Large-Scale SVM Optimization: Taking a Machine Learning Perspective

Shai Shalev-Shwartz

Toyota Technological Institute at Chicago

Joint work with Nati Srebro

Talk at NEC Labs, Princeton, August, 2008

Motivation

$10k$ training examples

1 hour

2.3% error

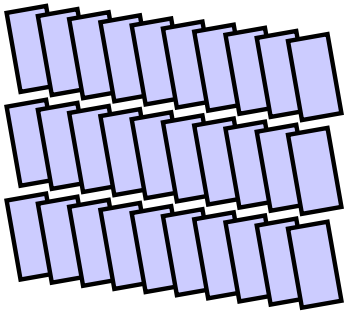
Motivation

$10k$ training examples

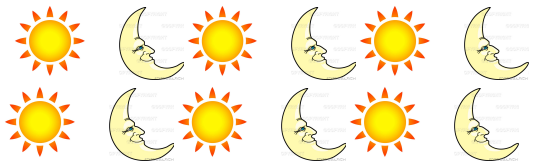
1 hour

2.3% error

$1M$ training examples



1 week



2.29% error

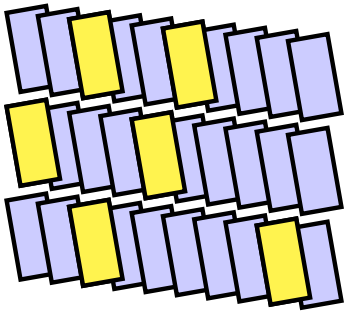
Motivation

$10k$ training examples

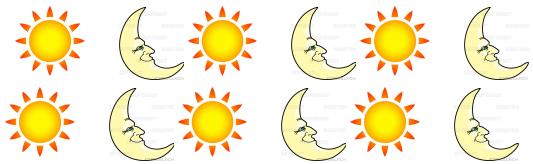
1 hour

2.3% error

$1M$ training examples



1 week



2.29% error

- Can always sub-sample and get error of 2.3% using 1 hour

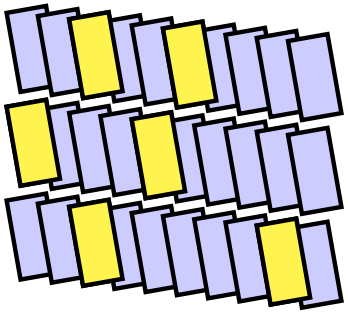
Motivation

$10k$ training examples

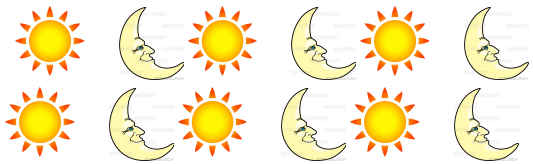
1 hour

2.3% error

$1M$ training examples



1 week



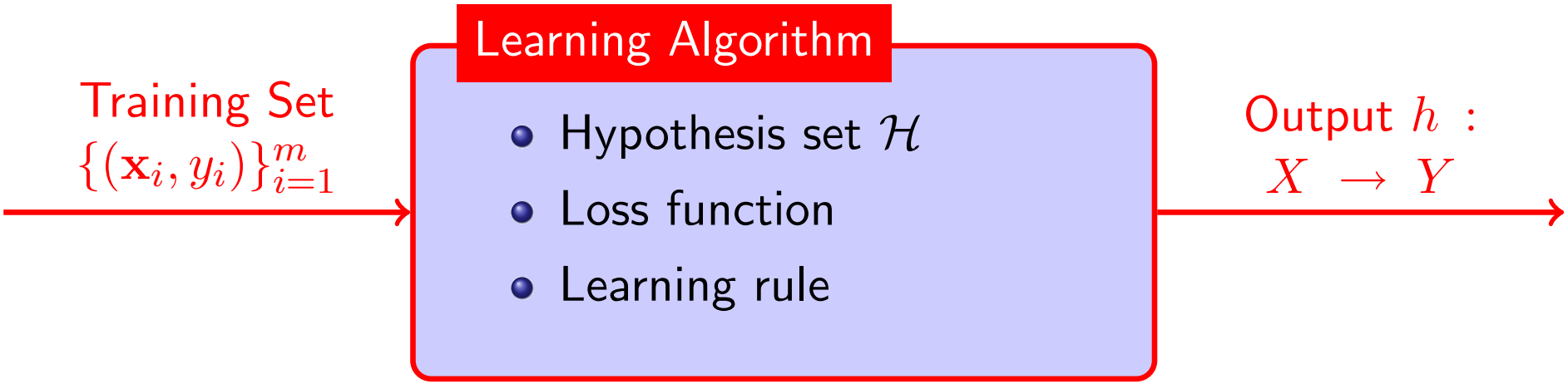
2.29% error

- Can always sub-sample and get error of 2.3% using 1 hour
- Can we leverage excess data to reduce runtime ?
  Say, achieve error of 2.3% using 10 minutes ?

Outline

- Background: Machine Learning, Support Vector Machine (SVM)
- SVM as an optimization problem
- A Machine Learning Perspective on SVM Optimization
  - Approximated optimization
  - Re-define quality of optimization using generalization error
  - Error decomposition
  - Data-Laden Analysis
- Stochastic Methods
  - Why Stochastic ?
  - PEGASOS (Stochastic Gradient Descent)
  - Stochastic Coordinate Dual Ascent

Background: Machine Learning and SVM



- Support Vector Machine
  - Linear hypotheses: $h_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$
  - Prefer hypotheses with large margin, i.e., low Euclidean norm
  - Resulting learning rule:

$$\operatorname*{argmin}_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m} \underbrace{\max\{0, 1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle\}}_{\text{Hinge-loss}}$$

Support Vector Machines and Optimization

