Practical considerations:

(1) (a) When we are looking at local gradient in BPA, it depends on derivative of activation function. Hence the Act. fun must be continuous and differentiable. [condition]
Therefore we use Sigmoidal non linear function.
It has 2 forms:

(i) Logistic sigmoidal act. function.

for a neuron j: $\phi_j(v_j(n)) = \frac{1}{1 + \exp(-a v_j(n))}$ $a > 0$ (condition)

$-\infty < v_j < \infty$.
(no restriction on boundary v_j)

WKT a: controls the slope of Act^n function.
v_j: induced local field.

$$\phi_j'(v_j(n)) = \frac{a \exp(-a v_j(n))}{[1 + \exp(-a v_j(n)]^2}$$

WKT $y_j(n) = \phi_j(v_j(n))$.

$$\phi_j'(v_j(n)) = a\, y_j(n)\,[1 - y_j(n)] \quad \text{---} \quad (2).$$

If neuron j is in output layer we simply write:

$$y_j(n) = \phi_j(v_j(n)) = o_j(n)$$

& local gradient:

$$\delta_j(n) = e_j(n)\,\phi_j'(v_j(n))$$
$$= a\,[d_j(n) - o_j(n)]\, o_j(n)\,[1 - o_j(n)]$$

(neuron j)

If j is in hidden layer.

$$\delta_j(n) = \phi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n)$$

$$= a\, y_j(n)[1 - y_j(n)] \sum_k \delta_k(n) w_{kj}(n).$$

The range of o/p $y_j(n)$ will be 0 to 1

When $y_j(n) = 1/2$, The eq 0 will be equal to

$$\phi_j'(v_j(n)) = a\frac{1}{2}\cdot[1 - 1/2] = a\,1/4.$$

$y_j(n) = 0 \qquad \phi'j(v_j(n)) = 0.$

$y_j(n) = 1 \qquad \phi_j'(v_j(n)) = 0.$

$\Rightarrow$ ϕ_j' has maximum value at $y_j(n) = 1/2$.
at this point the w value adjustment will be maximum in the $0 \le y \le 1$.

(2) Hyperbolic tangent function:

$$\phi_j(v_j(n)) = a\tanh(b\,v_j(n)) \qquad (a, b) > 0.$$

$$\phi_j'(v_j(n)) = ab\,\mathrm{sech}^2(b\,v_j(n))$$
$$= ab\,(1 - \tanh^2(b\,v_j(n))$$
$$= \frac{b}{a}[a - y_j(n)][a + y_j(n)]$$

If neuron j is at output layer

local gradient

$$\delta_j(n) = g_j(n)\,\varphi_j'(v_j(n))$$

$$= \frac{b}{a}\,[y_{dj}(n) - g_j(n)]\,[a - g_j(n)]\,[a + g_j(n)]$$

If neuron j in a hidden layer

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n)\, w_{kj}(n)$$

$$= \frac{b}{a}\,[a - y_j(n)]\,[a + y_j(n)] \sum \delta_k(n)\, w_{kj}(n)$$

∴ Local gradient may calculate w/o requirement of activation function knowledge.

(2) Rate of learning:

If we see carefully the BPA it is nothing but modified / approximated version of steepest descent method.

The trajectory in perfect SDM is smooth. but in BPA is not smooth ∴ it depends on η. If η is small, trajectory will be smooth, but at cost of a slower rate of learning. on other hand if η is too large in learning speed will be fast ∴ of large change in w, which may lead to unstable (oscillatory).

A simple way to over come this method

it by consider the following

$$\Delta W_{ji}(n) = \alpha \Delta W_{ji}(n-1) + \eta f_j(n) y_i(n) \quad \text{— ①}$$

NKT : $W(k+1) = W(k) + \Delta W(k)$

Where $\Delta W(k) = -\eta \frac{\partial \mathcal{E}(k)}{\partial W}$ in general case

local gradient

$$f_j(n) = -e_j(n) \odot \phi'(v(\cdot))$$

$$\frac{\partial \mathcal{E}(k)}{\partial W} = -\eta f_j(n) y_j(n)$$

In Eq^n ① 2nd term has η measures how much the current situation affects the next step.
1st term has α : momentum which measures how much past steps affect the next step.

