Instructions for
Recompiling the PDP Programs

Complete source code for all the programs described in this book is provided in the archive src.arc. If you have Version 3.0 (or higher) of the Microsoft C compiler, you should be able to modify the programs and recompile them on your PC. If you have a UNIX system (Berkeley 4.2 or higher) with the standard UNIX C compiler, you should also be able to recompile the programs to run on that system. We first provide a general inventory of the components of the PDP software and of the dependencies among these components. Then we describe the procedure for recompiling for the PC with Microsoft C. Following this we briefly describe how to set up the PDP software on UNIX systems. You are free to try to use other C compilers, but with others you are completely on your own. There is every reason to expect that some tinkering will be required to recompile the software with non-UNIX C compilers other than Microsoft C.

Components of the PDP Software

The software includes seven executable programs: aa, bp, cl, cs, ia, iac, and pa. For each of these programs, there is a source file with the same name (e.g., aa.c). In addition, there are certain other source files that several programs share. The object files that all programs share are grouped into one library file called libpc.a. This library file is made up of the compiled versions of the routines from the files command.c, display.c, general.c, io.c, main.c, patterns.c, template.c, and variable.c. The software also includes the two utility programs: plot and colex; each of these is constructed from a single corresponding .c file.
If you change any source file, you should recompile and relink all object and executable files that depend upon that source file. The following dependency list shows which executables depend upon which source files. Where `libpc.a` is shown, all eight source files in the library are included in the dependency.

```
aa:   aa.c, libpc.a
bp:   bp.c, weights.c, libpc.a
cl:   cl.c, libpc.a
cs:   cs.c, weights.c, libpc.a
ia:   ia.c, iaaux.c, iatop.c, libpc.a
iac:  iac.c, weights.c
pa:   pa.c, weights.c, libpc.a
plot: plot.c
colex: colex.c
```

Note also that there are many header files (with names ending in `.h`) in the PDP software package. These files often contain declarations that are used in several different modules. This is particularly true for the header files associated with the modules in `libpc.a`. If one of these files is modified, it is prudent to recompile all modules that include this file. The following list indicates which `.h` files are included in each `.c` file:

```
general.h: display.h
aa.c:     general.h, aa.h, variable.h, patterns.h, command.h
bp.c:     general.h, bp.h, variable.h, weights.h, patterns.h,
           command.h
cl.c:     general.h, variable.h, patterns.h, command.h, cl.h
command.c: general.h, io.h, command.h
cs.c:     general.h, cs.h, variable.h, command.h, patterns.h,
           weights.h
display.c: general.h, io.h, variable.h, template.h, weights.h,
           command.h
general.c: general.h, command.h, variable.h
ia.c:     ia.h, io.h, general.h
iaaux.c:  ia.h
iac.c:    general.h, iac.h, variable.h, command.h, weights.h,
           patterns.h
iatop.c:  general.h, cs.h, variable.h, command.h, ia.h
io.c:     io.h
main.c:   general.h, variable.h, command.h, patterns.h
pa.c:     general.h, pa.h, variable.h, weights.h, patterns.h,
           command.h
patterns.c: general.h, command.h, variable.h, patterns.h
template.c: general.h, command.h, variable.h, display.h, template.h
```
variable.c: general.h, variable.h, command.h, patterns.h, weights.h
weights.c: general.h, command.h, weights.h, variable.h

Note that all files that include general.h also implicitly include display.h.

To Recompile for a PC Using Microsoft C

Three batch files are provided in the src directory to aid in compiling and linking: compile.bat, makelib.bat, and pdplink.bat. These files can be executed as though they were programs.

The compile batch file includes the command to compile a source file (.c) into an object file (.obj). The first step in creating an executable (.exe) file is to recompile all of the source files upon which it depends, including the files in libpc.a. To recompile all the source files, execute the following command in the directory that contains the files:

    compile all

This will produce a .obj file for each .c file in that directory. To compile only one source file, use its name instead of all when giving the compile command. Thus, to compile aa.c you would enter

    compile aa.c

This will produce a file called aa.obj in that directory. If there are errors in your source code, the compiler may abort the command file and display the error messages.

