Case Study 1: Estimating Click Probabilities

Adaptive Gradient Methods

AdaGrad / Adam

Machine Learning for Big Data
CSE547/STAT548, University of Washington
Sham Kakade
The Problem with GD (and SGD)
Adaptive Gradient Methods: Convex Case

• What we want?

• Newton’s method:

\[ w \leftarrow w - [\nabla^2 L(w)]^{-1} \nabla L(w) \]

• Why is this a good idea?
  – Guarantees?
  – Stepsize?

• Related ideas:
  – Conjugate Gradient/Acceleration:
  – L-BFGS
  – Quasi-Newton methods
Adaptive Gradient Methods: Non-Cvx Case

• What do we want?
  – Hessian may not be PSD, so is Newton’s method a descent method?

• Other ideas:
  – Natural Gradient methods:

  – Curvature adaptive:
    • Adagrad, AdaDelta, RMS prop, ADAM, l-BFGS, heavy ball gradient, momentum
  – Noise injection:
    • Simulated annealing, dropout, Langevin methods

• Caveats:
  – Batch methods may be poor: “On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima”
Natural Gradient Idea

• Probabilistic models and maximum likelihood estimation:
  \[ \hat{L}(w) = -\log Pr(\text{data}|w) \]

• True likelihood function:
  \[ L(w) = -E_{z \sim D} \log Pr(z|w) \]
  where \( z \) is sampled form the underlying data distribution \( D \).

• Suppose the model is correct, i.e. \( z \sim Pr(z|w^*) \) for some \( w^* \)
  – Let’s look at the Hessian at \( w^* \)
    \[
    \nabla^2 L(w^*) = \mathbb{E}_{z \sim Pr(z|w^*)}[-\nabla^2 \log Pr(z|w^*)] \\
    = \mathbb{E}_{z \sim Pr(z|w^*)}[\nabla \log Pr(z|w^*)(\nabla \log Pr(z|w^*))^T]
    \]

• How do we approximate the Hessian at \( w \)?
Fisher Information Matrix

- Define the Fisher matrix:

\[ F(w) := \mathbb{E}_{z \sim \text{Pr}(z|w)}[\nabla \log \text{Pr}(z|w)(\nabla \log \text{Pr}(z|w))^\top] \]

- If the model is correct and if \( w \rightarrow w^* \), then \( F(w) \rightarrow F(w^*) \)

- Natural Gradient: Use the update rule:

\[ w \leftarrow w - [F(w)]^{-1} \nabla L(w) \]

- Empirically, use \( L^\wedge(w) \) and

\[ \hat{F}(w) := \frac{1}{t} \sum_t g_t(w)g_t(w)^\top \]

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

• One idea:

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

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)
Motivating AdaGrad  (Duchi, Hazan, Singer 2011)

• Assuming $\mathbf{w} \in \mathbb{R}^d$, standard stochastic (sub)gradient descent updates are of the form:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} - \eta_t g_{t,i}$$

• Should all features share the same learning rate?

• Motivating AdaGrad (Duchi, Hazan, Singer 2011):
  Often have high-dimensional feature spaces
  – Many features are irrelevant
  – Rare features are often very informative

• Adagrad provides a feature-specific adaptive learning rate by incorporating knowledge of the geometry of past observations
Why Adapt to Geometry?

Examples from Duchi et al. ISMP 2012 slides

<table>
<thead>
<tr>
<th>$y_t$</th>
<th>$X_{t,1}$</th>
<th>$X_{t,2}$</th>
<th>$X_{t,3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>.5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>-.5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>-.5</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

1. Frequent, irrelevant
2. Infrequent, predictive
3. Infrequent, predictive
Motivation

Text data:
The most unsung birthday in American business and technological history this year may be the 50th anniversary of the Xerox 914 photocopier.\textsuperscript{a} 

\textsuperscript{a}The Atlantic, July/August 2010.

Other motivation:
selecting advertisements in online advertising, document ranking, problems with parameterizations of many magnitudes...

---

Images from Duchi et al. ISMP 2012 slides

©Sham Kakade 2017
Visualizing Effect

Credit: http://imgur.com/a/Hqolp

©Sham Kakade 2017
Regret Minimization

• How do we assess the performance of an online algorithm?

