OEUVRE: OnlinE Unbiased Variance-Reduced loss Estimation

Kanad Pardeshi, Bryan Wilder, Aarti Singh
Carnegie Mellon University

Problem: Online Loss Estimation

- Consider sequential data $\{z_t\}_{t \geq 1}$, where $z_t = (x_t, y_t) \sim \mathscr{D}$
- Learning algorithm $\mathscr{A}$ trained on data, with function $f_t : \mathscr{X} \to \mathscr{Y}$ learned using first $(t-1)$ samples
- **Goal**: Obtain accurate estimate of expected loss for currently learned function

$$\mathbb{E}_{(X,Y)\sim\mathscr{D}}[\ell(f_t(X), Y) \mid \mathscr{F}_{t-1}]$$

Problem: Online Loss Estimation

- Consider sequential data $\{z_t\}_{t \geq 1}$, where $z_t = (x_t, y_t) \sim \mathscr{D}$
- Learning algorithm $\mathscr{A}$ trained on data, with function $f_t : \mathscr{X} \to \mathscr{Y}$ learned using first $(t-1)$ samples
- **Goal**: Obtain accurate estimate of expected loss for currently learned function

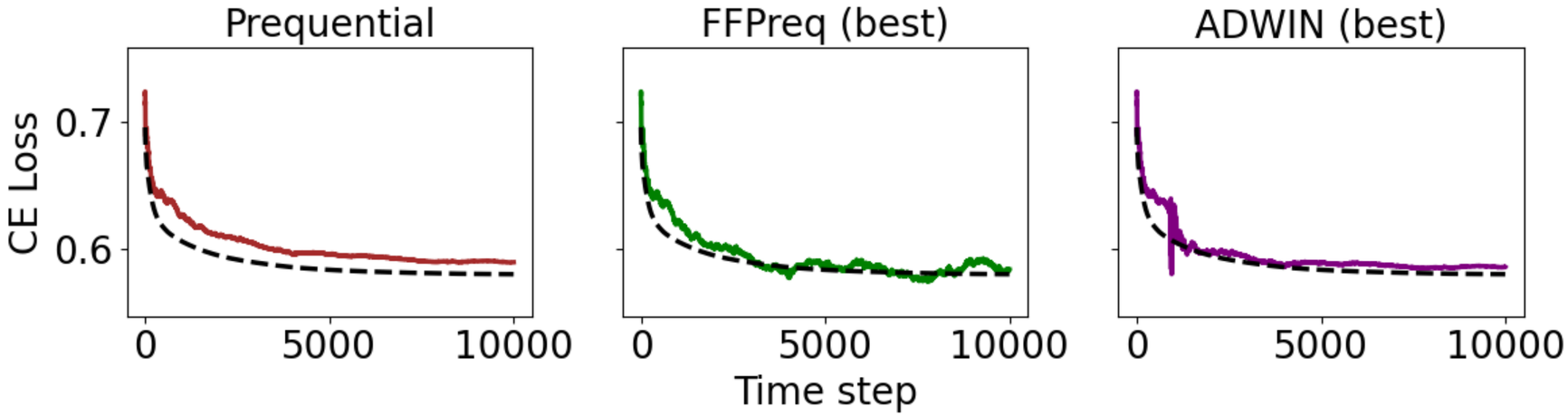
$$\mathbb{E}_{(X,Y)\sim\mathscr{D}}[\mathscr{l}(f_t(X), Y) \mid \mathscr{F}_{t-1}]$$

