Improving convergence of stochastic gradient descent

Neural network – large general parametrization

How to train them: choose these parameters to represent given data?

By some gradient descent … how to choose step size, pass saddles?

$\theta^{t+1} = \theta^t - \eta g^t$ e.g. for $g^t = \nabla_\theta F(\theta^t)$ $F(\theta) = \frac{1}{n}\sum_{i=1}^n L(x_i, \theta)$

Gradient for entire (huge n) dataset, or online updates: mini-batches

Small batch – cheap, but large noise, how to extract statistical trends?

Linear or model parabola - modelling distance to extremum? (step size)

In one or multiple directions? especially near saddles

exp(dim) saddles >> # minima at least from parameter permutations

Jarek Duda

There is some idealized

hidden probability distribution $p(x)$

e.g. of [pictures of digits] among bitmaps of given resolution, we want to label them

We need **parameters** $\theta \in \mathbb{R}^D$

minimizing loss/objective function:

$$\tilde{F}(\theta) = \int L(x,\theta)p(x)dx$$

But we know only samples $\{x_i\}_{i=1..n}$

Instead find local minimum of:

$$F(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(x_i,\theta)$$

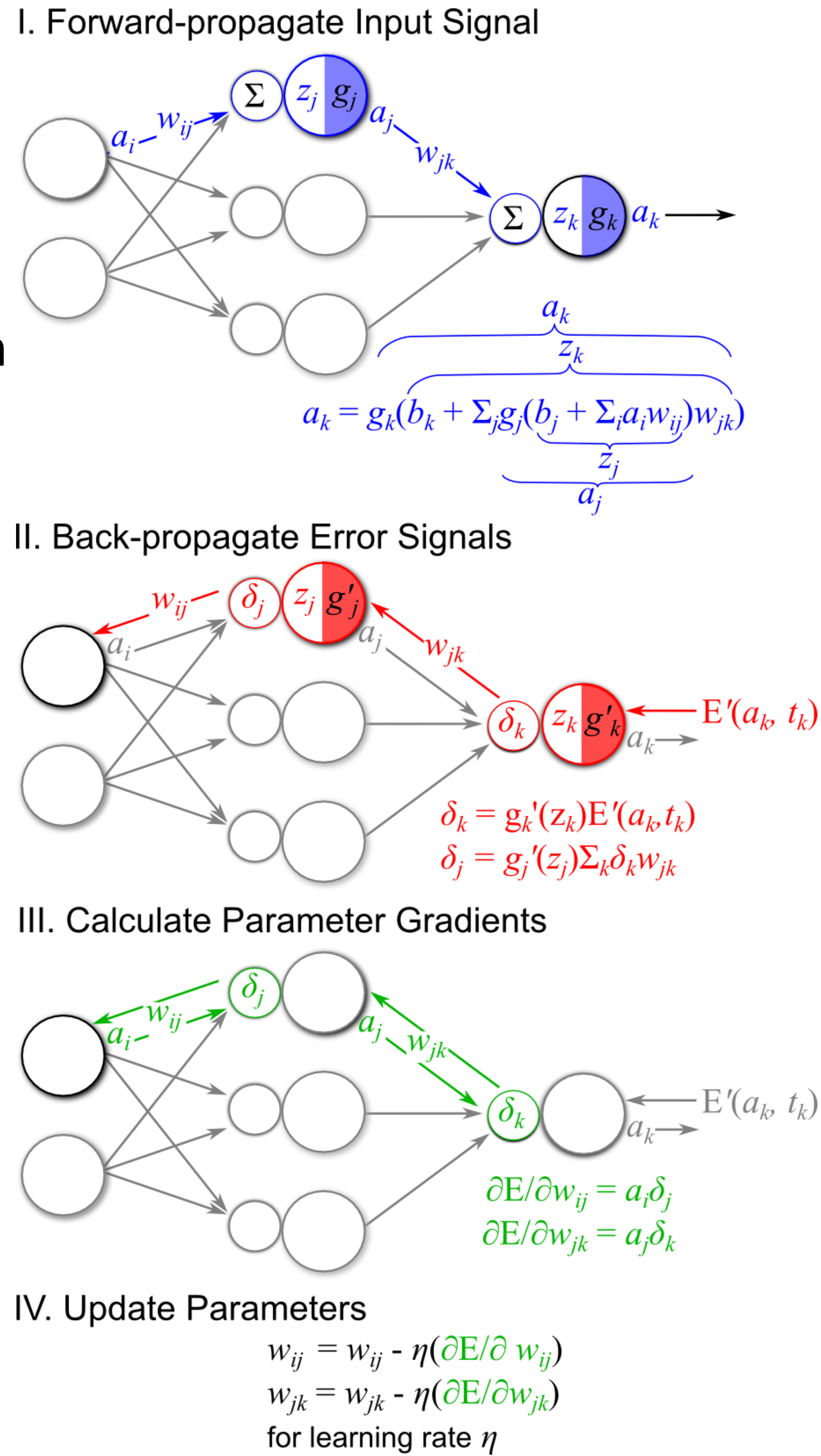
using **gradient sequence** $\{\nabla_{\boldsymbol{\theta}} \boldsymbol{F}(\boldsymbol{\theta}^t)\}_t$

e.g. from **[backpropagation]** of errors.

Generalization: to prevent **overfitting**,

train on one subset,

test on separate validation set.