Once all of the necessary object files are created, the libpc.a file can be built. The command file for building the library is executed as follows:

    makelib

This will create a file called libpc.a in the current directory from the eight object files, which should be in the same directory. Whenever you recompile any of the eight programs in libpc.a, you should use the makelib command again to update the library. (The makelib command requires all the object files to be present, so it is best to keep these files around while you are actively involved in modifying the programs. Once you stop making changes, you can delete the .obj files to save space.)

The final stage in compiling is linking. The command file pdplink.bat will link the necessary files to create each of the executables. To link the object files for a particular program, enter pdplink with the program name as argument. Thus,

    pdplink bp
will link the files `bp.obj` and `weights.obj` with the library `libpc.a` to create the executable `bp.exe`. The `pdplink` command will also work with `plot` or `colex`. To create executables for all seven PDP simulation programs, use the following command:

```
pdplink all
```

Note that `pdplink all` does not link `colex` and `plot`; these must be linked individually.

If you wish to recompile and relink all of the programs at once, use the following three commands:

```
compile all
makelib
pdplink all
```

Once you have created new executables, you will want to move these files into the appropriate working directories, to be used with the relevant `.tem` and `.str` files. The MS-DOS copy utility can be used to do this.

**Instructions for Setting Up the PDP Software on UNIX Systems**

For UNIX systems, we suggest that you set up a parent directory system for the PDP software and copy the extracted contents of each of the `.arc` files into a separate subdirectory of the parent, giving the subdirectory the same name as the archive. For example, if your parent directory were called `/usr/yourname/pdp`, you would put the contents of `aa.arc` into a subdirectory of this directory called `/usr/yourname/pdp/aa`. The only files that you will not want to include in this directory system are the `.exe` files, since these will only run on PCs. You would also create a subdirectory called `/usr/yourname/pdp/src` containing the source files and other materials necessary to recompile the package from the `src.arc` file.

Once the directories have been set up, you will want to change directories to the `src` subdirectory. It is an easy matter to recompile all the programs because we have supplied a `makefile`. This file is used by the UNIX `make` program to manage the PDP software. To compile all of the PDP simulation programs, you need only execute the following command:

```
make
```

To compile a single program, simply give `make` the name of that program as an argument. For example, to recompile the `aa` program, enter

```
make aa
```
This form also works with the plot and colex programs; they are not updated if make is executed with no arguments.

In either case, make will check the makefile to see which source files need to be recompiled and will recompile them. It will update libpc.a if necessary. And it will link the necessary object modules together to create the necessary executable files. The supplied makefile places the seven PDP executables in directories that are on the same level as the source directory and have the same name as the executable. For example, if the src directory is /usr/yourname/pdp/src then the aa executable would be placed in the directory /usr/yourname/pdp/aa. If you have set up subdirectories for each program as suggested above, this will all work fine. If you have chosen to organize the directories differently, the makefile can be modified to change where each program is placed. For each program there is a variable that specifies the destination directory for the executable version of the program. The names of these variables are uppercase and consist of the program name followed by DEST. Thus for aa, there is a line in the makefile that looks like this:

\[
\text{AADEST = ../aa/}
\]

