Online Learning: Robustness and Adaptivity

Wouter M. Koolen

CWI
Centrum Wiskunde & Informatica

ELC Fall School, Shonan Village
Tuesday 20th September, 2016

In These Three Lectures

I hope to show you

- a rich class of learning problems
- algorithms and analysis
- ways to think about easy/hard problems

Part I

Introduction to Online Learning

Outline

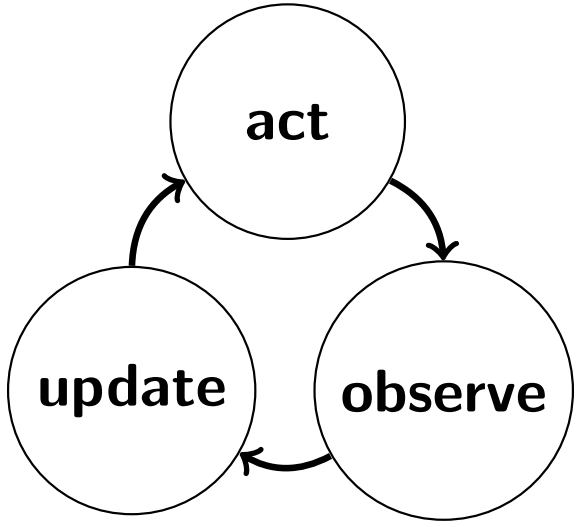
Grand Goal of Machine Learning

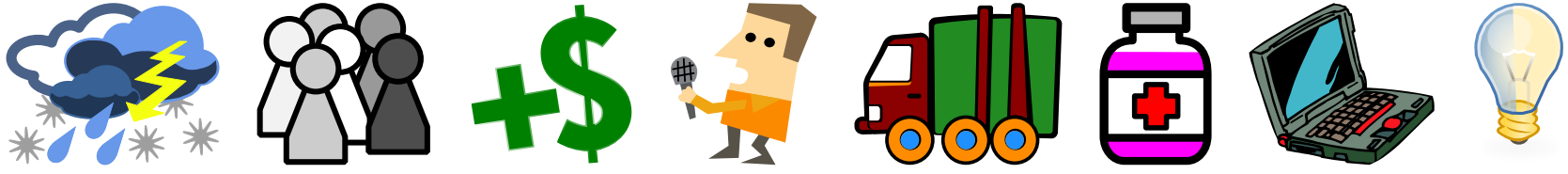
Design systems that improve performance by learning from data.

$$\text{system} + \text{data} \;=\; \text{better system}$$

- Batch learning: training $\rightarrow$ production
- Online learning: continuously improving.

Online Learning Challenges Everywhere





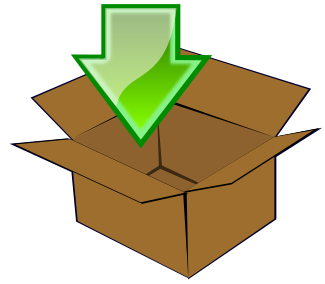
. . .

Example 1: Spam filtering

- A new email arrives
- The system puts it in **inbox** or spam folder
- User intervenes if misclassified ☹

Goal: design **learning** spam filter that improves continuously

Example 2: (Streaming) Data Compression



- The system proposes a **code**
- The next input symbol arrives
- The associated codeword is stored/transmitted (cost proportional to length)

Goal: design compressor that leans from past data to improve encoding of future data.

Example 3: Route planning

Fix a graph w. start and destination nodes.

- The system proposes a **path**
- The user drives and incurs traffic delays
- Delays are observed
  - on all road segments (full information)
  - on road segments along chosen path (semi-bandit)
  - total along chosen path (bandit)

We will look at full information.
For more on bandits see Honda (ELC Fall School 2014) or Bubeck (ALT 2011 etc.)

Formalising what it means to learn

Select a **class** of possibly helpful

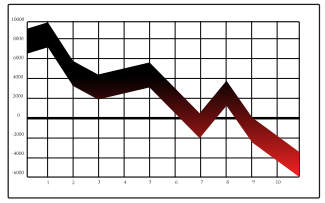
- patterns
- hypotheses
- models
- predictors
- decision making strategies
- ...

Goal: perform well if any element of the class **useful for your data**

Note: behaviouristic criterion

Outline

Stock Market Investment, Protocol



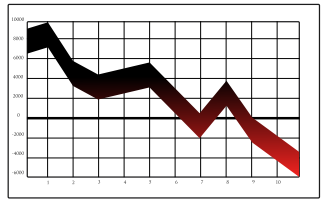
Imagine K stocks.
Investor starts with $\mathcal{K}_0 = 1¥$.
On each day $t = 1, 2, \ldots$

- Investor picks portfolio $\boldsymbol{w}_t \in \triangle_K$
  (w_t^k is the fraction of capital invested in stock k)
- Market reveals returns $\boldsymbol{x}_t \in [0, \infty)^K$
  ($1¥$ invested in stock k becomes $x_t^k ¥$)
- Investors capital evolves to $\mathcal{K}_t = \mathcal{K}_{t-1} \sum_{k=1}^K w_t^k x_t^k$

After T days, Investor's capital is

$$\mathcal{K}_T \;=\; \prod_{t=1}^{T} \left(\sum_{k=1}^{K} w_t^k x_t^k \right).$$

Stock Market Investment, Objective



After T days, Investor's capital is

$$\mathcal{K}_T \;=\; \prod_{t=1}^{T} \left(\sum_{k=1}^{K} w_t^k x_t^k \right) .$$

With prior knowledge of the best stock he could have had

$$\mathcal{K}_T^* \;=\; \max_k \mathcal{K}_T^k \qquad \text{where} \qquad \mathcal{K}_T^k \;=\; \prod_{t=1}^{T} x_t^k$$

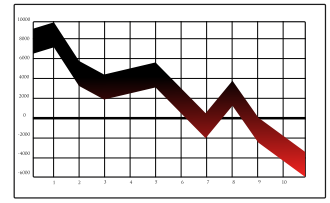
We define Investor's **regret** compared to the best stock by

$$R_T \;:=\; \ln \mathcal{K}_T^* - \ln \mathcal{K}_T$$

(logarithms make things additive)

Question: can we keep the regret small? How small?

Stock Market Investment, First Algorithm



We want to perform like the best stock.

Idea: Play best-stock-so-far (Follow the Leader):

$$\boldsymbol{w}_t \;=\; \text{uniform on } \{k \mid \mathcal{K}_{t-1}^k = \mathcal{K}_{t-1}^*\}$$

This is extremely dangerous.

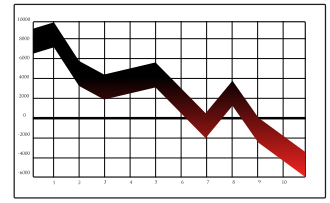
Consider two stocks, two rounds

x_t^k	round 1	round 2
stock A	1	1
stock B	2	0

Now best stock A has $\mathcal{K}_2^* = 1$ but Investor bankrupt $\mathcal{K}_2 = 0$.

$$R_2 \;=\; \ln \mathcal{K}_2^* - \ln \mathcal{K}_2 \;=\; \infty.$$

Stock Market Investment, Minimax Algorithm



We define Investor's **regret** compared to the best stock by

$$R_T \;:=\; \ln \mathcal{K}_T^* - \ln \mathcal{K}_T$$

Let's consider **minimising the regret** as a game

$$\text{minimax}\; R_T \;=\; \min_{\boldsymbol{w}_1} \max_{\boldsymbol{x}_1} \cdots \min_{\boldsymbol{w}_T} \max_{\boldsymbol{x}_T}\; R_T .$$

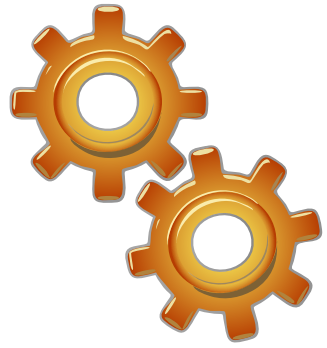
Minimax regret:

$$\text{minimax}\; R_T \;=\; \ln K$$

Minimax algorithm:

$$w_t^k \;=\; \frac{\prod_{s=1}^{t-1} x_s^k}{\sum_{j=1}^{K} \prod_{s=1}^{t-1} x_s^j}$$

Stock Market Investment, Minimax Algorithm



Minimax algorithm:

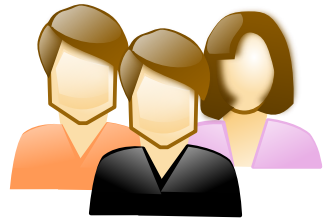
$$w_t^k = \frac{\prod_{s=1}^{t-1} x_s^k}{\sum_{j=1}^{K} \prod_{s=1}^{t-1} x_s^j} = \frac{w_{t-1}^k x_{t-1}^k}{\sum_{j=1}^{K} w_{t-1}^j x_{t-1}^j}$$

Sometimes called the **Aggregating Algorithm**.