Applications
Online model monitoring
Early stopping
Online model selection

Desirable Estimator Properties

We want an estimator with

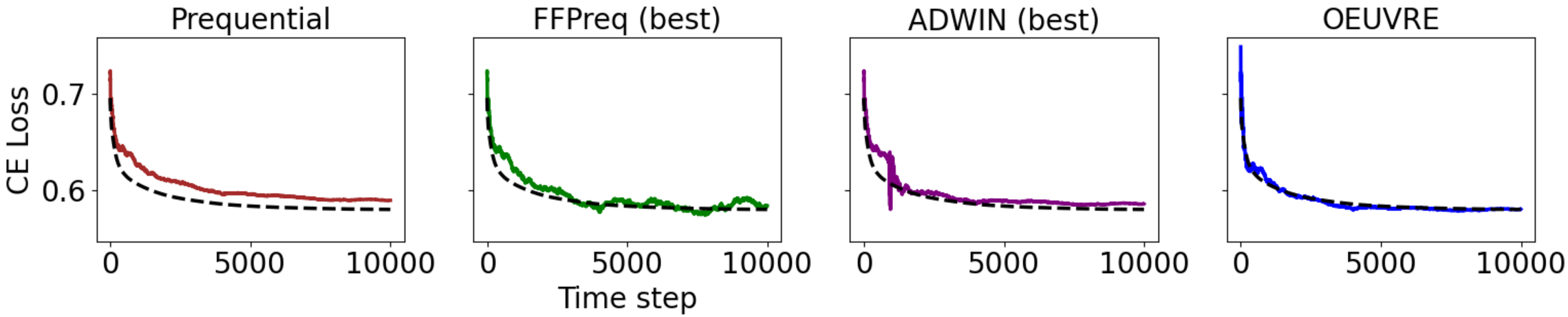
- quick and efficient updates,
- provably strong convergence properties, and
- use of properties of the learning algorithm $\mathscr{A}$



Desirable Estimator Properties

We want an estimator with

- quick and efficient updates,
- provably strong convergence properties, and
- use of properties of the learning algorithm $\mathscr{A}$



Enter *OEUVRE*!

OEUVRE Estimator

OEUVRE Estimator

- Interleaved test-then-train: Evaluate loss of f_t on $z_t = (x_t, y_t)$, then train on z_t

OEUVRE Estimator

- Interleaved test-then-train: Evaluate loss of f_t on $z_t = (x_t, y_t)$, then train on z_t
- **OEUVRE**: Evaluate loss of f_{t-1} *and* f_t on z_t, then train on z_t

OEUVRE Estimator

- Interleaved test-then-train: Evaluate loss of f_t on $z_t = (x_t, y_t)$, then train on z_t
- **OEUVRE**: Evaluate loss of f_{t-1} *and* f_t on z_t, then train on z_t
  - Recursive update rule

OEUVRE Estimator

- Interleaved test-then-train: Evaluate loss of f_t on $z_t = (x_t, y_t)$, then train on z_t
- **OEUVRE**: Evaluate loss of f_{t-1} *and* f_t on z_t, then train on z_t
  - Recursive update rule

$$L_t = \ell(f_t(x_t), y_t) + (1 - \gamma_t) \cdot \left[L_{t-1} - \ell(f_{t-1}(x_t), y_t)\right]$$

OEUVRE Estimator

- Interleaved test-then-train: Evaluate loss of f_t on $z_t = (x_t, y_t)$, then train on z_t
- **OEUVRE**: Evaluate loss of f_{t-1} *and* f_t on z_t, then train on z_t
  - Recursive update rule

$$L_t = \ell(f_t(x_t), y_t) + (1 - \gamma_t) \cdot \left[L_{t-1} - \ell(f_{t-1}(x_t), y_t)\right]$$

Variance reduction term

OEUVRE Estimator

- Interleaved test-then-train: Evaluate loss of f_t on $z_t = (x_t, y_t)$, then train on z_t
- **OEUVRE**: Evaluate loss of f_{t-1} *and* f_t on z_t, then train on z_t
  - Recursive update rule

$$L_t = \ell(f_t(x_t), y_t) + (1 - \gamma_t) \cdot \left[L_{t-1} - \ell(f_{t-1}(x_t), y_t)\right]$$

Weight term.
Set γ_t according to learning algorithm!

