

This excerpt from

Neural Smithing.
Russell D. Reed and Robert J. Marks II.
© 1999 The MIT Press.

is provided in screen-viewable form for personal use only by members of MIT CogNet.

Unauthorized use or dissemination of this information is expressly forbidden.

If you have any questions about this material, please contact
cognetadmin@cognet.mit.edu.

Index

- Active layers, counting of, 31
- Adaline networks, 29–30
- Adaptive learning rate methods, 135–145, 147–148
 - benefits of, 72, 85, 95, 135–136, 151–152
 - limitations of, 152
- Adaptive linear neurons. *See* Adaline networks
- Adding units, in constructive methods, 198–199
- Akaike's final prediction error (FPE), 260–261
- Akaike's information criterion (AIC), 260–261
- Algorithms. *See also names of individual algorithms*
 - adaptive (*see* Adaptive learning rate methods)
 - back-propagation, 57–70
 - bias of, 249, 267
 - classic (*see* Classical optimization methods)
 - constructive (*see* Constructive methods)
 - factors in selection of, 156–158
 - generalization and, 157, 249–253
 - genetic, 178–179, 185–195
 - k-means, 107
 - LMS, 29, 90, 123, 124, 298
 - parameters for, 57, 62, 63, 71, 157, 185
 - perceptron learning, 23–27, 202, 205, 210–212
 - pocket, 28, 205
 - pruning (*see* Pruning methods)
 - robustness of, 157
- Ancillary units, 208–209
- AND function, 39, 109, 110
- Approximation error, 37
- Architecture selection, 197
- Artificial neural networks
 - benefits of, 4–5
 - components of, 1
 - definition of, 1
- ART networks, 198
- Assumptions
 - error surface and, 120
 - optimization methods and, 179–180
- Autoassociative networks, 12
- Autoencoder networks, 12, 303–305
- Axons, 1
- Back-propagation
 - benefits of, 4
 - as derivative calculation, 49, 53–57, 66
 - dual meaning of, 49, 66
 - fuzzy control of, 148–149
 - pseudocode examples for, 63–66
 - as training algorithm (*see* Back-propagation algorithm)
- Back-propagation algorithm
 - batch mode for, 57–58, 65, 68
 - benefits of, 49, 180–182
 - classical alternatives to, 158–183
 - as gradient descent, 57, 163
 - modifications for, 62–63
 - as one of many methods, 67
 - on-line mode for, 59–62, 65–66, 68
 - popularity of, 49, 155
 - purpose of, 49
 - training time for, 67–70
 - variations of, 135–153
- Back-propagation network, 66
- Batch learning, 57–58, 155
 - algorithm variations and, 147, 153
 - generalization and, 251
 - pseudocode example for, 65
 - random initialization and, 105
 - training time and, 68, 77–79
- Bayesian methods, 258–260
- Best-step steepest descent, 165–166. *See also* Cauchy's method
- BFGS (Broyden-Fletcher-Goldfarb-Shanno) method, 174–175
- Bias
 - generalization and, 240, 249, 256, 258, 267
 - in hyperplane geometry, 17–18
 - initialization and, 110–111
- Bias weights, 125–126, 231
- Bold driver method, 136–137
- Boolean functions, 19, 39–41, 109
- Bootstrapping, 258, 273
- Bottlenecks, and pruning, 232–234
- Brent's method, 159
- Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, 174–175
- Capacity, 41–47. *See also* Representational capabilities
- Cascade-correlation algorithm, 201–204, 268
- Cauchy's method, 150, 165–166
- Cell bodies, 1
- CGD (conjugate gradient descent), 166–169, 179, 182, 252
- Chain rule, 53, 66–67
- Classical optimization methods. *See also* Genetic algorithm; Optimization methods
 - best-step steepest descent, 165–166
 - conjugate gradient descent, 166–169, 179, 182, 252
 - deterministic, 158, 159–175
 - Gauss-Newton, 130, 172
 - gradient-based, 158, 163–175, 182
 - Hooke-Jeeves pattern search, 160–161
 - Levenberg-Marquardt, 128, 144, 172–173, 182
 - Nelder-Mead simplex search, 161
 - Newton's, 128, 145, 152, 170–172, 179, 294–295
 - Powell's conjugate direction, 161–163
 - quasi-Newton, 174–175
 - simulated annealing, 157, 176–178

