MetaGrad
Multiple Learning Rates in Online Learning

http://bitbucket.org/wmkoolen/metagrad

Tim van Erven
Wouter M. Koolen

Universiteit Leiden
Centrum Wiskunde & Informatica

NIPS, Barcelona
Tuesday 6\textsuperscript{th} December, 2016
In a Nutshell

MetaGrad optimisation alg.

Worst case

Stochastic data

Curvature

......
Optimisation Pervasive in Machine Learning

\[
\min_w \sum_{t=1}^{T} f_t(w)
\]
Optimisation Pervasive in Machine Learning

\[
\min_w \sum_{t=1}^{T} f_t(w)
\]

Batch Training (classification)
Optimisation Pervasive in Machine Learning

$$\min_w \sum_{t=1}^{T} f_t(w)$$

- Batch Training (classification)
- Time Series (investment)
Optimisation Pervasive in Machine Learning

\[
\min_w \sum_{t=1}^{T} f_t(w)
\]

- Batch Training (classification)
- Time Series (investment)
- Big Data
Online Convex Optimisation
Online Convex Optimisation

\[ f_1(w_1), \nabla f_1(w_1) \]

\[ f_2(w_2), \nabla f_2(w_2) \]

\[ f_1 w_1 f_2 w_2 \]
Online Convex Optimisation

\[ f_1(w_1), \nabla f_1(w_1) \]

\[ f_2(w_2), \nabla f_2(w_2) \]

...
Online Convex Optimisation

\[ f_1(w_1), \nabla f_1(w_1) \]
Online Convex Optimisation

\[ f_1(w_1), \nabla f_1(w_1) \]

\[ f_2(w_2), \nabla f_2(w_2) \]
Online Convex Optimisation

\[ f_1(w_1), \nabla f_1(w_1) \]

\[ f_2(w_2), \nabla f_2(w_2) \]
Online Convex Optimisation

\[ f_1(w_1), \nabla f_1(w_1) \]

\[ f_2(w_2), \nabla f_2(w_2) \]
Online Convex Optimisation

\[ f_1(w_1), \nabla f_1(w_1) \]

\[ f_2(w_2), \nabla f_2(w_2) \]

\[ w_1 \]

\[ w_2 \]
**Definition (Regret)**

\[
R_T = \sum_{t=1}^{T} f_t(w_t) - \min_u \sum_{t=1}^{T} f_t(u)
\]

- Online loss
- Optimal loss
Online Gradient Descent [Zinkevich, 2003]

\[ \mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla f_t(\mathbf{w}_t) \]
Online Gradient Descent [Zinkevich, 2003]

\[ \mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla f_t(\mathbf{w}_t) \]

Worst-case regret guarantee:

\[ R_T = O \left( \sqrt{T} \right) \]
Online Gradient Descent [Zinkevich, 2003]

\[ \mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla f_t(\mathbf{w}_t) \]

Worst-case regret guarantee:

\[ R_T = O\left(\sqrt{T}\right) \]
Loss Taxonomy \sim Curvature

\[ \sqrt{T} \]

Worst-case regret

Convex
linear, hinge
absolute

Exp-concave
logistic
squared

Strongly convex
squared distance

\[ d \ln T \]
\[(w \in \mathbb{R}^d)\]

\[ \ln T \]

Online Gradient Descent \[\text{(Zinkevich, 2003)}\]

Online Newton Step \[\text{(Hazan et al., 2007)}\]
Loss Taxonomy $\sim$ Curvature

- **Convex**: linear, hinge, absolute
- **Exp-concave**: logistic, squared
- **Strongly convex**: squared distance

Worst-case regret

$d \ln T$ ($w \in \mathbb{R}^d$)

Online Gradient Descent [Zinkevich, 2003]

Online Newton Step [Hazan et al., 2007]
Loss Taxonomy \sim \text{Curvature}

- Convex: linear, hinge, absolute
- Exp-concave: logistic, squared
- Strongly convex: squared distance