Eq ① : Generalized delta rule.

Now find the solution for eq ① since it has first difference.

$$\Delta W(n) - \alpha \Delta W(n-1) = \eta f(n) y(n)$$

$n = 0$

$$\Delta W(0) = \alpha \Delta W(-1) + \eta f(0) y(0) = \eta f(0) y(0)$$

$W(-1) = 0$.

$n = 1$

$$\Delta W(1) = \alpha \Delta W(1) + \eta f(1) y(1)$$
$$= \alpha W(0) + \alpha \eta f(0) y(0) + \eta f(1) y(1).$$

$n=2$ $\Delta W(2) = \alpha \Delta W(1) + \eta \delta(2) y(2)$

$= \alpha^3 \Delta W(0) + \alpha^2 \eta \delta(0) y(0) + \alpha \eta \delta(1) y(1)$

$+ \eta \delta(2) y(2)$

If $W(0) = 0$

and so on.

$\Rightarrow$

$$\Delta W(n) = \eta \sum_{t=0}^{n} \alpha^{n-t} \delta(t) y(t)$$

$$\Delta W(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial \mathcal{E}(t)}{\partial W(t)}$$

The following observations can be made w.r.t above eqn.

① The above eqn converges if $0 \leq |\alpha| < 1$
If $\alpha = 0$, BPA operates w/o momentum.

② If $\frac{\partial \mathcal{E}(t)}{\partial W(t)}$ has same sign in the consecutive iteration then the sum grows in magnitude as a result ΔW will be adjusted by a large amount.

③ If $\frac{\partial \mathcal{E}(t)}{\partial W(t)}$ has opposite sign in the consecutive iteration, sum will decrease. So W is adjusted by a small amount.

(3)

Sequential and Batch mode of training:

One complete presentation of the entire training set during the learning process is called epoch

The learning process is maintained on an epoch by epoch basis until the synaptic weights and bias level of the network stabilize and avg sq. error over the entire training set converges to some minimum value.

BP learning may proceed in one of 2 ways.

① Sequential mode: It is also referred as on-line/ pattern / stochastic mode.
In this mode of operation weight updating is performed after the presentation of each training example.

② Batch mode:
Here weight updating is performed after the presentation of all the training examples that constitute an epoch.
Here we consider average squared error.

$$E_{av} = \frac{1}{2N} \sum_{n=1}^{N} \sum_{j \in C} e_j^2(n).$$

The $e_j(n)$ pertains to o/p neuron j for training example n.
Inner Summation: w.r.t j is performed over all neurons in the o/p layer of N/w.
Outer ——u——: w to n is performed over the entire training set in the epoch at hand.

$$\Delta w_{ji} = -\eta \frac{\partial \mathcal{E}_{av}}{\partial w_{ji}}$$

$$= -\frac{\eta}{N} \sum_{n=1}^{N} e_j(n) \frac{\partial e_j(n)}{\partial w_{ji}}$$

Advantages of sequential mode:

① It has storage for each synaptic connection.
② Easy to implement
③ provides effective solution to large & difficult pblm.
④ ability to take the adv. of redundancy by presenting example one at a time.

Disadvantage:

① Stochastic nature of presenting samples to N/w is a pblm ∴ it is trapped in a local minimum.

③ Stopping Criteria:

In general the BPA cannot be shown to converge and there are no well defined criteria for stopping its operation.

2 ways to stop BP learning

The BPA is considered to have converged

(i) when the Euclidean norm of the gradient vector reaches sufficiently small gradient threshold.

Drawback:

(ii) when the absolute rate of change in the averaged squared error per epoch is sufficiently small.

Heuristic For Making the BPA perform better.

1. Sequential v/s batch mode update:
   For large & redundant data, sequential processing is faste
   For smaller data & N/w, B.processing is faster.

