Online machine learning

Chris | Dong | Edvards | Hasan | Yang

Introduction

- Data becomes available sequentially
- New data incrementally updates the model
- Useful when:
  - The entire dataset it too large
  - Patterns emerge dynamically
  - Data is generated over time

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 selection
  - Decider chooses distribution of wealth over assets
  - Adversary chooses market returns
  - Decider learns to rebalance portfolio

Basic concepts

- The framework is game-theoretic and adversarial
- For each iteration
  a. The decider makes a choice
  b. A convex cost function is revealed
  c. The decider incurs a cost
  d. The decider make a new choice to minimise regret
- Regret is the difference between the incurred cost and the cost of the best decision in hindsight

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

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

First and second order methods

- Online gradient descent
  - Step in the direction of the gradient of the previous cost
  - If the new point is outside the underlying convex set, project it back within.
  - The regret is sublinear
  - But projection is burdensome
- Online Newton step
  - Approximates second derivative
  - Requires less iterations
  - But each step is costly
- Both algorithms require projection back into the convex set if they step out
- Projection is “expensive”

Regularisation

- Follow the leader (FTL)
  - At any point in time, use the optimal decision in hindsight
  - Simple strategy
  - Regret is linear in iterations
  - Very unstable, changing decision too often
- Regularised FTL (RFTL)
  - Adds a regularisation function
  - Gives asymptotically optimal regret bounds
  - Stabilises the prediction

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!

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 $f_t(\mathbf{x}_t)$.

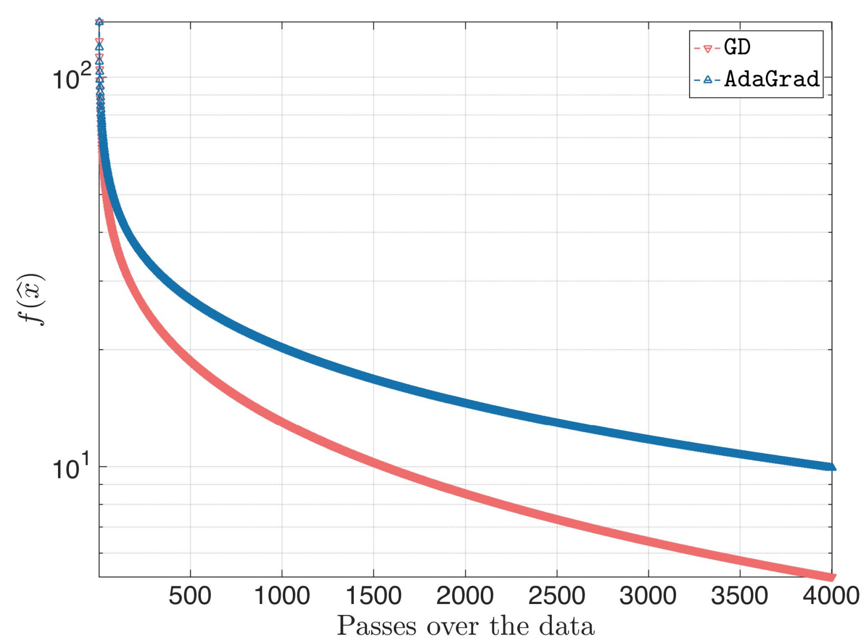
Update:

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

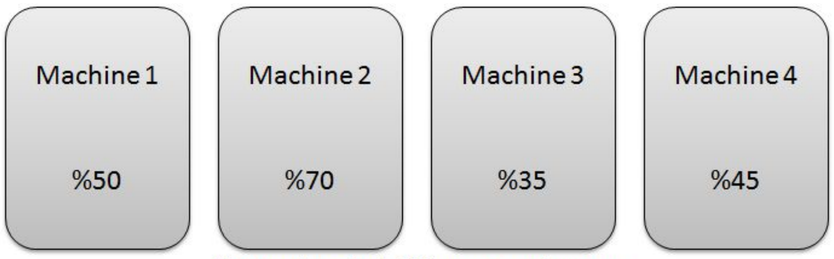
$$\mathbf{y}_{t+1} = \mathbf{x}_t - \eta G_t^{-1} \nabla_t$$

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

end for



Online decision-making Bandit & Reinforcement learning



Reward probabilities are unknown

- At each step *t=1,2,...,T,* a decision-maker
  - Observes the state,
  - Chooses an action from a given action set *A*,
  - Receives reward.
- Goal is to maximize the collected rewards.
- *Regret for decision-making:*

$$Regret_{\pi,T} := \mathbb{E}^{\pi^*}\{\sum_{t=1}^{T} r_t\} - \mathbb{E}^{\pi}\{\sum_{t=1}^{T} r_t\}$$

- *Regret for RL in MDPs:*

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

- *Can't be arbitrarily minimized due to some fundamental performance limits.*

A simple multi armed bandit algorithm:

- With some probability, explore the action space.
  - Use the feedback to construct an estimate of the actions' losses.
- Otherwise, use the estimates to select the optimum choice.
  - Suppose the estimates are the true historical costs.

