Online machine learning

Chris | Dong | Edvards | Hasan | Yang

Introduction

What is online machine learning?

- Data becomes available in sequential order
- New data is used to incrementally update our model rather than batch learning
- Useful when
  - training over the entire dataset is intractable
  - new patterns dynamically emerge
  - data is generated over time

Basic concept

- The framework is game-theoretic and adversarial
- Regret is the difference between the total incurred cost and the cost of the best decision in hindsight

$$\text{regret} = \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x})$$

For each iteration

1. The decider makes a choice
   $\mathbf{x}_t \in \mathcal{K}$
2. A convex cost function is revealed
   $f_t \in \mathcal{F} : \mathcal{K} \mapsto \mathbb{R}$
3. The decider incurs a cost
   $f_t(\mathbf{x}_t)$
4. The decider make a new choice to minimise regret

Convex optimisation

- We seek to minimise a continuous convex function over a convex subset of Euclidean space
- Gradient descent (GD) is the simplest and oldest optimisation method
- GD lays the foundation for more efficient and forthcoming algorithms

Example applications

- Prediction from expert advice
  - Decider tries to perform as well as experts in hindsight
- Online spam filtering
  - Learning a binary classifier
- Online shortest paths in graph
  - Decider chooses the path
  - Adversary chooses the cost
- Portfolio allocation
  - Decider chooses distribution of wealth over assets
  - Adversary chooses market returns
  - Decider learns to rebalance portfolio

Example of applications in use

- Twitter
  - Automatic identification of breaking news from the twitter stream
- Delays in overlay networks
  - Peep-to-peer applications change unpredictably as the load in the underlay network fluctuates
- Supervised learning on large
  - Considering one data point at a time reduces complexity per iteration but might increase the number of iterations
- Financial data analysis
  - Invest and maximize expected utility

First and second order methods

First order

Online gradient descent

- Step in the direction of the gradient of the previous cost
- If the new point is extraneous to the underlying convex set, project it back within
- The regret is sublinear
- But projection is burdensome

1: Input: convex set $\mathcal{K}$, T, $\mathbf{x}_1 \in \mathcal{K}$, step sizes $\{\eta_t\}$
2: **for** $t = 1$ to T **do**
3: Play $\mathbf{x}_t$ and observe cost $f_t(\mathbf{x}_t)$.
4: Update and project:

$$\mathbf{y}_{t+1} = \mathbf{x}_t - \eta_t \nabla f_t(\mathbf{x}_t)$$
$$\mathbf{x}_{t+1} = \prod_{\mathcal{K}}(\mathbf{y}_{t+1})$$

5: **end for**

Second order

Online Newton step

- Approximates second derivative
- Requires less iterations
- But each step is costly

Input: convex set $\mathcal{K}$, T, $\mathbf{x}_1 \in \mathcal{K} \subseteq \mathbb{R}^n$, parameters $\gamma, \varepsilon > 0$, $A_0 = \varepsilon \mathbf{I}_n$

for $t = 1$ to T **do**

  Play $\mathbf{x}_t$ and observe cost $f_t(\mathbf{x}_t)$.

  Rank-1 update: $A_t = A_{t-1} + \nabla_t \nabla_t^\top$

  Newton step and projection:

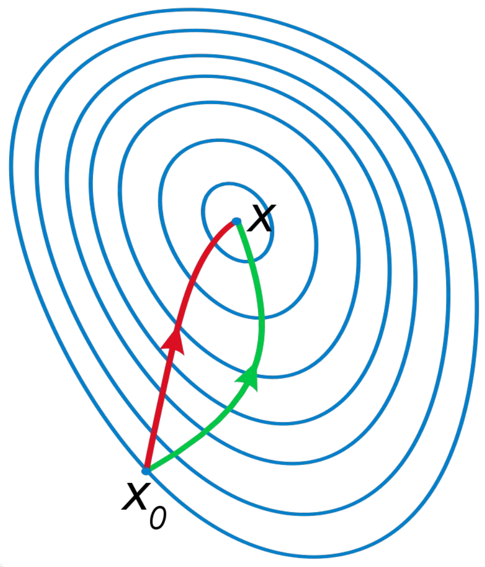
$$\mathbf{y}_{t+1} = \mathbf{x}_t - \frac{1}{\gamma} A_t^{-1} \nabla_t$$

$$\mathbf{x}_{t+1} = \prod_{\mathcal{K}}^{A_t} (\mathbf{y}_{t+1})$$

end for

Pros and cons...

