BATCH NORM

→ Avoids the problem of internal co-variate shifts
→ When performing GD update, for a given weight matrix, SGD GD updates assume that for a given W, the other ones are held constant. This is not true in general & therefore gradient steps in earlier layers may affect the statistics of the neurons for the next layer. BN helps normalize these statistics so the G's are scaled well & aren't drastically different after each G step.
→ BN prevents outputs of each layer being too large or too small

BN → linear transformation
→ accelerates training by requiring fewer iterations
→ Introduces noise to h_n activation bc mean & SD are estimated with a mini-batch of data
→ Uses running mean & variance from train. stats
→ Reduces I.C.S
→ Two learnable params (γ, β) - to scale, shift, & normalize values
→ Regularizes, bc adds noise to each mini batch
→ During inference - it uses population stats computed during training on the entire data rather than the mini-batch
→ For deeper networks & CNNs
→ can introduce dependency between examples in a mini-batch

BN	LN
→ Per neuron per batch that is mean & variance calculated for all instances in a batch per channel	→ Mean & variance calculated for a single instance across a layer.
→ Per feature across batch & spatial dims.	→ Per sample across all features
→ Large batches for stable statistics	→ Independent, works with a single sample
→ CNNs	→ RNNs, Transformers
→ uses batch stats. for training & running averages during inference	→ Sample stats for both training & testing
→ Image classification tasks	→ NLP tasks, variable length sequences
	→ Also used when batch size is reduced, for sequential data

This shift in input distribution to hidden layers is called Internal cov. shift.
→ This is where batch normalization helps

layer norm → Statistics of hidden layers
batch norm → Statistics for mini-batch

Batch normalization → addresses the problem of ICS by normalizing the input of each layer

Let $x \in B$ (batch) denote an input to a BN layer
Then it is transformed as follows.

training samples → mini batch

$$BN(x) = \gamma \odot \frac{x - \mu_B}{\sigma_B} + \beta$$

statistics of mini batch

$$\mu_B = \frac{1}{|B|} \sum_{x \in B} x$$ Minibatch sample mean (samples in mini-batch)

→ W/out the scaling & shifting parameters, BN would always follow a unit variance & 0 mean distribution.

$$\sigma_B^2 = \frac{1}{|B|} \sum_{x \in B} (x - \mu_B)^2 + \varepsilon$$ Minibatch sample variance → for avoiding addition to 0.

→ β & γ help prevent constraint allowing the network to flexibly adjust feature representations.

γ → Scaling parameter
β → Shifting parameter
→ Learnable parameters

how we apply it

$$\boxed{h_i = f(BN(W_i h_{i-1} + b_i))}$$ matrix × vector × vector

a) Regularization
→ Helpful for better generalization
→ Improves test set prediction

b) Mini batch SGD
→ Its a noisy approximation because only use a few samples
→ Only a part of the training data is involved

c) Adagrad optimizer
→ has equal or smaller successive LRs.
→ The gradient history stays the same or increases in each dimension for every iteration of Adagrad

d) SGD $\varepsilon = 5 \times 10^{-b}$
→ Smaller LR's usually lead to getting stuck at a local minima
→ Slower convergence
→ Poor generalization

$$\mathcal{L}(w,b) = \frac{1}{k} \sum_{i=1}^{k} \text{hinge}_{y^{(i)}}(x^{(i)}) + \lambda \|w\|_1$$

$$\nabla_w \mathcal{L}(w,b) = \frac{1}{k} \sum_{i=1}^{k} \nabla_w \text{hinge}_{y^{(i)}}(x^{(i)}) + \lambda \nabla_w \|w\|_1$$

Linear

2-terms to compute the gradients for

$$\text{hinge}_{y^{(i)}}(x^{(i)}) = \max(0, 1 - y^{(i)}(w^T x^{(i)} + b))$$ Piece-wise linear function

$$= \begin{cases} 0, & y^{(i)}(w^T x^{(i)} + b) > 1 \\ 1 - y^{(i)}(w^T x^{(i)} + b), & y^{(i)}(w^T x^{(i)} + b) < 1 \end{cases}$$

If $y^{(i)}(w^T x^{(i)} + b) > 1$, then $\nabla_w \text{hinge}_{y^{(i)}}(x^{(i)}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^d$ (vector of 0's)

If $y^{(i)}(w^T x^{(i)} + b) < 1$, then $\nabla_w \text{hinge}_{y^{(i)}}(x^{(i)}) = -y^{(i)} x^{(i)}$

} yellow term gradients

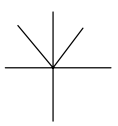
Putting all together, $\nabla_w \text{hinge}_{y^{(i)}}(x^{(i)}) = \underbrace{\mathbb{1}\{y^{(i)}(w^T x^{(i)} + b) < 1\}}_{\text{vector of all 1's}} \odot (-y^{(i)} x^{(i)})$

$\underbrace{y^{(i)}(w^T x^{(i)} + b) > 1}_{\text{vector of all 0's}}$

$$\|w\|_1 = \sum_{i=1}^{d} |w_i|$$

$$\nabla_w \|w\|_1 = \begin{bmatrix} \nabla_{w_1} |w_1| \\ \nabla_{w_2} |w_2| \\ \vdots \\ \nabla_{w_d} |w_d| \end{bmatrix}$$

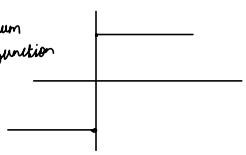
$$|w_i| = \begin{cases} w_i, & w_i > 0 \\ -w_i, & w_i < 0 \end{cases}$$



$$\nabla_{w_i} |w_i| = \begin{cases} 1, & w_i > 0 \\ -1, & w_i < 0 \end{cases}$$

Sub-gradient - anything between 1 & -1

Signum function



If $w_i = 0$, $\nabla_{w_i}(w_i) = 0$

$$\nabla_{w_i} |w_i| = \text{Sgn}(w_i)$$

$$\nabla_w \|w\|_1 = \text{Sgn}(w) \in \mathbb{R}^d$$

$$\nabla_w \mathcal{L}(w,b) = \frac{1}{k} \sum_{i=1}^{k} \mathbb{1}\{y^{(i)}(w^T x^{(i)} + b) < 1\} \odot (-y^{(i)} x^{(i)}) + \lambda \, \text{Sgn}(w)$$

$$\arg\min_\theta \; \frac{1}{2} \sum_{i=1}^{N} (y^{(i)} - \theta^T \hat{x}^{(i)})^2 + \frac{\lambda}{2} \|\theta\|_2^2$$

$$\mathcal{L} = \frac{1}{2} \sum_{i=1}^{N} (y^{(i)} - \theta^T \hat{x}^{(i)})^T (y^{(i)} - \theta^T \hat{x}^{(i)}) + \frac{\lambda}{2} \|\theta\|_2^2$$

$$= \frac{1}{2} \sum_{i=N}^{N} (y^{(i)T} y^{(i)} - y^{(i)T} \theta^T \hat{x}^{(i)} - \theta \hat{x}^{(i)T} y^{(i)} + \theta \hat{x}^{(i)T} \theta^T \hat{x}^{(i)}) + \frac{\lambda}{2} \|\theta\|_2^2$$

$$= \frac{1}{2} \sum_{i=N}^{N} (y^{(i)T} y^{(i)} - 2 y^{(i)T} x^{(i)} \theta + \theta^T \hat{x}^{(i)T} \hat{x} \theta) + \frac{\lambda}{2} \|\theta\|_2^2$$

Let, $\mathbb{X} = [x^{(1)}, x^{(2)}, x^{(3)}, \ldots x^{(i)}]$ → data

$\mathbb{Y} = [y^{(1)}, y^{(2)}, y^{(3)}, \ldots y^{(i)}]$ → output, be matrices

$$= \frac{1}{2} (\mathbb{Y}^T \mathbb{Y} + 2 \mathbb{Y}^T \mathbb{X} \theta + \theta^T \mathbb{X}^T \mathbb{X} \theta) + \frac{\lambda}{2} \|\theta\|_2^2$$

$$\nabla_\theta \mathcal{L} = \frac{1}{2} (-2 \mathbb{X}^T \mathbb{Y} + 2 \mathbb{X}^T \mathbb{X} \theta) + \lambda \theta$$

$$= -\mathbb{X}^T \mathbb{Y} + \mathbb{X}^T \mathbb{X} \theta + \lambda \theta$$

$$\mathbb{X}^T \mathbb{Y} = \theta (\mathbb{X}^T \mathbb{X} + \lambda I)$$

$$\theta = (\mathbb{X}^T \mathbb{X} + \lambda I)^{-1} \mathbb{X}^T \mathbb{Y}$$

$$\mathcal{L}(w,b) = \frac{1}{k} \sum_{i=1}^{k} \text{hinge}_{y^{(i)}}(x^{(i)})$$

$$= \max(0, 1 - y^{(i)}(w^T x^{(i)} + b))$$

we can treat it as a piece-wise function

$$\text{hinge}_{y^{(i)}}(x^{(i)}) = \begin{cases} 0, & \text{If } y^{(i)}(w^T x^{(i)} + b) \geq 1 \\ 1 - y^{(i)}(w^T x^{(i)} + b, & \text{If } y^{(i)}(w^T x^{(i)} + b) < 1 \end{cases}$$