- SVM learning rule:

$$\operatorname*{argmin}_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle\}$$

- SVM optimization problem can be written as a Quadratic Programming problem

$$\operatorname*{argmin}_{\mathbf{w},\boldsymbol{\xi}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m} \xi_i$$
$$\text{s.t.} \quad \forall i, \; 1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \leq \xi_i \quad \wedge \quad \xi_i \geq 0$$

- Standard solvers exist. End of story ?

Approximated Optimization

- If we don't have infinite computation power, we can only approximately solve the SVM optimization problem
- **Traditional analysis**
  - SVM objective:

$$P(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m}\ell(\langle \mathbf{w}, \mathbf{x}_i\rangle, y_i)$$

  - $\tilde{\mathbf{w}}$ is ρ-accurate solution if

$$P(\tilde{\mathbf{w}}) \leq \min_{\mathbf{w}} P(\mathbf{w}) + \rho$$

  - Main focus: How optimization runtime depends on ρ ? E.g. IP methods converge in time $O(m^{3.5}\ \log(\log(\frac{1}{\rho})))$
- Large-scale problems: How optimization runtime depends on m ? E.g. SMO converges in time $O(m^2\ \log(\frac{1}{\rho}))$

  SVM-Perf runtime is $O(\frac{m}{\lambda\,\rho})$

Machine Learning Perspective on Optimization

- Our real goal is *not* to solve the SVM problem $P(\mathbf{w})$
- Our goal is to find $\mathbf{w}$ with low generalization error:

$$L(\mathbf{w}) \;=\; \mathbb{E}_{(\mathbf{x},y)\sim\mathbb{P}}[\ell(\langle\mathbf{w},\mathbf{x}\rangle, y)]$$

- Redefine approximated accuracy:
  - $\tilde{\mathbf{w}}$ is ϵ-accurate solution w.r.t. margin parameter B if

$$L(\tilde{\mathbf{w}}) \leq \min_{\mathbf{w}:\|\mathbf{w}\|\leq B} L(\mathbf{w}) + \epsilon$$

  - Study runtime as a function of ϵ and B

Error Decomposition

Theorem (S, Srebro '08)

If $\tilde{\mathbf{w}}$ *satisfies*

$$P(\tilde{\mathbf{w}}) \leq \min_{\mathbf{w}} P(\mathbf{w}) + \rho$$

then, w.p. at least $1-\delta$ *over choice of training set,* $\tilde{\mathbf{w}}$ *satisfies*

