SVM Optimization: An Inverse Dependence on Data Set Size

Shai Shalev-Shwartz  
Nati Srebro  
Toyota Technological Institute—Chicago

(a philanthropically endowed academic computer science institute dedicated to basic research and graduate education in computer science)
More Data ⇒ More Work?

10k training examples
1 hour
2.3% error (when using the predictor)

1M training examples
1 week (or more…)
2.29% error

Can always sample and get same runtime:
1 hour
2.3% error

Can we leverage the excess data to reduce runtime?
10 minutes
2.3% error

But I really care about that 0.01% gain

Study runtime increase as a function of target accuracy

My problem is so hard, I have to crunch 1M examples

Study runtime increase as a function of problem difficulty (e.g. small margin)
SVM Training

- Optimization objective:
  \[ f(w) = \frac{\lambda}{2} \|w\|^2 + \frac{1}{n} \sum_{i=1}^{n} [1 - y_i \langle w, x_i \rangle]^+ \]

- True objective: prediction error
  \[ \text{err}(w) = E_{x,y}[\text{error of } \langle w, x \rangle \text{ vs. } y] \]

- Would like to understand computational cost in terms of:
  - **Increasing** function of:
    - Desired generalization performance (i.e. as err(w) decreases)
    - Hardness of problem:
      - Margin, noise (unavoidable error)
  - **Decreasing** function of available data set size
Error Decomposition

- **Approximation error:**
  - Best error achievable by large-margin predictor
  - Error of population minimizer
    \[ w_0 = \text{argmin}_w E[f(w)] = \text{argmin}_w \lambda |w|^2 + E_{x,y}[\text{loss}(\langle w, x \rangle; y)] \]

- **Estimation error:**
  - Extra error due to replacing \( E[\text{loss}] \) with empirical loss
    \[ w^* = \text{arg min}_w f_n(w) \]

- **Optimization error:**
  - Extra error due to only optimizing to within finite precision
The Double-Edged Sword

- When data set size increases:
  - **Estimation error** decreases
  - Can increase **optimization error**, i.e. optimize to within lesser accuracy ⇒ fewer iterations
  - But handling more data is expensive e.g. runtime of each iteration increases

- **Stochastic Gradient Descent**, e.g. **PEGASOS** (Primal Efficient Sub-Gradient Solver for SVMs) [Shalev-Shwartz Singer Srebro, ICML’07]
  - Fixed runtime per iteration
  - Runtime to get fixed accuracy does not increase with n
**PEGASOS: Stochastic (sub-)Gradient Descent**

\[
f(w) = \lambda \|w\|^2 + \frac{1}{n} \sum_{i=1}^{n} [1 - y_i \langle w, x_i \rangle]_+
\]

- Initialize \( w = 0 \)
- At each iteration \( t \), with random data point \( (x_i, y_i) \):
  \[
  \nabla = 2\lambda w - \begin{cases} 
  y_i x_i & \text{if } y_i \langle w, x_i \rangle < 1 \\
  0 & \text{otherwise}
  \end{cases}
  \]
  \[
  w \leftarrow w - \frac{1}{2\lambda t} \nabla
  \]

- **Theorem**: After at most \( \tilde{O}\left(\frac{1}{\lambda \epsilon}\right) \) iterations, \( E[f(w_{\text{PEGASOS}})] \leq \min_w f(w) + \epsilon \)
- With \( d \)-dimensional (or \( d \)-sparse) features, each iteration takes time \( O(d) \)
- **Conclusion**: Run-time required for PEGASOS to find \( \epsilon \) accurate solution:
  \[
  \tilde{O}\left(\frac{d}{\lambda \epsilon}\right)
  \]
- Run-time does not depend on \#examples
Comparison of Runtime Guarantees

\[ f(w) = \lambda \|w\|^2 + \frac{1}{n} \sum_{i=1}^{n} [1 - y_i \langle w, x_i \rangle] \]

- Runtime to get \( \varepsilon_{\text{acc}} \)-accurate solution: \( f(w) \leq \min f(w) + \varepsilon_{\text{acc}} \)
  - PEGASOS: \( d / (\lambda \varepsilon_{\text{acc}}) \)
  - SVMPerf: \( n \, d / (\lambda \varepsilon_{\text{acc}}) \)
  - Dual Decomposition (SMO): \( n^2 \, d \log(1/\varepsilon_{\text{acc}}) \)
  - Interior Point: \( n^{3.5} \log(\log(1/\varepsilon_{\text{acc}})) \)

(ignoring log-factors)

- Would like to understand computational cost in terms of:
  - Increasing function of:
    - Desired generalization performance (i.e. as \( \text{err}(w) \) decreases)
    - Hardness of problem:
      - margin, noise (unavoidable error)
  - Decreasing function of available data set size
Comparison of Runtime Guarantees

