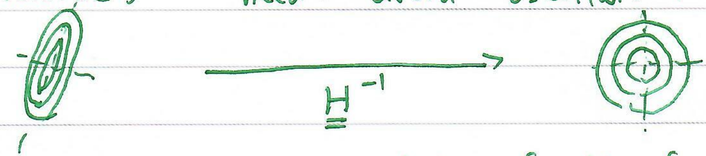
Subject : ....

No. :

Date : ..../........./..........

* difficulty in DL Optimization

- local min : not as bad as imagined
- saddle / local max : easily escapable (esp. w/ SGD)
- plateau : need larger η (learning rate) :
- ravines : need avoid oscillation



$\underline{\underline{H}}^{-1}$

(Quasi-)Newton, but infeasible for DL usually

- slow computation of gradient : SGD on minibatch

$\Downarrow$

"instable" estimate of gradient

* $\boxed{\boxed{\text{running}}\text{ average estimate}}$ of gradient from $\boxed{\text{SG}}$

$$\boxed{\underline{v}_t} = \beta \cdot \boxed{\underline{v}_{t-1}} + (1-\beta)\boxed{\underline{\Delta}_t}$$

current average

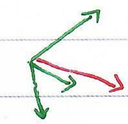
previous average

stochastic gradient at t-th iteration

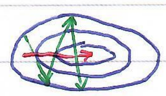
update : $\underline{w}_t = \underline{w}_{t-1} - \eta\, \underline{v}_t$

(SGD with) momentum

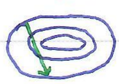
- stochastic instability cancels out



- pass ravine faster



* $\boxed{\text{oscillation}}$ prevention by per-component learning rate



larger gradient component

want: smaller step

update : $\underline{w}_t = \underline{w}_{t-1} - \eta\ \underline{\Delta}_t \ \cdot/$ "magnitude($\underline{\Delta}_t$)"

$$\text{magnitude}(\underline{\Delta}_t) \equiv \sqrt{\underline{u}_t + \epsilon}$$

running average

$$\underline{u}_t = \beta\, \underline{u}_{t-1} + (1-\beta)\,(\underline{\Delta}_t \ .\!*\ \underline{\Delta}_t)$$

RMSprop

Subject : .... No. : Date : ..../........../..........

* Adam = SGD + momentum + RMSProp (+ other tricks)

$$\underline{V}_t = \underset{\downarrow \atop 0.9}{\beta_1}\, \underline{V}_{t-1} + (1-\beta_1)\, \underline{\Delta}_t$$

$$\underline{U}_t = \underset{\downarrow \atop 0.999}{\beta_2}\, \underline{U}_{t-1} + (1-\beta_2)\, (\underline{\Delta}_t \,.\!*\, \underline{\Delta}_t)$$

$$0.001 \leftarrow \frac{\eta}{\sqrt{t/N}} \quad \text{(decaying)}$$

$$\underline{W}_t = \underline{W}_{t-1} - \frac{\eta_t}{\sqrt{\underline{U}_t + \underset{\downarrow \atop 10^{-8}}{\epsilon}}} \,.\!*\, \underline{V}_t$$

* difficulty in DL generalization

- co-adaptation :

(consistent mistakes from some neurons) corrected by fitting other neurons

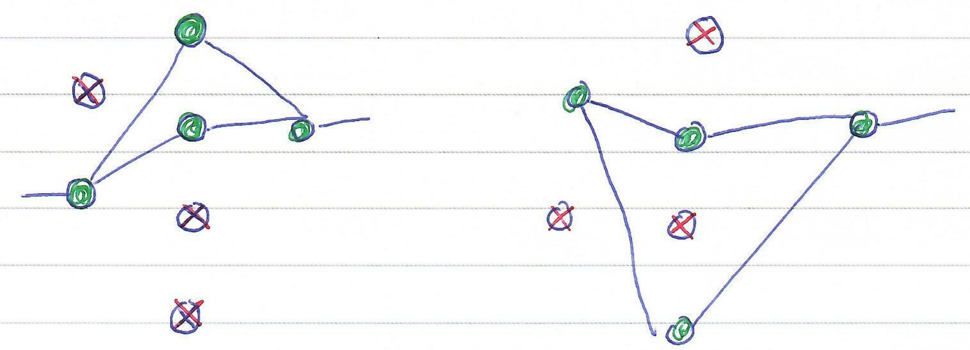
$\Downarrow$ like noise

$\Downarrow$ like overfit

and vice versa

(dependence)

* break the dependence : shut down neurons (randomly)



mini-batch-1 mini-batch-2

dropout : drop p, keep $(1-p)$ $\Rightarrow$ many thin networks (combined)

$\Downarrow$ slows down convergence but faster per-iteration

$\Rightarrow$ noise as regularization like DAE

Subject : ....

No. :
Date : ..../........../..........

* dropout during testing
  - full-net prediction w/o changing

$$\underset{\text{in training}}{E(s_i^{(l)})} = \underset{\text{in testing}}{(1-p)\, s_i^{(l)}}$$

  - test-time "pseudo-" dropout

$$x_i^{(l)} = \theta\big(\underbrace{(1-p)}\, s_i^{(l)}\big)$$

need to record p, less flexibility for changing p per neuron or dynamically

  - inverted dropout

training : dropout & $x_i^{l} = \theta\left(s_i^{(l)} / (1-p)\right)$

testing : $x_i^{(l)} = \theta\left(\cancel{(1-p)}\, s_i^{(l)} / \cancel{(1-p)}\right)$

$= \theta\left(s_i^{(l)}\right)$ unchanged (usually preferred)