We would like to minimize $F(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(x_i, \theta)$ based on gradients:

1) **Batch/vanilla gradient descent**: using $g^t = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta L(x_i, \theta^t)$
2) **Stochastic gradient descent (SGD)**: "online" $g^t = \nabla_\theta L(x_{i^t}, \theta^t)$
3) **Mini-batch** (~100) **gradient descent**: using gradient from subsets

Averaging over entire dataset (batch) – accurate but extremely costly, over a subset – cheaper but noisy – average over time to real gradient

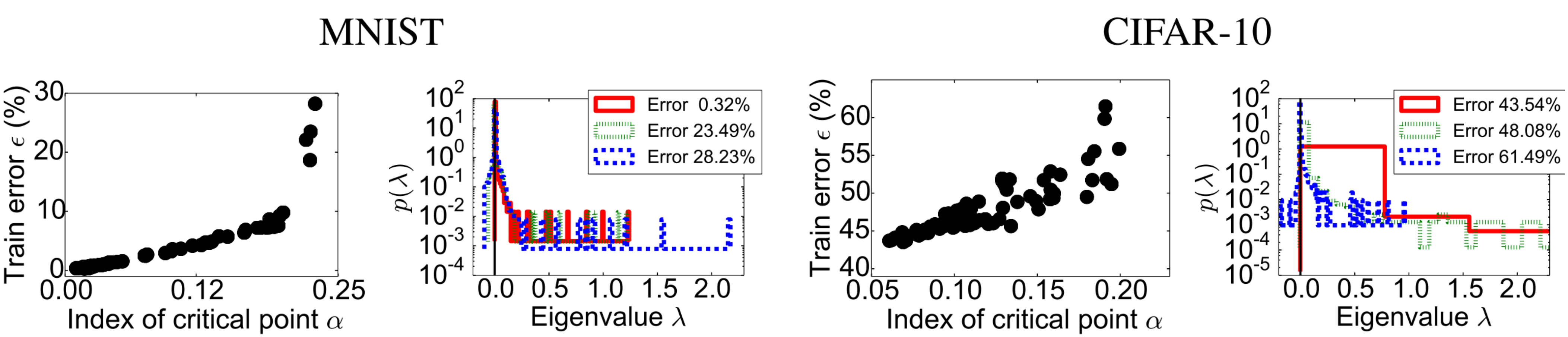
Let's combine 2) and 3): gradients from size 1+ subsets, **averaging** to ~real

Symmetry of parameters – lower bound for number of local minima

It seems most of **local minima** have close value to **global minimum** here

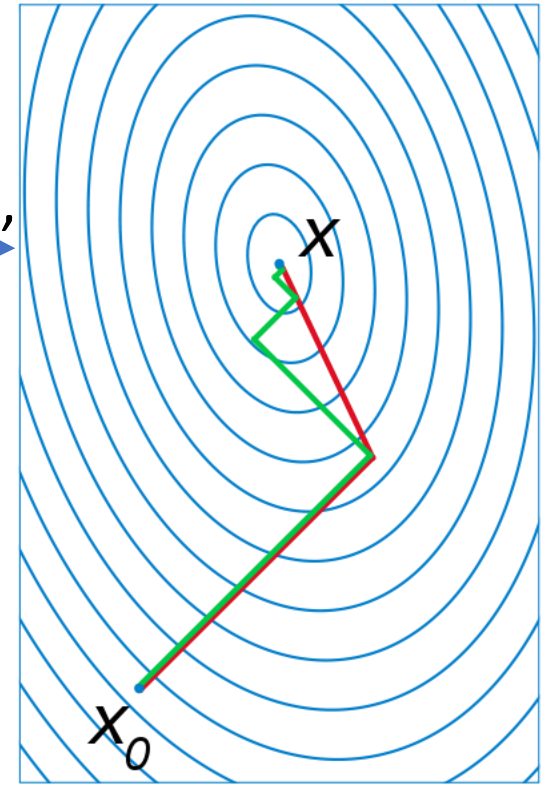
Usually much more saddles: $\binom{D}{\alpha D} \approx 2^{D\, h(\alpha)}$ assuming some randomness

α – percent of positive Hessian eigenvalues vs loss function:

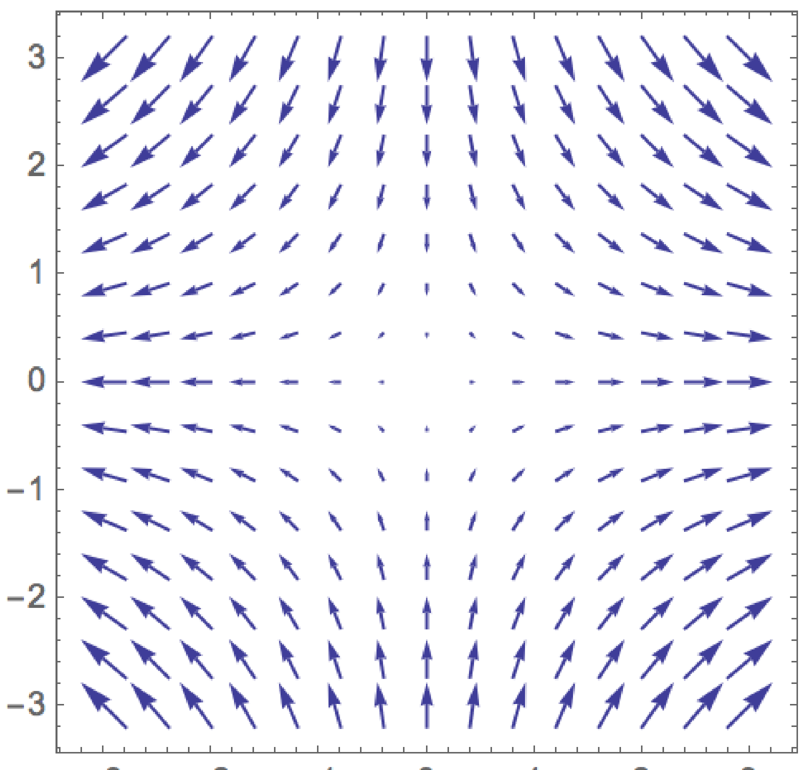


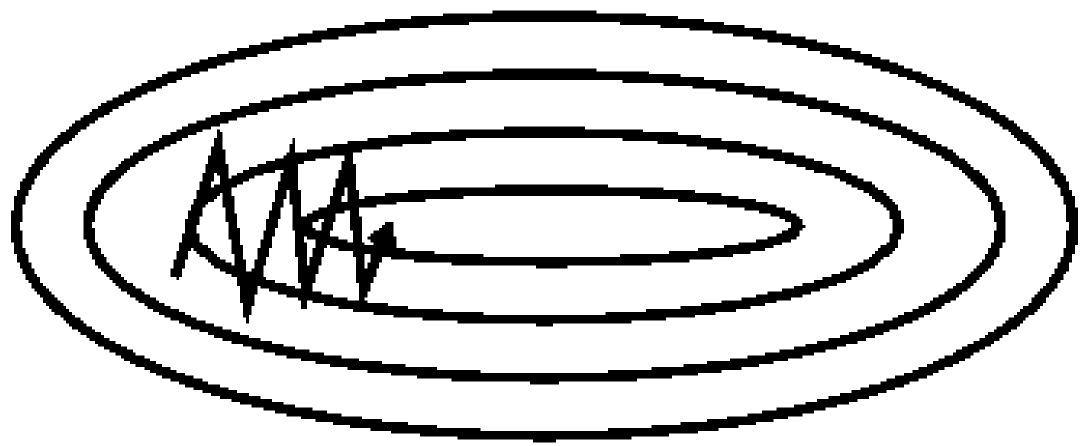
SGD (overview) – "**update** parameters **during** calculating gradient"