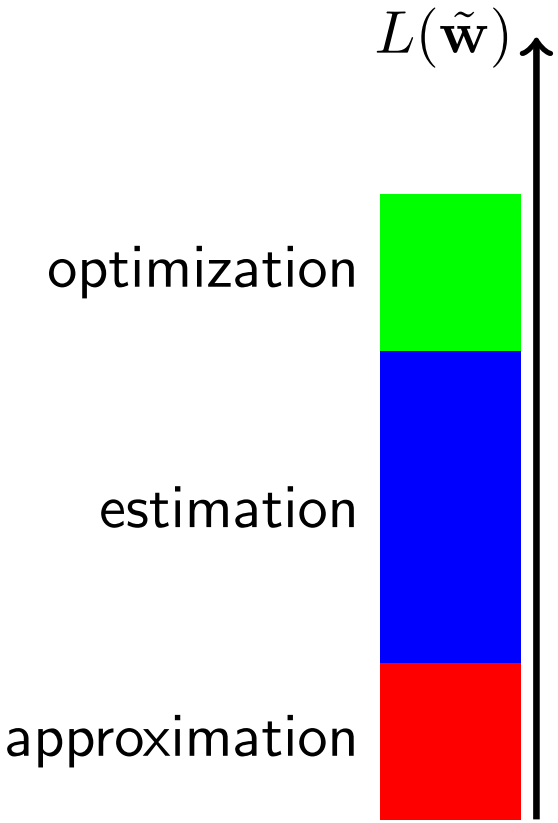
$$L(\tilde{\mathbf{w}}) \leq \min_{\mathbf{w}:\|\mathbf{w}\|\leq B} L(\mathbf{w}) + \epsilon$$

with

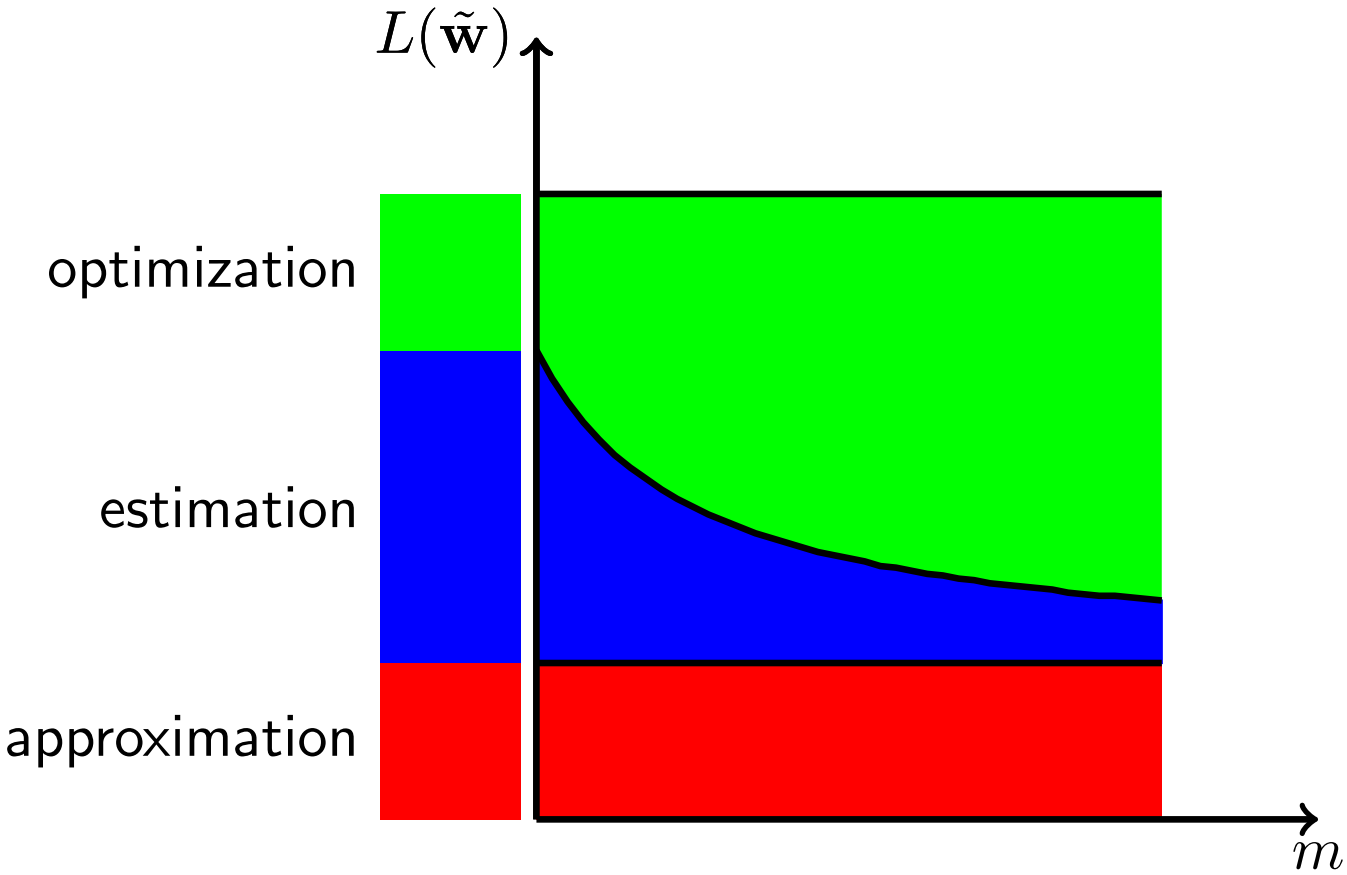
$$\epsilon = \frac{\lambda B^2}{2} + \frac{c \log(1/\delta)}{\lambda m} + 2\rho$$

