A pre-Weekend Talk on Online Learning
TGIF Talk Series
Puruushottam Kar
Outline

• Some Motivating Examples
  • Discovering customer preferences
  • Learning investment profiles
  • Detecting credit card fraud

• The Formal Online Learning Framework
  • Notion of regret
  • Formalization of motivating examples

• Simple Online Algorithms
  • Online classification, regression
  • Online ranking
  • Batch solvers for large scale learning problems

• Other “Feedback-based” Learning Frameworks
Some Motivating Examples
Why Online Learning can be Useful
The Cook’s Dilemma
Discovering Customer Preferences

Loss

2

1

1

0

2
Learning Investment Profiles

• $k$ assets $a_1, a_2, \ldots, a_k$ that give returns proportional to investment

• Asset $a_i$ gives back $r_i$ as return per dollar invested
  • If I invest $d_i$ in $a_i$ then total return is $\sum d_i r_i = d^T r$
  • Return profile $r$ depends on market forces, other investors and keeps changing

• I have corpus of $D$ that I decide to invest completely in these assets
  • Let $p_i$ decide proportion of investment in asset $a_i$, i.e. investment is $p_i D$

• Corpus at time $T$ becomes $D \prod_{t=1}^{T} \langle p^t, r^t \rangle$: reward to be maximized
Detecting Credit Card Fraud

• Classify credit card payments into \{+,-\}
  • Each payment $p \in \mathcal{P}$ is described by a vector $x_p \in \mathbb{R}^d$
  • Other problems such as branch prediction/churn prediction

• Linear classification model
  • Choose $w \in \mathbb{R}^d$ and classify $p$ as $\text{sign}(w^T x_p)$

• Online process; at each time $t$
  • A credit card payment $p_t$ is detected
  • We propose a linear classifier $w_t$ and classify $p_t$ as $\text{sign}(w_t^T x_{p_t})$
  • True status of payment $y_t$ is made known and our mistake (if any) is revealed

• Wish to minimize the number of mistakes made by us
  • Wish to propose a “good” sequence of $w$
The Formal Online Learning Framework
How we assess Online Learning Algorithms
The Online Learning Model

• An attempt to model an interactive and adaptive environment
  • We have a set of actions \( \mathcal{A} \)
  • Environment has a set of loss functions \( \mathcal{L} = \{ \ell : A \to \mathbb{R}_+ \} \)

• In each round \( t \)
  • We play some action \( a_t \in \mathcal{A} \)
  • Environment responds with a loss function \( \ell_t \in \mathcal{L} \)
  • We are forced to incur a loss \( \ell_t(a_t) \)
  • Environment can adapt to our actions (or even be adversarial)

• Our goal: minimize cumulative loss \( \sum_{t=1}^{T} \ell_t(a_t) \)
  • Can cumulative loss be brought down to zero : mostly no !
  • More reasonable measure of performance: single best action in hindsight
  • Regret: \( R_T := \sum_{t=1}^{T} \ell_t(a_t) - \min_{a \in \mathcal{A}} \sum_{t=1}^{T} \ell_t(a) \)
  • Why is this a suitable notion of performance ?
Motivating Examples Revisited

• Detecting customer preferences
  • Assume we can represent customer $c \in C$ as a vector $x_c \in \mathbb{R}^d$
  • Set of actions are linear functions predicting spice levels for that customer
    \[ \hat{s}_c = w^T x_c \]
  • Loss function given by squared difference between true and preferred spiciness
    \[ \ell_{\text{abs}}(w, x_c) = (\hat{s}_c - s_c)^2 \]
  • At time step $t$ customer $c_t$ comes and $\ell_t(w_t) = \ell_{\text{abs}}(w_t, x_{c_t})$
  • Goal: make customers as happy as the single best spice level

• Credit card fraud detection
  • Actions are the set of linear classifiers $\mathcal{W} = \{w \in \mathbb{R}^d\}$
  • Loss functions are mistake functions
    \[ \ell_{0/1}(w, x_p) = \mathbb{I}\{y_p w^T x_p < 0\} = \begin{cases} 1 & \text{if } y_p \neq \text{sign}(w^T x_p) \\ 0 & \text{otherwise} \end{cases} \]
    \[ \ell_t(w_t) = \ell_{0/1}(w_t, x_{p_t}) \]
  • Detection of credit card fraud might change buying profiles (adversarial)
  • Goal: make (almost) as few mistakes as single best classifier
Motivating Examples Revisited

