An Introduction to Adaptive Online Learning

Tim van Erven

Universiteit Leiden

Joint work with: Wouter Koolen, Peter Grünwald

ABN AMRO
October 18, 2018

Example: Sequential Prediction for Football Games

Precursor to modern football in China,
Han Dynasty (206 BC – 220 AD)

- Before every match t in the English Premier League, my PhD student Dirk van der Hoeven wants to predict the goal difference Y_t
- Given feature vector $\boldsymbol{X}_t \in \mathbb{R}^d$, he may predict $\hat{Y}_t = \boldsymbol{w}_t^\intercal \boldsymbol{X}_t$ with a linear model
- After the match: observe Y_t
- Measure loss by $\ell_t(\boldsymbol{w}_t) = (Y_t - \hat{Y}_t)^2$ and improve parameter estimates: $\boldsymbol{w}_t \to \boldsymbol{w}_{t+1}$

Example: Sequential Prediction for Football Games

Precursor to modern football in China, Han Dynasty (206 BC – 220 AD)

- Before every match t in the English Premier League, my PhD student Dirk van der Hoeven wants to predict the goal difference Y_t
- Given feature vector $\boldsymbol{X}_t \in \mathbb{R}^d$, he may predict $\hat{Y}_t = \boldsymbol{w}_t^\intercal \boldsymbol{X}_t$ with a linear model
- After the match: observe Y_t
- Measure loss by $\ell_t(\boldsymbol{w}_t) = (Y_t - \hat{Y}_t)^2$ and improve parameter estimates: $\boldsymbol{w}_t \to \boldsymbol{w}_{t+1}$

Goal: Predict almost as well as the best possible parameters $\boldsymbol{u}$:

$$\text{Regret}_T^{\boldsymbol{u}} = \sum_{t=1}^T \ell_t(\boldsymbol{w}_t) - \sum_{t=1}^T \ell_t(\boldsymbol{u})$$

General Framework: Online Convex Optimization

```
1: for $t = 1, 2, \ldots, T$ do
2:     Learner estimates $\boldsymbol{w}_t$ from convex $\mathcal{U} \subset \mathbb{R}^d$
3:     Nature reveals convex loss function $\ell_t : \mathcal{U} \to \mathbb{R}$
4:     Learner incurs loss $\ell_t(\boldsymbol{w}_t)$
5: end for
```

Goal: Predict almost as well as the best possible parameters $\boldsymbol{u}$:

$$\text{Regret}_T^{\boldsymbol{u}} = \sum_{t=1}^{T} \ell_t(\boldsymbol{w}_t) - \sum_{t=1}^{T} \ell_t(\boldsymbol{u})$$

Learner tries to **minimize** regret

Nature tries to **maximize** regret

Online Learning Example: Electricity Forecasting

Every day t an electricity company needs to predict how much electricity Y_t is needed the next day [Devaine et al., 2013]

Approach:

- Given side-information (day lengths, temperature, wind, cloud cover, ...)
- $d = 24$ different prediction models $\hat{Y}_t^1, \dots \hat{Y}_t^d$ constructed by different teams in the company
- Want to learn best combination of predictions: $\hat{Y}_t = w_{t,1}\hat{Y}_t^1 + \ldots + w_{t,d}\hat{Y}_t^d$

Online Learning Example: Electricity Forecasting

Every day t an electricity company needs to predict how much electricity Y_t is needed the next day [Devaine et al., 2013]

Approach:

- Given side-information (day lengths, temperature, wind, cloud cover, ...)
- $d = 24$ different prediction models $\hat{Y}_t^1, \ldots \hat{Y}_t^d$ constructed by different teams in the company
- Want to learn best combination of predictions: $\hat{Y}_t = w_{t,1}\hat{Y}_t^1 + \ldots + w_{t,d}\hat{Y}_t^d$

Online Learning Formulation:

For $t = 1, 2, \ldots, T$:

- Learner chooses $\boldsymbol{w}_t = (w_{t,1}, \ldots, w_{t,d})$
- Nature chooses loss function $\ell_t(w_1, \ldots, w_d) = (Y_t - w_1\hat{Y}_t^1 - \ldots - w_d\hat{Y}_t^d)^2$
- Learner's loss is $\ell_t(\boldsymbol{w}_t)$

Software

High-quality Open Source Software:

- Vowpal Wabbit (Yahoo, Microsoft):
  `https://github.com/VowpalWabbit/vowpal_wabbit/wiki`
- Built-in in standard software to train deep neural networks (TensorFlow (Google), PyTorch, etc.)

Example: Web Spam Detection

- 24 GB of data: **350 000** websites, **16 600 000** trigram features $\boldsymbol{x}$ per website
- Goal: classify website as regular ($y = +1$) or fraudulent ($y = -1$)
- Logistic loss: $f_t(\boldsymbol{w}) = \log(1 + e^{-y_t \boldsymbol{w}^\intercal \boldsymbol{x}_t})$ on t-th website
- Vowpal Wabbit:
  - Training: 5 passes over 270 000 websites in **4m11s**
  - Accuracy: 0.5% error on test set with 80 000 websites
  - Default algorithm: online gradient descent + bells and whistles

Standard Methods

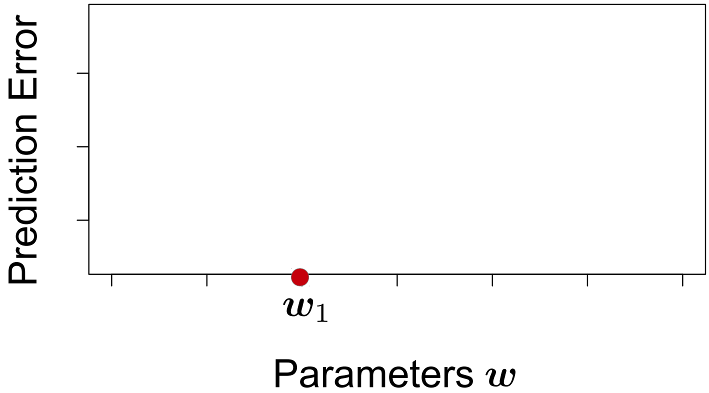
Methods: Efficient computations using only gradient $\boldsymbol{g}_t = \nabla \ell_t(\boldsymbol{w}_t)$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \boldsymbol{g}_t \qquad \text{(online gradient descent)}$$
$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \Sigma_{t+1} \boldsymbol{g}_t \qquad \text{(online Newton Step)}$$