Worst-case regret:
- $\sqrt{T}$
- $d \ln T$ ($w \in \mathbb{R}^d$)
- $\ln T$

Online Gradient Descent [Zinkevich, 2003]
Online Newton Step [Hazan et al., 2007]
Loss Taxonomy \sim \text{Curvature}

- **Convex**
  - Linear, hinge
  - Absolute

- **Exp-concave**
  - Logistic
  - Squared

- **Strongly convex**
  - Squared distance

Worst-case regret:
- $\sqrt{T}$
- $d \ln T$ ($w \in \mathbb{R}^d$)
- $\ln T$
Big Questions

Can we make adaptive methods for online convex optimisation that are

- worst-case safe
- exploit curvature automatically
- computationally efficient
Big Questions

Can we make **adaptive** methods for **online convex optimisation** that are

- **worst-case safe**
- exploit **curvature** automatically
- computationally **efficient**

And can we adapt to other **important regimes**?

- **Mixed** or **in-between** cases?
- **Stochastic** data? Bandits [Seldin and Slivkins, 2014]
- Absence of **curvature**? Experts [Koolen and Van Erven, 2015]
Main Idea

For every optimisation algorithm tuning is **crucial**.
Main Idea

For every optimisation algorithm tuning is **crucial**.

So let’s **learn optimal tuning from data**.
Main Idea

For every optimisation algorithm tuning is **crucial**.

So let’s **learn optimal tuning** from **data**.

Key obstacle: avoid learning $\eta$ at **slow rate** itself.
Main Idea

For every optimisation algorithm tuning is crucial.

So let’s learn optimal tuning from data.

Key obstacle: avoid learning $\eta$ at slow rate itself.

Breakthrough: Multiple Eta Gradient algorithm (MetaGrad)
MetaGrad Algorithm

\[ \eta_1, \eta_2, \eta_3, \eta_4, \ldots, \ln(T) \leq 16 \]
MetaGrad Algorithm

\[ \eta_1, \eta_2, \eta_3, \eta_4, \ldots \ln(T) \leq 16 \]

\[ \sum_i \pi_i \eta_i w_i = \sum_i \pi_i \sum_i \pi_i \eta_i w_i g \]

Tilted Exponential Weights

\[ \pi_i \leftarrow \pi_i e^{-\eta_i r_i - \eta_2 r_2} \]

where

\[ r_i = (w_i - w_{i-1})' g \]

\[ \sum_i \left( \sum_i \pi_i \eta_i w_i g (1 + 2 \eta_i r_i) \right) \approx \text{Online Newton Step} \]
MetaGrad Algorithm

\[ \eta_1, \eta_2, \eta_3, \eta_4, \ldots \leq 16 \]

\[ \ln(T) \]

where

\[ r_i = (w_i - w) \]

\[ g \]

\[ \Sigma_i \rightarrow (\Sigma_i^{-1} + 2\eta_i g g^\top)^{-1} \]

\[ w_i \rightarrow w_i - \eta_i \Sigma_i g (1 + 2\eta_i r_i) \]

\[ \pi \]

\[ \pi_i \leftarrow \pi_i e^{-\eta_i r_i - \eta_i^2 r_i^2} \]
MetaGrad Algorithm

\[ w = \frac{\sum_i \pi_i \eta_i w_i}{\sum_i \pi_i \eta_i} \]

\[ \ln(T) \leq 16 \]
MetaGrad Algorithm

\[ w = \frac{\sum_i \pi_i \eta_i w_i}{\sum_i \pi_i \eta_i} \]

\[ \ln(T) \leq 16 \]
MetaGrad Algorithm

\[ w = \frac{\sum_i \pi_i \eta_i w_i}{\sum_i \pi_i \eta_i} \]

\[ g = \nabla f(w) \]

\[ \ln(T) \leq 16 \]
MetaGrad Algorithm

\[ \eta_1 \]
\[ \Sigma_1 \]
\[ w_1 \]