- These are reliable algorithms
- But, both algorithms require projection back into the convex set if they step out
- Projection is “expensive”



Green: gradient descent
Red: Newton descent

Regularisation

Follow the leader (FTL)

- At any point in time, use the optimal decision in retrospect
- Simple strategy
- Regret is linear in iterations
- Very unstable, changing decision too often
- Considered a greedy algorithm

Updates with the rule:

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathcal{K}} \sum_{\tau=1}^{t} f_\tau(\mathbf{x})$$

FTL example

Consider

$$\mathcal{K} = [-1, 1],$$

Let

$$f_1(x) = \frac{1}{2}x,$$

And let

$$f_\tau \text{ for } \tau = 2, \ldots, T$$

Alternate between **-x** and **x**

Thus

$$\sum_{\tau=1}^{t} f_\tau(x) = \begin{cases} \frac{1}{2}x, & t \text{ is odd} \\ -\frac{1}{2}x, & \text{otherwise} \end{cases}$$

This strategy will keep shifting between

$$x_t = -1 \text{ and } x_t = 1$$

This will always give the wrong choice because it is unstable.

Regularised follow the leader (RFTL)

- Adds a regularisation function
- Stabilises the prediction
- Gives asymptotically optimal regret bounds
- The regulariser is strongly convex, smooth, and twice differentiable

Algorithm 10 Regularized Follow The Leader

Input: $\eta > 0$, regularization function R, and a convex compact set $\mathcal{K}$.
Let $\mathbf{x}_1 = \arg\min_{\mathbf{x} \in \mathcal{K}} \{R(\mathbf{x})\}$.
for $t = 1$ to T **do**
  Predict $\mathbf{x}_t$.
  Observe the payoff function f_t and let $\nabla_t = \nabla f_t(\mathbf{x}_t)$.
  Update

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x} \in \mathcal{K}} \left\{ \eta \sum_{s=1}^{t} \nabla_s^\top \mathbf{x} + R(\mathbf{x}) \right\}$$

end for

Optimal regularisation

- We assume the regulariser is a strongly convex function, but which one?
    - It should depend on the decision set and cost function
- Adaptive subgradient method (AdaGrad)
    - Learns the optimal regulariser in hindsight online!

Algorithm 16 AdaGrad

Input: parameters $\eta, \mathbf{x}_1 \in \mathcal{K}$.
Initialize: $S_0 = G_0 = \mathbf{0}$,
for $t = 1$ to T **do**
  Predict $\mathbf{x}_t$, suffer loss $f_t(\mathbf{x}_t)$.
  Update:

$$S_t = S_{t-1} + \nabla_t \nabla_t^\top, \; G_t = {S_t}^{1/2}$$

$$\mathbf{y}_{t+1} = \mathbf{x}_t - \eta G_t^{-1} \nabla_t$$

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x} \in \mathcal{K}} \|\mathbf{y}_{t+1} - \mathbf{x}\|_{G_t}^2$$

end for

A^{-1} refers to the Moore-Penrose pseudoinverse of the matrix

Comparing regrets

RFTL

Theorem 5.1. The RFTL Algorithm 10 attains for every $\mathbf{u} \in \mathcal{K}$ the following bound on the regret:

$$\text{regret}_T \leq 2\eta \sum_{t=1}^{T} \|\nabla_t\|_t^{*2} + \frac{R(\mathbf{u}) - R(\mathbf{x}_1)}{\eta}.$$

AdaGrad considers the set of all strongly convex regularisers with a fixed and bound Hessian in

$$\forall \mathbf{x} \in \mathcal{K} \ . \ \nabla^2 R(\mathbf{x}) = \nabla^2 \in \mathcal{H} \triangleq \{X \in \mathbb{R}^{n \times n} \ ; \ \mathbf{Tr}(X) \leq 1 \ , \ X \succcurlyeq 0\}$$

Theorem 5.9. Let $\{\mathbf{x}_t\}$ be defined by Algorithm 16 with parameters $\eta = D$, where

$$D = \max_{\mathbf{u} \in \mathcal{K}} \|\mathbf{u} - \mathbf{x}_1\|_2.$$

Then for any $\mathbf{x}^\star \in \mathcal{K}$,

$$\text{regret}_T(\text{AdaGrad}) \leq 2D \sqrt{\min_{H \in \mathcal{H}} \sum_t \|\nabla_t\|_H^{*2}}. \tag{5.6}$$

The regret bound is as good as the regret of RFTL for the class of regularization functions

Online Decision-Making

Online Decision Making: Multi-Armed Bandits and Reinforcement learning

General Settings:

- At each time step t=1,2,...T, the decision maker observes a state s_t , choose an actior a_t and receive a reward r_t .
- The action is chosen based on a policy, i.e., a mapping from history to an action.