- Classical optimization methods (*cont.*)
 - stochastic, 158, 175–179
- Classification
 - initialization and, 107–109, 110
 - regression versus, 156–157
- Clipped linear function, 317
- Cluster centers, 107
- Clustering, 158
- Competition, and pruning, 232–233
- Complexity, 241–242, 244–247
- Computation speed, 4–5
- Conjugate gradient descent (CGD), 166–169, 179, 182, 252
- Conjugate methods, 144, 152, 161–163
- Connected, fully, 3
- Constraints, 240–241, 244–245
- Constructive methods, 197–199, 217
 - cascade-correlation, 201–204, 268
 - dynamic node creation, 199–201
 - generalization and, 268
 - Marchand's, 209–212
 - meiosis networks and, 212–213
 - node splitting, 212–215
 - pruning and, 198, 201
 - tiling, 206–209
 - upstart, 204–206
 - Voronoi tessellation, 215–216
- Convergence time. *See* Training time
- Convex regions, 33
- Convolution, 277–281, 311–314
- Cost function, 7, 155–156. *See also* Error function; Objective function
- Cross-entropy error function, 9, 50, 110, 155, 276
- Crossover, 186, 187, 188
- Cross-validation, 257–258, 273, 291
- Data, and generalization, 242–248
- Davidon-Fletcher-Power (DFP) method, 174–175
- Decay. *See* Weight decay
- Decision tree classifier, 108–109, 152, 217
- Defining length of schemata, 188
- Degrees of freedom, 245–246
- Delta attenuation, 72, 84–85
- Delta-bar-delta algorithm, 137–139, 140–141, 152
 - justification for, 139–140
- Delta rule, 29, 86. *See also* Widrow-Hoff learning rule
- Delta values, 55–57
- Dendrites, 1
- Density, 243
- Dependency, 50–51
- Depth versus size, 38–41
- Derivative calculation, 49, 53–57, 66
- Deterministic methods, 158, 159–175
- DFP (Davidon-Fletcher-Power) method, 174–175
- Diagonal approximation, 131
- Dichotomies, 20–22, 42–44
- Differentiability, 156
- Direction set method, 161–163
- Discriminant analysis, 107–108, 306–310
- Divide and conquer, 204
- Dynamic node creation, 199–201
- Early stopping, 265–266
- Effective learning rate, 77, 87, 88, 90
- Effective target, 278–281, 289
- Eigenvalues, 127–128. *See also* Hessian matrix
 - initialization and, 106
 - learning rate and, 81–84, 85
 - node splitting and, 215
 - search then converge method and, 147
- Energy
 - pruning and, 227–228
 - simulated annealing and, 177
- Entropic error function, 123. *See also* Cross-entropy error function
- Entropy, 271–272
- Epochs, 58
- Error function, 7, 37. *See also* $E(t)$ curves
 - constructive methods and, 198–201, 205–206
 - cross-entropy, 9, 50, 110, 155, 276
 - entropic, 123
 - initialization and, 110
 - learning rate and, 71–80
 - LMS-threshold, 123, 124
 - MSE (*see* Mean squared error function)
 - penalty-term methods and, 220–221, 226–231
 - RMS (*see* Root mean square error function)
 - selection of, 253
 - sensitivity methods and, 220, 221–225
 - SSE (*see* Sum of squared errors function)
 - system energy and, 177
 - in training steps, 49–50, 52–69
- Error surface
 - algorithm assumptions and, 120
 - characteristics of, 113–117
 - gain scaling and, 114, 132–134
 - Hessian matrix and, 127–132
 - learning rate and, 72
 - local minima of, 115–116, 117, 121–126
 - momentum and, 87–89
 - quadratic function and, 179
 - radial features of, 114–116
 - stair-steps on, 113–114, 121
 - total gradient of, 117
 - troughs and ridges of, 116–117
 - weight-space symmetries for, 118–119
- $E(t)$ curves
 - constructive methods and, 199, 201

