Saturation, Flat-spotting

- Shift up Derivative
- Weight Decay
- No derivative on output nodes
Weight Initialization

- Can get stuck if initial weights are 0 or equal
- If too large - node saturation from $f'(net)$
- If too small - very slow due to propagation back through weights
- Usually small Gaussian distribution around a 0 mean
- $C/\text{square-root(fan-in)}$
- Background knowledge
Learning Rate

- Very unstable for high learning rates (typical .1 - .25)
- Can estimate optimal if calculate the Hessian - $1/\max$ eigenvalue of the Hessian
- Larger rate for hidden nodes
- Rate divided by fan-in - more equitable rate of change across nodes
- Learning speed vs. Generalization accuracy
Adaptive Learning Rates

• Local vs. Global - start small
• Increase LR when error is consistently decreasing (long valleys, smooth drops, etc.) - stop when gradient is changing (non-smooth areas of error surface) - decrease (rapidly) when error begins to increase
  – $C(t) = 1.1C(t-1)$ if $E(t) < E(t-1)$
  – $C(t) = .9C(t-1)$ if $E(t) < E(t-1)$ - often a fast non-linear drop-off
  – $C(t) = C(t-1)$ if $E(t) \sim E(t-1)$
• Second derivative of error
• Sign change in derivative
Momentum

- Amplifies effective Learning rate when there is a consistent gradient change
- Helps avoid cross-stitching
- Can avoid local minima (for better or for worse)
Generalization/Overfitting

- Inductive Bias - small, similarity, critical variables, etc.
- Optimal model/architecture
- Neural Net - tendency to build (from simple weights) until sufficient, even though large, vs. a pre-set polynomial, etc.
- Holdout set - keep separate from Test set
- Stopping criteria - especially w/constructive
- Noise vs. Exceptions
- Regularization - favor smooth functions
- Jitter
- Weight Decay
Empirical Testing/Comparison

• Proper use of Test/hold-out sets
• Tuned algorithm problem
• Cross-Validation - small data, which to use
• Statistical Significance
• Large cross-section of applications
Higher Order Gradient Descent

- QuickProp
- Conjugate Gradient, Newton Methods
- Hessian Matrix
- Less iterations, more work/iteration, assumptions about error surface
- Levenberg-Marquardt
Training set/Features

- Relevance
- Invariance
- Encoding
- Normalization/Skew
- How many - curse of dimensionality - most relevant, PCA, etc.
- Higher order - Combined algorithms, feature selection, domain knowledge
Training Set

- How large
- Same distribution as will see in future
- Iteration vs. Oracle
- Skew
- Error Thresholds
- Cost function
- Objective functions
Constructive Networks

• ASOCS
• DMP - Dynamic Multilayer
• Convergence Proofs - Stopping criteria
• Cascade Correlation
• Many variations
• BP versions - node splitting
Pruning Algorithms

• Drop node/weight and see how it effects performance - Brute force
• Drop nodes/weights with least effect on error
• Do additional training after each prune
• Approximate the above in a parsimonious fashion
• First and second order error estimates for each weight
• Penalize nodes with larger weights (weight decay) - if they are driven close to 0 then can be dropped
• If output is relatively constant from a node
• If output of multiple nodes correlate (redundant)
Learning Ensembles

- Modular Networks
- Stacking
- Bagging
- Boosting
- Wagging