- Efficiently implementable (K space and $O(K)$ time per round)
- Analogue of Bayes Rule (w. uniform prior)
- Anytime: same algorithm optimal for any horizon T (!)
- Generalisations ubiquitous in online learning.
- Evolution (x_t^k fitness of species k at time t)

Outline

The Hedge Setting, Protocol



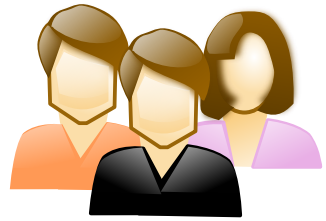
We now consider a simple additive game.
Imagine K experts.
Learner starts with loss $L_0 = 0$.
On each day $t = 1, 2, \ldots$

- Learner picks distribution $\boldsymbol{w}_t \in \triangle_K$
  (w_t^k is the chance of picking expert k)
- Environment reveals losses $\boldsymbol{\ell}_t \in [0,1]^K$
  (expert k incurs loss ℓ_t^k)
- Learner's loss evolves to $L_t = L_{t-1} + \sum_{k=1}^K w_t^k \ell_t^k$

After T days, Learner's loss is

$$L_T = \sum_{t=1}^{T} \sum_{k=1}^{K} w_t^k \ell_t^k$$

The Hedge Setting, Objective



After T days, Learner's loss is

$$L_T = \sum_{t=1}^{T} \sum_{k=1}^{K} w_t^k \ell_t^k$$

The loss of the best expert is

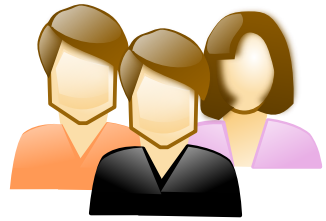
$$L_T^* = \min_k \sum_{t=1}^{T} \ell_t^k$$

and the **regret** compared to the best expert is

$$R_T = L_T - L_T^*$$

How small can we keep the regret? And how?

The Hedge Setting, Analysis



The **regret** compared to the best expert is

$$R_T \;=\; L_T - L_T^*$$

Consider minimising the regret as a game

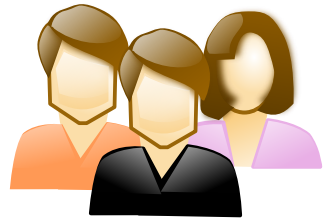
$$\text{minimax}\; R_T \;=\; \min_{\boldsymbol{w}_1} \max_{\boldsymbol{\ell}_1} \cdots \min_{\boldsymbol{w}_T} \max_{\boldsymbol{\ell}_T}\; R_T$$

Exact minimax analysis **intractable** ☹.

- Adversaries witness lower bounds
- Algorithms witness upper bounds

Try to make them match (up to constant factor, asymptotically).

Hedge Lower Bound, Probabilistic Method



Fix **any** algorithm (strategy for Learner to pick $\boldsymbol{w}_t$).
Say Adversary picks loss $\boldsymbol{\ell}_t \in \{0,1\}^K$ **uniformly at random**.

- Expected loss of any expert: $\mathbb{E}[L_T^k] = T/2$.
- Expected loss of algorithm: $\mathbb{E}[L_T] = T/2$.
- Expected loss of best expert: $\mathbb{E}[L_T^*] = T/2 - \sqrt{T/2 \ln K}$ (large K).

So the expected regret is at least

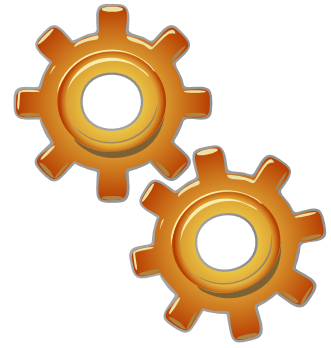
$$\mathbb{E}[R_T] \;\geq\; \sqrt{T/2 \ln K}.$$

Hence there is at least one loss sequence $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_T$ on which

$$R_T \;\geq\; \sqrt{T/2 \ln K}.$$

Note: related to Sequential Rademacher Complexity [Rakhlin et al., 2015]

Hedge Upper Bound, Algorithm



Idea: use minimax algorithm for stock market investment:

$$w_t^k \;=\; \frac{\prod_{s=1}^{t-1} x_s^k}{\sum_{j=1}^{k} \prod_{s=1}^{t-1} x_s^j}$$

Now how to supply x_t^k (non-negative, higher-is-better)? Idea:

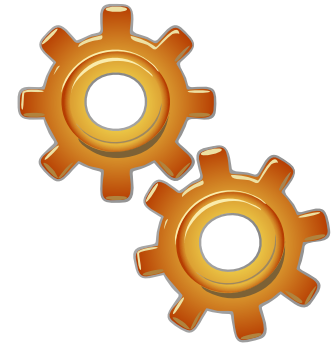
$$x_t^k \;\leftarrow\; e^{-\eta \ell_t^k} \qquad \text{(pseudo-likelihood)}$$

We arrive at

$$w_t^k \;=\; \frac{\prod_{s=1}^{t-1} e^{-\eta \ell_s^k}}{\sum_{j=1}^{k} \prod_{s=1}^{t-1} e^{-\eta \ell_s^j}} \;=\; \frac{e^{-\eta \sum_{s=1}^{t-1} \ell_s^k}}{\sum_{j=1}^{k} e^{-\eta \sum_{s=1}^{t-1} \ell_s^j}} \;=\; \frac{e^{-\eta L_{t-1}^k}}{\sum_{j=1}^{k} e^{-\eta L_{t-1}^j}}$$

Now called the **Hedge** algorithm [Freund and Schapire, 1997].

Hedge Upper Bound, Analysis



Recall AA guarantees $-\ln \mathcal{K}_T \leq -\ln \mathcal{K}_T^* + \ln K$, i.e.

$$-\ln \prod_{t=1}^{T} \sum_{k=1}^{K} w_t^k x_t^k \;\leq\; \min_k -\ln \prod_{t=1}^{T} x_t^k + \ln K.$$

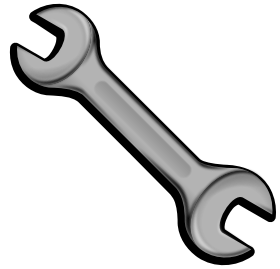
By **Hoeffding's inequality**, for all $\eta \geq 0$, $\boldsymbol{w} \in \triangle_K$ and $\boldsymbol{\ell} \in [0,1]^K$

$$\sum_{k=1}^{K} w^k \ell^k \;\leq\; -\frac{1}{\eta} \ln \sum_{k=1}^{K} w^k e^{-\eta \ell^k} + \frac{\eta}{8}$$

So

$$\begin{aligned} L_T \;=\; \sum_{t=1}^{T} \sum_{k=1}^{K} w_t^k \ell^k \;&\leq\; \frac{1}{\eta} \sum_{t=1}^{T} -\ln \sum_{k=1}^{K} w_t^k e^{-\eta \ell_t^k} + \frac{T\eta}{8} \\ &\leq\; \frac{1}{\eta} \left(\min_k \sum_{t=1}^{T} -\ln e^{-\eta \ell_t^k} + \ln K \right) + \frac{T\eta}{8} \\ &=\; L_T^* + \frac{\ln K}{\eta} + \frac{T\eta}{8} \end{aligned}$$

Hedge Upper Bound, Tuning



We derived

$$R_T \leq \frac{\ln K}{\eta} + \frac{T\eta}{8}$$

The tuning for η that optimises the bound is given by

$$\eta = \sqrt{\frac{8 \ln K}{T}}$$

and the bound becomes

$$R_T \leq \sqrt{T/2 \ln K}$$

Matching the lower bound (for large K).

