Learning and Generalization in Artificial Neural Networks

Martin Hagan
Oklahoma State University

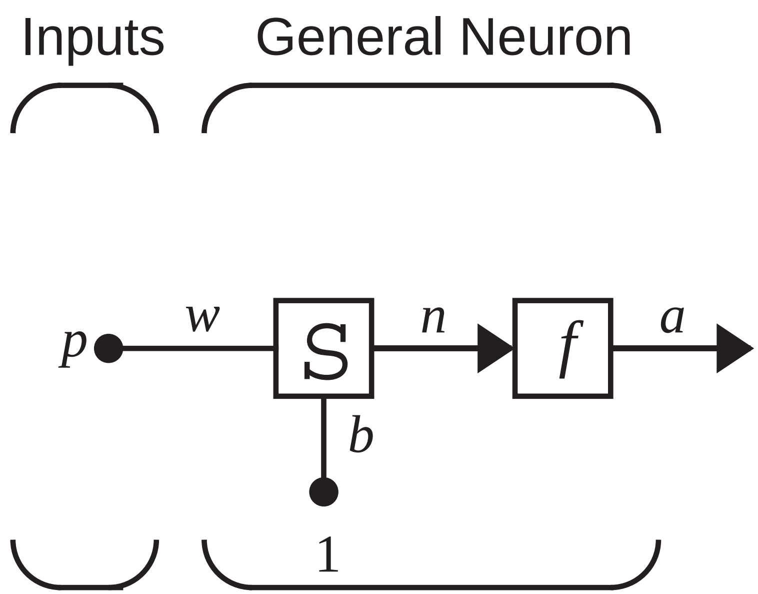
A cat that once sat on a hot stove
will never again sit on a hot stove
or on a cold one either.

Mark Twain

Outline

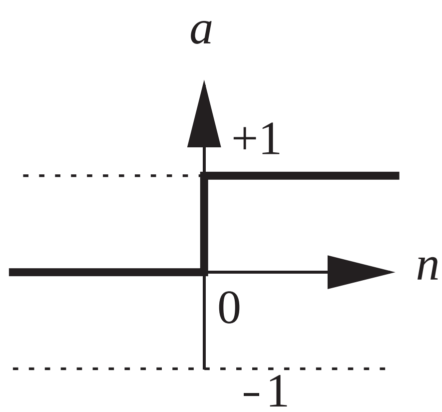
- What is an Artificial Neural Network?
- Nonlinear Regression
- Generalization
- Techniques for Improving Generalization
- Regularization
  - Bayesian Regularization
- Early Stopping
- Relationship Between Early Stopping and Regularization
- Summary