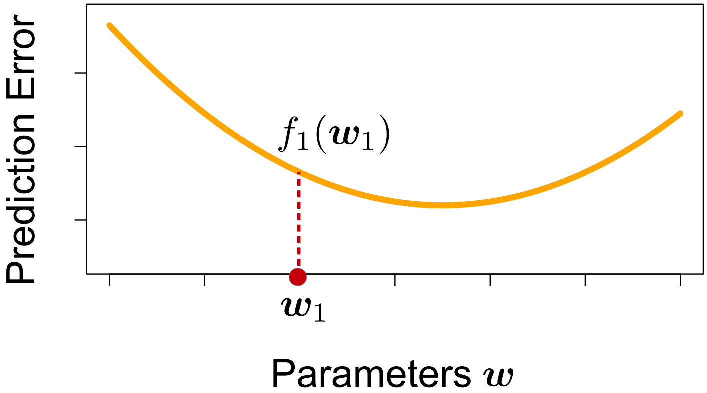
where $\Sigma_{t+1} = (\epsilon \boldsymbol{I} + 2\eta^2 \sum_{s=1}^{t} \boldsymbol{g}_s \boldsymbol{g}_s^\intercal)^{-1}$.

- Big obstacle (in theory and practice): how to tune η?

Day 0

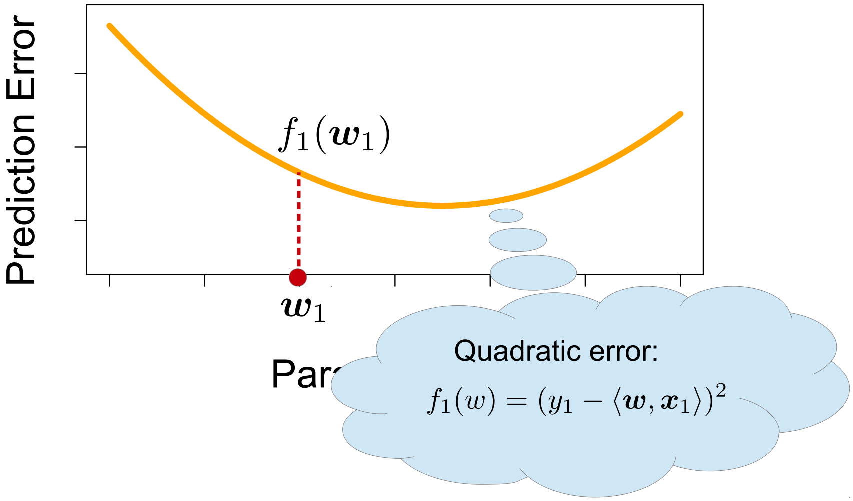


Day 1



Prediction Error
$f_1(\boldsymbol{w}_1)$
$\boldsymbol{w}_1$
Parameters $\boldsymbol{w}$

Day 1

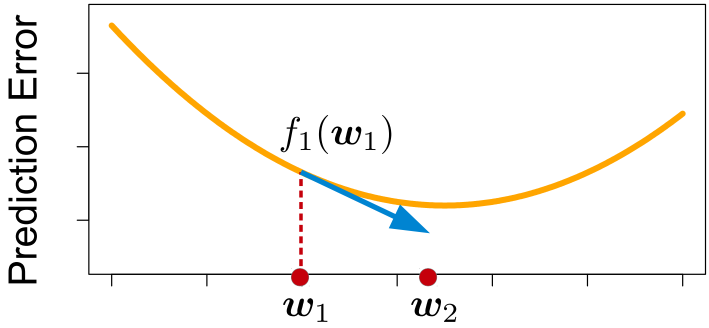


Quadratic error:

$$f_1(w) = (y_1 - \langle \boldsymbol{w}, \boldsymbol{x}_1 \rangle)^2$$

Online Gradient Descent

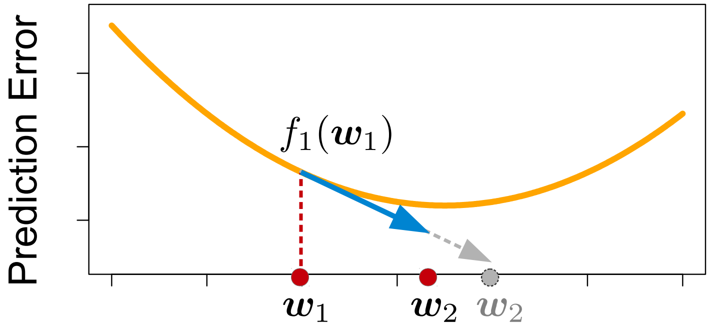
Day 1



Move in **direction** of steepest descent

Online Gradient Descent

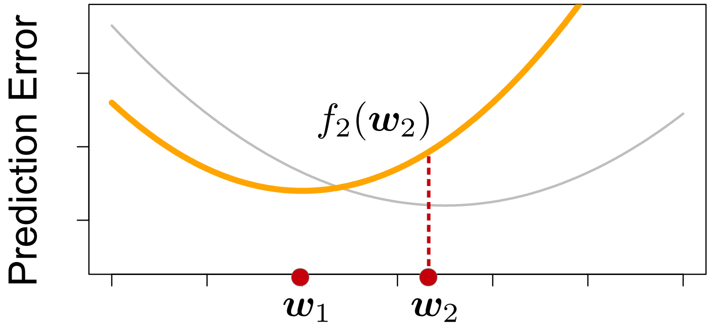
Day 1



Step size determined by **learning rate** η

Online Gradient Descent

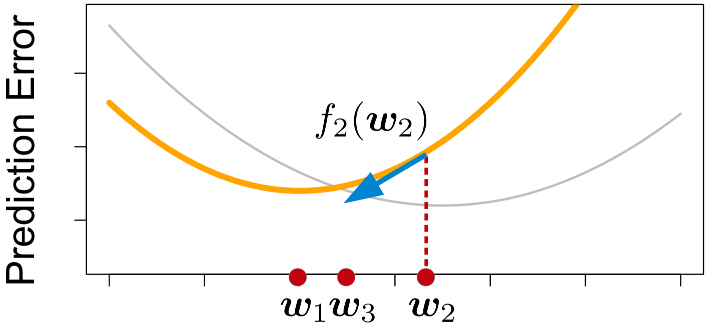
Day 2



Step size determined by **learning rate** η

Online Gradient Descent

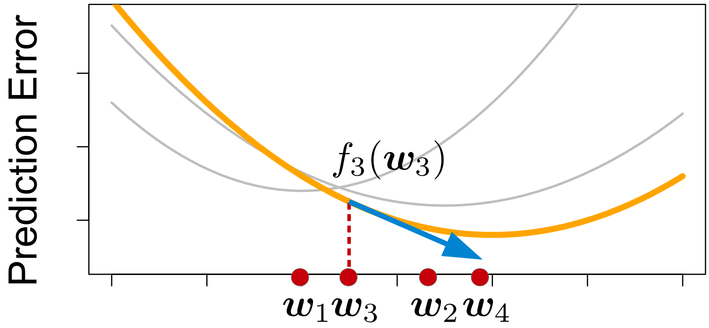
Day 2



Step size determined by **learning rate** η

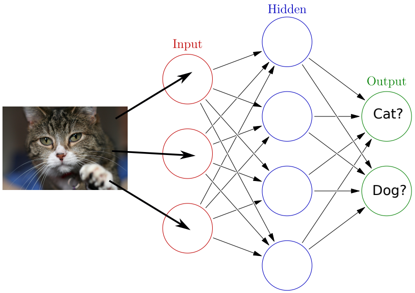
Online Gradient Descent

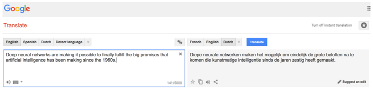
Day 3



Step size determined by **learning rate** η

Example: Deep Neural Networks





Deep neural networks are making it possible to finally fulfill the big promises that artificial intelligence has been making since the 1960s.

Diepe neurale netwerken maken het mogelijk om eindelijk de grote beloften na te komen die kunstmatige intelligentie sinds de jaren zestig heeft gemaakt.

Machine translation

Speech recognition

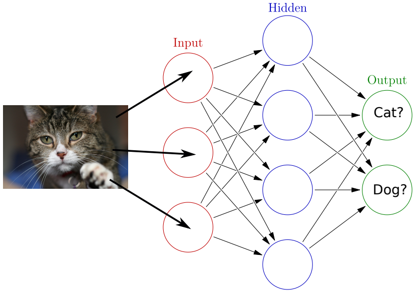
Self-driving cars

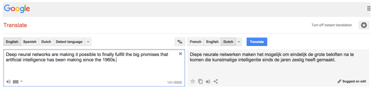
Class of **non-convex** functions parametrized by matrices $\boldsymbol{w} = (A_1, \ldots, A_m)$:

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = A_m \sigma_{m-1} A_{m-1} \cdots \sigma_1 A_1 \boldsymbol{x},$$

where $\sigma_i(z) = \max\{0, z\}$ applied component-wise to vectors.

Example: Deep Neural Networks





Deep neural networks are making it possible to finally fulfill the big promises that artificial intelligence has been making since the 1960s.

Diepe neurale netwerken maken het mogelijk om eindelijk de grote beloften na te komen die kunstmatige intelligentie sinds de jaren zestig heeft gemaakt.

Machine translation

Speech recognition

Self-driving cars

Trained by learning parameters online (non-convex task):

- Millions of images: too many to process all at once
- Process one image at a time using online learning algorithms:
  - Online gradient descent (OGD)
  - AdaGrad = OGD with separate η_t per dimension
  - Adam = AdaGrad + extensions for deep learning

Mathematical Theory

Guaranteed Bounds on the Regret (bounded domain and gradients) [Hazan, 2016]:

Convex ℓ_t	$\sqrt{T}$	OGD with $\eta_t \propto \frac{1}{\sqrt{t}}$
Strongly convex ℓ_t	$\ln T$	OGD with $\eta_t \propto \frac{1}{t}$
Exp-concave ℓ_t	$d \ln T$	ONS with $\eta \propto 1$

- **Strongly convex:** second derivative at least $\alpha > 0$, implies exp-concave
- **Exp-concave:** $e^{-\alpha\,\ell_t}$ concave
  Satisfied by log loss, logistic loss, squared loss, but not hinge loss

Mathematical Theory

Guaranteed Bounds on the Regret (bounded domain and gradients) **[Hazan, 2016]:**

Convex ℓ_t	$\sqrt{T}$	OGD with $\eta_t \propto \frac{1}{\sqrt{t}}$
Strongly convex ℓ_t	$\ln T$	OGD with $\eta_t \propto \frac{1}{t}$
Exp-concave ℓ_t	$d \ln T$	ONS with $\eta \propto 1$

Limitations:

- Different method in each case. (Requires sophisticated users.)
- Theoretical tuning of η_t **very conservative**
- What if curvature varies between rounds?
- In many applications data are **stochastic** (i.i.d.) Should be easier than worst case...

