Winnowing with Gradient Descent

Ehsan Amid
Manfred K. Warmuth
Google Brain & UCSC

COLT 2020

Two main families of updates

Additive updates:
GD: stochastic gradient descent, backprop, Newton's update, kernel methods
Multiplicative updates:
EG: expert algorithms, Boosting, Bayes
EGU: Winnow

Performance of GD linear in n for sparse targets

Performance of EG & EGU grows as $\log n$ for sparse targets

Here we will reparameterize EG & EGU as GD:
Reparameterized forms act like original EG & EGU

Winnowing with GD!

Two main families of updates

Additive updates:
GD: stochastic gradient descent, backprop, Newton's update, kernel methods
Multiplicative updates:
EG: expert algorithms, Boosting, Bayes
EGU: Winnow

Performance of GD linear in n for sparse targets

Performance of EG & EGU grows as $\log n$ for sparse targets

Here we will reparameterize EG & EGU as GD:
Reparameterized forms act like original EG & EGU

Winnowing with GD!

Paradigmatic sparse linear problem

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix}$$

Hadamard matrix or random $\pm$

After receiving example $(\boldsymbol{x}_t, y_t)$
and incurring loss $(\boldsymbol{x}_t^\top \boldsymbol{w}_t - y_t)^2$ update:

multiplicative, EGU: $w_{t+1,i} = w_{t,i} \exp(-2\eta(\boldsymbol{x}_t^\top \boldsymbol{w}_t - y_t)x_{t,i})$

additive, GD: $w_{t+1,i} = w_{t,i} - 2\eta \underbrace{(\boldsymbol{x}_t^\top \boldsymbol{w}_t - y_t)x_{t,i}}_{\text{gradient}}$

Special cases of mirror descent (MD):

$$\boldsymbol{w}_{s+1} = f^{-1}(f(\boldsymbol{w}_s) - \eta \nabla L(\boldsymbol{w}_s))$$

with $f(\boldsymbol{w}) = \log \boldsymbol{w}$ or $f(\boldsymbol{w}) = \boldsymbol{w}$

Paradigmatic sparse linear problem

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad = \quad \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix}$$

Hadamard matrix or random $\pm$

After receiving example $(\boldsymbol{x}_t, y_t)$
and incurring loss $(\boldsymbol{x}_t^\top \boldsymbol{w}_t - y_t)^2$ update:

multiplicative, EGU: $w_{t+1,i} = w_{t,i} \ \exp(-2\eta(\boldsymbol{x}_t^\top \boldsymbol{w}_t - y_t)x_{t,i})$

additive, GD: $w_{t+1,i} = w_{t,i} \ - 2\eta \underbrace{(\boldsymbol{x}_t^\top \boldsymbol{w}_t - y_t)x_{t,i}}_{\text{gradient}}$

Special cases of mirror descent (MD):

$$\boldsymbol{w}_{s+1} = f^{-1}(f(\boldsymbol{w}_s) - \eta \nabla L(\boldsymbol{w}_s))$$

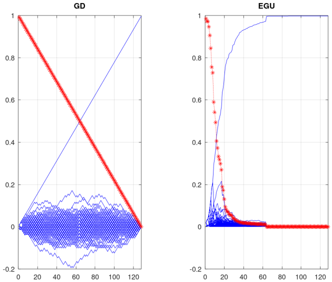
with $f(\boldsymbol{w}) = \log \boldsymbol{w}$ or $f(\boldsymbol{w}) = \boldsymbol{w}$

Major differences between the two families

Paradigmatic setup:

128x128 Hadamard matrix

Permuted rows are instances, labels are any fixed column



x-axis: $s = 1..128$

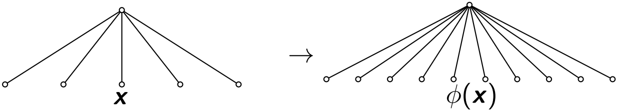
y-axis: all 128 weights Loss when trained on examples $1..s$

Upshot: After half examples, GD has average loss $1/2$

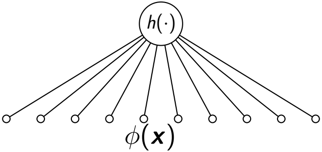
EG family converges in $\log n$ many examples

