Online Learning Applications

Part 5: Online gradient descent

Batched supervised learning

We observe a dataset

$$\mathcal{D}_n := \{(x_i, y_i)\}_{i=1}^n \in \mathcal{X} \times \mathcal{Y}.$$

containing pairs of "labeled examples": features x_i, label y_i.
Typically $\mathcal{X} = \mathbb{R}^d$ (features are represented by vectors) and

- $\mathcal{Y} = \{0, 1\}$: binary classification
- $3 \leq |\mathcal{Y}| < \infty$: multi-class classification
- $\mathcal{Y} = \mathbb{R}$: regression

Goal: build a predictor $\hat{g}_n : \mathcal{X} \to \mathcal{Y}$ which is a function that depends on the data $\mathcal{D}_n$, such that for a new observation $(\hat{x}, \hat{y}) \notin \mathcal{D}_n$

$$\hat{g}_n(\hat{x}) \backsimeq \hat{y}.$$

Good predictor if it can **generalize from training examples**.

Examples

Image classification

Features : pixel values
Label : type of image
(classification)

House prices prediction

Features : information on the house
Label : selling price
(regression)

Mathematical formalization

Modelling assumption: $\mathcal{D}_n$ contains **i.i.d. samples** whose distribution is that of a random vector $(x, y) \sim \mathcal{P}$.

Goal: given a loss function $\ell : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, build a predictor with small risk

$$R(g) \coloneqq \mathbb{E}_{(x,y)\sim\mathcal{P}}[\ell(g(x), y)].$$

Empirical risk minimization

Given a class $\mathcal{G}$ of possible predictors, one can compute/approximate

$$\hat{g}_n^{\mathrm{ERM}} \in \arg\min_{g \in \mathcal{G}} \left[\frac{1}{n} \sum_{i=1}^{n} \ell(g(x_i), y_i) \right].$$

Supervised learning algorithms

Some of them can be related to an ERM:

- linear regression (Gauss, 1795)
- logistic regression (1950s)
- Support Vector Machines (1995)
- Neural Networks (1960s-80s, Deep Learning 2010s)

Example: linear regression

$\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}$.

Linear regression

$\hat{g}_n : \mathcal{X} \ni x \mapsto \langle x, \hat{\theta}_n \rangle$ where

$$\hat{\theta}_n \in \arg\min_{\theta \in \mathbb{R}^d} \sum_{i=1}^{n} (y_i - \langle x_i, \theta \rangle)^2.$$

Linked to ERM by choosing:

- $\mathcal{G}$: space of linear functions
- Squared error loss: $\ell(z, y) = (z - y)^2$

Example: logistic regression

$\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{-1, +1\}$ (binary classification).

Logistic regression

$\hat{g}_n : \mathcal{X} \ni x \mapsto \operatorname{sgn}\left(\langle x, \hat{\theta}_n \rangle\right)$ where

$$\hat{\theta}_n \in \operatorname*{arg\,min}_{\theta \in \mathbb{R}^d} \sum_{i=1}^{n} \ln\left(1 + e^{-y_i \cdot \langle x_i, \theta \rangle}\right).$$

Linked to ERM by choosing:

- $\mathcal{G}$: space of linear functions
- Logistic loss: $\ell(z, y) = \ln(1 + e^{-zy})$

Online learning vs supervised learning

Online learning is the process of answering a sequence of questions given (possibly partial) knowledge of the correct answers to previous questions and possibly additional available information

Online learning vs supervised learning

Online learning is the process of answering a sequence of questions given (possibly partial) knowledge of the correct answers to previous questions and possibly additional available information

- **Supervised learning**: predictions based on large database (batch). Predict the label of a new data point (e.g., from a test set)
- **Online learning: data is collected sequentially, we have to predict labels one-by-one, after which the true label is revealed**
  - Decisions/predictions can influence the data collection process, and are based on past observations
  - Collect data in a smart way in order to optimize some criterion (e.g., maximize some cumulated reward)

Online learning vs supervised learning

Online learning is the process of answering a sequence of questions given (possibly partial) knowledge of the correct answers to previous questions and possibly additional available information

