Online Learning

Tong Zhang

Rutgers University

Topics

- Traditional Online Learning: Perceptron
- Online Convex Optimization
- Stochastic Gradient Descent

Online Learning Setup

Problem:

- We sequentially observe $(x_1, y_1), \ldots, (x_t, y_t), \ldots$.
- We are interested in making a prediction at each time based on the current information
- Result is observed and evaluated (suffer a loss).
- The performance is measured by aggregated loss

Online Prediction Algorithm

Start with an initial prediction rule $f_0(x_i)$.
Iterate $t = 1, 2, \ldots$

- At each time we observe x_t and make a prediction $f_{t-1}(x_t)$
- We observe the true outcome y_t and then compute a loss $\phi(f(x_t), y_t)$
- The online algorithm update the prediction rule using the new example and construct $f_t(x)$.

The total error of the method is

$$\sum_{t=1}^{T} \phi(f_{t-1}(x_t), y_t).$$

Want this error to be as small as possible.
Predict unknown future one step a time: similar to generalization error.

Regret Analysis

- $f_*(x)$: optimal prediction function from a class C (e.g. the class of linear classifiers)

$$f_*(\cdot) = \arg\min_{f \in C} \sum_{t=1}^{T} \phi(f(x_t), y_t).$$

  minimize the error after seeing all examples:
- Online regret bound of the learning algorithm is

$$\text{regret} = T^{-1} \sum_{t=1}^{T} [\phi(f_{t-1}(x_t), y_t) - \phi(f_*(x_t), y_t)]$$

  want regret as small as possible.

Perceptron algorithm

- Goal: find a linear classifier with small error.
- Let $w_0 = 0$
- For $t = 1, \ldots,$:
  - Observe x_t and predict $\mathrm{sign}(w_{t-1}^T x_t)$
  - Update:
    - if $w_{t-1}^T x_t y_t \leq 0$, then $w_t = w_{t-1} + x_t y_t$;
    - otherwise $w_t = w_{t-1}$
- Number of mistakes: the number of times $w^T x_t y_t \leq 0$.
- Question: how many mistakes the algorithm makes?
- Partial answer: if there is a large margin linear separator, then the perceptron algorithm makes finite number of mistakes

Perceptron mistake bound

- Consider w_* that separate the data: $w_*^T X_i Y_i > 0$.
- Define margin

$$\gamma = \frac{\min_i w_*^T X_i Y_i}{\|w_*\|_2 \sup_i \|X_i\|_2}$$

- The number of mistakes perceptron makes is at most γ^{-2}.

The key of Online Learning

General approach:

- Keep a compressed version of historic data (state)
- Make prediction based on the compressed information
- Update the state

Applications:

- sequential prediction and quick model updates
- optimization: stochastic gradient descent

Online Convex Optimization

Consider a convex set S and convex functions $\phi_t(\cdot)$; the goal is to pick parameters $w_t \in S$ based on observed information to minimize aggregated loss:

$$\sum_{t=1}^{T} \ell_t(w_{t-1}).$$

Iterate $t = 1, 2, \ldots$

- Learner picks $w_{t-1} \in S$
- Observe a loss $\ell_t(\cdot)$
- Learner suffers loss $\ell_t(w_{t-1})$

Online Convex Optimization

Consider a convex set S and convex functions $\phi_t(\cdot)$; the goal is to pick parameters $w_t \in S$ based on observed information to minimize aggregated loss:

$$\sum_{t=1}^{T} \ell_t(w_{t-1}).$$

Iterate $t = 1, 2, \ldots$

- Learner picks $w_{t-1} \in S$
- Observe a loss $\ell_t(\cdot)$
- Learner suffers loss $\ell_t(w_{t-1})$

Regret:

$$\text{regret} = T^{-1} \left[\sum_{t=1}^{T} \ell_t(w_{t-1}) - \min_{w \in S} \sum_{t=1}^{T} \ell_t(w) \right]$$

