical next step would be to acquire more data for the training phase of this model. This is typically what we would consider the largest challenge of building sufficient convolutional neural networks: getting enough training data. For many different commercial products, acquisition of this data legally can prove an exhaustive task and in a worst-case scenario would require acquisition of the data by the team itself in the real world. Readers should be mindful of this when creating CNNs for specific tasks, because sometimes the feasibility of the task is purely a matter of how accessible the data is. In this instance, the data set we're using is roughly only in total 170 photos, of which we train over 75%.

Another suggestion you might want to take note of is using another network architecture, or if you feel ambitious enough, trying to create your own. However, creating your own network architecture can be an extremely daunting task. Another possible avenue to explore is creating several convolutional neural networks. From these models, we can create a data set where each feature is the output of a given CNN. This data set could then be inputted into a traditional machine learning model. You should be conscious, though, that these approaches might in an of themselves require significant tuning of the approach outlined earlier.

Collaborative Filtering

For our final example, we'll be briefly tackling the problem of recommendation systems as was briefly addressed in prior chapters. Recommendation systems are constantly evolving, but it's useful to address the concept because of the application of data science within them. It's here that you'll be introduced to the practical applications of imputation in addition to some of the soft skills of data science such as data transformation that have been briefly addressed but never walked through.

Recommendation systems are particular to e-commerce websites like Amazon.com but are also present in content-based sites such as Netflix. The motivation is fairly straightforward in that it is reasonable to recommend products to customers that they would reasonably like. The task of doing so is more difficult than it seems, though. Most users don't use the entirety of all products offered by a given company. Even if they did, it doesn't mean they would rate every single product they used. That leaves us with the problem of having a matrix that is sparsely populated with values. Nevertheless, we've reviewed techniques to handle this and will be moving on to inspecting our data set.

For this experiment, we'll be using the third Jester data set (http://goldberg.berkeley.edu/jester-data/). The features all represent individual jokes, and the rows represent users. Each entry within the matrix is a rating for a joke, where the lower bound is –10 and the upper bound is 10. However, whenever there isn't an entry for a joke, this is represented by a 99. When inspecting the head of the data set, we see the matrix shown in Figure 11-14.

	2	3	4	5	6	7	8	9	10
1	99	99.00	99	99	-9.27	99	-9.17	-8.59	99
2	99	99.00	99	99	-6.12	99	-7.48	-7.77	99
3	99	0.05	99	99	-2.82	99	-4.85	-0.87	99
4	99	99.00	99	99	-4.95	99	6.21	2.72	99
5	99	99.00	99	99	3.11	99	4.42	1.41	99
6	99	99.00	99	99	-0.05	99	-8.11	-7.38	99

Figure 11-14. *Snapshot of the Jester data set*

The goal here will be to measure the similarity of different users' tastes based on the similarities between the jokes themselves. To do this, we'll be calculating the cosine similarity between column vectors. Briefly, let's discuss the concept of cosine similarity before speaking about combining matrix factorization and RBMs to impute missing values. When working with problems in which you're trying to compare vectors, cosine similarity is a concept that will often be referenced. Intuitively, we define *cosine similarity* as the degree to which two non-zero vectors are distinct. Mathematically, we define cosine similarity with the following equation

$$\text{similarity} = \cos(\theta) = \frac{A * B}{\|A\|_2 \|B\|_2}$$

where A, B = two distinct vectors.

Similarly to a correlation coefficient, cosine similarity values range from –1 to 1. A cosine similarity of 1 indicates that values are exactly the same, whereas –1 means they are exactly opposite. A value of zero indicates no relationship between vectors at all. With this in mind, we'll compare the consumption patterns of certain music with one another such that we can compare which items are most like each other and therefore should be recommended to other individuals.

However, for those who paid close attention, cosine similarity is used with two *non-zero* vectors—meaning we have to generate values for our dataset where they are missing. There are many techniques that have been discussed for imputation, but one that has been described as useful by Geoffrey Hinton in this instance is *matrix factorization*. Specifically, I suggest you use *singular value decomposition* (SVD).

