11/15/22

Lecture 14

fast, cheap descent algo.

- SGD
- coordinate descent
- stopping
- heavy ball

l13 – nov 10 TB posted
UV - part II posted
HW 5 P63 → postponed
HW6
PROJECT t.b. posted

Lecture IV: Training predictors, Part II

Marina Meilă
mmp@stat.washington.edu

Department of Statistics
University of Washington

October, 2021

Stochastic gradient methods

Examples: Linear classification with hinge loss, Perceptron

Accelerated gradient

No gradient methods: Coordinate descent

Stopping descent algorithms

Reading HTF Ch.: –, Murphy Ch.: 8.5.2-3 Stochastic gradient descent For more advanced treatment Nocedal and Wrigth.

Stochastic gradient descent (SGD) methods

[Optimization framework, minimize f over x]

SGD methods are the "cheap and slow [convergence]" methods which can however be very useful. One should not confuse "theoretically slow" with "slow in practice" and on some problems the former is true of the simpler methods but the latter is not. On other occasions, these methods perform well because they make fewer assumptions about the smoothness of the surface $f(x)$.

- Typical algorithm: steepest descent methods with diminishing step size.
- Assumption: the gradient $g^k = \nabla f(x^k)$ is computed with some error, that has 0 mean and bounded variance.
- This is the case for fitting of a model to data.

"fast" = # iterations (SGD: large) × (# ops/iteration (very small)) ~ log(tolerance)

tol : STOP when $e^k \le tol$

e^k: $f(x^k) - f(x^*)$, $\|x^k - x^*\|$

$\beta < 1$

$e^k \le \beta e^{k-1} \le \cdots \le \beta^k e^0$

Sufficient $\beta^k e^0 = tol$

$\Rightarrow k = \log_\beta\left(\frac{tol}{e^0}\right)$

Stochastic gradient for Machine Learning: idea

– on an example –

[ML framework, minimize J or $\hat{L}$ over parameters, (x, y) =data, f = predictor]

- Let $\mathcal{D}_n$ be an i.i.d sample of size n from an unknown distribution.
- Denote by $\hat{L}(\theta) = -\frac{1}{n} \ln P(y^{1:n}|x^{1:n}, \theta)$ the negative log-likelihood to be minimized.
- Because the sample is i.i.d., $\hat{L}(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \ln p(y^i|x^i, \theta)$. Since f is a sum, so will be the gradient:

$$\nabla \hat{L}(\theta) \;=\; -\frac{1}{n} \sum_{i=1}^{n} \frac{\frac{\partial p(y^i|x^i,\theta)}{\partial \theta}}{p(y^i|x^i, \theta)} \tag{1}$$

- If n is large (good from the statistical point of view) then the computation of (1) linear in n (very costly).
- The SGD idea is to set

$$d^k \;=\; \frac{\frac{\partial p(y^i|x^i,\theta)}{\partial \theta}}{p(y^i|x^i, \theta)} \tag{2}$$

  where (x^i, y^i) is randomly sampled from $\mathcal{D}_n$.
- Let P_X be the true data distribution and $\hat{P}_X$ the empirical distribution induced by the sample $\mathcal{D}_n$. Note that the direction d^k satisfies $E_{\hat{P}}[d^k] = \nabla \hat{L}$.

Objective in ML

$$J(\theta) = \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y^i, f_\theta(x^i))}_{\times n \text{ } f \text{ evaluations}} + \underbrace{\left(\lambda R(\theta)\right)}_{\text{constant}}$$

ignore — little influence on computing time

in G.D

$$\nabla J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\nabla_\theta L(y^i, f_\theta(x^i))}_{g^i(\theta)} = \frac{1}{n}\sum_i g^i(\theta) = E_{\hat{P}}[g(\theta)]$$

direction of descent

Idea of Stochastic G.D.