- learning rate and, 77–80
- momentum and, 90–93
- Evaluation-only methods
 - deterministic, 159–163
 - stochastic, 175–179
- Evolutionary algorithm. *See* Genetic algorithm
- Exclusive-OR function, 18, 19, 107, 251–253
- Exploratory moves, 160
- Fault tolerance, 4
- Feedforward indexing, 50–51
- Feedforward structure, introduction to, 2–5
- Final prediction error (FPE), 260–261
- Finite-difference approximation, 56–57, 132
- Fitness, 185, 186, 189, 192
- Fletcher-Reeves form, 167
- Forward propagation, 51–52, 63–64
- Fourier transforms, 36–37
- FPE (final prediction error), 260–261
- Functions. *See also names of individual functions*
 - components of, 31
 - introduction to, 1–2, 7, 9–11
 - variety of, 35, 37
- Fuzzy logic, 148–149
- Fuzzy rule systems, 110
- GA. *See* Genetic algorithm
- Gain competition, 232–233
- Gain scaling, 105, 114
 - generalization and, 133–134, 274, 283–287
 - hard-limiters and, 134
 - learning rate and, 132–133
 - sigmoid saturation and, 133
- Game playing, 12
- Gaussian cumulative distribution function, 284–285, 288, 311–314
- Gaussian probability distribution function, 288–289, 311–314
- Gaussian weight distribution, 100–102
- Gauss-Newton method, 130, 172
- Generalization
 - Akaike's methods and, 260–261
 - algorithm factors and, 157, 249–253
 - Bayes' rule and, 258–260
 - bias and, 240, 249, 256, 258, 267
 - complexity and, 241–242, 244–247
 - constructive methods and, 198–199, 268
 - cross-validation of, 257–258, 273, 291
 - data factors and, 242–248
 - definitions for, 239–240
 - early stopping for, 265–266
 - factors affecting, 240–256
 - gain scaling and, 133–134
 - gradient correlation and, 151
 - improvement of, 265–276
 - information and, 240–241, 266–267, 271–272, 274–276
 - as interpolation, 239, 240, 242
 - model mismatch and, 248
 - modularity and, 255
 - momentum and, 137
 - noise and, 247–248, 273–274, 277–292
 - overtraining and, 249–251
 - PAC learning and, 261–263
 - performance assessment of, 257–264
 - physical models and, 276
 - pruning and, 219, 237, 268
 - regularization and, 266–267, 281–283
 - replicated networks and, 272–273
 - rules of thumb for, 219, 241, 245, 246
 - size and, 241–242, 244–247, 248
 - training time and, 153
 - variables and, 253–255
 - VC dimension and, 261–264
 - weight decay and, 269–270, 285–287
- General position, vectors in, 20–23
- Genetic algorithm (GA), 178–179
 - advantages of, 185, 194, 195
 - applications of, 191–194
 - constructive methods and, 217
 - crossover and, 186, 187, 188
 - disadvantages of, 185, 194, 195
 - error surface and, 119
 - example for, 189–191
 - fitness scaling and, 189
 - incompatible genomes and, 193–194
 - mutation and, 186, 187, 188–189
 - pruning and, 193, 235
 - steps in, 186–187
 - variations of, 187–188
- Genetic code, 185, 186
- Genetic programming, 193
- Genomes, incompatible, 193–194
- Goals, 155. *See also* Objective function
- Graceful degradation, 4
- Gradation, in hyperplane geometry, 18
- Gradient-based methods, 158
 - first-order, 163–169
 - second-order, 169–175, 182
- Gradient correlation, 150–151
- Gradient descent, 57–63
 - best-step steepest, 165–166
 - classical optimization and, 155, 163–169, 179
 - conjugate, 166–169, 179, 182, 252
 - constructive methods and, 212
 - convergence rate of, 164, 295–297
 - jitter and, 280