\[ \eta_2 \]
\[ \Sigma_2 \]
\[ w_2 \]

\[ \eta_3 \]
\[ \Sigma_3 \]
\[ w_3 \]

\[ \eta_4 \]
\[ \Sigma_4 \]
\[ w_4 \]

\[ \cdots \]
\[ \ln(T) \leq 16 \]

\[ w = \frac{\sum_i \pi_i \eta_i w_i}{\sum_i \pi_i \eta_i} \]

\[ \pi_i \leftarrow \pi_i e^{-\eta_i r_i - \eta_i^2 r_i^2} \]

where \( r_i = (w_i - w)^\top g \)

Tilted Exponential Weights

\[ g = \nabla f(w) \]
MetaGrad Algorithm

\[ \eta_1 \eta_2 \eta_3 \eta_4 \ldots \ln(T) \leq 16 \]

\[ \sum_i \pi_i \eta_i w_i \]

\[ \pi_i \leftarrow \pi_i e^{-\eta_i r_i - \eta_i^2 r_i^2} \]

where \( r_i = (w_i - w) \mathbf{T} g \)

Tilted Exponential Weights
MetaGrad Algorithm

\[ \eta_1, \eta_2, \eta_3, \eta_4, \ldots \]

\[ \ln(T) \leq 16 \]

\[ \Sigma_i \leftarrow (\Sigma_i^{-1} + 2\eta_i^2 g g^T)^{-1} \]

\[ w_i \leftarrow w_i - \eta_i \Sigma_i g (1 + 2\eta_i r_i) \]

\[ \approx \text{Online Newton Step} \]

\[ w = \frac{\sum_i \pi_i \eta_i w_i}{\sum_i \pi_i \eta_i} \]

\[ \pi_i \leftarrow \pi_i e^{-\eta_i r_i - \eta_i^2 r_i^2} \]

where \( r_i = (w_i - w)^\top g \)

Tilted Exponential Weights
Second-order Regret Bound

The regret of MetaGrad is bounded by

$$ R_T = O \left( \min \left\{ \sqrt{T}, \sqrt{V_T d \ln T} \right\} \right), $$

where

$$ V_T = \sum_{t=1}^{T} \left( (w_t - u^*)^T \nabla f_t(w_t) \right)^2 $$

measures variance compared to the offline optimum

$$ u^* = \arg \min_u \sum_{t=1}^{T} f_t(u) $$

Note: Optimal tuning depends on unknown optimum $u^*$. 
MetaGrad Adapts to Curvature

MetaGrad regret bound:

\[ R_T = O \left( \sqrt{V_T d \ln T} \right) \]

Corollary

For \( \alpha \)-exp-concave or \( \alpha \)-strongly convex losses, MetaGrad ensures

\[ R_T = O (d \ln T) \]

without knowing \( \alpha \).
MetaGrad Adapts to Curvature

MetaGrad regret bound:

$$R_T = O\left(\sqrt{V_T d \ln T}\right)$$

Corollary

For $\alpha$-exp-concave or $\alpha$-strongly convex losses, MetaGrad ensures

$$R_T = O\left(d \ln T\right)$$

without knowing $\alpha$.

Same result for fixed $f_t = f$ (classical optimisation) even without curvature via derivative condition.
MetaGrad Adapts to Curvature

MetaGrad regret bound:

\[ R_T = O\left(\sqrt{V_T} d \ln T\right) \]

Corollary

For \( \alpha \)-exp-concave or \( \alpha \)-strongly convex losses, MetaGrad ensures

\[ R_T = O\left(d \ln T\right) \]

without knowing \( \alpha \).

Same result for fixed \( f_t = f \) (classical optimisation) even without curvature via derivative condition.

