Adaptive Gradient Methods
AdaGrad / Adam

Machine Learning for Big Data
CSE547/STAT548, University of Washington
Sham Kakade

Announcements:

• HW3 posted
  – Dual coordinate ascent
  – (some review of SGD and random features)
• Projects: the term end is approaching!

• Today:
  – Review: adaptive gradient methods
  – Today: momentum; parallelization
Review

Curvature approximation:

• One idea:

  \[ \nabla^2 \hat{L}(w) \approx \frac{1}{t} \sum_{t} g_t(w) g_t(w)^T \]

  where \( g_t(w) \) is the gradient of the \( t \)-th data point

• Many ideas try to use this approximation
  – Quasi-Newton methods, Gauss newton methods
  – Ellipsoid method (sort of)
Mahalanobis Regret Bounds

\[ w^{(t+1)} = \arg \min_{w \in \mathcal{W}} \| w - (w^{(t)} - \eta A^{-1} g_t) \|_A^2 \]

- What \( A \) to choose?
  - Regret bound now:
  \[
  \sum_{t=1}^{T} \left( \ell_t(w^{(t)}) - \ell_t(w^*) \right) \leq \frac{1}{2\eta} \| w^{(1)} - w^* \|_A^2 + \frac{\eta}{2} \sum_{t=1}^{T} \| g_t \|_{A^{-1}}^2
  \]

- What if we minimize upper bound on regret w.r.t. \( A \) in hindsight?
  \[
  \min_A \sum_{t=1}^{T} g_t^T A^{-1} g_t
  \]

Mahalanobis Regret Minimization

- Objective:
  \[
  \min_A \sum_{t=1}^{T} g_t^T A^{-1} g_t \quad \text{subject to } A \succeq 0, \text{tr}(A) \leq C
  \]

- Solution:
  \[
  A = c \left( \sum_{t=1}^{T} g_t g_t^T \right)^{\frac{1}{2}}
  \]

For proof, see Appendix E, Lemma 15 of Duchi et al. 2011. Uses “trace trick” and Lagrangian.

- \( A \) defines the norm of the metric space we should be operating in.
AdaGrad Algorithm

- At time $t$, estimate optimal (sub)gradient modification $A$ by

$A_t = \left( \sum_{\tau=1}^{t} g_{\tau} g_{\tau}^T \right)^{\frac{1}{2}}$

- For $d$ large, $A_t$ is computationally intensive to compute. Instead,

$A_t \approx \left( \begin{array}{ccc}
A_{11} & \cdots & 0 \\
0 & \ddots & \vdots \\
0 & \cdots & A_{dd}
\end{array} \right)$

$(A_t)_{i,i} = \sqrt{ \frac{\sum_{\tau=1}^{t} g_{\tau,i}^2}{t} }$

- Then, algorithm is a simple modification of normal updates:

$w^{(t+1)} = \arg \min_{w \in \mathcal{W}} ||w - (w^{(t)} - \eta \text{diag}(A_t)^{-1} g_t)||^2_{\text{diag}(A_t)}$

AdaGrad in Euclidean Space

- For $\mathcal{W} = \mathbb{R}^d$,

- For each feature dimension,

$w^{(t+1)}_i \leftarrow w^{(t)}_i - \eta_{t,i} g_{t,i}$

where

$\eta_{t,i} = \frac{\eta}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}^2}}$

- That is,

$w^{(t+1)}_i \leftarrow w^{(t)}_i - \frac{\eta}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t,i}$

- Each feature dimension has its own learning rate!
  - Adapts with $t$
  - Takes geometry of the past observations into account
  - Primary role of $\eta$ is determining rate the first time a feature is encountered

