Optimization for Training Deep Models

ADVANCED ARTIFICIAL INTELLIGENCE
JUCHEOL MOON

1

Indirect learning

- In most machine learning scenarios, we care about some performance measure P
  - that is defined with respect to the test set.
- We therefore optimize P only indirectly.
  - We reduce a different cost function $J(\theta)$
  - in the hope that doing so will improve P.
  - typically optimizes a surrogate loss function
- Why?

$\nabla_{\vec{w}} P \Rightarrow \frac{\partial P}{\partial w_i} = 0$ est.

	actual: true	actual: false
true	n_1	n_2
false	n_3	n_4

$w_i \leftarrow w_i - \eta \frac{\partial P}{\partial w_i}$

$\frac{\partial P}{\partial w_i} \simeq 0$ $= 0$

2

2

Empirical risk minimization

- Minimize the corresponding objective function
  - $J^*(\vec{\theta}) =$ $\mathbb{E}_{(\vec{x},y)\sim p_{data}} L(f(\vec{x};\vec{\theta}), y)$ $(\hat{y} = f(\vec{x};\vec{\theta}))$
  - The goal of a machine learning algorithm is to reduce the expected generalization error
- However, we do not know p_{data}.
  - Minimize the empirical risk
  - $J(\vec{\theta}) =$ $\mathbb{E}_{(\vec{x},y)\sim \hat{p}_{data}} L(f(\vec{x};\vec{\theta}), y)$
  - $\mathbb{E}_{(\vec{x},y)\sim \hat{p}_{data}} L(f(\vec{x};\vec{\theta}), y) =$ $\frac{1}{m}\sum_{i}^{m} L(f(\vec{x}^{(i)};\vec{\theta}), y^{(i)})$

3

3

Minibatch

$J = \frac{1}{m}\sum_{i}^{m} L$

- Optimization algorithms that use the entire training set are called batch or deterministic gradient methods
- Optimization algorithms that use only a single example at a time are called stochastic or online methods
  - The term online is reserved for the case where the examples are drawn from a stream of continually created example
- Using more than one but less than all
  - traditionally called minibatch or minibatch stochastic method (simply Stochastic)

4

4

Minibatch

- Larger batches provide a more accurate estimate of the gradient
- It is common for power of 2 batch sizes to offer better runtime. Typical batch sizes range from 32 to 256
- Generalization error is often best for a batch size of 1.
  - Training with such a small batch size might require a small learning rate
  - The total runtime can be very high due to the need to make more steps

5

5

Jacobian matrix

- The matrix containing all partial derivatives of a function whose input and output are both vectors.
  - $\vec{f}: \mathbb{R}^m \to \mathbb{R}^n$,
  - $\vec{J} \in \mathbb{R}^{m \times n}$
  - $J_{i,j} = \frac{\partial}{\partial x_j} f(\vec{x})_i$
  - $\vec{J}(f)(\vec{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} f(\vec{x})_1 & \cdots & \frac{\partial}{\partial x_m} f(\vec{x})_1 \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial x_1} f(\vec{x})_n & \cdots & \frac{\partial}{\partial x_m} f(\vec{x})_n \end{bmatrix}$

$f: \mathbb{R}^n \to \mathbb{R}$ $\quad D_{\vec{x}} f = \left[\frac{\partial f}{\partial x_1} \frac{\partial f}{\partial x_2} \cdots \frac{\partial f}{\partial x_n}\right]^T$

$f(\vec{x}) = [f(\vec{x})_1, f(\vec{x})_2 \cdots f(\vec{x})_i \cdots]^T$

6

6

Hessian matrix

- The matrix containing all second-order partial derivatives of a function whose input is a vector and output is a scalar.
  - $\vec{f}: \mathbb{R}^n \to \mathbb{R}$, $f(x_1, \cdots, x_n) = y$
  - $\vec{H} \in \mathbb{R}^{n \times n}$
  - $H_{i,j} = \frac{\partial}{\partial x_i \partial x_j} f(\vec{x}) = \frac{\partial}{\partial x_j \partial x_i} f(\vec{x})$
  - $\vec{H}(f)(\vec{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1 \partial x_1} f(\vec{x}) & \cdots & \frac{\partial}{\partial x_1 \partial x_n} f(\vec{x}) \\ \vdots & & \\ \frac{\partial}{\partial x_n \partial x_1} f(\vec{x}) & \cdots & \frac{\partial}{\partial x_n \partial x_n} f(\vec{x}) \end{bmatrix}$