When $y^{(i)}(w^T x^{(i)} + b) < 1$,

$$\nabla_w \text{hinge}(x^{(i)}) = -y^{(i)} x^{(i)}$$

$$\therefore \nabla_w \mathcal{L} = \frac{1}{k} \sum_{i=k}^{k} \mathbb{1}\{y^{(i)}(w^T x^{(i)} + b) < 1\} (-y^{(i)} x^{(i)})$$

$$\nabla_b \mathcal{L} = \frac{1}{k} \sum_{i=k}^{k} \mathbb{1}\{y^{(i)}(w^T x^{(i)} + b) < 1\} (-y^{(i)})$$

SOFTMAX CLASSIFIER

$$\mathcal{L} = \sum_{i=1}^{m} \left(\log \sum_{j=1}^{c} e^{a_j(x^{(i)})} - a_{y^{(i)}}(x^{(i)}) \right)$$

$$a_j(x^{(i)}) = w_j^T x^{(i)} + b_j$$

$\nabla_{w_k} \mathcal{L}$ & $\nabla_b \mathcal{L}$

$$\frac{\partial \mathcal{L}}{\partial w_k} = \sum_{i=1}^{m} \frac{\partial}{\partial w_k} \left(\log \sum_{j=1}^{c} e^{a_j(x^{(i)})} - a_{y^{(i)}}(x^{(i)}) \right)$$

using chain rule: $\log \sum_{i=1}^{m} \frac{\partial}{\partial w_k} e^{a_j(x^{(i)})} = \frac{1}{\sum_{j=1}^{c} e^{a_j(x^{(i)})}} \cdot \frac{\partial}{\partial w_k} \sum_{j=1}^{c} e^{a_j(x^{(i)})}$

For each term in the summation: $\frac{\partial}{\partial w_k} e^{a_j(x^{(i)})} = e^{a_j(x^{(i)})} \frac{\partial}{\partial w_k} a_j(x^{(i)})$

$$\frac{\partial}{\partial w_k} a_j(x^{(i)}) = \begin{cases} x^{(i)}, & \text{if } j = k \\ 0, & \text{otherwise} \end{cases} \qquad \frac{\partial}{\partial w_k} (w_j^T x^{(i)} + b_j) = x^{(i)}$$

Therefore, $\frac{\partial}{\partial w_k} \sum_{j=1}^{c} e^{a_j(x^{(i)})} = e^{a_k(x^{(i)})} x^{(i)}$

$$\frac{\partial}{\partial w_k} \log \sum_{j=1}^{c} e^{a_j(x^{(i)})} = \frac{e^{a_j(x^{(i)})}}{\sum_{j=1}^{c} e^{a_j(x^{(i)})}} x^{(i)}$$

$$\text{Softmax}_k(x^{(i)}) = \frac{e^{a_k(x^{(i)})}}{\sum_{j=1}^{c} e^{a_j(x^{(i)})}}$$

For true class $y^{(i)}$: $\frac{\partial}{\partial w_k} a_{y^{(i)}}(x^{(i)}) = 1(k = y^{(i)}) \cdot x^{(i)}$

$$\frac{\partial}{\partial b_k} a_j(x^{(i)}) = \begin{cases} 1, & \text{if } i = k, \\ 0, & \text{otherwise} \end{cases} \qquad \frac{\partial}{\partial b_k} (w_j^T x^{(i)} + b^{(i)}) = 1$$

$$\therefore \frac{\partial \mathcal{L}}{\partial w_k} = \sum_{i=1}^{m} (\text{softmax}(x^{(i)}) - 1(k = y^{(i)}))\, x^{(i)}$$

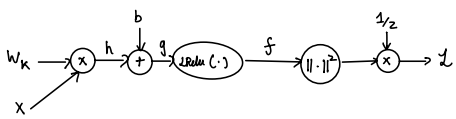
$$\frac{\partial \mathcal{L}}{\partial b_k} = \sum_{i=1}^{m} \text{Softmax}(x^{(i)}) - 1(k = y^{(i)})$$

$$\mathcal{L} = \frac{1}{2} \|f - y\|^2$$

$f = \max(\alpha \cdot g, g), 0 < \alpha < 1$ Leaky Relu

$g = h + b$

$h = W_K x$



b. $\frac{\partial \mathcal{L}}{\partial f_i} = \frac{\partial g}{\partial f_i} \frac{\partial \mathcal{L}}{\partial g} = \frac{\partial}{\partial f_i} \left[\frac{1}{2} (f-y)^T (f-y) \right]$

EXPANSION

$= \frac{\partial}{\partial f_i} \left[\frac{1}{2} (f^T f - 2 f y^T - y^T y) \right] = \frac{1}{2} (f_i - 2 y_i) = f_i - y_i$

$$\frac{\partial \mathcal{L}}{\partial f} = f - y$$

$$\frac{\partial \mathcal{L}}{\partial g} = \frac{\partial f}{\partial g} \frac{\partial \mathcal{L}}{\partial f}$$

Leaky Relu

$$\frac{\partial \mathcal{L}}{\partial g_i} = \frac{\partial}{\partial g} [\max(\alpha g_i, g_i)] = \begin{cases} 0 & \text{if } i \neq j \\ \alpha & \text{if } i = j \ \& \ \alpha g_j > g_j \\ 1 & \text{if } i = j \ \& \ \alpha g_j > g \end{cases}$$

$$\therefore \frac{\partial f}{\partial g} = \text{diag}(I(\alpha g_1 > g_1)\alpha + (1 - I(\alpha g_1 > g_1)), \ldots, I(\alpha g_m > g_m)\alpha + (1 - I(\alpha g_m > g_m))$$

$$\frac{\partial \mathcal{L}}{\partial h} = \frac{\partial g}{\partial h} \frac{\partial \mathcal{L}}{\partial g} = \frac{\partial}{\partial h_i} (h_j + b_j) = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{else} \end{cases}$$

$\therefore \frac{\partial g}{\partial h} = I$ and $\frac{\partial g}{\partial b} = I$ $\therefore \frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial h} = \frac{\partial \mathcal{L}}{\partial g}$

$$\frac{\partial \mathcal{L}}{\partial W_k} = \frac{\partial h}{\partial W_k} \frac{\partial \mathcal{L}}{\partial h} = \frac{\partial \mathcal{L}}{\partial h} x^T$$

---

b) $z_1 = W x^{(i)} + b_1$

$a_1 = \text{Relu}(z_1)$

$z_2 = W_2 a_1 + b_2$

$\hat{y}^{(i)} = \sigma(z_2)$

BCE

$\mathcal{L} = \alpha \cdot y^{(i)} \cdot \log(\hat{y}^{(i)}) + \beta (1 - y^{(i)}) \cdot \log(1 - \hat{y}^{(i)})$

$J = -\frac{1}{m} \sum_{i=1}^{m} \mathcal{L}^{(i)}$

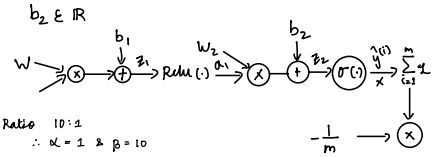
$\hat{y}^{(i)} \in \mathbb{R}$, $y^{(i)} \in \mathbb{R}$, $x^{(i)} \in \mathbb{R}^{D_x \times 1}$

$W_1 \in \mathbb{R}^{D_{a1} \times D_x}$, $W_2 \in \mathbb{R}^{1 \times D_{a1}}$

$W_1 x^{(i)} \to \mathbb{R}^{D_{a1} \times D_x} \cdot \mathbb{R}^{D_x \times 1}$

$b_1 \in \mathbb{R}^{D_{a1}}$

$b_2 \in \mathbb{R}$



iii. Ratio 10:1

$\therefore \alpha = 1$ & $\beta = 10$

iv. $\nabla_{\hat{y}^{(i)}} \mathcal{L}^{(i)}$ & $\delta_{\hat{y}^{(i)}}$

$\mathcal{L} = \alpha \cdot y^{(i)} \cdot \log(\hat{y}^{(i)}) + \beta (1 - y^{(i)}) \cdot \log(1 - \hat{y}^{(i)})$

$\nabla_{\hat{y}^{(i)}} = \frac{\partial \mathcal{L}}{\partial \hat{y}^{(i)}}$ $\quad \frac{d}{d\hat{y}^{(i)}} \log(\hat{y}^{(i)}) = \frac{1}{\hat{y}^{(i)}}$ $\quad \frac{d}{d\hat{y}^{(i)}} \log(1 - \hat{y}^{(i)}) = \frac{-1}{1 - \hat{y}^{(i)}}$