©Sham Kakade 2017
AdaGrad Theoretical Guarantees

- AdaGrad regret bound:
  \[ \sum_{t=1}^{T} (\ell_t(w^{(t)}) - \ell_t(w^*)) \leq 2R_\infty \sum_{i=1}^{d} \|g_{1:T,i}\|_2 \]
  - In stochastic setting:
  \[ \mathbb{E} \left[ \ell \left( \frac{1}{T} \sum_{t=1}^{T} w^{(t)} \right) \right] - \ell(w^*) \leq \frac{2R_\infty}{T} \sum_{i=1}^{d} \mathbb{E} [\|g_{1:T,i}\|_2] \]

- This is used in practice.
- Many cool examples. Let’s just examine one…

AdaGrad Theoretical Example

- Expect to out-perform when gradient vectors are \textit{sparse}
- SVM hinge loss example:
  \[ \ell_t(w) = [1 - y_t \langle \mathbf{x}^t, w \rangle]_+ \]
  \[ \mathbf{x}^t \in \{-1, 0, 1\}^d \]
- If \( x_j^t \neq 0 \) with probability \( \propto j^{-\alpha}, \quad \alpha > 1 \)

\[ \mathbb{E} \left[ \ell \left( \frac{1}{T} \sum_{t=1}^{T} w^{(t)} \right) \right] - \ell(w^*) = \mathcal{O} \left( \frac{\|w^*\|_\infty}{\sqrt{T}} \cdot \max \{ \log d, d^{1-\alpha/2} \} \right) \]

- (sort of) previously bound:
  \[ \mathbb{E} \left[ \ell \left( \frac{1}{T} \sum_{t=1}^{T} w^{(t)} \right) \right] - \ell(w^*) = \mathcal{O} \left( \frac{\|w^*\|_\infty}{\sqrt{T}} \cdot \sqrt{d} \right) \]
Today: Adam, Momentum, Comparisons

ADAM

• Like AdaGrad but with “forgetting”
• The algo has component-wise updates

Adam update rule consists of the following steps:

- Compute gradient $g_t$ at current time $t$
- Update biased first moment estimate
  $$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
- Update biased second raw moment estimate
  $$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$
- Compute bias-corrected first moment estimate
  $$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
- Compute bias-corrected second raw moment estimate
  $$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$
- Update parameters
  $$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$
Momentum Algorithm

- (Polyak 1964) The Heavy Ball method
- Two step procedure:
  \[ p_k = -\nabla f(x_k) + \beta_k p_{k-1} \]
  \[ x_{k+1} = x_k + \alpha_k p_k \]
  - Theory: asymptotically, it replaces condition number $\kappa$ with $\sqrt{\kappa}$.
  - Practice: Used with stochastic gradients. The results are mixed (both in the exact and stochastic case).

Nesterov’s Acceleration Algorithm

- (Nesterov 1983) Momentum done right:
- Two step procedure:
  \[ y_{k+1} \leftarrow x_k + \beta_k (x_k - x_{k-1}) \]
  \[ x_{k+1} \leftarrow y_{k+1} - \frac{1}{L} \nabla f(y_{k+1}) \]
  - Theory: It replaces condition number $\kappa$ with $\sqrt{\kappa}$.
  - Practice: We need a stochastic variant. (It’s “great” in the deterministic case)
Comparisons: MNIST, Sigmoid 100 layer

Comparisons: MNIST, Tanh 100 layer
Comparisons: Sigmoid, ReLu, Sigmoid

Curvature adaptive methods can (in principle and in practice) speed up the optimization
  - For exact gradient methods, they are widely used

With regards to SGD, the empirical results are more mixed.

Scalar learning rate case: In practice, we often need to turn the learning rate down. What is the “right” way to do this?
  - Sadly, there isn’t a clear “universal” picture in the convex case (1/T, 1/Root(t), constant, etc depending on the setting)
  - So how do we expect reasonable adaptive algorithms in the non-convex case?

Using “matrix” valued curvature: Often use diagonal scalings
  - The ‘choice’ is problem dependent (scale subsets of coordinates/nodes jointly, scale individual coordinates, etc)
  - How to effectively turn down learning rates?

Can we get a clearer picture?
Acknowledgments