Goal:

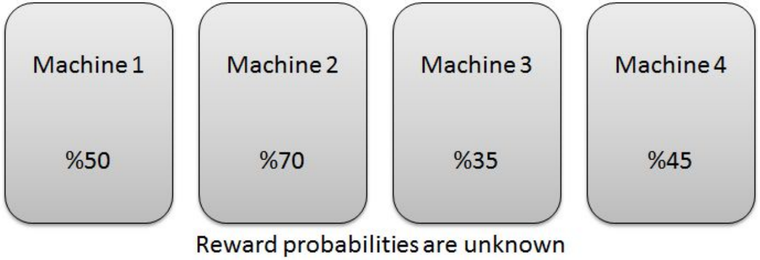
- Find the optimal policies $\{\delta_t : \mathcal{S} \to \mathcal{A}\}_{t \in \mathbb{Z}_+}$ to optimize an objective in terms r_t rewards .

Regret in context of Online Decision-Making: $Regret_{\pi,T} := \mathbb{E}^{\pi^*}\{\sum_{t=1}^{T} r_t\} - \mathbb{E}^{\pi}\{\sum_{t=1}^{T} r_t\}$

- Alternatively, the goal of a decision maker can be minimizing the regret.

Multi Armed Bandit problem

- The exploration vs exploitation dilemma
  - Problem: Where to eat?
  - Dilemma comes from the incomplete information.
  - Exploitation: We take advantage of the best option we know.
  - Exploration: We take some risk to collect information about the unknown options.
- Best long-term strategy may involve some sacrifices.



- Which machine to pick next?
- What is the best strategy to achieve highest long-term rewards?
  - K machines with reward prob. $\{p_1, \dots p_K\}$
  - Each step, we take an action a and receive reward r.
  - The goal is to maximize $\sum r_t$ or to minimize the total regret.

Reinforcement learning in MDPs

Markov decision process:

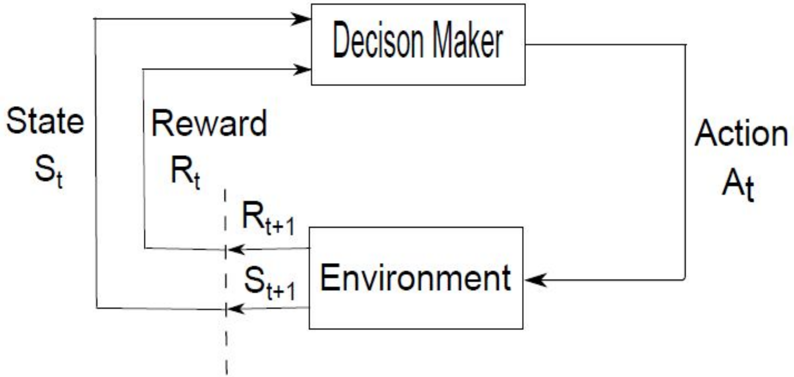
- Considering the average reward MDP here, i.e., the overall objective in terms of reward function is given by:

$$g^{\pi}(s_1) := \liminf_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} r(s_t, \pi(s_t))\right]$$

- The goal is finding a policy π to maximize the gain $g^{\pi}(s_1)$, the maximal gain independent with initial state for communicating MDPS

Basic elements for MDP:

- State : **s**
- Action : **a**
- State transition probability: $\mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a)$
- Random reward independently draw from a distribution, i.e., $r(s,a) \sim \nu(s,a)$ with mean $\mu(s,a)$



Reinforcement learning in MDPs : Optimal Solution

Bellman's Optimality Equation :

$$g^\star + b^\star(s) = \max_{a \in \mathcal{A}} \Big(\mu(s,a) + p(\cdot|s,a)^\top b^\star \Big), \quad \forall s$$

- Where g* is the optimal gain and b* is the bias, i.e., the asymptotic difference in total reward that results from starting the process in different states

Gap for sub-optimal state-action pair:

$$\varphi(s,a) := \big(\mu(s,\pi^\star(s)) - \mu(s,a)\big) + \big(p(\cdot|s,\pi^\star(s)) - p(\cdot|s,a)\big)^\top b^\star$$