By linearity & chain rule,

$$\frac{\partial \mathcal{L}}{\partial \hat{y}^{(i)}} = \alpha \cdot \frac{y^{(i)}}{\hat{y}^{(i)}} - \beta \frac{1 - y^{(i)}}{1 - \hat{y}^{(i)}} = \delta_{\hat{y}^{(i)}}$$

v. $\nabla_{b_2} \mathcal{L}^{(i)} \to \delta_{b_2}$ $\quad \frac{\partial \hat{y}^{(i)}}{\partial z_2} = \sigma(z_2)[1 - \sigma(z_2)]$

back prop to z_2: $\frac{\partial \mathcal{L}}{\partial z_2} = \frac{\partial \hat{y}^{(i)}}{\partial z_2} \frac{\partial \mathcal{L}}{\partial \hat{y}^{(i)}} = \sigma(z_2)[1 - \sigma(z_2)]\, \delta_{\hat{y}^{(i)}}$

gradient passes through the (+) operator: $\frac{\partial \mathcal{L}}{\partial b_2} = \sigma(z_2)[1 - \sigma(z_2)]\, \delta_{\hat{y}^{(i)}}$

vi. $\nabla_{W_2} \mathcal{L}^{(i)} \to \delta_{W_2}$

$$\frac{\partial \mathcal{L}}{\partial W_2} = \frac{\partial b_2}{\partial W_2} \frac{\partial \mathcal{L}}{\partial b_2} = a_1^T \cdot \delta_{b_2} = \delta_{W_2}$$

vii. $\nabla_{b_1} \mathcal{L}^{(i)} \to \delta_{b_1}$

$$\frac{\partial \mathcal{L}}{\partial a_1} = \frac{\partial W_2}{\partial a_1} \frac{\partial \mathcal{L}}{\partial W_2} = W_2^T \cdot \delta_{W_2}$$

$$\frac{\partial \mathcal{L}}{\partial b_1} = \frac{\partial a_1}{\partial b_1} \frac{\partial \mathcal{L}}{\partial a_1} = \mathbb{I}(z_1 > 0) \odot W_2^T \delta_{b_2}$$

viii. $\nabla_{W_1} \mathcal{L}^{(i)} \to \delta_{W_1}$

operator ⊗: $\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial z_1}{\partial W_1} \frac{\partial \mathcal{L}}{\partial z_1} = \delta_{b_1} x^{(i)T}$

BCE Loss:

we know,

$$Pr(y^{(i)} = 1 \mid x^{(i)}, \theta) = \hat{y}^{(i)}$$

$$Pr(y^{(i)} = 0 \mid x^{(i)}, \theta) = 1 - \hat{y}^{(i)}$$

The following exp,

$$P(y^{(i)} \mid x^{(i)}, \theta) = (\hat{y}^{(i)})^{y^{(i)}} \cdot (1 - \hat{y}^{(i)})^{1 - y^{(i)}}$$

captures both the cases

$$\mathcal{L}(\theta) = \prod_{i=1}^{N} P(x^{(i)}, y^{(i)} \mid \theta) = \prod_{i=1}^{N} P(x^{(i)} \mid \theta)\, P(y^{(i)} \mid x^{(i)}, \theta)$$

$$= \prod_{i=1}^{N} P(x^{(i)} \mid \theta)\, P(y^{(i)} = 1 \mid x^{(i)}, \theta)^{y^{(i)}} (1 - P(y^{(i)} = 1 \mid x^{(i)}, \theta))^{(1 - y^{(i)})}$$

we take the log on both sides

$$\arg\max_\theta \log \mathcal{L}(\theta) = \arg\max_\theta \sum_{i=1}^{N} \log P(y^{(i)} \mid x^{(i)}, \theta)$$

$$= \arg\max_\theta \sum_{i=1}^{N} \log P(y^{(i)} = 1 \mid x^{(i)}, \theta)^{y^{(i)}} + \log (1 - P(y^{(i)} = 1 \mid x^{(i)}, \theta))^{(1 - y^{(i)})}$$

$$= \arg\max_\theta \sum_{i=1}^{N} y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \cdot \log(1 - \hat{y}^{(i)})$$

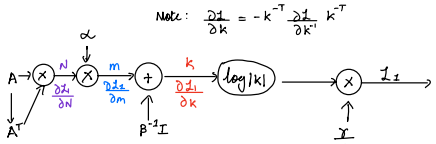
$$\mathcal{L} = -C - \frac{D}{2} \log|K| - \frac{1}{2} tr(K^{-1} Y Y^T)$$

$K = \alpha x x^T + \beta^{-1} I$

$\mathcal{L}_1 = -\frac{D}{2} \log|\alpha x x^T + \beta^{-1} I|$

$\mathcal{L}_2 = -\frac{1}{2} tr((\alpha x x^T + \beta^{-1} I)^{-1} Y Y^T)$

Note: $\frac{\partial \mathcal{L}}{\partial K} = -K^{-T} \frac{\partial \mathcal{L}}{\partial K^{-1}} K^{-T}$



$\mathcal{L}_1 = -\frac{\gamma}{2} \log|K| = -\frac{\gamma}{2} \log(\det(K))$

$\frac{\partial \mathcal{L}_1}{\partial K} = -\frac{\gamma}{2} (K^{-1})^T = -\frac{\gamma}{2} (K^T)^{-1}$

$\mathcal{L}_1 = \frac{\gamma}{2} \log(x)$ ⊕ gradient flows: $\frac{\partial \mathcal{L}_1}{\partial m} = -\frac{\gamma}{2} (K^T)^{-1}$

$K = mA$ ⊗ Switches gradients: $\frac{\partial \mathcal{L}_1}{\partial N} = -\frac{\alpha\gamma}{2} (K^T)^{-1}$

$N = AA^T$ $\quad \frac{\partial \mathcal{L}_1}{\partial A} = \frac{\partial \mathcal{L}_1}{\partial N} (A^T)^T = -\frac{\alpha\gamma}{2} (K^T)^{-2} \cdot A$

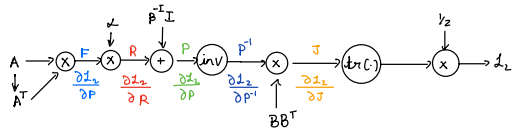
$\frac{\partial \mathcal{L}_1}{\partial A^T} = A^T \frac{\partial \mathcal{L}_1}{\partial N} = A^T \left[-\frac{\alpha\gamma}{2} (K^T)^{-2} \right] = -\frac{\alpha\gamma}{2} (K^{-1}A)^T$

Adding the gradient paths through A & A^T we get

$\frac{\partial \mathcal{L}_1}{\partial A} = -\frac{\alpha\gamma}{2} (K^T)^{-2} A - \left[\frac{\alpha\gamma}{2} (K^{-1}A)^T \right]^T$

K is symmetric

$\frac{\partial \mathcal{L}_1}{\partial A} = -\alpha\gamma K^{-1} A$



$\mathcal{L}_2 = -\frac{1}{2} tr(J)$

$\frac{\partial \mathcal{L}_2}{\partial J} = -\frac{1}{2} I$

$J = P^{-1}(BB^T)$

$\frac{\partial \mathcal{L}_2}{\partial P^{-1}} = \frac{-1}{2} (BB^T)$

$\frac{\partial \mathcal{L}_2}{\partial P} = -P^{-1} \frac{\partial \mathcal{L}_2}{\partial P^{-1}} P^{-1}$

$\frac{\partial \mathcal{L}_2}{\partial R} = -P^{-1} \frac{\partial \mathcal{L}_2}{\partial P^{-1}} P^{-1}$

$\frac{\partial \mathcal{L}_2}{\partial F} = -\alpha P^{-1} \frac{\partial \mathcal{L}_2}{\partial P^{-1}} P^{-1}$

$F = AA^T$

$\frac{\partial \mathcal{L}_2}{\partial A} = -\alpha P^{-1} \frac{\partial \mathcal{L}_2}{\partial P^{-1}} P^{-1} A$

$\frac{\partial \mathcal{L}_2}{\partial A} = -\frac{\alpha}{2} P^{-1} BB^T P^{-1} A$

$\frac{\partial \mathcal{L}_2}{\partial A^T} = -\frac{\alpha}{2} A^T P^{-1} BB^T P^{-1}$

$= \frac{\partial \mathcal{L}_2}{\partial A} + \left[\frac{\partial \mathcal{L}_2}{\partial A^T} \right]^T$

P is symmetric ∴

$= -\alpha P^{-1} BB^T P^{-1} A$

$\Omega(\theta) = \|W\|_1 = \sum_{i,j} |W_{i,j}|$

$\tilde{J}(W) = \alpha \|W\|_1 + J(W)$

$\Omega(\theta) = \frac{1}{2} \|W\|_2 = \frac{1}{2} \sum W_{ij}^2$

$\tilde{J}(\theta) = \frac{\alpha}{2} W^T W + J(\theta)$

$\nabla_W \tilde{J}(\theta) = \alpha W + \nabla_W J(\theta)$

$W \leftarrow W - \varepsilon(\alpha W + \nabla_W J(\theta))$

$\leftarrow (1 - \varepsilon\alpha) W - \varepsilon \nabla_W J(\theta)$

weight decay term

→ weight matrix shrinks by a constant factor on each training iteration before applying the gradient update.