(Following:
Bottou and Bousquet, "The Tradeoffs of Large Scale Learning", NIPS '08)

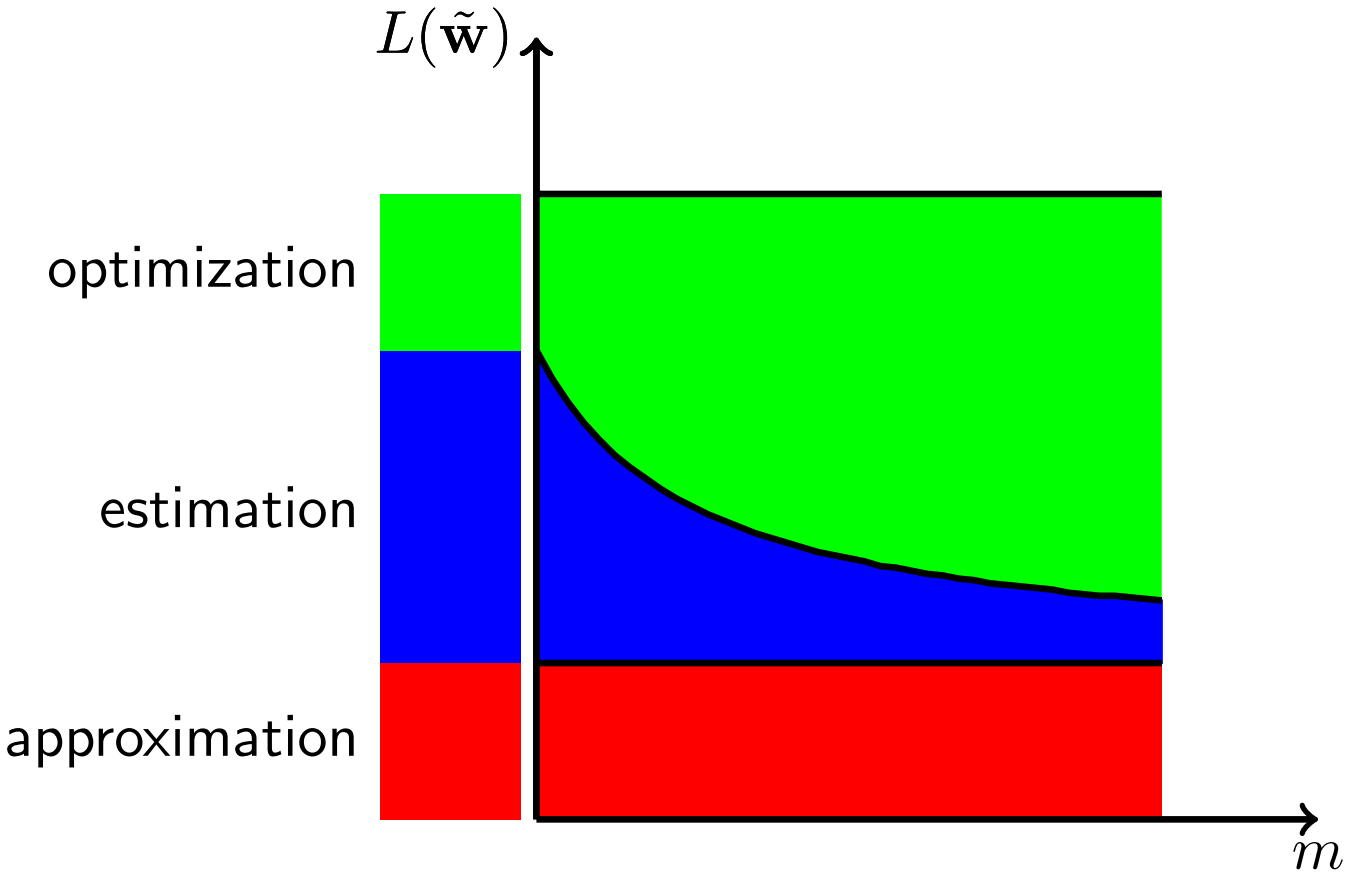
More Data $\Rightarrow$ Less Work ?