Projection-free algorithms

- In many computational and learning scenarios the main bottleneck of optimization, both online and offline, is the computation of projections onto the underlying decision set.
- The conditional gradient(CG) method, or Frank-Wolfe algorithm, is a simple algorithm for minimizing a smooth convex function f over a convex set.

```
Input: step sizes {η_t ∈ (0,1), t ∈ [T]}, initial point x_1 ∈ K
for t = 1 to T do
    v_t ← arg min_{x∈K} {x^T ∇f(x_t)}.
    x_{t+1} ← x_t + η_t(v_t − x_t).
end for
```

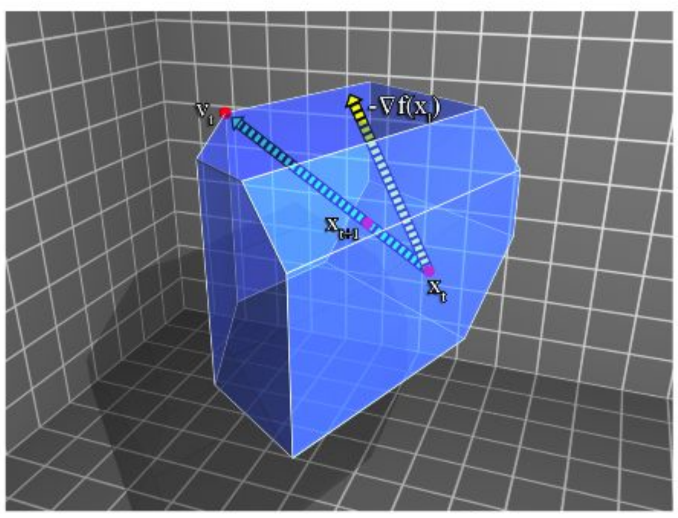
- Matrix completion problem:

$$\min_{X \in \mathbb{R}^{n\times m}} \frac{1}{2}\|X - M\|_{OB}^2$$
$$\text{s.t.} \quad \mathrm{rank}(X) \leq k.$$

- CG for matrix completion

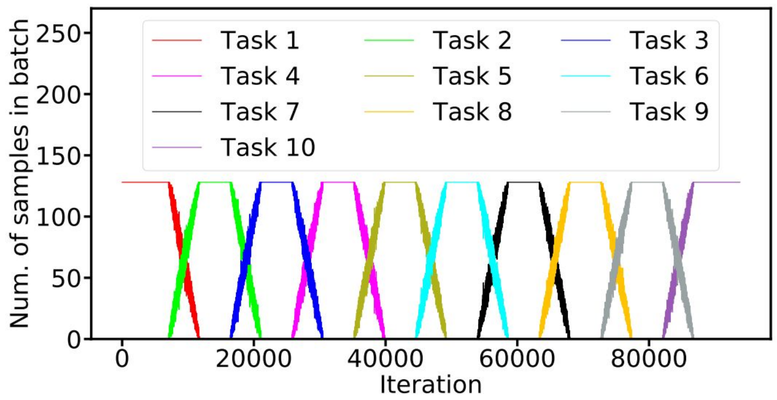
```
Let X^1 be an arbitrary matrix of trace k in K.
for t = 1 to T do
    v_t = √k · v_max(−∇_t).
    X^{t+1} = X^t + η_t(v_t v_t^T − X^t) for η_t ∈ (0,1).
end for
```

In the CG method, the update to the iterate $\mathbf{x}_t$ may not be in the direction of the gradient, as $\mathbf{v}_t$ is the result of linear optimization procedure.



Online Bayes learning

- Common continuous learning issues:
  - Data distribution is changing while learning (Catastrophic forgetting problem)
  - Multiple related tasks while no clear boundaries
  - Asynchronous data arrival
  - Reliability of gradient



- Bayes rule: encode past information into posterior, which is used as prior for future predictions

- Practical solutions for posteriors estimation:
  - Variational methods
  - Monte Carlo methods
  - Laplace/Mean-field approximations/ADF/EP
- Encode estimation belief into learning rate/speed:
  - Bayesian Gradient Descent

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

Conclusion

Online machine learning is useful in many 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