Neural Networks

Determination of the Weights

Kei Takahashi Yoan Miché

Department of Computer Science
TKK

October, 5th 2005

Outline

Why Determine Weights ?

- Data Set acquired & Model Structure selected.
- Need to refine the Model, tune it.

- Use Data Set

$$Z^N = \{[u(t), y(t)], T = 1, ..., N\}$$

to pick the best Model among the subset of models

$$y(t) = \hat{y}(t|\theta) + e(t) = g[t, \theta] + e(t)$$

$\rightarrow$Training

- Training : Determine a mapping $Z^N \rightarrow \hat{\theta}$

Criterion

- Usually a mean square error type :

$$V_N\left(\theta, Z^N\right) = \frac{1}{2N}\sum_{t=1}^{N}[y(t) - \hat{y}(t|\theta)]^2 = \frac{1}{2N}\sum_{t=1}^{N}\epsilon^2(t,\theta)$$

$\longrightarrow$ Prediction Error Method :

- Maximum likelihood estimation, estimating the noise signal $e(t)$ to be Gaussian.

Methods Overview

Objective : Determine the weights in the system as the minimizer of the criterion :

$$\hat{\theta} = \underset{\theta}{\mathtt{argmin}}\, V_N(\theta, Z^N)$$

Methods for search of a minimum :

- Gradient Method.
- Newton Method.
- Gauss Newton Method.

The Search for a Minimum : Notations and Ideas

Taylor series expansion of the criterion in θ^* (2nd order) gives :

$$V_N(\theta, Z^N) = V_N(\theta^*, Z^N) + (\theta - \theta^*)^T G(\theta^*) + \frac{1}{2}(\theta - \theta^*)^T H(\theta^*)(\theta - \theta^*)$$

where the gradient $G(\theta^*)$ is defined by

$$G(\theta^*) = \left. \frac{dV_N(\theta, Z^N)}{d\theta} \right|_{\theta=\theta^*}$$

and the second-order derivative matrix, the Hessian, by

$$H(\theta^*) = \left. \frac{d^2V_N(\theta, Z^N)}{d\theta^2} \right|_{\theta=\theta^*}$$

The Search for a Minimum : Notations and Ideas

Thus, suficient conditions for θ^* being a min of $V_N(\theta, Z^N)$ are

- Gradient equals zero, $G(\theta^*) = 0$
- Hessian matrix is positive definite, $\nu^T H(\theta^*)\nu > 0$ for all non-zero vectors ν

Iterative Search Methods are used to find a minimum, usually following the pattern

$$\theta^{(i+1)} = \theta^{(i)} + \mu^{(i)} f^{(i)}$$

with

- $f^{(i)}$ the search direction
- $\mu^{(i)}$ the step size

The Search for a Minimum : Gradient Method

- Principle : Modify the weights along the opposite direction of the gradient

$$\theta^{(i+1)} = \theta^{(i)} - \mu^{(i)} G(\theta^{(i)})$$

The Search for a Minimum : Gradient Method (2)

- Convergence : Depends on the step size $\mu^{(i)}$
  - Line Search : Rapid convergence but many network evaluations per iteration
  - Adaptively Controlled
  - Constant

  Local convergence is linear.

The Search for a Minimum : Gradient Method (3)

- Pros/Cons :
  - Slow Convergence
  - Easy to implement
  - Modest data storage requirements
  - Possible to use parallelizing

The Search for a Minimum : Newton Method

The new iterate is determined as the minimizer of a second-order expansion of the criterion around the current iterate

$$\tilde{V}_N(\theta, Z^N) = V_N(\theta^{(i)}, Z^N) + \left[\theta - \theta^{(i)}\right]^T G(\theta^{(i)})$$
$$+\frac{1}{2}\left[\theta - \theta^{(i)}\right]^T H(\theta^{(i)})\left[\theta - \theta^{(i)}\right]$$

Some work lead us to the update rule

$$\theta^{(i+1)} = \theta^{(i)} - H^{-1}(\theta^{(i)})G(\theta^{(i)})$$

The Search for a Minimum : Newton Method (2)

Based on an approximation of the criterion :
Convergence :

- Approximation valid only in the neighborhood of the current iterate.
- A too big step might bring next iterate far from the expected point.
- Can't be sure the method will converge at all.