- Learning investment profiles
  - Set of actions is the $d$-dimensional simplex $\mathcal{A} = \{ p \in \mathbb{R}^d, p \geq 0, \| p \|_1 = 1 \}$
  - Reward received at $t^{\text{th}}$ step is $\langle p^t, r^t \rangle$ where $r^t$ is the return given by market
  - Total reward (assume w.l.o.g. initial corpus is $D = 1$)
    \[ \prod_{t=1}^{T} \langle p_t, r_t \rangle = \exp \left( \sum_{t=1}^{T} \log \langle p_t, r_t \rangle \right) \]
  - Returns affected by investment, other market factors (adaptive, adversarial)
  - Can think of $\ell(p, r) = -\log \langle p, r \rangle$ as a negative reward or a loss
    \[ \ell_t(p_t) = -\log \langle p_t, r_t \rangle \]
  - Regret (equivalently) given by
    \[ \mathcal{R}_T = \sum_{t=1}^{T} \ell(p_t, r_t) - \min_{p \in \mathcal{A}} \sum_{t=1}^{T} \ell(p, r_t) \]
  - Goal: make as much profit as the single best investment profile in hindsight
Simple Online Algorithms

What makes online learning click?
Online Linear Classification

• Perceptron Algorithm
1. Start with $w_0 = 0$
2. Classify $o_t$ as $\text{sign}(w_{t-1}^T x_{o_t})$
3. If correct classification i.e. $y_t = \text{sign}(w_t^T x_{o_t})$, then let $w_t = w_{t-1}$
4. Else $w_t = w_{t-1} + y_t x_{o_t}$

• Loss function $\ell_{0/1}(w, o) = \mathbb{I}\{y_o w^T x_o < 0\}$ i.e. 1 iff $w$ misclassifies $o$
• If there exists a perfect linear separator $w^*$ such that $y_tw^*_T x_{o_t} \geq \gamma$, 
  $\mathcal{R}_T = \sum \ell_{0/1}(w_t, o_t) - \sum \ell_{0/1}(w^*, o_t) \leq \frac{1}{\gamma^2}$
• If there exists an imperfect separator $w^*$ such that $y_tw^*_T x_{o_t} \geq \gamma - \xi_t$, 
  $\mathcal{R}_T = \sum \ell_{0/1}(w_t, o_t) - \sum \ell_{0/1}(w^*, o_t) \leq \frac{1}{\gamma^2} + \frac{1}{\gamma} \sum \xi_t$
The Perceptron Algorithm in action

\[ 2\gamma \]
Online Regression

• The Perceptron Algorithm was (almost) a gradient descent algorithm
• Consider the loss function
  \[ \ell_{\text{hinge}}(w, x) = \max\{1 - yw^T x, 0\} \]
• \(\ell\) is a convex surrogate to the mistake function \(\ell_{0/1}(w, x) = \mathbb{I}\{yw^T x < 0\}\)
  \[ \ell_{\text{hinge}}(w, x) \geq \ell_{0/1}(w, x) \]

• When perceptron makes a mistake i.e. \(\ell_{0/1}(w, x) = 1\), we have
  \[ \nabla_w \ell_{\text{hinge}}(w, x) = -yx \]
• Thus the perceptron update step \(w_t = w_{t-1} + y_t x_{o_t}\) is a gradient step!
Online Regression via Online Gradient Descent

• Suppose we are taking actions $a_t \in A$ and receiving losses $\ell_t \in \mathcal{L}$
  • Assume that all loss function $\ell_t : A \rightarrow \mathbb{R}_+$ are convex and Lipchitz
  • Examples $\ell_t(a) = (a^T x_t - y_t)^2$, $\ell_t(a) = -\log(a^T x_t)$, $\ell_t(a) = [1 - y_t a^T x_t]_+

