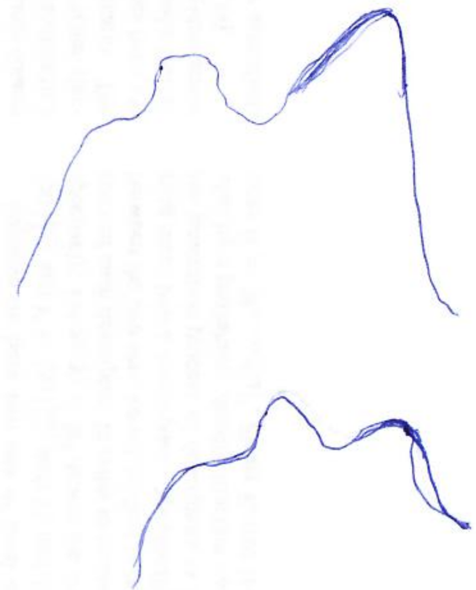
- Problem to Solve
  1. Internal Covariate Shift
  2. pathological curvature of first-order gradient descent
- Previous Approach
  → weighting / domain adaptation for covariate shift
  → Whitening activation

* Batch normalization
* Advantages / Acceleration / criticism of BN
* Layer normalization
* Weight normalization
* comparisons regarding < invariance / purposes

→ makes sense to apply normalization + saturating nonlinearity

→ data-dependent initialization

→ layer norm ~~on~~ conv is against convolution property



⊛ internal covariate shift
$\overset{\shortparallel}{\text{dependent variable}}$

loss $l = F_2(\underbrace{F_1(u, \theta_1)}_{= x}, \theta_2)$

$= F_2(x, \theta_2)$: sub network with sub input

- if F_2 contains ~~eg. input~~
g = saturating (regime of) the nonlinearity, eg. $g(v) = \frac{1}{1+\exp(-v)}$

if |input of g| ↑ ; $g'|_v \to 0$
$\overset{\shortparallel}{v}$

thus, F_1 trains slowly
and $F_1(u, \theta_1)$ moves to saturated regimes.

⇛ the goal of batch normalization
: make the distribution of x (input of saturating nonlinearity) fixed of time & batch

⊛⊛ naive whitening activation

~~normalize~~
if normalization parameters are outside gradient descent,

eg. $\hat{x} = x - E[x]$ where $x = Wu + b$ (input: u)
$b \to b + \Delta b$ do not affect $\hat{x}$
$\because x - E[x] = Wu + b - E[Wu + b]$
$= Wu + b + \Delta b - E[Wu + b + \Delta b]$
b will explode without reducing loss function

→ To solve ⊛ and ⊛⊛
- fixed distribution over time.
- differentiable
- preserve normalization param to ~~ga~~ network

result (advantages)
- use of saturating nonlinearity
- increase of learning rate
- model regularization due to sampling
- ~~More~~ resilient parameter scaling to initialization
- conjecture : condition # ~ 1

⊛ BN (④ more tricks in 4.2.1)

① mini-batch statistics to estimate mean and variance (decorrelated features)

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \sigma_B^2 = \sum_{i=1}^{m}(x_i - \mu_B)^2$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \qquad y_i = \gamma \hat{x}_i + \beta$$

Parameter to be learned

from each batch $B = \{x_1, \dots, x_m\}$

② training

- take y_i in ① as inputs
- train to optimize $\{\gamma^{(k)}, \beta^{(k)}\}_k$ together with other parameters

eg? in convolution per feature map

③ inference using unbiased estimators

$$E[x] = E_B[\mu_B] \qquad Var[x] = \frac{m}{m-1} E_B[\sigma_B^2]$$

$$y = \frac{\gamma}{\sqrt{Var[x]+\varepsilon}} \cdot x + \left(\beta - \frac{\gamma E[x]}{\sqrt{Var[x]+\varepsilon}}\right)$$

↳ input to the network.

⊛ for convolution

→ immediately before nonlinearity

→ jointly normalize all activations in a minibatch over all locations of convolution layer with different $\gamma^{(k)}, \beta^{(k)}$ pairs per feature map

∵ for convolution property!

⊛ criticism of BN in LN perspectives

⊕ ~~batc~~ mini-batch statistics are only estimates

② mini-batch size is constrained

③ different parameters for each activation

→ variable length of RNN ?

④ dependency within mini-batch

⊛ layer normalization

→ normalization statistics over [all the hidden units] in [the same layer] [per sample]

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^l - \mu^l)^2}$$

→ in CNN : batch norm outperforms

⇛ RNN & Online — learning rate regularization; robust (resilient) to input parameter scaling

⊛ gained parameters $\xleftarrow{\text{"scaling"}}$ ||incoming weight||

⊛ RNN

$$a^t = W_{hh} h^{t-1} + W_{xh} x^t$$

$$h^t = f\left[\frac{\vec{g}}{\sigma^t} \odot (a^t - \mu^t) + b\right]$$

↑ element-wise multiplication

$$\mu^t = \frac{1}{H}\sum_{i=1}^{H} a_i^t$$

$$\sigma^t = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^t - \mu^t)^2}$$

~~⊛ Weight normalization~~

⊛ pathological curvature
of the objective at optimum.