Mathematical Theory

Guaranteed Bounds on the Regret (bounded domain and gradients) **[Hazan, 2016]:**

Convex ℓ_t	$\sqrt{T}$	OGD with $\eta_t \propto \frac{1}{\sqrt{t}}$
Strongly convex ℓ_t	$\ln T$	OGD with $\eta_t \propto \frac{1}{t}$
Exp-concave ℓ_t	$d \ln T$	ONS with $\eta \propto 1$

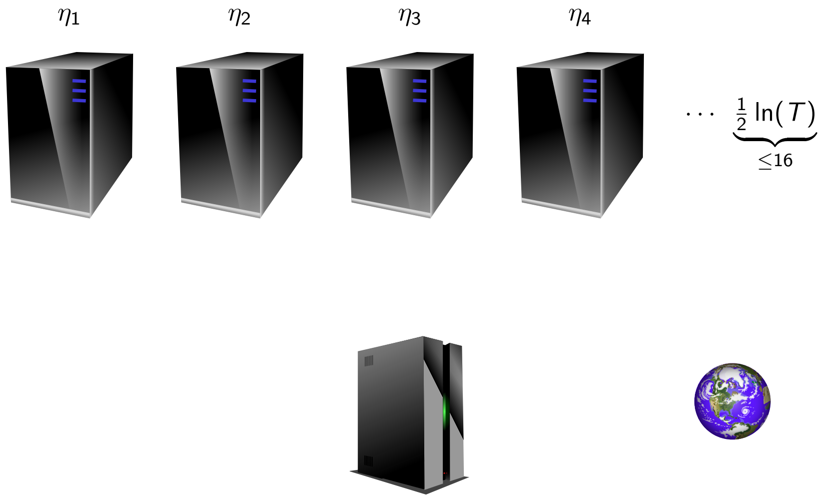
Limitations:

- Different method in each case. (Requires sophisticated users.)
- Theoretical tuning of η_t **very conservative**
- What if curvature varies between rounds?
- In many applications data are **stochastic** (i.i.d.) Should be easier than worst case. . .

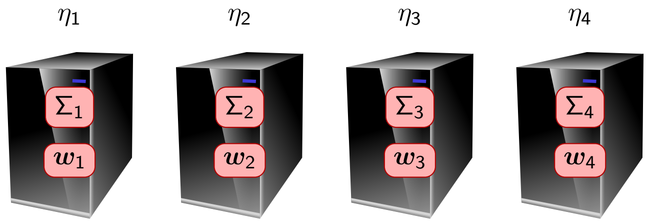
Need Adaptive Methods!

- Difficulty: All existing methods learn η at too slow rate [HP2005] so **overhead of learning best η ruins potential benefits**

