Inverse Dependency on Training Data Size

Zeyuan Zhu
Weizhuu Chen
Gang Wang
Train data size is large, learning time is large?

All learning algorithms have the inverse dependency?

Which kind of algorithms can benefit from inverse dependency?

Before answering these questions, let us see an algorithm called PEGASOS and its inverse dependency work.
Pegasos: Primal Estimated sub-GrAdient Solver for SVM

Shai Shalev-Shwartz
Yoram Singer
Nathan Srebro
SVM Objective: \( S = \{ w : \| w \|_2 \leq 1/\sqrt{\sigma} \} \)

\[
\min_{w \in S} \frac{\sigma}{2} \| w \|_2^2 + \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \langle w, x_i \rangle\}
\]

\( I_t \subset [m] \)

\[
g_t(w) = \frac{\sigma}{2} \| w \|_2^2 + \frac{1}{|I_t|} \sum_{i \in I_t} \max\{0, 1 - y_i \langle w, x_i \rangle\}
\]

We write the gradient of \( g_t(w) \)

\[
\nabla_t = \sigma w + \frac{1}{|I_t^+|} \sum_{i \in I_t^+} \max\{0, 1 - y_i \langle w, x_i \rangle\}
\]
Initially \( w_0 = 0 \)

Iteration \( T \) times

- Do the gradient descent with learning rate \( 1/\sigma t \)

\[
\begin{align*}
    w_{t+1/2} &= w_t - \frac{1}{\sigma t} \nabla_t \\
    \text{For general algorithm, learning rate has to be in the order of} \quad 1/\sqrt{t}, \text{but for strongly-convex objective...} \\
    \text{o Project it back to a sphere} \\
    w_{t+1} &= \min \left\{ 1, \frac{1/\sqrt{\sigma}}{\|w_{t+1/2}\|} \right\} \cdot w_{t+1/2} \\
    \text{It can be proved that the optimal solution is within this sphere}
\end{align*}
\]
Table 1. Training time in CPU-seconds

<table>
<thead>
<tr>
<th></th>
<th>Pegasos</th>
<th>SVM-Perf</th>
<th>SVM-Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCAT</td>
<td>2</td>
<td>77</td>
<td>20,075</td>
</tr>
<tr>
<td>Covertype</td>
<td>6</td>
<td>85</td>
<td>25,514</td>
</tr>
<tr>
<td>astro-ph</td>
<td>2</td>
<td>5</td>
<td>80</td>
</tr>
</tbody>
</table>

\( \tilde{O} \left( \frac{1}{\sigma \delta \epsilon_{acc}} \right) \) iterations
Assume that \( \forall i \in [m], \|x_i\| \leq R \), \( u \) be an arbitrary vector in \( S \)

\[
\frac{1}{T} \sum_{t=1}^{T} g_t(w_t) \leq \frac{1}{T} \sum_{t=1}^{T} g_t(u) + \frac{(\sqrt{\sigma} + R)^2}{2\sigma T} (1 + \log(T))
\]

Corollary, if \( I_t = [m] \), let \( \bar{w} = \frac{1}{T} \sum_{t=1}^{T} w_t \), then

\[
g(\bar{w}) \leq g(u) + \frac{(\sqrt{\sigma} + R)^2 (1 + \log(T))}{2\sigma T}
\]

If \( I_t \neq [m] \), the expectation still obeys the above inequality, then use Markov bound, incorporate a confidence parameter \( \delta \)
SVM Optimization: Inverse Dependence on Training Set Size

Shai Shalev-Shwartz
Nathan Srebro
Hinge loss: \( \hat{\ell}(w) = \frac{1}{m} \sum_i \ell(w; (x_i, y_i)) \)

Regularized empirical hinge loss: \( \hat{f}_\lambda(w) = \hat{\ell}(w) + \frac{\lambda}{2} \|w\|^2 \)

Generalization error: \( \ell(w) = E_{X,Y \sim P}[\ell(w; X, Y)] \)

Predictor / Empirical optimum: \( \hat{w} = \text{argmin}_w \hat{f}_\lambda(w) \)

We are interested in \( \ell(\hat{w}) \) !!!!

Population optimum: \( w^* = \text{argmin}_w f_\lambda(w) \)

\( \epsilon_{acc} \) -optimal predictor: \( \tilde{w} \) s.t. \( \hat{f}_\lambda(\tilde{w}) \leq \hat{f}_\lambda(\hat{w}) + \epsilon_{acc} \)

**Figure 1.** Decomposition of the generalization error of the output \( \tilde{w} \) of the optimization algorithm: \( \ell(\tilde{w}) = \epsilon_{aprx} + \epsilon_{est} + \epsilon_{opt} \).
\[ \ell(\tilde{w}) \leq \ell(w_0) + \tilde{O}\left(\frac{d}{\lambda T}\right) + \frac{\lambda}{2} \|w_0\|^2 + O\left(\frac{1}{\lambda m}\right) \quad (8) \]