Reinforcement learning in MDPs : Regret

Define the regret of a learning algorithm $\mathbb{A}$ after T steps as:

$$\text{Regret}_{\mathbb{A},T}(s_1) := Tg^\star(s_1) - \sum_{t=1}^{T} r(s_t, a_t)$$

- Where,

$$a_t = \mathbb{A}(s_t, (s_{t'}, a_{t'}, r_{t'})_{t'<t})$$

Definition (Diameter (Jaksch et al., 2010))

Let $T_\pi(s'|s)$ denote the first hitting time of state s' when following stationary policy π from initial state s. The diameter D of an MDP M is defined as

$$D := \max_{s \neq s'} \min_{\pi} \mathbb{E}[T_\pi(s'|s)].$$

Any communicating MDP has a finite diameter.

Reinforcement learning in MDPs : Fundamental Performance Limits

Problem-specific regret lower bound:

- For any admissible algorithm $\mathbb{A}$ and any ergodic MDP, the regret satisfies:

$$\liminf_{T\to\infty} \frac{\mathbb{E}[\mathrm{Regret}_{\mathbb{A},T}]}{\log(T)} \geq c_{\mathrm{bk}}(M) := \sum_{(s,a)\in\mathcal{C}_M} \frac{\varphi(s,a)}{\inf\{\mathtt{KL}(p(\cdot|s,a),q) : q \in \Lambda(s,a)\}}$$

- $\Lambda(s,a)$: set of distributions q over states such that replacing $p(\cdot|s,a)$ by q makes a the unique optimal action in s
- $\mathcal{C}_M$: the set of critical state-action pairs in M

Minimax regret lower bound:

Theorem (Minimax LB (Jaksch et al., 2010))

For any T there is an MDP with S states and A actions such that any learning algorithm suffers expected regret of

$$\Omega\left(\sqrt{DSAT}\right),$$

after $T \geq DSA$ steps.

Online Bayes

Scenario of online continuous learning

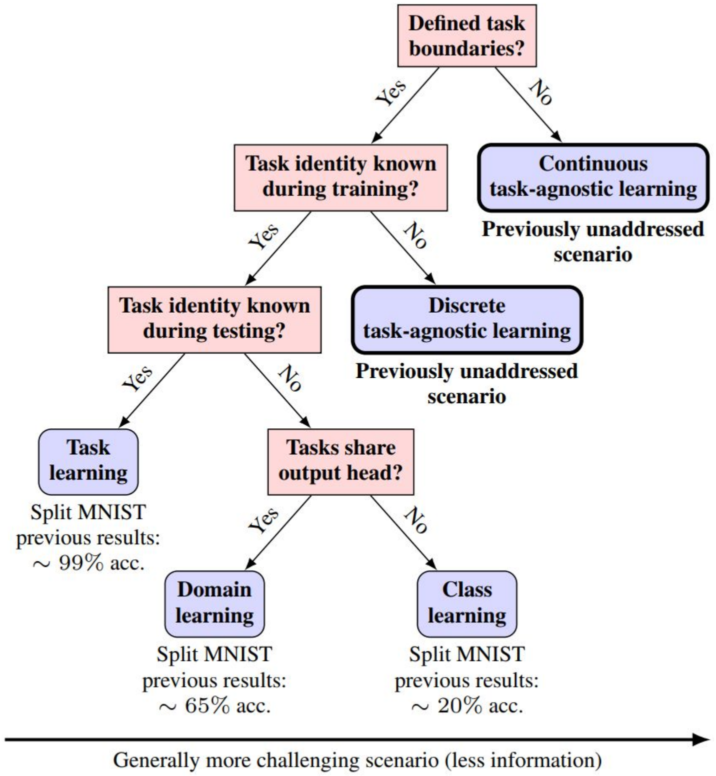
1. Data (oriented for different tasks) does not synchronously arrive to learning algorithms