$\longrightarrow$ Usually use a gradient method to adjust weights in the beginning.
When close enough to the minimum, switch to Newton Method to get rapid local convergence.

The Search for a Minimum : Newton Method (3)

Pros/Cons :

- Not to be used directly for non-linear least square problems.
- Poor convergence when far from a local minimum.
- Lots of computations for a step $\rightarrow$ Approximation of the Hessian.
- Quadratic convergence around a minimum.

The Search for a Minimum : Gauss-Newton Method

- Meant for non-linear least squares problems.
- Based on a linear approximation of the criterion :

$$\tilde{\epsilon}(t,\theta) = \epsilon\left(t,\theta^{(i)}\right) + \left[\epsilon'\left(t,\theta^{(i)}\right)\right]^T \left(\theta - \theta^{(i)}\right)$$

- *Gauss-Newton Hessian* is different than from Newton Method one :

$$R\left(\theta^{(i)}\right) = \left.\frac{d^2 L^{(i)}(\theta)}{d\theta^2}\right|_{\theta=\theta^{(i)}}$$

$$= \frac{1}{N}\sum_{t=1}^{N} \psi\left(t,\theta^{(i)}\right)\psi^T\left(t,\theta^{(i)}\right) \ , \ \psi(t,\theta) = \frac{d\hat{y}(t|\theta)}{d\theta}$$

- Requires only first order derivative information$\rightarrow$ easy to compute.

The Search for a Minimum : Gauss-Newton Method (2)

- Again, update is derived as the minimizer of

$$\theta^{(i+1)} = \theta^{(i)} - R^{-1}\left(\theta^{(i)}\right) G\left(\theta^{(i)}\right)$$

- And Gauss-Newton direction is not calculated with the inversion but by solving

$$R\left(\theta^{(i)}\right) f^{(i)} = -G\left(\theta^{(i)}\right)$$

The Search for a Minimum : Gauss-Newton Method (3)

Convergence :

- Linear convergence.
- For zero or small residual problems, convergence is particularly fast.
- Large noise leads to large residuals but are indepedent on the data.

The Search for a Minimum : Gauss-Newton Method (3)

Pros/Cons :

- Theoretically slower local convergence than Newton Method but in practice, faster, especially when far from the minimum.
- Always reasonable to deploy the method in any case.

The Search for a Minimum : Levenberg-Marquardt Method

Search direction based on the approximation $L^{(i)}(\theta)$ of the criterion $V_N\left(\theta, Z^N\right)$

$$V_N\left(\theta, Z^N\right) \simeq L^{(i)}(\theta) = \frac{1}{2N}\sum_{t=1}^{N} \tilde{\epsilon}^2(t,\theta)$$

Idea is to search for the minimum of $L^{(i)}(\theta)$ in a ball of radius $\delta^{(i)}$ around the current iterate. The optimization problem is then

$$\theta^{(i+1)} = \underset{\theta}{\texttt{argmin}}\, L^{(i)}(\theta) \text{ subject to } \left|\theta - \theta^{(i)}\right| \leq \delta^{(i)}$$

The Search for a Minimum : Levenberg-Marquardt Method (2)