- Challenges from standard gradient descent:
  oscillations in high curvature directions,
  better conjugate gradients using also low curvature,
  saddles, also degenerated: with $\lambda_i = 0$
  often large **plateaus** especially near saddles
- Choosing **step size,** their **schedule?**
- plus huge dimension and **noisy** gradients:
  need to **extract statistical trends** …



$(x, -y)$





SGD optimization: $g^t = \nabla_\theta F^t(\theta^t)$ noisy, **averages** to $\nabla_\theta F(\theta^t)$

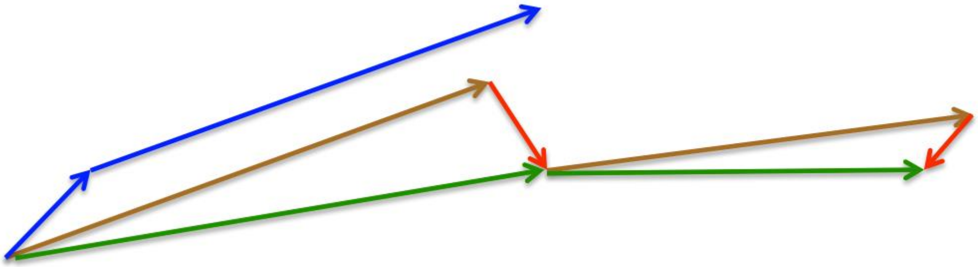
Momentum - use **exponential moving average** of stochastic gradients:

$$v^t = \gamma v^{t-1} + (1-\gamma) g^t \quad = (1-\gamma) g^t + \gamma(1-\gamma) g^{t-1} + \cdots$$

$$\theta^{t+1} = \theta^t - \eta v^t \qquad \gamma \approx 0.9$$

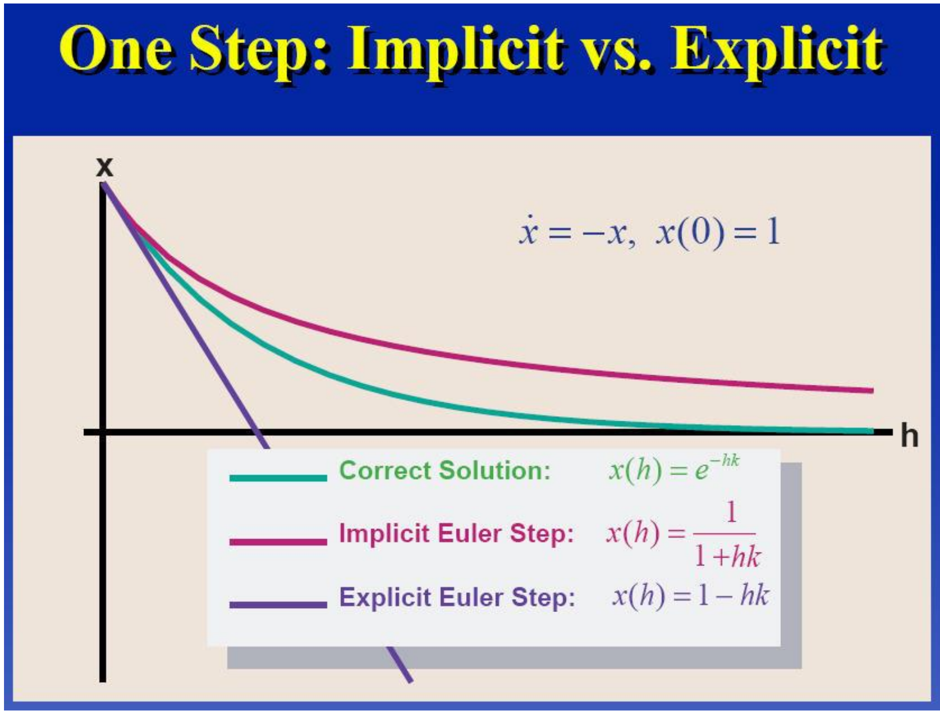
Nesterov accelerated gradient (NAG):

"implicit Euler momentum"



$$v^t = \gamma v^{t-1} + \eta \nabla_\theta F^t(\theta^t - \gamma v^{t-1})$$
$$\theta^{t+1} = \theta^t - v^t$$

plot



There are better ODE methods:

Higher order error $= \boldsymbol{O(h^{p+1})}$

Like Runge-Kutta?

Should we go this way for SGD???

Adagrad – larger updates for rare parameters (i), smaller for common

$$\theta_i^{t+1} = \theta_i^t - \frac{\eta}{\sqrt{G_i^t + \epsilon}} g_i^t \qquad G_i^t = G_i^{t-1} + \left(g_i^t\right)^2 \qquad \epsilon \approx 10^{-8}$$

RMSprop – Adagrad with exponential moving average instead of sum

$$\theta_i^{t+1} = \theta_i^t - \frac{\eta}{\sqrt{G_i^t + \epsilon}} g_i^t \qquad G_i^t = \gamma G_i^{t-1} + (1-\gamma)\left(g_i^t\right)^2$$

Adadelta – analogously estimate step size (diagonal Hessian approx.?)

$$\theta_i^{t+1} = \theta_i^t - \frac{\sqrt{\Delta_i^t + \epsilon}}{\sqrt{G_i^t + \epsilon}} g_i^t \qquad \Delta_i^t = \gamma \Delta_i^{t-1} + (1-\gamma)\left(\theta_i^t - \theta_i^{t-1}\right)^2$$