Hardness for GD Hadamard

- Linear decay of loss remains for GD even if
  - linear neuron with kernel inputs [WV05]



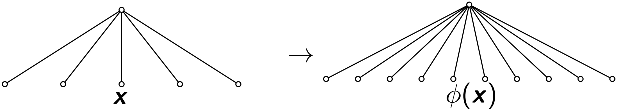
  - neuron with any transfer function h and kernel inputs [DW14]



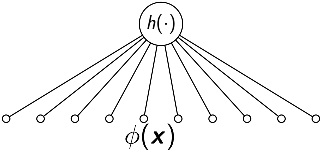
Conjecture: Hadamard problem remains hard
for any neural net trained with GD [DW14]

Hardness for GD Hadamard

- Linear decay of loss remains for GD even if
  - linear neuron with kernel inputs [WV05]



  - neuron with any transfer function h and kernel inputs [DW14]



Conjecture: Hadamard problem remains hard
for any neural net trained with GD [DW14]

Crux: consider continuous time MD [NY83]

- Parameter vector $\boldsymbol{w}(t)$ continuous function of time
- Continuous update

$$\dot{f}(\boldsymbol{w}(t)) = -\eta \nabla L(\boldsymbol{w}(t))$$

- Examples are still discrete

$$(\boldsymbol{x}_s, y_s) \text{ for time } t \in [s, s+1)$$

Again two main updates:

$$\begin{array}{rr} \text{GD} & \dot{\boldsymbol{w}}(t) = -\eta \nabla L(\boldsymbol{w}(t)) \\ \text{EGU} & \dot{\log}(\boldsymbol{w}(t)) = -\eta \nabla L(\boldsymbol{w}(t)) \end{array}$$

We motivate updates in the continuous domain
and then "discretize" these updates

Two stunning surprises

I) Continuous EGU can be simulated with continuous GD

Here: discretized versions of continuous GD simulation solves the Hadamard problem efficiently

Conjecture about GD training of neural nets is false
Neural nets trained w. GD more powerful than kernel methods

II) The structure of the network determines regularization when training with GD

Two stunning surprises

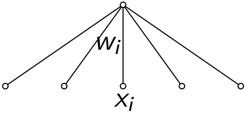
I) Continuous EGU can be simulated with continuous GD

Here: discretized versions of continuous GD simulation solves the Hadamard problem efficiently

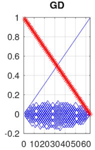
Conjecture about GD training of neural nets is false
Neural nets trained w. GD more powerful than kernel methods

II) The structure of the network determines regularization when training with GD

I) Pictorially



When linear neuron is trained with GD, then linear decrease of loss



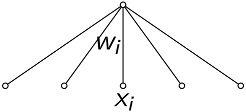
Reparameterize weights w_i by u_i^2

Continuous GD on u_i exactly simulates EGU on w_i

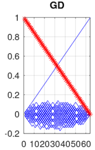
$$\dot{\boldsymbol{u}} = -2\eta\left(\boldsymbol{u}\odot\boldsymbol{u}\cdot\boldsymbol{x} - y\right)\,\boldsymbol{u}\odot\boldsymbol{x} \quad \text{simulates}$$

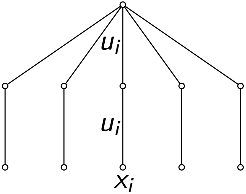
$$\dot{\log}(\boldsymbol{w}) = -2\eta\left(\boldsymbol{w}\cdot\boldsymbol{x} - y\right)\,\boldsymbol{x}$$

I) Pictorially



When linear neuron is trained with GD, then linear decrease of loss





Reparameterize weights w_i by u_i^2

Continuous GD on u_i exactly simulates EGU on w_i

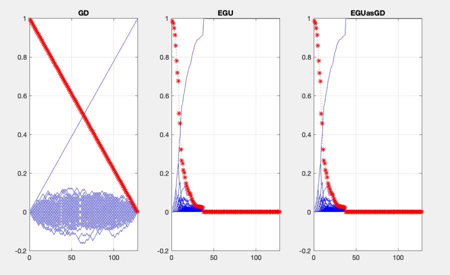
$$\dot{\boldsymbol{u}} = -2\eta\,(\boldsymbol{u}\odot\boldsymbol{u}\cdot\boldsymbol{x} - y)\;\boldsymbol{u}\odot\boldsymbol{x}\ \text{ simulates}$$

$$\dot{\log}(\boldsymbol{w}) = -2\eta\,(\boldsymbol{w}\cdot\boldsymbol{x} - y)\;\boldsymbol{x}$$