$\hat{P}$ = empirical distribution

- approximate

$$E_{\hat{P}}[g(\theta^k)] \approx \frac{1}{n'}\sum_{i \in \mathcal{D}'} g^i(\theta^k) \equiv d^k$$

sub sample from $\mathcal{D}$

$|\mathcal{D}'| = n'$

Remark

$$E[d^k] = \nabla J(\theta^k)$$

$\mathcal{D}'$

$$\mathrm{Var}[d^k] = \mathrm{Var}(g) \cdot \frac{1}{n'} \Rightarrow d^k = \nabla J(\theta^k) + \text{noise}$$

$=$

↑ population Var of g^i

- Special case $n' = 1$

⊕ computation of d^k faster

⊖ noise in d^k → stochastic

⇓

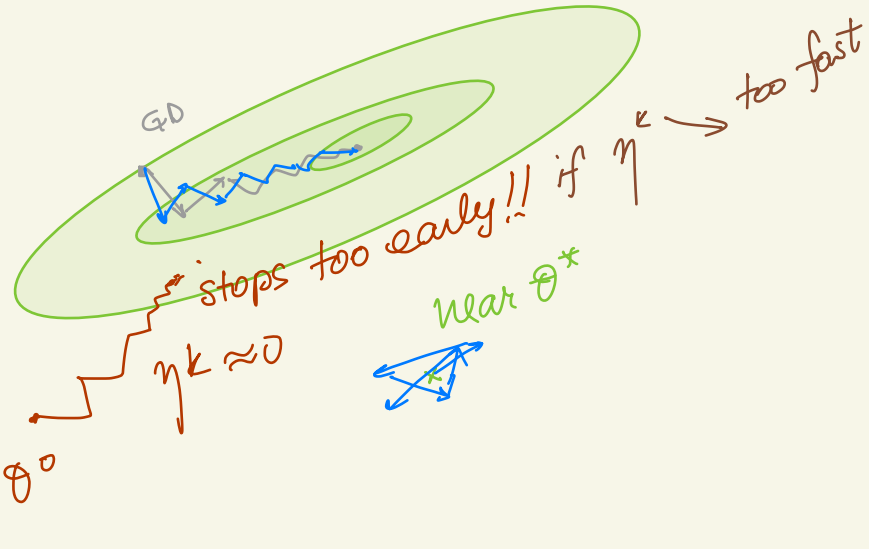
$d^k \not\to 0$!!

Fix 1. $\eta^k \to 0$

(two line search)

Fix 2. $n' \nearrow n$

Fix 3. average over k



[Optimization framework, minimize f over x]

This example points out that in Machine Learning

- the function f and its gradient ∇f are both expensive to evaluate, because they are are sums over the potentially large sample size n
- therefore, we want to avoid not only the $\nabla^2 f$ computation, but also the ∇f computation and even the line search which entails repeated evaluations of f
- on the other hand, evaluating a noisy version of the gradient is fast, because it involves one (or a few) samples

Thus, SGD takes many imprecise steps, instead of few but very computationally demanding precise steps. It remains to see if such a method can effectively find a minimum.

Convergence of SGD

Theory It has been proved under various technical conditions[1] that stochastic gradient converges to the true optimum if

- the step sizes η^k satisfy

$$\sum_k \eta^k = \infty, \quad \sum_k (\eta^k)^2 < \infty \quad \text{(the latter implies } \eta^k \to 0\text{)} \tag{3}$$

go as far as needed

cancel noise

- and the noise variance $Var[d^k]$ is bounded.

Practically η^k's should decrease like $\frac{1}{k}$. Note however that typically in practice the decrease needs to be very slow, almost constant e.g $\frac{1}{b+k/c}$ with b, c large numbers.

↑ ≈ constant η^k for small k

[1] These results are best known under the name of Robbins-Munro theory of stochastic approximation.

Variations and practicalities

- SGD with a subsample instead of single point ("old-fashioned" SGD). At each step k, take a subsample $\mathcal{D}' \subset \mathcal{D}$ with size $|\mathcal{D}'| = n' \ll n$, compute d^k as $\frac{1}{n'} \sum_{i \in \mathcal{D}'} \nabla L(y^i, \hat{y^i})$. $\mathcal{D}'$ must be a different subset at each time.
- n' can be varied as the training progresses towards the optimum, from n' small at the start, to approach the optimum fast, to n' large near the end, to reduce variance.
- When to use "modern" SGD and when to use "old-fashioned"? Rule of thumb:
  - If you know that your function is λ-strongly convex, then use the modern algorithm with a fixed K derived from λ.
  - Otherwise, use old-fashioned.
- If the sample $\mathcal{D}$ is truly iid (if it is true e.g. that example i and $i+1$ are independent) then picking a random i can be replaced by chosing i sequentially
- On-line learning and streaming data Remarkably, SGD can naturally handle **on-line learning**, i.e. situations where data come one by one, and are not stored but discarded, after being used immediately to update the parameters.
  In on-line learning, $(x^i, y^i) \sim P_{XY}$, we would be optimizing the expected loss $L = E_P[-\ln p(Y|X, \theta)]$, and d^i would satify $E_P[d^i] = \nabla L$. Showing that the variance of d^k is bounded is no more difficult (or easy than in the finite sample case).
- Stochastic gradient and analog techniques are widely used in machine learning: training of neural networks, reinforcement learning (the TD-λ and Q-learning procedures are stochastic gradient methods), speedup of boosting.