More Data ⇒ Less Work ?



More Data ⇒ Less Work ?



- When data set size increases:
  - Can increase $\rho \Rightarrow$ can optimize less accurately $\Rightarrow$ runtime decreases
  - But handling more data may be expensive $\Rightarrow$ runtime increases

Machine Learning Analysis of Optimization Algorithms

- Given solver with opt. accuracy $\rho(T, m, \lambda)$
- To ensure excess generalization error $\leq \epsilon$ we need that

$$\min_{\lambda} \frac{\lambda B^2}{2} + \frac{c \log(1/\delta)}{\lambda m} + 2\,\rho(T, m, \lambda) \leq \epsilon$$

- From the above we get runtime T as a function of m, B, ϵ

Machine Learning Analysis of Optimization Algorithms

- Given solver with opt. accuracy $\rho(T, m, \lambda)$
- To ensure excess generalization error $\leq \epsilon$ we need that

$$\min_{\lambda} \frac{\lambda B^2}{2} + \frac{c \log(1/\delta)}{\lambda m} + 2\,\rho(T, m, \lambda) \leq \epsilon$$

- From the above we get runtime T as a function of m, B, ϵ
- Examples (ignoring logarithmic terms and constants, and assuming linear kernels):

	$\rho(T, m, \lambda)$	$T(m, B, \epsilon)$
SMO (Platt '98)	$\exp(-T/m^2)$	$\left(\frac{B}{\epsilon}\right)^4$
SVM-Perf (Joachims '06)	$\frac{m}{\lambda T}$	$\left(\frac{B}{\epsilon}\right)^4$
SGD (S, Srbero, Singer '07)	$\frac{1}{\lambda T}$	$\left(\frac{B}{\epsilon}\right)^2$

Stochastic Gradient Descent (Pegasos)

- Initialize $\mathbf{w}_1 = \mathbf{0}$
- For $t = 1, 2, \ldots, T$
  - Choose $i \in [m]$ uniformly at random
  - Define

$$\nabla_t = \lambda \mathbf{w}_t - I_{[y_t \langle \mathbf{w}_t, \mathbf{x}_t \rangle > 0]} \, y_t \, \mathbf{x}_t$$

    Note: $\mathbb{E}[\nabla_t]$ is a sub-gradient of $P(\mathbf{w})$ at $\mathbf{w}_t$
  - Set $\eta_t = \frac{1}{\lambda t}$
  - Update:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla_t = (1 - \tfrac{1}{t})\mathbf{w}_t + \frac{1}{\lambda t} I_{[y_t \langle \mathbf{w}_t, \mathbf{x}_t \rangle > 0]} \, y_t \, \mathbf{x}_t$$

