Case Study 1: Estimating Click Probabilities

SGD cont’d
AdaGrad

Machine Learning for Big Data
CSE547/STAT548, University of Washington
Sham Kakade
March 31, 2015
Support/Resources

• Office Hours
  – Yao Lu: Tue 1:30-2:30, CSE 220
  – John Thickstun: Weds 4-5, CSE 220
Learning Problem for Click Prediction

- Prediction task: \[ X \rightarrow \{0,1\}^3 \quad \Pr (\text{click} = 1 \mid X) \]

- Features:
  \[ X = (\text{feats of page, ad, user, keyword}) \]
  \( \subseteq \) web pages, adIDs, user IDs, ...

- Data:
  \( (X_i, y_i) \)
  - Batch:
    Fixed dataset \( (X_1, y_1), ..., (X_n, y_n) \)
  - Online:
    Data as stream \( X_t \) predict \( Y_t \) user arrives at time \( t \) \( \overset{\text{observe}}{\leftarrow} \)

- Many approaches (e.g., logistic regression, SVMs, naïve Bayes, decision trees, boosting,...)
  - Focus on logistic regression; captures main concepts, ideas generalize to other approaches
Challenge 1: Complexity of computing gradients

• What’s the cost of a gradient update step for LR???

\[
w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 | x^j, w)] \right\}
\]

\[O(NA)\] for this update.

Naively, \[O(NA^2)\] for all features.

With “caching” \(y_j^i\): \[O(NA)\] (\(N\) is large, \(A\) is fixed).
Challenge 2: Data is streaming

- Assumption thus far: **Batch data**

- But, click prediction is a streaming data task:
  - User enters query, and ad must be selected:
    - Observe $x^i$, and must predict $y^i$
    - User either clicks or doesn’t click on ad:
      - Label $y^i$ is revealed afterwards
        - Google gets a reward if user clicks on ad
  - Weights must be updated for next time:
SGD: Stochastic Gradient Ascent (or Descent)

• “True” gradient: $\nabla \ell(\mathbf{w}) = \mathbb{E}_{\mathbf{x}} [\nabla \ell(\mathbf{w}, \mathbf{x})]$

• Sample based approximation:

• What if we estimate gradient with just one sample???
  – Unbiased estimate of gradient
  – Very noisy!
  – Called stochastic gradient ascent (or descent)
    • Among many other names
  – VERY useful in practice!!!
Stochastic Gradient Ascent: General Case

- Given a stochastic function of parameters:
  - Want to find maximum

- Start from $w^{(0)}$
- Repeat until convergence:
  - Get a sample data point $x^t$
  - Update parameters:

- Works in the online learning setting!
- Complexity of each gradient step is constant in number of examples!
- In general, step size changes with iterations
Stochastic Gradient Ascent for Logistic Regression

• Logistic loss as a stochastic function:

\[
E_x \left[ \ell(w, x) \right] = E_x \left[ \ln P(y|x, w) - \frac{\lambda}{2} \|w\|^2 \right]
\]

• Batch gradient ascent updates:

\[
w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \frac{1}{N} \sum_{j=1}^{N} x_i^{(j)} [y^{(j)} - P(Y = 1|x^{(j)}, w^{(t)})] \right\}
\]

• Stochastic gradient ascent updates:

– Online setting:

\[
w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta_t \left\{ -\lambda w_i^{(t)} + x_i^{(t)} [y^{(t)} - P(Y = 1|x^{(t)}, w^{(t)})] \right\}
\]
Convergence Rate of SGD

• **Theorem:**
  – (see Nemirovski et al ‘09 from readings)
  – Let $f$ be a strongly convex stochastic function
  – Assume gradient of $f$ is Lipschitz continuous and bounded

  – Then, for step sizes:

  – The expected loss decreases as $O(1/t)$:
Convergence Rates for Gradient Descent/Ascent vs. 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:
    • SGD:
      • SGD can win when we have a lot of data
    – See readings for more details
Constrained SGD: Projected Gradient

• Consider an arbitrary restricted feature space $\mathbf{w} \in \mathcal{W}$

• Optimization objective:

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

\[ \mathbf{w}^{(t+1)} = \]
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?

- 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>$\hat{X}_{t,1}$</th>
<th>$\hat{X}_{t,2}$</th>
<th>$\hat{X}_{t,3}$</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
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.

Images from Duchi et al. ISMP 2012 slides
Visualizing Effect

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

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

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

- Standard projected gradient stochastic updates:

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

- Standard regret bound:

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

• Standard projected gradient stochastic updates:

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

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

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

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

- **What \( \mathbf{A} \) to choose?**
- **Regret bound now:**

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

- **What if we minimize upper bound on regret w.r.t. \( \mathbf{A} \) in hindsight?**

\[
\min_{\mathbf{A}} \sum_{t=1}^{T} \mathbf{g}_t^T \mathbf{A}^{-1} \mathbf{g}_t
\]
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:

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

• For $\mathcal{W} = \mathbb{R}^d$,

• For each feature dimension,

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

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 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:

\[ R_\infty := \max_t \|w^{(t)} - w^*\|_\infty \]

\[
\sum_{t=1}^{T} \ell_t(w^{(t)}) - \ell_t(w^*) \leq 2R_\infty \sum_{i=1}^{d} \|g_{1:T,i}\|_2
\]

– In stochastic setting:

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

• 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 \( \propto j^{-\alpha} \), \( \alpha > 1 \)

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

• Previously best known method:

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

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
  - Convergence rates and stopping criterion

- Regularized optimization
  - Convergence rates of SGD

- Stochastic gradient ascent for large/streaming data
  - Convergence rates of SGD

- AdaGrad motivation, derivation, and algorithm