First-Order Optimization Algorithms for Machine Learning

Online Learning

Summer 2020

The Question I Hate the Most…

- How much data do we need?

- A difficult if not impossible question to answer.

- My usual answer: “more is better”.
  - With the warning: “as long as the quality doesn’t suffer”.

- Another popular answer: “ten times the number of features”.

The Question I Hate the Most…

- Let's assume you have a new supervised learning application.
  - But you have no data.
- You have some way to collect IID samples.
  - So you have to decide how much data to collect.

- Since it's supervised learning, our goal is to minimize a test error:

$$\tilde{f}(w) = \mathbb{E}[f_i(w)] \quad \text{"test error"}$$

  - Expected loss over IID examples from the test distribution.
  - Here, $f_i(w)$ could be the squared error or some other loss.

Usual Approach: Collect Data then Optimize

- We want to minimize the test error (which we cannot compute):

$$\tilde{f}(w) = \mathbb{E}[f_i(w)] \quad \text{"test error"}$$

- We approximate this with training error over 'n' IID samples:

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w) \quad \text{"train error"}$$

- And we need to decide how large 'n' should be.

- We can find 'n' if we use stochastic gradient descent (SGD).

SGD Speed of Convergence (Training Error)

- "How much data" can be related to "how fast does SGD converge"?

- Assumptions:
  - 'f' is strongly-convex: $\nabla^2 f(w) \geq \mu I$
  - 'f' is strongly-smooth: $LI \geq \nabla^2 f(w)$
  - "Variance" of gradients is bounded: $\frac{1}{n}\sum_{i=1}^{n} \|\nabla f_i(w) - \nabla f(w)\|^2 \leq \sigma^2$

- If we use SGD under these assumptions (and suitable α_k):
  - $E[f(w^k)] - f^* = O(1/k)$, where f^* is training error of the global optimum.
  - Implies we need $k=O(1/\epsilon)$ iterations to have $f(w^k) - f^* \leq \epsilon$.

Training Error vs. Testing Error

- We don't care about training error, we want to minimize test error.
  - And our goal was to decide how many examples 'n' to collect.

- We considered SGD on collected data (Approach 1):
  - Choose a random training example i_k (among the 'n' training examples).
  - Perform the SGD step.

- Now consider SGD while collecting data (Approach 2):
  - Collect a new random example i_k (IID from the true distribution).
  - Perform the SGD step.

- Approach 1 uses unbiased estimates of training error gradient.
- Approach 2 uses unbiased estimates of test error gradient.

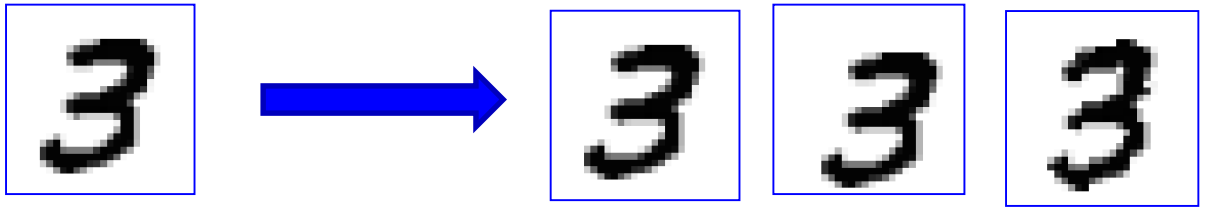
SGD Speed of Convergence (Test Error)

- With Approach 1, train error after 'k' iterations is O(1/k).
- With Approach 2, test error after 'k' iterations is O(1/k).
  - And we are using 1 new example on each iteration.
  - So with 'n' examples, this approach has test error of O(1/n).
  - And we need n=O(1/ϵ) training examples to get within ϵ of best test error.