Theorem (Pegasos Convergence)

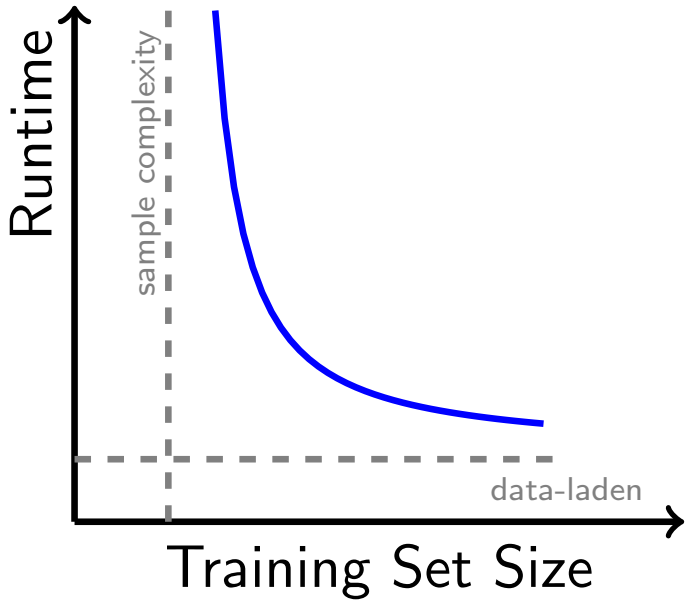
$$\mathbb{E}[\rho] \;\leq\; O\left(\frac{\log(T)}{\lambda T}\right)$$

Dependence on Data Set Size

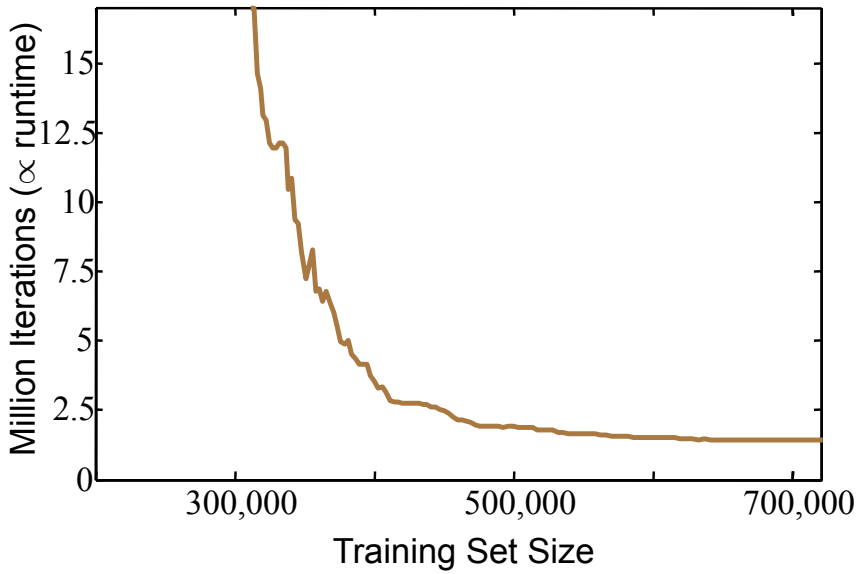
Corollary (Pegasos generalization analysis)

$$T(m;\epsilon,B) = \tilde{O}\left(\frac{1}{\left(\frac{\epsilon}{B}-\frac{1}{\sqrt{m}}\right)^2}\right)$$

Theoretical



Empirical (CCAT)



Intermediate Summary

- Analyze runtime (T) as a function of
  - excess generalization error (ϵ)
  - size of competing class (B)
- Up to constants and logarithmic terms, stochastic gradient descent (Pegasos) is optimal – its runtime is order of sample complexity $\Omega\left(\left(\frac{B}{\epsilon}\right)^2\right)$
- For Pegasos, running time decreases as training set size increases
- Coming next
  - Limitations of Pegasos
  - Dual Coordinate Ascent methods