- Gradient descent (*cont.*)
 - momentum and, 85–87
- Gradient reuse, 150
- Hard-limiters, 134
- Hard weight sharing, 269
- Hebbian learning, 30
- Hessian matrix
 - approximation of, 129–132
 - gradient-based methods and, 163, 166–167, 170–174, 182
 - ill-conditioned, 127–128, 179
 - learning rate and, 72, 81–84
 - node splitting and, 214–215
 - problem size and, 157
 - properties of, 127–132
 - pruning and, 223–225
 - quadratic function and, 179
 - search then converge method and, 147
- Hidden layers
 - definition of, 2, 31
 - equivalent solutions and, 194
 - error surface and, 119, 124, 126
 - representational capabilities and, 32–47
 - weight-space symmetries and, 119
- Hidden nodes
 - approximation error and, 37
 - derivative calculation for, 54, 56
 - nonrandom initialization and, 106, 107, 108
 - random initialization and, 102, 103–104
- Hint functions, 275
- Holdout method, 257–258
- Homogeneously linear separability, 20
- Hooke-Jeeves pattern search, 160–161
- Hopfield networks, 49, 183
- Hyperplanes, 15–18
 - capacity of, 20–23
 - nonrandom initialization and, 107
 - random initialization and, 102, 103
 - representational capability and, 42–44
 - stair-steps and, 114
- If-then rules, 148–149
- Ill-conditioning, 127–128, 179
- Ill-posed problems, 266
- Indexing, feedforward, 50–51
- Information minimization, 271–272
- Initialization. *See* Weight initialization
- Input distributions, 100–102
- Input nodes, 31
 - derivative calculation for, 55–56
 - random initialization and, 102
- Interactive pruning, 232
- Interpolation, 239, 240, 242
- Inverse abs function, 317–318
- Jitter
 - calculations for, 311–314
 - convolution and, 277–281, 311–314
 - definition of, 277
 - generalization and, 247, 273–274, 277–292
 - on-line learning and, 60
 - random restarts and, 123
 - regularization and, 281–283
 - sigmoid scaling and, 283–287
- k-fold cross-validation, 257, 273
- k-means algorithm, 107
- Knowledge
 - Bayesian methods and, 258
 - constraining of, 244–245
 - initialization and, 106
- Knowledge-based systems, 275–276. *See also* Rule-based systems
- Laplace transforms, 91, 92, 94
- Layered architecture, introduction to, 2–6
- Layers
 - components of, 31
 - counting of, 31
 - delta attenuation in, 72, 84–85
 - one hidden, sufficiency of, 33–41
 - representational capabilities and, 32–47
 - two hidden, sufficiency of, 32–33, 38–39, 109
- Learning. *See* Algorithms; Supervised learning; Training
- Learning rate, 57, 62
 - additive versus multiplicative changes in, 140–141
 - algorithm variations and, 135–151
 - batch, 77–79
 - classical optimization and, 164
 - delta attenuation and, 84–85
 - effective, 77, 87, 88, 90
 - example for, 73–74
 - gain scaling and, 132–133
 - momentum and, 71, 72–77, 80, 87–90, 92
 - MSE function and, 71, 73
 - on-line, 80
 - randomness and, 80
 - scaling of, 85
 - SSE function and, 71
 - training time and, 71–85
 - values for, 71
- Learning rate selection
 - adaptive methods and, 95
 - factors affecting, 71–72
 - momentum and, 89
 - from trace(**H**), 81–82

- Learning rules, 23–30. *See also* Algorithms
- Least mean squares (LMS) algorithm, 29, 298. *See also* Widrow-Hoff learning rule
 - error function and, 123, 124
 - momentum and, 90
- Leave-one-out method, 257
- Levenberg-Marquardt method, 128, 144, 172–173, 182
- Linear discriminant analysis, 306–309
- Linear output networks, 283–285
- Linear regression, 23, 29, 293–298
- Linear separability, 18–23, 28, 124
- Linear threshold units (LTUs), 15, 23, 113
 - constructive methods and, 204–209
 - pruning and, 232
- Line search, 158–159
- LMS. *See* Least mean squares algorithm
- Loading problem, 197
- Local bottlenecks, 232–233
- Local minima, 115–117, 121–126, 157
- Logarithmic error function, 9, 50. *See also* Cross-entropy error function
- Logistic regression, 23
- LTUs. *See* Linear threshold units
- LVQ method, 107