Single-Input Neuron



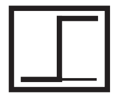
$$a = f(wp + b)$$

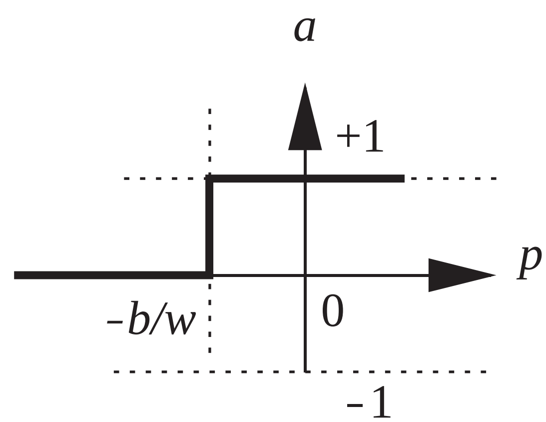
Transfer Functions



$a = hardlim(n)$

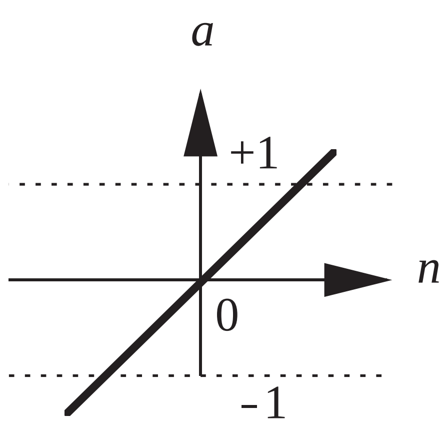
Hard Limit Transfer Function





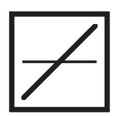
$a = hardlim(wp + b)$

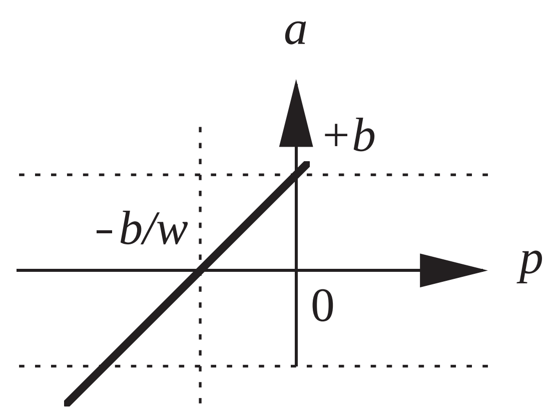
Single-Input *hardlim* Neuron



$a = purelin(n)$

Linear Transfer Function

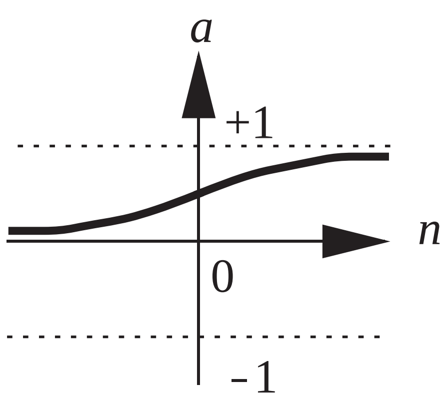


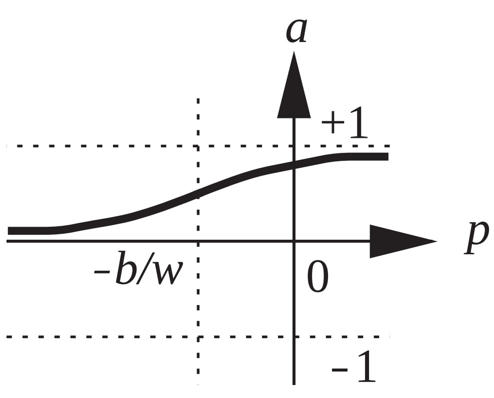


$a = purelin(wp + b)$

Single-Input *purelin* Neuron

Transfer Functions





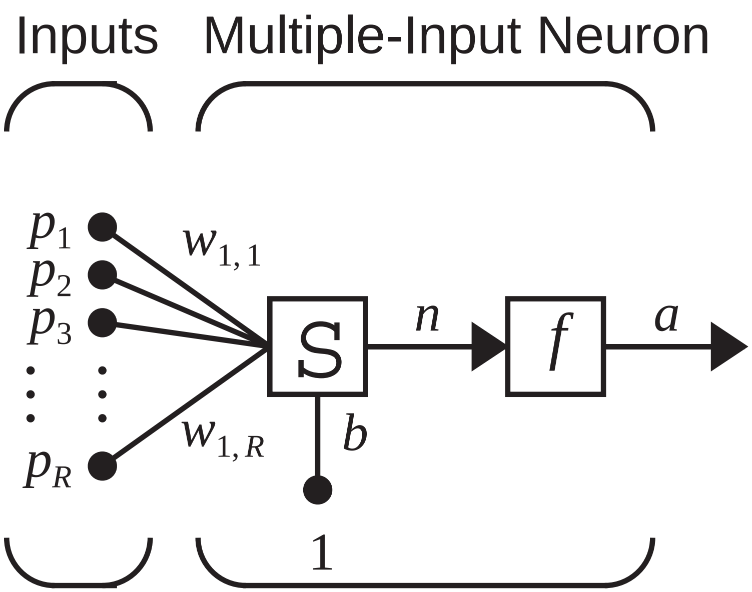
$$a = logsig(n)$$

Log-Sigmoid Transfer Function

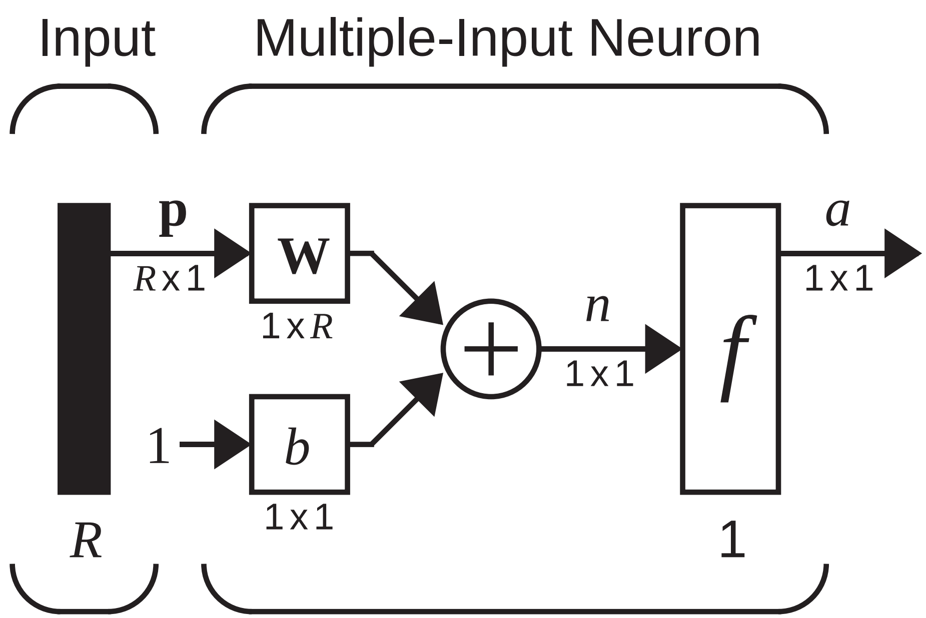
$$a = logsig(wp + b)$$

Single-Input *logsig* Neuron

Multiple-Input Neuron



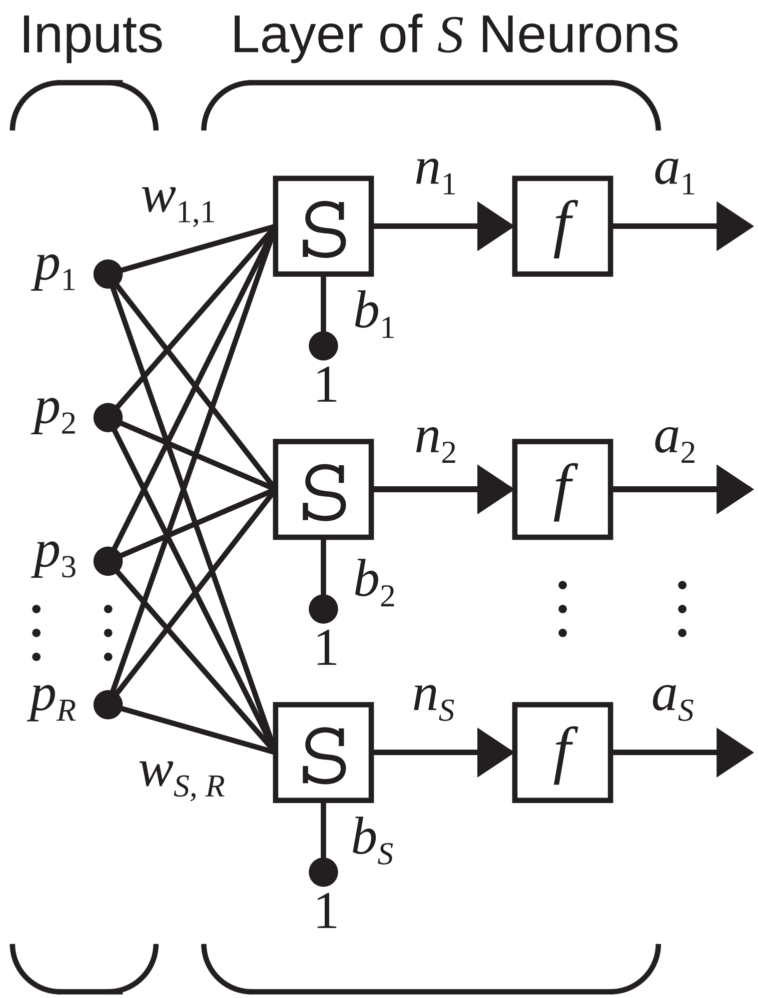
$a = f(\mathbf{W}\mathbf{p} + b)$



$a = f(\mathbf{W}\mathbf{p} + b)$

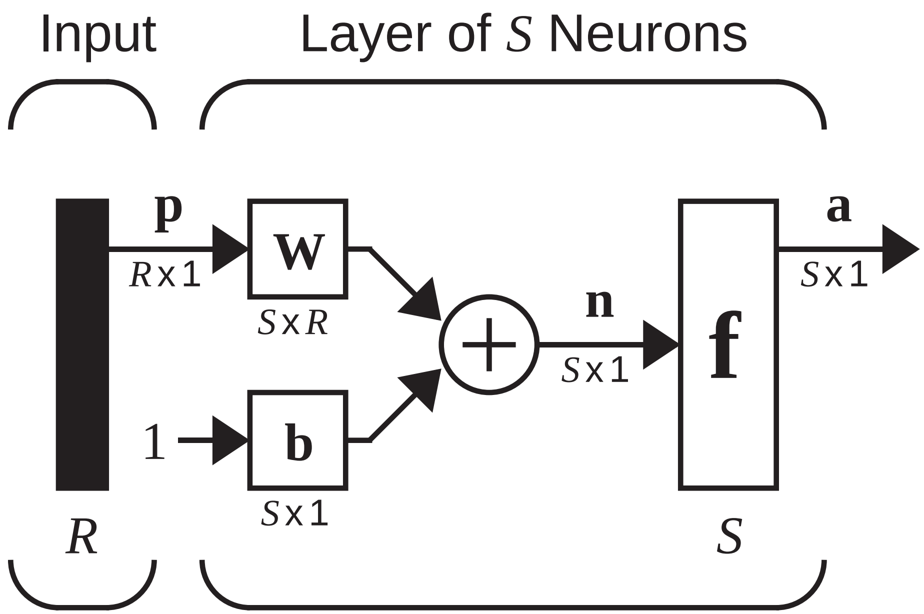
Abbreviated Notation

Layer of Neurons



$$\mathbf{a} = \mathbf{f}(\mathbf{Wp} + \mathbf{b})$$

Abbreviated Notation

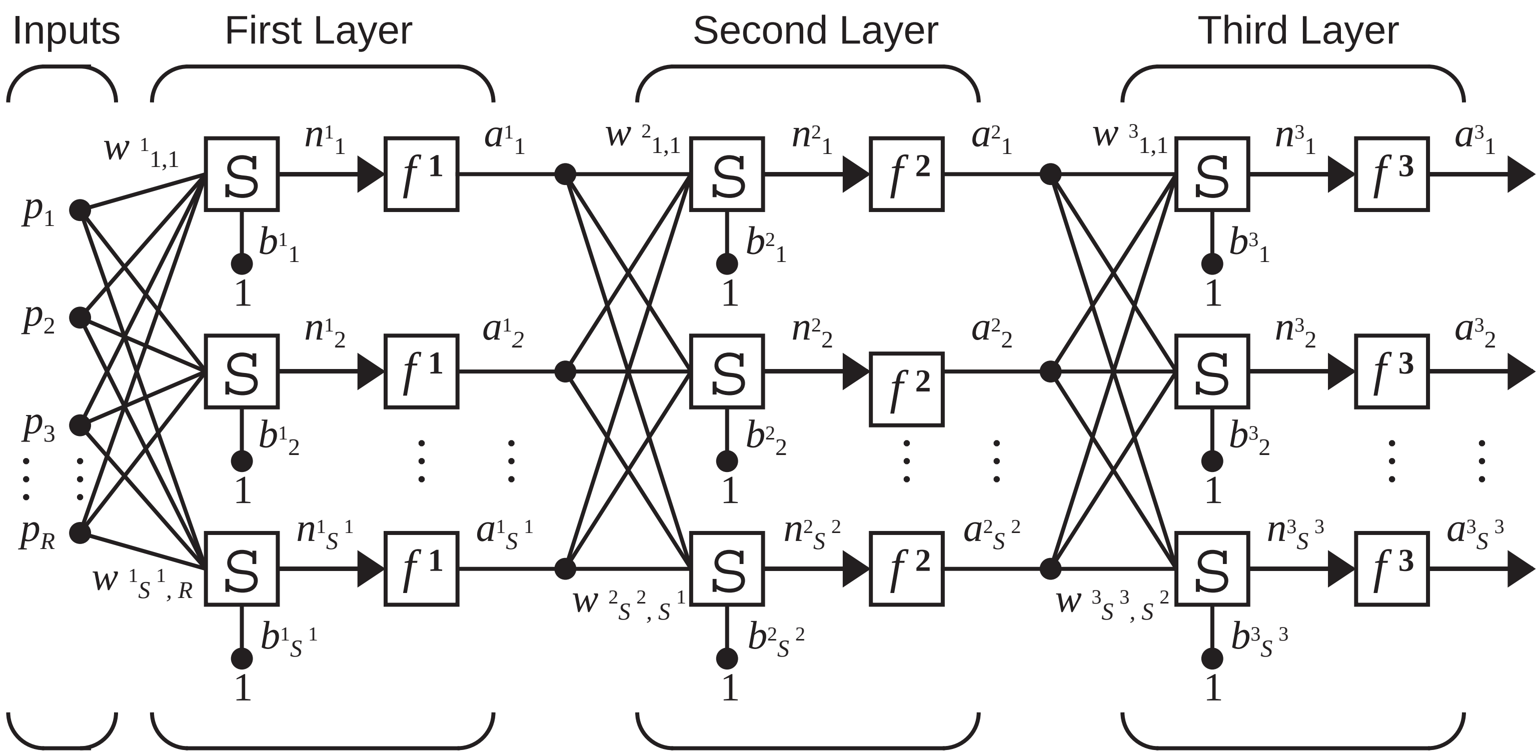


$$\mathbf{a} = \mathbf{f}(\mathbf{W}\mathbf{p} + \mathbf{b})$$

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,R} \\ w_{2,1} & w_{2,2} & \dots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \dots & w_{S,R} \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_R \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_S \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_S \end{bmatrix}$$

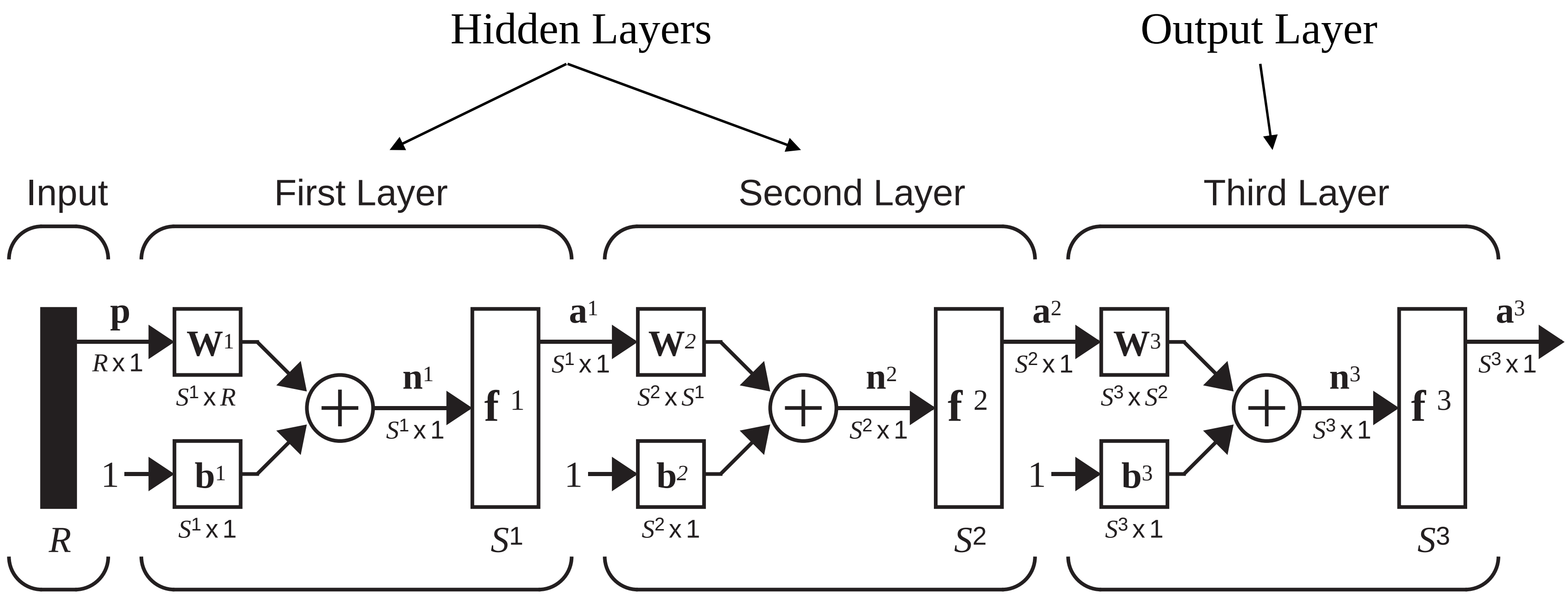
Multilayer Network



$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b}^1) \qquad \mathbf{a}^2 = \mathbf{f}^2(\mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^2) \qquad \mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{a}^2 + \mathbf{b}^3)$$

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3)$$

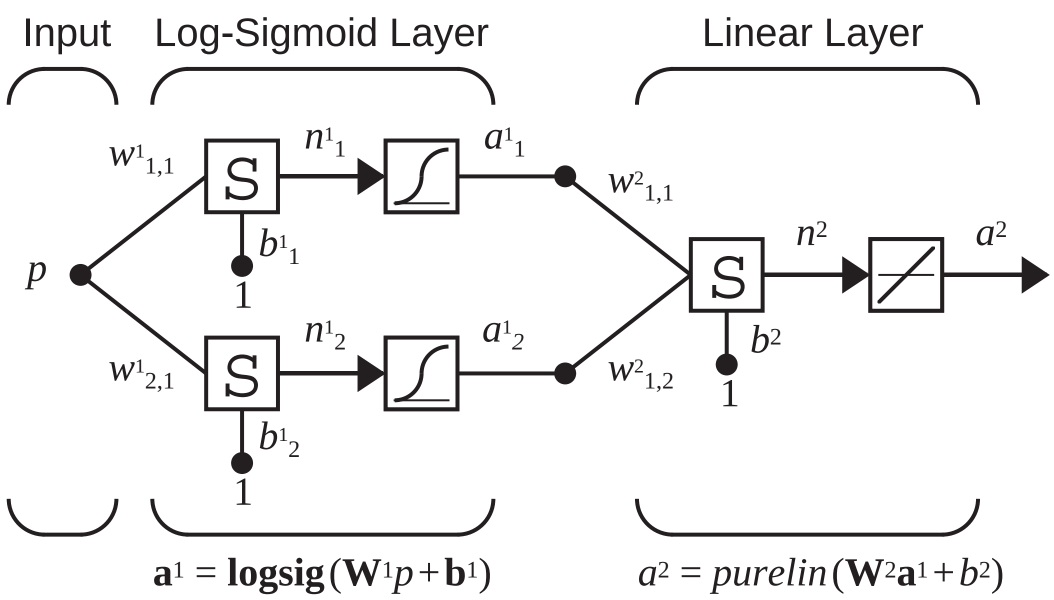
Abbreviated Notation



$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b}^1)$ $\mathbf{a}^2 = \mathbf{f}^2(\mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^2)$ $\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{a}^2 + \mathbf{b}^3)$

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3)$$

Function Approximation Example



$$f^1(n) = \frac{1}{1 + e^{-n}}$$

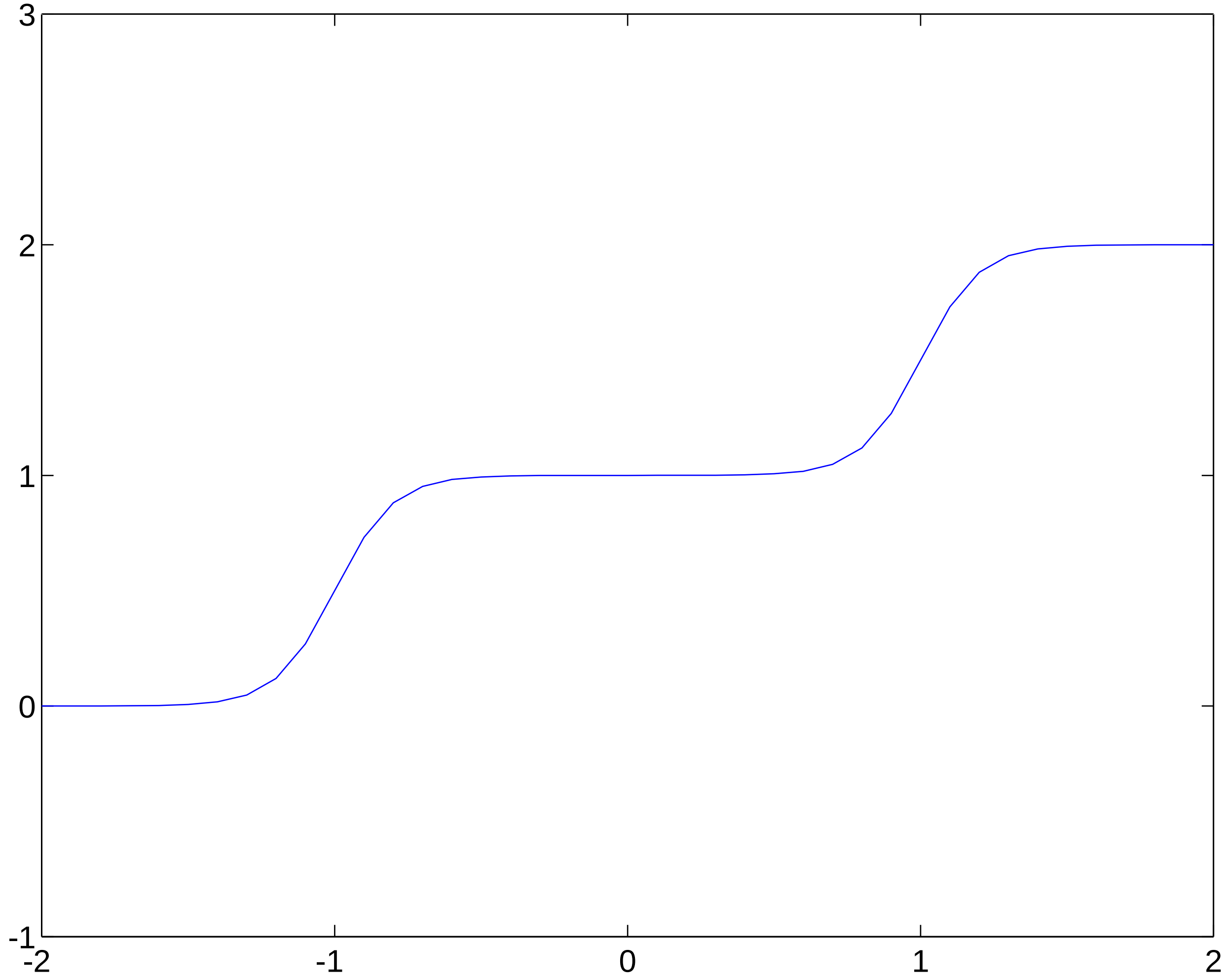
$$f^2(n) = n$$

Nominal Parameter Values

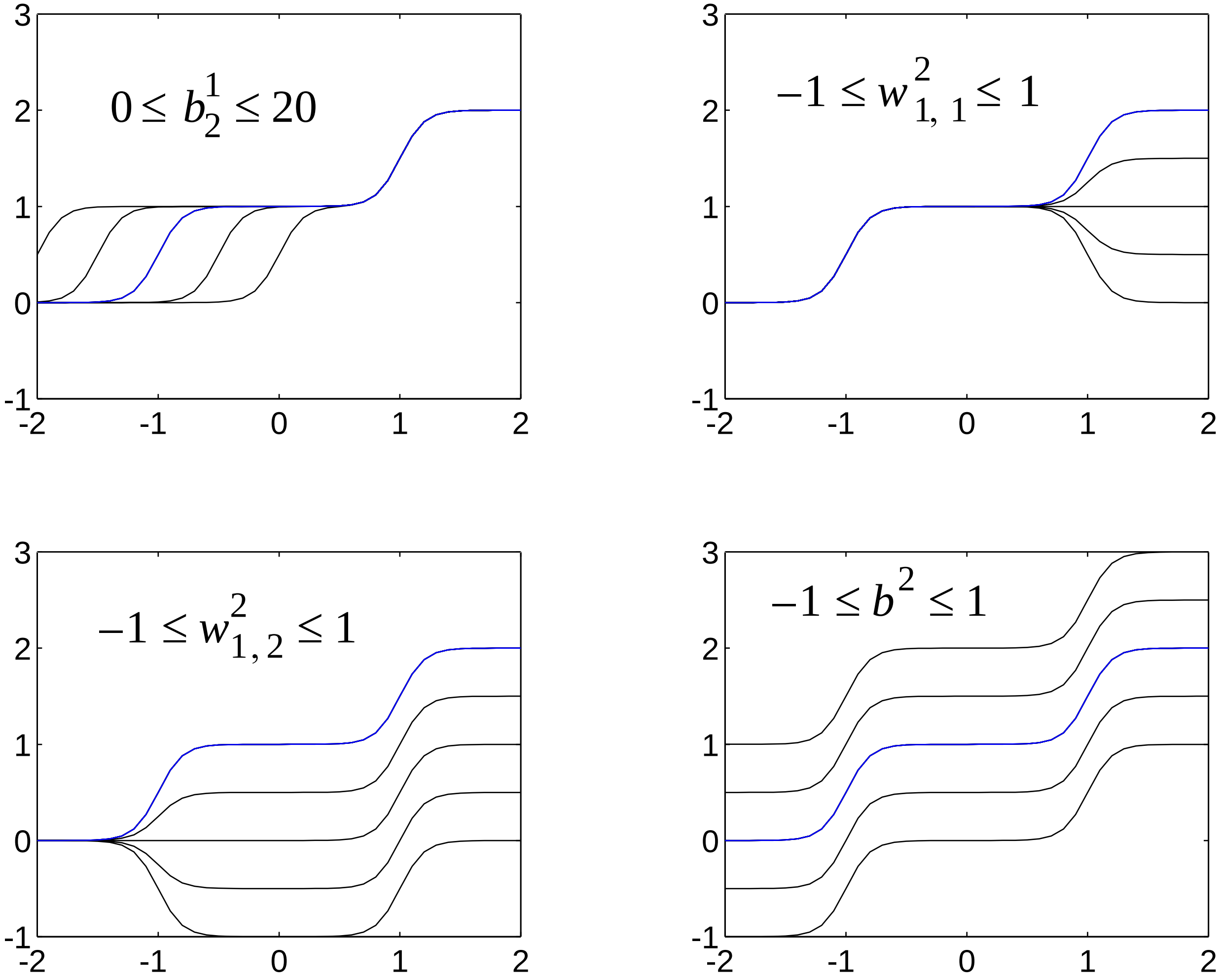
$$w^1_{1,1} = 10 \qquad w^1_{2,1} = 10 \qquad b^1_1 = -10 \qquad b^1_2 = 10$$

