A review of Random feedback weights support learning in deep neural networks

SangHyun Yi

sangyi92@snu.ac.kr

November 7, 2016

Table of contents

The idea of Random feedback

Recap backprop

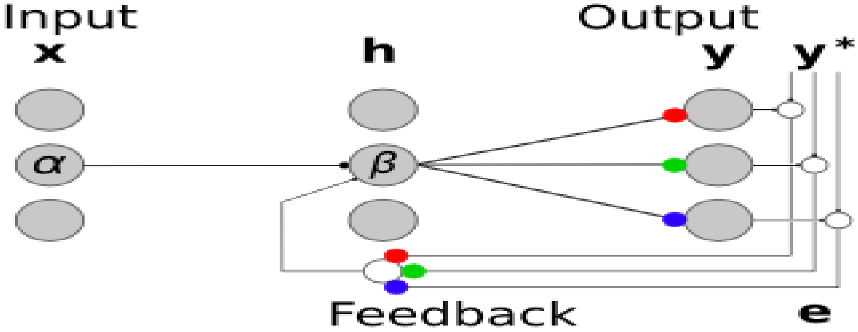
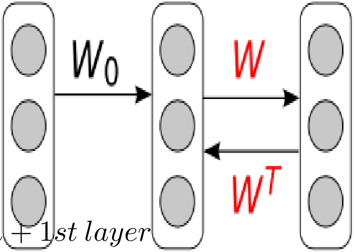


Figure: A sketch of the backprop

Recap backprop



$n\ layer\ MLP$

$\sigma(\mathbf{x}) : activation\ function$

$\mathbf{y}_k = \sigma(\mathbf{x}_k)$

$W_m : a\ weight\ multiplied\ to\ m\ th\ layer\ to\ m+1st\ layer$

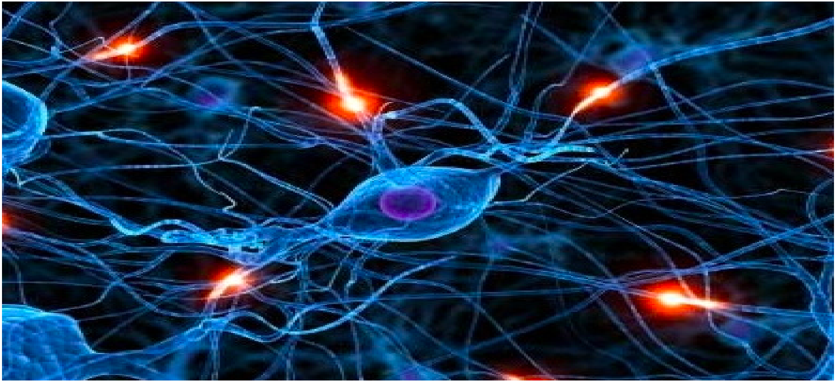
$\delta_n = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_n} = \frac{\partial \mathbf{y}_n}{\partial \mathbf{x}_n} \frac{\partial \mathcal{L}}{\partial \mathbf{y}_n} = diag(\sigma'(\mathbf{x}_n)) \frac{\partial \mathcal{L}}{\partial \mathbf{y}_n}$

$\delta_{k-1} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_{k-1}} = \frac{\partial \mathbf{x}_k}{\partial \mathbf{x}_{k-1}} \frac{\partial \mathcal{L}}{\partial \mathbf{x}_k} = diag(\sigma'(\mathbf{x}_{k-1})) W_{k-1}^t \delta_k$

$\Delta W_k = \frac{\partial \mathcal{L}}{\partial W_k} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_{k+1}} \frac{\partial \mathbf{x}_{k+1}}{\partial W_k} = \delta_{k+1} \mathbf{y}_k^t$

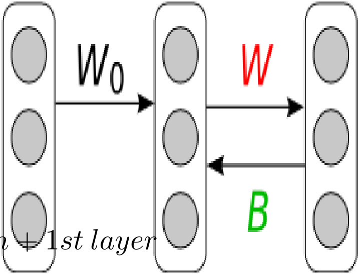
$W_k^{(t+1)} = W_k^{(t)} - \eta \Delta W_k^{(t)}$

Backprop is not the way a brain learns



- Backprop requires a precise transport of synaptic weight information. i.e. impossible
- Synapses in brain functions in a unidirectional way. Not bidirectional.
- Reinforcement learning? Shallow mechanism?
- How about putting a random matrix B to the place of W^t ?