Limitations of Pegasos

Pegasos is simple and efficient optimization method. However, it has some limitations:

- log(sample complexity) factor in convergence rate
- No clear stopping criterion
- Tricky to obtain a good single solution with high confidence
- Too aggressive at the beginning (especially when λ very small)
- When working with kernels, too much support vectors

Hsieh et al recently argued that empirically dual coordinate ascent outperforms Pegasos

Dual Methods

The dual SVM problem:

$$\min_{\boldsymbol{\alpha}\in[0,1]^m} \mathcal{D}(\boldsymbol{\alpha}) \quad \text{where} \quad \mathcal{D}(\boldsymbol{\alpha}) = \frac{1}{m}\sum_{i=1}^{m}\alpha_i - \frac{1}{2\lambda\, m^2}\left\|\sum_i \alpha_i y_i \mathbf{x}_i\right\|^2$$

Decomposition Methods

- Dual problem has a different variable for each example
- $\Rightarrow$ can optimize over subset of variables at each iteration
- Extreme case
  - Dual Coordinate Ascent (DCA) – optimize $\mathcal{D}$ w.r.t. a single variable at each iteration
  - SMO – optimize over 2 variables (necessary when having a bias term)

Linear convergence for decomposition methods

- General convergence theory of (Luo and Tseng '92) implies linear convergence
- But, dependence on m is quadratic. Therefore

$$T = O(m^2 \log(1/\rho))$$

- This implies the Machine Learning analysis

$$T = O(B^4/\epsilon^4)$$

- Why SGD is much better than decomposition methods ?
  - Primal vs. dual ?
  - Stochastic ?

Stochastic Dual Coordinate Ascent

The stochastic DCA algorithm

- Initialize $\boldsymbol{\alpha} = (0, \ldots, 0)$ and $\mathbf{w} = \mathbf{0}$
- For $t = 1, 2, \ldots, T$
  - Choose $i \in [m]$ uniformly at random
  - Update: $\alpha_i = \alpha_i + \tau_i$ where
    $$\tau_i = \max\left\{-\alpha_i\,,\, \min\left\{1-\alpha_i,\, \frac{\lambda\, m\, (1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle)}{\|\mathbf{x}_i\|^2}\right\}\right\}$$
  - Update: $\mathbf{w} = \mathbf{w} + \frac{\tau_i}{\lambda\, m}\, y_i \mathbf{x}_i$

- Hsieh et al showed encouraging empirical results
- No satisfactory theoretical guarantee

Analysis of stochastic DCA

Theorem (S '08)

With probability at least $1-\delta$*, the accuracy of stochastic DCA satisfies*

$$\rho \leq \frac{8 \ln(1/\delta)}{T} \left(\frac{1}{\lambda} + m \right)$$

Analysis of stochastic DCA

Theorem (S '08)

With probability at least $1-\delta$, the accuracy of stochastic DCA satisfies

$$\rho \leq \frac{8 \ln(1/\delta)}{T} \left(\frac{1}{\lambda} + m \right)$$

Proof idea:

- Let $\boldsymbol{\alpha}^\star$ be optimal dual solution
- Upper bound dual sub-optimality at round t by the double potential

$$\frac{1}{2\,\lambda\,m} \mathbb{E}_i \left[\|\boldsymbol{\alpha}^t - \boldsymbol{\alpha}^\star\|^2 - \|\boldsymbol{\alpha}^{t+1} - \boldsymbol{\alpha}^\star\|^2 \right] + \mathbb{E}_i \left[\mathcal{D}(\boldsymbol{\alpha}^{t+1}) - \mathcal{D}(\boldsymbol{\alpha}^t) \right]$$