MetaGrad: Multiple Eta Gradient Algorithm

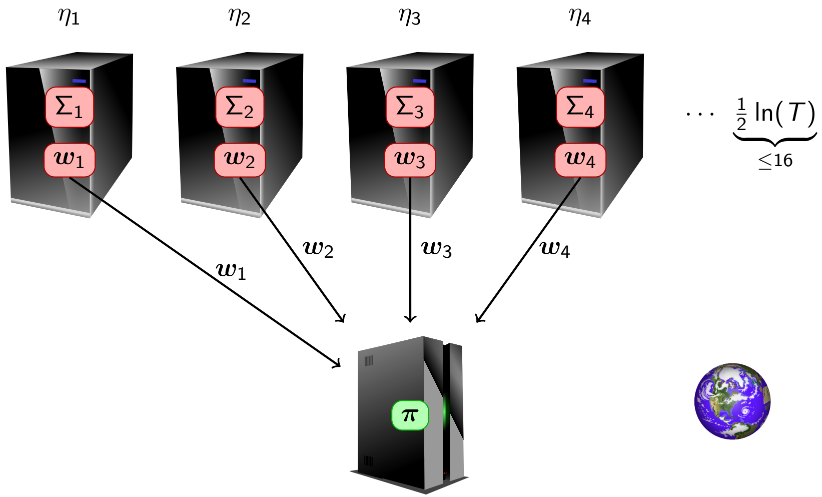


MetaGrad: Multiple Eta Gradient Algorithm

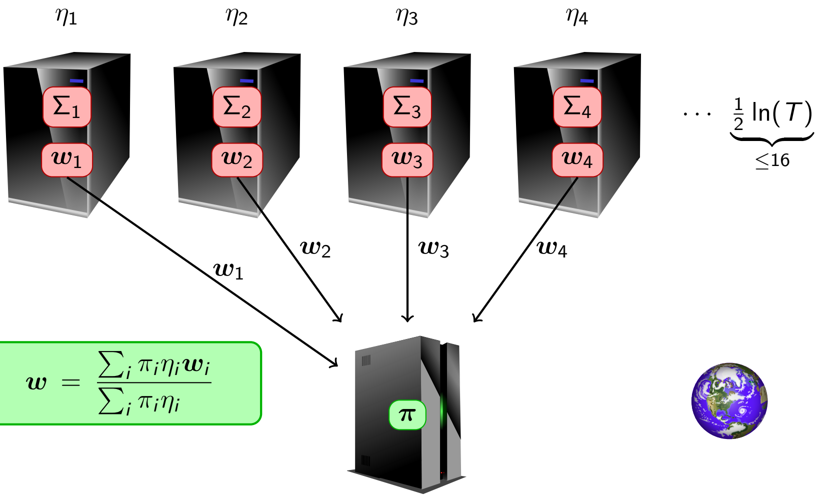


$\cdots \quad \underbrace{\tfrac{1}{2}\ln(T)}_{\leq 16}$

MetaGrad: Multiple Eta Gradient Algorithm

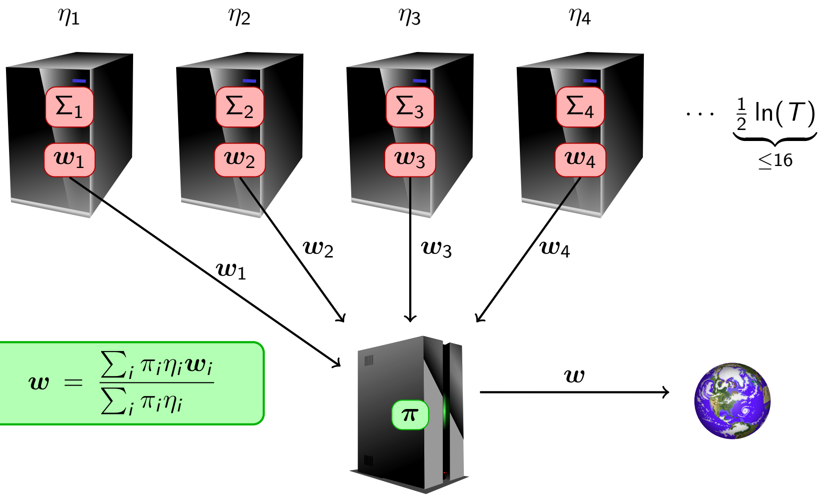


MetaGrad: Multiple Eta Gradient Algorithm



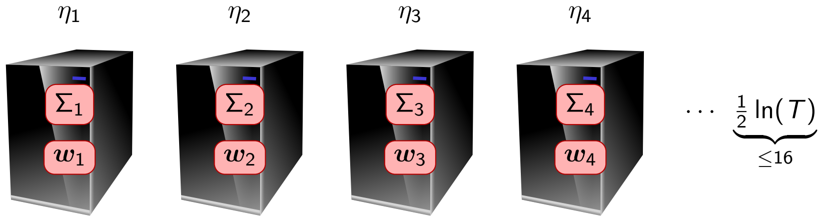
$$\boxed{\boldsymbol{w} = \frac{\sum_i \pi_i \eta_i \boldsymbol{w}_i}{\sum_i \pi_i \eta_i}}$$

MetaGrad: Multiple Eta Gradient Algorithm

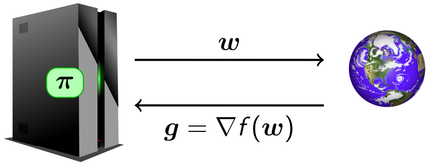


$$\boldsymbol{w} = \frac{\sum_i \pi_i \eta_i \boldsymbol{w}_i}{\sum_i \pi_i \eta_i}$$

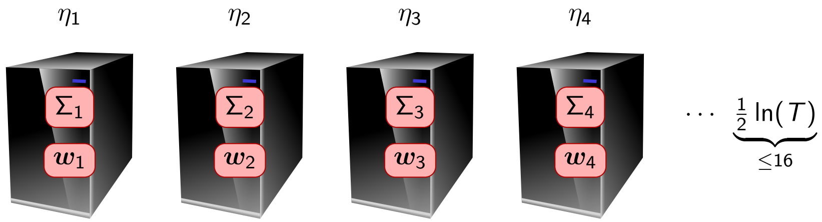
MetaGrad: Multiple Eta Gradient Algorithm



$$\boxed{\boldsymbol{w} = \frac{\sum_i \pi_i \eta_i \boldsymbol{w}_i}{\sum_i \pi_i \eta_i}}$$



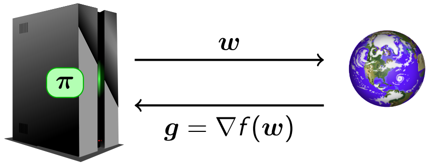
MetaGrad: Multiple Eta Gradient Algorithm



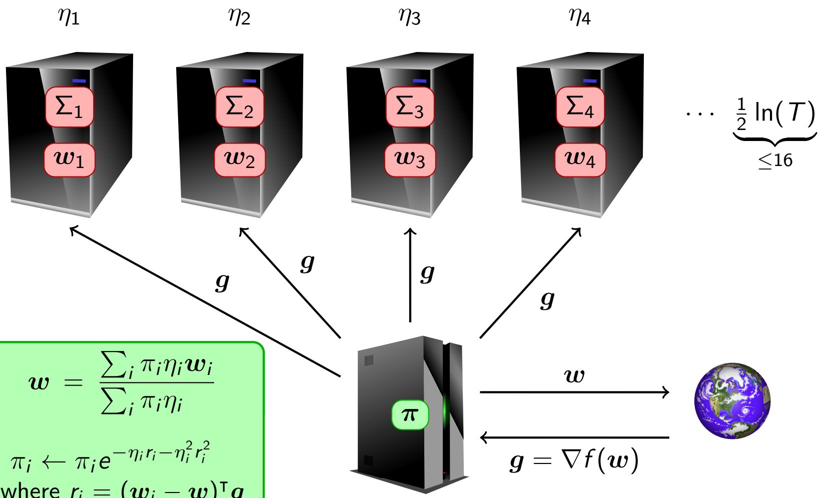
$$\boldsymbol{w} = \frac{\sum_i \pi_i \eta_i \boldsymbol{w}_i}{\sum_i \pi_i \eta_i}$$

$$\pi_i \leftarrow \pi_i e^{-\eta_i r_i - \eta_i^2 r_i^2}$$
where $r_i = (\boldsymbol{w}_i - \boldsymbol{w})^\intercal \boldsymbol{g}$