(2). Maximizing information content:
   Aim: Every training example presented to the BPA should be chosen on the basis that its information content is the largest possible for the task at hand

To acheive this aim ~~xxx~~ there are 2 ways.

(1). The use of an example that results in the largest training error.
   ∴ for updating w-value local gradient will be used which is a function of error when an example produces the larger error which leads to have larger local gradient, so faster will be the learning process.

(2). The use of example that is radially different from all those previously used.

3. Activation function: choice of

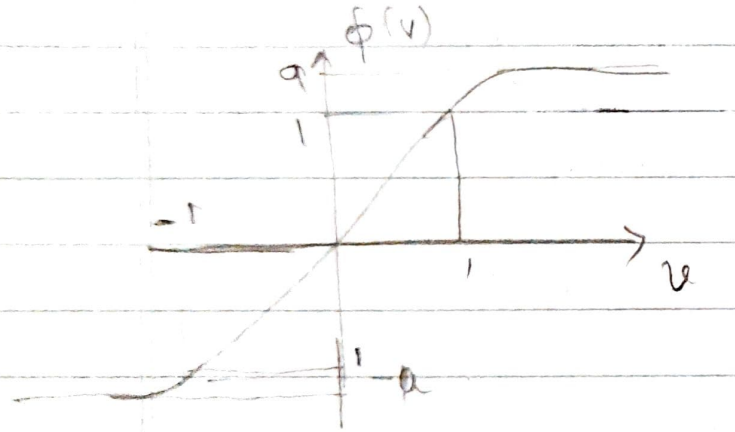
As we discussed earlier, activation function plays an impt role in training NN using BPA.

If we use tangnt hyperbolic activation fun learning will be faster than logistic. Because T.Hyperbolic is antisymmetric where as logistic is non symmetric about 0.

i.e $\phi(-v) = -\phi(v)$.

$\varphi(v) = a \tanh(bv)$.

LeCun chook particular value for $a = 1.7159$ and $b = 2/3$. Because the following useful ppties.



① $\varphi(1) = 1$

$\varphi(-1) = -1$

② At the origin the slope (i.e effective gain) of the activation fun is close to 1

$\varphi(0) = ab$.

$= 1.1424$

③ $\varphi''(v)$ attains its maximum value at $v = 1$.

$|v_i| \leq 1$: Region is linear

$|v_i| > 1$: Region is referred as Saturation

④ Target value:

It is impt to choose the target value within the activation function range i.e ±1 for example

Let us assume the desired o/p at neuron is y_{dj}. If its it should be offset by some amount ϵ.

i.e $y_{dj} = a - \epsilon$

(or) $y_{dj} = -a + \epsilon$

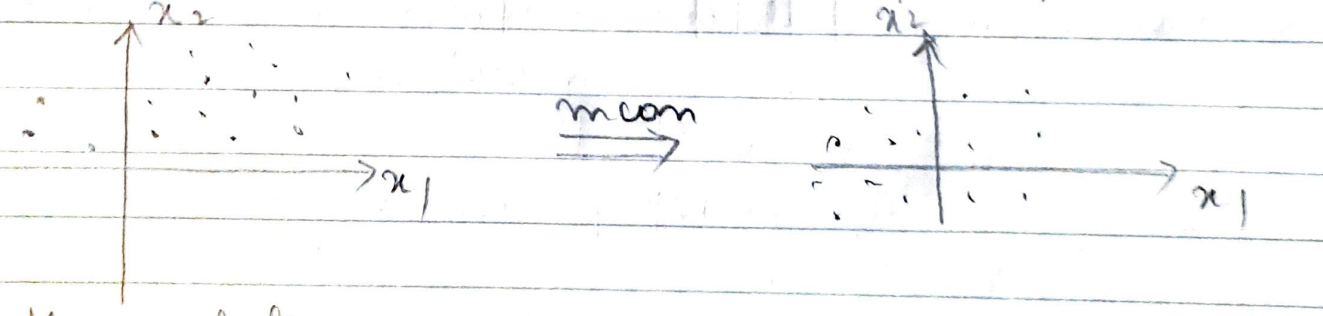
So that it will in the linear region of activation function. Otherwise the BPA tends to drive the free parameters (w) of the NN to inf and thereby slow down the learning process ~~deleting~~.