Mixability

We saw two loss functions:

$$\boldsymbol{w}, \boldsymbol{x} \;\mapsto\; -\ln \sum_{k=1}^{k} w^k x^k \qquad \text{and} \qquad \boldsymbol{w}, \boldsymbol{\ell} \;\mapsto\; \sum_{k=1}^{K} w^k \ell^k$$

with associated minimax regret rates

$$\ln K \qquad \text{and} \qquad \sqrt{T/2 \ln K}$$

Q: What can we say in general?
[Vovk, 1998] shows $O(\ln K)$ rate iff loss function is mixable.

Conclusion

- Looked at two prototypical settings in detail
- One-and-a-half algorithm
- Minimax rate
- Complexity of problems (so far) property of **loss function**
  - Mixable loss are easy (fast rates).
  - Other losses are hard (slow rates).

Part II

Online Optimization and Curvature

Overview

- Continuous domains (vs finite set of experts)
- Extend methods (AA) to a wide class of applications
- Keep computation in check

Outline

Online Optimization

Domain $\mathcal{U}$.

For $t = 1, 2, \ldots$

- Learner plays action $\boldsymbol{w}_t \in \mathcal{U}$
- Environment reveals loss function $f_t : \mathcal{U} \to \mathbb{R}$
- Learner incurs loss $f_t(\boldsymbol{w}_t)$

Example

Setting	loss function $f_t(\boldsymbol{u})$
Hedge setting	$\boldsymbol{u}^\mathsf{T}\boldsymbol{\ell}_t$
Point prediction	$\|\boldsymbol{u} - \boldsymbol{x}_t\|^2$
Regression	$(\boldsymbol{u}^\mathsf{T}\boldsymbol{x}_t - y_t)^2$
Logistic regression	$-\ln\left(1 + e^{-y_t \boldsymbol{u}^\mathsf{T}\boldsymbol{x}_t}\right)$
Hinge loss	$\max\{0, 1 - y_t \boldsymbol{u}^\mathsf{T}\boldsymbol{x}_t\}$
Investment	$-\ln(\boldsymbol{u}^\mathsf{T}\boldsymbol{x}_t)$
Offline optimization	$f(\boldsymbol{u})$

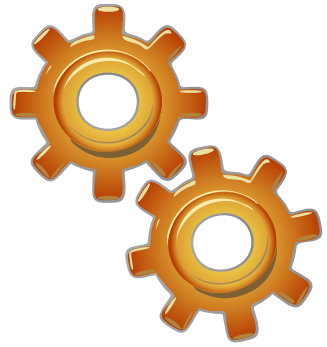
Regret

Regret after T rounds compared to action $\boldsymbol{u} \in \mathcal{U}$:

$$R_T^{\boldsymbol{u}} := \sum_{t=1}^{T} \big(f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{u}) \big).$$

Goal: small regret compared to any $\boldsymbol{u} \in \mathcal{U}$.

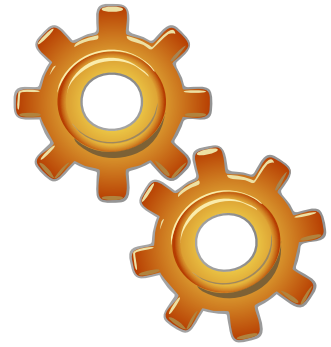
Why Change a Working System?



Q: Can We Already Solve Online Optimisation?
A: Yes. Discretise $\mathcal{U}$. Run Hedge.

Problem: Number of discretisation points **exponential** in dimension d. Typically $O(T^{d/2})$.

Why Change a Working System?



Let us upgrade the AA to continuous sets $\mathcal{U}$.

Start from a prior π on $\mathcal{U}$. Set $\pi_1 = \pi$. Each round t

- Play the mean (called **Exponentially Weighted Average**)

$$\boldsymbol{w}_t \;=\; \int_{\mathcal{U}} \boldsymbol{u}\pi_t(\boldsymbol{u})\,\mathrm{d}\boldsymbol{u}$$

- Update

$$\pi_{t+1}(\boldsymbol{u}) \;=\; \frac{\pi_t(\boldsymbol{u})e^{-\eta f_t(\boldsymbol{u})}}{\int_{\mathcal{U}} \pi_t(\boldsymbol{u})e^{-\eta f_t(\boldsymbol{u})}\,\mathrm{d}\boldsymbol{u}}$$

Problem: hard to compute $\boldsymbol{w}_t$.

- Shape of f_t.
- Shape of $\mathcal{U}$.

Surrogate Loss (computational tool)

Idea: replace loss function f_t with **simpler** function $\breve{f}_t$.

Say functions $\breve{f}_t : \mathcal{U} \to \mathbb{R}$ satisfy

$$f_t(\boldsymbol{u}) \;\geq\; \breve{f}_t(\boldsymbol{u}) \quad \text{for all } \boldsymbol{u} \in \mathcal{U} \qquad \text{(lower bound)}$$

and

$$f_t(\boldsymbol{w}_t) \;=\; \breve{f}_t(\boldsymbol{w}_t) \qquad \text{(tightness)}$$

Then

$$R_T^{\boldsymbol{u}} \;=\; \sum_{t=1}^{T} \big(f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{u})\big) \;\leq\; \sum_{t=1}^{T} \big(\breve{f}_t(\boldsymbol{w}_t) - \breve{f}_t(\boldsymbol{u})\big) \;=:\; \breve{R}_T^{\boldsymbol{u}}.$$

Ok, so sufficient to control the surrogate regret $\breve{R}_t^{\boldsymbol{u}}$.

Dealing with $\mathcal{U}$

Even with surrogate losses, maintaining π_t on $\mathcal{U}$ is too costly. The real constraint is that Learner plays $\boldsymbol{w}_t \in \mathcal{U}$.

Idea: maintain π_t on all of $\mathbb{R}^d$ and ensure mean $\boldsymbol{w}_t \in \mathcal{U}$. Bonus: surrogate loss is defined on all of $\mathbb{R}^d$.

So, first compute unconstrained update

$$\tilde{\pi}_{t+1}(\boldsymbol{u}) \;=\; \frac{\pi_t(\boldsymbol{u}) e^{-\eta \check{f}_t(\boldsymbol{u})}}{\int_{\mathbb{R}^d} \pi_t(\boldsymbol{u}) e^{-\eta \check{f}_t(\boldsymbol{u})} \,\mathrm{d}\boldsymbol{u}}$$

and then project on distributions with their mean in $\mathcal{U}$

$$\pi_{t+1} \;=\; \operatorname*{argmin}_{\pi : \mathbb{E}_\pi[\boldsymbol{u}] \in \mathcal{U}} \mathrm{KL}(\pi \| \tilde{\pi}_{t+1}) \;=\; \mathrm{project}^{\mathcal{U}}_{\mathrm{KL}}(\tilde{\pi}_{t+1})$$