SVD and PCA, discussed elsewhere in this book, are highly related techniques. They both are perform eigendecompositions of a matrix, but SVDs applications differ from that of PCA. Particularly, SVD can be used to approximate the missing values. As such, let's impute our values using the `impute.svd()` function:

```
require(lsa)
require(bcv)
require(gdata)
require(Matrix)

#Upload the data set
#Please be patient this may take a handful of seconds to load.
data <- read.xls("/path/to/data/.xls", sheet = 1)
colnames(data) <- seq(1, ncol(data), 1)

#Converting 99s to NA Values (1)
data[data == 99] <- NA

#Converting 99s to Mean Column Values (2)
for (i in 1:ncol(data)){
  data[is.na(data[,i]), i] <- mean(data[,i], na.rm = TRUE)
}
```

We begin by converting the 99s (1) to NA values and then changing the NA values to the column means (2). After this point, we can move forward and impute the values:

```
#Imputing Data via SVD
newData <- impute.svd(data, k = qr(data)$rank, tol = 1e-4, maxiter = 200)
print(newData$rss)
head(data[, 2:10])
```

Be aware that the `impute.svd()` function requires that you impute either column means for the missing values, or if an entire column's observations are missing to make it 0. If you don't follow these instructions, you'll receive incorrect results. When executing the preceding code, we yield the outputs shown in Figure 11-15.

```
       2        3     4    5     6        7     8     9   10
1.987538 1.336403 1.506 1.83 -9.27 4.056429 -9.17 -8.59 -1.8
1.987538 1.336403 1.506 1.83 -6.12 4.056429 -7.48 -7.77 -1.8
1.987538 0.050000 1.506 1.83 -2.82 4.056429 -4.85 -0.87 -1.8
1.987538 1.336403 1.506 1.83 -4.95 4.056429  6.21  2.72 -1.8
1.987538 1.336403 1.506 1.83  3.11 4.056429  4.42  1.41 -1.8
1.987538 1.336403 1.506 1.83 -0.05 4.056429 -8.11 -7.38 -1.8
```

Figure 11-15. *Head of imputed data set*

When executing the SVD, we also calculated a sum of squares of 4.398197e-20 with respect to the non-missing values and the predictions of these non-missing values. Readers who feel inclined to challenge themselves here can, instead of using SVD impute the values, use an RBM. Be aware, though, that this task can be extremely computationally expensive, and the modification of the RBM for this task is not easy. Look for high-level overviews given by Geoffrey Hinton on this topic (http://www.machinelearning.org/proceedings/icml2007/papers/407.pdf).

We can now calculate the cosine distances between the columns:

```
itemData <- matrix(NA, nrow = ncol(data), ncol = 11,
                   dimnames=list(colnames(data)))
#Getting Cosine Distances
for (i in 1:nrow(itemData)){
  for (j in 1:ncol(itemData)){
    itemData[i,j] <- cosine(data[,i], data[,j])
  }
}
```

When executing the preceding code, we yield the data set shown in Figure 11-16.

```
       [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
1 1.00000000 0.96586519 0.93061731 0.97344905 0.97438177 0.03342678 0.97393042 -0.1563475
2 0.96586519 1.00000000 0.95495652 0.99103253 0.99129663 0.03424918 0.99084916 -0.1648979
3 0.93061731 0.95495652 1.00000000 0.95527979 0.95531230 0.03379699 0.95469894 -0.1585386
4 0.97344905 0.99103253 0.95527979 1.00000000 0.99928582 0.03380817 0.99849612 -0.1672893
5 0.97438177 0.99129663 0.95531230 0.99928582 1.00000000 0.03365341 0.99952555 -0.1675603
6 0.03342678 0.03424918 0.03379699 0.03380817 0.03365341 1.00000000 0.03385248  0.2222439
```

Figure 11-16. *Head of the cosine distance data set*

From this data set, we can now perform the final data transformation such that each row represents a particular joke and each column represents the jokes most similar in descending order from left to right. We do this initially by instantiating an empty matrix with the proper dimensions (1). After this matrix is instantiated, we can then fill in the data by sorting the cosine values and taking the indices that contain the top 11 values—we take the top 11 because the number 1 value will always be the same item itself:

```
#Creating Matrix for ranking similarities (1)
similarMat <- matrix(NA, nrow = ncol(itemData), ncol = 11)

#Sorting Data Within Item Data Matrix (2)
for(i in 1:ncol(itemData)) {
  rows <- order(itemData[,i], decreasing = TRUE)
  similarMat[i,] <- (t(head(n=11, rownames(data[rows ,][i]))))
}

#Printing Result
similarMat
```

When executing the preceding code, we reach our final answer, shown in Figure 11-17.