⑤ Normalization of inputs:

The training samples must be preprocessed before presenting to the NN

The preprocessing steps are

① ~~Mean~~ Subtracting each input from mean



Here data was not distributed properly as a result after training process down the sample coming from -ve side NN not ab

Datas are balanc

Non zero mean → large & values → condition no is large → cost fun. surface will be steep

(2) de correlate the Training samples using PCA otherwise NN will be classify order

(3) Covariances : It ensures the different sy wt in the n/w learn at approximately the same speed

(6) Learning from hints:
learning basically means finding $f(\cdot)$ which is mapping unknown input-output pairs.
During this process it exploits the inf^{n} present in the t-example. This process can be made faster by giving hints along with input examples hints can be symmetric, invariance ppties etc.

(7) Learning rate:

$$\eta_1 > \eta_2 > \eta_3 \ldots > \eta_L$$

Initialization:

A good choice for the initial values of the synaptic weights and thresholds of the N/W can be more useful in a successful N/W design. Then what is a good choice?

If I choose large values for w, it is highly likely that the neurons in N/W will be driven into saturation. If this happens the local gradient in BPA assume small values, which in Turn will cause the learning process to slow down.

If w's are assigned with small initial values, BPA may operate on a very flat area around the origin of the error surface; where origin is a saddle point

Let the bias applied to each neuron in the network be set to zero & $v_j = \sum_{i=1}^{m} w_{ji} y_i$, $\mu_y = E[y_i] = 0 \quad \forall i$

and $\sigma_y^2 = E(y_i - \mu_i)^2 = E[y_i^2] = 1$

$$E[y_i y_k] = \begin{cases} 1 & k = i \\ 0 & k \neq i \end{cases}$$

and w's are drawn from a uniformly distributed set of no. with '0' mean

$$\mu_w = E[w_{ji}] = 0 \quad \& \quad \sigma_w^2 = E[w_{ji} - \mu_w]^2 = E[$$

mean & variance of v_j

$$\mu_v = E[v_j] = E \sum w y = 0.$$

$$\sigma_v^2 = E[v_j^2].$$

$$= \sum E[w_{ji}^2] \qquad i = k.$$

$$= m \sigma_w^2$$

where m is the no. of synaptic con

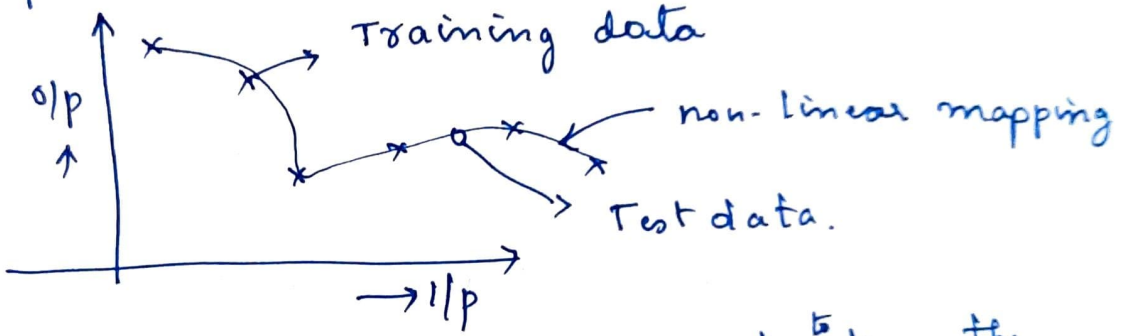
Generalization & Cross validation.

(i) Generalization:

A N/W is said to generalize, when the I/p - O/p mapping computed by N/W is correct for test data not used in the training.

To understand the concept of generalization, we consider the N/W as s/m. which performs non linear Input - output mapping.