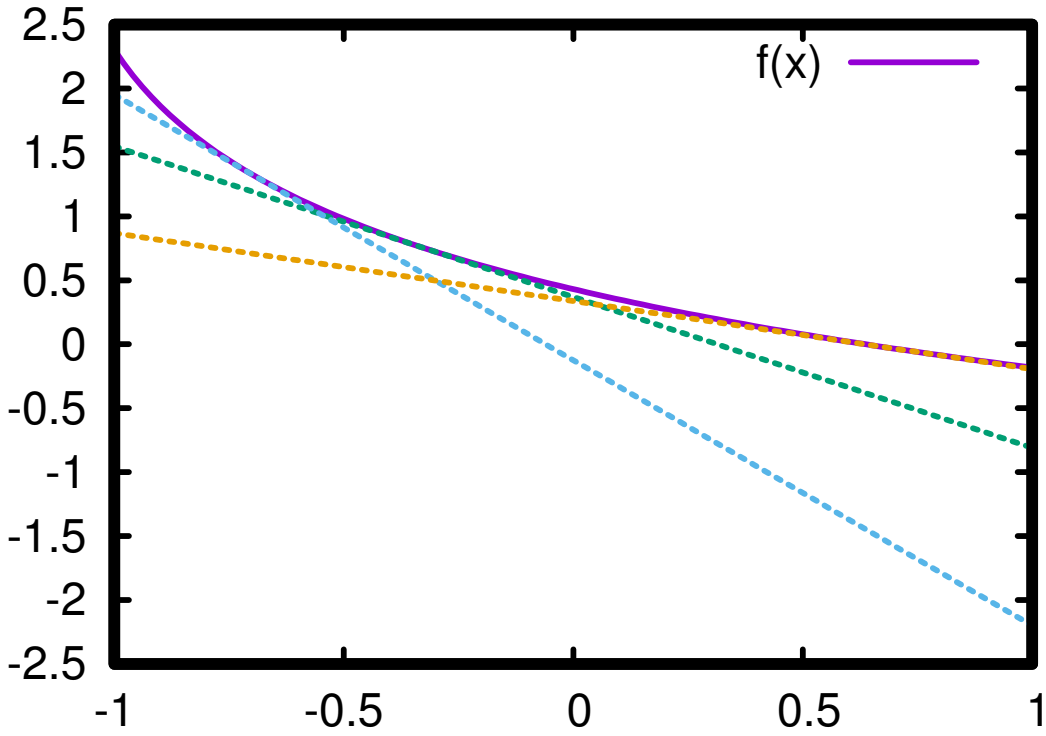
(Kullback-Leibler divergence is distance between distributions)

Outline

Convex Function

A function $f : \mathcal{U} \to \mathbb{R}$ is **convex** if

$$f(\boldsymbol{u}) \;\geq\; f(\boldsymbol{w}) + (\boldsymbol{u} - \boldsymbol{w})^{\mathsf{T}} \nabla f(\boldsymbol{w}) \qquad \text{for all } \boldsymbol{u}, \boldsymbol{w} \in \mathcal{U}.$$



Note: randomisation never smart.

Convex Function, Surrogate

Let us abbreviate

$$\boldsymbol{g}_t \;=\; \nabla f_t(\boldsymbol{w}_t)$$

So for convex functions we can take surrogate loss

$$\breve{f}_t(\boldsymbol{u}) \;\coloneqq\; f_t(\boldsymbol{w}_t) + (\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t.$$

Convex Function, Algorithm

Putting our three ingredients together:

- Gaussian prior $\mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$, i.e. $\pi(\boldsymbol{u}) \propto e^{-\frac{\|\boldsymbol{u}\|^2}{2\sigma^2}}$ (on $\mathbb{R}^d$)
- Tangent bound surrogate
- Mean-in-$\mathcal{U}$-projection

Update steps

$$\begin{aligned} \tilde{\pi}_{t+1}(\boldsymbol{u}) &\propto \pi_t(\boldsymbol{u}) e^{-\eta \breve{f}_t(\boldsymbol{u})} \\ \pi_{t+1} &= \text{project}_{\text{KL}}^{\mathcal{U}}(\tilde{\pi}_{t+1}) \end{aligned}$$

Net effect: $\tilde{\pi}_{t+1}$ and π_{t+1} are Gaussian,

$$\tilde{\pi}_{t+1} = \mathcal{N}(\tilde{\boldsymbol{w}}_t, \sigma^2 \boldsymbol{I}) \qquad \pi_{t+1} = \mathcal{N}(\boldsymbol{w}_t, \sigma^2 \boldsymbol{I})$$

with

$$\begin{aligned} \tilde{\boldsymbol{w}}_{t+1} &= \boldsymbol{w}_t - \eta \sigma^2 \boldsymbol{g}_t \\ \boldsymbol{w}_{t+1} &= \operatorname*{argmin}_{\boldsymbol{u} \in \mathcal{U}} \|\boldsymbol{u} - \tilde{\boldsymbol{w}}_{t+1}\|^2 \end{aligned}$$

This is **Online Gradient Descent** [Zinkevich, 2003].

Online Gradient Descent

Online Gradient Descent

$$\begin{aligned} \tilde{\boldsymbol{w}}_{t+1} &= \boldsymbol{w}_t - \eta\sigma^2 \boldsymbol{g}_t \\ \boldsymbol{w}_{t+1} &= \operatorname*{argmin}_{\boldsymbol{u}\in\mathcal{U}} \|\boldsymbol{u} - \tilde{\boldsymbol{w}}_{t+1}\|^2 \end{aligned}$$

- Generalisation of gradient descent for offline optimisation
- Update: $O(d)$ time per round, d memory
- Projection: depends on $\mathcal{U}$.

Online Gradient Descent Analysis

Theorem (Zinkevich 2003)

Online Gradient Descent guarantees

$$\check{R}_T^{\boldsymbol{u}} \;\leq\; \frac{1}{2\eta\sigma^2}\|\boldsymbol{u}\|^2 + \frac{\eta\sigma^2}{2}\sum_{t=1}^{T}\|\boldsymbol{g}_t\|^2.$$

Assume $\|\boldsymbol{u}\| \leq D$ for all $\boldsymbol{u} \in \mathcal{U}$ and $\|\boldsymbol{g}_t\| \leq G$ for all t. Then

$$\check{R}_T^{\boldsymbol{u}} \;\leq\; \frac{1}{2\eta\sigma^2}D^2 + \frac{\eta\sigma^2}{2}G^2\,T$$

Now we may optimise the bound by picking

$$\eta\sigma^2 \;=\; \sqrt{\frac{D^2}{G^2\,T}}$$

to find

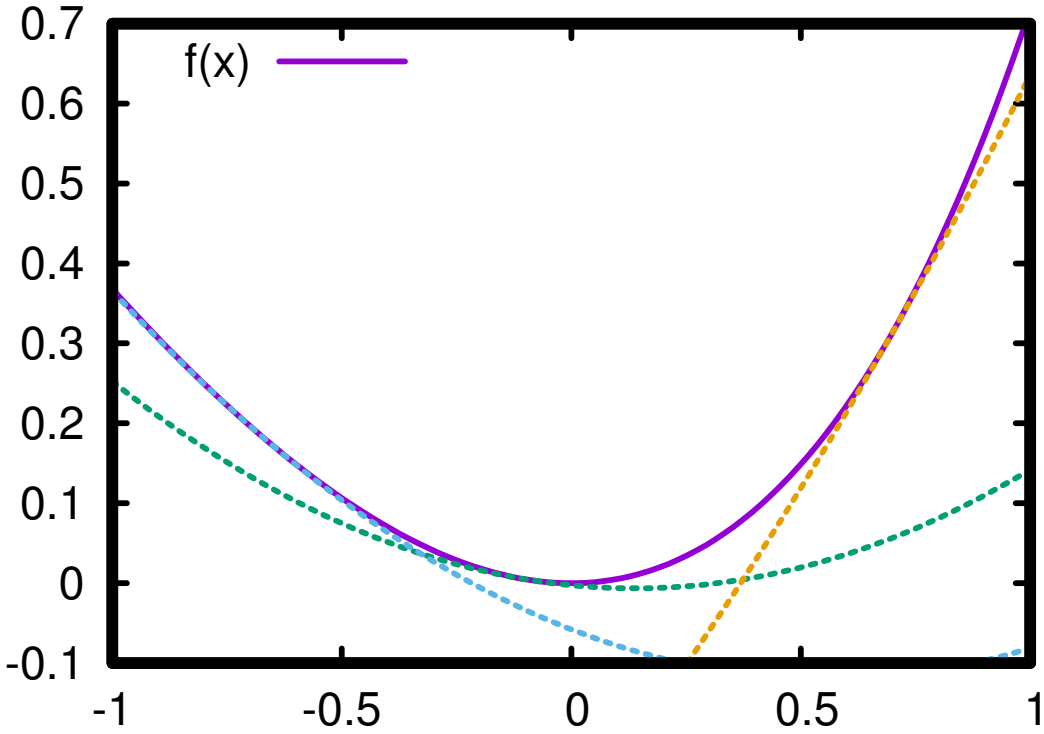
$$\check{R}_T^{\boldsymbol{u}} \;\leq\; GD\sqrt{T}.$$