- Marchand's algorithm, 209–212
- Master units, 206–209
- Mean squared error (MSE) function, 9, 50, 155
 - dynamic node creation and, 200
 - error surface and, 117, 123, 129
 - generalization and, 239, 243, 253, 259, 261, 272, 276
 - initialization and, 110
 - learning rate and, 71, 73, 77, 139
 - pattern weighting and, 151
 - Widrow-Hoff learning rule and, 29–30
- Meiosis networks, 212–213
- Meta-optimization, 157
- Minima. *See* Local minima
- Mirror symmetry problem, 211
- MLP. *See* Multilayer perceptron
- Model mismatch, 248
- Modularity, 255
- Momentum, 62–63
 - algorithm variations and, 137, 140, 142, 144
 - classical optimization and, 165
 - frequency response and, 93
 - gradient correlation and, 150–151
 - impulse response and, 91–92, 94
 - inertia and, 89
 - learning rate and, 71, 72–77, 80, 87–90, 92
 - small-signal analysis and, 90–95
 - step response and, 92, 94–95
 - training time and, 74–77, 85–95
 - values for, 71
- MSE. *See* Mean squared error function
- Multilayer networks, importance of, 19
- Multilayer perceptron (MLP)
 - as cascade of single-layer perceptrons, 31
 - compared to real nervous systems, 3–4
 - component layers of, 31
 - definition of, 29
 - introduction to, 2–6
 - representational capabilities of, 31–47
 - universal approximation by, 35–38
- Mutation, 186, 187, 188–189

- Nearest neighbor methods, 107–108, 152, 216
- Nelder-Mead simplex search, 161
- Nervous systems, 3–4
- Neural networks
 - as optimizing systems, 183
 - power of, 35, 37
- Neurons, 1
- Newton's method, 128, 145, 152. *See also* Quasi-Newton methods
 - as classical technique, 170–172, 179
 - linear regression and, 294–295
- Node splitting, 212–215
- Noise, 157, 247–248, 273–274. *See also* Jitter
 - effects of, 277–292
 - static, 290–291
- Nonconvex regions, 33–35
- Nonlinearity, 15, 284–285, 288
- Nonrandom initialization, 105–106
 - decision tree classifier for, 108–109
 - discriminant analysis for, 107–108
 - fuzzy rule systems for, 110
 - principal components analysis for, 106
 - prototype, 108
 - symbolic rule systems for, 109–110
- Notation for layered network, 31
- NOT function, 109

- OBD (optimal brain damage), 131, 223, 225, 235
- Objective function. *See also* Error function
 - introduction to, 7, 9–11
 - penalty terms for, 10–11
 - selection of, 155–156
- OBS (optimal brain surgeon), 223–225, 235
- Off-line algorithms, 49, 62
- One-hidden-layer networks, 33–41, 124
- On-line learning, 59–62, 155
 - algorithm variations and, 147
 - eigenvalue estimation and, 82–84
 - error surface and, 117, 122–123
 - generalization and, 251