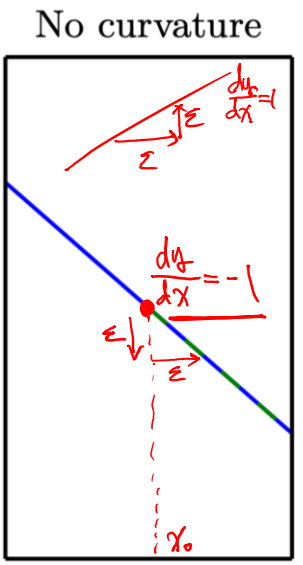
7

7

Curvature

- We can think of the second derivative as measuring curvature.
  - Suppose we have a ~~quadratic~~ linear function
  - a second order derivative of zero
    - then there is no curvature
    - If the gradient is -1
      - then we can make a step of size ϵ along the gradient, and the cost function will decrease by ϵ

$y = x \to \frac{dy}{dx} = 1 \to \frac{d^2y}{dx^2} = 0$

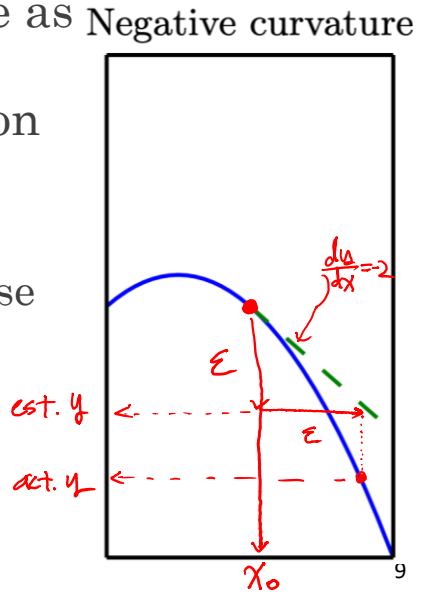


8

8

Curvature

- We can think of the second derivative as measuring curvature.
  - Suppose we have a quadratic function
  - If the second derivative is negative,
    - the function curves _____downward_____,
    - so the cost function will actually decrease by _____more_____ than ϵ



$y=-x^2 \rightarrow \frac{dy}{dx}=-2x \rightarrow \frac{d^2y}{dx^2}=-2$

9

Taylor series approximation

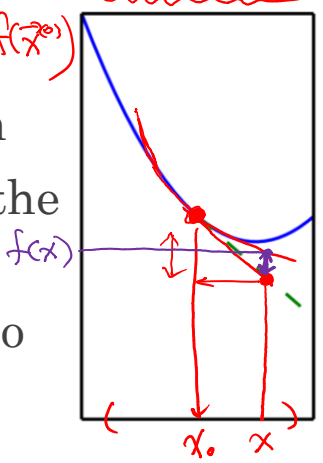
- Taylor series approximation around $x^{(0)}$
  - $f(x) \approx f(x^{(0)}) + f'(x^{(0)})(x-x^{(0)}) + \frac{1}{2}f''(x^{(0)})(x-x^{(0)})^2$
- Taylor series approximation around $\vec{x}^{(0)}$
  - $f(\vec{x}) \approx f(\vec{x}^{(0)}) + (\vec{x}-\vec{x}^{(0)})^T \vec{g} + \frac{1}{2}(\vec{x}-\vec{x}^{(0)})^T \vec{H} (\vec{x}-\vec{x}^{(0)})$
  - where $\vec{g} = \nabla_{\vec{x}} f(\vec{x}^{(0)})$ and $\vec{H} = \vec{H}(f)(\vec{x}^{(0)})$
- With a learning rate ϵ
  - the new point $\vec{x} = \vec{x}^{(0)} - \epsilon \vec{g}$
  - $f(\vec{x}) \approx f(\vec{x}^{(0)}) - \epsilon \vec{g}^T \vec{g} + \frac{1}{2}\epsilon^2 \vec{g}^T H^T \vec{g}$