- **Supervised learning**: predictions based on large database (batch). Predict the label of a new data point (e.g., from a test set)
- **Online learning: data is collected sequentially, we have to predict labels one-by-one, after which the true label is revealed**
  - Decisions/predictions can influence the data collection process, and are based on past observations
  - Collect data in a smart way in order to optimize some criterion (e.g., maximize some cumulated reward)

Examples:

- predict the value of a stock
- day-ahead prediction of electricity supply/demand
- predict behavior of user on a web page/app

Online learning: general framework

Online learning

At every time step $t = 1, 2, \dots T$,

1. Observe (features) $x_t \in \mathcal{X}$
2. Predict (label) $\hat{y}_t \in \mathcal{Y}$
3. Observe true label y_t and incur loss $\ell(y_t, \hat{y}_t)$

Goal: minimize cumulated loss $\sum_{t=1}^{T} \ell(y_t, \hat{y}_t)$

Online learning: general framework

Online learning

At every time step $t = 1, 2, \ldots T$,

1. Observe (features) $x_t \in \mathcal{X}$
2. Predict (label) $\hat{y}_t \in \mathcal{Y}$
3. Observe true label y_t and incur loss $\ell(y_t, \hat{y}_t)$

Goal: minimize cumulated loss $\sum_{t=1}^{T} \ell(y_t, \hat{y}_t)$

We compare the performance of the online algorithm to a suitable baseline.
Example:

- performance of best predictor in a family $\mathcal{G}$

Online convex optimization

Learning the best predictor online

Let $\mathcal{G}$ be a class of predictors

A particular online learning problem

At every time step $t = 1, 2, \dots T$,

1. Choose a predictor $g_t \in \mathcal{G}$
2. Observe (features) $x_t \in \mathcal{X}$ and predict $\hat{y}_t = g_t(x_t)$
3. Observe true label y_t and incur loss $\ell(y_t, \hat{y}_t)$

Goal: minimize regret

Regret: difference between the cumulative loss of the online algorithm and the cumulative loss of the best predictor in $\mathcal{G}$ up to time T:

$$R_T := \sum_{t=1}^{T} \ell(y_t, \hat{y}_t) - \min_{g \in \mathcal{G}} \sum_{t=1}^{T} \ell(y_t, g(x_t)).$$

Example: online logistic regression

$\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$ (can be converted to predictions in $\{-1, 1\}$).

- $\mathcal{G} := \{g(x) = \langle x, \theta \rangle, \theta \in \mathbb{R}^d\}$ is the class of linear predictors parametrized in θ
- The predictor g_t corresponds to parameters $\theta_t \in \mathbb{R}^d$: $g_t(x) = \langle x, \theta_t \rangle$
- ℓ is the logistic loss: $\ell(y_t, \hat{y}_t) := \ln(1 + \exp\{-y_t \langle \theta_t, x_t \rangle\})$

Example: online logistic regression

$\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$ (can be converted to predictions in $\{-1, 1\}$).

- $\mathcal{G} := \{g(x) = \langle x, \theta \rangle, \theta \in \mathbb{R}^d\}$ is the class of linear predictors parametrized in θ
- The predictor g_t corresponds to parameters $\theta_t \in \mathbb{R}^d$: $g_t(x) = \langle x, \theta_t \rangle$
- ℓ is the logistic loss: $\ell(y_t, \hat{y}_t) := \ln(1 + \exp\{-y_t \langle \theta_t, x_t \rangle\})$

At each time $t = 1, \ldots, T$,

1. Choose a vector $\theta_t \in \mathbb{R}^d$
2. Observe function $\ell_t(\theta) = \ln(1 + \exp\{-y_t \langle x_t, \theta \rangle\})$
3. Suffer loss $\ell_t(\theta_t)$

Example: online logistic regression

$\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$ (can be converted to predictions in $\{-1, 1\}$).