On-line learning

- at time t, (x^t, y^t) observed $\sim P_{XY}$
  - use it to update θ^t
  - discard it

$\Rightarrow g^t(\theta^t) = \nabla L(y^t, f_{\theta^t}(x^t)) = d^t$

$\theta^{t+1} \leftarrow \theta^t - \eta^t g^t(\theta^t)$ $\leftarrow n' = 1$

same for all t

$E[d^t] = E_{P_{XY}}[\nabla L(Y, f_{\theta^t}(X))] \Rightarrow$ SGD for P_{XY} true distribution

$Var[d^t] = Var_{P_{XY}}[$ —"— $]$

assume bounded

Modern results

LTFP (Bach) Ch 5 end: table of results

[ML framework, minimize J or $\hat{L}$ over parameters, (x, y) =data, f = predictor]

- Handling constraints If there are constraints $\theta \in \mathcal{A}$, we additionally assume that the *projection* $\Pi_{\mathcal{A}}\theta = \operatorname{argmin}_{\theta' \in \mathcal{A}} ||\theta - \theta'||$ can be computed efficiently (e.g. $\theta \succeq 0$).
- State of the art algorithms are very simple

Stochastic Gradient Descent (SGD)

Input $\lambda \leq \lambda_{mir}(\nabla^2 f(x^*))$, $c > 1/2$ a constant giving the step-size, [optional $\alpha \in (0, 1]$, K=total number steps]

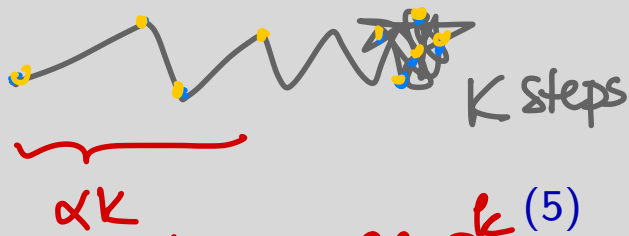
for $k = 1, 2, \ldots K$

1. get d^k: $\eta^i = 1$
   - 1.1 sample a point x^i at random from $\mathcal{D}$ OR pick x^i sequentially from a random permutation of $\mathcal{D}$
   - 1.2 compute $d^k = \nabla f_\theta(x^i)$
2. update θ: η^k

$$\theta^{k+1} \leftarrow \theta^k - \frac{c}{\lambda k} d^k \quad (4)$$

$tol^2 = \frac{ct}{\lambda^2 K^2} \Rightarrow K \sim \frac{1}{tol}$

average $\theta^k \Rightarrow \bar{\theta}$



3. if $k > (1 - \alpha)K$ accumulate $\bar{\theta} \rightarrow \bar{\theta} + \theta^k$

Average

$$\bar{\theta} \leftarrow \frac{\bar{\theta}}{\alpha K} \quad (5)$$

don't average θ^k

Output $\bar{\theta}$ (or optionally θ^K)

$$E\left[\left(J(\bar{\theta}^K) - J(\theta^*)\right)^2\right] \leq \frac{ct}{\lambda^2 K^2}$$

Remarks

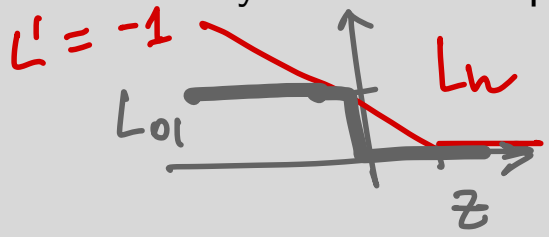
- steps proportional to $1/k$ and averages the last α fraction of steps.
- Empirically it was observed that no averaging, i.e. θ^K itself, has also good convergence properties, but (as expected) larger variance than $\bar{\theta}$.