- On-line learning (*cont.*)
 - pseudocode example for, 65–66
 - random initialization and, 105
 - training time and, 68, 80
- Optimal brain damage (OBD), 131, 223, 225, 235
- Optimal brain surgeon (OBS), 223–225, 235
- Optimal perceptron, 26
- Optimization
 - as cost minimization, 156
 - meta-, 157
- Optimization methods
 - back-propagation (*see* Back-propagation algorithm)
 - classical (*see* Classical optimization methods)
 - classification of, 158
 - selection of, 156–158, 182
 - variety of, 57, 67, 155
- OR function, 39, 109
- Orthogonal vectors, 29
- Orthonormal vectors, 29
- Oscillation, 213–214
- Outer-product approximation, 130–131
- Output nodes, 31
 - derivative calculation for, 53–54
 - random initialization and, 102, 104
- Overtraining
 - complexity and, 241–242
 - constructive methods and, 198
 - jitter and, 292
 - learning dynamics and, 249–251
 - pruning and, 237
- PAC (probably approximately correct) learning, 261–263
- Paralysis, from sigmoid saturation, 68, 70
- Parametric classifier, 152–153
- Parametric models, 248
- Pattern-mode learning, 62
- Pattern moves, 160
- Pattern weighting heuristics, 151
- PCA. *See* Principal components analysis
- Penalty terms, 10–11, 220–221, 226–231, 237
- Perceptron learning algorithm, 23–24
 - constructive methods and, 202, 205, 210–212
 - convergence proof for, 25–27
- Perceptrons
 - definitions for, 1–2, 28–29
 - multilayer (*see* Multilayer perceptron)
 - Rosenblatt's original, 28–29
- Perturbation, 56
- Pinnacle function, 38
- Pocket algorithm, 28, 205
- Polak-Ribiere form, 167
- Powell's conjugate direction method, 161–163
- Prediction systems, and generalization, 257–264
- Principal components analysis (PCA), 299–302
 - autoencoder networks and, 303–305
 - discriminant analysis and, 306–310
 - for initialization, 106
 - for pruning, 234–235
 - unsupervised learning and, 12
- Processing units, 1
- Projection pursuit regression, 217
- Projections, hyperplane, 15
- Prototypes, 108, 207–208
- Pruning methods
 - bottlenecks and, 232–234
 - constructive methods and, 198, 201
 - example for, 219
 - gain competition, 232–233
 - generalization and, 219, 237, 268
 - genetic algorithm, 193, 235
 - interactive, 232
 - optimal brain damage, 131, 223, 225, 235
 - optimal brain surgeon, 223–225, 235
 - penalty terms, 220–221, 226–231, 237
 - principal components, 234–235
 - sensitivity, 220, 221–225, 237
 - skeletonization, 221–222
 - weight decay, 230–231
 - weight elimination, 228–229
- Quadratic approximation, 179–180
- Quasi-Newton methods, 174–175
- Quickprop method, 145–147, 152
- Radial basis functions, 198, 288
- Radial features, 114–116
- Random initialization, 97–98
 - calculations for, 98–102
 - constrained, 103–105
 - error surface and, 120, 121, 133
 - parameters for, 100–102
- Randomness
 - learning rate and, 80
 - on-line learning and, 60, 80
 - search then converge method and, 147
- Random restarts, 121–123, 157
- Regions, and hyperplanes, 42–44
- Regression, 23, 29, 293–298
 - classification versus, 156–157
 - projection pursuit, 217
- Regularization, 266–267, 281–283
- Reinforcement learning, 12
- Replicated networks, 272–273
- Representational capabilities, 31–47
- Robustness, 157
- Root mean square (RMS) error function, 77, 222, 285
- Rosenblatt's perceptron, 28–29