I) Simulations

Discretization $\quad \boldsymbol{u}_{t+1} = \boldsymbol{u}_t - 2\eta \left(\boldsymbol{u}_t \odot \boldsymbol{u}_t \cdot \boldsymbol{x}_t - y_t\right) \boldsymbol{u}_t \odot \boldsymbol{x}_t$ tracks

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t \odot \exp(-2\eta \left(\boldsymbol{w}_t \cdot \boldsymbol{x}_t - y_t\right) \boldsymbol{x}_t)$$



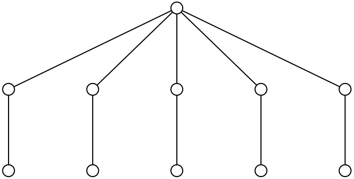
Simulation visually identical but slightly different numerically

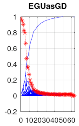
Same regret bounds

Upshot: 2-layer neural net trained w. GD cracks Hadamard

Not just a matter of initialization

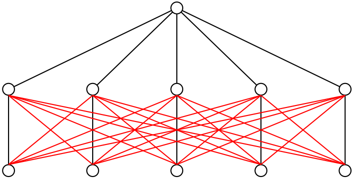
Case A

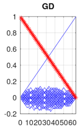




When trained with GD: approximates EGU and cracks Hadamard

Case B





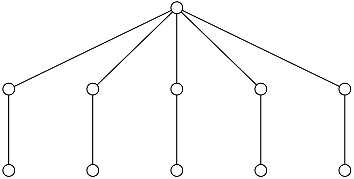
Red weights initialized to zero

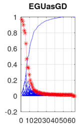
Linear loss on Hadamard when trained with GD

Also true if all bottom weights initialized to zero

Not just a matter of initialization

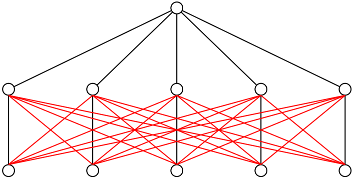
Case A

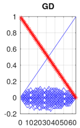




When trained with GD: approximates EGU and cracks Hadamard

Case B





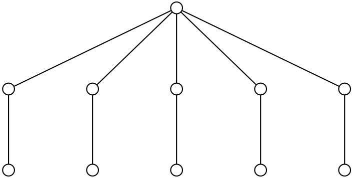
Red weights initialized to zero

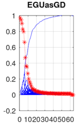
Linear loss on Hadamard when trained with GD

Also true if all bottom weights initialized to zero

II) Structure determines regularization

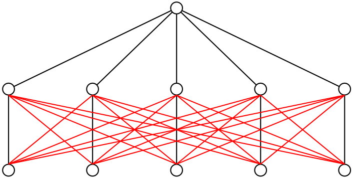
Case A

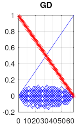




In continuous case, converges to smallest L_1 norm solution
In discrete case, same regret bounds as for EGU

Case B

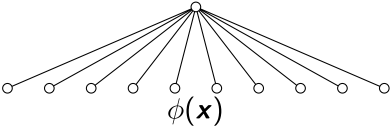




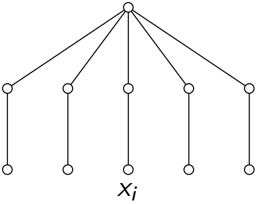
$\rightarrow$ smallest L_2 norm solution when bottom weights initialized to 0
More complicated for other initializations, but experimentally satisfies linear lower bound