Adam (18k citations): Adadelta + bias while starting exp. moving avg.

$$m_i^t = \beta_1 m_i^{t-1} + (1-\beta_1) g_i^t \qquad v^t = \beta_2 v^{t-1} + (1-\beta_2)\left(g_i^t\right)^2$$

$$\theta^{t+1} = \theta^t - \frac{\eta}{\sqrt{v^t/(1-(\beta_2)^t) + \epsilon}} \frac{m^t}{1-(\beta_1)^t} \qquad \beta_1 = 0.9, \beta_2 = 0.999$$

$$m_i^t = \beta_1 m_i^{t-1} + (1-\beta_1) g_i^t \qquad v^t = \beta_2 v^{t-1} + (1-\beta_2)\left(g_i^t\right)^2$$

Adam: $$\theta^{t+1} = \theta^t - \frac{\eta}{\sqrt{v^t/(1-(\beta_2)^t)}+\epsilon} \frac{m^t}{1-(\beta_1)^t}$$

AdaMax – Adam with maximum norm for stability (?)

$$\theta^{t+1} = \theta^t - \frac{\eta}{u_i^t} \frac{m^t}{1-(\beta_1)^t} \qquad u_i^t = \max\left(\beta_2 u_i^{t-1}, \left|g_i^t\right|\right)$$

Nadam – Adam + Nesterov (m^t estimated one step forward)

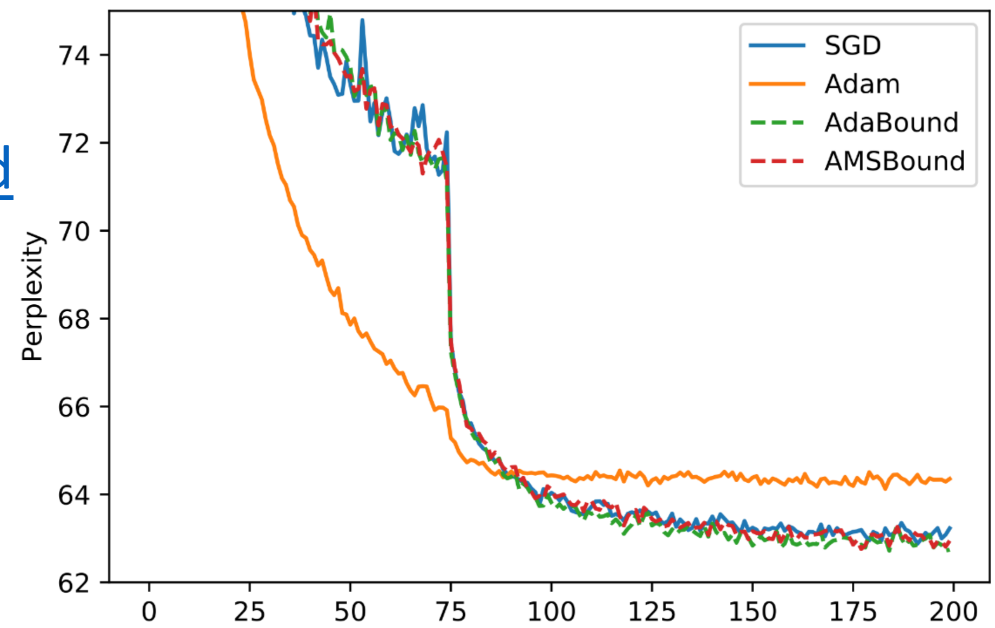
$$\theta^{t+1} = \theta^t - \frac{\eta}{\sqrt{v^t(1-(\beta_2)^t)}+\epsilon}\left(\frac{\beta_1 m^t + (1-\beta_1) g_t}{1-(\beta_1)^t}\right)$$

AMSGrad – Adam often suboptimal

AdaBound

$$\hat{v}^t = \max(\hat{v}^{t-1}, v^t)$$

$$\theta^{t+1} = \theta^t - \frac{\eta}{\sqrt{\hat{v}^t}+\epsilon} m^t$$



Lots of heuristics, based on experiments, tasks of various specifics …

They do not **estimate distance to extremum**, **trace only single direction**

Exponential number of saddles due to parameter permutation invariance

To quickly pass problematic **saddle** we could **model two parabolas** …

Maybe let's try to go to **second order methods** … not successful so far (?)

Newton-Raphson, $\boldsymbol{H} > \mathbf{0}$: $\nabla_\theta F(\theta) \approx \nabla_\theta F(\theta^t) + H(\theta^t) \cdot (\theta - \theta^t)$

$\nabla_\theta F(\theta) \approx 0$ for "**natural gradient**": $\theta - \theta^t = -H^{-1}(\theta^t) \cdot \nabla_\theta F(\theta^t)$

However, **huge dimensions** $(10^6 \dots 10^9)$ and **noisy gradients** – need to

extract statistics, **restrict Hessian**, **avoid inversion**, **avoid saddles** …