Thus, the direction and step size are defined by

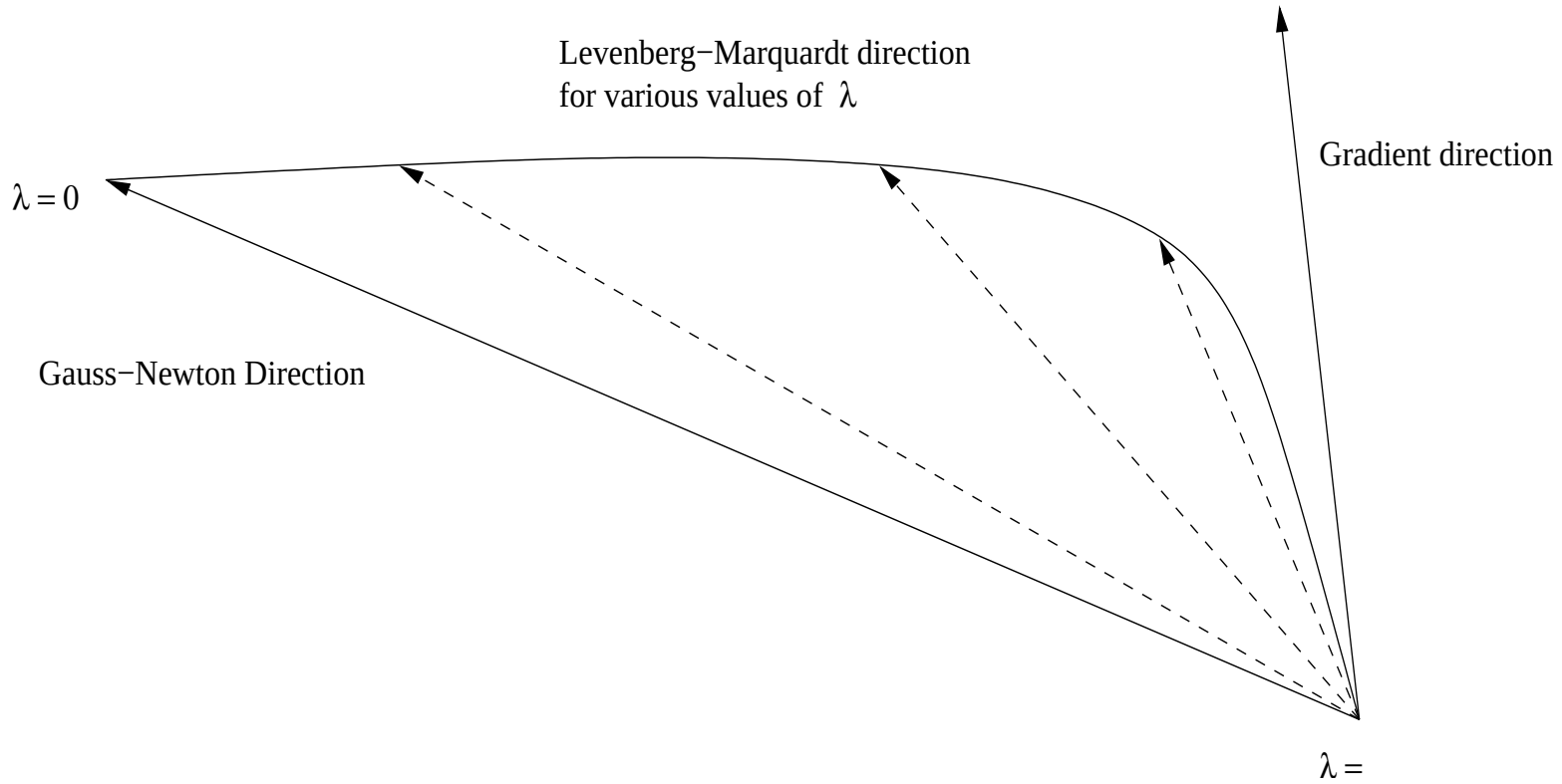
$$\theta^{(i+1)} = \theta^{(i)} + f^{(i)}$$

$$\left[R\left(\theta^{(i)}\right) + \lambda^{(i)} I\right] f^{(i)} = -G\left(\theta^{(i)}\right)$$

There is a monotonic relation between $\lambda^{(i)}$ and $\delta^{(i)}$ but usually hard to get.
Some considerations about λ :

- $\lambda \longrightarrow \infty$, diagonal matrix will dominate $R(\theta)$ leading to gradient method with a very small step size.
- $\lambda \longrightarrow 0$, we come back to the Gauss-Newton Method.