$(\vec{x}-\vec{x}^{(0)} = (\vec{x}^{(0)} - \epsilon\vec{g}) - \vec{x}^{(0)} = -\epsilon\vec{g})$

10

Optimal step size

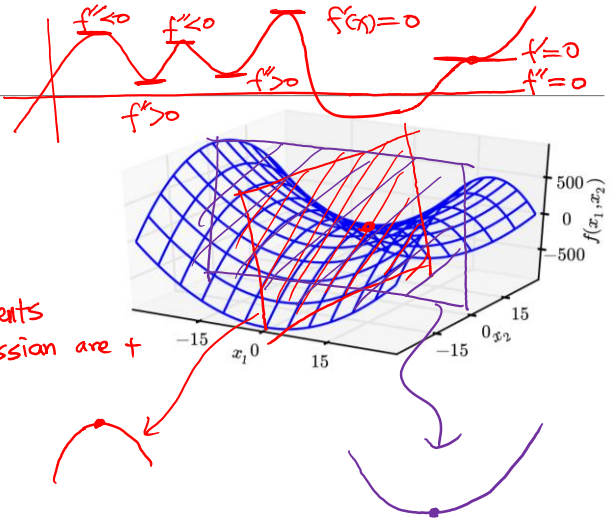
- $f(\vec{x}) \approx f(\vec{x}^{(0)}) - \epsilon \vec{g}^T \vec{g} + \frac{1}{2}\epsilon^2 \vec{g}^T \vec{H} \vec{g}$
- $f(\vec{x}^{(0)})$: the original value of the function
- $\epsilon \vec{g}^T \vec{g}$: the expected improvement due to the slope of the function
- $\frac{1}{2}\epsilon^2 \vec{g}^T \vec{H} \vec{g}$: the correction we must apply to account for the curvature of the function.
  - If step size $\epsilon = 2\frac{\vec{g}^T \vec{g}}{\vec{g}^T \vec{H} \vec{g}}$,
  - Heuristic choice $\epsilon^* = \frac{\vec{g}^T \vec{g}}{\vec{g}^T \vec{H} \vec{g}}$



11

Saddle Points

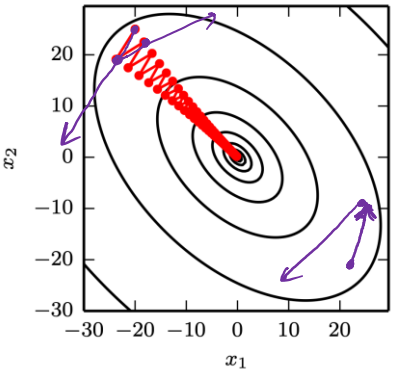
- $f'(x) = 0$
  - $f''(x) > 0$: local minimum
  - $f''(x) < 0$: local maximum
  - $f''(x) = 0$: flat
- $\nabla_{\vec{x}} f(\vec{x}) = 0$
  - Hessian are positive definite
    - local minimum
  - Hessian are negative definite
    - local maximum
  - $\vec{x}$ is a local maximum on one cross section of f but a local minimum on another cross section



12

Optimization based on the Hessian

- Condition number of Hessian
  - E.g., condition number 5
    - The direction of most curvature has five times more curvature than the direction of least curvature.
- The steepest direction is not actually a promising search direction in this context



13

13

Ill-Conditioning

- $f(\vec{x}) \approx f(\vec{x}^{(0)}) - \epsilon \vec{g}^T \vec{g} + \frac{1}{2}\epsilon^2 \vec{g}^T \vec{H} \vec{g}$
- A gradient descent step of $-\epsilon \vec{g}$ will add
  - $\frac{1}{2}\epsilon^2 \vec{g}^T \vec{H} \vec{g} - \epsilon \vec{g}^T \vec{g}$
- Ill-conditioning of the gradient becomes a problem when $\frac{1}{2}\epsilon^2 \vec{g}^T \vec{H} \vec{g} > \epsilon \vec{g}^T \vec{g}$
- To determine whether ill-conditioning is detrimental to a neural network training task?

