Convergence rate of SGD

**Theorem:**
- Let $f$ be a strongly convex stochastic function with $\mathbb{E} [\nabla f(w)] = w^*$.
- Assume gradient of $f$ is Lipschitz continuous and bounded:
  \[ \forall x \quad \| \nabla f(w) - \nabla f(w') \|_2 \leq L\| w - w' \|_2 \quad \text{for } L > 0 \]
- Then, for step sizes:
  \[ \eta = \frac{K}{\epsilon} \quad \text{where } K > 0 \]
- The expected loss decreases as $O(1/t)$:
  \[ \mathbb{E} [f(w(t)) - f(w^*)] \leq \frac{1}{t} \left( \frac{M^2 + \| w(0) - w^* \|_2^2}{\epsilon^2} \right) \]

Convergence rates for gradient descent/ascent versus SGD

- **Number of Iterations to get to accuracy**
  \[ \ell(w^*) - \ell(w) \leq \epsilon \]
  - Gradient descent:
    - If func is strongly convex: $O(\ln(1/\epsilon))$ iterations
  - Stochastic gradient descent:
    - If func is strongly convex: $O(1/\epsilon)$ iterations
  - Seems exponentially worse, but much more subtle:
    - Total running time, e.g., for logistic regression:
      - Gradient descent:
        - $O(1/\epsilon)$ iterations
      - SGD:
        - $O(\sqrt{d}/\epsilon)$ iterations
        - SGD can win when we have a lot of data
    - And, when analyzing true error, situation even more subtle... expected running time about the same, see readings
Motivating AdaGrad (Duchi, Hazan, Singer 2011)

- Assuming $w \in \mathbb{R}^d$, standard stochastic (sub)gradient descent updates are of the form:
  $$w_i^{(t+1)} \leftarrow w_i^{(t)} - \eta g_{t,i}$$
  - Step size
  - Learning rate

- Should all features share the same learning rate?

- 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?

<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>0</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>0</td>
<td>0</td>
</tr>
<tr>
<td>-3</td>
<td>-.5</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Examples from Duchi et al. ISMP 2012 slides.

- Frequent, irrelevant
- Infrequent, predictive
- Infrequent, predictive
Not All Features are Created Equal

Examples:

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

\(^a\)The Atlantic, July/August 2010.

High-dimensional image features

Projected Gradient

Brief aside…

Consider an arbitrary feature space \( w \in \mathcal{W} \subseteq \mathbb{R}^d \)

If \( w \in \mathcal{W} \), can use projected gradient for (sub)gradient descent

\[
\begin{align*}
\mathbf{w}^{(t+1)} &= \mathbf{w}^{(t)} - \eta g_t, i \\
&= \arg \min_{w \in \mathcal{W}} \| w - (\mathbf{w}^{(t)} - \eta g_t) \|_2^2
\end{align*}
\]

efficient for some \( w \)

\( \eta \ll \| \mathbf{w} \| \)
Regret Minimization

- How do we assess the performance of an online algorithm?
  - Algorithm iteratively predicts $w^{(t)}$
  - Incur loss $f_t(w^{(t)})$
  - **Regret**: What is the total incurred loss of algorithm relative to the best choice of $w$ that could have been made retrospectively.

$$R(T) = \sum_{t=1}^{T} f_t(w^{(t)}) - \inf_{w \in \mathcal{W}} \sum_{t=1}^{T} f_t(w)$$

Regret Bounds for Standard SGD

- Standard projected gradient stochastic updates:
  $$w^{(t+1)} = \arg \min_{w \in \mathcal{W}} \|w - (w^{(t)} - \eta g_t)\|_2^2$$

- Standard regret bound:
  $$\sum_{t=1}^{T} f_t(w^{(t)}) - f_t(w^*) \leq \frac{1}{2\eta} \|w^{(1)} - w^*\|_2^2 + \frac{\eta}{2} \sum_{t=1}^{T} \|g_t\|_2^2$$
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 \textbf{Mahalanobis} norm
  \[ w^{(t+1)} = \arg \min_{w \in \mathcal{W}} ||w - (w^{(t)} - \eta A^{-1} g_t)||_A^2 \]

Mahalanobis Regret Bounds

- What \( A \) to choose?