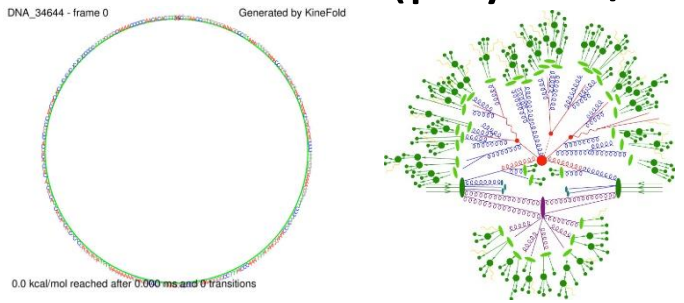
- Notice that there is no overfitting.
  - Approach 2 is doing SGD on the test error.
  - It's like doing SGD with n=∞, where train error = test error.

Scenarios where you can use Approach 2

- Here are some scenarios where you effectively have "n = ∞":
  - A dataset that is so large we cannot even go through it once (Gmail).
  - A function you want to minimize that you can't measure without noise.
  - You want to encourage invariance with a continuous set of transformation:
    - Infinite number of translations/rotations instead of a fixed number (or "dropout").



  - Learning from simulators with random numbers (physics/chem/bio):



http://kinefold.curie.fr/cgi-bin/form.pl
https://sciencenode.org/feature/sherpa-and-open-science-grid-predicting-emergence-jets.php

One-Pass SGD, Multi-Pass, and Caveats

- One-pass SGD:
  - If you already have an IID training set, you can simulate 'n' steps of Approach 2.
  - Go through your 'n' examples once, doing SGD step on each example.
    - Gets within O(1/n) of optimal test error.

- Under (ugly) assumptions, this "O(1/n) rate with 'n' examples" is unimprovable.
  - Even for methods that go through the dataset more than once or that minimize train error.

- In practice: one-pass SGD often doesn't work well.
  - It can't overfit, but it can definitely "underfit".
  - Doing multiple passes almost always helps.
  - Multiple passes can potentially improve constants in O(1/n) rate.
  - One-pass SGD is also very sensitive to the step-size.
  - Our "loss" might not be the error. For example, 0-1 error is approximated by logistic loss.
  - Some recent works have been exploring assumptions where O(1/n) is improvable.
  - So if you have $n=\infty$, but finite time: may be better to work with large-but-finite dataset.
    - "Optimize better on less data".

A Practical Answer to "How Much Data"?

- Whether we use one-pass SGD or minimize training error,

  E[test error of model fit on training set] – (best test error in class) = O(1/n).

  (under reasonable assumptions, and with parametric model)

- You rarely know the constant factor, but this gives some guidelines:
  - Adding more data helps more on small datasets than on large datasets.
    - Going from 10 training examples to 20, difference with best possible error gets cut in half.
      - If the best possible error is 15% you might go from 20% to 17.5% (this does **not** mean 20% to 10%).
    - Going from 110 training examples to 120, gap only goes down by ~10%.
    - Going from 1M training examples to 1M+10, you won't notice a change.
  - Doubling the data size cuts the error in half:
    - Going from 1M training to 2M training examples, gap gets cut in half.
    - If you double the data size and your test error doesn't improve, more data might not help.

(pause)

Online Learning

- Usual supervised learning setup:
  - Training phase:
    - Build a model 'w' based on IID training examples (x_t, y_t).
  - Testing phase:
    - Use the model to make predictions $\hat{y}_t$ on new IID testing examples $\hat{x}_t$.
    - Our "score" is the total difference between predictions $\hat{y}_t$ and true test labels y_t.

- In online learning there is no separate training/testing phase:
  - We receive a sequence of features x_t.
  - You make prediction $\hat{y}_t$ on each example x_t as it arrives.
    - You only get to see y_t after you've made prediction $\hat{y}_t$.
  - Our "score" is the total difference between predictions $\hat{y}_t$ and true labels y_t.
    - We need to predict well as we go (not just at the end).
    - You pay a penalty for having a bad model as you are learning.

Online Learning

- In online learning, we typically don't assume data is IID.
  - Often analyze a weaker notion of performance called "regret" (discussed later).

- Classic applications: online ads, spam filtering.