Tilted Exponential Weights



MetaGrad: Multiple Eta Gradient Algorithm



$$\boldsymbol{w} = \frac{\sum_i \pi_i \eta_i \boldsymbol{w}_i}{\sum_i \pi_i \eta_i}$$

$$\pi_i \leftarrow \pi_i e^{-\eta_i r_i - \eta_i^2 r_i^2}$$

where $r_i = (\boldsymbol{w}_i - \boldsymbol{w})^\intercal \boldsymbol{g}$

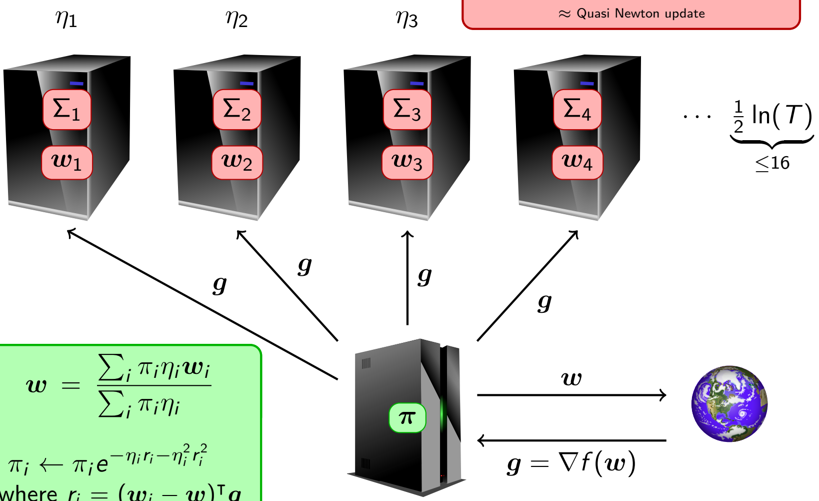
Tilted Exponential Weights

MetaGrad: Multiple Eta G

$$\Sigma_i \leftarrow (\Sigma_i^{-1} + 2\eta_i^2 \boldsymbol{g}\boldsymbol{g}^\intercal)^{-1}$$

$$\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i - \eta_i \Sigma_i \boldsymbol{g}(1 + 2\eta_i r_i)$$

$\approx$ Quasi Newton update



η_1
η_2
η_3
Σ_1 $\boldsymbol{w}_1$
Σ_2 $\boldsymbol{w}_2$
Σ_3 $\boldsymbol{w}_3$
Σ_4 $\boldsymbol{w}_4$
$\cdots \ \underbrace{\tfrac{1}{2}\ln(T)}_{\leq 16}$
$\boldsymbol{g}$
$\boldsymbol{g}$
$\boldsymbol{g}$
$\boldsymbol{g}$
π
$\boldsymbol{w}$
$\boldsymbol{g} = \nabla f(\boldsymbol{w})$

$$\boldsymbol{w} = \frac{\sum_i \pi_i \eta_i \boldsymbol{w}_i}{\sum_i \pi_i \eta_i}$$

$$\pi_i \leftarrow \pi_i e^{-\eta_i r_i - \eta_i^2 r_i^2}$$

where $r_i = (\boldsymbol{w}_i - \boldsymbol{w})^\intercal \boldsymbol{g}$

Tilted Exponential Weights

MetaGrad: Provable Adaptive Fast Rates

Theorem (Van Erven, Koolen, 2016)

MetaGrad's $\text{Regret}_T^{\boldsymbol{u}}$ *is bounded by*

$$\text{Regret}_T^{\boldsymbol{u}} \leq \sum_{t=1}^{T} (\boldsymbol{w}_t - \boldsymbol{u})^\intercal \boldsymbol{g}_t \preccurlyeq \begin{cases} \sqrt{T \ln \ln T} \\ \\ \sqrt{V_T^{\boldsymbol{u}}\, d \ln T} + d \ln T \end{cases}$$

where

$$V_T^{\boldsymbol{u}} = \sum_{t=1}^{T} ((\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t)^2.$$

- By convexity, $\ell_t(\boldsymbol{w}_t) - \ell_t(\boldsymbol{u}) \leq (\boldsymbol{w}_t - \boldsymbol{u})^\intercal \boldsymbol{g}_t$.
- Optimal learning rate η depends on $V_T^{\boldsymbol{u}}$, but $\boldsymbol{u}$ unknown!
  Crucial to learn best learning rate from data!

Consequences

1. Non-stochastic adaptation:

Convex ℓ_t	$\sqrt{T \ln \ln T}$
Exp-concave ℓ_t	$d \ln T$
Fixed convex $\ell_t = \ell$	$d \ln T$

Consequences

1. Non-stochastic adaptation:

Convex ℓ_t	$\sqrt{T \ln \ln T}$
Exp-concave ℓ_t	$d \ln T$
Fixed convex $\ell_t = \ell$	$d \ln T$

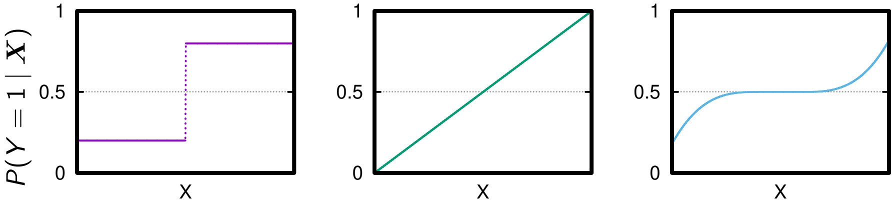
2. Stochastic without curvature

Suppose ℓ_t i.i.d. with stochastic optimum $\boldsymbol{u}^* = \arg\min_{\boldsymbol{u} \in \mathcal{U}} \mathbb{E}_\ell[\ell(\boldsymbol{u})]$.
Then expected regret $\mathbb{E}[\text{Regret}_T^{\boldsymbol{u}^*}]$:

Absolute loss* $\ell_t(w) = \lvert w - X_t \rvert$	$\ln T$
Hinge loss $\max\{0, 1 - Y_t \langle \boldsymbol{w}, \boldsymbol{X}_t \rangle\}$	$d \ln T$
(B, β)-**Bernstein**	$(Bd \ln T)^{1/(2-\beta)}\ T^{(1-\beta)/(2-\beta)}$

*Conditions apply

Related Work: Adaptivity to Stochastic Data in Batch Classification [Tsybakov, 2004]