- Rprop (resilient propagation), 85, 142–145, 152, 252, 291
- Rule-based systems, 109–110, 217, 275–276
- SAB (self-adapting back propagation), 142
- Sample size. *See* Size
- Saturation. *See* Sigmoid saturation
- Scaled conjugate gradient descent, 168
- Scaling, 189. *See also* Gain scaling
- Schemata theory, 188
- Search then converge method, 147–148
- Self-repair, 4
- Sensitivity methods, for pruning, 220, 221–225, 237
- Separability. *See* Linear separability
- Sigmoid function, 1, 54, 315–318
 - bias weights and, 231
 - error surface and, 113–116, 118–119, 123, 132
 - for single-layer networks, 15
- Sigmoid-like functions, 51, 315–318
- Sigmoid saturation, 68, 69, 70
 - algorithm variations and, 141
 - gain scaling and, 133
 - ill-conditioning and, 128
 - random initialization and, 97, 102
- Sigmoid scaling, 283–287
- Sign function, 317
- Simplex search, 161
- Simplicity, 241. *See also* Complexity
- Simulated annealing, 157, 176–178
- Single-hidden-layer networks, 33–41, 124
- Single-layer networks
 - hyperplane geometry and, 15–18, 20–23
 - importance of, 20
 - learning rules for, 23–30
 - limitations of, 18, 19
 - linear separability and, 18–23
 - local minima for, 123–124
- Size. *See also* Pruning methods
 - algorithm selection and, 157
 - capacity versus, 41–47
 - complexity and, 241–242, 244–247
 - constructive methods and, 197, 199
 - depth versus, 38–41
 - local minima and, 126
 - model mismatch and, 248
 - training time and, 68, 71, 199
- Skeletonization, 221–222
- Small-signal analysis, 90–95
- Soft weight sharing, 269
- Squashing function, 1, 51. *See also* Sigmoid function
- SSE. *See* Sum of squared errors function
- Stacked-generalization, 273
- Stair-steps, 113–114, 121. *See also* Step function
- Standard optimization. *See* Classical optimization methods
- Star topology, 114–115
- Static noise, 290–291
- Step function, 1, 113–114, 121, 317
 - in Adaline networks, 29
 - hard-limiters and, 134
 - momentum and, 92, 94–95
 - in perceptron learning algorithm, 24
 - for single-layer networks, 15, 24, 29
- Step size parameter, 57
- Stochastic methods, 60, 62, 72, 158, 175–179
- Structure
 - constructive methods and, 197–217
 - pruning and, 219–237
 - selection of, 197
- Subpopulations, and genetic algorithm, 192–193
- Subproblems, 255
- Sum of squared errors (SSE) function, 9, 50, 52, 54
 - error surface and, 117, 134
 - generalization and, 259
 - jitter and, 280
 - learning rate and, 71, 77
 - pruning and, 221, 227, 230
- SuperSAB, 142
- Supervised learning. *See also* Training
 - in back-propagation, 49
 - definition of, 7
 - introduction to, 7–14
 - model of, 7–11
- Symbolic rule systems, 109–110. *See also* Rule-based systems
- Symmetric functions, 40
- Symmetries, weight-space, 118–119
- Symmetry breaking, 97
- System energy, and simulated annealing, 177
- Tanh function, 1, 315–316
 - back-propagation and, 54
 - error surface and, 115, 118, 119, 123, 128
 - generalization and, 251
 - for single-layer networks, 15
- Target, effective, 278–281, 289
- Target outputs, introduction to, 7–10
- Teacher, 7, 11, 12–14, 49
- Tessellation, Voronoi, 215–216
- Threshold term, 17–18. *See also* Linear threshold units
- Tiling algorithm, 206–209
- Trace(**H**), 81–82
- Training. *See also* Algorithms; Supervised learning
 - definition of, 7
 - jitter and, 277–292
 - optimal amount of, 250

- Training (*cont.*)
 - as optimization problem, 155
 - over-, 198, 237, 241–242, 249–251, 292
 - steps in, 49
- Training set size. *See* Size
- Training time, 67–85
 - factors that increase, 68–69
 - initialization and, 97, 105, 106
 - momentum and, 74–77, 85–95
 - pruning and, 219, 237
 - scaling of, 68–70
 - training set size and, 68, 71, 199
 - variations to improve, 135–153
- Two-hidden-layers networks, 32–33, 38–39, 109
- Two-spirals problem, 108, 144, 254
- Underfitting, 241
- Uniform weight distribution, 100, 105
- Universal approximation, 5, 35–38
- Unsupervised learning, 11–12
- Upstart algorithm, 204–206
- VC (Vapnik-Chervonenkis) dimension, 261–264
- Vogl's method, 136–137
- Voronoi tessellation, 215–216
- Weight clipping, 225
- Weight decay, 63, 230–231
 - generalization and, 269–270, 285–287
 - momentum and, 93–95
- Weight distributions, 100–102, 105
- Weight elimination, 228–229
- Weight initialization
 - constructive methods and, 217
 - nonrandom, 105–110
 - random, 97–105, 120, 121, 133
- Weights
 - in back-propagation, 49–70
 - oscillation of, 213–214
 - perturbation of, 56
 - representational capability and, 32–46
 - updates for (*see* Algorithms)
- Weight-space symmetries, 118–119
- Well-posed problems, 266
- Widrow-Hoff learning rule, 29–30, 202, 298. *See also*
 - Least mean squares algorithm
- Wrongly OFF error, 205–206
- Wrongly ON error, 205–206
- XOR function, 18, 19, 107, 251–253

This excerpt from

Neural Smithing.
Russell D. Reed and Robert J. Marks II.
© 1999 The MIT Press.

is provided in screen-viewable form for personal use only by members of MIT CogNet.

Unauthorized use or dissemination of this information is expressly forbidden.

If you have any questions about this material, please contact
cognetadmin@cognet.mit.edu.