The path name to the right of the equal sign can be replaced by any other valid UNIX path name. Once it is, aa will be stored in the directory specified by the path. Thus

\[
\text{AADEST = /usr/foo/pc/bin/}
\]

would cause make to put the aa executable in the directory /usr/foo/pc/bin.
References


and some tests and extensions of the model. *Psychological Review, 89*, 60-94.
Index

aa program, 169-174
  commands in, 171-172
  core routines in, 169-170
  implementation of, 169-170
  modes of, 170
  specification of architecture for, 170
  use of, 170-174
  variables in, 172-174
Adams, M. J. 205, 331
Anderson, J. A., 4, 83, 84, 162, 165, 168, 331, 332
Annealing, simulated, 71-72
  exercises on, 74-75
Answers to questions in exercises, 289-320
Appendices, overview of, 4
Asynchronous update, 52
Attractor states as minima, 70
Auto-associator
  linear
    assumptions of, 167
    delta rule in, 179-181
    exercises on, 174-178
    explosive growth of activations in, 167, 177-178
    learning orthogonal patterns in, 175-178
    linear predictability constraint in, 180-181
    multilayer, 166
  one layer
    background on, 161-165
    essential properties of, 163
    example of, 162
    exercises on, 174-188
    implementation of, 169-174
    learning regimes for, 165-166
    limitations of, 165-166
    pattern completion in, 164
    pattern rectification in, 164
    psychological applications of, 181-188
    recurrent processing in, 165
    variants of, 167-169, 174-188
Back propagation
  in cascaded networks, 153-155
  exercises with, 145-152
  extensions of, 152-159
  gradient descent and local minima in, 132-135
  implementation of, 137-141
  learning by pattern and by epoch in, 136-137
  the learning rule described, 130-131
  minimizing mean squared error, 126-130
  momentum in, 135-136
  precursors and their limitations, 121-126
Back propagation (continued)
in recurrent networks, 155-156
role of the activation function in, 131-132
in sequential networks, 156-159
symmetry breaking in, 136
Bagley, W. C., 203, 331
Baron, J., 204, 331
Best match problem, 50
Blake, A., 50, 331
Blocking in IAC networks, 16-17
Bobrow, D. G., 206, 333
Boltzmann machine, 73-75
exercises on, 74-75
implementation of, 73-74
simulated annealing in, 74-75
suggested experiments with, 75
use of to avoid local minima, 71-75

bp program
commands in, 142
core routines in, 138-140
implementation of, 137-141
modes and measures in, 140
use of, 141-145
variables in, 142-145
Brain-state-in-the-box model
assumptions of, 167-168
completion and rectification in, 178-179
prototype learning in, 179
self-connections in, 179
Broadbent, D. E., 235, 331

C, the programming language, 9. See also Pseudo-C code, conventions used in
Carr, T. H., 238, 239, 320, 331
Cascaded feedforward networks, back propagation in, 153-154
asymptotic activation in, relation to standard back propagation, 153
exercise on, 154-155
Cattell, J. M., 203, 321
Central tendency learning in pattern associators, 113-114. See also prototype learning
change weights routine
in bp, 140
in cl, 195-196
in pa, 102-103
c program
commands in, 196
core routines in, 195-196
exercises with, 197-201
implementation of, 194-196
use of, 196-197
variables in, 196-197
Clamped activations of units, 65, 78
Clustering in competitive learning, 197-200
effects of the number of clusters, 199-200
colex program, use of, 284-285
Commands, general information on, 25-28
abbreviations of, 26
entering, 25-26
executing lists of, 27
passing out of programs, 28
recursive command level, 28
syntax for, 26
Commands, summary of, 245-262
disp commands, 248-249
exam commands. See set commands
get commands, 249-251
save commands, 251
set commands, 251-262
top-level, 245-248
Competition in IAC networks, 14-15
Competitive learning
architecture of, 189-190
background on, 188-194
exercises on, 197-201
features of, 193-194
geometric analogy, 191-193
graph partitioning with, 200-201
implementation of, 194-197
pattern classification and clustering in, 193, 197-200
variants of, 189
version implemented in c program, 189-191
compute_error routine
in bp, 138, 139
in pa, 102
compute_output routine
in bp, 138
in cl, 195
in pa, 101-102
compute_weight routine in bp, 139
INDEX  337