Random feedback alignment



$n\ layer\ MLP$

$\sigma(\mathbf{x}) : activation\ function$

$\mathbf{y}_k = \sigma(\mathbf{x}_k)$

$W_m : a\ weight\ multiplied\ to\ m\ th\ layer\ to\ m+1st\ layer$

$$\delta_n = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_n} = \frac{\partial \mathbf{y}_n}{\partial \mathbf{x}_n}\frac{\partial \mathcal{L}}{\partial \mathbf{y}_n} = diag(\sigma'(\mathbf{x}_n))\frac{\partial \mathcal{L}}{\partial \mathbf{y}_n}$$

$$\delta_{k-1} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_{k-1}} = \frac{\partial \mathbf{x}_k}{\partial \mathbf{x}_{k-1}}\frac{\partial \mathcal{L}}{\partial \mathbf{x}_k} = diag(\sigma'(\mathbf{x}_{k-1}))B_{k-1}\delta_k$$

$$\Delta W_k = \frac{\partial \mathcal{L}}{\partial W_k} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_{k+1}}\frac{\partial \mathbf{x}_{k+1}}{\partial W_k} = \delta_{k+1}\mathbf{y}_k^t$$

$$W_k^{(t+1)} = W_k^{(t)} - \eta \Delta W_k^{(t)}$$

Empirical results of random feedback

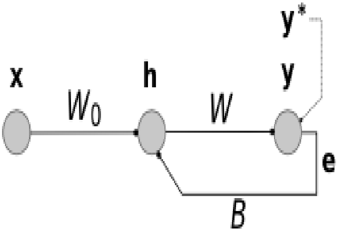
Linear function-approximation

$T : target\ linear\ function$

$\mathbf{y} = W\mathbf{h}$

$\mathbf{h} = W_0\mathbf{x}$

$\mathcal{L} = \frac{1}{2}\mathbf{e}^t\mathbf{e},\ \mathbf{e} = \mathbf{y}^* - \mathbf{y},\ \mathbf{y}^* = T\mathbf{x}$



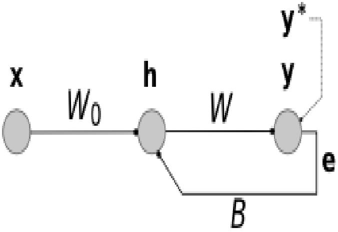
Linear function-approximation

$$\Delta W_0 \propto \frac{\partial \mathcal{L}}{\partial W_0} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial W_0} = -(W^t \mathbf{e}) \mathbf{x}^t$$

$$\Delta W_{\propto} \frac{\partial \mathcal{L}}{\partial W} = -\mathbf{e}\mathbf{h}^t$$

$$\Delta \mathbf{h}_{BP} = W^t e$$

$$\Delta \mathbf{h}_{FA} = Be$$



Linear function-approximation

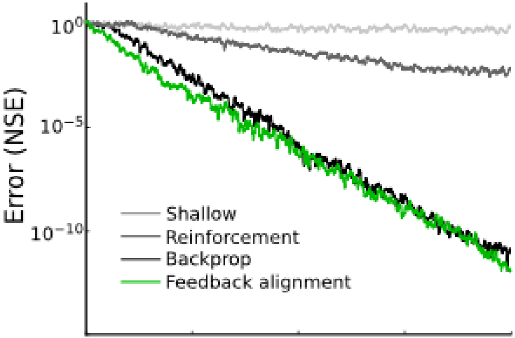


Figure: Four algorithms learn to mimic a linear function

- $NSE(\mathbf{y}^*, \mathbf{y}) = \frac{MSE(\mathbf{y}^*, \mathbf{y})}{MSE(\mathbf{y}^*, 0)}$

Linear function-approximation

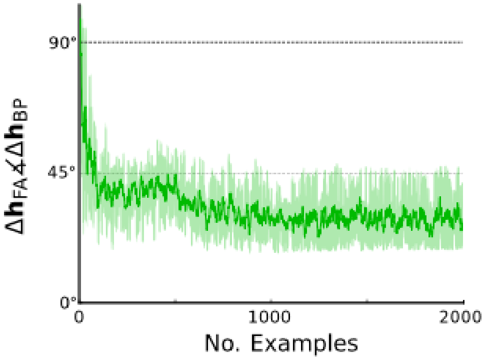


Figure: Angle between the hidden-unit update vector prescribed by feedback alignment and that prescribed by backprop

- $\mathbf{e}^t W B \mathbf{e} > 0$
- This alignment of the Δh's implies that B has begun to act like W^t.