Outline

Strong Convexity

A function f is β-**strongly-convex** if

$$f(\boldsymbol{u}) \geq f(\boldsymbol{w}) + (\boldsymbol{u}-\boldsymbol{w})^{\mathsf{T}} \nabla f(\boldsymbol{w}) + \frac{\beta}{2}\|\boldsymbol{u}-\boldsymbol{w}\|^2 \qquad \text{for all } \boldsymbol{u}, \boldsymbol{w} \in \mathcal{U}.$$



Canonical example: $f_t(\boldsymbol{u}) = \frac{1}{2}\|\boldsymbol{u} - \boldsymbol{x}_t\|^2$ is 1-strongly convex.

Strong Convexity, Surrogate

We take surrogate loss

$$\breve{f}_t(\boldsymbol{u}) \;:=\; f_t(\boldsymbol{w}_t) + (\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t + \frac{\beta}{2}\|\boldsymbol{u} - \boldsymbol{w}_t\|^2.$$

Q: a quadratic contribution. What effect would that have on the algorithm?

Strong Convexity, Algorithm

Starting with Gaussian $\pi = \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$, executing updates

$$\tilde{\pi}_{t+1}(\boldsymbol{u}) \;\propto\; \pi_t(\boldsymbol{u}) e^{-\eta \check{f}_t(\boldsymbol{u})} \qquad \text{and} \qquad \pi_{t+1} \;=\; \text{project}_{\text{KL}}^{\mathcal{U}}(\tilde{\pi}_{t+1})$$

we find that all distributions remain Gaussian, namely

$$\tilde{\pi}_{t+1} \;=\; \mathcal{N}(\tilde{\boldsymbol{w}}_{t+1}, \sigma_{t+1}^2 \boldsymbol{I}) \qquad \text{and} \qquad \pi_{t+1} \;=\; \mathcal{N}(\boldsymbol{w}_{t+1}, \sigma_{t+1}^2 \boldsymbol{I})$$

where

$$\begin{aligned}
\sigma_{t+1}^2 \;&=\; \frac{1}{\frac{1}{\sigma_t^2} + \eta\beta} \;=\; \cdots \;=\; \frac{1}{\frac{1}{\sigma^2} + t\eta\beta} \\
\tilde{\boldsymbol{w}}_{t+1} \;&=\; \boldsymbol{w}_t - \eta\sigma_{t+1}\boldsymbol{g}_t \\
\boldsymbol{w}_{t+1} \;&=\; \operatorname*{argmin}_{u \in \mathcal{U}} \; \|\boldsymbol{u} - \tilde{\boldsymbol{w}}_{t+1}\|^2
\end{aligned}$$

This is **Online Gradient Descent w.** $1/t$ **learning rate** [Hazan et al., 2007].

Strong Convexity, Analysis

Theorem (Hazan et al. 2007)

Let $\|\boldsymbol{u}\| \leq D$ and $\|\boldsymbol{g}_t\| \leq G$. EWA with strongly convex surrogate losses satisfies

$$R_T^{\boldsymbol{u}} \;\leq\; \frac{D^2}{2\eta\sigma^2} + \frac{G^2}{2\beta}\ln\left(1+\beta\eta\sigma^2 T\right)$$

Tuning

$$\eta\sigma^2 \;=\; \frac{\beta D^2}{G^2}$$

results in

$$R_T^{\boldsymbol{u}} \;=\; O(\ln T).$$

- Much faster rate than convex regime $O(\sqrt{T})$.
- No dependence on ambient dimension d
- Same run-time as Online Gradient Descent.

Outline

Exp-Concavity

A function f is α-**exp-concave** if $e^{-\alpha f(\boldsymbol{u})}$ is concave.

In one dimension this means

$$f''(u) \;\geq\; \alpha f'(u)^2$$

and in general

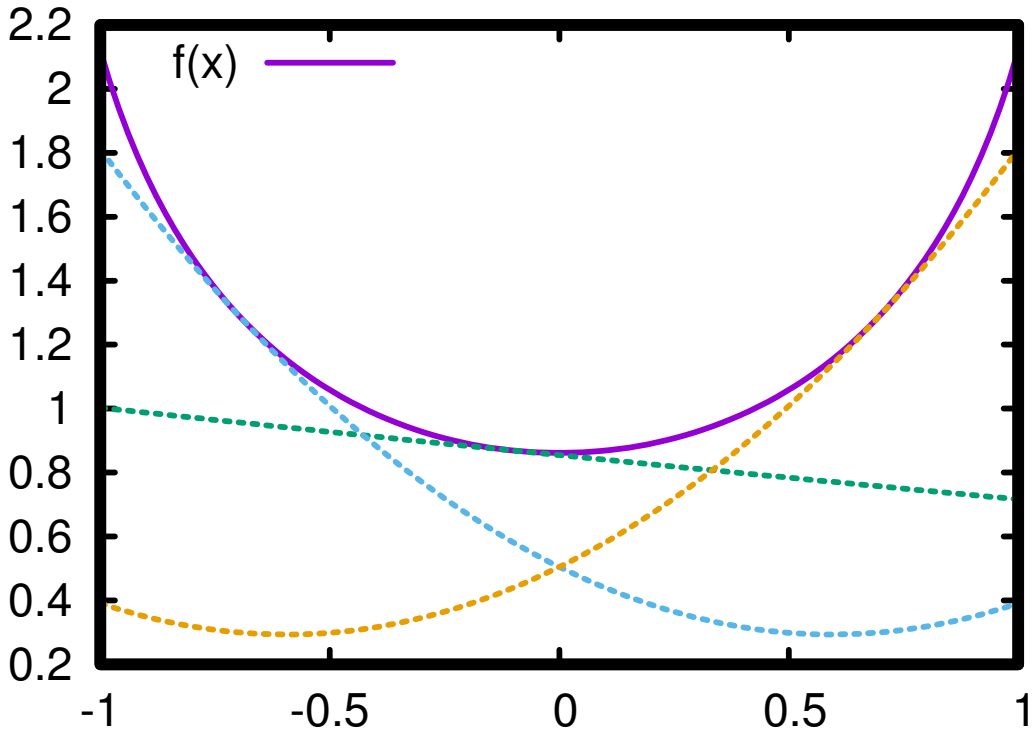
$$\nabla^2 f(\boldsymbol{u}) \;\succeq\; \alpha \nabla f(\boldsymbol{u}) \nabla f(\boldsymbol{u})^\intercal.$$

- Canonical example: $f(\boldsymbol{u}) = -\ln(\boldsymbol{u}^\intercal \boldsymbol{x})$ is 1-exp-concave.
- $f(\boldsymbol{u}) = g(\boldsymbol{u}^\intercal \boldsymbol{x}_t)$ is α-exp-concave if $g : \mathbb{R} \to \mathbb{R}$ is.
- Weak requirement: may be linear orthogonal to gradient.

From Exp-Concavity to Quadratics

Exp-concavity implies [Hazan et al., 2007] for all $\boldsymbol{u}, \boldsymbol{w} \in \mathcal{U}$

$$f(\boldsymbol{u}) \;\geq\; f(\boldsymbol{w}) + (\boldsymbol{u}-\boldsymbol{w})^\intercal \nabla f(\boldsymbol{w}) + \frac{\alpha'}{2}\big((\boldsymbol{u}-\boldsymbol{w})^\intercal \nabla f(\boldsymbol{w})\big)^2$$



Flat near minimum, curved far away.