$$w^2_{1,1} = 1 \qquad w^2_{1,2} = 1 \qquad b^2 = 0$$

Nominal Response



Parameter Variations

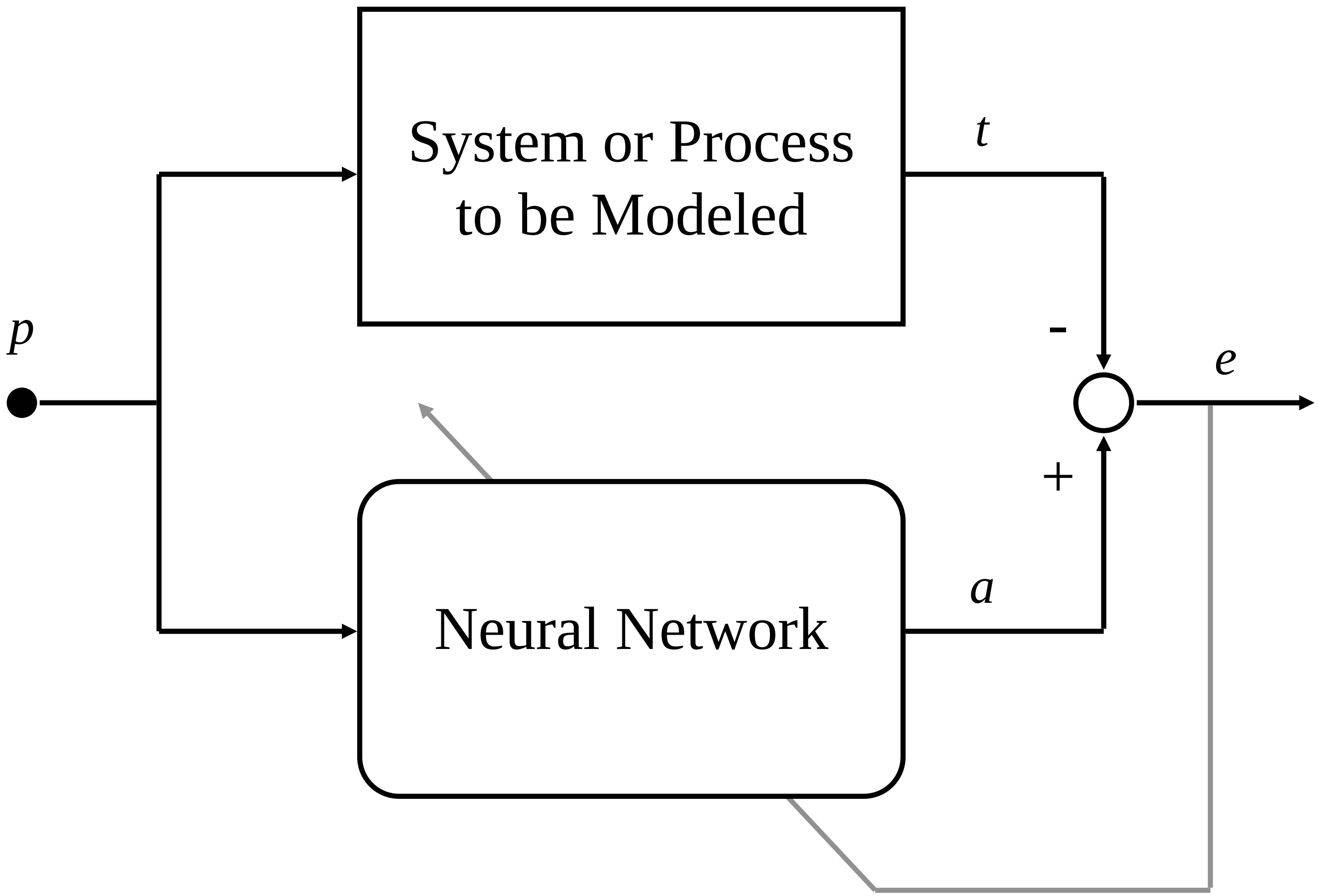


Universal Approximator

A two-layer network with a bounded, monotone-increasing transfer function in the first layer and a linear transfer function in the second layer can approximate any continuous function to an arbitrary accuracy over a bounded interval, given a sufficient number of neurons in the first layer.

Cybenko (1988)
Weierstrass (1885)

Neural Network Training



Problem Statement

Training Set

$$\{p_1, t_1\}, \{p_2, t_2\}, \dots, \{p_n, t_n\}$$

Target Generation

$$t_i = g(p_i) + \varepsilon_i$$

Performance Index for Training

$$F = E_D = \sum_{i=1}^{n} (t_i - a_i)^2$$

Regression Output

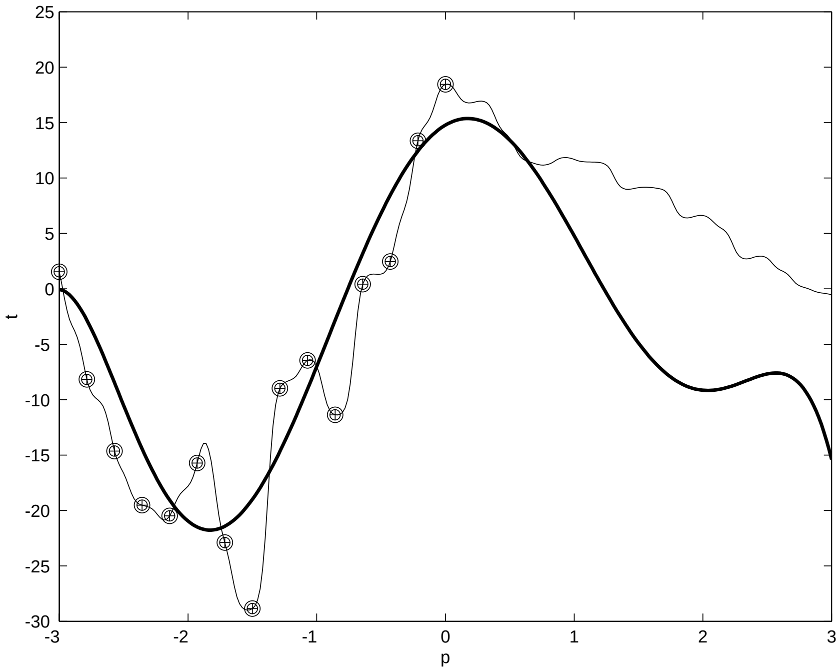
Gradient Descent Optimization

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha \frac{\partial F}{\partial w_{i,j}^m} \qquad b_i^m(k+1) = b_i^m(k) - \alpha \frac{\partial F}{\partial b_i^m}$$

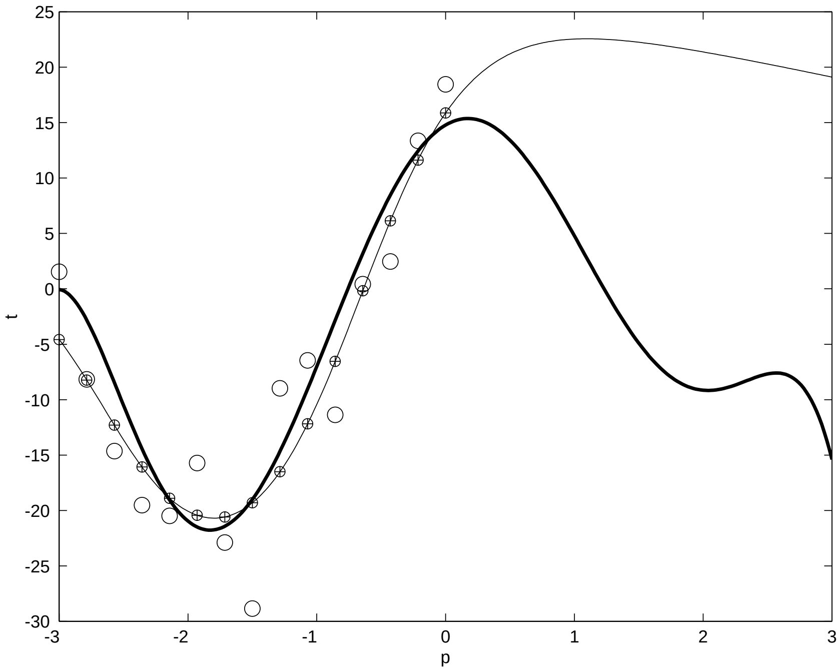
Modeling Problems

- Overfitting
- Extrapolation

Overfitting & Extrapolation



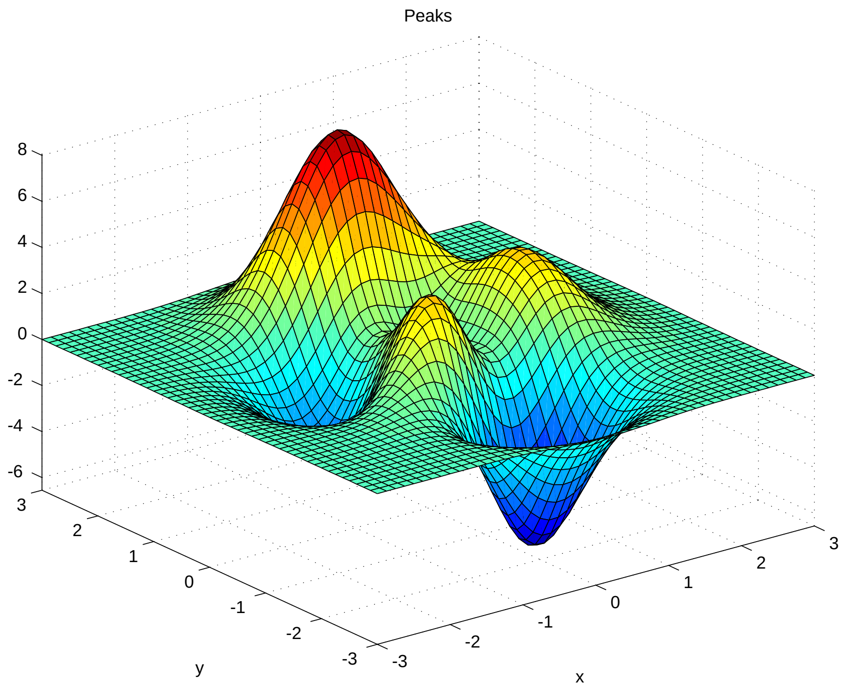
Extrapolation



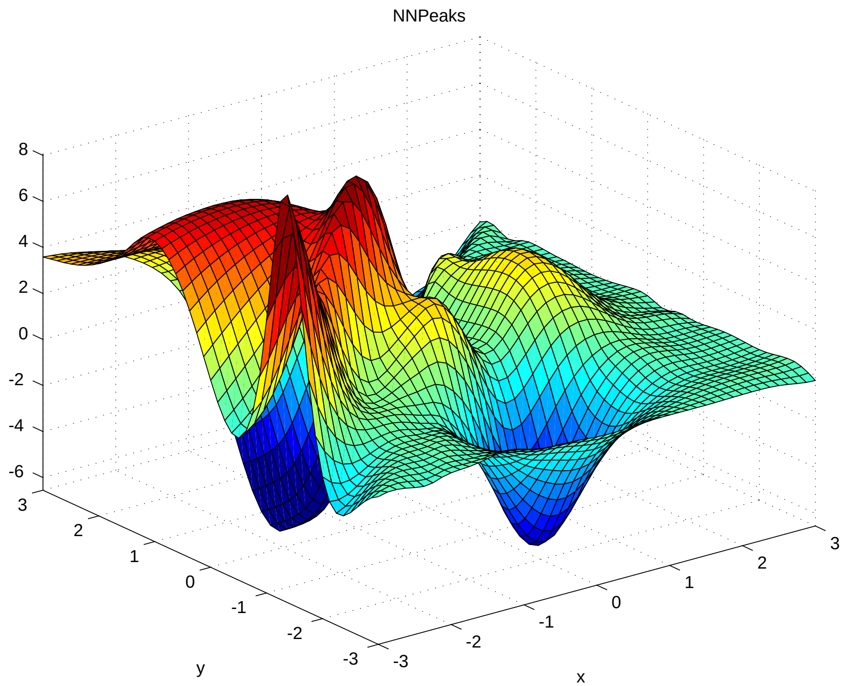
Two-Input Example

Extrapolation

True Function



Network Model



Generalization

- The network input-output mapping is accurate for the training data and for test data never seen before.
- The network interpolates well.

Cause of Overfitting

Poor generalization is caused by using a network that is too complex (too many neurons/parameters). To have the best performance we need to find the least complex network that can represent the data (Occam's Razor).

Methods for Improving Generalization

- Pruning (removing neurons) until the performance is degraded.
- Growing (adding neurons) until the performance is adequate.
- Regularization
- Validation Methods