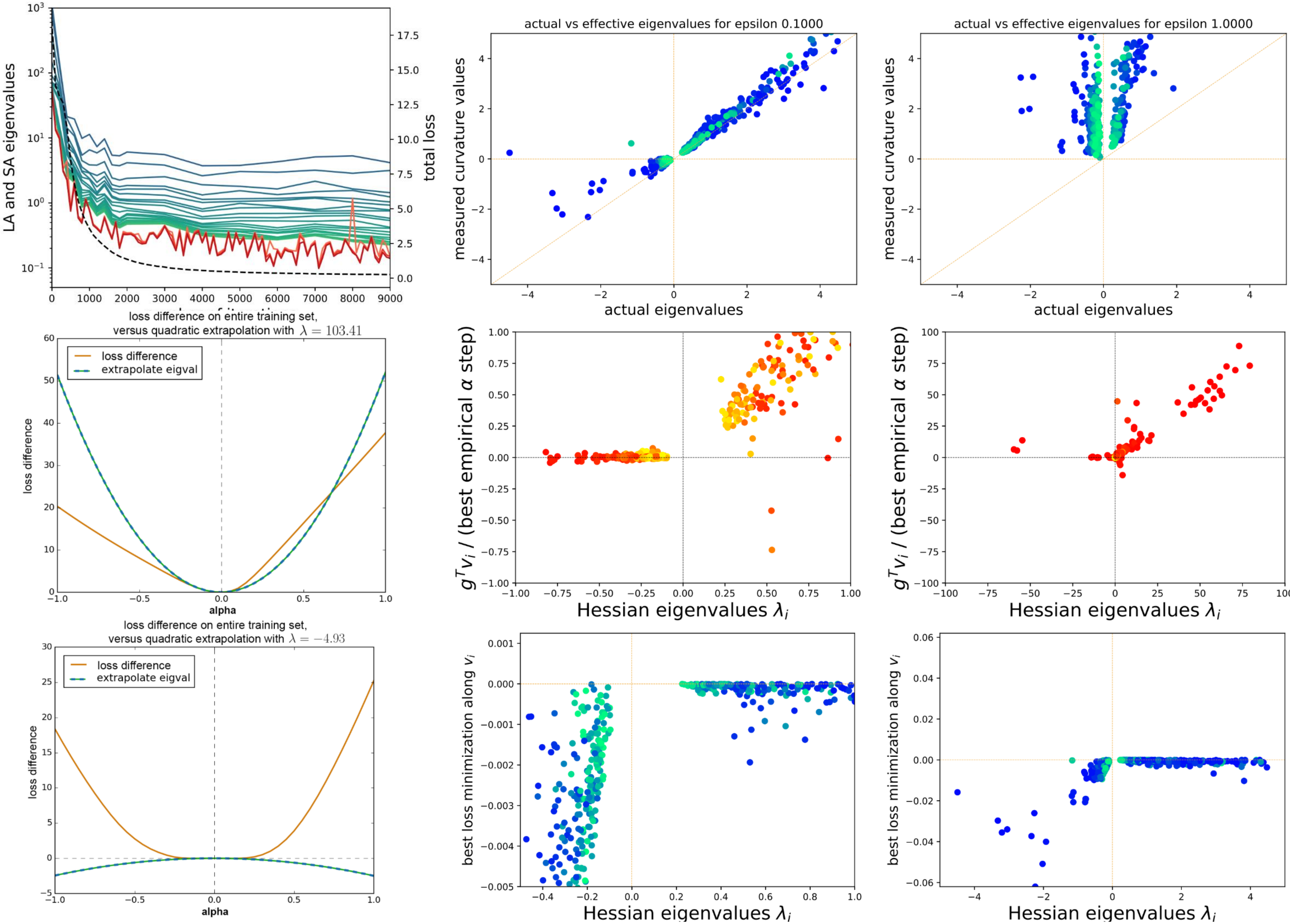
Let's start with simpler question (~"line search"):

How to handle 1D minimization asking for noisy derivatives?

$f = \lambda(x-p)^2/2$ $f' = \lambda(x-p)$ (expon. weight) linear regression?

Then: do it simultaneously for a few directions? How to explore them?

arXiv:1902.02366 RMSprop on MNIST, $d = 3.3 \cdot 10^6, F(\theta + \alpha(g \cdot v_i)v_i)$, early: blue/red



Saddle-free Newton (~600cit.) [GitHub]

https://arxiv.org/pdf/1406.2572

$-|H|^{-1}g$ direction $\qquad \lambda_i \to |\lambda_i|$

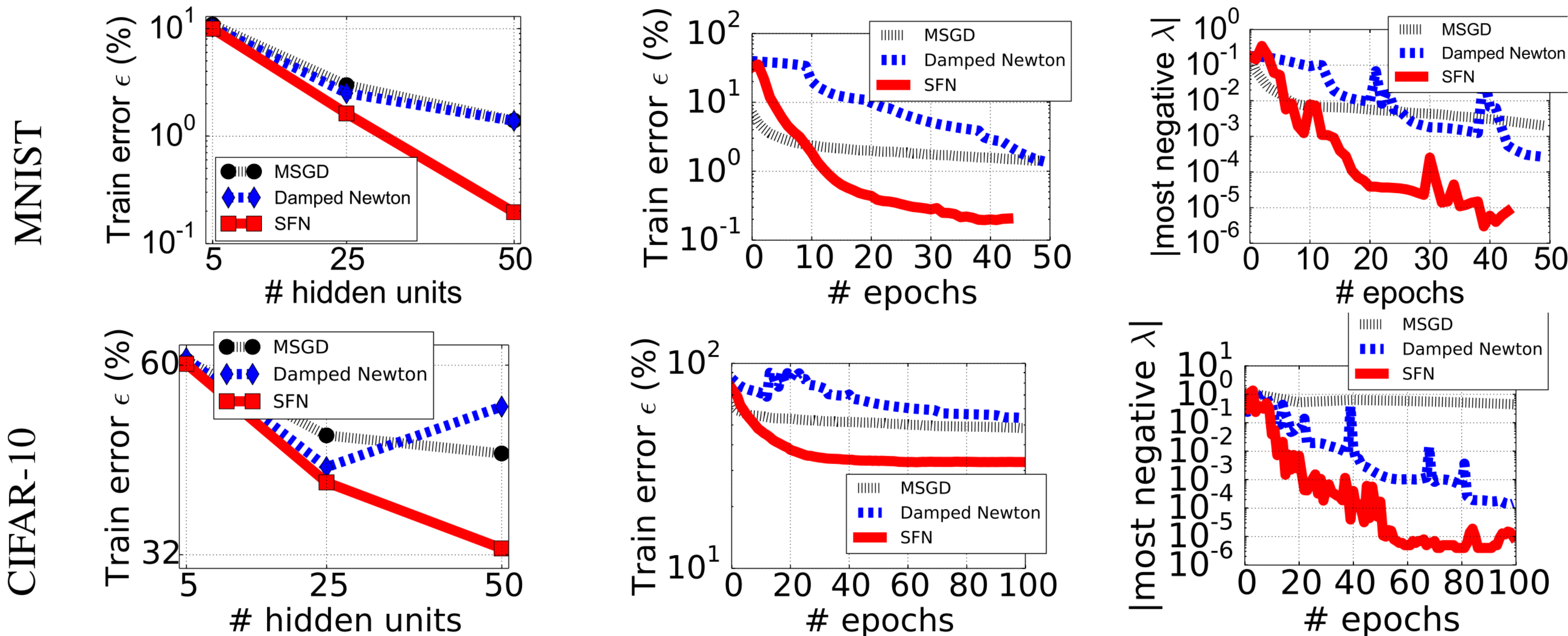
Lower dim. [Krylov]: $\text{span}\{v, Hv, \ldots, H^{k-1}v\}$