If there is some predictor $w_0$ with low $|w_0|$ and low $\text{err}(w_0)$, how much time to find predictor with $\text{err}(w) \leq \text{err}(w_0) + \epsilon$

$$\text{err}(w) = \text{err}(w_0) + \lambda(|w_0|^2 - |w|^2) + E[f(w)] - E[f(w_0)] + 2(f(w) - f(w_0)) + O(1/(\lambda n)) + O(\epsilon)$$

$$\leq \text{err}(w_0) + \lambda|w_0|^2 + 2\epsilon_{\text{acc}} + O(1/(\lambda n)) + O(\epsilon)$$

To get $\text{err}(w) \leq \text{err}(w_0) + O(\epsilon)$:

- $\lambda = O(\epsilon/|w_0|^2)$
- $\epsilon_{\text{acc}} = O(\epsilon)$
- $n = \Omega(1/(\lambda \epsilon)) = \Omega(|w_0|^2/\epsilon^2)$

Unlimited data available, can choose working data-set size.
Comparison of Runtime Guarantees

If there is some predictor $w_0$ with low $|w_0|$ and low $err(w_0)$, how much time to find predictor with $err(w) \leq err(w_0) + \varepsilon$

Traditional
$f(w) < f(w^*) + \varepsilon_{acc}$

IP
$n^{3.5} \log(\log(1/\varepsilon_{acc}))$

SMO
$n^2 d \log(1/\varepsilon_{acc})$

SVMPperf
$n d / (\lambda \varepsilon_{acc})$

PEGASOS
$d / (\lambda \varepsilon_{acc})$

To get $err(w) \leq err(w_0) + O(\varepsilon)$:

$\lambda = O(\varepsilon/|w_0|^2)$

$\varepsilon_{acc} = O(\varepsilon)$

$n = \Omega(1/(\lambda \varepsilon)) = \Omega(|w_0|^2/\varepsilon^2)$

Unlimited data available, can choose working data-set size

Data Laden analysis: Restricted by computation, not data
Dependence on Data Set Size

PEGASOS guaranteed runtime to get error $\text{err}(w_0) + \epsilon$ with $n$ training points:

$$T = \tilde{\Omega} \left( \frac{d}{\epsilon M - O\left(\frac{1}{\sqrt{n}}\right)} \right)^2$$

- Minimal Training Size (Stat Learning Theory)
- Target error
- Prediction error
- Minimal Runtime (Data Laden)
- Runtime to get target error
- Increases for smaller target error
- Increases for smaller margin
- Decreases for larger data set
- Optimal error
- Estimation error
- Approximation error
Dependence on Data Set Size

PEGASOS guaranteed runtime to get error $\text{err}(w_0) + \epsilon$ with $n$ training points:

$$\text{PEGASOS runtime} \propto \frac{1}{\lambda n} + O(d/\lambda T)$$

Increase $\lambda$ as training size increases!

More regularization, less predictors allowed

Larger approximation error $\text{err}(w_0) + \lambda |w_0|^2$

Faster runtime $T \propto 1/\lambda$
Dependence on Data Set Size: Traditional Optimization Approaches

- SMO
- SVM-Perf

**Graph:**
- CoverType
- Runtime vs. Training Set Size
- SVM-Perf
- SVM-Light
- CPU seconds
- Training set size
Dependence on Data Set Size: Traditional Optimization Approaches
Beyond PEGASOS

• Stochastic sub-gradient descent (e.g. PEGASOS) effective for SVMs with a linear kernel (i.e. feature vectors given explicitly)
  – Relevant especially in text analysis, where feature vectors are sparse, very high dimensional, bags-of-words

• Kernelized SVMs (i.e. given access to a non-linear kernel):
  – Stochastic sub-gradient descent applicable, but runtime to get fixed $\varepsilon_{\text{acc}}$ does increase linearly with $n$
  – Can we get similar behavior for general kernels?

• Can we more explicitly leverage excess data?
  – Playing only on the error decomposition, $\text{const} \times \text{minimum-sample-complexity}$ is enough to get to $\text{const} \times \text{minimum-data-laden-runtime}$

• Other machine learning problems…
More Data ⇒ Less Work

• Required runtime:
  – increases with complexity of the answer (separation, decision boundary)
  – increases with desired accuracy
  – decreases with amount of available data

• Stochastic (sub)-Gradient Descent for linear SVMs:
  – Runtime to get fixed optimization accuracy doesn’t depend on data set size
  – Runtime to get fixed prediction accuracy decreases as more data is available

Clustering (and other combinatorial search problems): Excess data, beyond what is statistically necessary, makes problem tractable
[Srebro Shakhnarovich Roweis ICML’06]