Example: regression

Observe (x_t, y_t) and define loss $\ell_t(w) = (w^T x_t - y_t)^2$. Equivalent to the following problem:
Iterate $t = 1, 2, \ldots$

- Learner picks w_{t-1}
- Observe data (x_t, y_t)
- Learner suffers loss $(w_{t-1}^T x_t - y_t)^2$

Goal is to minimize

$$\sum_{t=1}^{T} (w_{t-1}^T x_t - y_t)^2.$$

Online Gradient Descent

For simplicity, assume S is the whole space:

- State being kept: only the current weight vector w_{t-1}
- Update rule with new observation (x_t, y_t):

$$w_t = w_{t-1} - \eta_t \nabla \ell_t(w_{t-1})$$

Very similar to gradient descent for optimization problem.

Online Gradient Descent

For simplicity, assume S is the whole space:

- State being kept: only the current weight vector w_{t-1}
- Update rule with new observation (x_t, y_t):

$$w_t = w_{t-1} - \eta_t \nabla \ell_t(w_{t-1})$$

Very similar to gradient descent for optimization problem.

Question: how good the algorithm is (in terms of regret bound)?

Convergence

- $\ell_t(\cdot)$ are strongly convex functions:

$$\text{regret} = O(\ln T / T),$$

  with learning rate $\eta_t = O(1/t)$
- $\ell_t(c\dot{)}$ are convex but not strongly convex:

$$\text{regret} = O(1/\sqrt{T}),$$

  with learning rate $\eta_t = O(1/\sqrt{t})$

Convergence

- $\ell_t(\cdot)$ are strongly convex functions:

$$\text{regret} = O(\ln T / T),$$

  with learning rate $\eta_t = O(1/t)$
- $\ell_t(c)$ are convex but not strongly convex:

$$\text{regret} = O(1/\sqrt{T}),$$

  with learning rate $\eta_t = O(1/\sqrt{t})$

We cannot do line search: learning rate is important

Proximal Extension

Assume that each $\ell_t(w)$ can be decomposed:

$$\ell_t(w) = \phi_t(w) + g(w),$$

then we have update rule:

$$w_t = \arg\min_w Q_t(w)$$

with

$$Q_t(w) = \phi_t(w_{t-1}) + \nabla\phi_t(w_{t-1})^T(w - w_{t-1}) + \frac{1}{2\eta_t}\|w - w_{t-1}\|_2^2 + g(w).$$

Similar convergence behavior as the non-proximal version.

Example: L_1-regularization

$$\ell_t(w) = \phi_t(w) + \lambda \|w\|_1.$$

Then

$$Q_t(w) := \phi_t(w_{t-1}) + \nabla \phi_t(w_{t-1})^T (w - w_{t-1}) + \frac{1}{2\eta_t} \|w - w_{t-1}\|_2^2 + \lambda \|w\|_1.$$

Solution of min $Q_t(w)$ is

$$w_t = \text{trunc}(w_{t-1} - \eta_t \nabla \phi_t(w_{t-1}))$$

where

$$\begin{aligned} \text{trunc}([u_1, \ldots, u_d]) &= [\text{trunc}(u_j)]_j \\ \text{trunc}(u_j) &= \text{sign}(u_j)(|u_j| - \lambda \eta_k)_+ \end{aligned}$$

Called truncated gradient method (requires other tricks).
One problem: $\eta_k \to 0$ implies the truncation is small thus ineffective.

Dual Averaging

- Problem: truncated gradient doesn't produce truly sparse w_t due to small η_t
- Fix: dual averaging which keeps two state representations: parameter w_t and average gradient vector

$$\bar{g}_t = n^{-1} \sum_{i=1}^{t} \nabla \phi_i(w_{i-1})$$

- Modified algorithm: let

$$Q_t(w) := \bar{g}_t^T (w - w_0) + \frac{1}{2t\eta_t} \|w - w_0\|_2^2 + \lambda \|w\|_1$$

and solve for $w_t = \arg\min_w Q_t(w)$.