Exp-concavity, Surrogate

We take surrogate loss

$$\breve{f}_t(\boldsymbol{u}) \;:=\; f_t(\boldsymbol{w}_t) + (\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t + \frac{\alpha'}{2}\big((\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t\big)^2.$$

Q: coefficient of quadratic not constant. What effect would that have on the algorithm?

Exp-concavity, Algorithm

Let us start with Guassian prior $\pi = \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$. Executing

$$\tilde{\pi}_{t+1}(\boldsymbol{u}) \;\propto\; \pi_t(\boldsymbol{u}) e^{-\eta \check{f}_t(\boldsymbol{u})} \qquad \text{and} \qquad \pi_{t+1} \;=\; \text{project}_{\text{KL}}^{\mathcal{U}}(\tilde{\pi}_{t+1})$$

we find that all distributions remain Gaussian, namely

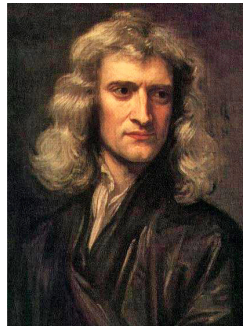
$$\tilde{\pi}_{t+1} \;=\; \mathcal{N}(\tilde{\boldsymbol{w}}_{t+1}, \boldsymbol{\Sigma}_{t+1}) \qquad \text{and} \qquad \pi_{t+1} \;=\; \mathcal{N}(\boldsymbol{w}_{t+1}, \boldsymbol{\Sigma}_{t+1})$$

where

$$\begin{aligned}
\boldsymbol{\Sigma}_{t+1} &= \left(\boldsymbol{I}/\sigma^2 + \eta\alpha' \sum_{s=1}^{t} \boldsymbol{g}_s \boldsymbol{g}_s^\intercal \right)^{-1} \\
\tilde{\boldsymbol{w}}_{t+1} &= \boldsymbol{w}_t - \eta \boldsymbol{\Sigma}_{t+1} \boldsymbol{g}_t \\
\boldsymbol{w}_{t+1} &= \operatorname*{argmin}_{\boldsymbol{u} \in \mathcal{U}} \; (\boldsymbol{u} - \tilde{\boldsymbol{w}}_{t+1})^\intercal \boldsymbol{\Sigma}_{t+1}^{-1} (\boldsymbol{u} - \tilde{\boldsymbol{w}}_{t+1})
\end{aligned}$$

Online Newton Step [Hazan et al., 2007].

Online Newton Step

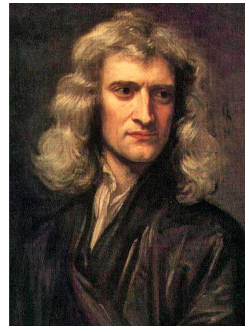


Online Newton Step

$$
\begin{aligned}
\boldsymbol{\Sigma}_{t+1} &= \left(\boldsymbol{I}/\sigma^2 + \eta\alpha' \sum_{s=1}^{t} \boldsymbol{g}_s \boldsymbol{g}_s^\intercal \right)^{-1} \\
\tilde{\boldsymbol{w}}_{t+1} &= \boldsymbol{w}_t - \eta \boldsymbol{\Sigma}_{t+1} \boldsymbol{g}_t \\
\boldsymbol{w}_{t+1} &= \operatorname*{argmin}_{\boldsymbol{u} \in \mathcal{U}} \ (\boldsymbol{u} - \tilde{\boldsymbol{w}}_{t+1})^\intercal \boldsymbol{\Sigma}_{t+1}^{-1} (\boldsymbol{u} - \tilde{\boldsymbol{w}}_{t+1})
\end{aligned}
$$

- $O(d^2)$ memory.
- $O(d^2)$ rank-one update for matrix inverse.
- Projection in Mahalanobis distance (may be $O(d^3)$).

Online Newton Step, Analysis



Theorem (Hazan et al. 2007)

Let $\|\boldsymbol{u}\| \leq D$ and $\|\boldsymbol{g}_t\| \leq G$. Online Newton Step guarantees regret at most

$$R_T^{\boldsymbol{u}} \;\leq\; \frac{D^2}{2\eta\sigma^2} + \frac{d}{2\alpha'} \ln\left(1 + \frac{\alpha'\eta\sigma^2}{d} G^2 T\right)$$

Tuning

$$\eta\sigma^2 \;=\; \frac{\alpha' D^2}{d}$$

results in

$$R_T^{\boldsymbol{u}} \;=\; O(d \ln T).$$

MetaGrad

Recent MetaGrad algorithm [Van Erven and Koolen, 2016] does not need to know **strength of curvature (α or β)** or even **type of curvature**.

- $O(\sqrt{T})$ worst-case bound
- $O(d \ln T)$ bound for exp-concave
- $O(d \ln T)$ bound for strongly convex work in progress

Conclusion

- Design of generic methods for continuous action space
- Based on quadratic (or linear) surrogate losses.
- Weights stay Gaussian
- (c.f. Bregman Divergence)
- Efficient algorithm
- Complexity of problems (so far) property of **curvature**
  - Strongly convex losses are easiest $O(\ln T)$
  - Exp-concave losses are easy $O(d \ln T)$
  - Convex loss are hard $O(\sqrt{T})$.
- MetaGrad adapts to unknown curvature.