Nonlinear, real-world problems

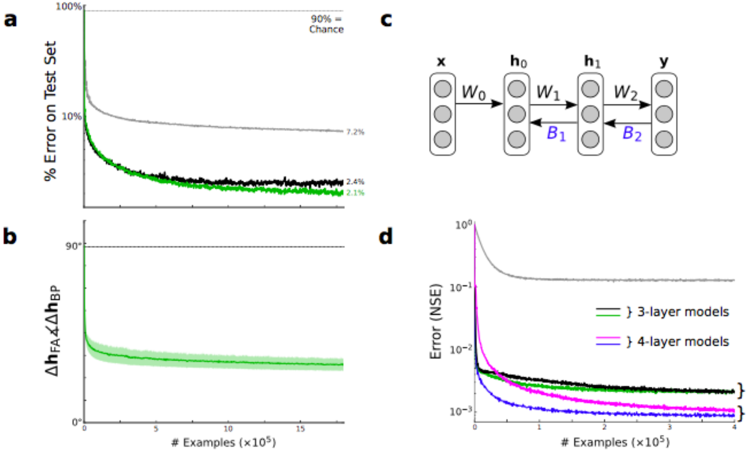


Figure: **a**: MNIST, **b**: angle between FA and BP, **c**: FA in deep network, **d**: nonlinear function-approximation

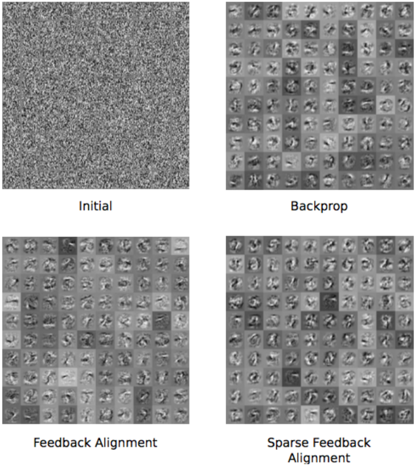


Figure: Receptive fields for 100 randomly selected hidden units

Why it works? Theorems

Why it works?

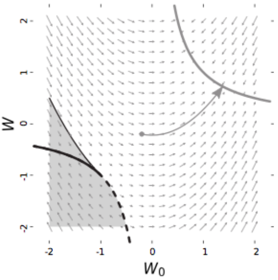


Figure: Vector flow field (small arrows) demonstrates the evolution of W_0 and W during feedback alignment

- Linear function
- $T = 1$ and B is 'randomly' chosen to be 1
- Thick lines are solutions. i.e. $T = WW_0 = 1$
- grey: $\mathbf{e}^t W B \mathbf{e} > 0$, black: $\mathbf{e}^t W B \mathbf{e} < 0$, dash: unstable solution

Why it works?

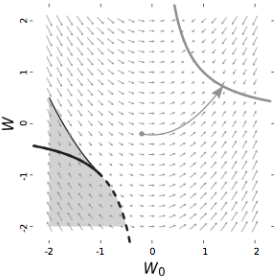


Figure: Vector flow field (small arrows) demonstrates the evolution of W_0 and W during feedback alignment

- Upper right hyperbola is a set of solutions where $W > 0$
- $\mathbf{e}^t W B \mathbf{e} > 0 \, \forall e$
- Meaning W has evolved so that the feedback matrix B is delivering useful teaching signals.