The Search for a Minimum : Levenberg-Marquardt Method (3)



$$\theta^{(i+1)} = \theta^{(i)} + f^{(i)}$$

$$\left[R\left(\theta^{(i)}\right) + \lambda^{(i)} I\right] f^{(i)} = -G\left(\theta^{(i)}\right)$$

The Search for a Minimum : Levenberg-Marquardt Method (4)

Hard to determine $\lambda \rightarrow$ Use of an indirect method :
Observe how well the reduction in the criterion matches the one predicted by $L^{(i)}(\theta)$ and adjust λ according to this. Using

$$r^{(i)} = \frac{V_N\left(\theta^{(i)}, Z^N\right) - V_N\left(\theta^{(i)} + f^{(i)}, Z^N\right)}{V_N\left(\theta^{(i)}, Z^N\right) - L^{(i)}\left(\theta^{(i)} + f^{(i)}\right)}$$

If the ratio is close to one, $L^{(i)}\left(\theta^{(i)} + f^{(i)}\right)$ is likely to be a good approximation of V_N and λ should be decreased. On the other case, λ should be increased.

The Search for a Minimum : Levenberg-Marquardt Method (5)

The Levenberg-Marquardt Algorithm :

- 1. Select initial parameter vector $\theta^{(0)}$ and initial value $\lambda^{(0)}$
- 2. Determine search direction from $\left[R\left(\theta^{(i)}\right) + \lambda^{(i)} I\right] f^{(i)} = -G\left(\theta^{(i)}\right)$
- 3. $r^{(i)} > 0.75 \Rightarrow \lambda^{(i)} = \lambda^{(i)}/2$
- 4. $r^{(i)} < 0.25 \Rightarrow \lambda^{(i)} = 2\lambda^{(i)}$
- 5. If $V_N\left(\theta^{(i)} + f^{(i)}, Z^N\right) < V_N\left(\theta^{(i)}, Z^N\right)$ then accept $\theta^{(i+1)} = \theta^{(i)} + f^{(i)}$ as a new iterate and let $\lambda^{(i+1)} = \lambda^{(i)}$
- 6. If the stopping criterion is not satisfied, go to 2.

The Search for a Minimum : Levenberg-Marquardt Method (6)

Using some computation tricks, the ratio $r^{(i)}$ becomes quite cheap and easy to calculate.
Thus, the Levenberg-Marquardt algorithm is quite easy to implement and use, and is also pretty fast to converge. This makes it the most convenient choice to be made for training neural networks.

The Search for a Minimum : Recursive Algorithms (1)

- Methods discussed up to now are *batch methods*.
  - Process entire data at each iteration
- We can also use *recursive algorithms*
  - Process one input/output pair at a time
    $\theta(t) = \theta(t-1) + \mu(t)f(t)$

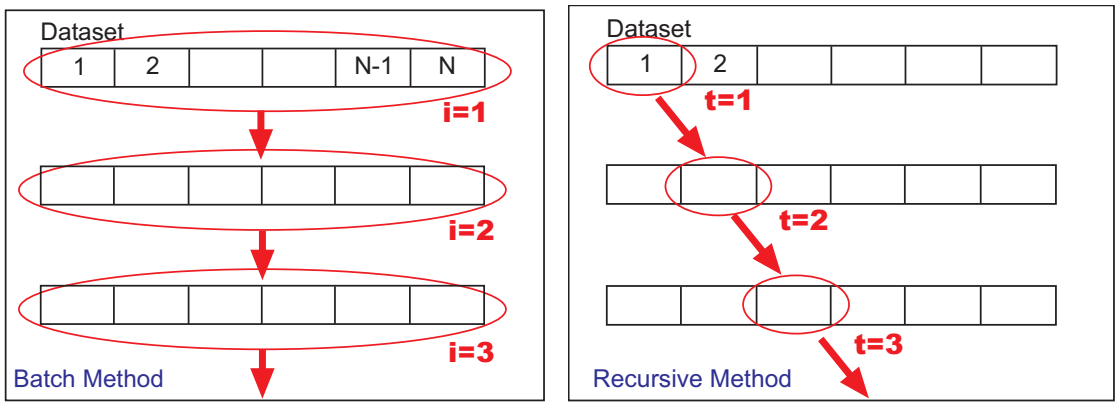


Figure: Batch Methods and Recursive Methods

The Search for a Minimum : Recursive Algorithms (2)

- Batch Algorithms :
  - (-) Consume as many memory as the data size
  - (-) Not suitable for on-line application
  - (+) Suitable for complicated problems (ex. no-linear system)
- Recursive Algorithms :
  - (+) Less memory consuming and simpler implementation
  - (+) Suitable for on-line applcation (ex. adaptive control)
  - (+) Data redduncy is utilized effectively
  - (-) Not suitable for non-linear system