Part III

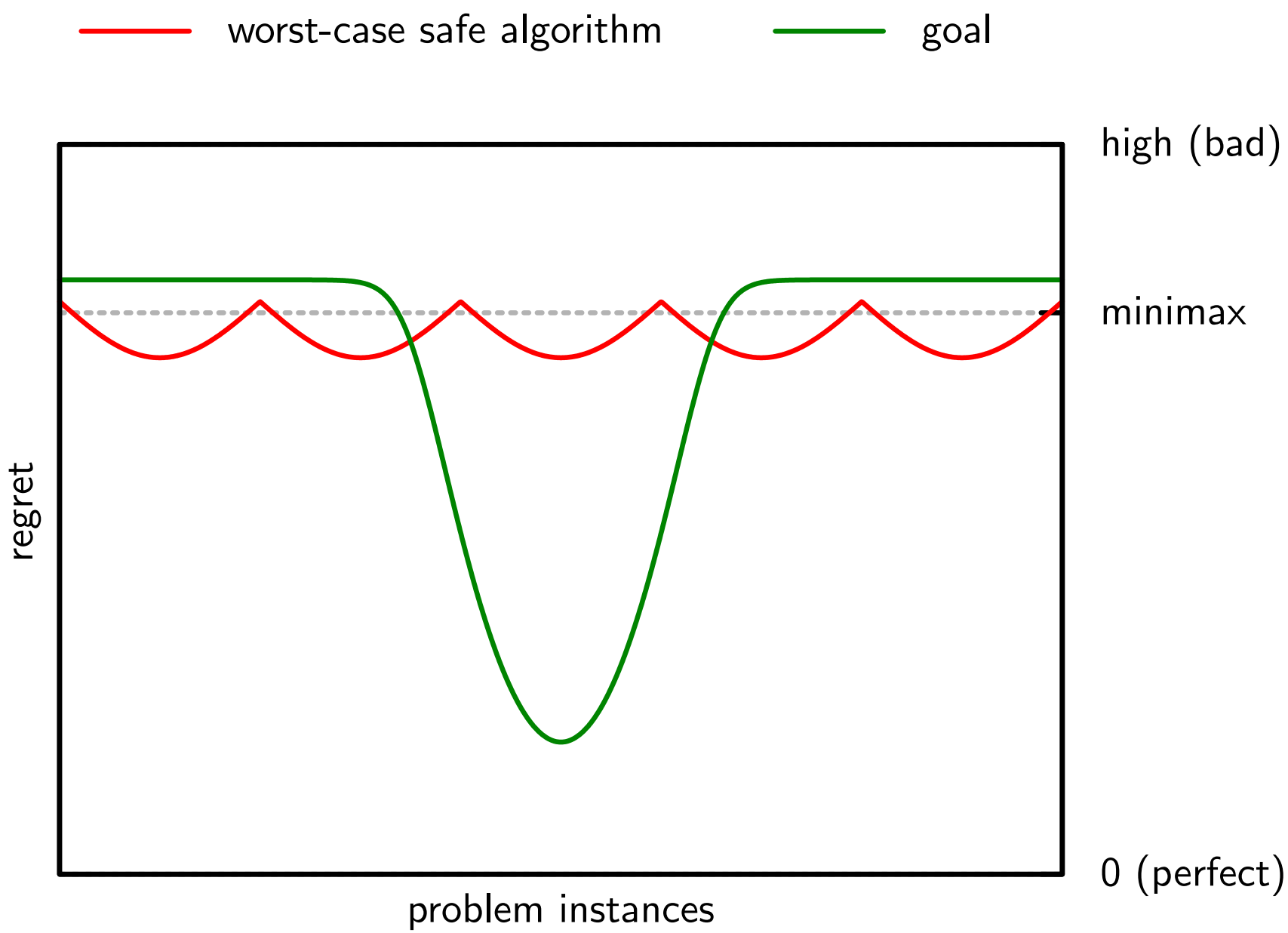
Exploiting Stochastic Luckiness

Overview

Focus on problems of practical importance

- The worst case is often too pessimistic
- Increase performance in “easy” cases
- Without sacrificing worst-case regret guarantees

Intrinsic task difficulty

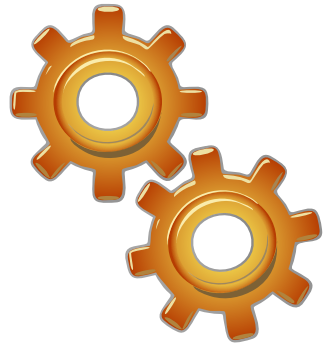


Outline

Individual Sequence Bounds

Stochastic Luckiness

Recap



Recall that Hedge with learning rate η plays weights

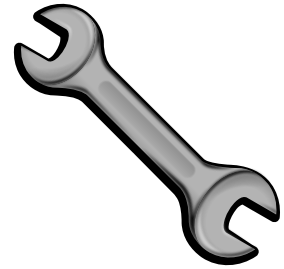
$$w_{t+1}^k \propto e^{-\eta L_t^k}$$

With properly tuned η, we find

$$R_T^k \preceq \sqrt{T \ln K}.$$

Q: Can we do better?
A: No, because matching lower bound!
Q: Can we refine T to quantity that is small for "easy data".

Improvement for Small Losses



Theorem (Auer et al. 2002)

A modification of the Hedge algorithm (w. decaying learning rate) guarantees

$$R_T^k \preceq \sqrt{L_T^k \ln K}.$$

Good when there is a good expert with a tiny (sublinear) loss.

Second-order Improvement

Cesa-Bianchi et al. [2007], Hazan and Kale [2010], Chiang et al. [2012], De Rooij et al. [2014], Gaillard et al. [2014], Steinhardt and Liang [2014], Luo and Schapire [2015], Koolen and Van Erven [2015] all show that

Theorem

A modification of the Hedge algorithm guarantees

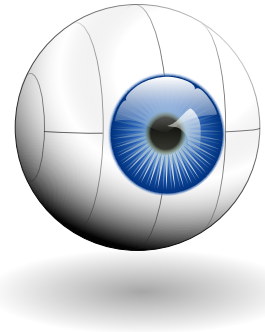
$$R_T^k \preceq \sqrt{V_T^k \ln K}.$$

for some second-order quantity $V_T^k \leq L_T^k \leq T$.

Typically hard to **judge exactly when** $V_T^k \ll T$.

We look at one such method next.

Squint



Recall that Hedge with learning rate η plays weights

$$w_{t+1}^k \;\propto\; e^{-\eta L_t^k} \;\propto\; e^{\eta R_t^k},$$

where $R_T^k = \sum_{t=1}^T (\boldsymbol{w}_t^\intercal \boldsymbol{\ell}_t - \ell_t^k)$ is regret w.r.t. expert k.

Squint [Koolen and Van Erven, 2015] learns the learning rate. It plays

$$w_{t+1}^k \;\propto\; \int_0^{1/2} e^{\eta R_t^k - \eta^2 V_t^k} \,\mathrm{d}\eta$$

where $V_T^k = \sum_{t=1}^T (\boldsymbol{w}_t^\intercal \boldsymbol{\ell}_t - \ell_t^k)^2$ is the excess loss squared.

Theorem (Koolen and Van Erven 2015)

Squint ensures

$$R_T^k \;\preceq\; \sqrt{V_T^k \ln K}$$

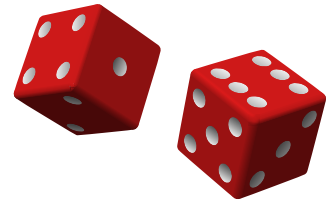
Run-time, non-uniform prior, quantiles, infinitely many experts, ...

Outline

Individual Sequence Bounds

Stochastic Luckiness

IID Stochastic Setting



We consider loss vectors $\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \ldots$ iid from $\mathbb{P}$.

Best action:

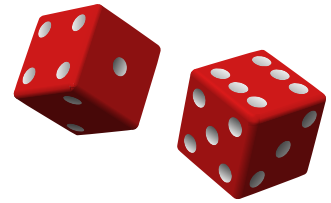
$$k^* \;=\; \operatorname*{argmin}_k \;\mathbb{E}[\ell^k]$$

We'd like to have small expected regret w.r.t. k^*, i.e.

$$\mathbb{E}[L_T] - \min_k \mathbb{E}[L_T^k] \;=\; \mathbb{E}[R_T^{k^*}]$$

without spoiling our worst-case regret bounds.

Gap

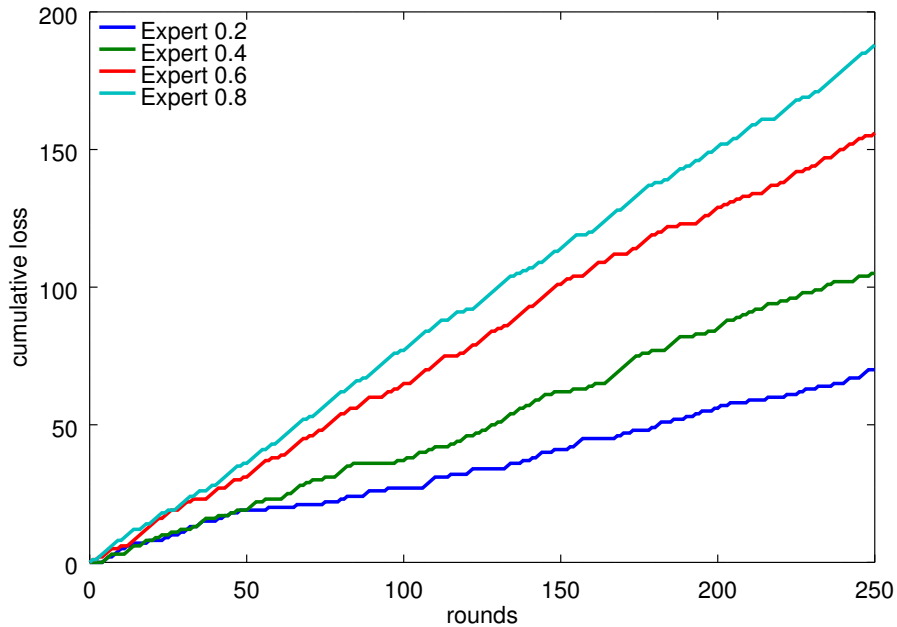


What if the best expert is **obvious**?

Definition

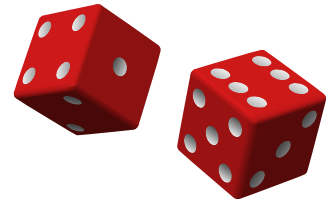
We say $\mathbb{P}$ has **gap** γ if

$$\mathbb{E}[\ell^k] \;\geq\; \mathbb{E}[\ell^{k^*}] + \gamma \qquad \text{for all } k \neq k^*.$$



Is this still $\sqrt{T}$ hard? It does outlaw the lower bound adversary . . .