Condition for alignment to reduce error to zero

$$\mathbf{y} = W\mathbf{h}$$

$$\mathbf{h} = A\mathbf{x}$$

$$E := T - WA$$

$$\mathbf{e} = E\mathbf{x}$$

$$\mathcal{L} = \frac{1}{2}\mathbf{e}^t\mathbf{e},\ \mathbf{e} = \mathbf{y}^* - \mathbf{y},\ \mathbf{y}^* = T\mathbf{x}$$

$$\Delta W = \eta\frac{\partial\mathcal{L}}{\partial\mathbf{y}}\frac{\partial\mathbf{y}}{\partial W} = \eta E\mathbf{x}\mathbf{x}^t A^t$$

$$\Delta A = \eta\frac{\partial y}{\partial\mathbf{h}}\frac{\partial\mathcal{L}}{\partial\mathbf{y}}\frac{\partial\mathbf{h}}{\partial A} = \eta BE\mathbf{x}\mathbf{x}^t$$

Condition for alignment to reduce error to zero

Assume $\mathbf{x} \sim iid\,\mathbf{N}(0, I)$.

$$\Delta W = \eta[E\mathbf{x}\mathbf{x}^t A^t] = \eta E A^t$$

$$\Delta A = \eta[BE\mathbf{x}\mathbf{x}^t] = \eta BE$$

$$\dot{W} = EA^t$$

$$\dot{A} = BE$$

Then by integration,

$$BW + W^t B^t = AA^t + C$$

Condition for alignment to reduce error to zero

Theorem (1)

Given the learning dynamics

$$\dot{W} = EA^t$$

$$\dot{A} = BE$$

and assuming that the constant C in the equation is zero and that the matrix B satisfies

$$B^+B = I$$

then

$$\lim_{t\to\infty} E = 0$$

Condition for alignment to reduce error to zero

Sketch of the proof

- Let $V := tr(BEE^tB^t)$
- V is lower bounded, $\dot{V}$ is negative semi-definite, and $\dot{V}$ is uniformly continuous in time, which is satisfied since $\ddot{V}$ is finite.
- $\therefore \dot{V} \to 0$ as $t \to \infty$, which means V converges to some value.
- Since $\dot{V} = -2tr(BEA^tAE^tB^t) - tr(A^tBEE^tB^tA) = 0$, $tr(BEA^tAE^tB^t) = 0$ i.e. $BEA^t = 0$
- $\therefore B^+BEA^t = EA^t = 0$
- Since $\dot{W} = EA^t$, W is constant.
- $\therefore AA^t = BW + W^tB^t$ is constant.
- Since $0 = BEA^t = BTA^t - BWAA^t$, BTA^t is constant.
- $\frac{dBTA^t}{dt} = 0 = BTE^tB^t$
- $0 = BTE^tB^t = TE^t = ET^t$
- $\therefore EE^t = ET^t - EA^tW^t = 0$

Gauss-Newton modification of backprop

$\mathbf{y} = W\mathbf{h}$

$\mathbf{h} = \sigma(A\mathbf{x})$

$E := T - WA$

$\mathbf{e} = E\mathbf{x}$

$\mathcal{L} = \frac{1}{2}\mathbf{e}^t\mathbf{e},\ \mathbf{e} = \mathbf{y}^* - \mathbf{y},\ \mathbf{y}^* = T\mathbf{x}$

$\mathcal{L}_x = \mathbf{e}_x^t\mathbf{e},\ \mathcal{L}_{xx} = \mathbf{e}_{xx}^t\mathbf{e} + \mathbf{e}_x^t\mathbf{e}_x \approx \mathbf{e}_x^t\mathbf{e}_x$ when $\mathbf{e}$ is small.

$\Delta x = -\mathcal{L}_{xx}^{-1}\mathcal{L}_x^t \approx -(\mathbf{e}_x^t\mathbf{e}_x)^{-1}\mathbf{e}_x^t\mathbf{e} = \mathbf{e}_x^+\mathbf{e}$

Gauss-Newton modification of backprop

Theorem (2)

Using W^+ instead of W^t in backprop algorithm makes the algorithm behave like Gauss-Newton method.