The Search for a Minimum : Recursive Gauss-Newton Method (1)

- An example of recursive method
- Original Gauss-Newton method :
  - Update θ in each iteration with entire data
    i.e. $\theta(i+1)$ depends on $\theta(i), y(1), \ldots, y(N)$
- Put one dataset(input/output) in each iteration
  - $\theta(t) \quad \leftarrow (\theta(t-1), y(1), \ldots, y(t))$
  - $\theta(t+1) \leftarrow (\theta(t), y(1), \ldots, y(t), y(t+1))$
- Considering that contribution to $\theta(t+1)$ from $y(1), \ldots, y(t)$ is neglectable, it is simplified as follows :
  - $\theta(t) \quad \leftarrow \theta(t-1), \varepsilon(t)$
  - $\theta(t+1) \leftarrow \theta(t), \varepsilon(t+1)$

The Search for a Minimum : Recursive Gauss-Newton Method (2)

- Original Gauss-Newton Method :
  $\theta^{(i+1)} = \theta^{(i)} - R^{-1}(\theta^{(i)})G(\theta^{(i)})$ (G : gradient, R : hessian)
- Change variable i to t (time series) :
  $\theta^{(t)} = \theta^{(t-1)} - R^{-1}(\theta^{(t-1)}G(\theta^{(t-1)})$.
- G depends on $\theta^{(t-1)}$ and Z^t. ($Z^t : y(1), \ldots, y(t)$)
  We can split the elements containing Z^{t-1} and $y(t)$:
  $G(\theta^{(t)}) = \frac{t-1}{t} V'_{t-1}(\theta^{t-2}, Z^{t-1}) - \frac{1}{t}\psi(t,\theta)\varepsilon(k,\theta)$

The Search for a Minimum : Recursive Gauss-Newton Method (3)

- (Show again : $G(\theta^{(t)}) = \frac{t-1}{t} G_{t-1}(Z^{t-1}) - \frac{1}{t}\psi(t,\theta)\varepsilon(k,\theta)$)
- Since $\frac{t-1}{t} V'_{t-1}(Z^{t-1}) << \frac{1}{t}\psi(t,\theta)\varepsilon(k,\theta)$,
  $V'_t(Z^t) = -\frac{1}{t}\psi(t,\theta^{(t-1)})\varepsilon(k,\theta^{(t-1)})$
- $\theta^{(t)} = \theta^{(t-1)} - \frac{1}{t} R^{-1}\psi(t)[y(t) - \hat{y}(t|\theta^{(t-1)})]$
  (while $\psi(t)$ only depends on $\theta^{(t-1)}$ and $y(t)$)

The Search for a Minimum : Recursive Gauss-Newton Method (4)

Recursive Gauss-Newton Method is mainly for off-line system, since adaptation speed is not enough for dynamic tracking. There are another algorithms such as *Recursive Least Squares(RLS) algorithm*. You can also modify *Recursive Gauss-Newton Method* to use in on-line system.

The Search for a Minimum : Exponential Forgetting

- Sometimes it is desirble to "forget" past learning (ex. adaptive filter)
- Introduce exponential decay to criterion in order to discard past learning

$$V_t(\theta, Z^t) = \frac{1}{2t} \sum_{k=1}^{t} \lambda^{t-k} \varepsilon^T(k, \theta) \varepsilon(k, \theta)$$

- Selecting adequate λ is difficult (decay factor) (Somtimes cause covariance blow-up)
- *Constant Trace* and *EFRA (Exponential Forgetting and Resetting Algorithm)* can prevent covariance blow-up

Necessity of Generalization (1)

- Our goal : A model which can explain any dataset
- Perform test procedures:
  - Choose a model (including # of weights)
  - Determine weights(θ) with traning data
  - Apply the model to test data
- It is desirable to explain both traning data and test data with one model

Necessity of Generalization (2)

- If the model fits to traning data so much, it gets away from test data. (*overfitting*)
- What is wrong? ...
  - Too complicated model
  - Small training dataset
  - Much data variance