2-layer linear neural net GD can beat any kernel

For Hadamard problem



Any kernel has linear decaying loss on average



EGUasGD has exponentially decaying loss

Two ways for obtaining discrete updates

1. As discretizations of continuous updates
2. Regularizing with Bregman divergences

For a strictly-convex function $F(\boldsymbol{w})$, the Bregman divergence is

$$\begin{aligned}\Delta_F(\boldsymbol{w}, \boldsymbol{w}_s) &= F(\boldsymbol{w}) - F(\boldsymbol{w}_s) - f(\boldsymbol{w}_s)^\top(\boldsymbol{w} - \boldsymbol{w}_s) \\ &= \Delta_{F^*}(\underbrace{f(\boldsymbol{w}_s)}_{\boldsymbol{w}_s^*}, \underbrace{f(\boldsymbol{w})}_{\boldsymbol{w}^*}) \qquad \text{(duality)}\end{aligned}$$

$F(\boldsymbol{w})$ convex, $\nabla F(\boldsymbol{w}) =: f(\boldsymbol{w}) = \boldsymbol{w}^*$ is the gradient

$f(\boldsymbol{w})$ called the link function

Two ways for obtaining discrete updates

1. As discretizations of continuous updates
2. Regularizing with Bregman divergences

For a strictly-convex function $F(\boldsymbol{w})$, the Bregman divergence is

$$\begin{aligned}\Delta_F(\boldsymbol{w}, \boldsymbol{w}_s) &= F(\boldsymbol{w}) - F(\boldsymbol{w}_s) - f(\boldsymbol{w}_s)^\top(\boldsymbol{w} - \boldsymbol{w}_s) \\ &= \Delta_{F^*}(\underbrace{f(\boldsymbol{w}_s)}_{\boldsymbol{w}_s^*}, \underbrace{f(\boldsymbol{w})}_{\boldsymbol{w}^*}) \qquad \text{(duality)}\end{aligned}$$

$F(\boldsymbol{w})$ convex, $\nabla F(\boldsymbol{w}) =: f(\boldsymbol{w}) = \boldsymbol{w}^*$ is the gradient

$f(\boldsymbol{w})$ called the link function

1. Via Bregman divergences [NY83,KW97]

$$\boldsymbol{w}_{s+1} = \operatorname*{argmin}_{\tilde{\boldsymbol{w}}} \;\; \Delta_F(\tilde{\boldsymbol{w}}, \boldsymbol{w}_s) + \eta L(\tilde{\boldsymbol{w}})$$

Setting derivative at $\boldsymbol{w}_{s+1}$ to zero

$$f(\boldsymbol{w}_{s+1}) - f(\boldsymbol{w}_s) + \eta \nabla L(\boldsymbol{w}_{s+1}) = \boldsymbol{0}$$

Implicit/Prox MD update [R76,NY83]

$$\boldsymbol{w}_{s+1} = f^{-1}(f(\boldsymbol{w}_s) - \eta \nabla L(\boldsymbol{w}_{s+1}))$$

Explicit MD update

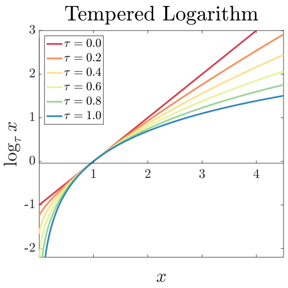
$$\boldsymbol{w}_{s+1} \approx f^{-1}(f(\boldsymbol{w}_s) - \eta \nabla L(\boldsymbol{w}_s))$$

Continuous MD [NY83]

$$\dot{f}(\boldsymbol{w}) = -\eta \nabla L(\boldsymbol{w})$$

Main examples:
GD ($f(\boldsymbol{w}) = \boldsymbol{w}$) and EGU ($f(\boldsymbol{w}) = \log(\boldsymbol{w})$)

$\log_\tau(\boldsymbol{w}) := \frac{1}{1-\tau}(\boldsymbol{w}^{1-\tau} - 1)$
τ is temperature
(we use $\tau \in [0, 1]$)



[N02]

Motivation with Bregman momentum

$$\boldsymbol{w}(t) = \operatorname*{argmin}_{\tilde{\boldsymbol{w}}(t)} \underbrace{\dot{\Delta}_F(\tilde{\boldsymbol{w}}(t), \boldsymbol{w}_s)}_{\text{Bregman momentum}} + \eta\, L(\tilde{\boldsymbol{w}}(t))$$