• Online Gradient Descent (for linear predictions problems)
  1. Start with $a_0 = 0$
  2. Receive object $x_t$ and predict value $a_{t-1}^T x_t$ for object $x_t$
  3. Receive loss function $\ell_t$ and update $a_t = a_{t-1} - \frac{1}{\sqrt{t}} \nabla_{a} \ell_t(a_{t-1})$
     • Some more work needed to ensure that $a_t \in A$ as well

• We can ensure that

$$R_T = \sum_{t=1}^{T} \ell_t(a_t) - \min_{a \in A} \sum_{t=1}^{T} \ell_t(a) \leq O(\sqrt{T})$$
Online Bipartite Ranking

- Documents $d_1, d_2, \ldots, d_t, \ldots$ arrive in a continuous stream to be ranked
- Each document is labelled either “relevant” (+) or “irrelevant” (-)
- Goal: somehow rank all relevant documents before irrelevant ones
- Method: assign relevance score $r: d_t \rightarrow r_t$ to document $d_t$ and sort

We incur loss for “swaps” $\ell_{\text{rank}}(r, d_t, d_{t'}) = \mathbb{I}\{(y_t - y_{t'})(r_t - r_{t'}) < 0\}$
- Minimize number of swaps $\sum_{t=1}^{T} \sum_{t'=1}^{T} \ell_{\text{rank}}(d_t, d_{t'})$
- Problem is equivalent to maximizing area under the ROC curve of TP/FP

Challenges
- No reference point: no “true” relevance score
- Need pairs of documents to learn a scoring function: get only singles
- Solution: keep (some) of the past points in a buffer to construct pairs on the fly
- Several interesting algorithmic and theoretical problems still open
Batch Solvers

• Statistical learning gets a batch of randomly chosen training examples
  \((x_1, y_1), \ldots, (x_n, y_n) \sim \mathcal{X} \times \mathcal{Y}\)

• We wish to learn a function \(f \in \mathcal{F}\) that does well on these examples
  \[
  \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f, x_i)
  \]
  where \(\ell: \mathcal{F} \times \mathcal{X} \to \mathbb{R}_+\) is a loss function (classification, regression etc)

• Statistical Learning Theory: such an \(f\) does well on unseen points as well!

• Solving “batch” problem may be infeasible: \(n \gg 1\), distributed storage etc.

• Solution: solve the online problem instead

• E.g. online gradient descent will solve for a \(f_i \in \mathcal{F}\) such that
  \[
  \sum_{i=1}^{n} \ell(f_i, x_i) \leq \min_{f \in \mathcal{F}} \sum_{i=1}^{n} \ell(f, x_i) + \mathcal{R}_n
  \]
  where \(\mathcal{R}_n = o(n)\)
Batch Solvers

• Thus we have an \( f_t \in \mathcal{F} \) such that
\[
\frac{1}{n} \sum_{i=1}^{n} \ell(f_t, x_i) \leq \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f, x_i) + \varepsilon
\]

where \( \varepsilon = \frac{\mathcal{R}_n}{n} = o(1) \)

• Online to batch conversion bounds
  • Argue for the performance of \( \hat{f} = \frac{1}{n} \sum_{i=1}^{n} f_i \) on random unseen points
  • Expected loss of \( \hat{f} \) on a random unseen point is bounded
\[
\mathbb{E}_x \left[ \ell(\hat{f}, x) \right] \leq \frac{\mathcal{R}_n}{n} + \mathcal{O} \left( \frac{1}{\sqrt{n}} \right)
\]

• Several batch solvers e.g. PEGASOS, MIDAS, LASVM use techniques such as Stochastic online gradient descent for large scale learning
Other Feedback based Learning Frameworks

• Two axes of variation: modelling of environment and feedback
  - Online Learning: some modelling of environment and full feedback
    • Losses are simple functions over linear models (can be made non linear though)
    • At each step the loss function itself is given to us: full information
    • Models are agnostic: no realizability assumptions are made

• Multi-armed Bandits: weak modelling of environment, weak feedback
  • Often no assumptions made on nature of loss function
  • Limited feedback: only loss value on played action made available
  • Contextual bandits try to model loss function but make realizability assumptions

• Reinforcement Learning: Strong modelling of environment, weak feedback
  • Environment modelled as a state space with adaptive stochastic transitions
  • Reward functions modeled as functions of state space and action
  • Limited feedback available: need to learn, state space as well as reward function