Gap



Kotłowski [2015] shows that the minimax algorithm **for iid data** is (binarised) **Follow The Leader**.
(c.f. popularity of **Empirical Risk Minimisation** in batch classification.)

But we already saw that FTL is very dangerous.
Robust hybrids by [De Rooij et al., 2014] and [Sani et al., 2014].

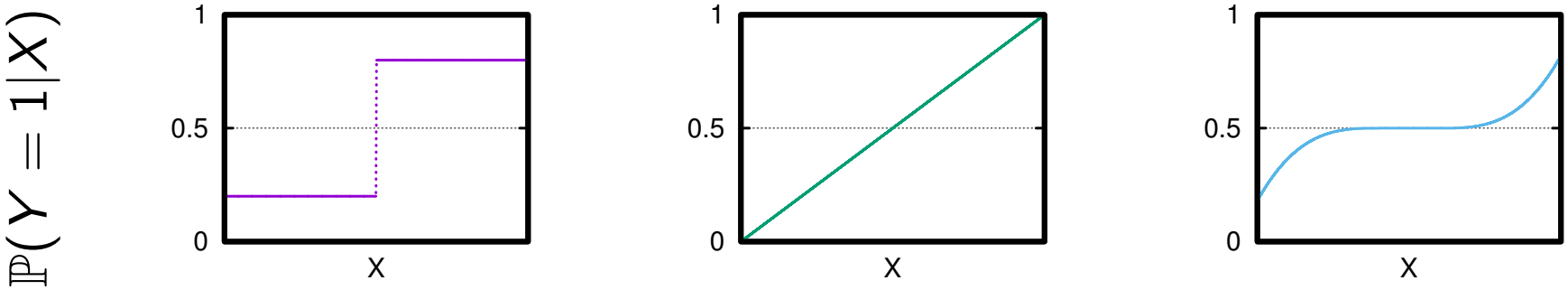
Also, FTL/ERM does not work on infinitely many experts.

Next: we consider a weaker condition and show that Squint exploits it.

Inspiration: Tsybakov margin condition for classification

Classification: $Y \in \{0, 1\}$.

$$\mathbb{P}\Big(\big|\mathbb{P}(Y = 1|X) - 1/2\big| \leq t\Big) \;\leq\; ct^{\alpha}$$



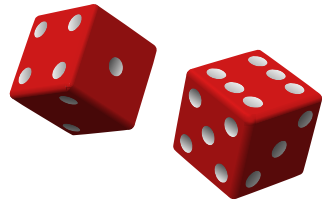
easy
$\alpha = \infty$

moderate
$\alpha = 1$

hard
$\alpha = 0$

Confusing case: predictors with equal risk but opposite predictions.

Massart Condition



Consider loss vectors $\ell_1, \ell_2, \ldots$ iid from $\mathbb{P}$.
Best action:

$$k^* \;=\; \operatorname*{argmin}_k \mathbb{E}\left[\ell^k\right]$$

Idea: if actions close to the best action are similar it is **easy** to detect the best action.

Definition

We say $\mathbb{P}$ is C-**Massart** if

$$\mathbb{E}\left[(\ell^k - \ell^{k^*})^2\right] \;\leq\; C\,\mathbb{E}\left[\ell^k - \ell^{k^*}\right] \qquad \text{for all } k.$$

Gap implies Massart

Recall gap: for $k \neq k^*$

$$\mathbb{E}[\ell^k - \ell^{k^*}] \geq \gamma$$

We show that γ-gap implies $\frac{1}{\gamma}$-Massart.
For $k = k^*$ this is trivial. For $k \neq k^*$

$$\mathbb{E}\left[(\ell^k - \ell^{k^*})^2\right] \leq 1 = \frac{\gamma}{\gamma} \leq \frac{1}{\gamma}\mathbb{E}[\ell^k - \ell^{k^*}]$$

So how do we exploit Massart?

Fast rates pipeline

Recall that Squint ensures

$$R_T^k \;\leq\; \sqrt{V_T^k \ln K} \qquad \text{where} \qquad V_T^k \;=\; \sum_{t=1}^{T} (\boldsymbol{w}_t^\mathsf{T} \boldsymbol{\ell}_t - \ell_t^k)^2$$

We now show that under the C-Massart condition

$$\mathbb{E}\left[R_T^{k^*}\right] \;\leq\; C \ln K .$$

The argument will have two parts

- Show that $\mathbb{E}[V_T^{k^*}] \leq C\,\mathbb{E}[R_T^{k^*}]$.
- "Solve" the regret bound

First step: $\mathbb{E}\, V \leq C\, \mathbb{E}\, R$

Each round

$$\mathbb{E}\left[(\boldsymbol{w}_t^\mathsf{T}\boldsymbol{\ell}_t - \ell_t^{k^*})^2\right] \;\leq\; \sum_k w_t^k\, \mathbb{E}\left[(\ell_t^k - \ell_t^{k^*})^2\right]$$

$$\leq\; C \sum_k w_t^k\, \mathbb{E}\left[\ell_t^k - \ell_t^{k^*}\right] \;=\; C\, \mathbb{E}\left[\boldsymbol{w}_t^\mathsf{T}\boldsymbol{\ell}_t - \ell_t^{k^*}\right].$$

Hence in sum

$$\mathbb{E}\left[V_T^{k^*}\right] \;\leq\; C\, \mathbb{E}\left[R_T^{k^*}\right].$$

Second step: solve regret bound

By the regret bound

$$\mathbb{E}\left[R_T^{k^*}\right] \;\leq\; \mathbb{E}\left[\sqrt{V_T^{k^*} \ln K}\right] \;\leq\; \sqrt{\mathbb{E}\left[V_T^{k^*}\right] \ln K}$$

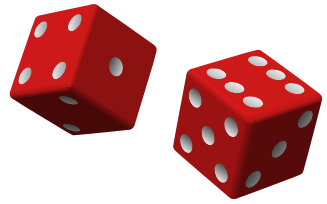
And hence

$$\mathbb{E}\left[R_T^{k^*}\right] \;\leq\; \sqrt{C\,\mathbb{E}\left[R_T^{k^*}\right] \ln K}$$

Squaring and dividing by $\mathbb{E}\left[R_T^{k^*}\right]$ results in

$$\mathbb{E}\left[R_T^{k^*}\right] \;\leq\; C \ln K.$$

Bernstein Condition



We may still show fast learning under an even weaker condition.

Definition (Bartlett and Mendelson 2006)

Fix $C > 0$ and $\kappa \in [0, 1]$. We say $\mathbb{P}$ is (C, κ)-**Bernstein** if

$$\mathbb{E}\left[(\ell^k - \ell^{k^*})^2\right] \leq C\,\mathbb{E}\left[\ell^k - \ell^{k^*}\right]^\kappa \qquad \text{for all } k.$$

It can be shown that now a regret bound of the form (e.g. Squint)

$$R_T^k \leq \sqrt{V_T^k \ln K}$$

implies

$$\mathbb{E}\left[R_T^{k^*}\right] \leq T^{\frac{1-\kappa}{2-\kappa}} (\ln K)^{\frac{1}{2-\kappa}}.$$

Interpolates from $\sqrt{T}$ for hard $\kappa = 0$ to a constant for easy $\kappa = 1$.

MetaGrad

Squint learns the learning rate for experts. The analogue for online convex optimisation is called MetaGrad.

- $\sqrt{T}$ worst-case bound
- $d \ln T$ bound for exp-concave and strongly convex.
- $d \ln T$ for OCO Massart
- $T^{\frac{1-\kappa}{2-\kappa}} (d \ln T)^{\frac{1}{2-\kappa}}$ for OCO κ-Bernstein

Conclusion

Individual sequence simplicity measures

- Small losses
- Zoo of second-order quantities

In stochastic setting quantify complexity **of the distribution**.

- Gap
- Massart condition
- Bernstein condition

In each case exploited by Squint or MetaGrad, due to their ability to **learn the learning rate**.

Thank you!