Derivation of the optimum curve $\boldsymbol{w}(t)$:

$$\frac{\partial}{\partial \tilde{\boldsymbol{w}}(t)}\Big(\frac{\partial}{\partial t}\Big(F(\tilde{\boldsymbol{w}}(t)) - f(\boldsymbol{w}_s)^\top \tilde{\boldsymbol{w}}(t)\Big) + \eta L(\tilde{\boldsymbol{w}}(t))\Big) \text{ (differentiate)}$$

$$= \frac{\partial}{\partial \tilde{\boldsymbol{w}}(t)}((f(\tilde{\boldsymbol{w}}(t)) - f(\boldsymbol{w}_s))^\top \dot{\tilde{\boldsymbol{w}}}(t)) + \eta\, \nabla L(\tilde{\boldsymbol{w}}(t))$$

$$= (\boldsymbol{J}f(\tilde{\boldsymbol{w}})\, \dot{\tilde{\boldsymbol{w}}}(t) + \underbrace{\Big(\frac{\partial \dot{\tilde{\boldsymbol{w}}}(t)}{\partial \tilde{\boldsymbol{w}}(t)}\Big)^\top}_{\mathbf{0}}(f(\tilde{\boldsymbol{w}}(t) - f(\boldsymbol{w}_s)) + \eta\, \nabla L(\tilde{\boldsymbol{w}}(t))$$

(By calculus of variations, $\tilde{\boldsymbol{w}}(t)$ and $\dot{\tilde{\boldsymbol{w}}}(t)$ are independent variables)

$$= \dot{f}(\tilde{\boldsymbol{w}}(t)) + \eta\, \nabla L(\tilde{\boldsymbol{w}}(t)) \overset{\tilde{\boldsymbol{w}}(t) = \boldsymbol{w}(t)}{=} \mathbf{0}$$

Reparameterizing cont. MD w. link f i.t.o. link g

Theorem For the reparameterization function $\boldsymbol{w} = q(\boldsymbol{u})$ with the property that range$(q) = $ dom(f), $\dot{g}(\boldsymbol{u}) = -\eta \nabla L \circ q\,(\boldsymbol{u})$ simulates $\dot{f}(\boldsymbol{w}) = -\eta \nabla L(\boldsymbol{w})$ if

$$(\boldsymbol{J} f(\boldsymbol{w}))^{-1} = \boldsymbol{J} q(\boldsymbol{u})\,(\boldsymbol{J} g(\boldsymbol{u}))^{-1}\,(\boldsymbol{J} q(\boldsymbol{u}))^{\top}$$

and $q(\boldsymbol{u}(0)) = \boldsymbol{w}(0)$

For reparameterization as GD use $g = id$

[details in a companion paper under review]

Our main example: EGU as GD

Link

$$f(\boldsymbol{w}) = \log(\boldsymbol{w})$$

Reparameterization

$$\begin{aligned} \boldsymbol{w} &= q(\boldsymbol{u}) := {}^1\!/\!_4\, \boldsymbol{u} \odot \boldsymbol{u} \\ \boldsymbol{u} &= 2\sqrt{\boldsymbol{w}} \end{aligned}$$

$$(\boldsymbol{J}f(\boldsymbol{w}))^{-1} = (\mathrm{diag}(\boldsymbol{w})^{-1})^{-1} = \mathrm{diag}(\boldsymbol{w})$$

$$\boldsymbol{J}q(\boldsymbol{u})(\boldsymbol{J}q(\boldsymbol{u}))^\top = {}^1\!/\!_2\, \mathrm{diag}(\boldsymbol{u})\, ({}^1\!/\!_2\, \mathrm{diag}(\boldsymbol{u}))^\top = \mathrm{diag}(\boldsymbol{w})$$

Conclusion

$$\dot{\log}(\boldsymbol{w}) = -\eta\, \nabla L(\boldsymbol{w}) \;\text{ equals }\; \dot{\boldsymbol{u}} = -\eta\, \underbrace{\nabla L \circ q\,(\boldsymbol{u})}_{\nabla_{\boldsymbol{u}} L\,({}^1\!/\!_4\, \boldsymbol{u} \odot \boldsymbol{u})} = -\eta\, {}^1\!/\!_2\, \boldsymbol{u} \odot \nabla L(\boldsymbol{w})$$