- $\mathcal{G} := \{g(x) = \langle x, \theta \rangle, \theta \in \mathbb{R}^d\}$ is the class of linear predictors parametrized in θ
- The predictor g_t corresponds to parameters $\theta_t \in \mathbb{R}^d$: $g_t(x) = \langle x, \theta_t \rangle$
- ℓ is the logistic loss: $\ell(y_t, \hat{y}_t) := \ln(1 + \exp\{-y_t \langle \theta_t, x_t \rangle\})$

At each time $t = 1, \ldots, T$,

1. Choose a vector $\theta_t \in \mathbb{R}^d$
2. Observe function $\ell_t(\theta) = \ln(1 + \exp\{-y_t \langle x_t, \theta \rangle\})$
3. Suffer loss $\ell_t(\theta_t)$

The regret reads as follows:

$$R_T = \sum_{t=1}^{T} \ln(1 + \exp\{-y_t \langle \theta_t, x_t \rangle\}) - \underbrace{\min_{\theta \in \mathbb{R}^d} \sum_{t=1}^{T} \ln(1 + \exp\{-y_t \langle \theta, x_t \rangle\})}_{\substack{\text{loss obtained by the best} \\ \text{logistic regression predictor} \\ \text{trained over the whole dataset}}}.$$

Online convex optimization

Online logistic regression fits the framework of **online convex optimization** introduced by Zinkevich [2003]

Online convex optimization (OCO)

Given a **convex set** $\mathcal{C}$,
at each time $t = 1, \dots, T$,

1. Choose $\theta_t \in \mathcal{C}$
2. Observe a **convex loss function** ℓ_t
3. Suffer loss $\ell_t(\theta_t)$

Goal: minimize

$$R_T = \sum_{t=1}^{T} \ell_t(\theta_t) - \underbrace{\min_{\theta \in \mathbb{R}^d} \sum_{t=1}^{T} \ell_t(\theta)}_{\substack{\text{loss obtained by the best} \\ \text{static choice of } \theta \text{ in hindsight}}}.$$

Recap: convex sets and convex function

Definition

A set $\mathcal{C} \subseteq \mathbb{R}^d$ is convex if, for any $x, y \in \mathcal{C}$ and $\alpha \in [0, 1]$, it holds $\alpha x + (1 - \alpha)y \in \mathcal{C}$.

We denote with $D = \max_{x,y} \|x - y\|_2$, the diameter of the set $\mathcal{C}$. This is a measure of how large is the set of possible decisions.

Definition

A function $\ell : \mathcal{C} \to \mathbb{R}$ is convex if for any $x, y \in \mathcal{C}$ and $\alpha \in [0, 1]$ it holds $\ell(\alpha x + (1 - \alpha y)) \leq \alpha \ell(x) + (1 - \alpha)\ell(y)$.

We assume that ℓ_t are differentiable and let $G = \max_{t \in \{1,\dots,T\}} \|\nabla \ell_t(\theta_t)\|_2$. G is a measure of the magnitude of the losses.

Recap: projection

Definition

We define as

$$\Pi_{\mathcal{C}}(\theta) = \arg\min_{\theta' \in \mathcal{C}} \|\theta - \theta'\|$$

the projection of θ on the convex set $\mathcal{C}$.

Theorem

Let $\theta \in \mathbb{R}^d$, and $\theta' = \Pi_{\mathcal{C}}(\theta)$. Then, for any $\hat{\theta} \in \mathcal{C}$, it holds

$$\|\theta - \hat{\theta}\| \geq \|\theta' - \hat{\theta}\|_2.$$

First algorithm for OCO: online gradient descent

Online gradient descent (OGD)

convex set $\mathcal{C} \subseteq \mathbb{R}^d$, number of rounds T, step size η, initial point $\theta_1 \in \mathcal{C}$
for $t = 1, 2, \ldots, T$ **do**
  take decision $\theta_t \in \mathcal{C}$
  observe loss function ℓ_t and suffer loss $\ell_t(\theta_t)$
  $\theta_{t+1} \leftarrow \Pi_{\mathcal{C}}(\theta_t - \eta \nabla \ell_t(\theta_t))$

Performance guarantees

Theorem

The online gradient descent algorithm with step size $\eta = \frac{D}{G\sqrt{T}}$ *achieves regret*

$$R_T \leq DG\sqrt{T}.$$

Performance guarantees

Theorem

The online gradient descent algorithm with step size $\eta = \frac{D}{G\sqrt{T}}$ *achieves regret*

$$R_T \leq DG\sqrt{T}.$$

Better regret guarantees are possible for "more regular functions" (e.g., smooth, strongly convex). See, e.g., Hazan [2022].