MSGD: minibatch SGD

Damped Newton: $-(H+\epsilon I)^{-1}g$

Algorithm 1 Approximate saddle-free Newton

Require: Function $f(\theta)$ to minimize
 for $i = 1 \to M$ **do**
  $\mathbf{V} \leftarrow k$ Lanczos vectors of $\frac{\partial^2 f}{\partial \theta^2}$
  $\hat{f}(\alpha) \leftarrow g(\theta + \mathbf{V}\alpha)$
  $|\hat{\mathbf{H}}| \leftarrow \left|\frac{\partial^2 \hat{f}}{\partial \alpha^2}\right|$ by using an eigen decomposition of $\hat{\mathbf{H}}$
  for $j = 1 \to m$ **do**
   $\mathbf{g} \leftarrow -\frac{\partial \hat{f}}{\partial \alpha}$
   $\lambda \leftarrow \arg\min_\lambda \hat{f}(\mathbf{g}(|\hat{\mathbf{H}}| + \lambda \mathbf{I})^{-1})$
   $\theta \leftarrow \theta + \mathbf{g}(|\hat{\mathbf{H}}| + \lambda \mathbf{I})^{-1}\mathbf{V}$
  end for
 end for



Natural gradient attracts to saddle, there are usually exp(dim) of them …

Gauss-Newton method: assume $F(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(f_i(\boldsymbol{\theta})\right)^2$ "sample errors"

$$H_{jk} = \frac{\partial F}{\partial\theta_j\,\partial\theta_k} = 2\sum_{i=1}^{n}\left(\frac{\partial f_i}{\partial\theta_j}\frac{\partial f_i}{\partial\theta_k} + f_i\frac{\partial^2 f_i}{\partial\theta_j\partial\theta_k}\right)$$

Assuming $\frac{\partial^2 f_i}{\partial\theta_j\partial\theta_k} = 0$, locally approximating f with linear functions:

$H_{jk} \approx 2\sum_{i=1}^{n} \frac{\partial f_i}{\partial\theta_j}\frac{\partial f_i}{\partial\theta_k}$ positive definite, only gradients needed (saddles?)

$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \left(J_f^T J_f\right)^{-1} J_f \boldsymbol{f}$ $\left(= \left(J_f\right)^{-1}\boldsymbol{f} \text{ for } n = D\right)$ $J_f = \frac{\delta f_i}{\delta\theta_j}(\boldsymbol{\theta}^t)$

Levenberg-Marquardt $(\lambda > 0)$: $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \left(J_f^T J_f + \lambda I\right)^{-1} J_f \boldsymbol{f}$

Other standard ways to **models Hessian**: **finite differences** (of gradients), or **backpropagating** – often dropping 2[nd] derivative of activation function

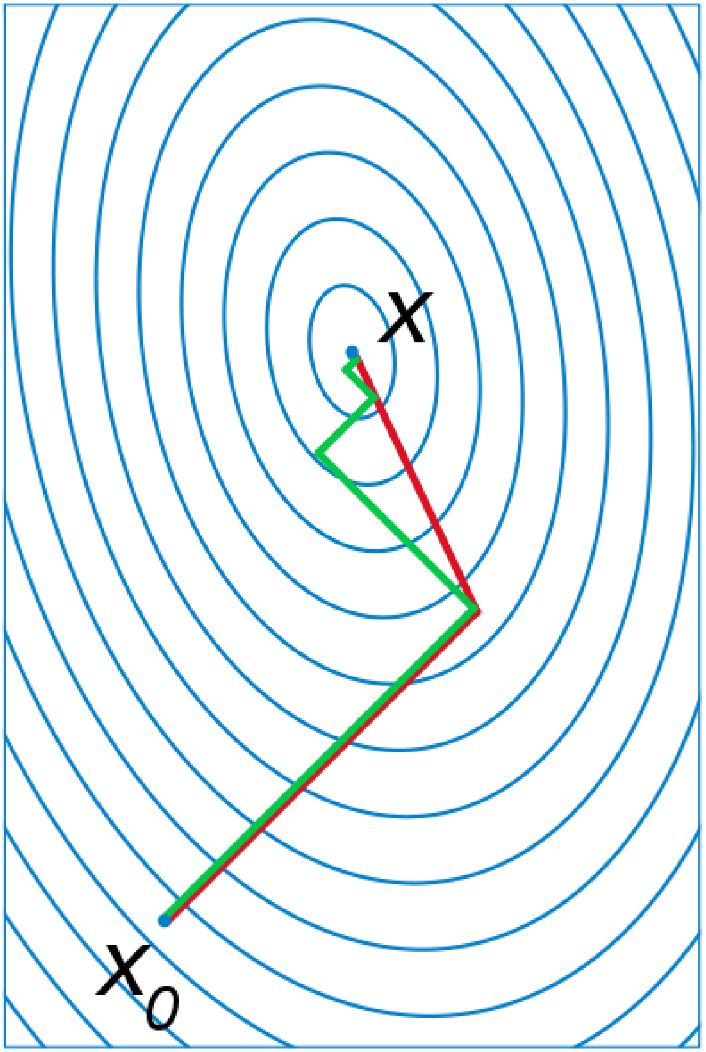
Classical second order: **conjugated gradients**

"unbend to Hessian eigenbasis" $\langle u, v\rangle_H := u^T H v$

$(v_1, \ldots, v_d)$: $\langle v_i, v_j\rangle_H = 0$ for $i \neq j$

$v_k = r_k - \sum_{i<k} \frac{v_i^T H\, r_k}{v_i^T H\, v_i} v_i$ $\qquad$ $\alpha_k = \frac{v_k \cdot r_k}{v_k^T H\, v_k}$

$v_{k+1} = x_k + \alpha_k v_k$



Learning: truncated Newton or

nonlinear conjugated gradients (NCG):