Gauss-Newton modification of backprop

Sketch of the proof

- $\Delta\mathbf{h}_{GN} = -\mathbf{e}_h^{+}\mathbf{e} = -W^{+}\mathbf{e}$
- $\Delta A_{BP} = -\eta Diag(\sigma'(\mathbf{x}))W^{t}\mathbf{e}\mathbf{x}^{t}$
- $\Delta A_{PBP} = -\eta Diag(\sigma'(\mathbf{x}))W^{+}\mathbf{e}\mathbf{x}^{t}$
- $\Delta\mathbf{h}_{PBP} = \Delta(\sigma(A\mathbf{x})) \approx Diag(\sigma'(A\mathbf{x}))\Delta(A\mathbf{x}) =$ $Diag(\sigma'(A\mathbf{x}))\Delta A_{PBP}\mathbf{x} = (\mathrm{Diag}(\sigma'(A\mathbf{x})))^{2}\Delta\mathbf{h}_{GN}\mathbf{x}^{\mathbf{t}}\mathbf{x}$
- $\therefore \Delta\mathbf{h}_{PBP} \propto \Delta\mathbf{h}_{GN}$ elementwise.

B acts like the pseudoinverse of W

$\mathbf{y} = W\mathbf{h}$

$\mathbf{h} = A\mathbf{x}$

$E := T - WA$

$\mathbf{e} = E\mathbf{x}$

$\mathcal{L} = \frac{1}{2}\mathbf{e}^t\mathbf{e},\ \mathbf{e} = \mathbf{y}^* - \mathbf{y},\ \mathbf{y}^*$: True value

$W^{(t+1)} = W^{(t)} + \Delta W^{(t)}$

$A^{(t+1)} = A^{(t)} + \Delta A^{(t)}$

B acts like the pseudoinverse of W

$\Delta W = \eta_W \mathbf{e}\mathbf{h}^{\mathbf{t}}$

$\Delta A = \eta_A B \mathbf{e}\mathbf{x}^t$

$\mathbf{h}^{(t+1)} = A^{(t+1)}\mathbf{x} = (A^{(t)} + \Delta A^{(t)})\mathbf{x} = \mathbf{h}^{(t)} + \eta_A B \mathbf{e}\mathbf{x}^t\mathbf{x} = \mathbf{h}^{(t)} + \eta_h B\mathbf{e}$

Let $\eta_h = \eta_W = \eta$. Then,
$W^{(t+1)} = W^{(t)} + \Delta W^{(t)}$

$\mathbf{h}^{(t+1)} = \mathbf{h}^{(t)} + \Delta\mathbf{h}^{(t)}$
with,

$\Delta W = \eta \mathbf{e}\mathbf{h}^{\mathbf{t}}$

$\Delta\mathbf{h} = \eta B\mathbf{e}$

B acts like the pseudoinverse of W

Theorem (3)

When W and A are initialized to zero, hidden unit updates prescribed by the the forward alignment algorithm, $\Delta\mathbf{h}_{FA}$, are always a positive scalar multiple of the hidden unit updates prescribed by the pseudobackprop algorithm, $\Delta\mathbf{h}_{PBP}$.

B acts like the pseudoinverse of W

Sketch of the proof

- $\exists s_h,\ s_W$ s.t. $\mathbf{h} = s_h B\mathbf{y}^*$, $W = s_W \mathbf{y}^*(B\mathbf{y}^*)^t$
- Trivial when $\mathbf{h} = 0, W = 0$. Then by induction, above equations hold for every time step.
- $\mathbf{e} = (1 - s_y)\mathbf{y}^*$ with $(1 - s_y)$ a positive scalar.
- $\therefore \Delta\mathbf{h}_{FA} = \eta B\mathbf{e} = \eta(1 - s_y)B\mathbf{y}^*$ and
  $\Delta\mathbf{h}_{PBP} = \eta W^+\mathbf{e} = \eta(1 - s_y)W^+\mathbf{y}^*$
- It suffices to show that $sB\mathbf{y}^* = (\mathbf{y}^*(B\mathbf{y}^*)^t)^+\mathbf{y}^*$.
- $(\mathbf{y}^*(B\mathbf{y}^*)^t)^+\mathbf{y}^* = (B\mathbf{y}^*)^{t+}\mathbf{y}^{*+}\mathbf{y}^* = (B\mathbf{y}^*)^{t+} = sB\mathbf{y}^*$ where
  $s = ((B\mathbf{y}^*)^t(B\mathbf{y}^*))^{-1}$.
- $\therefore \Delta\mathbf{h}_{FA} = s\Delta\mathbf{h}_{PBP}$

B acts like the pseudoinverse of W