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]  [,9]  [,10] [,11]
 [1,] "1"  "5"  "7"  "4"  "26" "31" "11" "2"   "101" "12"  "15"
 [2,] "2"  "5"  "4"  "7"  "26" "11" "31" "101" "12"  "15"  "13"
 [3,] "3"  "5"  "4"  "2"  "7"  "11" "26" "31"  "101" "12"  "15"
 [4,] "4"  "5"  "7"  "26" "31" "2"  "11" "101" "12"  "15"  "13"
 [5,] "5"  "7"  "4"  "26" "31" "11" "2"  "101" "12"  "15"  "13"
 [6,] "6"  "20" "21" "28" "8"  "18" "37" "19"  "36"  "51"  "54"
 [7,] "7"  "5"  "4"  "26" "31" "11" "2"  "101" "12"  "15"  "13"
 [8,] "8"  "21" "19" "16" "20" "18" "9"  "14"  "17"  "6"   "75"
 [9,] "9"  "16" "14" "18" "17" "10" "25" "24"  "59"  "65"  "58"
[10,] "10" "24" "25" "59" "58" "75" "72" "45"  "38"  "34"  "61"
[11,] "11" "5"  "7"  "4"  "26" "2"  "31" "12"  "101" "15"  "13"
```

Figure 11-17. *Top 10 recommendations for 11 separate jokes*

We interpret the result as yielding the top 10 recommendations for 11 separate jokes. You can implement this in a platform such that on a web page, users receive recommendations for different pages, products, and or similar entities.

Summary

We now have reached the end of this chapter and our review of deep learning and machine learning techniques entirely. Chapter 12 provides brief advice that all data scientists should be aware of as they move forward in their research or professional endeavors.

CHAPTER 12

Closing Statements

We have reached the end of this book. By now, you should feel comfortable that you've acquired a general overview of data science, machine learning, and deep learning. If not, you should at least be adequately aware of where you need to focus your efforts in reviewing and further research. The purpose of this book is *not* intended to make anyone an expert. Rather it should be used to highlight the respective power of these techniques in a given field. I would like to end by imparting advice for all readers with my thoughts on the best way to use these models and the general methodology of machine learning.

In every field, there are idiosyncratic characteristics that have long been studied. This is generally what I would describe as the *science of X*, where *X* is a given field we're discussing. Sometimes specific quantitative subfields have been developed within the broader field to tackle these goals. Given the complex nature of the world, it can't be overemphasized that studying the broader field and the specific subfield you're interested holistically is a requirement *before* you seek to implement machine learning methods to problem solving. Among the complaints I have heard from many colleagues and friends is that there is one overwhelming deficiency that many data scientists often have: *domain knowledge*. Machine learning and deep learning algorithms have gotten very good at performing in a variety of contexts and increasingly have been able to produce robust solutions. however, using a good tool poorly in a given context can produce results just as bad, if not worse, as using the wrong tool poorly in a given context.

You should be sure that you deeply understand the algorithms you choose prior to implementing them at scale. There is seldom anything worse than providing a solution, seeing the process for it fail, and being unable to provide counsel on how to fix it. Beyond something bad happening, often you'll be expected to discuss these algorithms with people who have less technical backgrounds. Although I have emphasized this in prior chapters, I must again state the power of good visualizations and succinct explanations. Although you might find intricate detail compelling, the average person doesn't have the time that you have spent educating yourself on this topic. So, only make things as complicated as they need to be.