Computer programs, user interface to, 9-10
goals of, 9-10
constrain_neg_pos routine in bp, 140
Constraint satisfaction, 49-81. See also constraint satisfaction models;
goodness
background on, 49-53
definition of, 50
energy as measure of, 70
exercises on, 58-68, 74-75, 78-81
goodness as measure of, 50-52
maxima in, 61-63, 68-73
models of, 53-54, 70, 72, 73-81
net input in, 52
physics analogy to, 68-73
simulated annealing and, 71
Constraint satisfaction models, 53-54, 70, 72, 73-75, 75-81
Boltzmann machine, 73-75
harmony theory, 75-81
Hopfield nets, 70
implementation of, 54-55, 73-74, 78
relations among, 72
schema model, 53-54
Constraints
hard, 50
weak, 50
Context effects in perception. See also contextual enhancement effect;
word superiority effect
evidence of, 203-204
simulation of, 228-230
Contextual enhancement effect,
simulation of, 237-238
Cooling schedule. See simulated annealing
Core routines
of aa, 169-170
of bp, 138-141
of cl, 195-196
of cs, 54-55, 73, 74, 78
of ia, 218-222
of iac, 21-24
of pa, 100-103
cs program
commands in, 56
core routines of, 54-55
implementation of, 54-55
use of, 55-56
variables in, 57-58
Cube example, exercises with, 58-63
cycle routine
in cs, 54
in ia, 218-219
in iac, 21
Davidson, B. J., 238, 331
Default assignment in IAC networks, 45
Delta
definition of for LMS, 128
implementation of recursive computation of, 138
recursive definition of for back propagation, 130-131
Delta rule
generalized. See back propagation
technique, 86-89, 93-96. See also perceptron, LMS
core, 88, 95
linear independence in, 95-96
linear predictability constraint in, 89, 95-96
mathematical formulation, 87
other names for, 87
in pattern associators, 93-96
performance measure for, 88
simple application of, 87-88
transfer effects in, 95
in one-layer auto-associators, 165-166, 179-181
Dipole problem for competitive learning, 201
Disclaimers, 2, 10. See also the PDP Software Package License Agreement
regarding possible bugs, 10
regarding recompilability, 2
regarding use on non-IBM computers, 2
Diskettes, organization of, 241-242
Display package, 20, 29, 41-42
Distributed memory and amnesia model. See DMA model
DMA model
aspects of learning in, 182
assumptions of, 168-169
coexistence of prototype and repeated exemplars, 186-188
DMA model (continued)
learning a prototype from exemplars, 182-184
learning several categories without labels, 184-186
memory for general and specific information in, 181-188

Eigenvalues. See eigenvectors
Eigenvectors, in auto-associators, 163, 175-178
Electricity problem solving, exercises on, 78-81
Energy, as measure of constraint satisfaction, 70
Epoch, training, 88
Epsilon (ε), learning rate parameter, 84
Equilibria in IAC networks, 13-14
Error messages, 27
during execution of a list of commands, 27
Error surface, 127-130, 133-135
bowl-shaped, 128-129
saddle-shaped, 134-135
Exclusive or function. See XOR
Execution of a list of commands, 27
processing of errors during, 27
Expectation effects in perception, simulation of, 238-239

Feldman, J. A., 58, 331
Files
log (.log) 28, 283-284
look (.loo), 47, 276-277, 278-279
network (.net), 24, 25, 40, 263-269
start-up (.str), 24, 25, 40
template (.tem), 24, 25, 40, 271-278
weight (.wts), 55, 269-271
Forced-choice test of contextual influences in perception, 204
assumptions for, in IA model, 213-214
results of, 204
simulation of basic results of, 231-233

Formats for files used by PDP programs, 263-281
log files, 283-284
look files, 278-279
network files, 263-269

pattern files, 280-281
template files, 271-278
weights files, 269-271
Francis, W., 209, 332
Fukushima, K., 189, 331