- Regret bound now:
  \[ \sum_{t=1}^{T} f_t(w^{(t)}) - f_t(w^*) \leq \frac{1}{2\eta} ||w^{(1)} - w^*||_A^2 + \eta \sum_{t=1}^{T} ||g_t||_A^{-2} \]

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

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

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,

\[ \text{diag} \left( A_t \right) \]

Then, algorithm is a simple modification of normal updates:

\[ w^{(t+1)} = \arg \min_{w \in \mathcal{W}} \left\| w - (w^{(t)} - \eta \text{diag}(A_t)^{-1} g_t) \right\|^2_{\text{diag}(A_t)} \]
AdaGrad in Euclidean Space

- For $W = \mathbb{R}^d$, no constraints on $w$
- For each feature dimension,
  $$w_i^{(t+1)} \leftarrow w_i^{(t)} - \eta_{t,i} g_{t,i}$$
  where
  $$\eta_{t,i} = \eta \sqrt{A_{t,i}}$$
- That is,
  $$w_i^{(t+1)} \leftarrow w_i^{(t)} - \eta \frac{g_{t,i}}{\sqrt{\sum_{\tau=1}^{T} g_{\tau,i}^2}}$$
- 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

AdaGrad Theoretical Guarantees

- AdaGrad regret bound:
  $$\sum_{t=1}^{T} f_t(w^{(t)}) - f_t(w^*) \leq 2R_\infty \sum_{i=1}^{d} ||g_{1:T,i}||_2$$
  where
  $$R_\infty := \max_t ||w^{(t)} - w^*||_\infty$$
  radius of spa
- So, what does this mean in practice?
  - Many cool examples. This really is used in practice!
  - Let’s just examine one…
AdaGrad Theoretical Example

- Expect to out-perform when gradient vectors are sparse
- SVM hinge loss example:
  \[ f_t(w) = [1 - y_t \langle x_t, w \rangle]_+ \]
  where \( x_t \in \{-1, 0, 1\}^d \)
  - If \( x'_t \neq 0 \) with probability \( \propto j^{-\alpha} \), \( \alpha > 1 \)
  
  \[ \mathbb{E} \left[ f \left( \frac{1}{T} \sum_{t=1}^{T} w^{(t)} \right) - f(w^*) \right] = O \left( \frac{||w^*||_\infty \sqrt{T}}{\sqrt{d}} \max \{ \log d, d^{1-\alpha/2} \} \right) \]

- Previously best known method:
  
  \[ \mathbb{E} \left[ f \left( \frac{1}{T} \sum_{t=1}^{T} w^{(t)} \right) \right] - f(w^*) = O \left( \frac{||w^*||_\infty}{\sqrt{T}} \cdot \sqrt{d} \right) \]

Neural Network Learning

- Very non-convex problem, but use SGD methods anyway
  
  \[ f(w; \xi) = \log(1 + \exp(\langle p(x_t, \xi) \cdots p(x_k, \xi), \xi \rangle)) \]

Distributed, \( d = 1.7 \cdot 10^9 \) parameters. SGD and AdaGrad use 80 machines (1000 cores), L-BFGS uses 800 (10000 cores)

Images from Duchi et al. ISMP 2012 slides
What you should know about Logistic Regression (LR) and Click Prediction

- Click prediction problem:
  - Estimate probability of clicking
  - Can be modeled as logistic regression
- Logistic regression model: Linear model
- Gradient ascent to optimize conditional likelihood
- Overfitting + regularization
- Regularized optimization
  - Convergence rates and stopping criterion
- Stochastic gradient ascent for large/streaming data
  - Convergence rates of SGD
- AdaGrad motivation, derivation, and algorithm