Burg as GD

Link

$$f(\boldsymbol{w}) = -\frac{1}{\boldsymbol{w}}$$

Reparameterization

$$\boldsymbol{w} = q(\boldsymbol{u}) := \exp(\boldsymbol{u})$$
$$\boldsymbol{u} = \log(\boldsymbol{w})$$

$$(\boldsymbol{J}f(\boldsymbol{w}))^{-1} = \operatorname{diag}\Big(\frac{1}{\boldsymbol{w}\odot\boldsymbol{w}}\Big)^{-1} = \operatorname{diag}(\boldsymbol{w})^2$$

$$\boldsymbol{J}q(\boldsymbol{u})(\boldsymbol{J}q(\boldsymbol{u}))^\top = \operatorname{diag}(\exp(\boldsymbol{u}))\operatorname{diag}(\exp(\boldsymbol{u}))^\top = \operatorname{diag}(\boldsymbol{w})^2$$

Conclusion

$$\dot{\left(-\frac{1}{\boldsymbol{w}}\right)} = -\eta\,\nabla L(\boldsymbol{w}) \text{ equals } \dot{\boldsymbol{u}} = -\eta\,\underbrace{\nabla L\circ q(\boldsymbol{u})}_{\nabla_{\boldsymbol{u}}L(\exp(\boldsymbol{u}))} = -\eta\,\exp(\boldsymbol{u})\odot\nabla L(\boldsymbol{w})$$

$\log_\tau \boldsymbol{w} = \frac{1}{1-\tau}(\boldsymbol{w}^{1-\tau} - 1)$ as GD

Link $\quad f(\boldsymbol{w}) = \log_\tau \boldsymbol{w}$

Reparameterization

$$\boldsymbol{w} = q(\boldsymbol{u}) := \left(\frac{2-\tau}{2}\right)^{\frac{2}{2-\tau}} \boldsymbol{u}^{\frac{2}{2-\tau}}$$

$$\boldsymbol{u} = \frac{2}{2-\tau}\boldsymbol{w}^{\frac{2-\tau}{2}}$$

$$(\boldsymbol{J}\log_\tau(\boldsymbol{w}))^{-1} = (\mathrm{diag}(\boldsymbol{w})^{-\tau})^{-1} = \mathrm{diag}(\boldsymbol{w})^{\tau}$$

$$\boldsymbol{J}q(\boldsymbol{u})(\boldsymbol{J}q(\boldsymbol{u}))^\top = \left(\left(\frac{2-\tau}{2}\right)^{\frac{\tau}{2-\tau}} \mathrm{diag}(\boldsymbol{u})^{\frac{\tau}{2-\tau}}\right)^2 = \mathrm{diag}(\boldsymbol{w})^{\tau}$$

Conclusion

$$\dot{\log}_\tau(\boldsymbol{w}) = -\eta\nabla L(\boldsymbol{w}) \text{ equals } \dot{\boldsymbol{u}} = -\eta\underbrace{\nabla L\circ q\,(\boldsymbol{u})}_{\nabla_{\boldsymbol{u}} L\left(\left(\frac{2-\tau}{2}\right)^{\frac{2}{2-\tau}}\boldsymbol{u}^{\frac{2}{2-\tau}}\right)} = -\eta\frac{2-\tau}{2}\boldsymbol{u}^{\frac{\tau}{2-\tau}}\odot\nabla L(\boldsymbol{w})$$

$\tau = 1$: EGU $\quad \tau = 0$: GD

$\log_\tau \boldsymbol{w} = \frac{1}{1-\tau}(\boldsymbol{w}^{1-\tau} - 1)$ as GD

Link $\quad f(\boldsymbol{w}) = \log_\tau \boldsymbol{w}$

Reparameterization

$$\boldsymbol{w} = q(\boldsymbol{u}) := \left(\frac{2-\tau}{2}\right)^{\frac{2}{2-\tau}} \boldsymbol{u}^{\frac{2}{2-\tau}}$$