Geman, D., 71, 332
Geman, S., 71, 332
Generalization
in IAC networks, 45-46
in pattern associator models, 108-112
getinput routine in iac, 21
getnet routine in iac
Grossberg version, 23
standard version, 22
Gibbs sampler, 71
Goodness, 50-52. See also maxima
definition of, 51
in harmony theory, 78
relation to energy, 70
relation of to net input in symmetric nets, 52
Gradient descent
and back propagation rule, 130-131
correlation of successive steps in, gcor
measure of, 141
example of in one-layer net, 128
and local minima in back propagation, 132-133
and momentum, 135-136
relation to size of weight changes, 130
in weight space, 127-130
Graph partitioning in competitive learning, 200-201
Graphs, how to make, 29, 283-288
Gregory, M., 235, 331
Grossberg, S., 3, 4, 11, 12, 15, 17, 18, 22, 23, 38, 46, 189, 194, 256, 292, 332
Grossberg’s version of IAC networks, 17, 46-47

Handbook, introduction to, 1-11
hardware requirements and recommendations for use of, 2-3
mathematical conventions in, 6
overview of, 3
pseudo-C code in, 7-9
purpose of, 1
as raw material for explorations, 10
software provided with, 2
use with PDP volumes, 1
Harmony, definition of, 77
Harmony theory, 75-81
application to electricity problem
  solving, exercises on, 78-81
feature units in, 75
goodness measure for, 78
implementation of, 78
knowledge atoms in, 75
parameters in, 76, 77
sequential problem solving and, 81
symmetry in, 76
Hawkins, H. L., 238, 331
Hebb, D. O., 83, 84, 332
Hebb rule, 84-86, 90-93
correlational character of, 85, 86
Hebb's statement of, 84
limitations of, 86, 93
mathematical formulation, 84
in one layer auto-associator, 165
in pattern associators, 90-93
simple application of, 85-86
Hidden units
definition of, 126
essential role of, 125-126
Hill-climbing, 53
Hinton, G. E., 50, 66, 68, 70, 71, 73,
  83, 332
Hoff, M. E., 83, 87, 121, 126, 334
Hopfield, J. J., 52, 70, 71, 72, 73, 332
Hopfield networks, 70
Hysteresis in IAC networks, 16-17
IAC model
approach to psychological modeling
  in, 207-208
architecture of, 208-210
background on, 203-208
basic assumptions of, 205-206
concept of trial in, 211-212
connections in, 210-211
display conditions in, 217
exercises for, 227-239
forced-choice test in, 213-214
implementation of, 218-222
input assumptions, 212
parameters of, 214-217
processing assumptions, 212
questions for, 206-208
readout from, 213
use of, 222-227
Ia program
commands in, 223-226
core routines in, 218-222
data structures in, 218
example screen display for, 229-230
exercises with, 227-239
implementation of, 218-227
processing in, 218-222
screen displays in, 222-223
trial and forced-choice specifications
  for, 222
use of, 222-227
variables in, 226-227
IAC model
exercises for, 38-47
implementation of, 19-38
overview of, 18
parameters in, 19
IAC networks
activation function for, 13
architecture of, 12, 18
construction of, 47
definition of, 12, 18
dynamics of, 12-13, 18
exercises on, 38-47
Grossberg’s version of, 17, 46-47
net input in, 12
output function for, 12
parameters of, 13, 19
properties of, 13-17
Iac program
command descriptions for, 30-33
components of, 20-21
core routines in, 21-24
example of use of, 40-42
use of, 24-30
variable list for, 35-38
interact routine in Ia., 219-221
Interactive activation and competition,
  11-47. See also IAC model; IA
model
background on, 11-18
exercises on, 38-47
IAC model of, 18-19
IAC program, 20-38
Interactive activation model. See IA
model
Interrupt prompt, 27
Interrupting processing, 27