→ Leads to weight matrices with smaller & more diffuse entries

---

correction factor γ_1:

$a_t = (1 - \beta_2) \sum_{i=1}^{t} \beta^{t-i} g_i^2$ $\quad \gamma_1 = \frac{1}{1 - \beta_1^t}$ $\quad \gamma_1 \mathbb{E}[v_t] = \mu = \mathbb{E}[g_t]$

$\mathbb{E}[v_t]$ & compare with μ

$$V_t = (1 - \beta_2) \sum_{i=0}^{t-1} \beta_1^i g_{t-i}$$

$$\mathbb{E}[v_t] = (1 - \beta_1) \sum_{i=0}^{t-1} \beta_1^i \mathbb{E}[g_{t-i}] = (1 - \beta_1) \frac{1 - \beta_1^t}{1 - \beta_1} \mu = (1 - \beta_1^t) \mu = \frac{1}{\gamma_1} \mu$$

$$\mathbb{E}[g_t] = \frac{1}{\gamma_1} S$$

b) L_∞ Norm

$$\mathcal{L}_{reg}(\theta) = \mathcal{L}(\theta) + R(w)$$

$$R(w) = \frac{\lambda}{n}\sum_{i=1}^{n} \|w_i\|_\infty$$ — Gradient non zero only for k^{th} row

$$\|A\|_\infty = \max_{1\le k\le p} \sum_{j=1}^{m} |A_{kj}|$$ — max row sum for absolute values in w_i

$$R(w) = \frac{\lambda}{n}\sum_{i=1}^{n}\left(\max_{1\le c\le p}\sum_{c=1}^{p}|w_{ic}^{i}|\right)$$

$$R(w) = \frac{\lambda}{n}\sum_{i=1}^{n}\sum_{c=1}^{p}|w_{jc}^{i}|$$

$$\frac{\partial R(w)}{\partial w_i} = \frac{\lambda}{n}\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \text{Sign}(w_{ij}^{i}) & \cdot & \cdots & \cdot \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$= \frac{\lambda}{n}\begin{bmatrix} 0 \\ 0 \\ \text{Sign}(w_{ij}^{i}) \\ 0 \end{bmatrix}$$

$$= \frac{\lambda}{n}\cdot \begin{cases} \text{Sign}(w_i[k^*, j]), & k^* \in m \\ 0 \end{cases}$$

L_1

$$\mathcal{L}_{reg}(w) \approx L(w^*) + \alpha\|w\|_1$$

$$w^* = \arg\min_w \mathcal{L}(w), \quad \nabla_w \mathcal{L}(w^*) = 0$$

$$\nabla_{w_i}\mathcal{L}_{reg}(w) = \nabla_w \mathcal{L}(w^*) + \nabla_w\left[\tfrac{1}{2}(w-w^*)^T H (w-w^*)\right] + \alpha\nabla_w\|w\|_1$$

Since $H = \text{diag}(h_1, h_2, \cdots, h_n)$, we can write the gradients in terms of w_i.

$$\nabla_{w_i}\mathcal{L}_{reg}(w) = \nabla_{w_i}\left[\tfrac{1}{2}h_i(w-w^*)^2\right] + \alpha\nabla_{w_i}\|w_i\|, \quad i = 1, 2, \cdots, n$$

$$= h_i(w-w^*) + \alpha\,\text{Sign}(w_i), \text{ for } i = 1, 2, \cdots n$$

Let $g(w_i) = h_i(w_i - w^*) + \alpha\,\text{Sign}(w_i)$

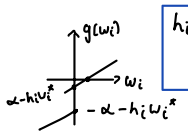
$$= h_i w_i + \alpha\,\text{Sign}(w_i) - h_i w_i^*$$

We want to find the solution $g(w_i) = 0$

$$w_i = \begin{cases} w_i^* - \frac{\alpha}{h_i}, & w_i^* \ge \frac{\alpha}{h_i} \\ 0, & -\frac{\alpha}{h_i} < w_i^* < \frac{\alpha}{h_i} \\ w_i^* + \frac{\alpha}{h_i}, & w_i^* \le -\frac{\alpha}{h_i} \end{cases}$$

Case 1: $w_i^* > 0$

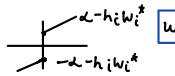
If $\alpha - h_i w_i^* \le 0$



$h_i w_i + \alpha - h_i w_i^* = 0$

$w_i = w_i^* - \frac{\alpha}{h_i}$

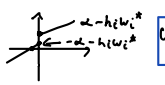
If $\alpha - h_i w_i^* > 0$



$w_i = 0$

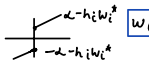
Case 2: $w_i^* < 0$

If $-\alpha - h_i w_i^* > 0$



$w_i = w_i^* + \frac{\alpha}{h_i}$

If $\alpha - h_i w_i^* < 0$



$w_i = 0$

L_2

$$\tilde{J} = J + \|w_2\|_2^2$$

$$\theta_{t+2} \leftarrow \theta_{t+1} + \varepsilon\nabla_\theta J(\theta)$$

$$w_2^{(k+1)} \leftarrow w_2^{(k+1)} + \varepsilon\left(\nabla_{w_2} J^k + 2\,w_2^k\right]$$

GD for ridge regression

b) $\tilde{\mathcal{L}}(\theta) = \mathcal{L}(\theta) + \frac{\alpha}{2}\|\theta\|_2^2$

$$\tilde{\mathcal{L}}(\theta) = \mathcal{L}(\theta) + \frac{\alpha}{2}\theta^T\theta$$

$$\frac{\partial \tilde{\mathcal{L}}}{\partial\theta} = \frac{\partial}{\partial\theta}\tilde{\mathcal{L}}\theta = \nabla_\theta\mathcal{L}(\theta) + \alpha\theta$$

$$\theta \leftarrow \theta - \varepsilon(\nabla_\theta\tilde{\mathcal{L}}(\theta)$$

$$\leftarrow \theta - \varepsilon[\nabla_\theta\mathcal{L}(\theta) + \alpha\theta]$$

$$\leftarrow (1-\varepsilon\alpha)\theta - \varepsilon\nabla_\theta\mathcal{L}(\theta)$$

GD for Linear Regression

$$\mathcal{L}(\theta) = \frac{1}{2}(y - X\theta)^T(y - X\theta)$$

$$\mathcal{L}(\theta) = \frac{1}{2}(y^Ty - 2\,y^TX\theta + \theta^TX^TX\theta)$$

$$\frac{\partial\mathcal{L}}{\partial\theta} = \frac{1}{2}(-2X^Ty + 2X^TX\theta)$$

$$= -X^Ty + X^TX\theta$$

$$\theta \leftarrow \theta - \varepsilon(-X^Ty + X^TX\theta)$$

mini-batch GD + momentum

$$\mathbb{E}(\theta_0) = 0$$

$$\text{Cov}(\theta_0) = \eta^2 I, \quad \theta \in \mathbb{R}^D$$

$$\mathbb{E}(g_k) = \mu$$

$$V_0 = 0$$

Express θ_t in terms of θ_0, α and $g_1, g_2, \cdots g_t$

GD + momentum

$$V_t = \alpha V_{t-1} + \varepsilon g_t$$

$$\theta_t = \theta_{t-1} + V_t$$

$$V_t = -\varepsilon\sum_{i=1}^{t}\alpha^{t-i}g_i$$

$$\theta_t = \theta_0 + \sum_{i=1}^{t} v_i$$

$$\theta_t = \theta_0 - \sum_{i=1}^{t}\left(\sum_{i=1}^{j}\alpha^{j-i}g_i\right)$$