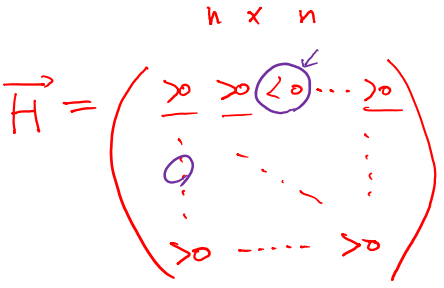
monitoring $\vec{g}$ & $\vec{H}$

14

14

Saddle Points

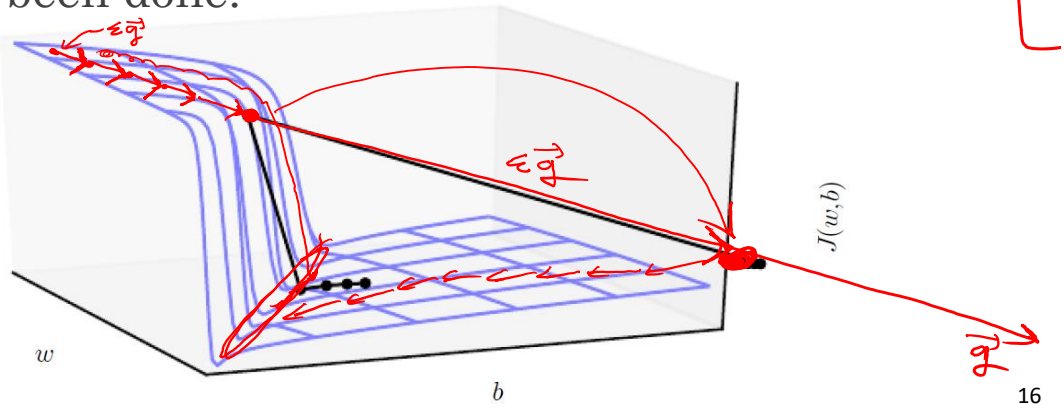
- In higher dimensional spaces, local minima are rare and saddle points are more common.
  - For a function $f: \mathbb{R}^n \to \mathbb{R}$ of this type, the expected ratio of the number of saddle points to local minima grows exponentially with n.
- Why?
  - $\nabla_{\vec{x}} f(\vec{x}) = 0$
    - Hessian are positive definite
      - local minimum
    - Hessian are negative definite
      - local maximum



15

Cliff

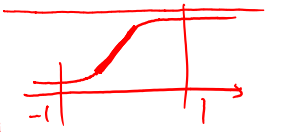
- When the parameters get close to such a cliff region, a gradient descent update can catapult the parameters very far, possibly losing most of the optimization work that had been done.



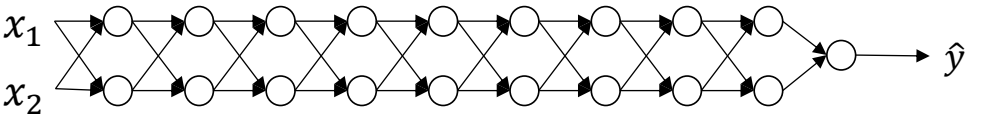
16

Vanishing and exploding

- Suppose that a computational graph contains a path that consists of repeatedly multiplying by a matrix $\vec{W}$ and that $\vec{W}$ has an eigen decomposition
  - $\vec{W} = \vec{V} diag(\lambda) \vec{V}^{-1}$
  - $(\vec{W})^t = (\vec{V} diag(\lambda) \vec{V}^{-1})^t = \vec{V} diag(\lambda)^t \vec{V}^{-1}$



$= \vec{V} diag(\lambda) \vec{V}^{-1} \cdot \vec{V} diag(\lambda) \vec{V}^{-1} \cdot \ldots \cdot \vec{V} diag(\lambda) \vec{V}^{-1}$



17

Stochastic Gradient Descent

- A crucial parameter for the SGD algorithm is the learning rate. In practice, it is gradually decreased over the iteration k and denoted by ϵ_k

Algorithm 8.1 Stochastic gradient descent (SGD) update at training iteration k

Require: Learning rate ϵ_k.
Require: Initial parameter $\boldsymbol{\theta}$
while stopping criterion not met **do**
  Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
  Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$ ($\hat{y}^{(i)}$)
  Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \hat{\boldsymbol{g}}$
end while

