Training Techniques and Tips

1

Launder Input

- If the network is to learn a function, make sure that the samples are functional, i.e. that they don’t specify conflicting outputs for the same input value.
- For example, if clinical outcomes are the output, it is possible that two patients with the same symptoms have different outputs; presenting these to the network will mean that it will never fully converge.

2

BackProp Technique & Tricks

(Some of these apply to General Neural Networks)

(Two References:

Neural Networks Tricks of the Trade, Orr and Muller, eds., LNCS 1524

http://www.dontveter.com/bpr/bpr.html)

- Choose examples with maximum information content
  - Shuffle the training set so that successive samples rarely belong to the same class.
  - Present input examples that produce a large error more frequently than ones that produce a small error.

3

Technique & Tricks

- Normalize the inputs
  - Better if mean of a particular variable is near 0.
    - Then weight changes are less likely to be synchronized, since some will be positive, others negative.
    - Therefore, **subtract the actual mean** from the variable before training.
  - Better if the variables are scaled to have similar auto-covariances, defined as

    (sum-of-squares of variable)/(number of samples)
    - Then the **weights will learn at similar rates**.
    - Exception: When some variables are known in advance to be of less significance.

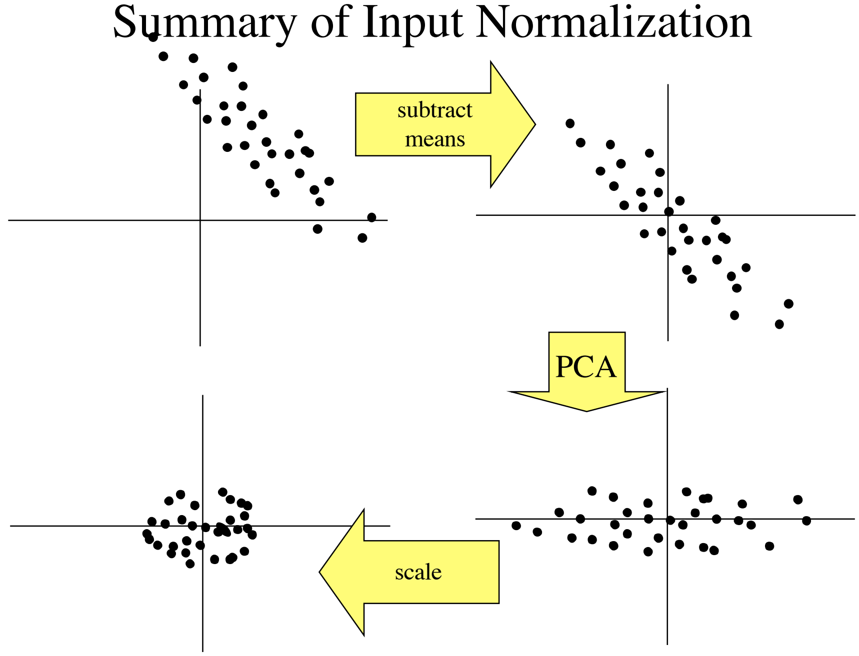
4

Technique & Tricks

- Decorrelate the inputs
  - Better if no two input variables are correlated.
  - Correlated inputs analogous to having linearly dependent variables in a linear system.
  - A technique called PCA (Principal Components Analysis), aka Karhunen-Loeve Expansion, can be used to remove linear correlations.
  - We will look at PCA later; PCA itself can be done by a PCA neural network.

5

Summary of Input Normalization



6

BackProp Technique & Tricks

- Prefer tansig (hyperbolic tangent) rather than logsig for inner layers.
  - tansig output is symmetric about origin, logsig is not.
  - tansig will more likely produce outputs close to 0 for the **next stage** of the network
- Some recommend adding a small linear constant to the output of tansig to "avoid flat spots"

7

Piecewise Quadratic Approx. to tanh (faster to compute)

```
 x                 | f(x)
-----------------------------------------------------------
x > 1.92033        | 0.96016
0 < x <= 1.92033   | 0.96016 - 0.26037 * (x - 1.92033)^2
-1.92033 < x < 0   | 0.26037 * (x + 1.92033)^2 - 0.96016
x <= -1.92033      | -0.96016
```

Derivative: $\tanh'(x) = 1-\tanh^2(x)$ can still be used.

8

Choice of Target Values

- Choosing target values of +1, -1 for a tansig causes the neuron to be driven toward the **saturation region**.
- To get into this region, the weights are large and may become "stuck" because small gradient values will not change them sufficiently.
- It may be better to **choose the targets offset** from these saturation values, or to scale the tansig to get the same effect, e.g.
  - f(x) = 1.7159 tanh(2x/3), which has a maximum 2nd derivative where the function's value is +/- 1.