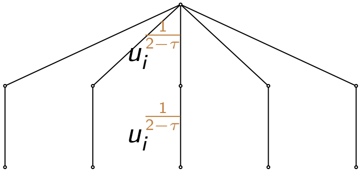
$$\boldsymbol{u} = \frac{2}{2-\tau} \boldsymbol{w}^{\frac{2-\tau}{2}}$$

$$(\boldsymbol{J}\log_\tau(\boldsymbol{w}))^{-1} = (\mathrm{diag}(\boldsymbol{w})^{-\tau})^{-1} = \mathrm{diag}(\boldsymbol{w})^{\tau}$$

$$\boldsymbol{J}q(\boldsymbol{u})(\boldsymbol{J}q(\boldsymbol{u}))^\top = \left(\left(\frac{2-\tau}{2}\right)^{\frac{\tau}{2-\tau}} \mathrm{diag}(\boldsymbol{u})^{\frac{\tau}{2-\tau}}\right)^2 = \mathrm{diag}(\boldsymbol{w})^{\tau}$$

Conclusion

$$\dot{\log_\tau}(\boldsymbol{w}) = -\eta \nabla L(\boldsymbol{w}) \text{ equals } \dot{\boldsymbol{u}} = -\eta \underbrace{\nabla L \circ q\,(\boldsymbol{u})}_{\nabla_{\boldsymbol{u}} L\left(\left(\frac{2-\tau}{2}\right)^{\frac{2}{2-\tau}} \boldsymbol{u}^{\frac{2}{2-\tau}}\right)} = -\eta \frac{2-\tau}{2} \boldsymbol{u}^{\frac{\tau}{2-\tau}} \odot \nabla L(\boldsymbol{w})$$



$\tau = 1$: EGU $\quad \tau = 0$: GD

Discrete multiplicative updates for dot loss $\sum_i w_i \ell_i$

EGU	$\tilde{w}_i = w_i \exp(-\eta \ell_i)$
Approx. EGU/PRODU	$\tilde{w}_i = w_i(1 - \eta \ell_i)$
EGUasGD	$\tilde{u}_i = u_i(1 - \eta \ell_i)$
	$(\tilde{u}_i^2 = u_i^2 (1 - \eta \ell_i)^2)$
EG/HEDGE	$\tilde{w}_i = \frac{w_i \exp(-\eta \ell_i)}{\sum_j w_j \exp(-\eta \ell_j)}$
Approx. EG	$\tilde{w}_i = w_i(1 - \eta \ell_i + \eta \sum_j w_j \ell_j)$
PROD	$\tilde{w}_i = \frac{w_i(1 - \eta \ell_i)}{\sum_j w_j (1 - \eta \ell_j)}$
EGasGD	$\tilde{u}_i = \frac{u_i(1 - \eta \ell_i)}{\lVert \sum_j u_j^2 (1 - \eta \ell_j)^2 \rVert_2^2}$
	$\left(\tilde{u}_i^2 = \frac{u_i^2 (1 - \eta \ell_i)^2}{\sum_j u_j^2 (1 - \eta \ell_j)^2}\right)$

Regret bounds

total online loss of update
$\leq$ total online loss of best comparator + norms $\sqrt{\text{loss of best}}$

update	regret bound
EGUasGD, hinge loss	as Winnow
EGUasGD, linear regression	as EGU but only one-sided case
EGasGD, linear regression	as EG
EGasGD, dot loss	as Hedge

All proofs done with relative entropy as a measure of progress

Open problems

- Need 2-sided regret bound for
  linear regression EGU and EGUasGD
  Or prove linear lower bound when
  Hadamard is replaced by 0/1 matrix
- Is there any natural problem in which GD beats EGU$^{\pm}$?
- Revisit vanishing gradient issue, batch normalization, dropout, learning rate heuristics for neural nets where all linear activations are replace by the sparse network
- Large scale simulations
  - Do multiplicative updates lead to sparse solutions?
- Revised open question about limited power of GD:
  Does any GD trained neural net with complete input neurons satisfy the linear lower bound for the Hadamard problem?

Thank you!

Open problems

- Need 2-sided regret bound for
  linear regression EGU and EGUasGD
  Or prove linear lower bound when
  Hadamard is replaced by 0/1 matrix
- Is there any natural problem in which GD beats EGU$^{\pm}$?
- Revisit vanishing gradient issue, batch normalization, dropout, learning rate heuristics for neural nets where all linear activations are replace by the sparse network
- Large scale simulations
  - Do multiplicative updates lead to sparse solutions?
- Revised open question about limited power of GD:
  Does any GD trained neural net with complete input neurons satisfy the linear lower bound for the Hadamard problem?

Thank you!