The learning process is viewed as a curve-fitting problem. Then problem of mapping simply boils down to that of interpolation



The training data is used to help the N/W to learn the nonlinear input-output mapping represented by the curve.

When a test data is presented, the N/W computes the O/p as a result of interpolation performed by the N/W.

When a NN learns too many I/p - O/p samples, the N/W may end up memorizing the training set. w/o trying to find the underlying function to be modelled.

- overfitting / over training.

when over trained the N/w loses the ability to generalize

This normally happens when more no. of H. Neuron than necessary are used resulting in undesired contributions in the input space due to noise stored in wt.

Generalization is influenced by.

- Size of training set and how close it is to the envt. of interest.
- Arch. of NN.
- Physical complexity of the pblm.

Good generalization

$$N = O\left(\frac{W}{\epsilon}\right). \qquad O(\cdot) : \text{order of quantity}$$

W: Total no. of free parameters.

ϵ : error permitted on the test data.

Cross Validation:

→ Choosing a suitable topology for a NN, given application is a difficult problem.

→ Usually, after a tedious trial & error process, oversized topology is chosen which is prone various drawbacks like:

① high demand on computational resource
② increase in training time
③ non convergence of parameters
④ Decrease in the generalisation capability a NN
⑤ price for H/W implementation ↑
⑥ less efficient.

If NN topology is too small, then it might not be le

To overcome this problem, we need to find minimal topologies with but performance

There are 2 ways to achieve

① Growing approach.
② Pruning Methods

Generally latter method starts the training NN which is expected to be big enough to en successful training

Pruning Methods
- ① deletion
- ② regularization

① Deletion (Hessian based N/W pruning)

The basic idea of 2nd approach is to use info of 2nd derivatives of error surface in order to make a trade off b/w E_{av} & error performance & network complexity

To do Fist we construct a local model of error surface by which predicting the effect of perturbation in W

consider

$$\mathcal{E}_{av}(W+\Delta W) = \mathcal{E}_{av}(W) + \frac{\partial \mathcal{E}_{av}}{\partial W}\frac{\Delta W}{1!} + \frac{\partial^2 \mathcal{E}_{av}}{\partial W^2}\frac{\Delta W^2}{2!} + H.O.T$$

(H ↙)

ΔW is perturbation applied to operating point W

Next, we need to identify a set of parameters whose deletion from N/W will cause the least increase in the $\mathcal{E}_{av}$. We identify as follows by assumption

① at $g^T \Delta W = 0$ after training ② error surface around local minima is highly quadratic & neglecting H.O.T

$$\Delta \mathcal{E}_{av} = \mathcal{E}(W+\Delta W) - \mathcal{E}(W) = \frac{1}{2}\Delta W^T H \Delta W$$

We adopt OBS procedure

Goal is set one of the synaptic weights to zero to minimize the incremental increase in $\mathcal{E}_{av}$ to do so we enforce $u_i^T \Delta W + w_i = 0$.

Minimize the quadratic form w.r.t to incremental change in the weight vector ΔW subject to the constraint that $u_i^T \Delta W + w_i$ is zero & then minimize it w.r.t index i

$$S = \frac{1}{2}\Delta W^T H \Delta W + \lambda (u^T \Delta W + w_i)$$

$$\frac{\partial S}{\partial \Delta W} = H\Delta W + \lambda u_i = 0 \Rightarrow \Delta W = -\lambda H^{-1} u_i$$

$$S = \frac{w_i}{(H^{-1})_{ii}}$$

Conjugate Gradient Method.

→ It is a second-order optimization methods

→ Let us assume minimization of quadratic function

$$f(x) = \frac{1}{2} x^T A x - b^T x + c.$$

Where x is a $N \times 1$ parameter vector

A : $N \times N$ symmetric positive definite matrix

b : $N \times 1$ vector

c : constant.

minimization of quadratic fun. $f(x)$ is achieved by

$$f'(x) = 0.$$

$$\Rightarrow \quad Ax^* - b = 0$$

$$x^* = A^{-1} b.$$

Thus minimizing of $f(x)$ and solving Linear sys of Eq^{n}s $Ax^* = b$ are equivalent pblms.