Regularization

Standard Performance Measure

$$F = E_D$$

Performance Measure with Regularization

$$F = \beta E_D + \alpha E_W$$

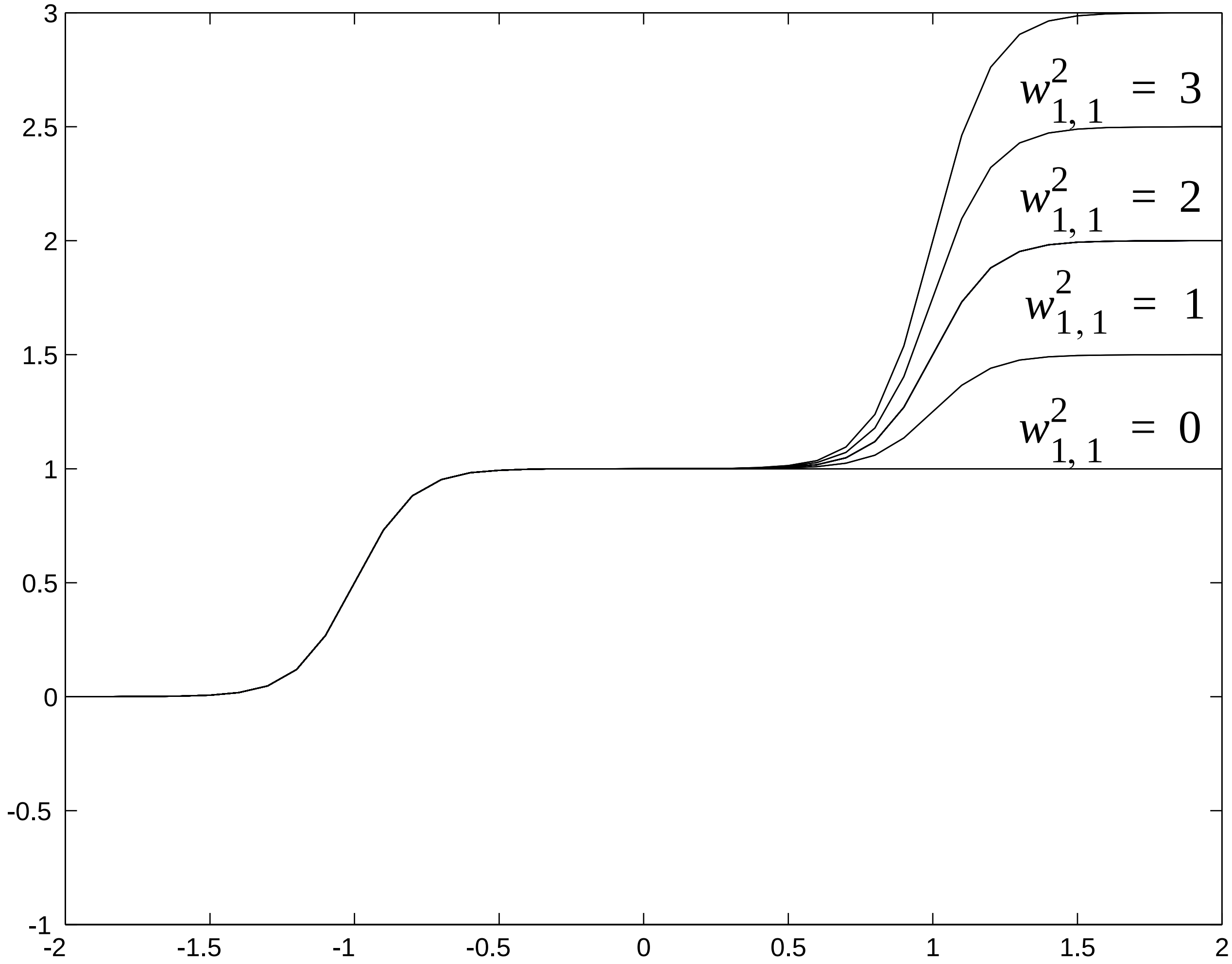
Complexity Penalty

where

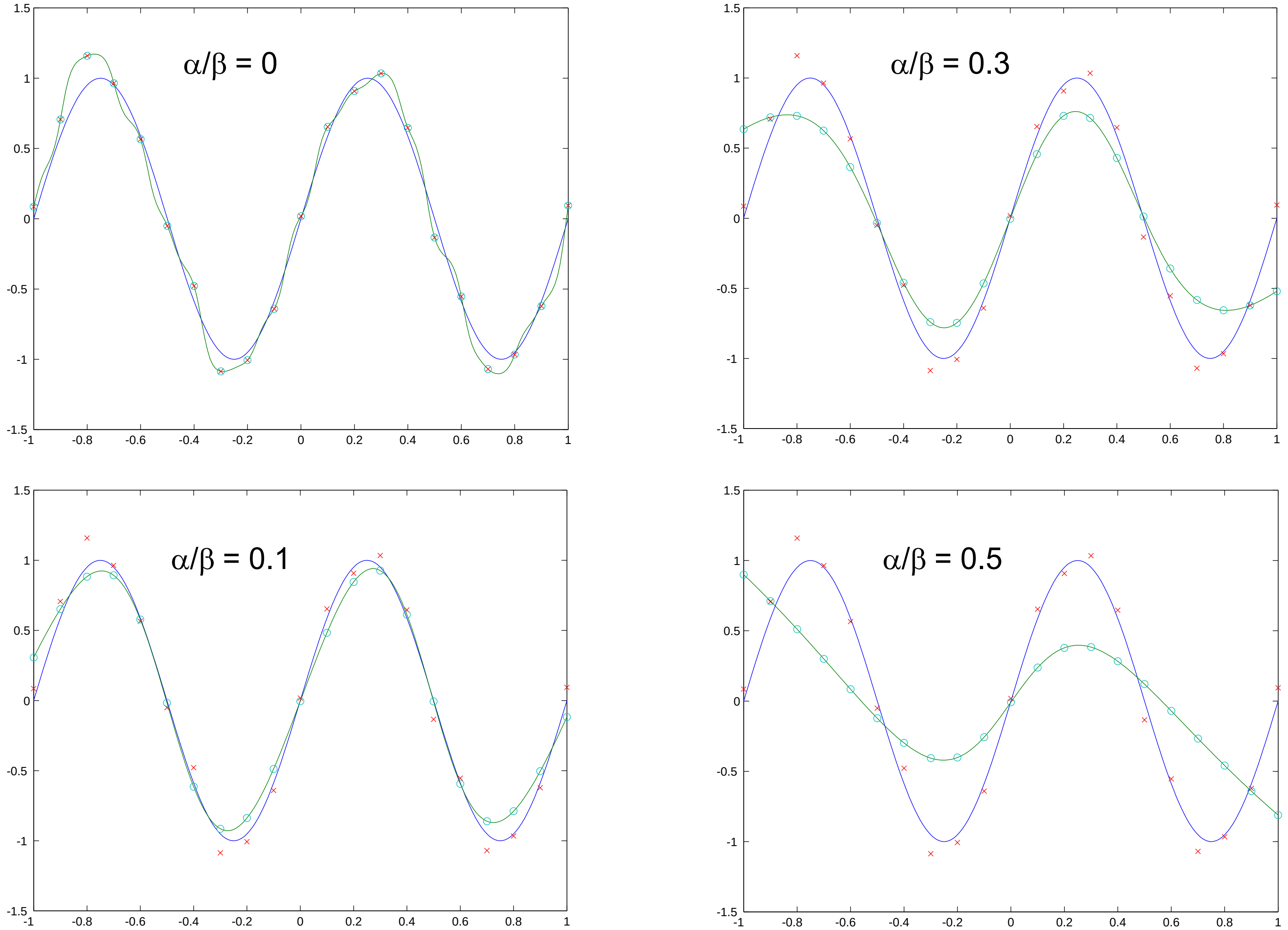
$$E_W = \sum_{i=1}^{N} w_i^2$$

(Smaller weights means a smoother function.)

Effect of Weight Changes



Effect of Regularization



NN Bayesian Framework

(MacKay 92)

$$\underbrace{P(\mathbf{w} \mid D, \alpha, \beta, M)}_{\text{Posterior (MP)}} = \frac{\underbrace{P(D \mid \mathbf{w}, \beta, M)}_{\text{Likelihood (ML)}}\,\underbrace{P(\mathbf{w} \mid \alpha, M)}_{\text{Prior}}}{\underbrace{P(D \mid \alpha, \beta, M)}_{\text{Normalization (Evidence)}}}$$

D - Data Set

M - Neural Network Model

$\mathbf{w}$ - Vector of Network Weights

Gaussian Assumptions

Gaussian Noise

$$P(D|\mathbf{w}, \beta, M) = \frac{1}{Z_D(\beta)} \exp(-\beta E_D) \qquad Z_D(\beta) = (\pi / \beta)^{n/2}$$

Gaussian Prior:

$$P(\mathbf{w}|\alpha, M) = \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W) \qquad Z_W(\alpha) = (\pi / \alpha)^{N/2}$$

$$P(\mathbf{w}|D, \alpha, \beta, M) = \frac{\frac{1}{Z_W(\alpha)} \frac{1}{Z_D(\beta)} \exp(-(\beta E_D + \alpha E_W))}{\text{Normalization Factor}}$$

$$= \frac{1}{Z_F(\alpha, \beta)} \exp(-F(\mathbf{w}))$$

Minimize F to Maximize P.

Optimizing Regularization Parameters

Evidence from First Level

Second Level of Inference

$$P(\alpha, \beta | D, M) = \frac{P(D | \alpha, \beta, M) P(\alpha, \beta | M)}{P(D | M)}$$

Evidence:

$$P(D | \alpha, \beta, M) = \frac{P(D | \mathbf{w}, \beta, M) P(\mathbf{w} | \alpha, M)}{P(\mathbf{w} | D, \alpha, \beta, M)}$$

$$= \frac{\left[\frac{1}{Z_D(\beta)} \exp(-\beta E_D)\right]\left[\frac{1}{Z_W(\alpha)} \exp(-\alpha E_W)\right]}{\frac{1}{Z_F(\alpha, \beta)} \exp(-F(\mathbf{w}))}$$

$$= \frac{Z_F(\alpha, \beta)}{Z_D(\beta) Z_W(\alpha)} \cdot \frac{\exp(-\beta E_D - \alpha E_W)}{\exp(-F(\mathbf{w}))} = \frac{Z_F(\alpha, \beta)}{Z_D(\beta) Z_W(\alpha)}$$

Quadratic Approximation

The only unknown term in the evidence is Z_F. It can be approximated using a second order Taylor series expansion.

$$Z_F \approx (2\pi)^{N/2} \left(\det\left((\mathbf{H}^{\mathrm{MP}})^{-1}\right)\right)^{1/2} \exp\left(-F(\mathbf{w}^{\mathrm{MP}})\right)$$

Hessian Matrix

$$\mathbf{H} = \beta \nabla^2 E_D + \alpha \nabla^2 E_W$$

Optimum Parameters

If we make this substitution for Z_F in the expression for the evidence and then take the derivative with respect to α and β to locate the minimum we find:

$$\alpha^{\mathrm{MP}} = \frac{\gamma}{2E_W(\mathbf{w}^{\mathrm{MP}})} \qquad \beta^{\mathrm{MP}} = \frac{n-\gamma}{2E_D(\mathbf{w}^{\mathrm{MP}})}$$

Effective Number of Parameters

$$\gamma = N - 2\alpha^{\mathrm{MP}}\mathrm{tr}(\mathbf{H}^{\mathrm{MP}})^{-1}$$

Gauss-Newton Approximation

It can be expensive to compute the Hessian matrix.

Try the Gauss-Newton Approximation.

$$\mathbf{H} = \nabla^2 F(\mathbf{w}) \approx 2\beta \mathbf{J}^T \mathbf{J} + 2\alpha \mathbf{I}_N$$

This is readily available if the Levenberg-Marquardt algorithm is used for training.

Algorithm

0. Initialize α, β and the weights.

1. Take one step of Levenberg-Marquardt to minimize $F(\mathbf{w})$.

2. Compute the effective number of parameters $\gamma = N - 2\alpha \mathrm{tr}(\mathbf{H}^{-1})$, using the Gauss-Newton approximation for $\mathbf{H}$.

3. Compute new estimates of the regularization parameters $\alpha = \gamma/(2E_W)$ and $\beta = (n-\gamma)/(2E_D)$.

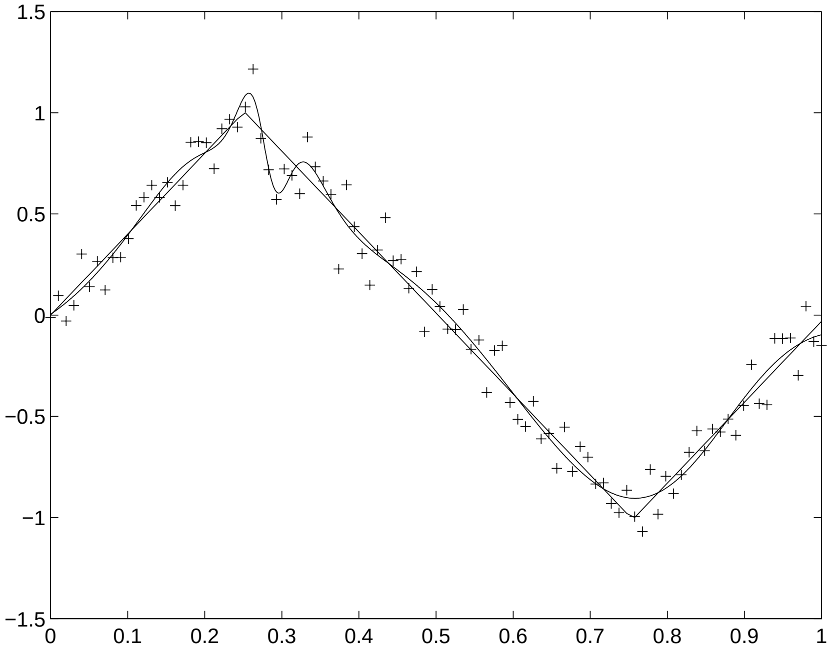
4. Iterate steps 1-3 until convergence.

Checks of Performance

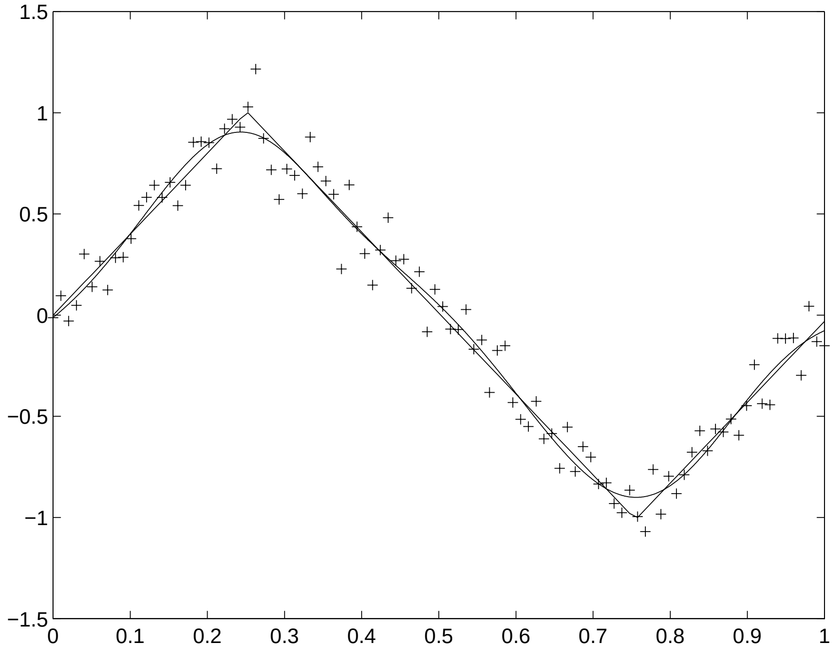
- If γ is very close to N, then the network may be too small. Add more hidden layer neurons and retrain.
- If the larger network has the same final γ, then the smaller network was large enough.
- Otherwise, increase the number of hidden neurons.
- If a network is sufficiently large, then a larger network will achieve comparable values for γ, E_D and E_W.

Simple Test Problem

1-6-1 Network
Without Regularization



1-6-1 Network
With Regularization



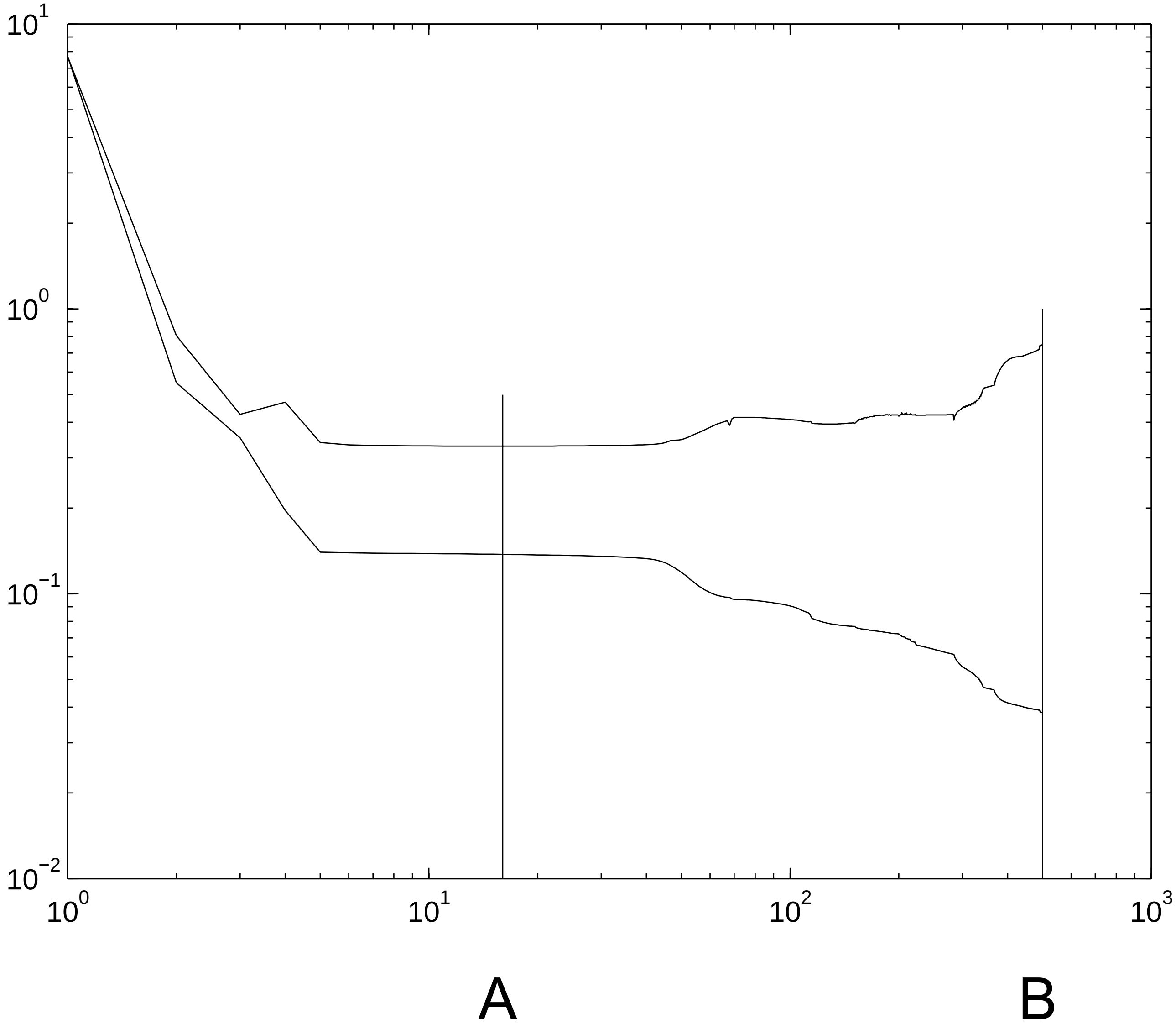
Triangle Wave Results

S	E_D	E_W	E_A	N	γ
2	1.612	203.0	0.5031	7	5.659
3	1.214	187.8	0.1954	10	8.468
4	1.144	177.0	0.1080	13	9.843
5	1.143	177.2	0.1085	16	9.906
6	1.143	177.2	0.1088	19	9.908
8	1.143	177.1	0.1091	25	9.911
10	1.142	177.1	0.1093	31	9.913
14	1.142	177.0	0.1095	43	9.915