2. Data pattern or tasks themself shifting along time

3. Detection of algorithms may or may not be aware of the shifting/varying of underline distribution

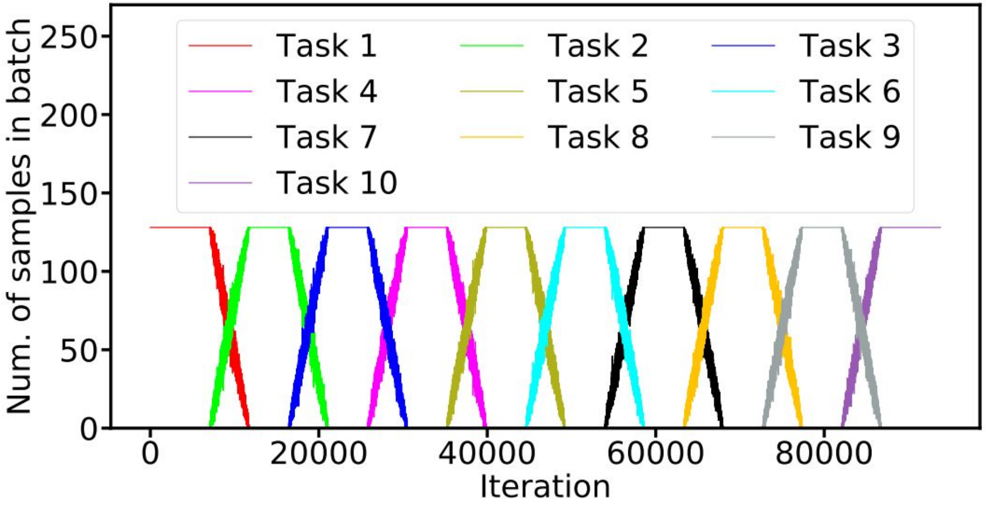
4. Catastrophic forgetting: vulnerability of deep neural network adapting to new data/task and forgetting past knowledge

Possible scenarios (with task information)



1. Are the boundaries between tasks well defined
2. Is the task identity known during training
3. Is the task identity known during inference (testing)
4. Should predictor take the number of tasks into account?
5. ….

online learning under task transitions



Task boundaries under typical situations:

1. The transition between different tasks themself occurs slowly over time
2. The data itself changes continuously towards a new distribution

Online Bayes as a solution

Key role in Bayes prediction:

$$p(\boldsymbol{\theta}|D_n) = \frac{p(D_n|\boldsymbol{\theta})\, p(\boldsymbol{\theta}|D_{n-1})}{p(D_n)}$$

Prevision estimation/posterior of model parameters serve as prior for new task/prediction.

Exact Bayesian inference is intractable (for most practical tasks):

1. Laplace approximation
2. Variational methods
3. Monte Carlo methods,
4. Assumed density filtering/Expectation propagation
5. …..

Exemplified Online Variational Bayes

Variational method:

$$\mathrm{KL}\left(q\left(\boldsymbol{\theta}|\phi\right)||p\left(\boldsymbol{\theta}|D\right)\right) = -\mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta}|\phi)}\left[\log\frac{p\left(\boldsymbol{\theta}|D\right)}{q\left(\boldsymbol{\theta}|\phi\right)}\right]$$

Advantages:

1. Last approximation can server for current approximation learning
2. Log-likelihood can be accumulated by Bayes rule
3. Bayesian Gradient Descent put link between the learning rate and uncertainty (stand deviation)

$$\begin{aligned}
\phi^* &= \arg\min_{\phi}\int q_n\left(\boldsymbol{\theta}|\phi\right)\log\frac{q_n\left(\boldsymbol{\theta}|\phi\right)}{p\left(\boldsymbol{\theta}|D_n\right)}d\boldsymbol{\theta} \\
&= \arg\min_{\phi}\int q_n\left(\boldsymbol{\theta}|\phi\right)\log\frac{q_n\left(\boldsymbol{\theta}|\phi\right)}{p\left(D_n|\boldsymbol{\theta}\right)q_{n-1}\left(\boldsymbol{\theta}\right)}d\boldsymbol{\theta} \\
&= \arg\min_{\phi}\mathbb{E}_{\boldsymbol{\theta}\sim q_n(\boldsymbol{\theta}|\phi)}\left[\log\left(q_n\left(\boldsymbol{\theta}|\phi\right)\right)\right. \\
&\quad \left. -\log\left(q_{n-1}\left(\boldsymbol{\theta}\right)\right) + \mathrm{L_n}\left(\boldsymbol{\theta}\right)\ \right],
\end{aligned}$$

Conclusions

Conclusions

Online machine learning is useful in different applications where data becomes available over time.

Some examples relating to our research:

- Adaptive control (e.g., under changing dynamics)
- Training over large datasets (e.g., in imitation learning)
- Reinforcement learning over a network of agents

Thanks for listening!