Definition:
Note: for a given matrix A, we say that a set of non zero vectors $[s(0)\ s(1) \ldots s(N-1)]$ is A-conjugate. If the following condition is satisfied.

$$s^T(n) A s(j) = 0 \qquad \forall\ n, j \text{ s.t } n \neq j.$$

For a given set of A-conjugate vectors $s(0), s(1), \ldots, s(W-1)$, the corresponding conjugate direction method for unconstrained minimization of the quadratic cost function $f(x)$ is defined by

$$x(n+1) = x(n) + \eta(n)\, s(n) \qquad n = 0, 1, \ldots, W-1$$

where $x(0)$ is an arbitrary starting vector

$\eta(n)$: scalar. defined by

$$f(x(n) + \eta(n)\, s(n)) = \min_{\eta} f(x(n) + \eta\, s(n))$$

The procedure of choosing η so as to minimize the fem $f(x(n) + \eta s(n))$ for some fixed n is referred to as line search.

$$f(x(n) + \eta(n) s(n)) = \frac{1}{2}\left[x(n) + \eta(n)\, s(n)\right]^T A \left[x(n) + \eta(n) s(n)\right] - b^T \left[x(n) + \eta(n) s(n)\right] + c$$

$$f(x(n) + \eta(n) s(n)) = \frac{1}{2}\left\{x^T(n) A x(n) + \eta(n)\, s^T(n) A x(n) + \eta x^T(n) A s(n) + \eta^2(n)\, s^T(n) A s(n)\right\} - b^T x(n) - \eta(n)\, b^T s(n) + c$$

$$\frac{\partial f(\cdot)}{\partial \eta} = 0 + \frac{1}{2} s^T(n) A x(n) + \frac{1}{2} x^T(n) A s(n) + \eta(n)\, s^T(n) A s(n) - b^T s(n)$$

$$0 = s^T(n) A x(n) - b^T s(n) + \eta(n)\, s^T(n) A s(n)$$

$$\eta(n)\, s^T(n) A s(n) = -\, s^T(n) \{Ax - b\} \qquad A = A^T = A^{-1}$$

$$\eta(n) = -\, \frac{s^T(n) A \{x(n) - A^{-1} b\}}{s^T(n) A s(n)}$$

$x(n) - x^* = \varepsilon$

where $x^* = A^{-1} b$

$$\eta(n) = \frac{-\, s^T(n)\, A\, \varepsilon(n)}{s^T(n) A s(n)}$$

starting from an arbitrary point $x(0)$ the conjugate direction method is guaranteed to find optimal sol^n x^* of $f(x)=0$ in at most N iterations.

For the CG method to work we require the availability of set A-conjugate vectors $s(0)\ s(1) \ldots s(N-1)$ it is determined in a sequential manner at successive step.

Define residual $r(n) = b - Ax(n)$

L.C of $r(n)$ & $s(n-1)$ $\Rightarrow$ $s(n) = r(n) + \beta(n)\, s(n-1)$

$n = 1 \ldots N-1$

Where $\beta(k)$ is chosen s.t A conjugacy preserves.

$\Rightarrow s^T(n-1)\, A\, s(n) = s^T(n-1)\, A\, r(n) + \beta(n)\, s^T(n-1)\, A\, s(n-1)$.

$0 = s^T(n-1)\, A\, r(n) + \beta(n)\, s^T(n-1)\, A\, s(n-1)$

$$\Rightarrow \quad \beta(n) = - \frac{s^T(n-1)\, A\, r(n)}{s^T(n-1)\, A\, s(n-1)}.$$

To avoid the dependency on A to generate s's vectors. we consider

Polak - Ribiere formula

$$\beta(n) = \frac{r^T(n)[r(n) - r(n-1)]}{r^T(n-1)\, r(n-1)}$$

Fletcher - Reeves formula

$$\beta(n) = \frac{r^T(n)\, r(n)}{r^T(n-1)\, r(n-1)}.$$