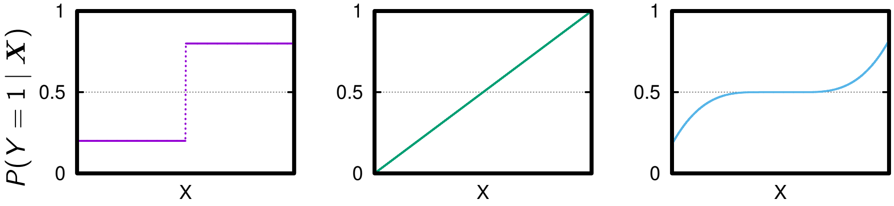


easy
$\beta = 1$

moderate
$\beta = \frac{1}{2}$

hard
$\beta = 0$

Related Work: Adaptivity to Stochastic Data in Batch Classification [Tsybakov, 2004]



easy $\beta = 1$ | moderate $\beta = \frac{1}{2}$ | hard $\beta = 0$

Definition ((B, β)-Bernstein Condition)

Losses are i.i.d. and

$$\mathbb{E}\left(\ell(\boldsymbol{w}) - \ell(\boldsymbol{u}^*)\right)^2 \le B\Big(\mathbb{E}\left[\ell(\boldsymbol{w}) - \ell(\boldsymbol{u}^*)\right]\Big)^{\beta} \qquad \text{for all } \boldsymbol{w},$$

where $\boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} \mathbb{E}[\ell(\boldsymbol{u})]$ minimizes the expected loss.

Bernstein Condition for Online Learning

Suppose ℓ_t i.i.d. with stochastic optimum $\boldsymbol{u}^* = \arg\min_{\boldsymbol{u} \in \mathcal{U}} \mathbb{E}_{\ell}[\ell(\boldsymbol{u})]$.

Standard Bernstein condition:

$$\mathbb{E}\left(\ell(\boldsymbol{w}) - \ell(\boldsymbol{u}^*)\right)^2 \leq B\Big(\mathbb{E}\left[\ell(\boldsymbol{w}) - \ell(\boldsymbol{u}^*)\right]\Big)^{\beta} \qquad \text{for all } \boldsymbol{w} \in \mathcal{U}.$$

Bernstein Condition for Online Learning

Suppose ℓ_t i.i.d. with stochastic optimum $\boldsymbol{u}^* = \arg\min_{\boldsymbol{u} \in \mathcal{U}} \mathbb{E}_{\ell}[\ell(\boldsymbol{u})]$.

Standard Bernstein condition:

$$\mathbb{E}\left(\ell(\boldsymbol{w}) - \ell(\boldsymbol{u}^*)\right)^2 \leq B\Big(\mathbb{E}\left[\ell(\boldsymbol{w}) - \ell(\boldsymbol{u}^*)\right]\Big)^{\beta} \qquad \text{for all } \boldsymbol{w} \in \mathcal{U}.$$

Replace by **weaker linearized version:**

- Apply with $\tilde{\ell}(\boldsymbol{u}) = \langle \boldsymbol{u}, \nabla \ell(\boldsymbol{w}) \rangle$ instead of ℓ!
- By convexity, $\ell(\boldsymbol{w}) - \ell(\boldsymbol{u}^*) \leq \tilde{\ell}(\boldsymbol{w}) - \tilde{\ell}(\boldsymbol{u}^*)$.

$$\mathbb{E}\left((\boldsymbol{w} - \boldsymbol{u}^*)^{\intercal} \nabla \ell(\boldsymbol{w})\right)^2 \leq B\Big(\mathbb{E}\left[(\boldsymbol{w} - \boldsymbol{u}^*)^{\intercal} \nabla \ell(\boldsymbol{w})\right]\Big)^{\beta} \quad \text{for all } \boldsymbol{w} \in \mathcal{U}.$$

Bernstein Condition for Online Learning

Suppose ℓ_t i.i.d. with stochastic optimum $\boldsymbol{u}^* = \underset{\boldsymbol{u}\in\mathcal{U}}{\arg\min}\, \underset{\ell}{\mathbb{E}}[\ell(\boldsymbol{u})]$.

Standard Bernstein condition:

$$\mathbb{E}\left(\ell(\boldsymbol{w}) - \ell(\boldsymbol{u}^*)\right)^2 \leq B\Big(\mathbb{E}\left[\ell(\boldsymbol{w}) - \ell(\boldsymbol{u}^*)\right]\Big)^{\beta} \qquad \text{for all } \boldsymbol{w} \in \mathcal{U}.$$

Replace by **weaker linearized version:**

- Apply with $\tilde{\ell}(\boldsymbol{u}) = \langle \boldsymbol{u}, \nabla\,\ell(\boldsymbol{w})\rangle$ instead of ℓ!
- By convexity, $\ell(\boldsymbol{w}) - \ell(\boldsymbol{u}^*) \leq \tilde{\ell}(\boldsymbol{w}) - \tilde{\ell}(\boldsymbol{u}^*)$.

$$\mathbb{E}\left((\boldsymbol{w}-\boldsymbol{u}^*)^\intercal \nabla\,\ell(\boldsymbol{w})\right)^2 \leq B\Big(\mathbb{E}\left[(\boldsymbol{w}-\boldsymbol{u}^*)^\intercal \nabla\,\ell(\boldsymbol{w})\right]\Big)^{\beta} \quad \text{for all } \boldsymbol{w} \in \mathcal{U}.$$

Hinge loss (domain, gradients bounded by 1): $\beta = 1$, $B = \frac{2\lambda_{\max}(\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^\intercal])}{\|\mathbb{E}[Y\boldsymbol{X}]\|}$

Bernstein Condition for Online Learning

Suppose ℓ_t i.i.d. with stochastic optimum $\boldsymbol{u}^* = \arg\min_{\boldsymbol{u}\in\mathcal{U}} \mathbb{E}_{\ell}[\ell(\boldsymbol{u})]$.