Dual Averaging

- Problem: truncated gradient doesn't produce truly sparse w_t due to small η_t
- Fix: dual averaging which keeps two state representations: parameter w_t and average gradient vector

$$\bar{g}_t = n^{-1} \sum_{i=1}^{t} \nabla \phi_i(w_{i-1})$$

- Modified algorithm: let

$$Q_t(w) := \bar{g}_t^T (w - w_0) + \frac{1}{2t\eta_t} \|w - w_0\|_2^2 + \lambda \|w\|_1$$

and solve for $w_t = \arg\min_w Q_t(w)$.

Note that $\eta_t \to 0$ but $t\eta_t$ may be constant:

- convergence behavior: similar to gradient descent
- advantage: sparse w_t for L_1
- disadvantage: keep a non-sparse g_t

Keeping Second Order Information

- Advantageous to keep more information than the current weight w_{t-1}.
  - example: dual averaging
- Keep second order information about Hessian.
- Diagonal Hessian approximation: advantageous when different features scale differently.

Example (AdaGrad):

$$Q_t := \phi_t(w_{t-1}) + \nabla\phi_t(w_{t-1})^T(w - w_{t-1}) + \frac{1}{2\eta_t}(w - w_{t-1})^T H(w - w_{t-1}),$$

where H is diagonal with approximation
$H_{jj} = \delta I + \sqrt{\sum_{i\le t}(\nabla_j \phi_i(w_{i-1}))^2}$.

Stochastic Gradient Descent

Using online gradient descent to solve optimization problem:

$$\hat{w} = \arg\min_w P(w) \qquad P(w) := n^{-1} \sum_{i=1}^{n} \phi_i(w^T x_i) + \lambda g(w).$$

SGD:

- at each step t, randomly draw i from $1 : n$
- obtain $\hat{w}_t$ from online update: with respect to $\ell_t(w) = \phi_i(w^T x_i) + \lambda g(w)$

Remarks:

- main tuning parameter: learning rate.
- convergence: one pass over data is n steps for SGD but only one step for gradient descent

Stochastic Gradient Descent with Averaging

- Last predictor: return $\hat{w}_T$
- Average predictor: return $T^{-1} \sum_{i=1}^{T} \hat{w}_i$ (have other variations)

Why averaging?

Stochastic Gradient Descent with Averaging

- Last predictor: return $\hat{w}_T$
- Average predictor: return $T^{-1} \sum_{i=1}^{T} \hat{w}_i$ (have other variations)

Why averaging?

Theoretical justification:

- Some analysis requires averaging: non-smooth functions
- Some analysis allow more flexible choice of learning rate

Practical observation (mine):

- λ large; function relatively smooth: $\hat{w}_T$ is stable and average hurts
- λ small; function non-smooth: $\hat{w}_T$ is unstable and average improves

Variations of SGD

- Different proximal functions
- Different learning rates
- Dual averaging versus non-dual averaging
- Averaging versus non-averaging
- Acceleration versus non-acceleration
  - Nesterov's accelerated gradient method can be applied to SGD with similar improvements
  - require smooth objective function (if non-smooth, need to smooth)

Comparison with Dual Coordinate Descent

- Dual averaging version of SGD is similar to the first iteration of dual coordinate descent with a different learning rate.
- SGD is often faster than dual coordinate descent in the first few passes over data (due to different learning rate)
- Dual coordinate descent is generally faster after the first few passes over data
- It is possible to fix dual coordinate descent to make it faster

Resources

Book and Survey:

- Prediction Learning and Games. N. Cesa-Bianchi and G. Lugosi. Cambridge university press, 2006.
- Online Learning and Online Convex Optimization". Shai Shalev-Shwartz. Foundations and Trends in Machine Learning, Volume 4, Issue 2, DOI: 10.1561/2200000018

Software:

- Pegasos: http://www.cs.huji.ac.il/∼shais/code/index.html
- VW: hunch.net/∼vw/
- SGD by Leon Bottou: http://leon.bottou.org/projects/sgd