18

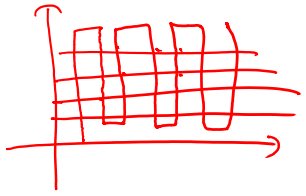
Stochastic Gradient Descent

- The learning rate may be chosen by trial and error, but it is best to choose it by monitoring learning curves that plot the objective function as a function of time.
  - This is more of an art than a science
  - $\epsilon_k = \left(1 - \frac{k}{\tau}\right)\epsilon_0 + \frac{k}{\tau}\epsilon_\tau$ if $k \le \tau$
    $= \epsilon_\tau$ o.w.
  - τ: iterations required to make a few hundred passes
  - ϵ_τ: 1% the value of ϵ_0

19

Stochastic Gradient Descent

- $\epsilon_k = \left(1 - \frac{k}{\tau}\right)e_0 + \frac{k}{\tau}\epsilon_\tau \ (k \le \tau)$
- The main question is how to set ϵ_0
  - If it is too large, the learning curve will show oscillations
  - If the learning rate is too low, learning proceeds slowly
- It is usually best to monitor the first several iterations and use a learning rate that is higher than the best-performing learning rate at this time



20

Moving average

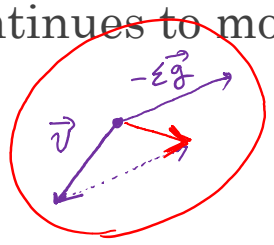
- Creating a series of averages of different subsets of the full data set

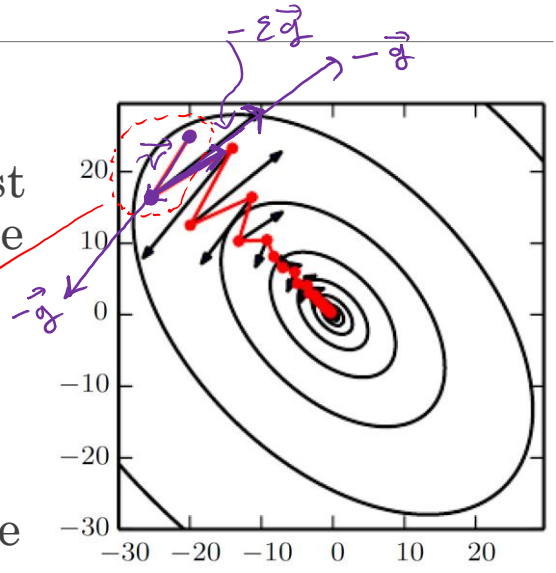
21

21

Momentum

- The momentum algorithm accumulates an exponentially decaying moving average of past gradients and continues to move in their direction
  - $\vec{v} \leftarrow \alpha\vec{v} - \varepsilon\vec{g}$
  - $\vec{\theta} \leftarrow \vec{\theta} + \vec{v}$
  - where $\alpha \in [0,1)$
  - The velocity $\vec{v}$ accumulates the gradient elements





22

22

Momentum

- α may also be adapted over time.
  - Typically, it begins with a small value and is raised

Algorithm 8.2 Stochastic gradient descent (SGD) with momentum

Require: Learning rate ϵ, momentum parameter α.
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$.
while stopping criterion not met **do**
  Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
  Compute gradient estimate: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
  Compute velocity update: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \boldsymbol{g}$
  Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$
end while

23

Parameter Initialization Strategies

- Training deep models is a sufficiently difficult task that most algorithms are strongly affected by the choice of initialization.
- Modern initialization strategies are simple and heuristic.
- break symmetry
  - If two hidden units with the same activation function are connected to the same inputs, then these units must have different initial parameters.
  - noticeable computational cost

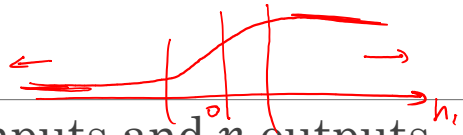
24

Random initialization

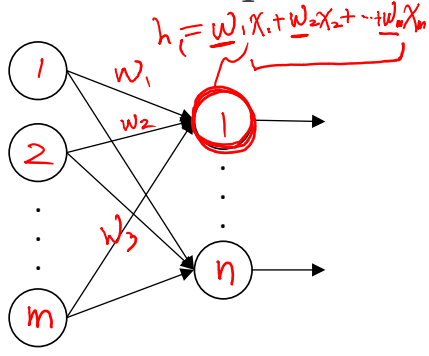
- Random initialization is computationally cheaper and unlikely to assign any units to compute the same function as each other.
  - We set the biases for each unit to heuristically chosen constants,
  - and initialize only the weights randomly from a uniform or Gaussian distribution
- Larger initial weights will yield a stronger symmetry breaking effect, however, result in exploding values during forward propagation or back-propagation.