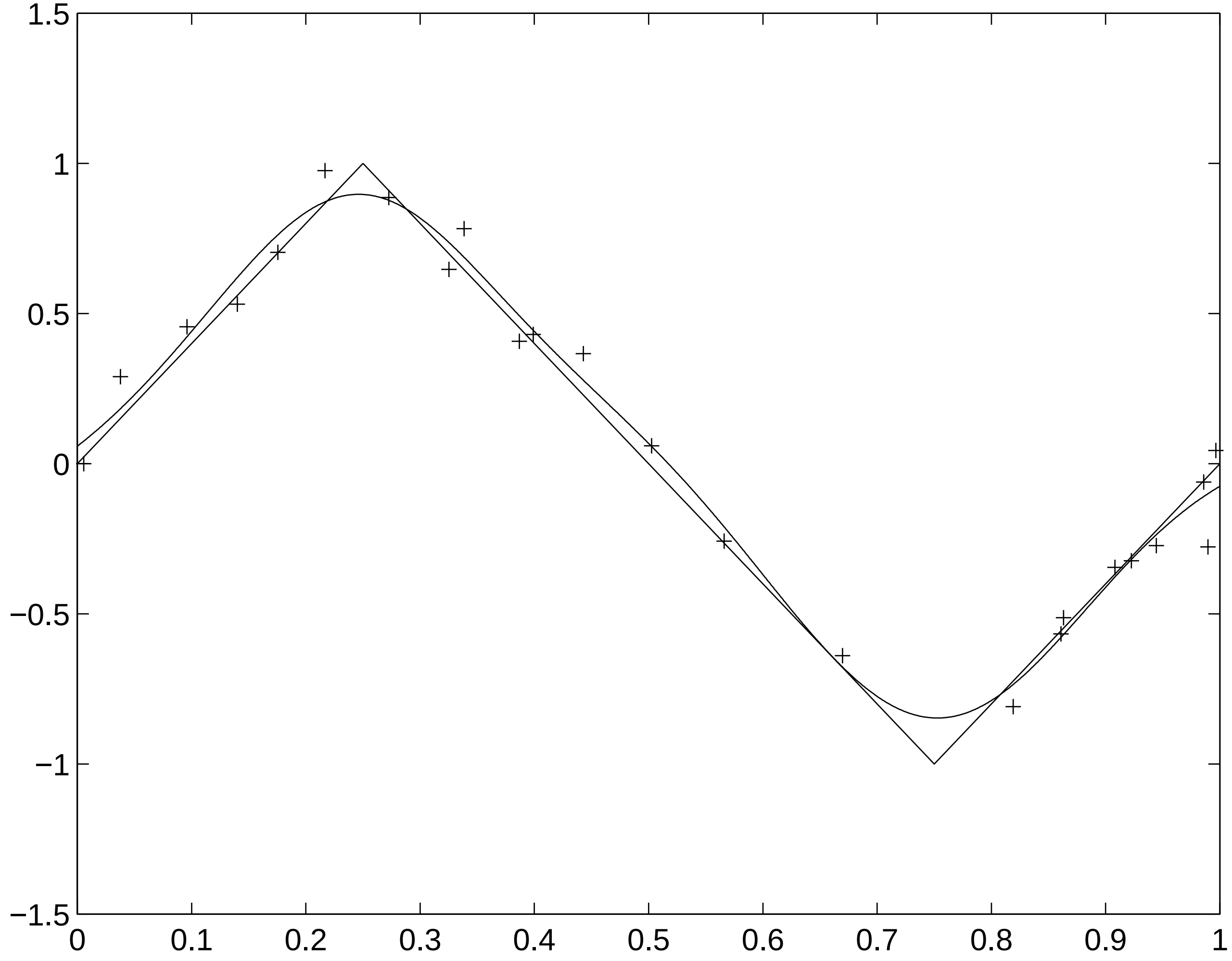
Early Stopping

- Break up data into training, *validation*, and test sets.
- Use only the training set to compute gradients and determine weight updates.
- Compute the performance on the validation set at each iteration of training.
- Stop training when the performance on the validation set goes up for a specified number of iterations.
- Use the weights which achieved the lowest error on the validation set.

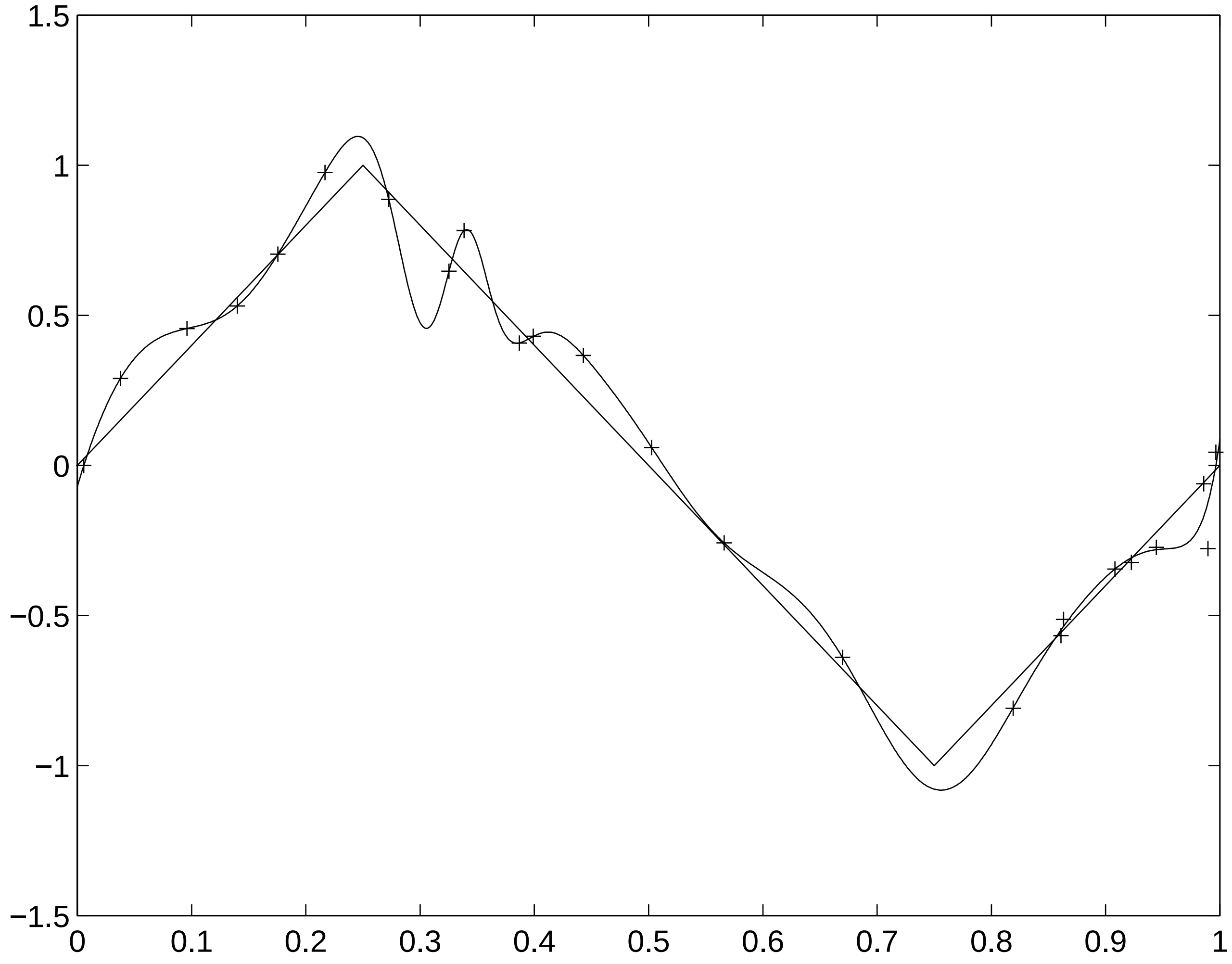
Training and Validation Error



Point A Response (Early Stopping)



Point B Response (Overfit)



Other Validation Set Uses

- Setting the regularization parameter
- Committee of networks
  - Averaging
  - Voting
- Boosting
- …

Relationship Between Early Stopping and Regularization

Quadratic Functions

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{d}^T\mathbf{x} + c \qquad \text{(Symmetric } \mathbf{A}\text{)}$$

Gradient and Hessian:

Useful properties of gradients:

$$\nabla(\mathbf{h}^T\mathbf{x}) = \nabla(\mathbf{x}^T\mathbf{h}) = \mathbf{h}$$

$$\nabla\mathbf{x}^T\mathbf{Q}\mathbf{x} = \mathbf{Q}\mathbf{x} + \mathbf{Q}^T\mathbf{x} = 2\mathbf{Q}\mathbf{x} \quad \text{(for symmetric } \mathbf{Q}\text{)}$$

Gradient of Quadratic Function:

$$\boxed{\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d}}$$

Hessian of Quadratic Function:

$$\boxed{\nabla^2 F(\mathbf{x}) = \mathbf{A}}$$

Eigensystem of the Hessian

Consider a quadratic function which has a stationary point at the origin, and whose value there is zero.

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}$$

Perform a similarity transform on the Hessian matrix, using the eigenvalues as the new basis vectors.

$$\mathbf{B} = \begin{bmatrix}\mathbf{z}_1 & \mathbf{z}_2 & \dots & \mathbf{z}_n\end{bmatrix}$$

Since the Hessian matrix is symmetric, its eigenvectors are orthogonal.

$$\mathbf{B}^{-1} = \mathbf{B}^T$$

$$\mathbf{A}' = [\mathbf{B}^T\mathbf{A}\mathbf{B}] = \begin{bmatrix}\lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \lambda_n\end{bmatrix} = \Lambda \qquad \mathbf{A} = \mathbf{B}\Lambda\mathbf{B}^T$$

Second Directional Derivative

$$\frac{\mathbf{p}^T \nabla^2 F(\mathbf{x}) \mathbf{p}}{\|\mathbf{p}\|^2} = \frac{\mathbf{p}^T \mathbf{A} \mathbf{p}}{\|\mathbf{p}\|^2}$$

Represent $\mathbf{p}$ with respect to the eigenvectors (new basis):

$$\mathbf{p} = \mathbf{B}\mathbf{c}$$

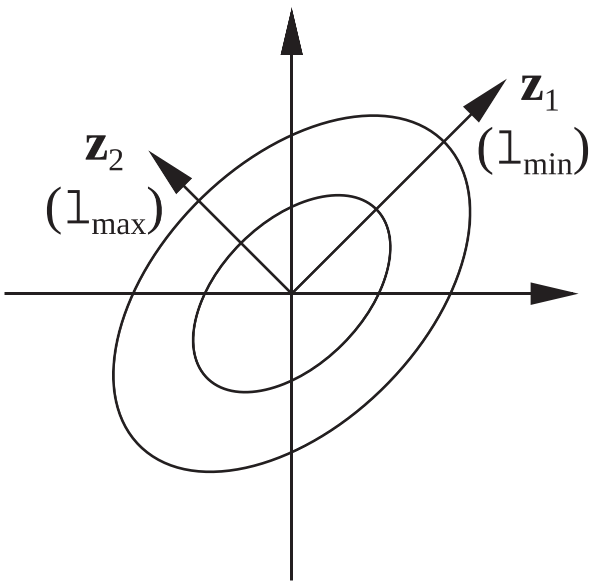
$$\frac{\mathbf{p}^T \mathbf{A} \mathbf{p}}{\|\mathbf{p}\|^2} = \frac{\mathbf{c}^T \mathbf{B}^T (\mathbf{B} \Lambda \mathbf{B}^T) \mathbf{B} \mathbf{c}}{\mathbf{c}^T \mathbf{B}^T \mathbf{B} \mathbf{c}} = \frac{\mathbf{c}^T \Lambda \mathbf{c}}{\mathbf{c}^T \mathbf{c}} = \frac{\sum_{i=1}^{n} \lambda_i c_i^2}{\sum_{i=1}^{n} c_i^2}$$

$$\lambda_{min} \le \frac{\mathbf{p}^T \mathbf{A} \mathbf{p}}{\|\mathbf{p}\|^2} \le \lambda_{max}$$

Eigenvector (Largest Eigenvalue)

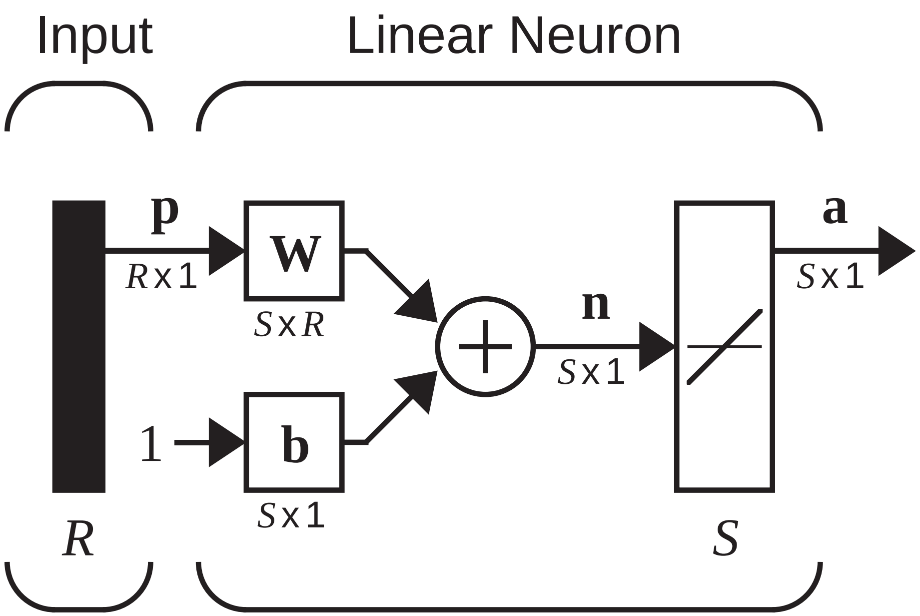
$$\mathbf{p} = \mathbf{z}_{max} \qquad\qquad \mathbf{c} = \mathbf{B}^T\mathbf{p} = \mathbf{B}^T\mathbf{z}_{max} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\frac{\mathbf{z}_{max}{}^T\mathbf{A}\mathbf{z}_{max}}{\|\mathbf{z}_{max}\|^2} = \frac{\sum_{i=1}^{n} \lambda_i c_i^2}{\sum_{i=1}^{n} c_i^2} = \lambda_{max}$$



The eigenvalues represent curvature (second derivatives) along the eigenvectors (the principal axes).