Finally, I urge you when producing your own solutions to be as creative as possible. The proliferation of machine learning algorithms is exciting for the way in which it revolutionizes our world, but it will also lead to great homogeneity among products if these algorithms aren't used in a unique way. The process of solving problems,

T. Beysolow II, *Introduction to Deep Learning Using R*, DOI 10.1007/978-1-4842-2734-3_12

while frustrating at times, should be challenging and exciting also. This should be an opportunity to use your ingenuity to create a unique solution—not use old and tired solutions. Reusing large portions of code, though tempting and often necessary to save time in the development stage, should be avoided as much as possible as well. Always force yourself to approach a problem from scratch, because that will inspire new and hopefully better solutions.

I wish all readers the best of success moving forward in their respective studies and careers, and also in life. Machine learning is one of the most frustrating concepts I have ever encountered, but through studying it I've learned an incredible amount about computer science and myself while also being introduced to an immense amount of incredibly intelligent individuals. I hope that the joy that has been brought into my life from studying this field is similarly brought to yours Godspeed.

Sincerely,
Taweh Beysolow II

Index

A

A/B testing, 148
  beta-binomial hierarchical model for, 149, 151
  simple two-sample, 149
Activation function, 3
Additive law of probability, 13
Akaike information criterion (AIC), 152
AlexNet, 110
Amazon Web Services (AWS), 167
Analysis of Variance (ANOVA), 137
  MANOVA, 138
  mixed-design, 138
  one-way, 137
  two-way (multiple-way), 137
Ant colony optimization (ACO), 159–160
Arithmetic mean, 15
Asset price prediction, 171–172
  description of experiment, 173–175
  feature selection, 175–176
  supervised learning, 172–173
Associative property, 19
Autoencoders, 125–126, 195–199, 201–202
  linear autoencoders *vs.* PCA, 126–127
Axioms, 19
  associative property, 19
  commutative property, 19
  distributivity of scalar multiplication, 20
  identity element of addition, 19
  identity element of scalar multiplication, 20
  inverse elements of addition, 19

B

Back-propagation algorithm, 95–97, 107
Back-propagation through time (BPPT), 114–115
Backward selection, 151
Batch learning, 131
Bayesian classifier, 191–193, 196
Bayesian learning, 83
  50/25/25 cross-validation, 85–86
  limitation, 84
  Naïve Bayes classifier, 84
  tuning machine learning algorithms, 85
Bayesian statistics, 149
Bayes information criterion (BIC), 152
Bayes' theorem, 14
Beta-binomial hierarchical model, 149, 151
Beta distribution, 150
Bi-infinite sequence, 40
Binary classifier, 66
Binomial distribution, 150, 151
Blocking process, 145
BPTT. *See* Back-propagation through time (BPTT)

C

Canonical correlation analysis (CCA), 156
Central processing unit (CPU), 168, 169
Coefficient of determination (R squared), 17
Collaborative filtering, 214–218
Commutative property, 19
Complex cells, 101
Confusion matrix, 68–69
  for Bayesian classifier, 85
  for classification tree, 80

T. Beysolow II, *Introduction to Deep Learning Using R*, DOI 10.1007/978-1-4842-2734-3

Conjugate distribution, 150
Conjugate gradient algorithms, 98, 119
Continuous random variables, 14
Contrasting divergence (CD) learning, 129–131
Convergent sequence, 41
Convolutional layer
  convolving, 104
  feature maps, 104
  filtering, 104
Convolutional neural networks (CNNs), 5, 108, 202, 204
  AlexNet, 110
  convolutional layer, 103–104
  depth, 108
  FC layer, 106
  GoogLeNet, 109
  history, 101
  loss layer, 107
  pooling layer, 105
  preprocessing, 204–206
  regularization, 111
  ReLU layer, 106
  ResNet, 110
  structure and properties, 101–103
  tuning parameters, 108
  VGGnet, 110
Convolving, 104
Cook's distance, 142