Reason

Curvature implies \( \Omega(V_T) \) cumulative slack between loss and its tangent lower bound.
MetaGrad Adapts to Stochastic Margin

Consider i.i.d. losses $f_t \sim P$ with stochastic optimum

$$u^* = \arg \min_u \mathbb{E} f(u)$$

Goal is small pseudo-regret compared to $u^*$:

$$R_T^* = \sum_{t=1}^T f_t(w_t) - \sum_{t=1}^T f_t(u^*)$$

Corollary

For any $\beta$-Bernstein $P$, MetaGrad keeps the expected regret below

$$\mathbb{E} R_T^* \leq O\left(\frac{d \ln T}{2} - \frac{1}{\beta T} - \frac{1}{\beta^2} \right).$$

Fast rates without curvature: e.g. absolute loss, hinge loss, . . .

Reason Bernstein bounds $\mathbb{E} V_T^*$ above by $\mathbb{E} R_T^*$. "Solve" regret bound.

Joint work with P. Grünwald

Come see more at poster #76
MetaGrad Adapts to Stochastic Margin

Consider i.i.d. losses $f_t \sim P$ with \textbf{stochastic optimum}

$$u^* = \arg\min_u \mathbb{E} f(u)$$

Goal is small \textbf{pseudo-regret} compared to $u^*$:

$$R_T^* = \sum_{t=1}^{T} f_t(w_t) - \sum_{t=1}^{T} f_t(u^*)$$

\textbf{Corollary}

\textit{For any $\beta$-Bernstein $P$, MetaGrad keeps the expected regret below}

$$\mathbb{E} R_T^* \leq O \left( (d \ln T)^{\frac{1}{2-\beta}} T^{\frac{1-\beta}{2-\beta}} \right).$$

\textbf{Fast rates without curvature: e.g. absolute loss, hinge loss, ...}
MetaGrad Adapts to Stochastic Margin

Consider i.i.d. losses $f_t \sim \mathbb{P}$ with stochastic optimum

$$u^* = \arg \min_u \mathbb{E} f(u)$$

Goal is small \textbf{pseudo-regret} compared to $u^*$:

$$R_T^* = \sum_{t=1}^{T} f_t(w_t) - \sum_{t=1}^{T} f_t(u^*)$$

**Corollary**

For any $\beta$-Bernstein $\mathbb{P}$, MetaGrad keeps the expected regret below

$$\mathbb{E} R_T^* \leq O \left( (d \ln T)^{\frac{1}{2-\beta}} T^{\frac{1-\beta}{2-\beta}} \right).$$

\textbf{Fast rates} without curvature: e.g. absolute loss, hinge loss, \ldots

\textbf{Reason}

Bernstein bounds $\mathbb{E}[V_T^*]$ above by $\mathbb{E}[R_T^*]$. “Solve” regret bound.
MetaGrad Adapts to Stochastic Margin

Consider i.i.d. losses $f_t \sim P$ with stochastic optimum

$$u^* = \arg\min_u \mathbb{E} f(u)$$

Goal is small pseudo-regret compared to $u^*$:

$$R_T^* = \sum_{t=1}^T f_t(w_t) - \sum_{t=1}^T f_t(u^*)$$

**Corollary**

For any $\beta$-Bernstein $P$, MetaGrad keeps the expected regret below

$$\mathbb{E} R_T^* \leq O \left( (d \ln T)^{\frac{1}{2-\beta}} \right)$$

Fast rates without curvature: e.g. absolute loss, hinge loss, ...

**Reason**

Bernstein bounds $\mathbb{E}[V_T^*]$ above by $\mathbb{E}[R_T^*]$. “Solve” regret bound.

Joint work with P. Grünwald

Come see more at poster #76
Conclusion

First contact with a new generation of adaptive algorithms.
Conclusion

First contact with a new generation of adaptive algorithms.

MetaGrad adapts to a wide range of environments:

- Stochastic data
- Curvature $d \ln T$
- Worst case $\sqrt{T}$
- $T^{\frac{1-\beta}{2-\beta}}$
- ...
Conclusion

First contact with a new generation of adaptive algorithms.

MetaGrad adapts to a wide range of environments:

Stochastic data $T$ 

Worst case $\sqrt{T}$

See you tonight at poster #187