Line search $\underset{\alpha>0}{\operatorname{argmin}}\ F(x_t - \alpha v_t)$ toward

$v_t = \beta_t v_{t-1} - \nabla F(\theta^t)$ $\qquad$ (+ reset sometimes)

satisfying $v_t H v_{t-1} \approx 0$. From LeCun et al. "Efficient BackProp" (1998):

Fletcher and Reeves (1964), or Polak and Riberre (1969):

$$\beta_t = \frac{\nabla F(\theta^t)^T \nabla F(\theta^t)}{\nabla F(\theta^{t-1})^T \nabla F(\theta^{t-1})}$$

$$\beta_t = \frac{(\nabla F(\theta^t) - \nabla F(\theta^{t-1}))^T \nabla F(\theta^t)}{\nabla F(\theta^{t-1})^T \nabla F(\theta^{t-1})}$$

Quasi-Newton – instead of inverting Hessian, update approximation of H^{-1}

BFGS (Broyden-Fletcher-Goldfarb-Shanno ~1970), positive definite H

Secant condition that gradients agree: $H_n^{-1}\Delta g_n = \Delta x_n$

$$\nabla F_n(x_n) = g_n,\ \nabla F_n(x_{n-1}) = g_{n-1} \quad \Rightarrow \quad H_n(x_n - x_{n-1}) = (g_n - g_{n-1})$$

Find $\operatorname*{argmin}_{H^{-1}=(H^{-1})^T} \|H^{-1} - H_n^{-1}\|_F^2$ s.t. $H_n^{-1}\Delta g_n = \Delta x_n$ getting:

$$H_{n+1}^{-1} = \left(I - \frac{\Delta g_n (\Delta x_n)^T}{\Delta g_n \cdot \Delta x_n}\right) H_n^{-1} \left(I - \frac{\Delta x_n (\Delta g_n)^T}{\Delta g_n \cdot \Delta x_n}\right) + \frac{\Delta x_n (\Delta x_n)^T}{\Delta g_n \cdot \Delta x_n}$$

$$x_{n+1} = x_n - H_{n+1}^{-1} g_n$$

or **line search**: $\operatorname*{argmin}_{\alpha>0} F(x_n - \alpha H_{n+1}^{-1} g_n)$

L-BFGS: limited memory: store $m \approx 10$ last $\Delta x, \Delta g$ instead of huge H^{-1}

"two loop recursion": $i = n \ldots \searrow\ \ldots n - m \ldots \nearrow\ \ldots n$ using $H_{n-m}^{-1} \approx I$

Rough approximation, numerical problem with **noisy gradients** in SGD

K-FAC (2015): "Kronecker-factored approximate curvature", slides

Natural gradient approximating full Hessian with **block-diagonal**

full Hessian inside **layers**, no inter-layer correlations (tri-block-diagonal…)

Huge cost: **~ dimension²/#layers** noisy estimations … **saddles** $(\approx \boldsymbol{H > 0})$

Maximizing likelihood with density f_θ: $F(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ln(f_\theta(x_i))$

$$\sum_{ij} u_i E\left[\left(\frac{\partial \ln f_\theta(X)}{\partial \theta_i}\frac{\partial \ln f_\theta(X)}{\partial \theta_j}\right)\right] u_j = E\left[\left(\sum_i u_i \frac{\partial \ln f_\theta(X)}{\partial \theta_i}\right)^2\right] > 0$$

Fisher information

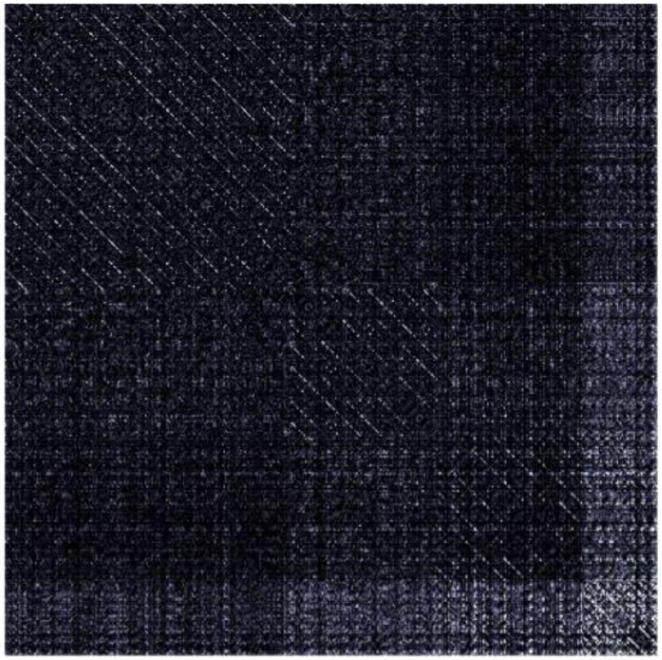
is positive definite

describes parameter

dependence/certainty

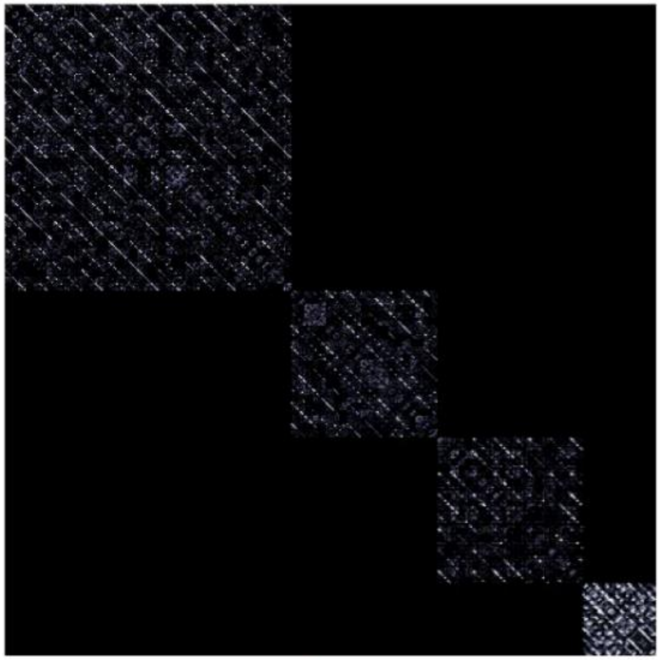
~Gauss-Newton

Hessian approx.