Standard Bernstein condition:

$$\mathbb{E}\left(\ell(\boldsymbol{w}) - \ell(\boldsymbol{u}^*)\right)^2 \leq B\Big(\mathbb{E}\left[\ell(\boldsymbol{w}) - \ell(\boldsymbol{u}^*)\right]\Big)^{\beta} \qquad \text{for all } \boldsymbol{w} \in \mathcal{U}.$$

Replace by **weaker linearized version:**

- Apply with $\tilde{\ell}(\boldsymbol{u}) = \langle \boldsymbol{u}, \nabla \ell(\boldsymbol{w})\rangle$ instead of ℓ!
- By convexity, $\ell(\boldsymbol{w}) - \ell(\boldsymbol{u}^*) \leq \tilde{\ell}(\boldsymbol{w}) - \tilde{\ell}(\boldsymbol{u}^*)$.

$$\mathbb{E}\left((\boldsymbol{w} - \boldsymbol{u}^*)^\intercal \nabla \ell(\boldsymbol{w})\right)^2 \leq B\Big(\mathbb{E}\left[(\boldsymbol{w} - \boldsymbol{u}^*)^\intercal \nabla \ell(\boldsymbol{w})\right]\Big)^{\beta} \quad \text{for all } \boldsymbol{w} \in \mathcal{U}.$$

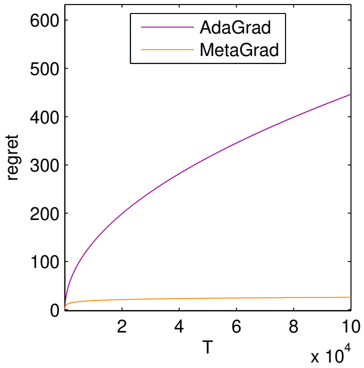
Hinge loss (domain, gradients bounded by 1): $\beta = 1$, $B = \frac{2\lambda_{\max}(\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^\intercal])}{\|\mathbb{E}[Y\boldsymbol{X}]\|}$

Theorem (Koolen, Grünwald, Van Erven, 2016)

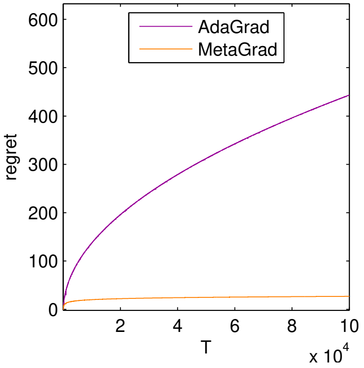
$$\mathbb{E}[\text{Regret}_T^{\boldsymbol{u}^*}] \preccurlyeq (Bd \ln T)^{1/(2-\beta)}\, T^{(1-\beta)/(2-\beta)}$$

$$\text{Regret}_T^{\boldsymbol{u}^*} \preccurlyeq (Bd \ln T - \ln\delta)^{1/(2-\beta)}\, T^{(1-\beta)/(2-\beta)} \quad \textit{w.p.} \geq 1-\delta$$

MetaGrad Simulation Experiments



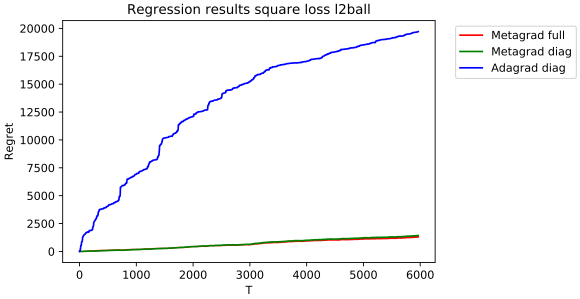
Offline: $\ell_t(u) = |u - 1/4|$



Stochastic Online: $\ell_t(u) = |u - X_t|$
where $X_t = \pm\frac{1}{2}$ i.i.d. w.p. 0.4 and 0.6.

- MetaGrad: $O(\ln T)$ regret, AdaGrad: $O(\sqrt{T})$, match bounds
- Functions neither strongly convex nor smooth
- **Caveat:** comparison more complicated for higher dimensions, unless we run a separate copy of MetaGrad per dimension, like the diagonal version of AdaGrad runs GD per dimension

MetaGrad Football Experiments



Dirk van der Hoeven
(my PhD student)

Raphaël Deswarte
(visiting PhD student)

- Predict difference in goals in 6000 football games in English Premier League (Aug 2000–May 2017).
- Square loss on Euclidean ball
- 37 features: running average of goals, shots on goal, shots over $m = 1, \dots, 10$ previous games; multiple ELO-like models; intercept.

Summary

Online Learning:

- Very fast algorithms that process one data point at a time
- Useful for:
  - Time-series data: football games, electricity forecasting, ...
  - Big data: web spam detection, deep neural networks, ...
- Big challenge: how to automatically adapt to learn optimally on different types of data?

MetaGrad Adaptive Online Learning:

- Consider **multiple learning rates η** simultaneously
- Learn η from the data, at very fast rate (pay only $\ln \ln T$)
- New adaptive variance bound that applies fast learning in **all known cases** and **new cases with stochastic data**

References

- T. van Erven and W. M. Koolen. **Metagrad: Multiple learning rates in online learning.** In Advances in Neural Information Processing Systems 29 (NIPS), pages 3666–3674, 2016.
- W. M. Koolen, P. Grünwald, and T. van Erven. **Combining adversarial guarantees and stochastic fast rates in online learning.** In Advances in Neural Information Processing Systems 29 (NIPS), pages 4457–4465, 2016.

N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

M. Devaine, P. Gaillard, Y. Goude, and G. Stoltz. Forecasting electricity consumption by aggregating specialized experts; a review of the sequential aggregation of specialized experts, with an application to Slovakian and French country-wide one-day-ahead (half-)hourly predictions. *Machine Learning*, 90(2):231–260, 2013.

E. Hazan. Introduction to online optimization. Draft, April 10, 2016, available from `ocobook.cs.princeton.edu`, 2016.

M. Hutter and J. Poland. Adaptive online prediction by following the perturbed leader. *Journal of Machine Learning Research*, 6:639–660, 2005.

A. B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *The Annals of Statistics*, 32(1):135–166, 2004.