Linear Network



$$\mathbf{a} = \mathbf{purelin}(\mathbf{Wp}+\mathbf{b})$$

$$\mathbf{a} = \mathbf{purelin}(\mathbf{Wp}+\mathbf{b}) = \mathbf{Wp}+\mathbf{b}$$

$$a_i = purelin(n_i) = purelin({}_i\mathbf{w}^T\mathbf{p} + b_i) = {}_i\mathbf{w}^T\mathbf{p} + b_i$$

$${}_i\mathbf{w} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix}$$

Performance Index

Training Set:

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

Input: $\mathbf{p}_q$ Target: $\mathbf{t}_q$

Notation:

$$\mathbf{x} = \begin{bmatrix} {}_1\mathbf{w} \\ b \end{bmatrix} \qquad \mathbf{z} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} \qquad a = {}_1\mathbf{w}^T\mathbf{p} + b \Rightarrow a = \mathbf{x}^T\mathbf{z}$$

Mean Square Error:

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2] = E[(t-\mathbf{x}^T\mathbf{z})^2] = E_D$$

Error Analysis

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2] = E[(t-\mathbf{x}^T\mathbf{z})^2]$$

$$F(\mathbf{x}) = E[t^2 - 2t\mathbf{x}^T\mathbf{z} + \mathbf{x}^T\mathbf{z}\mathbf{z}^T\mathbf{x}]$$

$$F(\mathbf{x}) = E[t^2] - 2\mathbf{x}^T \mathrm{E}[t\mathbf{z}] + \mathbf{x}^T E[\mathbf{z}\mathbf{z}^T]\mathbf{x}$$

$$\boxed{F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x}}$$

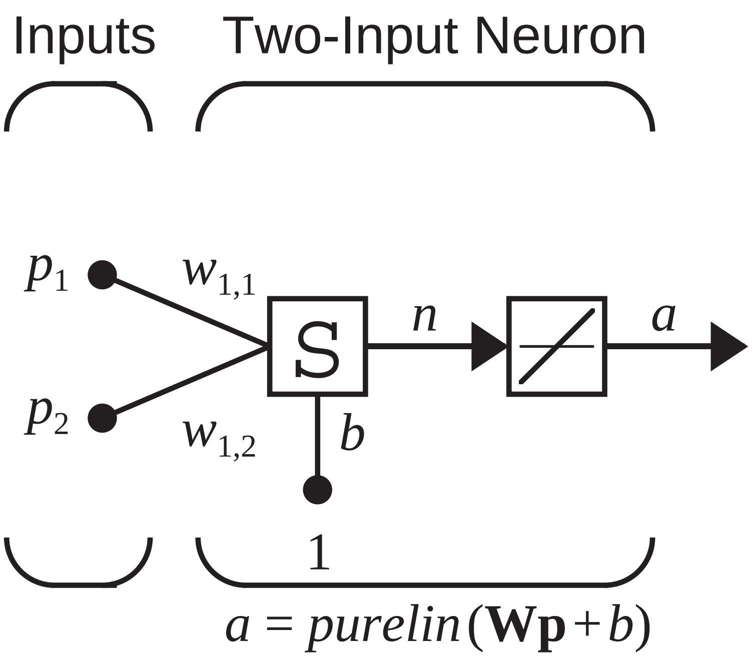
$$c = E[t^2] \qquad \mathbf{h} = E[t\mathbf{z}] \qquad \mathbf{R} = E[\mathbf{z}\mathbf{z}^T]$$

The mean square error for the Linear Network is a quadratic function:

$$F(\mathbf{x}) = c + \mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}$$

$$\mathbf{d} = -2\mathbf{h} \qquad \mathbf{A} = 2\mathbf{R}$$

Example



$$\left\{\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_1 = 1\right\} \quad \text{(Probability = 0.75)}$$

$$\left\{\mathbf{p}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, t_2 = -1\right\} \quad \text{(Probability = 0.25)}$$

$$F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x} = E_D$$

$$c = E[t^2] = (1)^2(0.75) + (-1)^2(0.25) = 1$$

$$\mathbf{h} = E[t\mathbf{z}] = (0.75)(1)\begin{bmatrix} 1 \\ 1 \end{bmatrix} + (0.25)(-1)\begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$

$$\mathbf{R} = E[\mathbf{z}\mathbf{z}^T] = \mathbf{p}_1\mathbf{p}_1^T(0.75) + \mathbf{p}_2\mathbf{p}_2^T(0.25)$$

$$= 0.75\begin{bmatrix} 1 \\ 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \end{bmatrix} + 0.25\begin{bmatrix} -1 \\ 1 \end{bmatrix}\begin{bmatrix} -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

Performance Contour

Optimum Point (Maximum Likelihood)

$$\mathbf{x}^{ML} = -\mathbf{A}^{-1}\mathbf{d} = \mathbf{R}^{-1}\mathbf{h} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Hessian Matrix

$$\nabla^2 F(\mathbf{x}) = \mathbf{A} = 2\mathbf{R} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

Eigenvalues

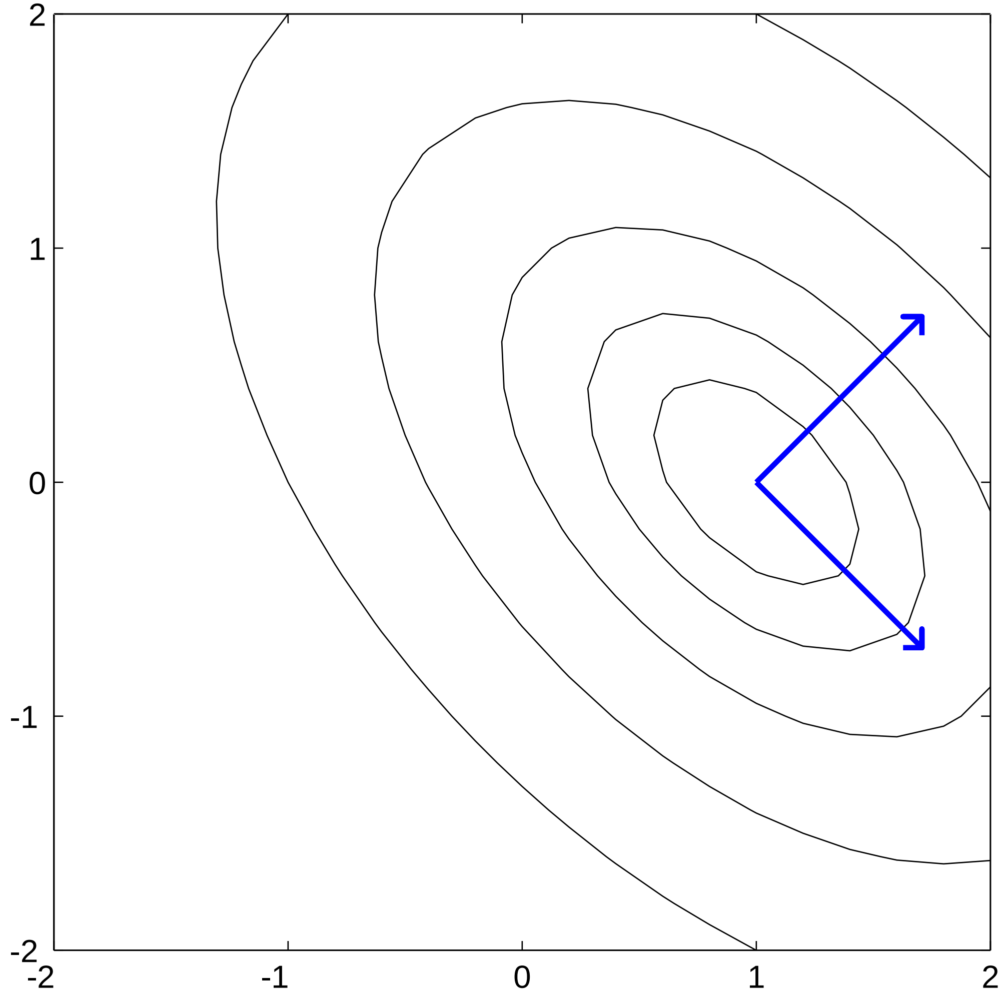
$$\left| \mathbf{A} - \lambda \mathbf{I} \right| = \begin{vmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{vmatrix} = \lambda^2 - 4\lambda + 3 = (\lambda - 1)(\lambda - 3) \Rightarrow \lambda_1 = 1, \qquad \lambda_2 = 3$$

Eigenvectors

$$\left[\mathbf{A} - \lambda \mathbf{I}\right]\mathbf{v} = 0$$

$$\lambda_1 = 1 \qquad \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\mathbf{v}_1 = 0 \qquad \mathbf{v}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad \lambda_2 = 3 \qquad \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}\mathbf{v}_2 = 0 \qquad \mathbf{v}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Contour Plot of E_D



$$\gamma = N - 2\alpha^{\mathrm{MP}} \mathrm{tr}(\mathbf{H}^{\mathrm{MP}})^{-1}$$

Steepest Descent Trajectory

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha\mathbf{g}_k = \mathbf{x}_k - \alpha(\mathbf{A}\mathbf{x}_k + \mathbf{d})$$

$$= \mathbf{x}_k - \alpha\mathbf{A}(\mathbf{x}_k + \mathbf{A}^{-1}\mathbf{d}) = \mathbf{x}_k - \alpha\mathbf{A}(\mathbf{x}_k - \mathbf{x}^{ML})$$

$$= [\mathbf{I} - \alpha\mathbf{A}]\mathbf{x}_k + \alpha\mathbf{A}\mathbf{x}^{ML} = \mathbf{M}\mathbf{x}_k + [\mathbf{I} - \mathbf{M}]\mathbf{x}^{ML}$$

$$\boxed{\mathbf{M} = [\mathbf{I} - \alpha\mathbf{A}]}$$

$$\mathbf{x}_1 = \mathbf{M}\mathbf{x}_0 + [\mathbf{I} - \mathbf{M}]\mathbf{x}^{ML}$$

$$\mathbf{x}_2 = \mathbf{M}\mathbf{x}_1 + [\mathbf{I} - \mathbf{M}]\mathbf{x}^{ML}$$

$$= \mathbf{M}^2\mathbf{x}_0 + \mathbf{M}[\mathbf{I} - \mathbf{M}]\mathbf{x}^{ML} + [\mathbf{I} - \mathbf{M}]\mathbf{x}^{ML}$$

$$= \mathbf{M}^2\mathbf{x}_0 + \mathbf{M}\mathbf{x}^{ML} - \mathbf{M}^2\mathbf{x}^{ML} + \mathbf{x}^{ML} - \mathbf{M}\mathbf{x}^{ML}$$

$$= \mathbf{M}^2\mathbf{x}_0 + \mathbf{x}^{ML} - \mathbf{M}^2\mathbf{x}^{ML} = \mathbf{M}^2\mathbf{x}_0 + [\mathbf{I} - \mathbf{M}^2]\mathbf{x}^{ML}$$

$$\boxed{\mathbf{x}_k = \mathbf{M}^k\mathbf{x}_0 + [\mathbf{I} - \mathbf{M}^k]\mathbf{x}^{ML}}$$

Regularization

$$F(\mathbf{x}) = E_D + \gamma E_W \qquad (\gamma = \alpha/\beta)$$

$$E_W = \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T(\mathbf{x} - \mathbf{x}_0)$$

To locate the minimum point, set the gradient to zero.

$$\nabla F(\mathbf{x}) = \nabla E_D + \gamma \nabla E_W$$

$$\nabla E_W = (\mathbf{x} - \mathbf{x}_0) \qquad \nabla E_D = \mathbf{A}(\mathbf{x} - \mathbf{x}^{ML})$$

$$\nabla F(\mathbf{x}) = \mathbf{A}(\mathbf{x} - \mathbf{x}^{ML}) + \gamma(\mathbf{x} - \mathbf{x}_0) = \mathbf{0}$$

MAP – ML

$$\mathbf{A}(\mathbf{x}^{MAP} - \mathbf{x}^{ML}) = -\gamma(\mathbf{x}^{MAP} - \mathbf{x}_0) = -\gamma(\mathbf{x}^{MAP} - \mathbf{x}^{ML} + \mathbf{x}^{ML} - \mathbf{x}_0)$$

$$= -\gamma(\mathbf{x}^{MAP} - \mathbf{x}^{ML}) - \gamma(\mathbf{x}^{ML} - \mathbf{x}_0)$$

$$(\mathbf{A} + \gamma\mathbf{I})(\mathbf{x}^{MAP} - \mathbf{x}^{ML}) = \gamma(\mathbf{x}_0 - \mathbf{x}^{ML})$$

$$(\mathbf{x}^{MAP} - \mathbf{x}^{ML}) = \gamma(\mathbf{A} + \gamma\mathbf{I})^{-1}(\mathbf{x}_0 - \mathbf{x}^{ML})$$

$$\mathbf{x}^{MAP} = \mathbf{x}^{ML} - \gamma(\mathbf{A} + \gamma\mathbf{I})^{-1}\mathbf{x}^{ML} + \gamma(\mathbf{A} + \gamma\mathbf{I})^{-1}\mathbf{x}_0 = \mathbf{x}^{ML} - \mathbf{M}_\gamma\mathbf{x}^{ML} + \mathbf{M}_\gamma\mathbf{x}_0$$

$$\mathbf{M}_\gamma = \gamma(\mathbf{A} + \gamma\mathbf{I})^{-1}$$

$$\boxed{\mathbf{x}^{MAP} = \mathbf{M}_\gamma\mathbf{x}_0 + [\mathbf{I} - \mathbf{M}_\gamma]\mathbf{x}^{ML}}$$

Early Stopping – Regularization

$$\mathbf{x}_k = \mathbf{M}^k \mathbf{x}_0 + [\mathbf{I} - \mathbf{M}^k]\mathbf{x}^{ML}$$

$$\mathbf{x}^{MAP} = \mathbf{M}_\gamma \mathbf{x}_0 + [\mathbf{I} - \mathbf{M}_\gamma]\mathbf{x}^{ML}$$

$$\mathbf{M} = [\mathbf{I} - \alpha\mathbf{A}]$$

$$\mathbf{M}_\gamma = \gamma(\mathbf{A} + \gamma\mathbf{I})^{-1}$$

Eigenvalues of $\mathbf{M}^k$:

$$[\mathbf{I} - \alpha\mathbf{A}]\mathbf{z}_i = \mathbf{z}_i - \alpha\mathbf{A}\mathbf{z}_i = \mathbf{z}_i - \alpha\lambda_i\mathbf{z}_i = \underbrace{(1 - \alpha\lambda_i)}_{\text{Eigenvalues of } \mathbf{M}}\mathbf{z}_i$$

$\mathbf{z}_i$ - eigenvector of $\mathbf{A}$

λ_i - eigenvalue of $\mathbf{A}$

$$\mathrm{eig}(\mathbf{M}^k) = (1 - \alpha\lambda_i)^k$$

Eigenvalues of $\mathbf{M}_\gamma$:

$$\mathrm{eig}(\mathbf{M}_\gamma) = \frac{\gamma}{(\lambda_\mathrm{i} + \gamma)}$$

Reg. Parameter – Iteration Number

$\mathbf{M}^k$ and $\mathbf{M}_\gamma$ have the same eigenvectors. They would be equal if their eigenvalues were equal.

$$\frac{\gamma}{(\lambda_i + \gamma)} = (1 - \alpha\lambda_i)^k \qquad \text{Taking log :} \qquad -\log\left(1 + \frac{\lambda_i}{\gamma}\right) = k\log(1 - \alpha\lambda_i)$$

Since these are equal at $\lambda_i = 0$, they are always equal if the slopes are equal.