25

Heuristic initialization



- A fully connected layer with m inputs and n outputs
  - $U\left(-\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}}\right)$
  - $U\left(-\frac{6}{\sqrt{m+n}}, \frac{6}{\sqrt{m+n}}\right)$
- One drawback to scaling rules
  - Weight becomes extremely small when the layers become large

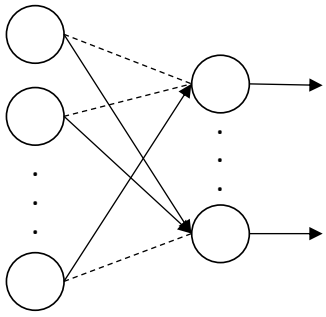


- Setting the biases to zero is compatible with most weight initialization schemes.

26

Sparse initialization

- Each unit is initialized to have exactly k non-zero weights
  - Keep the total amount of input to the unit independent from the number of inputs m without making the magnitude of individual weight elements shrink with m.



27

27

Adaptive Learning Rates

- The learning rate was reliably one of the hyperparameters that is the most difficult to set because it has a significant impact on model performance

- Early heuristic approach (delta-bar-delta)
  - If the partial derivative of the loss remains the same sign, then the learning rate should increase
  - If the partial derivative with respect to that parameter changes sign, then the learning rate should decrease

28

28

AdaGrad

- The parameters with the largest partial derivative of the loss have a correspondingly rapid decrease in their learning rate
- Empirically it has been found that the accumulation of squared gradients from the beginning of training can result in a premature and excessive decrease in the effective learning rate

Accumulate squared gradient: $\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g}$
Compute update: $\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta + \sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$. (Division and square root applied element-wise)
Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

29

RMSProp

- RMSProp uses an exponentially decaying average to discard history from the extreme past
- Empirically, RMSProp has been shown to be an effective and practical optimization algorithm for deep neural networks.
- It is currently one of the default optimization methods being employed routinely.

Accumulate squared gradient: $\boldsymbol{r} \leftarrow \rho\boldsymbol{r} + (1-\rho)\boldsymbol{g} \odot \boldsymbol{g}$
Compute parameter update: $\Delta\boldsymbol{\theta} = -\frac{\epsilon}{\sqrt{\delta + \boldsymbol{r}}} \odot \boldsymbol{g}$. ($\frac{1}{\sqrt{\delta+\boldsymbol{r}}}$ applied element-wise)
Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

30

Adam

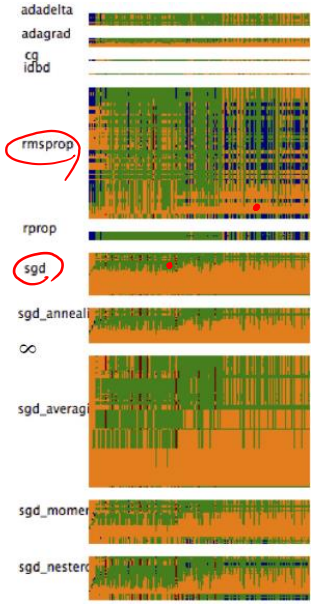
- Combination of RMSProp and momentum
- Adam is generally regarded as being fairly robust to the choice of hyperparameters, though the learning rate sometimes needs to be changed from the default.