Proof.

Let θ^* be the best fixed decision in hindsight, namely $\theta^* \in \arg\min_{\theta\in\mathcal{C}} \sum_{t=1}^{T} \ell_t(\theta)$. Then,

$$\begin{aligned}
\|\theta_{t+1} - \theta^*\|^2 - \|\theta_t - \theta^*\|^2 &= \|\Pi_{\mathcal{C}}(\theta_t - \eta\nabla\ell_t(\theta_t)) - \theta^*\|^2 - \|\theta_t - \theta^*\|^2 \\
&\leq \|\theta_t - \eta\nabla\ell_t(\theta_t) - \theta^*\|^2 - \|\theta_t - \theta^*\|^2 \\
&= -2\eta\langle\nabla\ell_t(\theta_t), \theta_t - \theta^*\rangle + \eta^2\|\nabla\ell_t(\theta_t)\|^2 \\
&\leq -2\eta\left(\ell_t(\theta_t) - \ell_t(\theta^*)\right) + \eta^2\|\nabla\ell_t(\theta_t)\|^2,
\end{aligned}$$

where the first inequality comes from the previous theorem about projections and the second one by the convexity of ℓ_t. Indeed, an equivalent definition of convex for a differentiable function ℓ is that for any θ, θ'

$$\ell(\theta) \geq \ell(\theta') + \langle\nabla\ell(\theta'), \theta - \theta'\rangle.$$

Proof.

Then,

$$
\begin{aligned}
\sum_{t=1}^{T} (\ell_t(\theta_t) - \ell_t(\theta^*)) &\le \sum_{t=1}^{T} \left(\frac{1}{2\eta} \|\theta_t - \theta^*\|^2 - \frac{1}{2\eta} \|\theta_{t+1} - \theta^*\|^2 + \frac{\eta}{2} \|\nabla \ell_t(\theta_t)\|^2 \right) \\
&= \frac{1}{2\eta} \|\theta_1 - \theta^*\|^2 - \frac{1}{2\eta} \|\theta_{T+1} - \theta^*\|^2 + \sum_{t=1}^{T} \frac{\eta}{2} \|\nabla \ell_t(\theta_t)\|^2 \\
&\le \frac{1}{2\eta} D^2 + \frac{\eta}{2} G^2 T \\
&= DG\sqrt{T}/2 + DG\sqrt{T}/2 \\
&= DG\sqrt{T},
\end{aligned}
$$

where the first equality is obtained by telescoping the terms in the summation. □

Online gradient descent for the expert problem

We can apply online gradient descent to the expert problem:

- The convex set of actions is $\mathcal{C} := \Delta_A$
- The **expected** loss of $\theta \in \Delta_A$ is $\langle \ell_t, \theta_t \rangle$ (linear loss)
- The diameter is $O(1)$
- $G \leq \sqrt{K}$ (upper bound on the Euclidean norm of gradients)

Theorem

Online gradient descent applied to the expert problem provides regret $O(\sqrt{KT})$.

- Gradient descent provides suboptimal performance
- It is possible to recover optimal regret bound with **mirror descent** (a generalization of gradient descent)

Online gradient descent for the expert problem

We can apply online gradient descent to the expert problem:

- The convex set of actions is $\mathcal{C} := \Delta_A$
- The **expected** loss of $\theta \in \Delta_A$ is $\langle \ell_t, \theta_t \rangle$ (linear loss)
- The diameter is $O(1)$
- $G \leq \sqrt{K}$ (upper bound on the Euclidean norm of gradients)

Theorem

Online gradient descent applied to the expert problem provides regret $O(\sqrt{KT})$.

- Gradient descent provides suboptimal performance
- It is possible to recover optimal regret bound with **mirror descent** (a generalization of gradient descent) $\rightarrow$ not in this course

References

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning (ICML)*, pages 928–936, 2003.

Elad Hazan. *Introduction to online convex optimization*. MIT Press, 2022.

Francesco Orabona. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*, 2019.