- A common variation is with bandit feedback:
  - We only observe loss y_t for action we choose.
    - You only observe whether they clicked on your ad, not which ads they would have clicked on.
  - Here we have an exploration vs. exploitation trade-off:
    - Should we explore by picking a y_t we don't know much about?
    - Should we exploit by picking a y_t that is likely to be clicked?

Follow the Leader

- An obvious strategy for online learning is follow the leader (FTL):
  - At time ‘t’, find the best model from the previous (t-1) examples.
  - Use this model to predict y_t.

- Problems:
  - It might be expensive to find the best model.
    - Have to solve an optimization problem over ‘t’ examples at time ‘t’.
  - It can perform very poorly.

Follow the Leader Counter-Example

- Consider this online convex optimization scenario:
  - At iteration ‘t’, we make a prediction w_t.
  - We then receive a convex function f_t and pay the penalty $f_t(w_t)$.
    - f_t could be the logistic loss on example ‘t’.

- In this setting, follow the leader (FTL) would choose:

$$w_t \in \operatorname{argmin}_w \sum_{i=1}^{t-1} f_i(w).$$

- The problem is convex but the performance can be arbitrarily bad…

Follow the Leader Counter Example

- Assume $x \in [-1,1]$ and:
  - $f_1(x_1) = (1/2)x$.
  - $f_2(x_2) = -x$.
  - $f_3(x_3) = x$.
  - $f_4(x_4) = -x$.
  - $f_5(x_5) = x$.
  - $f_6(x_6) = -x$.
  - $f_7(x_7) = x$.
  - …
- FTL objective:
  - $F_1(x_1)$ undefined
  - $F_2(x_2) = (1/2)x$.
  - $F_3(x_3) = -(1/2)x$.
  - $F_4(x_4) = (1/2)x$.
  - $F_5(x_5) = -(1/2)x$.
  - $F_6(x_6) = (1/2)x$.
  - $F_7(x_7) = -(1/2)x$.
  - …
- FTL predictions:
  - x_1 = (initial guess)
  - $x_2 = -1$ (worst possible)
  - $x_3 = 1$ (worst possible)
  - $x_4 = -1$ (worst possible)
  - $x_5 = 1$ (worst possible)
  - $x_6 = -1$ (worst possible)
  - $x_7 = 1$ (worst possible)
  - …

Regularized FTL and Regret

- Worst possible sequence:
  - {+1,-1,+1,-1,+1,-1,+1,-1,…}
- FTL produces the sequence:
  - {x0,-1,+1,-1,+1,-1,+1,-1,…}, which is close to the worst possible.
- Best possible sequence:
  - {0,+1,-1,+1,-1,+1,-,1,+1,…}
- Best sequence with a fixed prediction:
  - {0,0,0,0,0,0,0,0,…}

- We have no way to bound error compared to best sequence: could have adversary.

- We instead consider a weaker notion of "success" called regret:
  - How much worse is our total error than optimal fixed prediction at time 't'.
  - Note that fixed prediction might change with 't'.

Regret

- Online algorithms typically focus on minimizing regret:

$$\text{regret}(w_1, w_2, \ldots, w_T) = \sum_{t=1}^{T} f_t(w_t) - \min_{w \in C} \sum_{t=1}^{T} f_t(w)$$

  - "How much worse than the best *fixed* parameters in hindsight".
- Regret of an algorithm should be sublinear with 'T' like $O(\sqrt{T})$.
  - Difference with algorithm's strategy and best strategy decreases with time.

- We typically need assumptions to guarantee regret is sublinear.
  - Loss is bounded above, constraint set 'C' is bounded, and so on.
  - There is a huge/growing literature on regret minimization.

Regret Analysis for Online Gradient Descent

- One way to achieve sublinear regret: follow the "regularized" leader.
  - Add a suitable regularizer to "follow the leader".

- We'll show that "Approach 2" from earlier also has sublinear regret.
  - Apply projected stochastic gradient to the new example on each iteration.
    - Pretending like the potentially non-IID sample is an IID sample.
  - Much cheaper than "follow the regularized leader".
  - We'll assume strong convexity of each f_t and bounded gradient: $\|\nabla f_t(x)\| \leq G$
    - And constraint set 'C' is convex and compact.
  - Here the regret is O(log(T)) with suitable α_t, which is also optimal.
    - Even though data is not IID, only worse by logarithmic factor than best fixed strategy.
    - If the f_t are convex but not strongly convex it achieves $O(\sqrt{T})$, which is also optimal.