H_k

$\approx$



$\hat{H}_k$

A fast **natural Newton** method (2010) – TONGA introduction + repair

Use (~PCA) correlation of recent gradients instead of Hessian

Assume $\boldsymbol{g}$: **"true" gradient**, $\hat{\boldsymbol{g}}$: **"empirical gradient"** from Gaussian:

$\hat{g}|g \sim \mathcal{N}(g, C/n)$ for **centered covariance matrix** C

$C = \int_x \left(\frac{\partial f(\theta,x)}{\partial\theta} - g\right)\left(\frac{\partial f(\theta,x)}{\partial\theta} - g\right)^T p(x)\,dx$ isotropic $g \sim \mathcal{N}(0, \sigma^2 I)$

$\hat{g}|g \sim \mathcal{N}\left(\left[I + \frac{C}{n\sigma^2}\right]^{-1}\hat{g}, [nC^{-1} + \sigma^{-2}I]^{-1}\right)$ saddle?

$\Delta\theta \propto -\left[I + \frac{\hat{C}}{n\sigma^2}\right]^{-1}\hat{g}$ for $C \approx \hat{C} = \frac{1}{n}\sum_{i=1}^n \left(\frac{\partial f(\theta,x_i)}{\partial\theta} - g\right)\left(\frac{\partial f(\theta,x_i)}{\partial\theta} - g\right)^T$

2010 improvement (full dimension?):

Assuming $F(\theta) \approx \frac{1}{2}(\theta-\theta^*)^T H(\theta-\theta^*)$ prior $g \sim \mathcal{N}(0, \sigma^2 H)$

$\Delta\theta \propto -\left[I + \frac{H^{-1}\hat{C}H^{-1}}{n\sigma^2}\right]^{-1}\hat{g}$ Hessian H from Quasi-Newton

Uncentered covariance matrix from exponential moving average …

TONGA (2008): reduced dimension, Le Roux, Bengio, Manzagol

non-centered covariance matrix from exponential moving average:

$C_t = \gamma \hat{C}_t + g_t g_t^T \approx X_t X_t^T$ $\qquad$ X_t is $D \times d$ for some $d \ll D$

(standard (centered) covariance matrix: $\frac{1}{n}\sum_{i=1}^{n}(g_i - \bar{g})(g_i - \bar{g})^T$)

Regularized **natural gradient**:

$v_t = (C_t + \lambda I)^{-1} g_t = X_t (X_t^T X_t + \lambda I)^{-1} y_t$ in $O(Dd + d^3)$ time

$\hat{C}_t$ – low rank approximation of C – keep only $k < d$ eigenvalues

$G_t = X_t X_t^T = VDV^T$ $\qquad$ $C_t = \left(X_t V D^{-1/2}\right) D \left(X_t V D^{-\frac{1}{2}}\right)^T$

eigendecomposition at cost $O(kd^2 + Ddk)$ every few steps

… maybe let's try to directly model and update only what we really need …

Update local parametrization (saddles), put eigendecomposition in iteration

Online gradient linear regression: let's start with 1D **fixed parabola**:

$f(\theta) = \frac{1}{2}\lambda(\theta - p)^2$ from noisy $g^t \approx f'(\theta^t) = \lambda(\theta^t - p)$

Least-square linear regression: $\arg\min_{\lambda,p} \sum_t w^t \left(g^t - \lambda(\theta^t - p)\right)^2$

$$\lambda = \frac{\overline{g\theta} - \overline{g} \cdot \overline{\theta}}{\overline{\theta^2} - \overline{\theta}^2} \qquad p = \frac{\lambda\overline{\theta} - \overline{g}}{\lambda} \qquad \text{e.g.} \quad \overline{g\theta} = \frac{1}{T}\sum_t g^t \theta^t$$

For $\overline{g}, \overline{\theta^2}, \overline{g\theta}, \overline{\theta^2}$ $\quad w^t = 1/T$ averages ....

In **general (non-parabola) case**: use **exponential moving average**

$w^t \sim \beta^{-t}$ **online** averages, e.g. $\overline{g\theta}^t = \beta\, \overline{g\theta}^{t-1} + (1-\beta)\, g^t \theta^t$

In D dimensions: do it in a few $d \ll D$ statistically relevant $(v_i)_{i=1..d}$

Attract $(\lambda > 0)$ **or repel** $(\lambda < 0)$ correspondingly to handle saddles

$\theta \leftarrow \theta + \alpha \sum_{i=1}^{d} \operatorname{sign}(\lambda_i)(p_i - \theta \cdot v_i)\, v_i$ **proper optimization step**

Maintain ≈**diagonal Hessian** by periodic diagonalization, orthogonalization

$$\mathrm{O}^{\mathrm{T}}\Lambda\mathrm{O} = \quad H = \left(\overline{g\theta} - \overline{g}\,\overline{\theta}^T\right)\left(\overline{\theta\theta} - \overline{\theta}\,\overline{\theta}^T\right)^{-1} \quad \text{, e.g.} \quad \overline{\theta\theta}_{ij} = \overline{\theta_i\theta_j}$$

Rotate subspace (online) to explore recent relevant directions

Many tough questions ….

- Should we ask for **gradients**, or maybe (also?) **values**, **2^{nd} derivatives**?
  Gradients: nice compromise – suggests direction, cheaper than Hessian
- **How many directions** to model? One (now) … a few … all (full Hessian)?
- How to **choose interesting directions** based on recent gradients?
- Online: **mini-batch size?** Step cost vs uncertainty?
- Strengthening **rarely represented coordinates** like $g_i/\langle g_i^2\rangle^{1/2}$?
- How to efficiently pass **plateaus**, **saddles**?
- How to efficiently handle noise – **extract statistical trends**?
- How frequent are saddles? (much more than minima)
- How bad are **positive Hessian approximations**? $F = \sum_i (f_i)^2$ only?
  aren't they attracting to saddles?

Simpler warmup question: how to optimally handle 1D problem?

… how to optimize problem to reduce iteration number…? [To one?]