James, W., 83, 332
Jets and Sharks example
   exercises on with IAC nets, 38-46
   exercises on in schema model, 67-68
   exercises on in competitive learning, 197-200
Johnston, J. C., 204, 205, 206, 207,
   217, 233, 234, 235, 318, 332, 333
Jones, R. S., 162, 331
Jordan, M. I., 156, 157, 158, 332

Kappa (κ), parameter in harmony
   theory, 77
Kernighan, B. W., 9, 321, 332
Klitzke, D., 204, 333
Kohonen, T., 4, 83, 95, 108, 162, 332
Kucera, H., 209, 332

Learning in PDP models. See also
   delta rule, Hebb rule, competitive
   learning, back propagation
delta rule, one-layer, 86-89, 93-96
Hebb rule, 84-86, 90-93
introduction to, 83-84
in pattern associators, 90-96
exercises on, 108-119
Least mean square associator. See LMS
Levin, J. A., 11, 332
LMS, 121, 126-130. See also delta
   rule, one layer
gradient descent and, 127
Local maxima, problem of, 68-73
attractor states as, 70
example of with necker cube, 69
physics analogy to, 70-73
probabilistic activation and, 71-72
simulated annealing and, 71-72
stochastic networks for avoiding,
   71-72
log (.loo) files, format of, 283-284
Logistic activation rule, use of in back
   propagation, 131-132
Logistic function, definition of, 71
   graph of, 72
   logistic routine, 74
look (.loo) files, format of, 278-279
   example of, 278-279
   for a matrix variable, example of, 279
Luce, R. D., 213, 315, 333

Making graphs, 29, 283-288
Manelis, L., 204, 333
Massaro, D. W., 204, 333
Mathematical notation, conventions of,
   6-7
counting, 7
matrices, 6
scalars, 6
vectors, 6
Maxima, global and local, 53, 61-63,
   68-73
Maxima, local, methods for avoiding, 71. See also local maxima
McClelland, J. L., vii, 1, 3, 4, 11, 37,
   39, 41, 153, 162, 183, 185, 186,
   187, 203, 204, 205, 206, 207, 208,
   209, 217, 227, 228, 231, 233, 234,
   236, 237, 290, 317, 318, 319, 320,
   332, 333, 334
Memory for general and specific
   information
   in auto-associator models, 181-188
   in IAC networks, Jets and Sharks
   example of, 38-44
Minima, local
   in weight space, example of, 132-133
Minsky, M., 50, 89, 122, 123, 125,
   126, 333
Models, relation to programs, 5
Morton, J., 205, 333