Variance reduction term

Choosing Weight γ_t

Choosing Weight γ_t

$$L_t = \ell_t(z_t) + (1 - \gamma_t)\left[L_{t-1} - \ell_{t-1}(z_t)\right]$$

Choosing Weight γ_t

$$L_t = \ell_t(z_t) + (1 - \gamma_t)\left[L_{t-1} - \ell_{t-1}(z_t)\right]$$

- Choose γ_t to minimize an upper bound on Var(L_t)

Choosing Weight γ_t

$$L_t = \ell_t(z_t) + (1 - \gamma_t)\left[L_{t-1} - \ell_{t-1}(z_t)\right]$$

- Choose γ_t to minimize an upper bound on Var(L_t)
- Use stability properties!

Choosing Weight γ_t

$$L_t = \ell_t(z_t) + (1 - \gamma_t)\left[L_{t-1} - \ell_{t-1}(z_t)\right]$$

- Choose γ_t to minimize an upper bound on Var(L_t)
- Use stability properties!

Algorithm $\mathscr{A}$ is β_t-*uniformly stable* if

$$\|\ell(f_t(x), y) - \ell(f_{t-1}(x), y)\|_\infty \leq \beta_t$$

Choosing Weight γ_t

$$L_t = \ell_t(z_t) + (1 - \gamma_t)\left[L_{t-1} - \ell_{t-1}(z_t)\right]$$

- Choose γ_t to minimize an upper bound on Var(L_t)
- Use stability properties!

> Algorithm $\mathscr{A}$ is β_t-*uniformly stable* if
>
> $$\|\ell(f_t(x), y) - \ell(f_{t-1}(x), y)\|_\infty \leq \beta_t$$

- Intuitively: bound max. change in loss

Choosing Weight γ_t

$$L_t = \ell_t(z_t) + (1 - \gamma_t)\left[L_{t-1} - \ell_{t-1}(z_t)\right]$$

- Choose γ_t to minimize an upper bound on Var(L_t)
- Use stability properties!

Algorithm $\mathscr{A}$ is β_t-*uniformly stable* if

$$\|\ell(f_t(x), y) - \ell(f_{t-1}(x), y)\|_\infty \leq \beta_t$$

- Intuitively: bound max. change in loss
- Use β_t to find optimal γ_t

Choosing Weight γ_t

$$L_t = \ell_t(z_t) + (1-\gamma_t)\left[L_{t-1} - \ell_{t-1}(z_t)\right]$$

- Choose γ_t to minimize an upper bound on $\mathrm{Var}(L_t)$
- Use stability properties!

Algorithm $\mathscr{A}$ is β_t-*uniformly stable* if

$$\|\ell(f_t(x), y) - \ell(f_{t-1}(x), y)\|_\infty \leq \beta_t$$

- Intuitively: bound max. change in loss
- Use β_t to find optimal γ_t

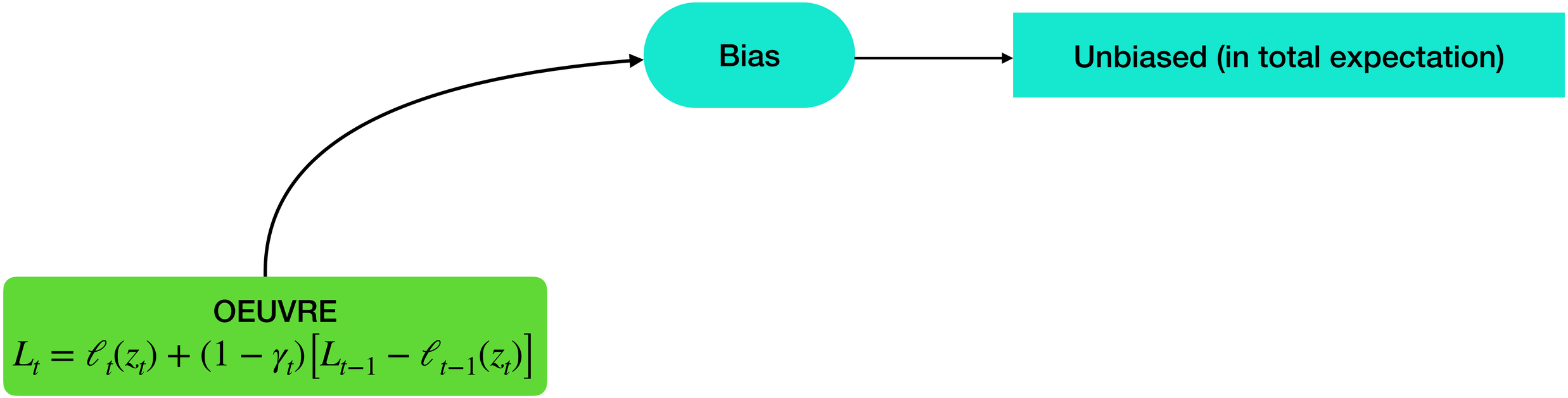
Algorithmic paradigm	Stability bound
Follow the leader (FTL)	$O(1/t)$
Follow the regularized leader (FTRL)	$O(1/\sqrt{t})$
Regularized dual averaging (RDA)	$O(1/\sqrt{t})$
Online mirror descent (OMD)	$O(\eta_t)$