Example: Linear classification with hinge loss

≈ SVM

[ML framework, minimize J or $\hat{L}$ over parameters, (x, y) =data, f = predictor]

The following example is a classic in statistical learning. We will examine it in two formulation. The first is an example of a problem where λ is known, and the SGD theory from above applies.



Problem setting

- $y \in \{\pm 1\}$ (binary classification)
- We fit the linear classifier

$$f(x) = w^T x \tag{6}$$

- Loss function = **hinge loss**

$$L_h(y, f(x)) = \begin{cases} 0 & \text{if } yf(x) \geq 1 \\ 1 - yf(x) & \text{if } yf(x) < 1 \end{cases} = [yf(x) - 1]_- \tag{7}$$

Define **margin** of example x

$$z = yf(x) \tag{8}$$

Under L_h an error is penalized linearly by how far $f(x)$ is in the "wrong direction" to which we add a penalty even for correctly classified examples if the margin $yf(x)$ is below 1.

- using the hinge loss.
- Simplifying assumption (for now, will remove it when we study SVM): the data $\mathcal{D} = \{(x^i, y^i)\}_{i=1:n}$ are **linearly separable**, i.e. there exists a w^* that classifies the sample with no error. Note that in general this w^* is not unique.

Linear Support Vector Machine SGD training algorithm

The optimization problem is a regularized one:

$$J(w) = \frac{1}{n}\sum_i L_h(y^i, w^T x^i) + \frac{\lambda}{2}||w||^2 \tag{9}$$

with $\lambda > 0$ a regularization parameter chosen by the user. Note that the non-quadratic loss term is linear (with unknown slope at w^*) and therefore the function $J(w)$ is by definition λ-strongly convex.

The stochastic part of the gradient is

$$\frac{\partial L_h}{\partial w} = \begin{cases} -y^i x^i & \text{if } i \text{ ``error''} \\ 0 & \text{if } i \text{ ``correct''} \end{cases} \tag{10}$$

where "correct" means that $y^i f(x^i) > 1$.

SGD for Linear SVM

Initialize with $w^0 = 0$, $\bar{w} = 0$

Iterate for $k = 1, 2, \ldots K$

1. Pick the next i in $1 : n$
2. compute direction $d^k = \lambda w^k - \mathbf{1}_{[i\text{ ``error''}]} y^i x^i$
3. update

$$w^{k+1} = w^k - \frac{c}{\lambda k}\left(\lambda w^k - \mathbf{1}_{[i\text{ ``error''}]} y^i x^i\right) = w^k(1 - c/k) + \frac{c}{\lambda k} y^i x^i \mathbf{1}_{[i\text{ ``error''}]} \tag{11}$$

4. average $\bar{w} \leftarrow \bar{w} + w^{k+1}$

Output $\bar{w}/K$

Same as λ on page 8

Accelerated gradient: the "heavy ball" method

Hot topic!!

$$x^{k+1} = x^k - \eta^k d^k + \gamma^k (x^k - x^{k-1}) \tag{14}$$

- Applies to both standard and stochastic gradient methods, i.e.

$$d^k = \begin{cases} \nabla f(x^k) & \text{gradient descent} \\ \text{noisy gradient} & \text{SGD} \\ \nabla f(x^k + \delta^k(x^k - x^{k-1})) & \text{extragradient methods} \end{cases} \tag{15}$$

- Setting the parameters
  - In the extragradient[3] methods, $\eta^k, \delta^k, \gamma^k$ are obtained by search (or knowledge about M, m)
  - For other methods fix $\gamma^k = \gamma \in (0.5, 1]$ OR use smaller γ early in the training and increase it to near 1 when the steps become smaller.
- More intuition
  - for ill conditioned problems $M \ll m$, the heavy ball "accumulates" the components of the step in the correct direction
  - for SGD, the heavy ball approximates the exact gradient

Effects
1. if d^k noisy $\Rightarrow$ averaging
2. on plateau $\Rightarrow$ accelerates

[3] Nesterov's "optimal", FISTA