Regret Analysis for Online Gradient Descent

- Proof starts out looking similar usual projected(SGD) proofs:

optimal fixed 'w' at time 'T' (not 't')

$$\|w_{t+1} - w_*\|^2 = \|\text{Proj}_C[w_t - \alpha_t \nabla f_t(w_t)] - w_*\|^2 \quad \text{(def'n of algorithm)}$$

$$\leq \|(w_t - \alpha_t \nabla f_t(w_t)) - w_*\|^2 \quad \text{(projecting onto convex 'C' reduces distance)}$$

$$= \|(w_t - w_*) - \alpha_t \nabla f_t(w_t)\|^2$$

$$= \|w_t - w_*\|^2 - 2\alpha_t \nabla f_t(w_t)^T (w_t - w_*) + \alpha_t^2 \|\nabla f_t(w_t)\|^2$$

Re-arrange:

$$\nabla f_t(w_t)^T (w_t - w_*) \leq \frac{\|w_t - w_*\|^2 - \|w_{t+1} - w_*\|^2}{2\alpha_t} + \frac{\alpha_t}{2} \underbrace{\|\nabla f_t(w_t)\|^2}_{\leq G^2}$$

Regret Analysis for Online Gradient Descent

- Using strong-convexity and result from previous slide in regret:

$$\text{regret}(w_1, w_2, \ldots, w_T) = \sum_{t=1}^{T} \left[f_t(w_t) - f_t(w_*) \right]$$

$$\leq \sum_{t=1}^{T} \left[\nabla f_t(w_t)^T (w_t - w_*) - \frac{\mu}{2} \|w_t - w_*\|^2 \right] \quad \text{(strong convexity)}$$

$$\leq \sum_{t=1}^{T} \left[\frac{\|w_t - w_*\|^2 - \|w_{t+1} - w_*\|^2}{2\alpha_t} + \frac{\alpha_t}{2} G^2 - \frac{\mu}{2} \|w_t - w_*\|^2 \right] \quad \text{(previous slide)}$$

$$\leq \frac{1}{2} \sum_{t=1}^{T} \left(\frac{1}{\alpha_t} - \frac{1}{\alpha_{t-1}} - \mu \right) \|w_t - w_*\|^2 + \frac{G^2}{2} \sum_{t=1}^{T} \alpha_t \quad \text{(re-arrange, define } 1/\alpha_0 = 0, \text{ and use } \|w_{t+1} - w_*\|^2 \geq 0\text{)}$$

$$= 0 + \frac{G^2}{2} \sum_{t=1}^{T} \frac{1}{\mu t} \quad \text{(defining } \alpha_t = \frac{1}{\mu t}\text{)}$$

$$\leq \frac{G^2}{2\mu} (1 + \log(T)) \quad \left(\sum_{t=1}^{T} \frac{1}{t} \leq \sum_{t=1}^{\infty} \frac{1}{t} = 1 + \log(T) \right)$$

Discussion

- Optimal regret under same step-size (and similar proof) as SGD.
  - Many variations on algorithm/assumptions/analysis.

- So even with non-IID data, SGD is doing something reasonable.
  - Basically optimal in terms of regret.
  - Gives justification for using SGD on practical problems (real data is not IID).

- But keep in mind that for some problems,
  no method may do well so "regret" is a weak notion of success.

- And using $\alpha_t = \frac{1}{\mu t}$ still seems like a bad idea.

Summary

- "How much data do you need" question.
- Stochastic gradient descent on the test error (with one pass).
- O(1/n) error rate when you have 'n' training examples.
- Online learning: make predictions and pay penalties as you go.
  - Without assuming data is IID.
- Regret: how well did you do compare to best fixed strategy.
- Online gradient descent: optimal regret for online convex learning.

- Next time: can deep learning go faster than O(1/n)?