Net input in constraint satisfaction
   models, 52
Network configuration package, 20
network (.net) files
   construction of, 47
   example of, 269
   format of, 263-269
   inclusion in .str file, 40
   overview, 263
   sections of, 264-269
   use of to specify architecture, 24
Nonwords, unpronounceable,
   facilitation of perception of, 236-237
Normalized dot product, 91
Norman, D. A., 206, 333
OR function, gradient descent learning, 128
Orthogonal patterns
definition of, 92
as eigenvectors in linear auto-associators, 163-165
elements of, 92
learning of, 112-113
Ortony, A., 63, 334
Ozog, G., 17, 334
pa program
code descriptions for, 104-105
core routines in, 100-103
error criterion in, 100
overview of, 100
overview of commands in, 103
overview of variables in, 103-104
training commands for, 100
variable list for, 105-108
Papert, S., 50, 89, 122, 123, 125, 126, 333
Parameter changes in IAC networks, 46
Past-tense learning, pattern associator model of, 115-118, 119
Pattern associator models, 97-103
activation functions in, 97-98
environment for, 98
family of, 97-98
implementation of, 100-103
learning rules for, 98-99
performance measures for, 99
training epochs in, 98
Pattern associators. See also pattern associator models; learning in PDP models; Hebb rule; delta rule architecture of, 89
delta rule in, 93-96, 112-114, 114-118
exercises on, 108-119
general properties of, 83-84
Hebb rule in, 90-93, 108-112
illustration of, 89-90
introduction of, 83-84
learning in, 90-95
learning sets of patterns in, 92-93, 94-96
nonlinear 96, 114-119
output in relation to learned patterns, 91-93
Pattern completion in auto-associators, 164, 178-179
Pattern (.pat) files, format of, 280-281
Pattern rectification in auto-associators, 164, 178-179
Pattern similarity, 91
dot product as measure of, 91
Pattern sum of squares, (pss) definition of, 100
in back propagation, 140
Patterns
learning sets of, 112-113
linear independence of, 95-96
orthogonal set of, 92
uncorrelated vs. anticorrelated, 92
Patterns package, 20
PDP:1, 4, 11, 39, 40
PDP:2, 53, 72, 130
PDP:5, 4, 166, 188, 191, 201
PDP:6, 4, 49, 68, 70, 75, 76, 78, 79
PDP:7, 3, 49, 66, 68, 70, 73
PDP:8, 4, 123, 130, 136, 145, 146, 152, 155
PDP:9, 4, 90, 99
PDP:11, 4, 83, 90, 95, 108, 114
PDP:14, 3, 49, 53, 58, 59, 60, 63, 64, 65, 66, 68, 294
PDP:16, 4
PDP:17, 4, 89, 162, 165, 166, 168, 169, 174, 181, 182, 184, 186, 188, 311
PDP:18, 4, 83, 90, 115, 116
PDP:19, 5, 83
PDP:21, 208
PDP:25, 4, 114, 162, 168, 169, 181, 311
PDP Research Group, vii, 1, 333, 334
command and variable summary for, 245-262
command descriptions for, 30-33
command interpreter, 20, 25-27, 28-29
displays in, 20, 29, 41-42
error messages, 27
example of use of, 40-44
formats for files used with, 263-281
hardware requirements for, 2
interruption of processing, 27
PDP software package (continued)
need for math co-processor, 2
network configuration package, 20
overview of, 20-21, 321-324
quitting programs, 29
recompilation of, 325-329
running commands outside the
programs, 28
setting up on PC, 241-244
setting up on UNIX systems, 328-329
single stepping, 27
starting up, 24-25
use of, 24-30, 40-42
variable types in, 33-35
what is provided, 2
Perceptron, 121-126. See also delta rule
definition of, 121-123
limitations of, 123-125
linear separability and, 123-126
Perceptrons, 123
Physics analogy to constraint
satisfaction systems, 70-73
Pillsbury, W. B., 203, 333
Pinker, S., 118, 333
plot program, use of, 285-288
Prince, A., 118, 333
probability routine, 74
Programs, relation to models, 5. See also PDP software package; aa, bp,
cI, es, ia, iac, pa programs
Prototype learning in auto-assessories,
179, 182-188
Pseudo-C code, conventions used in,
7-9
array indexes, 8-9
comments, 7
curly braces, 8
if statements, 7
incrementing, 8, 9
loop constructs, 8
semicolons, 8
Pseudowords, perception of, 232-233,
233-236, 236-239
Psychological modeling
approach to in IA model, 207-208
role of simplifying assumptions in,
207-208
Questions in exercises, answers to,
289-320
Recurrent networks, back propagation
in, 155-156
shift register as example of, 155-156
Reicher, G. M., 204, 206, 207, 212,
223, 333
reset routine in iac, 21
Resetting, commands for, 55-56
Resonance in IAC networks, 15-16
Retrieval and generalization in IAC
networks, exercises on, 38-46
retrieval, graceful degradation in, 45
retrieval by name, 40-44
retrieval from partial description, 44-
45
Riley, M. S., 78, 333
Ritchie, D. M., 9, 321, 332
Ritz, S. A., 162, 331
rnd routine, 74
Rosenblatt, F., 83, 87, 97, 121, 333
Routines. See core routines; PDP
software package
Rule learning in pattern associators,
114-118
Rule of 78, 115-118
Rules and exceptions, handling of in
pattern associators, 118
Rumelhart, D. E., vii, 1, 4, 11, 37, 63,
162, 183, 185, 186, 187, 189, 190,
192, 203, 206, 207, 208, 209, 210,
218, 224, 227, 228, 231, 234, 236,
237, 290, 317, 319, 320, 333, 334
reply routine in es
Boltzmann version, 73
harmony version, 78
schema model version, 54-55