  Mean square error criterion does not take these factors into acount
- More general estimation is needed

Bias Error and Variance Error

- Predicted value and real value are always different (error)
- Conceptually, you can divide the error into two:
  - Bias error . . . comes from model inaccuracy
  - Variance error . . . comes from data variation
- Complicated model(= with many weights) may learn variance errors of training data

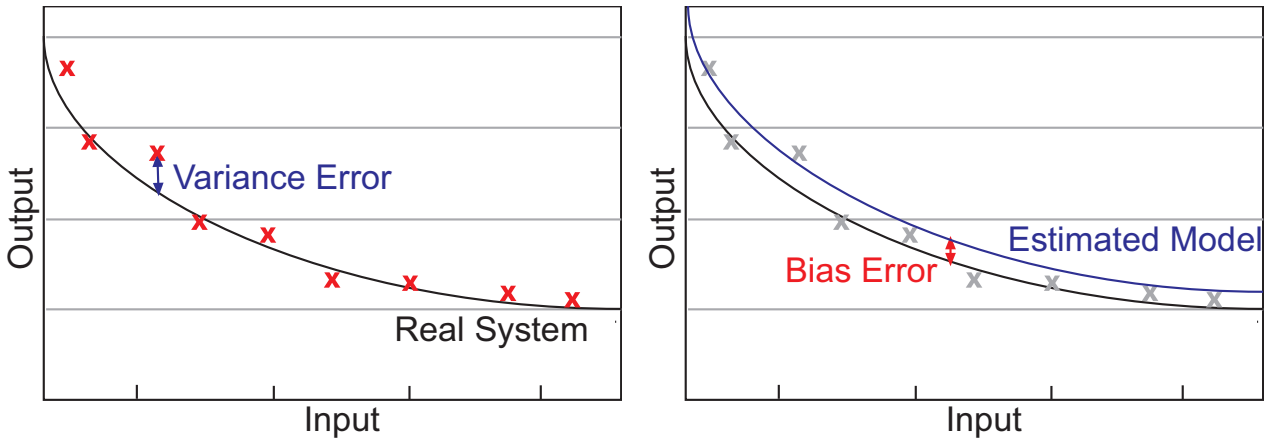


Figure: Variance Error and Bias Error

Akaike's Final Prediction Error(FPE) Estimate

- A measurement of a model quality $\hat{V_M} = \frac{1}{2}\sigma_e^2(1 + \frac{p}{N})$
  (p : # of weights, N : # of training data)
- Estimate errors(bias + variance) for arbital dataset only with training data
- Qualitative interpretation :
  - Increasing number of weights may cause overfitting
  - It is better to use more training data
  - Error with the training data is also taken into acount
- In the weight determination stage, p and N is already fixed

Regularization : Weight Decay (1)

- Regularization is needed

  (Not persist on training data, but create more general model)
- Introducing *weight decay* can avoid overfitting
  - The real system must be simple
  - Weak connections are caused by errors
  - Put a penalty on weak connections (= small weight value), and reduce avairable connection
  - Only strong connection (= large weight value) can survive
- Add one term to the criterion

$$W_N(\theta, Z^N) = \frac{1}{2N}\sum_{t=1}^{N}[y(t) - \hat{y}(t|\theta)]^2 + \frac{1}{2N}\theta^T D\theta$$

  (D is most often selected as $D = \alpha I$)
- α determines the strength of the penalty

Regularization : Weight Decay (2)

- With adequate weight decay, you can get a better model
  - No decay : the model is apart from test data
  - $\alpha = 1$: the model explains both traning and test data well