The above bound is minimized when \( \lambda = \tilde{O}\left(\sqrt[\frac{d}{T} + \frac{1}{m}}\|w_0\|\right) \), yielding \( \ell(\tilde{w}) \leq \ell(w_0) + \epsilon(T, m) \) with

\[ \epsilon(T, m) = \tilde{O}\left(\|w_0\| \sqrt[\frac{d}{T}}\right) + O\left(\frac{\|w_0\|}{\sqrt{m}}\right). \quad (9) \]

Inverting the above expression, we get the following bound on the runtime required to attain generalization error \( \ell(\tilde{w}) \leq \ell(w_0) + \epsilon \) using a training set of size \( m \):

\[ T(m; \epsilon) = \tilde{O}\left(\frac{d}{\left(\epsilon \left(\frac{1}{\|w_0\|} - O\left(\frac{1}{\sqrt{m}}\right)\right)^2 \right)}\right). \quad (10) \]
Inverse Dependency

Upper Bound!
If the desired error rate $\epsilon$ obeys $l(\tilde{w}) \leq l(w_0) + \epsilon$, we invert the equation and have

$$T = O\left(\frac{1/\delta}{2\epsilon^2(p-1) - \tilde{\delta}\left(\frac{1}{m}\right)}\right)$$

(22)

Choosing $^2 k = 1$ and integrating (22) into the complexity of the Primal Gradient Solver, we conclude that:

**Fig. 3:** Inverse time dependency with fixed generalization loss
Why this stochastic algorithm runs so efficient and accurate?

Online Learning algorithm plays an important role in it!
Mind the Duality Gap: Logarithmic regret algorithms for online optimization
Sham M. Kakade; Shai Shalev-Shwartz
Logarithmic Regret for Online Optim

INPUT: A strongly convex function $f$

FOR $t = 1, 2, \ldots, T$:

1) Define $w_t = \nabla f^* \left( -\frac{\lambda_{1:t-1}}{\sqrt{t}} \right)$

2) Receive a function $\ell_t$

3) Suffer loss $\ell_t(w_t)$

4) Update $\lambda_1^{t+1}, \ldots, \lambda_t^{t+1}$ s.t. there exists $\lambda_t \in \partial \ell_t(w_t)$ with

$$\mathcal{D}_{t+1}(\lambda_1^{t+1}, \ldots, \lambda_t^{t+1}) \geq \mathcal{D}_{t+1}(\lambda_1^t, \ldots, \lambda_{t-1}^t, \lambda_t)$$

INPUT: A $\sigma$-strongly convex function $f$

FOR $t = 1, 2, \ldots, T$:

1) Define $w_t = \nabla f^* \left( -\frac{\lambda_{1:t-1}}{\sigma_{1:t}} \right)$

2) Receive a function $\ell_t = \sigma f + g_t$

3) Suffer loss $\ell_t(w_t)$

4) Update $\lambda_1^{t+1}, \ldots, \lambda_t^{t+1}$ s.t. there exists $\lambda_t \in \partial g_t(w_t)$ with

$$\mathcal{D}_{t+1}(\lambda_1^{t+1}, \ldots, \lambda_t^{t+1}) \geq \mathcal{D}_{t+1}(\lambda_1^t, \ldots, \lambda_{t-1}^t, \lambda_t)$$

Figure 3: Primal-dual template algorithms for general online convex optimization (left) and online strongly convex optimization (right). Here $a_{1:t} = \sum_{i=1}^{t} a_i$, and for notational convenience, we implicitly assume that $a_{1:0} = 0$. See text for description.
Train data size is large, learning time is large?
- Not always!!

Which kind of algorithms can benefit from inverse dependency?
- Why inverse dependency? Each iteration the complexity is independent of the document size! So I personally believe: stochastic ones! The accuracy must be non-relative to the document size.

All learning algorithm have the inverse dependency?
Inverse Time Dependency in Regularized Learning with Convex Loss and $L_p$ Regularizer

Zeyuan Zhu$^{1,2}$, Weizhu Chen$^1$, Chenguang Zhu$^{1,3}$, Gang Wang$^1$

In submission to ECML 2009
Our work

\[
\hat{F}_{\sigma}(\mathbf{w}) = \frac{\sigma}{2(p-1)} \|\mathbf{w}\|^2_p + \frac{1}{m} \sum_{i=1}^{m} l(\langle \mathbf{w}, \phi(\mathbf{\theta}_i) \rangle; \mathbf{\theta}_i), (\mathbf{w} \in S)
\]

\[p \in (1,2]\]

- The SVM loss: \(l(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle; \mathbf{x}, y) = \max\{0, 1 - y\langle \mathbf{w}, \phi(\mathbf{x}) \rangle\}\)
- The Logistic loss: \(l(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle; \mathbf{x}, y) = \log(1 + e^{-y\langle \mathbf{w}, \phi(\mathbf{x}) \rangle})\)
- The Least Square loss: \(l(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle; \mathbf{x}, y) = (\langle \mathbf{w}, \phi(\mathbf{x}) \rangle - y)^2\)
Current work

PEGASOS non-linear, Gaussian kernel
- CCAT accuracy 94.95% within 10min on 80 cores
- CCAT linear accuracy 94.55%

SVMLight
- CCAT 91.5% accuracy, hours of training

PSVM – approximated objective
- CCAT 91.5% accuracy, 1.5hours (256cores?)