Correlation coefficients, 16
Correlation matrix, 175
Cosine similarity, 215

D

Data science, 219
Decision tree learning, 78
  classification trees, 79–80
  limitations, 81
  regression trees, 80–81
Deep belief network (DBN), 6, 134–135
Deep learning, 219
  autoencoders, 195–199, 201–202
  CNNs, 202, 204
    preprocessing, 204–206
  model building and training, 206–214
    collaborative filtering, 214–218
  models, 3
    applied machine learning and, 7
    CNNs, 5
    DBNs, 6
    experimental design, 7
    feature selection, 7
    history, 8
    MLP, 4
    restricted Boltzmann machines, 6
    RNNs, 5
    SLP, 3–4
    structure of, 2
Deep neural networks, 2
Derivatives and differentiability, 42
Diagonal matrix, 22
Discrete random variables, 14
Discriminant, 43
Distributional prediction, 79
DropConnect, 111
DropOut, 111

E

Eigenvalues, 34–36
Eigenvectors, 34–36
Elman neural networks, 115
Embedded algorithms,157. *See also* Wrappers, Filters, and Embedded (WFE) algorithms
Ensemble methods, 82
  gradient boosting algorithm, 82–83
  random forest, 83
    limitations, 83
Euclidean function
  Euclidean loss, 107
  softmax loss function, 107
  softmax normalization, 107
Euclidean norm, 29
Expectation maximization (EM) algorithm, 76
  expectation step, 77
  maximization step, 77–78

F

Factor analysis, 154–155
  limitations, 155
Factor loadings, 155
Fast learning algorithm, 135
  steps, 136
Feature maps, 104
Feature/variable selection techniques, 151
  backwards and forward selection, 151–152

factor analysis, 154–155
limitations, 155
PCA, 152–154
FeedForward() function, 206
Fisher's principles, 144–145
Fixed tabu search, 163–164
F-statistic and F-distribution, 138–145
Full factorial, 147
Fully connected (FC) layer, 102, 106
Fully recurrent networks, 113–114

G

Genetic algorithms (GAs), 158
Geometric mean, 15
Gibbs sampling, 129, 135
Global minimizers, 47–48
Global optimum, 95
Google Finance API, 172
GoogLeNet, 109
Gradient, 42
Gradient boosting algorithm, 82–83
Gradient descent algorithm, 53–54
Graphics processing unit (GPU), 168

H

Hadamard matrix, 146
Halton, Faure, and Sobol sequences, 148
Hamming distance, 164
Handling categorical data, 155
categorical label problems, 156
CCA, 156
encoding factor levels, 156
Hard drive disk (HDD), 167
Hardware and software suggestions
CPU, 169
GPU, 168
motherboard, 169
optimizing machine learning software, 170
processing data with standard hardware, 167
PSU, 170
RAM, 169
solid state drive and HDD, 167
Having memory, 113, 116
Hessian-free optimization, 48
Hessian matrix, 43, 49
Hidden layers, 99
Hill climbing search methods, 158

I

Identity matrix, 22
ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), 109–110
Inception architecture, 109
Instantaneous algorithm, 90
Intelligent optimization, 161
Interpretation, 145

J

Jacobian matrix, 49

K

Kernels, 72
K-means clustering, 74, 156
limitations, 75–76
K-nearest neighbors (KNN), 165, 189–191

L

Learning rate, 54–55, 209
choosing, 55–56, 58–60
Levenberg-Marquardt heuristic, 61
Newton's method, 60–61
Least Absolute Shrinkage and Selection Operator (LASSO), 63
ridge regression and, 64
LeNet, 108
Levenberg-Marquardt (LM) algorithm, 61
Leverage, 142
Linear algebra, 17
axioms, 19
associative property, 19
commutative property, 19
distributivity of scalar multiplication, 20
identity element of addition, 19
identity element of scalar multiplication, 20
inverse elements of addition, 19
matrices, 20
addition, 21
column vector and square matrix, 24
derivatives and differentiability, 42
distributive over matrix addition, 27

Linear algebra(*cont.*)
eigenvalues and eigenvectors, 34–36
Euclidean norm, 29
Hessian, 43
hyperplanes, 39–40
inner products, 32
L1 norm, 29–30
L2 norm, 29
limits, 41
linear transformations, 36–37
matrix by matrix multiplication, 23
multiplication, 22
multiplication properties, 26
norms, 29, 31