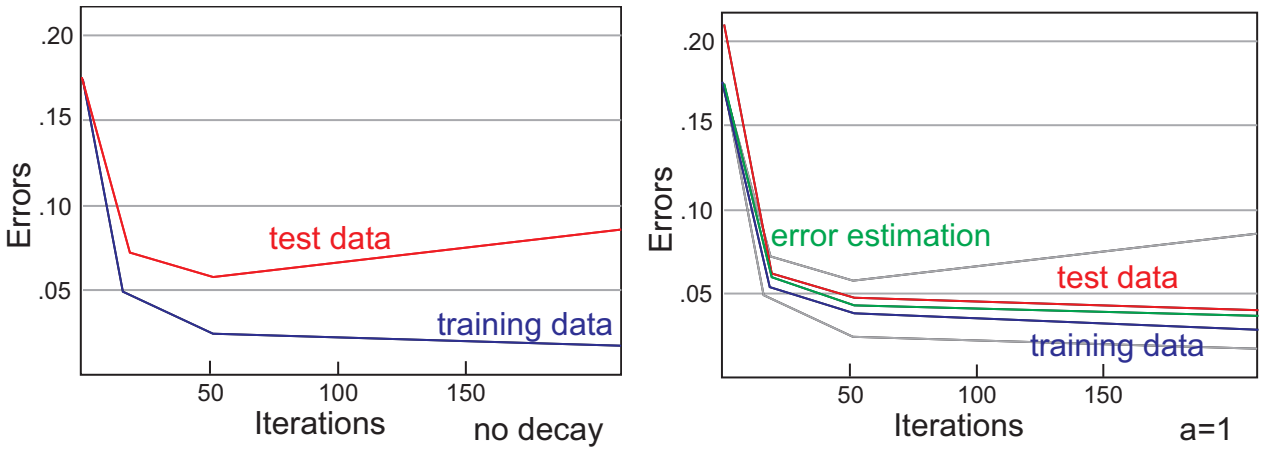


Figure: Weight Decay and Errors (1)

Regularization : Weight Decay (3)

- Inadequate α causes *underfitting*
  - $\alpha = 100$: the model is worse than no decay learning
  - In this time, the best model is obtained when $\alpha = 1$

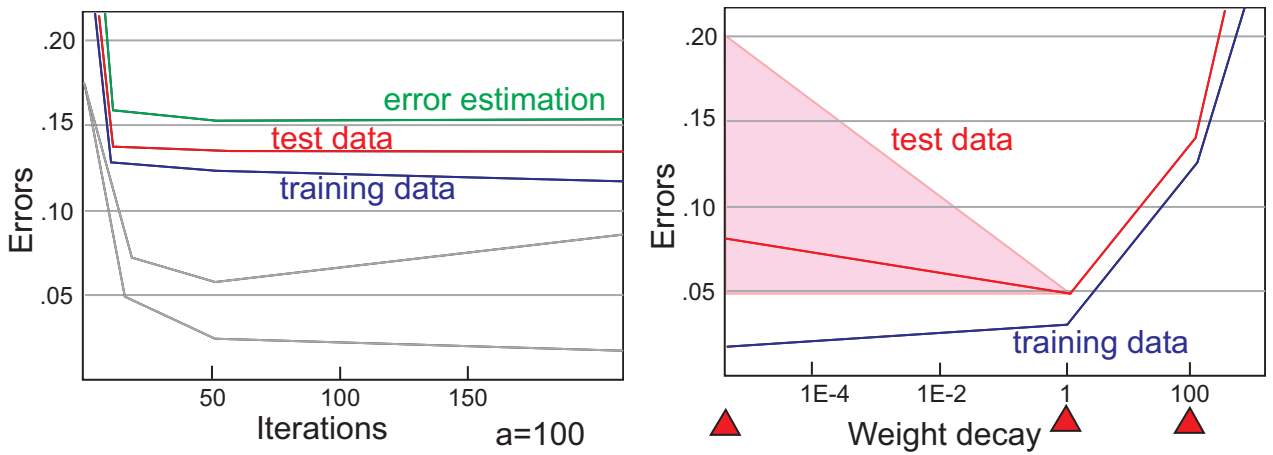


Figure: Weight Decay and Errors (2)

Main Ideas : Summary

- Introduction : With changing weights, a model can express many systems
- PEM method : Criterion function can assess model inaccuracy. To minimize the criterion, we presented some training methods. The most basic method is Gradient Method, and Especially Levenberg-Marquardt Method is quite useful for practical use. Recursive methods are suitable for on-line system.
- Generalization : Sometimes overfitting and underfitting occur. Error can be divided into bias error and variance error. In order to measure errors, Akaike's Final Prediction Error is introduced.
- Regularization : Weight decay serves practical way to prevent from overfitting