• Algorithm iteratively predicts $\mathbf{w}^{(t)}$

• Incur loss $\ell_t(\mathbf{w}^{(t)})$

• **Regret:**
  What is the total incurred loss of algorithm relative to the best choice of $\mathbf{w}$ that could have been made *retrospectively*

\[
R(T) = \sum_{t=1}^{T} \ell_t(\mathbf{w}^{(t)}) - \inf_{\mathbf{w} \in \mathcal{W}} \sum_{t=1}^{T} \ell_t(\mathbf{w})
\]
Regret Bounds for Standard SGD

- Standard projected gradient stochastic updates:

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

- Standard regret bound:

\[
\sum_{t=1}^{T} \ell_t(w^{(t)}) - \ell_t(w^*) \leq \frac{1}{2\eta} \| w^{(1)} - w^* \|^2 + \frac{\eta}{2} \sum_{t=1}^{T} \| g_t \|^2
\]
Projected Gradient using Mahalanobis

• Standard projected gradient stochastic updates:

\[ w^{(t+1)} = \arg \min_{w \in \mathcal{W}} \| w - (w^{(t)} - \eta g_t) \|_2^2 \]

• What if instead of an \( L_2 \) metric for projection, we considered the **Mahalanobis** norm

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

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

- What \( A \) to choose?
- Regret bound now:

\[
\sum_{t=1}^{T} \ell_t(\mathbf{w}^{(t)}) - \ell_t(\mathbf{w}^*) \leq \frac{1}{2\eta} \| \mathbf{w}^{(1)} - \mathbf{w}^* \|_2^2 + \frac{\eta}{2} \sum_{t=1}^{T} \| g_t \|_A^{-1}^2
\]

- 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:

\[
A = c \left( \sum_{t=1}^{T} g_t g_t^T \right)^{1/2}
\]

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,

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

$$ w^{(t+1)} = \arg \min_{w \in \mathcal{W}} \|w - (w^{(t)} - \eta \text{diag}(A_t)^{-1} g_t)\|_2^2 \text{diag}(A_t) $$
AdaGrad in Euclidean Space

• For $\mathcal{W} = \mathbb{R}^d$, 

• For each feature dimension,

\[ w_i^{(t+1)} \leftarrow w_i^{(t)} - \eta_{t,i} g_{t,i} \]

where

\[ \eta_{t,i} = \]

• That is,

\[ w_i^{(t+1)} \leftarrow w_i^{(t)} - \frac{\eta}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t,i} \]

• Each feature dimension has it’s 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
AdaGrad Theoretical Guarantees

- AdaGrad regret bound:
  \[
  \sum_{t=1}^{T} \ell_t(w^{(t)}) - \ell_t(w^*) \leq 2R_\infty \sum_{i=1}^{d} \|g_{1:T,i}\|_2
  \]
  \[
  R_\infty := \max_t \|w^{(t)} - w^*\|_\infty
  \]
  - 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 really is used in practice!
- Many cool examples. Let’s just examine one…
AdaGrad Theoretical Example

• Expect to out-perform when gradient vectors are *sparse*
• SVM hinge loss example:

\[
\ell_t(w) = \left[1 - y^t \langle x^t, w \rangle \right]_+
\]

\[x^t \in \{-1, 0, 1\}^d\]

• If \(x_j^t \neq 0\) with probability \(\propto j^{-\alpha}\), \(\alpha > 1\)

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

• (sort of) previously bound:

\[
\mathbb{E} \left[ \ell \left( \frac{1}{T} \sum_{t=1}^{T} w^{(t)} \right) \right] - \ell(w^*) = O \left( \frac{\|w^*\|_{\infty}}{\sqrt{T}} \cdot \sqrt{d} \right)
\]
Neural Network Learning

- Very non-convex problem, but use SGD methods anyway

\[ \ell(w, x) = \log(1 + \exp(\langle [p(\langle w_1, x_1 \rangle) \cdots p(\langle w_k, x_k \rangle)] , x_0 \rangle)) \]

\[ p(\alpha) = \frac{1}{1 + \exp(\alpha)} \]

(Dean et al. 2012)

Distributed, \( d = 1.7 \cdot 10^9 \) parameters. SGD and AdaGrad use 80 machines (1000 cores), L-BFGS uses 800 (10000 cores)
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
  \[ \theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \]
Comparisons: MNIST, Sigmoid 100 layer

- **MNIST**
- **Sigmoid 100 layer**

Graph showing test set accuracy for different optimization methods:
- **momentum [0.125]**
- **adam [0.001]**
- **adadelta [0.9]**
- **adagrad [0.075]**
- **sgd [0.5]**

© Sham Kakade 2017
Comparisons: MNIST, Tanh 100 layer
Comparisons: Sigmoid, ReLu, Sigmoid
Acknowledgments