norms on inner product spaces, 32–33
nullspace, 39
orthogonality, 34
orthogonal projections, 38
outer product, 34
partial derivatives and gradients, 42
P-norm, 30–31
proofs, 33–34
properties, 21
quadratic forms, 37
range, 38
rectangular, 26
row and column vector multiplication, 24
row vector, square matrix, and column vector, 25
scalar multiplication, 21, 23, 27
sequences, 40
sequences, properties, 40
square, 25
Sylvester's criterion, 37–38
trace, 28
transpose, 28
transposition, 21
types, 21–22
scalars and vectors, 17
subspaces, 20
vectors, properties, 18
addition, 18
element wise multiplication, 19
subtraction, 18
Linear autoencoders *vs.* PCA, 126–127
Linear regression, 51
gradient descent algorithm, 53–54
learning rate, 54–55
multiple linear regression via gradient descent, 54
OLS, 51–53
Linear transformation, 36–37
Local minimizers, 47
conditions for, 48–49
Local search methods, 157
ACO, 159–160
genetic algorithms (GAs), 158
hill climbing, 158
simulated annealing (SA), 159
VNS, 160–161
Logistic function, 66
Logistic regression, 66–67, 186–189
limitations, 69–70
Long short-term memory (LSTM)
applications, 117
distinguishing factor, 117
forget gate, 117
overview, 116
traditional, 118
training, 118
visualization, 117
Loss layer, 107

M

Machine learning, 1, 50, 219
algorithms, 51
asset price prediction, 171–172
description of experiment, 173–175
feature selection, 175–176
supervised learning, 172–173
feature selection, 185–186
history, 50
model evaluation, 176
ridge regression, 176–178
SVR, 178–180
model training and evaluation, 186
Bayesian classifier, 191–193
KNN, 189–191
logistic regression, 186–189
proliferation, 219
speed dating, 180
classification, 181–182
data cleaning and imputation, 182–185
unsupervised learning, 74
assignment step, 74

K-means clustering, 74
K-means clustering, limitations, 75–76
update step, 75
Markov process, 87
Matrices, 20
addition, 21
column vector and square matrix, 24
derivatives and differentiability, 42
distributive over matrix addition, 27
eigenvalues and eigenvectors, 34–36
Euclidean norm, 29
Hessian, 43
hyperplanes, 39–40
inner products, 32
L1 norm, 29–30
L2 norm, 29
limits, 41
linear transformations, 36–37
matrix by matrix multiplication, 23
multiplication, 22
multiplication properties, 26
norms, 29, 31
norms on inner product spaces, 32–33
nullspace, 39
orthogonality, 34
orthogonal projections, 38
outer product, 34
partial derivatives and gradients, 42
P-norm, 30–31
proofs, 33–34
properties, 21
quadratic forms, 37
range, 38
rectangular, 26
row and column vector multiplication, 24
row vector, square matrix, and column vector, 25
scalar multiplication, 21, 23, 27
sequences, 40
properties, 40
square, 25
Sylvester's criterion, 37–38
trace, 28
transpose, 21, 28
types, 21–22
Mean squared error (MSE), 17, 65
Mixed-design ANOVA, 138
mlp() function, 100
MLP. *See* Multilayer perceptron (MLP) model
Momentum within RBMs, 132
Motherboard, 169
Multicollinearity, 62
confusion matrix, 68–69
logistic regression, 69–70
regression models, 64–67
ridge regression, 62–64
ROC curve, 67–68
SVM, 70–73
testing, 62
VIF, 62
Multilayer perceptron (MLP) model, 4
back-propagation algorithm, 95–97
considerations, 97–99
distinguishing factor from SLPs, 94
global optimum, 95
limitations, 97–99
Multiple linear regression via gradient descent, 54
Multiplicative law of probability, 13
Multivariate ANOVA (MANOVA), 138
Mxnet, 99

N

Naïve Bayes classifier, 84
Neighborhoods, concept, 49
interior and boundary points, 50
Netflix, 214
Neural history compressor, 116
Newton's method, 60–61
Non-parametric bootstrapping, 81
Norms, 29
Null hypothesis, 145

O

One-way ANOVA, 137
Online learning, 131
Optimization, 45
unconstrained, 45–46
global minimizers, 47–48
local minimizers, 47
local minimizers, conditions, 48–49
Ordinary least squares (OLS), 51–53
Orthogonality, 34