Recursive eqn

$V_t = \alpha V_{t-1} - \varepsilon g$

$V_{t-1} = \alpha V_{t-2} - \varepsilon g_{t-1}$

$V_t = \alpha(\alpha V_{t-2} - \varepsilon g_{t-1}) - \varepsilon g_t$

$= \alpha^2 V_{t-2} - \alpha\varepsilon g_{t-1} - \varepsilon g_t$

$$\mathbb{E}(\theta_t) = \mathbb{E}(\theta_0) - \varepsilon\mathbb{E}\left(\sum_{i=1}^{t}\sum_{i=1}^{j}\alpha^{j-i}g_i\right)$$

$$\mathbb{E}(\theta_t) = \mathbb{E}(\theta_0) - \varepsilon\sum_{i=1}^{t}\sum_{i=1}^{j}\alpha^{j-i}\mathbb{E}(g_i)$$

$$= -\varepsilon\sum_{i=1}^{t}\sum_{i=1}^{j}\alpha^{j-i}\mu$$

$$\mathbb{E}(\theta_t) = -\varepsilon\mu\sum_{j=1}^{t}(1+\alpha+\alpha^2\cdots\alpha^{j-1})$$

$$= -\varepsilon\mu\sum_{j=1}^{t}\frac{(1-\alpha^j)}{(1-\alpha)}$$

$$= \frac{-\varepsilon\mu}{(1-\alpha)}\sum_{j=1}^{t}(1-\alpha^j)$$

5. $\mathbb{E}(g_t) = \mathbb{E}(v_t) = \mu = 0$

V_t -> running average of momentum

$$\mathbb{E}(V_t V_t^T)$$

$$V_t = -\varepsilon\sum_{i=1}^{t}\alpha^{t-i}g_i \quad \text{recursively}$$

$$\text{Cov}(V_t) = \mathbb{E}(V_t V_t^T) \quad (0 \text{ mean for momentum update})$$

$$= \varepsilon^2\mathbb{E}\left(\sum_{i=1}^{t}\alpha^{t-i}g_i\left[\sum_{j=1}^{t}\alpha^{t-j}g_j\right]^T\right)$$

$$= \varepsilon^2\mathbb{E}\left(\sum_{i=1}^{t}\sum_{j=1}^{t}\alpha^{t-i}\alpha^{t-j}g_i g_j^T\right)$$

$$= \varepsilon^2\sigma^2 I\sum_{i=1}^{t}\alpha^{2(t-i)}$$

$$\text{Cov}(\hat{V}_t) = \varepsilon^2\sigma^2\left(\frac{1-\alpha^{2t}}{1-\alpha^2}\right)I$$

Recall that batch normalization transforms the activation of a single neuron over B examples in a batch, $\{x^{(1)}, x^{(2)}, \ldots, x^{(B)}\}$ into $\{y^{(1)}, y^{(2)}, \ldots, y^{(B)}\}$ according to the following set of equations:

BATCH NORM

$$\mu_B = \frac{1}{B}\sum_{j=1}^{B}x^{(j)}$$

$$\sigma_B^2 = \frac{1}{B}\sum_{j=1}^{B}(x^{(j)} - \mu_B)^2$$

$$\hat{x}^{(j)} = \frac{x^{(j)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \quad \forall j = 1, 2, \cdots, B$$

$$y^{(j)} = \gamma\hat{x}^{(j)} + \beta, \quad \forall j = 1, 2, \cdots, B$$

There is another type of normalization called "layer normalization." In "layer normalization," the input is $\mathbf{x} \in \mathbb{R}^D$, the activity of D artificial neurons, and the output is $\mathbf{y} \in \mathbb{R}^D$, the layer normalized activity of the D artificial neurons, according to the following equations:

LAYER NORM

$$\mu_\ell = \frac{1}{D}\sum_{i=1}^{D}x_i$$

$$\sigma_\ell^2 = \frac{1}{D}\sum_{i=1}^{D}(x_i - \mu_l)^2$$

$$\hat{x}_i = \frac{x_i - \mu_\ell}{\sqrt{\sigma_\ell^2 + \epsilon}}, \quad \forall i = 1, 2, \cdots, D$$

$$y_i = \gamma\hat{x}_i + \beta, \quad \forall i = 1, 2, \cdots, D$$

where x_i and y_i are the i^{th} elements of $\mathbf{x}$ and $\mathbf{y}$, respectively. Therefore, μ_ℓ is the mean of all the D artificial neurons and σ_ℓ^2 is the variance across the D artificial neurons in the l^{th} layer.

a) $x^{(i)} \in \mathbb{R}^d$, $\hat{x}^{(i)} \in \mathbb{R}^d$ SGD $\theta_{t+1} \leftarrow \theta + \varepsilon\nabla_\theta J_i(\theta)$

$$\frac{\partial\mathcal{L}}{\partial z} = -y\frac{\partial s}{\partial z} \qquad \frac{\partial\mathcal{L}}{\partial\hat{z}} = -y\frac{\partial s}{\partial\hat{z}}$$

COSINE SIMILARITY

$$\frac{\partial s}{\partial z} = \frac{(\|z\|_2\|\hat{z}\|_2)\hat{z} - (z^T\hat{z})\left(\frac{\|\hat{z}\|_2}{\|z\|_2}\right)z}{\|z\|_2^2\|\hat{z}\|_2^2}$$

$$\frac{\partial s}{\partial\hat{z}} = \frac{(\|z\|_2\|\hat{z}\|_2)z - (z^T\hat{z})\left(\frac{\|z\|_2}{\|\hat{z}\|_2}\right)\hat{z}}{\|z\|_2^2\|\hat{z}\|_2^2}$$

b) $\nabla_{w_2}\mathcal{L}(\delta_{w_2})$

2 paths to w_2: $\frac{\partial\mathcal{L}}{\partial w_2} = \frac{\partial z}{\partial w_2}\frac{\partial\mathcal{L}}{\partial z} + \frac{\partial\hat{z}}{\partial w_2}\frac{\partial\mathcal{L}}{\partial\hat{z}}$

with the tensor trick we have $\delta_{w_2} = \delta_z h^T + \delta_{\hat{z}}\hat{h}^T$

e) $\nabla_{w_1}\mathcal{L}$

2 paths $\nabla_{w_1}\mathcal{L} = \frac{\partial m}{\partial w_1}\frac{\partial\mathcal{L}}{\partial m} + \frac{\partial n}{\partial w_1}\frac{\partial\mathcal{L}}{\partial n}$

$\delta_{w_1} = \delta_m x^T + \delta_n\hat{x}^T$

c) $\nabla_{h_1}\mathcal{L}$, $\nabla_{\hat{h}_1}\mathcal{L}$

$$\nabla_{h_1}\mathcal{L} = \frac{\partial z}{\partial h_1}\frac{\partial\mathcal{L}}{\partial z}$$

$$\nabla_{\hat{h}_1}\mathcal{L} = \frac{\partial\hat{z}}{\partial\hat{h}_1}\frac{\partial\mathcal{L}}{\partial\hat{z}}$$

$$\frac{\partial z}{\partial h_1} = \frac{\partial z}{\partial\hat{h}_1} = W_2^T$$

$$\delta_{h_1} = W_2^T\delta_z$$

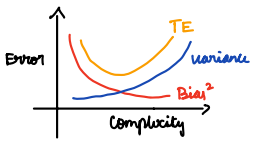
$$\delta_{\hat{h}_1} = W_2^T\delta_{\hat{z}}$$

d) $\nabla_m\mathcal{L}$, $\nabla_n\mathcal{L}$

Relu route for gradient

$$\nabla_m\mathcal{L} = \frac{\partial h_1}{\partial m}\frac{\partial\mathcal{L}}{\partial h_1} \quad \delta_m = 1(m\ge 0)\odot\delta_{h_1}$$

$$\nabla_n\mathcal{L} = \frac{\partial\hat{h}_1}{\partial n}\frac{\partial\mathcal{L}}{\partial\hat{h}_1} \quad \delta_n = 1(n\ge 0)\odot\delta_{\hat{h}_1}$$

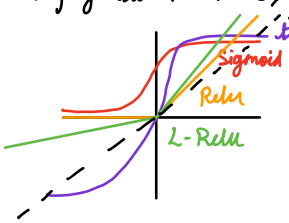


$\lambda\uparrow$ underfit training data -> $\uparrow$ bias - $\downarrow$ variance (less complex)
$\lambda\downarrow$ overfit training data -> $\downarrow$ bias - $\uparrow$ variance (more complex)

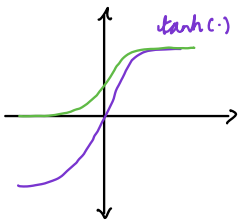
Dropout (an approx. to bagging - where all models share weights)

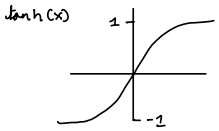
-> During each forward pass, some activations may be zeroed out
-> But will somehow get included in the training process
-> ∴ they have the same no. of parameters as the vanilla network
-> The weights dropped during the FP are not zeroed out but they are simply not updated in subsequent BP steps. ∴ Scaling by p does not reduce the size of the model

Vanishing gradients: Relu, Tanh, Sigmoid



bc Relu, Tanh, & Sigmoid saturate



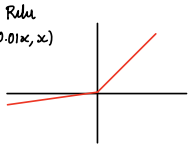


-> Saturates as x becomes very (+) or (-)
-> Is differentiable everywhere as there are no discontinuities
-> Approximates identity near origin