9

Weight Initialization

- Assuming that the training set has been normalized and the previous sigmoid is used,
- Draw the initial weights from a distribution, such as a uniform distribution, with mean 0 and standard deviation 1/sqrt(m) where m is the **fan-in** (number of inputs to the node).
- Increases likelihood that the input to the sigmoid will have a standard deviation of 1
  (since the latter is the sqrt of the sum of the squares of the weights, for normalized input).

10

Learning Rates

- Ideally, each **weight** should have its own learning rate. See the Neural Networks Tricks of the Trade, Orr and Muller, eds., LNCS 1524 for how to choose learning rate based on 2nd derivatives.
- As a substitute, each neuron, or each layer could have its own learning rate.
- Learning rates should be proportional to the sqrt of the number of inputs to the neuron.
- Weights in **earlier layers should be larger** than those in later layers, since the earlier layers tend to have a smaller 2nd derivative of the MSE.

11

Second Derivatives by Layer

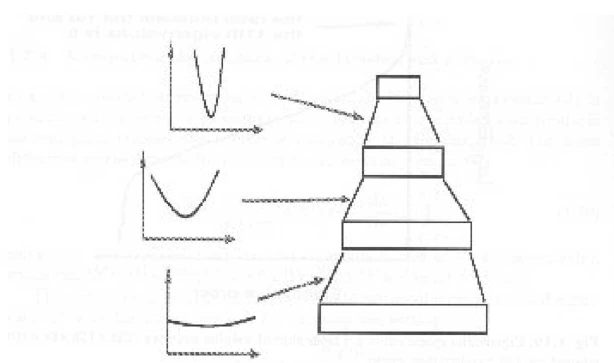


Fig. 1.21. Multilayered architecture: the second derivative is often smaller in lower layers

12

Validation Technique (“Cross-Validation”) & Early Stopping

- Split the training set into **training** and **validation** subsets, e.g. 2:1 or 5:1 ratio.
- Train only on the training subset; use the validation set for MSE, every so often (e.g. every 5 epochs).
- **For early stopping:** Stop training **as soon as the validation error goes up**.
- Use the weights **before** the error went up.
- Rational: Even though a lower minimum might have been reached, the local minima tend to be fairly close in value in practice.

13

A Validation Error Curve

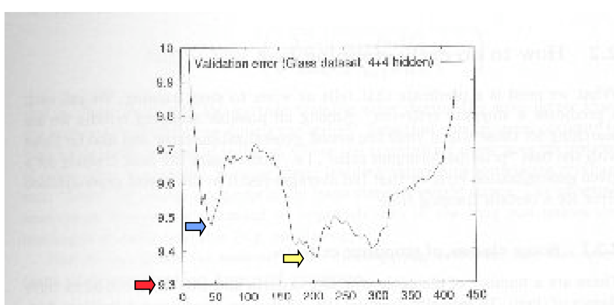


Fig. 2.2. A real validation error curve. Vertical: validation set error; horizontal: time (in training epochs).

See Neural Networks Tricks of the Trade, Orr and Muller, eds., LNCS 1524 for further refinements of the validation idea.

14

Over-Fitting

- It is possible for a network to *over-fit* the data, meaning that it learns small variations in the data which might actually be due to noise.
- Another way of saying this is that the network does not generalize well; it is **too specialized**.
- **Validation** is one technique used to help avoid over-fitting.
- Over-fitting can result if the network has **too many neurons** at its disposal.

15

Sizing a Network

- Given a problem:
  - How many layers?
  - How many neurons per layer?
  - What activation functions?

16

Layers

- Theoretically, any function can be emulated over a given range by a network with just one hidden layer and one output layer (two layers total), with sufficient neurons in that layer.
- Practically, 2-3 layers suffice for large families of problems, although more may be used, especially when special feature-selection layers are used, as in the zip-code recognition network.

17

Neurons

- Choose number of neurons based on the assessed complexity within a layer (number of crests and valleys of a function, for example).
- Two approaches for experimental determination:
  - Start with a large number of neurons and prune.
  - Start with a small number of neurons and build up.

18

Pruning

- Negligible weights can be eliminated (set to 0).
- If all input weights to a node are 0, the node can be eliminated.
- If all weights a node feeds are 0, the node itself can be eliminated.
- Vary weights w to see whether $\partial J / \partial w$ is significant; if not, prune the weight.

Building

- Cascade-Correlation Network (Fahlman) adds one neuron at a time, testing the quality of the results and stopping when they are adequate.
- Training by correlation is a technique to be explored later.
- Problem with cascade correlation is that **each added neuron is effectively a new layer**.

Doubling

- Start with a small number of neurons in the inner layer.
- If at the conclusion of a training cycle, the MSE is inadequate, repeat with double the number of neurons.

21

Number of Training Samples for a Given Size Network

- Baum-Hausler rule (1989):

Necessary condition:

(number of samples) > W / (1-a)

where W is the number of weights in the network and a is the desired accuracy on the test set.

- **Sufficient condition**:

(number of samples) ≥ log(N / (1-a)) * W/(1-a)

where N is the number of neurons.

22