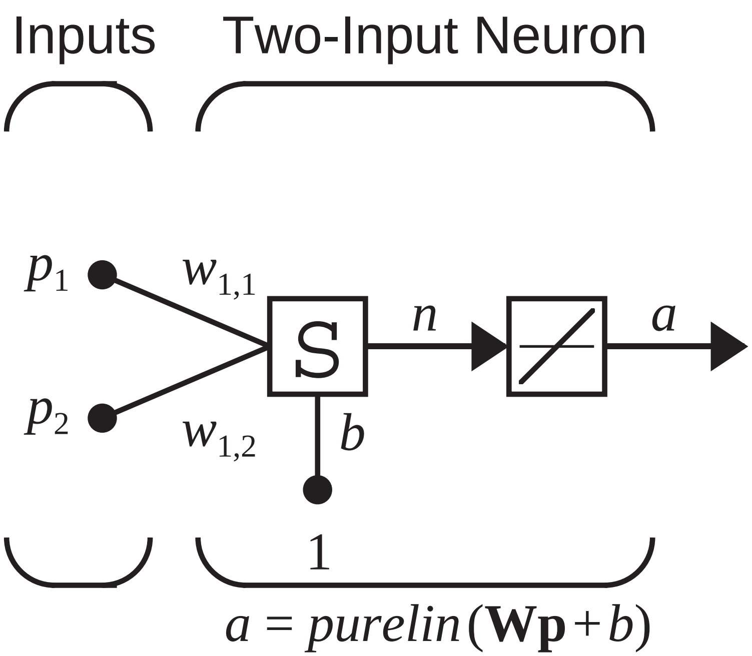
$$-\frac{1}{\left(1 + \frac{\lambda_i}{\gamma}\right)}\frac{1}{\gamma} = \frac{k}{1 - \alpha\lambda_i}(-\alpha) \quad \Longrightarrow \quad \alpha k = \frac{1}{\gamma}\,\frac{(1 - \alpha\lambda_i)}{(1 + \lambda_i/\gamma)}$$

If $\alpha\lambda_i$ and λ_i/γ are small, then:

$$\boxed{\alpha k \cong \frac{1}{\gamma}}$$

(Increasing the number of iterations is equivalent to decreasing the regularization parameter!)

Example



$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\1\end{bmatrix}, t_1 = 1\right\}$$ (Probability = 0.75)

$$\left\{\mathbf{p}_2 = \begin{bmatrix}-1\\1\end{bmatrix}, t_2 = -1\right\}$$ (Probability = 0.25)

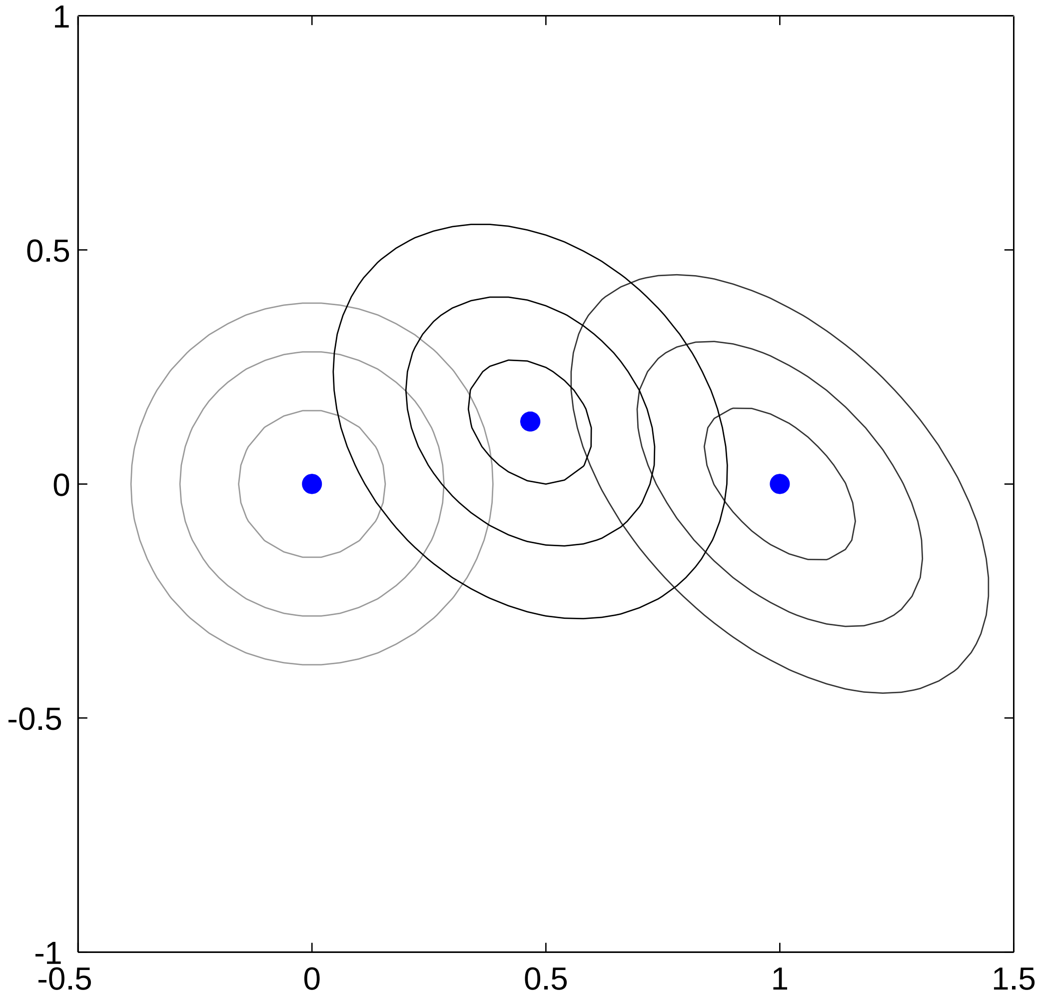
$$F(\mathbf{x}) = E_D + \gamma E_W$$

$$E_D = c + \mathbf{x}^T\mathbf{d} + \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}$$

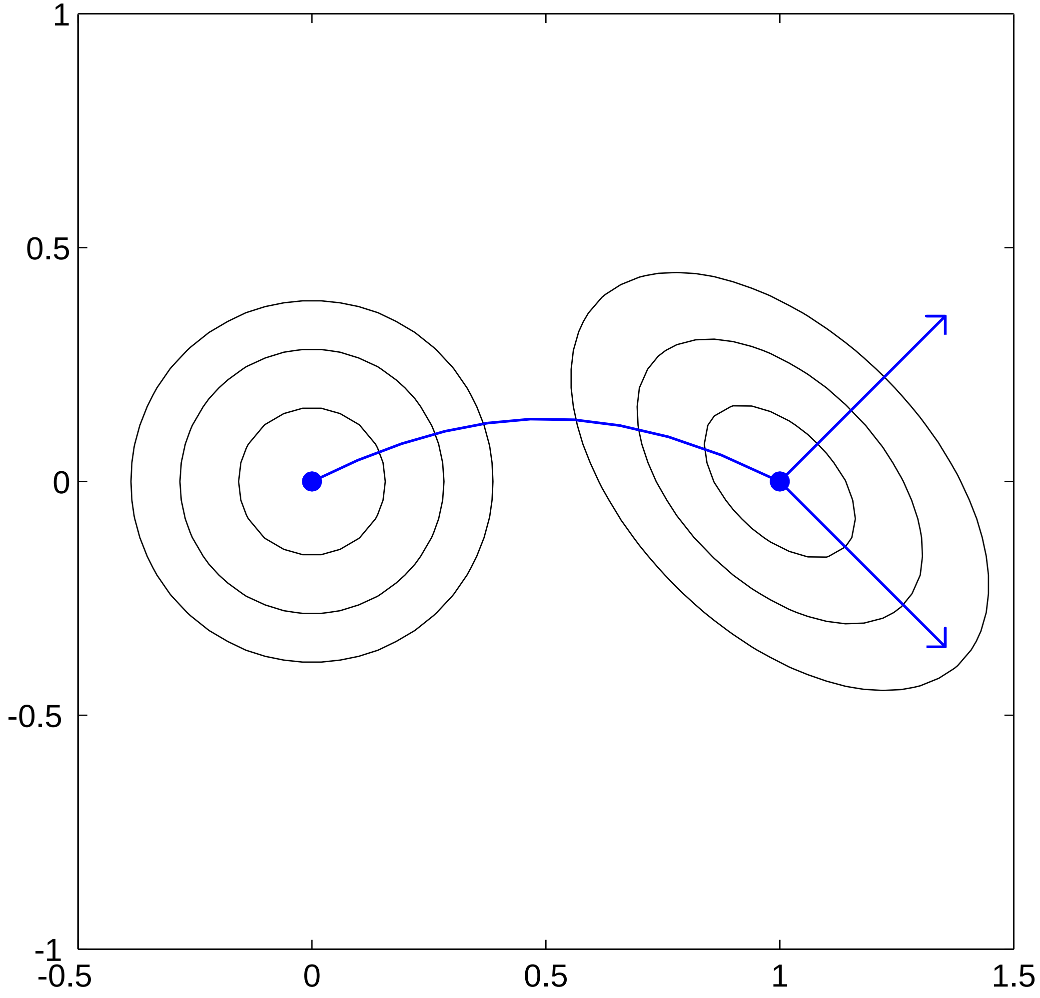
$$E_W = \frac{1}{2}\mathbf{x}^T\mathbf{x} \qquad c = 1 \qquad \mathbf{d} = -2\mathbf{h} = \begin{bmatrix}-2\\-1\end{bmatrix} \qquad \mathbf{A} = 2\mathbf{R} = \begin{bmatrix}2 & 1\\1 & 2\end{bmatrix}$$

$$\nabla^2 F(\mathbf{x}) = \nabla^2 E_D + \gamma\nabla^2 E_W = \begin{bmatrix}2 & 1\\1 & 2\end{bmatrix} + \gamma\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} = \begin{bmatrix}2+\gamma & 1\\1 & 2+\gamma\end{bmatrix}$$