Gaussian Error Linear Unit (GELU)
$\text{Gelu}(x) = 0.5x\,(1+\tanh[\sqrt{2/\pi}\,(x+0.044715x^3)])$

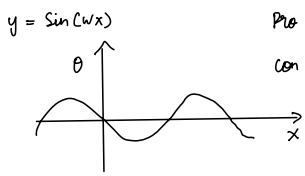
-> Saturates when x is very (-)
-> Is differentiable everywhere
-> Does not approximate identity near origin

Leaky Relu
$\max(0.01x, x)$



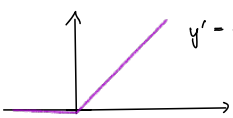
-> does not saturate
-> not diff. at $x=0$
-> does not approx. identity near origin

$\max(\alpha \cdot g, g), 0 < \alpha < 1$

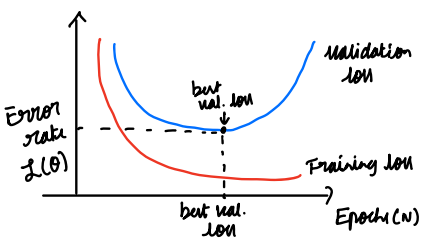


Pro: ① non-linear / differentiable everywhere

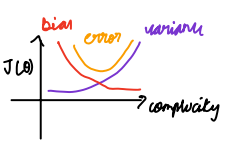
Con: Periodic $\sin(wx_1)$ may equal to non-unique outputs for different inputs. The oscillatory gradient can result in unstable gradients -> leading to unstable training



$y' = \begin{cases} 1 & i=j \\ 0 & \text{else} \end{cases}$



↑Train accuracy, bad testing error -> Overfitting -> Do not add layers!
↓bias, ↑variance, very complex -> Regularize
- augmentation
- dropout



Initializations

a. Constant values may not be a good idea for a fully connected NN. This can lead to symmetry breaking issues where neurons in the layers start to behave identically, limiting the network capacity to learn diverse features. Each unit gets signal equal to the sum of inputs. The gradients of all the neurons will be the same in the layer.

-> Weights initialized at 0 will result in 0 gradients, ∴ no gradients
-> Large initialization can lead to vanishing gradients (saturation ranges for tanh & sigmoid).
-> "0 initialization kills learning"
-> Init. with the same weight will not lead to the weights being equal
-> Symmetry can cause G_x to be equal for weights sharing the same input feature, but weights connected to different input features can break symmetry
-> Random init. -> exploding / vanishing gradients & worsen I.C.S.
-> For the right batch size, random init. does not affect BN strongly bc the μ & σ of the output of BN is constrained & learned

Name	Plot	Equation	Derivative (with respect to x)
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$ [1]	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
ElliotSig[9][10][11] Softsign[12][13]		$f(x) = \frac{x}{1+\lvert x\rvert}$	$f'(x) = \frac{1}{(1+\lvert x\rvert)^2}$
Inverse square root unit (ISRU)[14]		$f(x) = \frac{x}{\sqrt{1+\alpha x^2}}$	$f'(x) = \left(\frac{1}{\sqrt{1+\alpha x^2}}\right)^3$
Inverse square root linear unit (ISRLU)[14]		$f(x) = \begin{cases} \frac{x}{\sqrt{1+\alpha x^2}} & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \left(\frac{1}{\sqrt{1+\alpha x^2}}\right)^3 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Square Nonlinearity (SQNL)[11]		$f(x) = \begin{cases} 1 & : x > 2.0 \\ x - \frac{x^2}{4} & : 0 \leq x \leq 2.0 \\ x + \frac{x^2}{4} & : -2.0 \leq x < 0 \\ -1 & : x < -2.0 \end{cases}$	$f'(x) = 1 \mp \frac{x}{2}$
Rectified linear unit (ReLU)[15]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Leaky rectified linear unit (Leaky ReLU)[16]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric rectified linear unit (PReLU)[17]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Randomized leaky rectified linear unit (RReLU)[18]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ [3]	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus[23]		$f(x) = \ln(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$
Bent identity		$f(x) = \frac{\sqrt{x^2+1}-1}{2} + x$	$f'(x) = \frac{x}{2\sqrt{x^2+1}} + 1$
SoftExponential[26]		$f(\alpha, x) = \begin{cases} -\frac{\ln(1-\alpha(x+\alpha))}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0 \\ \frac{e^{\alpha x}-1}{\alpha} + \alpha & \text{for } \alpha > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \frac{1}{1-\alpha(\alpha+x)} & \text{for } \alpha < 0 \\ e^{\alpha x} & \text{for } \alpha \geq 0 \end{cases}$
Soft Clipping[27]		$f(\alpha, x) = \frac{1}{\alpha}\log\frac{1+e^{\alpha x}}{1+e^{\alpha(x-1)}}$	$f'(\alpha, x) = \frac{1}{2}\sinh\left(\frac{p}{2}\right)\operatorname{sech}\left(\frac{px}{2}\right)\operatorname{sech}\left(\frac{p}{2}(1-x)\right)$
Sinusoid[28]		$f(x) = \sin(x)$	$f'(x) = \cos(x)$
Sinc		$f(x) = \begin{cases} 1 & \text{for } x = 0 \\ \frac{\sin(x)}{x} & \text{for } x \neq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x = 0 \\ \frac{\cos(x)}{x} - \frac{\sin(x)}{x^2} & \text{for } x \neq 0 \end{cases}$
Gaussian		$f(x) = e^{-x^2}$	$f'(x) = -2xe^{-x^2}$

GD $\theta_{t+1} \leftarrow \theta_t + \varepsilon \nabla_\theta J(\theta)$

SGD $\theta_{t+1} \leftarrow \theta_t + \varepsilon \nabla_\theta J_i(\theta)$

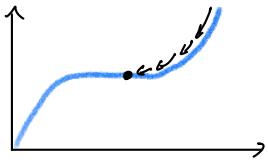
SGD → faster than GD when dataset is large
→ sensitive to noise in the data
→ more efficient in terms of memory usage

GD + momentum update: $V_{t+1} = \alpha V_t - \varepsilon \nabla_\theta \mathcal{L}(\theta_t)$
→ does not adjust step size per parameter
→ overshoot in ↑ curvature regions

GD + Nesterov update: $V_{t+1} = \alpha V_t - \varepsilon \nabla_\theta \mathcal{L}(\theta_t + \alpha V_t)$
Weight update: $\theta_{t+1} = \theta_t + V_{t+1}$

Momentum - avg. the zigzagging components
$V \leftarrow \alpha V - \varepsilon g$ V is the running avg. of historical gradients
$\theta \leftarrow \theta + V$
- helpful in larger networks
- SGD + momentum → helps find shallow local optimum
→ sensitive to noise

Nesterov momentum will help with faster convergence after a step where the accumulated momentum will be large. The weight update is always opposite to the direction of the gradient, ∴ in case it will be a large (+) update on weights overshooting in the case of GD + momentum



Nesterov momentum (anticipates upcoming gradients)
→ Gradient is calculated at parameter sitting after taking a step direction along the direction of momentum

$V \leftarrow \underbrace{\alpha V}_{\text{history}} - \varepsilon \nabla_\theta J(\theta + \alpha V)$
$\theta \leftarrow \theta + V$
$V_{new} = \alpha V_{old} - \varepsilon \nabla_{old} J(\tilde{\theta}_{old})$

$V = \alpha V - \varepsilon \nabla_\theta \mathcal{L}(\theta + \alpha V)$
$\tilde{\theta}_{new} = \theta_{new} + \alpha V_{new}$
$= \theta_{old} + V_{new} + \alpha V_{new}$
$= \tilde{\theta}_{old} - \alpha V_{old} + V_{new} + \alpha V_{new}$
$= \tilde{\theta}_{old} + V_{new} + \alpha(V_{new} + V_{old})$

$\tilde{\theta}_{new} = \tilde{\theta}_{old} + V_{new} + \alpha(V_{new} - V_{old})$
Don't have to evaluate ∇_θ at $\theta + \alpha V$

Adaptive LR

Adagrad (accumulate squared gradients)
$a \leftarrow a + g \odot g$ → 2nd moment taking variance into account
$\theta \leftarrow \theta - \frac{\varepsilon}{\sqrt{a} + V} \odot g$
historical gradient norms
→ smaller steps to approx. anneali → reduces effective LR when ∇_θ's are large
→ a → accum. sum of part ∇_θ squared
→ division by a larger number → small step
→ global minimum reached!
- NO momentum calculated, only update bc. on part ∇_θ.