Theoretical Properties

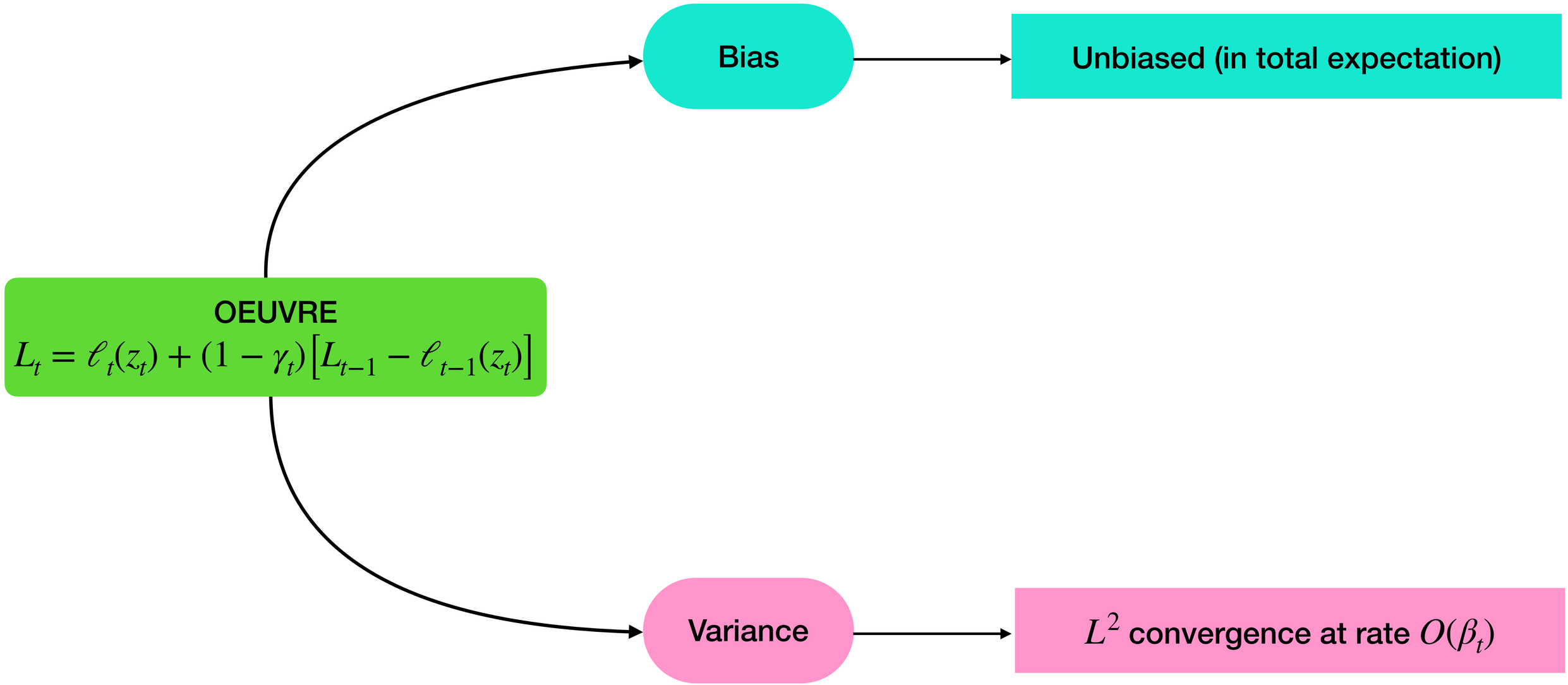
OEUVRE

$$L_t = \ell_t(z_t) + (1 - \gamma_t)\left[L_{t-1} - \ell_{t-1}(z_t)\right]$$

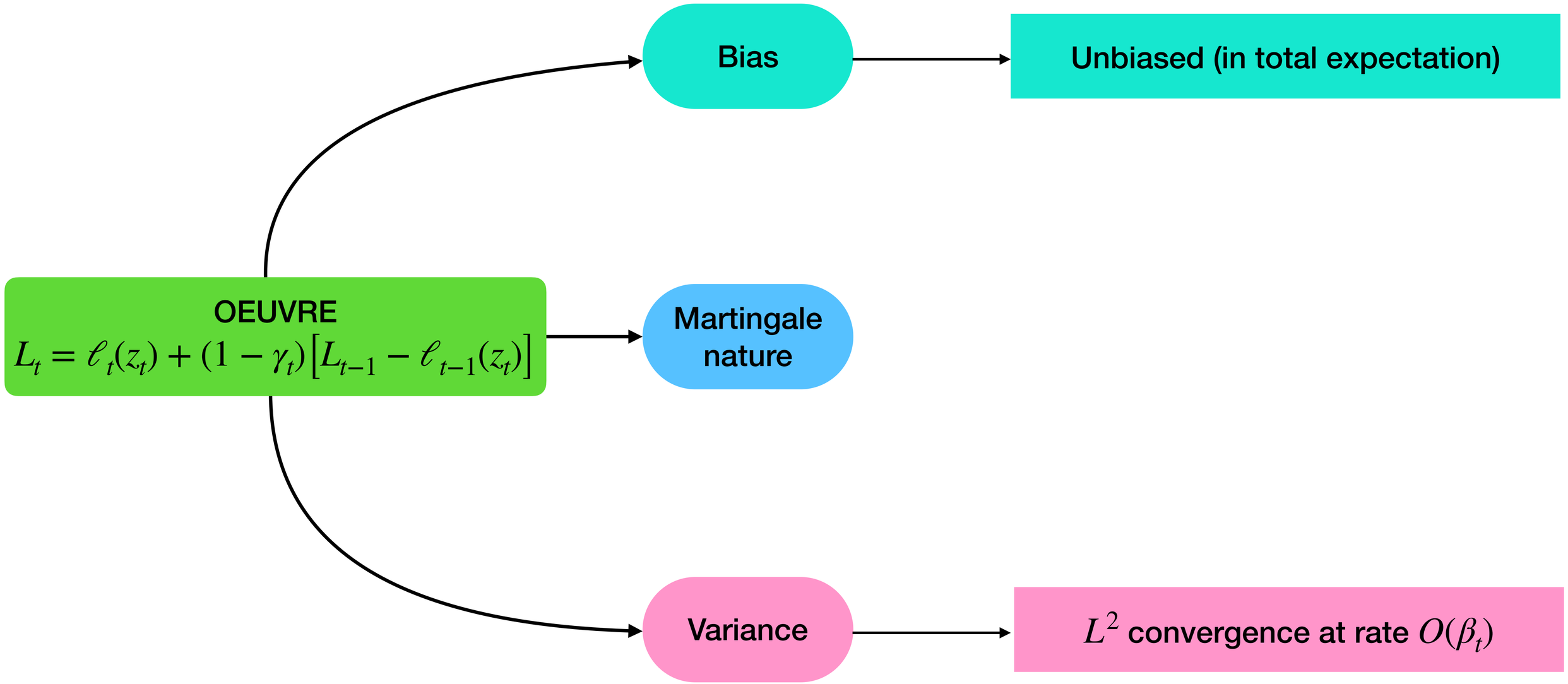
Theoretical Properties



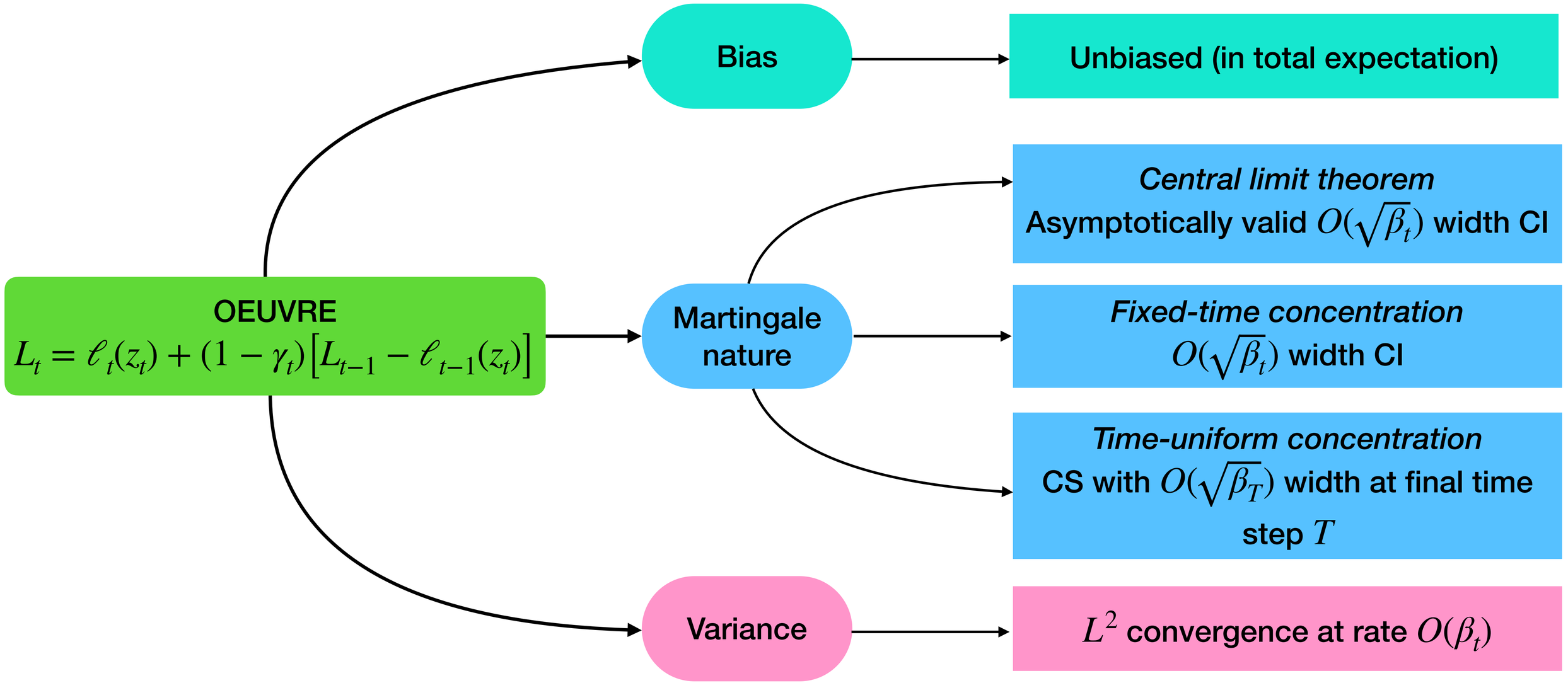
Theoretical Properties



Theoretical Properties



Theoretical Properties



Summary

- OEUVRE: extra evaluation at each time step for variance reduction
- Leverage algorithmic stability properties to design plug-and-play estimator
- Strong statistical properties (convergence, confidence intervals)
- Hyperparam-free OEUVRE: Adaptively estimate constants in stability rate
- Experiments:
    1. Linear regression
    2. Prediction with expert advice
    3. Simple neural networks

Check out our paper!
Poster #173
3:00pm