(∵ the condition # of the Hessian matrix
at optimum is low; ⌈unstable gradient descent)

→ curvature ~ parameterization

⊛ whitening (the cost) gradient (natural gradient)

- left multiply $(\text{Fisher info matrix})^{-1}$

~~(whitening input~~

→ approximation & overhead

⊛ weight normalization

1. $\vec{w} = \frac{g}{\|\vec{v}\|}\vec{v}$ weight vector

$y = \phi(\vec{w}\cdot\vec{x} + b)$ $\quad \vec{w} \to \frac{\vec{v}}{\|\vec{v}\|}, g$.

$(g \to e^{s})$

$$\therefore \begin{cases} \nabla_g L = \dfrac{\nabla_{\vec{w}} L \cdot \vec{v}}{\|\vec{v}\|} \\ \nabla_{\vec{v}} L = \dfrac{g}{\|\vec{v}\|}\nabla_{\vec{w}} L - g\dfrac{\nabla_g L}{\|\vec{v}\|^2}\vec{v} \\ \quad = \underbrace{\dfrac{g}{\|\vec{v}\|}}_{\text{scale}} \underbrace{M_{\vec{w}}}_{\text{projection}} \nabla_{\vec{w}} L \quad \text{where } M_w = I - \dfrac{\vec{w}\vec{w}'}{\|\vec{w}\|^2} \end{cases}$$

; compute orthogonal increament to the current $\vec{w}$

→ self stabilizing is not compatible with Adam, momentum optimizers.

; $\mathrm{Cov}(\nabla_{\vec{v}} L) = \left(\frac{g^2}{\|v\|^2}\right) M_{\vec{w}}\, \mathrm{Cov}(\nabla_{\vec{w}} L) M_{\vec{w}} \sim 1$

; stabilizing noise.

⊛ Weight normalization

2. Data-dependent Initialization
(∵ missing scaling of features)

$y = \phi\left(\frac{g}{\|\vec{v}\|} \vec{v} \cdot \vec{x} + b\right)$ then

initialize $g \leftarrow \frac{1}{\sigma[t]}$ $b \leftarrow \frac{-\mu[t]}{\sigma[t]}$

where $\sigma[t]$, $\mu[t]$: batch-statistics.

→ not applicable for RNN

3. Mean-only Batch normalization.

$\tilde{t} = t - \mu[t] + b$ where $t = \vec{w} \cdot \vec{x}$

$y = \phi(\tilde{t})$

$\mu[t]$: running avg of mini batch
↓
test time

⊛ advatages

faster, robust to noise

~~noise to~~

⊛ invariance analysis

Weight matrix

BN, WN, LN – invariant over scaling

LN – invariant over centering.

Weight vector (feature)

BN, WN – invariant over scaling

Dataset ~~rescal~~

BN, LN – invariant over scaling

BN – invariant over centering

Single training

LN – invariant over scaling.

⊛ Riemannian metric (curvature) under KL

$$ds^2 = D_{KL}[P(y|\vec{x};\theta) \,\|\, P(y|\vec{x};\theta+\delta)]$$

$$\approx \frac{1}{2}\delta^T F(\theta)\delta$$

where $F(\theta) = \underset{\substack{\vec{x}\sim P(\vec{x}) \\ y\sim P(y|\vec{x})}}{E}\left[\frac{\partial \log P(y|x;\theta)}{\partial\theta}\frac{\partial \log P(y|\vec{x};\theta)}{\partial\theta}\right]$

in GLM

$$\begin{cases} \log P(y|x;w,b) = \dfrac{(a+b)y - \eta(a+b)}{\phi} + c(y,\phi) \quad \text{log partition} \\ E[y|x] = f(a+b) \quad Var[y|x] = \phi f'(a+b) \\ a = wx \end{cases}$$

$$\Rightarrow F(\theta) = \underset{\vec{x}\sim P(\vec{x})}{E}\left[\frac{Cov[y|\vec{x}]}{\phi^2} \otimes \begin{bmatrix} \vec{x}\vec{x}^T & \vec{x}^T \\ \vec{x}^T & 1 \end{bmatrix}\right]$$

in normalized GLM g: parameter scales

$$[\bar{F}_{ij}] = \left[\underset{x\sim P(x)}{E}\left[\frac{Cov[y_i,y_j|\vec{x}]}{\phi^2}\right]\begin{bmatrix} \frac{g_i g_j}{\sigma_i\sigma_j}\chi_i\chi_j^T & \chi_i\frac{g_i}{\sigma_i} & \chi_i\frac{g_i(a_j-\mu_j)}{\sigma_i\sigma_j} \\ \chi_j^T\frac{g_j}{\sigma_j} & 1 & \frac{a_j-\mu_j}{\sigma_j} \\ \chi_j^T\frac{g_j(a_i-\mu_i)}{\sigma_i\sigma_j} & \frac{a_i-\mu_i}{\sigma_i} & \frac{(a_i-\mu_i)(a_j-\mu_j)}{\sigma_i\sigma_j} \end{bmatrix}\right]$$

$$\chi_i = \vec{x} - \frac{\partial\mu_i}{\partial w_i} - \frac{a_i-\mu_i}{\sigma_i}\frac{\partial\sigma_i}{\partial w_i}$$

⇒ as $w\uparrow$: output fixed : hard to change w : lr ↓

↳ robust of input parameter scale due to $\frac{g_i}{\sigma_i}$ -scale