Schema model, 53-73
cube example, exercises with, 58-63
exercises for, overview of, 58
implementation of, 54-55
Jets and Sharks example, exercise
with, 67-68
local minima in, 61-63
purpose of, 53
INDEX 343

room example, exercises with, 63-67
sequential processes in, 68
tic-tac-toe example for, 68
update rule for, 53
Schemata, 63-65
completion and, 64
canonical view of, 63
maxima and, 64
prototypes and, 64
room example of, 64-67
subunits in, 66-67
view of in PDP, 64
Sejnowski, T. J., 66, 68, 70, 71, 73, 208, 332
Selfridge, O. G., 206, 334
Sequence generation, plan-dependent,
in sequential back propagation
networks, exercise on, 158-159
Sequential networks, 156-159
exercise on, 158
implementation of in bp
program, 157
Sequential processing, 68, 81
setinput routine in bp, 219
Setting up a PDP program, discussion
of, 241-244
script for, 243-244
Shift register, learning of in recurrent
back propagation networks, 155-156
Sigma (σ), parameter in harmony
theory, 76
Silverstein, J. W., 162, 331
Simulated annealing, 71-72
Single stepping, 27
Siple, P., 203, 209, 210, 218, 224, 334
Smith, E., 204, 334
Smolensky, P., 68, 70, 71, 75, 78, 81,
256, 296, 333, 334
Spoehr, K., 204, 334
Starting up PDP programs, 24-25
Start-up (.str) files, use of at run time,
24, 25, 40
Stochastic, definition of, 71
Subroutines. See core routines
sum_linked_weds routine in bp, 140
Synchronous update, 52
Szoc, R., 17, 334
Temperature, 71-72
Template (.tem) files, format of,
271-278
layout section of, 271-272
template specifications for, 272-275,
277-278
template types, 275-278
use of at run time, 24, 25, 40
Thurston, I., 204, 331
Total sum of squares, (ss) definition
of, 88, 100
in back propagation, 140
trial routine in pa, 101
train routine in pa, 100-101
Turvey, M., 206, 334
update routine
in 1a, 219, 221-222
in 1ac
Grossberg version, 24
standard version, 23
Update
asynchronous, 52
synchronous, 52
Utility programs plot and colex, use of,
283-288
Variables, types of, 33-35
accessing, 35
configuration variables, 34
environment variables, 34
mode variables, 34
parameter variables, 34
state variables, 34
top-level variables, 35
VIR problem, exercise on, 78-81
von der Malsberg, C., 4, 189, 334
Weight error derivative, in back
propagation, 131
Weights, constraints on, in back
propagation, 137
weights (.wts) files, format of, 269-271
example of, 270-271
Weisstein, N., 17, 334
Wheeler, D. D., 204, 206, 334
Whittlesea, B. W. A., 311, 334
Widrow, G., 83, 87, 121, 126, 334
Word superiority effect
  bigram frequency effects in, 233-235
  effects of contextual constraint in,
    235-236
  extension to pseudowords, 232-233
  introduction of, 203-204
  simulation of the basic effect,
    231-233
  simulation of subtler aspects of,
    233-236
Working directories, how to set up,
  242-244

XOR
  in a cascaded feedforward network,
    exercise on, 154-155
  as an illustration of problems with
    perceptrons, 123-126
  linear separability and, 125-126
  solution of by error propagation,
    exercises on, 145-152
  solution involving hidden units,
    125-126
Zipser, D., 189, 190, 192, 334