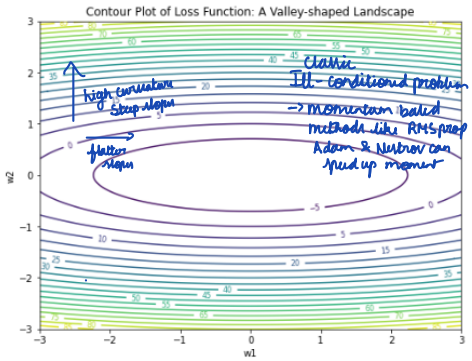
Adagrad / RMS-prop + momentum = Adam

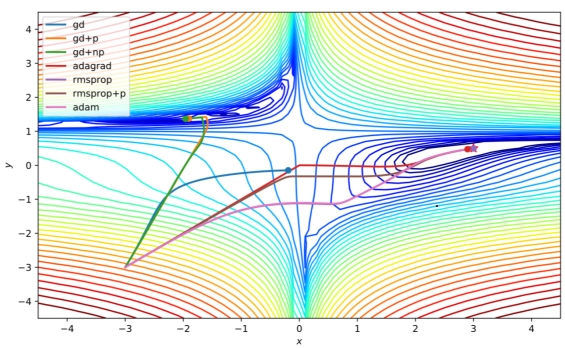
1st moment first $V \leftarrow \beta_1 V + (1-\beta_1) g$ (momentum like)
$a \leftarrow \beta_2 a + (1-\beta_2) g \odot g$ (∇_θ normalization)
$\theta \leftarrow \theta - \frac{\varepsilon}{\sqrt{a} + V} \odot V$

RMS Prop $a \leftarrow \beta a + (1-\beta) g \odot g$
$\theta \leftarrow \theta - \frac{\varepsilon}{\sqrt{a} + V} \odot g$
→ makes a an exponentially weighted moving avg.
→ does not get slower towards the end like Adagrad bc of the β parameter
→ can take bigger steps
→ leads to global minimum.

RMS Prop + momentum
→ Although it makes a large oscillation, gets to optim quickly
$a \leftarrow \beta a (1-\beta) g \odot g$
$V \leftarrow \alpha V - \frac{\varepsilon}{\sqrt{a} + V} \odot g$
$\theta \leftarrow \theta + V$

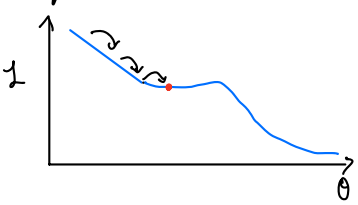
$g_{t+1} = \beta g_t + (1-\beta)(\nabla_\theta \mathcal{L}(\theta_t))^2$
$V_{t+1} = \alpha V_t - \frac{\varepsilon}{\sqrt{g_{t+1} + \delta}} \nabla_\theta \mathcal{L}(\theta_t)$
$\theta_{t+1} = \theta_t + V_{t+1}$





Adam with bias correction
$V \leftarrow \beta_1 V + (1-\beta_1) g$
$a \leftarrow \beta_2 a + (1-\beta_2) g \odot g$
$\tilde{V} = \frac{1}{1-\beta_1^t} V$; $\tilde{a} = \frac{1}{1-\beta_2^t} a$;
$\theta \leftarrow \theta - \frac{\varepsilon}{\sqrt{\tilde{a}} + \nu} \odot \tilde{V}$
at t gets large, $\beta_1^t \to 0$, $\beta_2^t \to 0$, correction disappears & we have $\tilde{V} = V$, $\tilde{a} = a$

$g_t : \nabla_\theta \mathcal{L}(\theta_{t-1})$
i) small step issue
GD + momentum / Adam will be good option to get out of the local minimum bc they would be able to gather momentum starting up-hill on the left which will eventually drive the optimizer out of the plateau where the ∇_θ can vanish.



Second order methods

→ use second derivative (or Hessian) to assess curvature
→ can capture curvature of the loss surface
→ But computationally expensive

Newton's method
$\theta \leftarrow \theta - H^{-1} g$ → memory storing is expensive
$\theta \leftarrow \theta - (H + \alpha I)^{-1} \nabla_\theta J(\theta_0)$ → Inverting H → $O(n^3)$
→ H requires very large batch size.

Update magnitude order

GD + momentum > Adam > Adagrad
momentum will help increase the update magnitude at the plateau while Adagrad will shrink it. Combining both, Adam will give an update magnitude in the middle of the 2 extremes

Vanishing Gradients

Update magnitude order
Newton's method > Adam + bias correction > Adam > Nesterov momentum > GD + momentum > RMSprop + momentum > SGD > GD > RMSprop > Adagrad

Layer Dimensions:

$$\frac{w - w_f + 2 \cdot pad}{stride} + 1 \qquad \frac{h - h_f + 2 \cdot pad}{stride} + 1$$

Total params $(3 \times 3 \times 3 + 1) \times 5 = 140$

n_f ↑ bias

MAXPOOL

→ max pool layer does not have any params.

→ to reduce invariance due to shifts

→ concept of retinotopic maps

→ Filter has its own bias term (+1 per filter)

# of neurons → output height × output width × n_f

# of params → (filter size +1) × n_f

Nesterov Momentum + GD

$$v_t = \alpha v_{t-1} - \epsilon \nabla f(\theta_{t-1} + \alpha v_{t-1})$$

$$\theta_t = \theta_{t-1} + v_t$$

$v_0 = 0$

$$v_1 = -\epsilon \nabla f(\theta_0)$$

$$v_2 = \alpha v_1 - \epsilon \nabla g_1$$

$$v_3 = \alpha v_2 - \epsilon \nabla f g_2$$

$$v_t = -\epsilon \sum_{i=1}^{t} \alpha^{t-i}$$

$$\theta_t = \theta_{t-1} + v_t \; ; \; \theta_t = \theta_0 + \sum_{j=1}^{t} v_j$$

$$\theta_t = \theta_0 - \epsilon \sum_{j=1}^{t} \sum_{i=1}^{j} \alpha^{j-1} \nabla f(\theta_{i-1} + \alpha v_{i-1})$$

OR

$$\theta_t = \theta_0 - \epsilon\alpha \sum_{i=1}^{t} \alpha^{t-i} g_i - \epsilon \sum_{j=1}^{t} \left(\sum_{i=1}^{j} \alpha^{j-1} g_i \right)$$

RMSprop + Momentum

The update equations are:

$$g_t = \beta g_{t-1} + (1-\beta)(\nabla_\theta L(\theta_{t-1}))^2$$

$$v_t = \alpha v_{t-1} - \frac{\epsilon}{\sqrt{g_t} + \delta} \nabla_\theta L(\theta_t)$$

$$\theta_t = \theta_{t-1} + v_t$$

Recursion Expansion

Expanding v_t recursively, assuming $v_0 = 0$:

$$v_t = -\sum_{i=1}^{t} \alpha^{t-i} \frac{\epsilon}{\sqrt{g_i} + \delta} g_i$$

Now, substituting into the weight update equation:

$$\theta_t = \theta_0 + \sum_{j=1}^{t} v_j$$

$$\theta_t = \theta_0 - \sum_{j=1}^{t} \sum_{i=1}^{j} \alpha^{j-i} \frac{\epsilon}{\sqrt{g_i} + \delta} g_i$$

Adagrad + Momentum

The update equations are:

$$g_t = g_{t-1} + (\nabla_\theta L(\theta_{t-1}))^2$$

$$v_t = \alpha v_{t-1} - \frac{\epsilon}{\sqrt{g_t} + \delta} \nabla_\theta L(\theta_t)$$

$$\theta_t = \theta_{t-1} + v_t$$

Recursion Expansion

Expanding v_t recursively:

$$v_t = -\sum_{i=1}^{t} \alpha^{t-i} \frac{\epsilon}{\sqrt{g_i} + \delta} g_i$$

Now, substituting into the weight update equation:

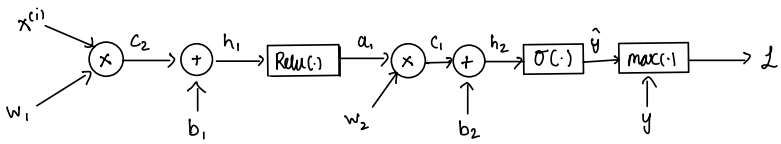
$$\theta_t = \theta_0 + \sum_{j=1}^{t} v_j$$

$$\theta_t = \theta_0 - \sum_{j=1}^{t} \sum_{i=1}^{j} \alpha^{j-i} \frac{\epsilon}{\sqrt{g_i} + \delta} g_i$$

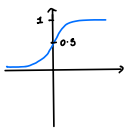
Consider the neural network with the following architecture

$$h_1 = W_1 x^{(i)} + b_1$$
$$a_1 = ReLU(h_1)$$
$$h_2 = W_2 a_1 + b_2$$
$$\hat{y} = \sigma(h_2)$$
$$L = \sum_{i=1}^{K} \max(0, 1 - y_i \hat{y}_i)$$

where $x^{(i)} \in \mathbb{R}^{D_x \times 1}, W_1 \in \mathbb{R}^{D_{a_1} \times D_x}, W_2 \in \mathbb{R}^{K \times D_{a_1}}$. σ is the sigmoid activation function and y is a K-dimensional vector of 1's and 0's. y_i represents the i^{th} element of y and $\hat{y}_i$ represents the i^{th} element of $\hat{y}$.