Orthogonal projections, 38

P

Parameter tuning, 173
Partial derivative, 42
Perceptron model, training, 90
Plackett-Burman designs, 146
Point prediction, 79
Pooling layer, 105
Positive semi-definite matrix, 22
Posterior distribution, 149
Power supply unit (PSU), 170
Principal components, 152
Principal components analysis (PCA), 36, 126–127, 152–154, 176
Prior distribution, 149
Probability, 11–12
Probability theory, 86
Pseudo-random numbers, 148

Q

Quadratic forms, 37
Quantitative finance, 171

R

Random access memory (RAM), 169
Random forest, 83
  limitations, 83
Randomization, 145
Random sampling, 14
Random variables, 14–15
Reactive search optimization (RSO), 161
  fixed tabu search, 163–164
  KNN, 165
  reactive prohibitions, 162–163
  RTS, 164
  WalkSAT algorithm, 165
Reactive tabu search (RTS), 164
Receiver Operating Characteristic (ROC) curve, 67–68
Rectangular matrices, 26
Rectified linear units (ReLU) layer, 106
Recurrent neural networks (RNNs), 5
  architecture, 114
  BPPT, 114–115
  Elman, 115
  example, 120–124
  fully, 113–114
  LSTM, 116–118
  neural history compressor, 116
  parameter update algorithm, 119–120
  structural damping within, 119
Regression, 172–173
Regression models, 51
  evaluating, 64–65
    classification, 65
    coefficient of determination, 65
    logistic regression, 66–67
    MSE, 65
    SE, 65
  linear regression, 51
    gradient descent algorithm, 53–54
    learning rate, 54–55
    multiple linear regression via gradient descent, 54
    OLS, 51–53
Regularization
  DropConnect, 111
  DropOut, 111
  L1 and L2, 111
  negative effect, 111
  stochastic pooling, 111
Reinforcement learning, 86–87
Relief algorithm, 157
ResNet, 110
Restricted Boltzmann machines (RBMs), 6, 125, 127
  energy function, 127
  Hopfield networks, 128
  implementations, 129
  individual activation probabilities, 129
  momentum within, 132
  probability distributions, 128
  standard, 127
  visualization, 135
Ridge regression, 62, 63, 176–178
  and LASSO, 64
Robust tabu search, 163

S

Simple cells, 101
Simulated annealing (SA), 159
Single layer perceptron (SLP) model, 3–4
  activation function, 90
  architecture, 89
  distinguishing factor from MLP, 95
  limitations, 91–93
  perceptron model, 90
  statistics, 94
  WH algorithm, 90

Singular value decomposition (SVD), 215–216
SLP. *See* Single layer perceptron (SLP) model
Solid state drives, 168
Space filling, 147
Sparsity, 133
Speed dating, 180
  classification, 181–182
  data cleaning and imputation, 182–185
Standard deviation, 16
Standard error (SE), 65
Statement of experiment, 144
Statistical concepts, 11
  and *vs.* or, 12–13
  Bayes' theorem, 14
  coefficient of determination (R squared), 17
  MSE, 17
  probability, 11–12
  random variables, 14–15
  standard deviation, 16
  variance, 15
Statistical replication, 145
Stochastic pooling, 111
Stride, 108
Structural damping, 119
Subspaces, 20
Supervised learning, 50
  regression, 172–173
Support vector machine (SVM), 70–72
  extensions, 73
  kernels, 72
  limitations, 73
  sub-gradient method applied to, 72
Support vector regression (SVR), 178–180
Sylvester's criterion, 37–38

T

Test of significance, 145
Transposition, 18
Two-way infinite sequence, 40
Two-way (multiple-way) ANOVA, 137

U

Unconstrained optimization, 45–46
  global minimizers, 47–48
  local minimizer, conditions, 48–49
  local minimizers, 47
Unsupervised learning, 74
  assignment step, 74
  K-means clustering, 74
    limitations, 75–76
  update step, 75

V

Vanishing gradient, 116
Variable neighborhood search (VNS), 160–161
Variance, 15
Variance inflation factor (VIF), 62
VGGnet, 110

W, X

Wake-sleep algorithm, 8
WalkSAT algorithm, 165
Weight decay, 133
Widrow-Hoff (WH) algorithm, 90
Wrappers, Filters, and Embedded (WFE) algorithms, 157
  relief algorithm, 157

Y

Yahoo! Finance API, 172

Z

Zero-padding, 108