$t \leftarrow t + 1$
Update biased first moment estimate: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g}$
Update biased second moment estimate: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}$
Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1-\rho_1^t}$
Correct bias in second moment: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1-\rho_2^t}$
Compute update: $\Delta\boldsymbol{\theta} = -\epsilon\frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}}+\delta}$ (operations applied element-wise)
Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

31

Choosing the Right Optimization Algorithm

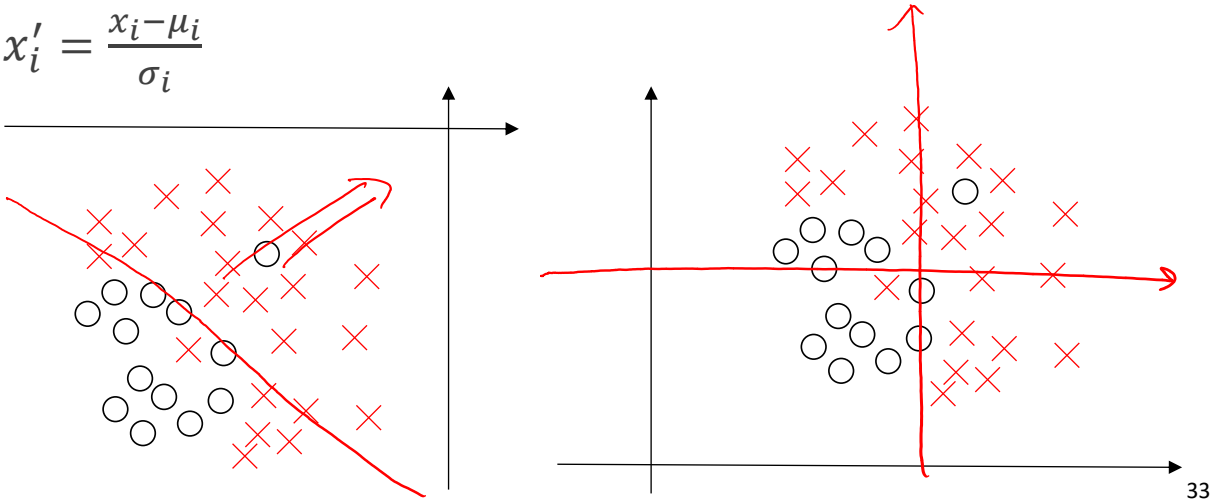
- Which algorithm should one choose?
  - The choice of which algorithm to use seems to depend largely on the user's familiarity with the algorithm
- The color code
  - red/violet=divergence
  - orange=slow
  - yellow=variability
  - green=acceptable
  - blue=excellent



32

Shifted distribution

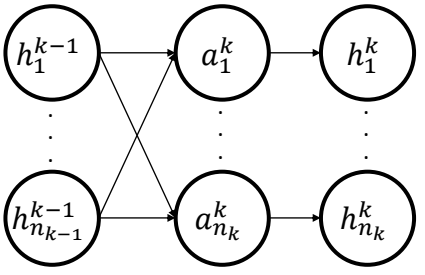
- What we have if a network is trained by two distributions?
  - $x_i' = \frac{x_i - \mu_i}{\sigma_i}$



33

Batch Normalization

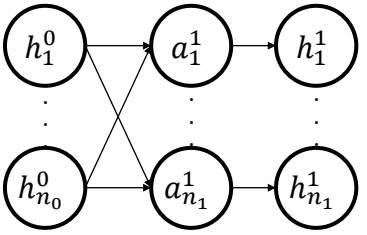
- Batch normalization can be applied to any input or hidden layer in a network and can be applied to h values or a values
- $\vec{\mu}^k = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_{n_k} \end{bmatrix} = \begin{bmatrix} \frac{1}{m}\sum_i a_1^k \\ \vdots \\ \frac{1}{m}\sum_i a_{n_k}^k \end{bmatrix}$
- $\sigma_j^{k\,2} = \frac{1}{m}\sum_i (a_j^k - \mu_j)^2$
- $h_j'^k = \frac{a_j^k - \mu_j^k}{\sqrt{\sigma_j^{k\,2} + \delta}}$ δ small positive



34

Batch Normalization

- $a'^k_j = \frac{a^k_j - \mu^k_j}{\sqrt{\sigma^{k^2}_j + \delta}}$
- It is common to introduce learned parameters γ, β
  - $\tilde{a}^k_j = \gamma^k_j a'^k_j + \beta^k_j$
- $\vec{x} \rightarrow \vec{h}_0 \rightarrow \vec{a}^1 \rightarrow \tilde{\vec{a}}^1 \rightarrow g(\tilde{\vec{a}}^1) \rightarrow \vec{h}^1 \rightarrow \cdots$
- On test time
  - Use μ, σ that were collected during training time



35