$$L = \sum_{i=1}^{K} \max(0, 1 - y_i \hat{y}_i)$$



$$\frac{\partial L}{\partial \hat{y}_i} = \frac{\partial}{\partial \hat{y}_i} [\max(0, 1 - y_i \hat{y}_i)] = \begin{cases} -y_i, & 1 - y_i \hat{y}_i > 0 \\ 0, & 1 - y_i \hat{y}_i < 0 \end{cases}$$

$\hat{y}$ is the output of a sigmoid operator, so $\hat{y}_i < 1$.

from problem statement $y_i \leq 1$. Hence $1 - y_i \hat{y}_i > 0$

$$\frac{\partial L}{\partial \hat{y}_i} = -y_i \quad , \quad \frac{\partial L}{\partial y_i} = -\hat{y}$$

$\mathbb{R}^{K \times K}$

$$\frac{\partial L}{\partial h_2} = \frac{\partial \hat{y}_i}{\partial h_2} \frac{\partial L}{\partial \hat{y}_i}$$

$\sigma(\cdot)$ is defined elementwise

$$\frac{\partial \hat{y}_i}{\partial h_2} = \sigma(h_{2,i})[1 - \sigma(h_{2,i})] = diag[\sigma(h_{2,1})\{1 - \sigma(h_{2,1})\} \ldots \sigma(h_{2,K})\{1 - \sigma(h_{2,K})\}]$$

$$\frac{\partial L}{\partial h_2} = -\hat{y} \odot (1 - \hat{y}) \odot y = \delta_{h_2}$$

⊕ $\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial h_2} = \delta_{h_2}$ $\quad \frac{\partial L}{\partial C_1} = \frac{\partial L}{\partial h_2} = \delta_{h_2}$

by tensor derivative trick

⊗ $\frac{\partial L}{\partial W_2} = \frac{\partial C_1}{\partial W_1} \frac{\partial L}{\partial C_1} = \delta_{h_2} a_1^T$

$$\frac{\partial L}{\partial a_1} = \frac{\partial C_1}{\partial a_1} \frac{\partial L}{\partial C_1} = W_2^T \delta_{h_2}$$

$a_1 = Relu(h_1)$

$$\frac{\partial L}{\partial h_1} = \frac{\partial a_1}{\partial h_1} \frac{\partial L}{\partial a_1} = \mathbb{I}(h_1 > 0) \odot \delta_{a_1}$$

⊕ $\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial h_1} = \delta_{h_1}$

passes gradient

$$\frac{\partial L}{\partial C_2} = \frac{\partial L}{\partial h_1} = \delta_{h_1}$$

⊗ $C_2 = W_1 x^{(i)}$ Tensor derivative trick

$$\frac{\partial L}{\partial W_1} = \frac{\partial C_2}{\partial W_1} \frac{\partial L}{\partial C_2} = \delta_{h_1} x^{(i)T}$$

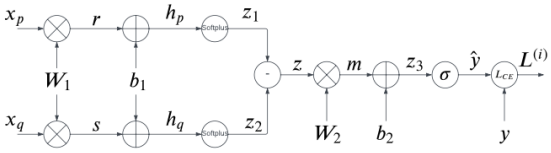
$\mathbb{R}^{D_{a_1} \times D_x \times D_{a_1}}$

2. a) $\nabla_{\hat{y}^{(i)}} \mathcal{L}^{(i)} \rightarrow \delta_{\hat{y}^{(i)}}$

$$\mathcal{L}^{(i)} = y^{(i)} \log(\hat{y}^{(i)}) + (1-y^{(i)}) \log(1-\hat{y}^{(i)})$$

$$\frac{\partial \mathcal{L}}{\partial \hat{y}^{(i)}} = \frac{y^{(i)}}{\hat{y}^{(i)}} - \left[\frac{1-y^{(i)}}{1-\hat{y}^{(i)}}\right]$$

$$h_p = W_1 x_p^{(i)} + b_1$$
$$z_1 = \text{softplus}(h_p)$$
$$h_q = W_1 x_q^{(i)} + b_1$$
$$z_2 = \text{softplus}(h_q)$$
$$z = z_1 - z_2$$
$$z_3 = W_2 z + b_2$$
$$\hat{y}^{(i)} = \sigma(z_3)$$
$$L^{(i)} = L_{CE}(y^{(i)}, \hat{y}^{(i)})$$
$$L = -\frac{1}{m}\sum_{i=1}^{m} L^{(i)}$$



b) $\nabla_{z_3} \mathcal{L}^{(i)} \rightarrow \delta_{z_3}$

$$\frac{\partial \mathcal{L}}{\partial z_3} = \frac{\partial \hat{y}}{\partial z_3} \frac{\partial \mathcal{L}}{\partial \hat{y}} = \sigma(z_3)[1-\sigma(z_3)] \cdot \delta_{\hat{y}^{(i)}} = \delta_{z_3}$$

c) $\nabla_{W_2} \mathcal{L}^{(i)} \rightarrow \delta_{W_2}$

$m = W_2 z$

$$\frac{\partial \mathcal{L}}{\partial W_2} = \frac{\partial z_3}{\partial W_2} \frac{\partial \mathcal{L}}{\partial z_3} = z^T \cdot \delta_{z_3}$$

d) $\nabla_z \mathcal{L}^{(i)} \rightarrow \delta_z$

$m = W_2 z$

$$\frac{\partial \mathcal{L}}{\partial z} = \delta_{z_3} \cdot W_2^T$$

e) $\nabla_{h_q} \mathcal{L}^{(i)} \rightarrow \delta_{h_q}$ → $\odot$ operator

$$\frac{\partial \mathcal{L}}{\partial h_q} = \frac{\partial z_2}{\partial h_q} \frac{\partial \mathcal{L}}{\partial z_2} = \left[\frac{e^{\beta h_q}}{1+e^{\beta h_q}}\right] \cdot -\delta_z$$

f) $\nabla_{h_p} \mathcal{L}^{(i)} \rightarrow \delta_{h_p}$

$$\frac{\partial \mathcal{L}}{\partial h_p} = \frac{\partial z_1}{\partial h_p} \frac{\partial \mathcal{L}}{\partial z_1} = \left[\frac{e^{\beta h_p}}{1+e^{\beta h_p}}\right] \cdot \delta_z$$

g) $\nabla_b \mathcal{L}^{(i)}$

$$\nabla_b \mathcal{L}^{(i)} = \delta_{h_p} + \delta_{h_q}$$

h) $\nabla_{W_1} \mathcal{L}^{(i)}$

$$\nabla_{W_1} \mathcal{L}^{(i)} = \delta_{h_p} x_p^{(i)T} + \delta_{h_p} x_q^{(i)T}$$

i) $\tilde{\mathcal{L}} = \left(-\frac{1}{m}\sum_{i=1}^{m} \mathcal{L}^{(i)}\right) + \ln\left(\sum_{i=1}^{n} e^{x_i}\right) + \|W_1\|_\infty$

Since gradient is a linear operator

$$\nabla_{W_1} \tilde{\mathcal{L}} = \frac{\partial \tilde{\mathcal{L}}}{\partial W_1} = -\frac{1}{m}\sum_{i=1}^{m} \nabla_{W_1} \mathcal{L}^{(i)} + \nabla_{W_1} \|W_1\|_\infty$$

$$\nabla_{W_2} \tilde{\mathcal{L}} = \frac{\partial \tilde{\mathcal{L}}}{\partial W_2} = -\frac{1}{m}\sum_{i=1}^{m} \nabla_{W_2} L^{(i)} + \nabla_{W_2} \ln\left(\sum_{i=1}^{n} e^{x_i}\right)$$

$$\nabla_{W_1} \|W_1\|_\infty = \begin{bmatrix} 0 \\ 0 \\ \text{sign}(w_j) \\ 0 \end{bmatrix} \qquad \nabla_{W_2} \text{LSE} = \left[\frac{e^{x_1}}{\sum_{i=1}^{n} e^{x_i}} \;\cdots\right]^T = \text{softmax}$$

$$\therefore \quad \nabla_{W_1} \tilde{\mathcal{L}} = -\frac{1}{m}\sum_{i=1}^{m} \nabla_{W_1} \mathcal{L}^{(i)} + \begin{bmatrix} 0 \\ 0 \\ \text{sign}(w_j) \\ 0 \end{bmatrix}$$

$$\nabla_{W_2} \tilde{\mathcal{L}} = -\frac{1}{m}\sum_{i=1}^{m} \nabla_{W_2} \mathcal{L}^{(i)} + \left[\frac{e^{x_1}}{\sum_{i=1}^{n} e^{x_i}} \;\cdots\right]^T$$