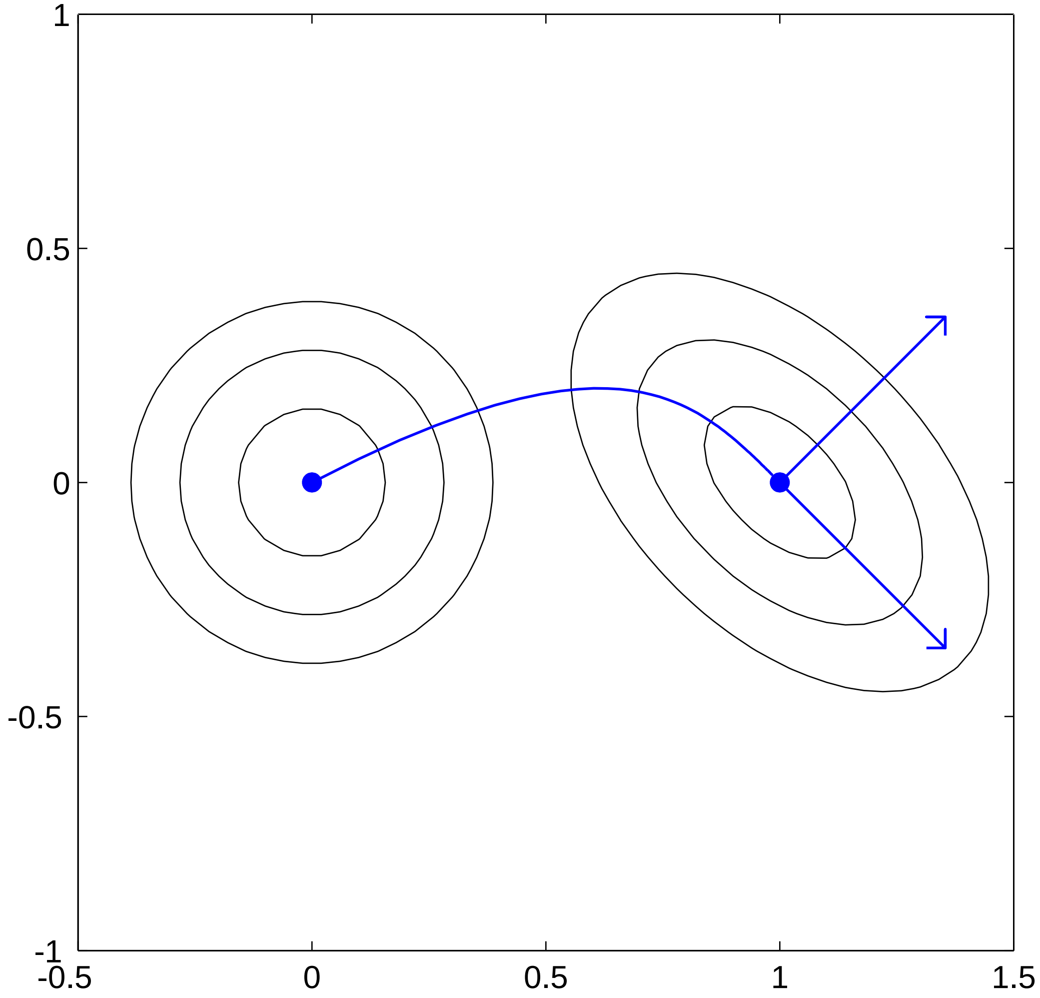
$\gamma = 0, 2, \infty$



$\gamma = 0 \to \infty$



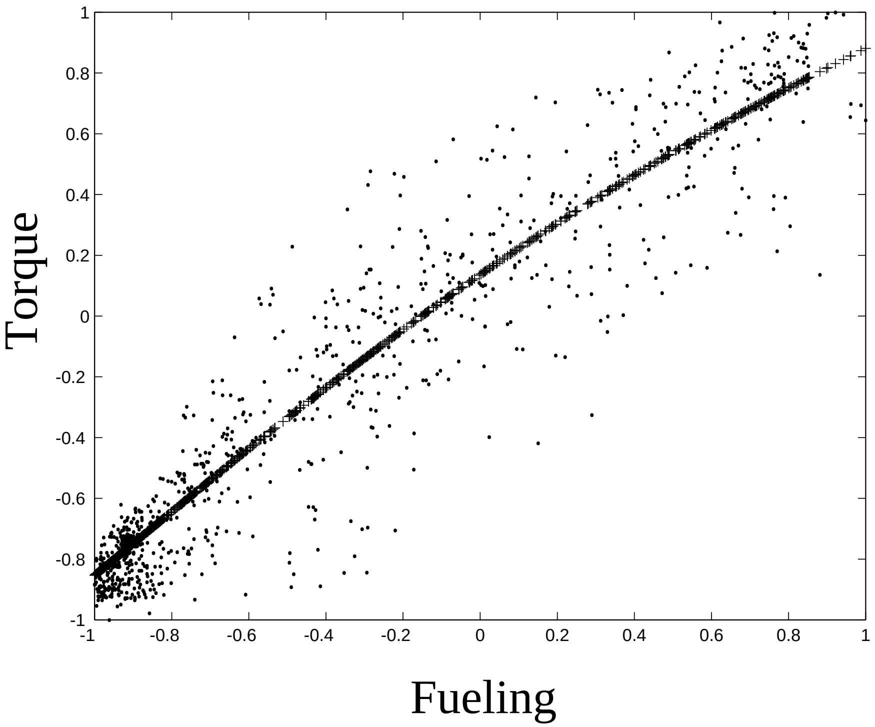
Steepest Descent Path



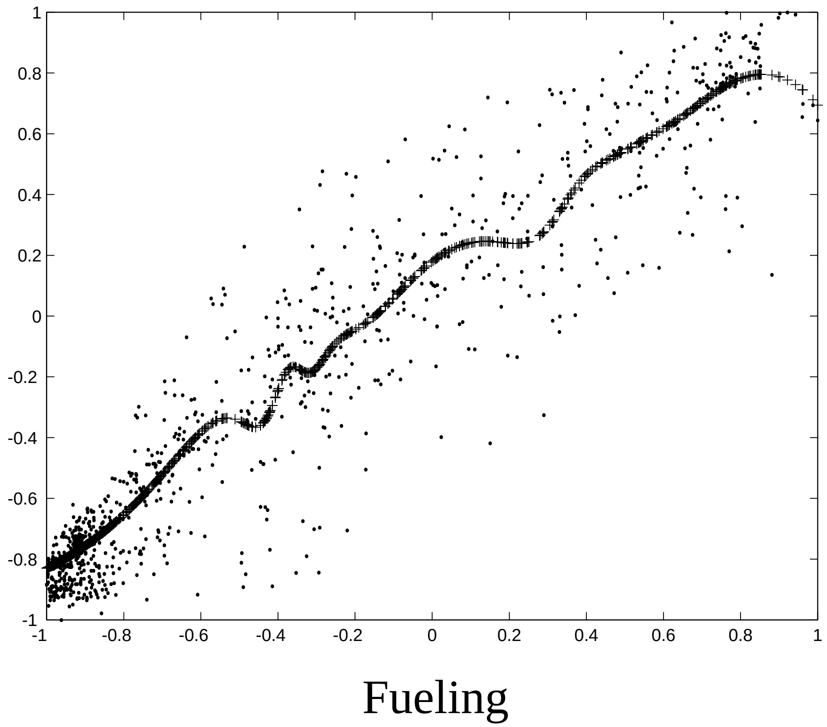
Engine Fueling-Torque

1-10-1 Network

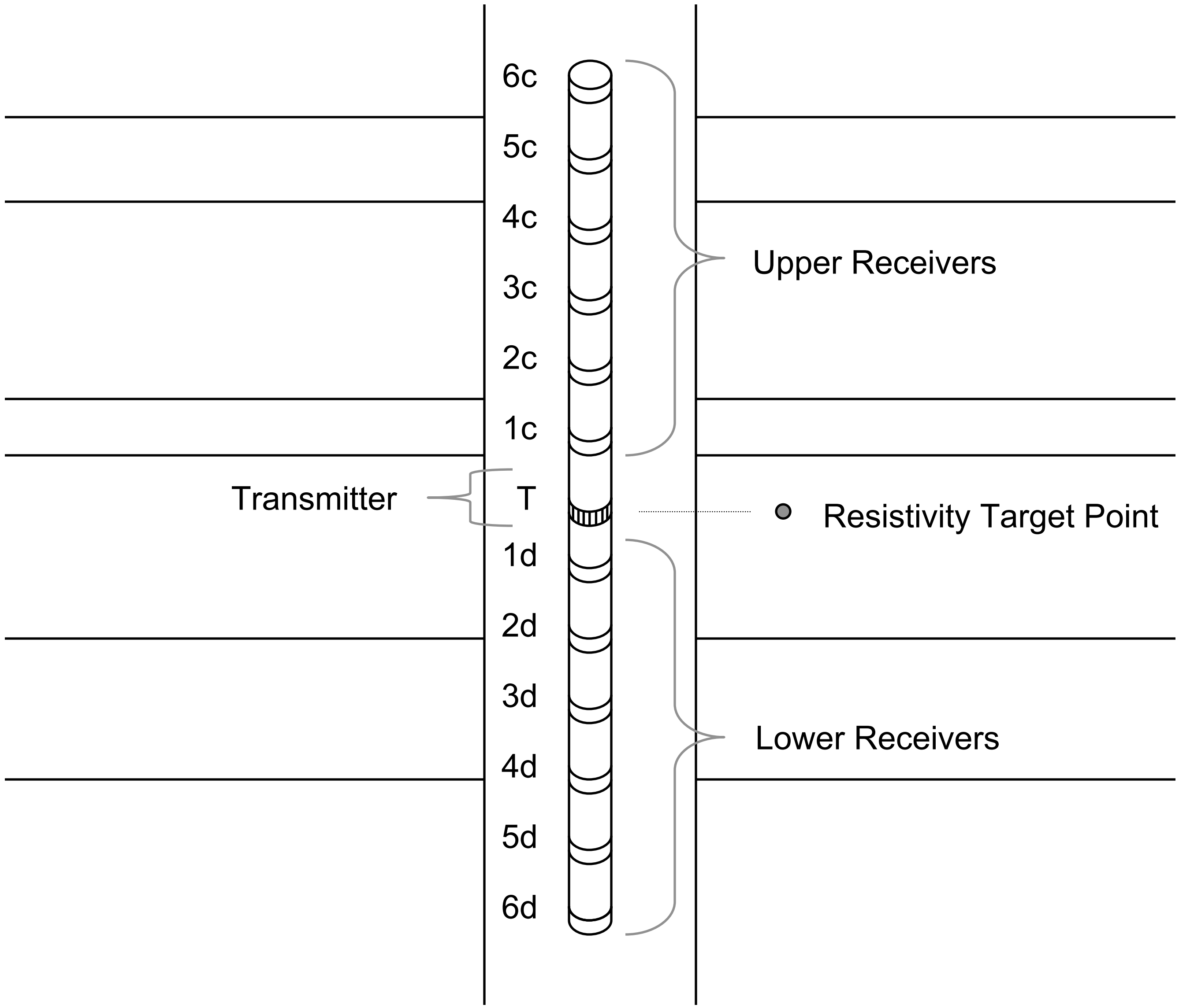
Using Bayesian Regularization



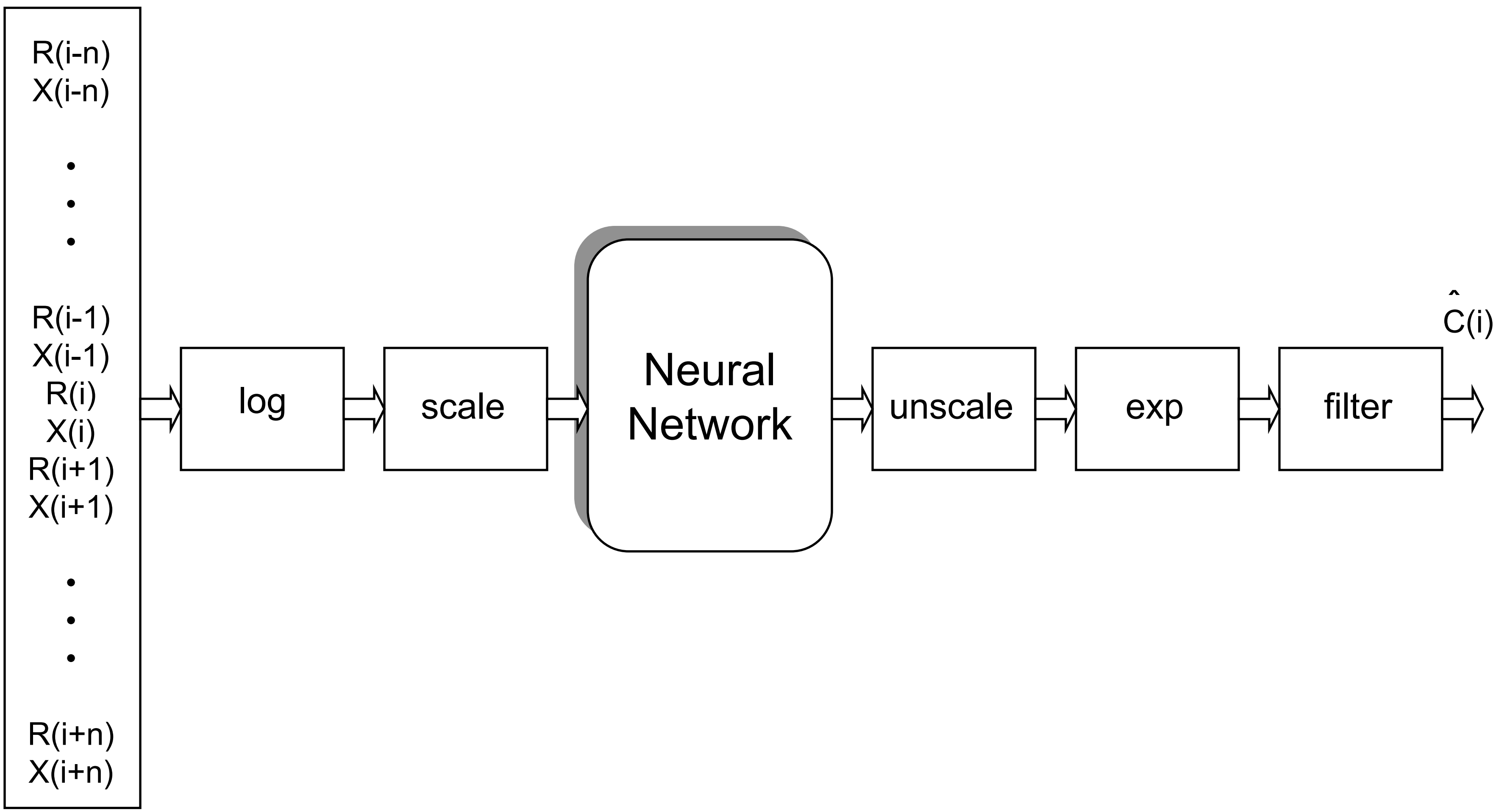
Using Early Stopping



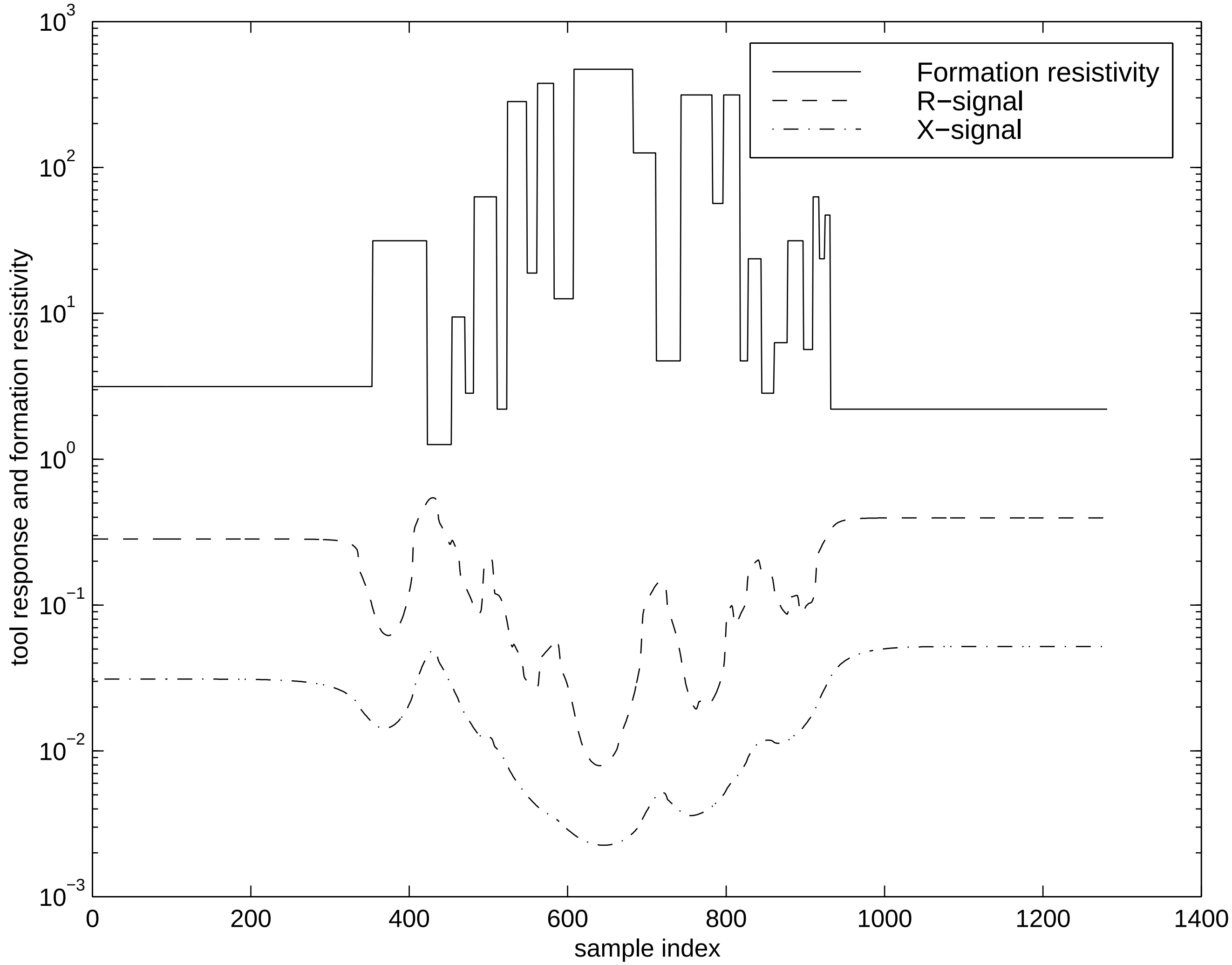
Induction Tool



Typical Filter Structure

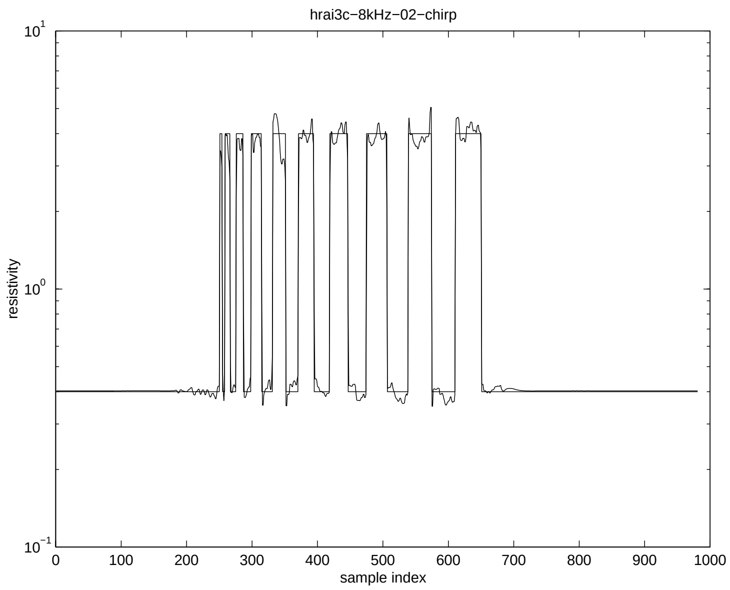


Typical Tool Response

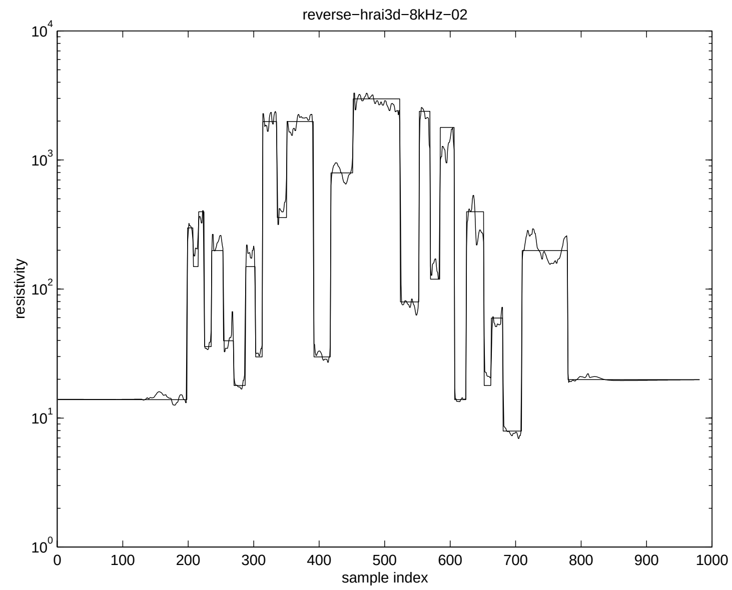


Prediction Results

D2 Testing Subset



RD1 Testing Subset



Summary

- Regularization improves generalization.
- Bayesian framework is attractive.
- Regularization and early stopping are approximately equivalent processes.
- When using early stopping a relatively slow training algorithm should be used.
- Increasing the length of training is equivalent to reducing the regularization.
- The effective number of parameters increases during the training process.
- Regularization usually produces a “smoother” function.

References

D. J. C. MacKay, “Bayesian Interpolation,” Neural Computation, vol. 4, pp. 415-447, 1992.

F.D. Foresee and M. Hagan, “Gauss-Newton Approximation to Bayesian Learning,” Proceedings of the 1997 International Joint Conference on Neural Networks, Houston, Texas, June, 1997.

J. Sjöberg and L. Ljung, “Overtraining, Regularization, and Searching for Minimum with Application to Neural Networks,” Technical Report LiTH-ISY-R-1567, Department of Electrical Engineering, Linköping University, Sweden, 1994. (ftp://ftp.control.isy.liu.se/pub/Reports/1994/1567.ps.Z)

S. Amari, et. al., “Asymptotic Statistical Theory of Overtraining and Cross-Validation,” IEEE Trans. on Neural Networks, vol. 8, no. 5, 1997, pp. 985-996.