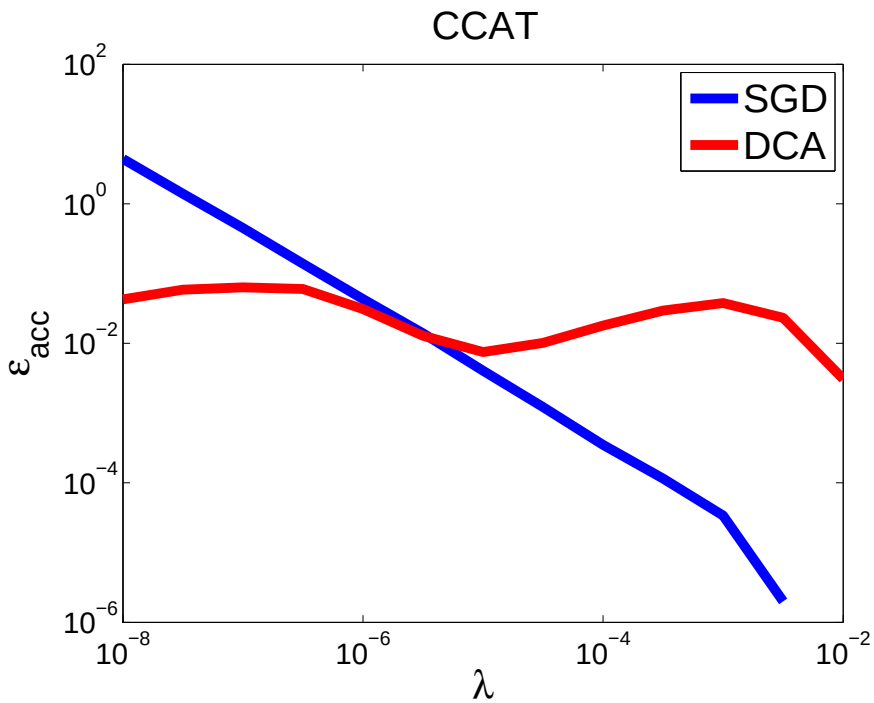
- Sum over t, use telescoping, and bound the result using weak-duality
- Use approximated duality theory (Scovel, Hush, Steinwart '08)
- Finally, use measure concentration techniques

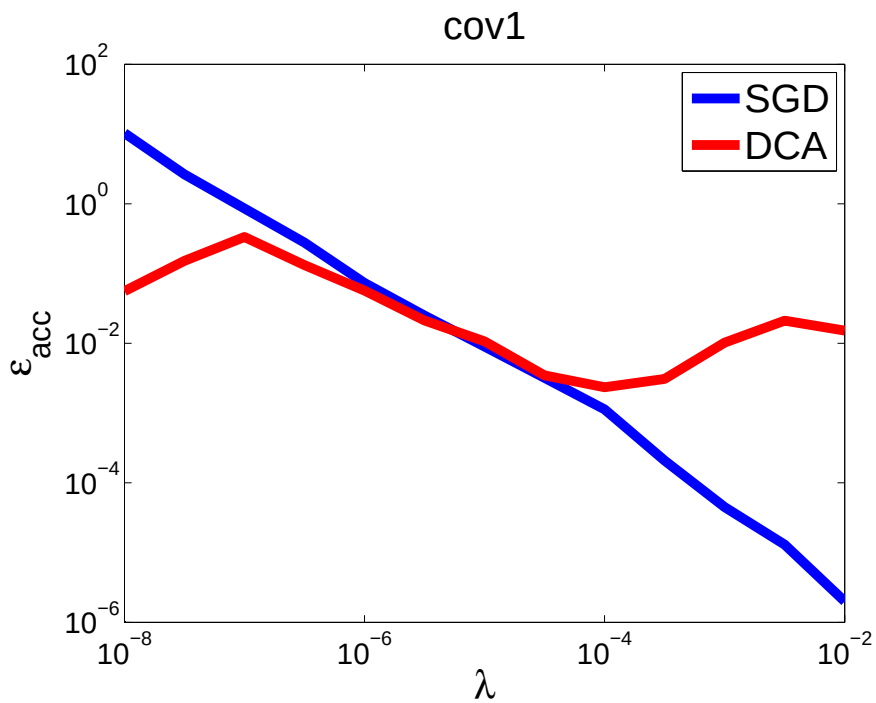
Comparing SGD and DCA

$$\text{SGD}: \quad \rho(m,T,\lambda) \leq \frac{1}{T}\frac{\log(T)}{\lambda}$$

$$\text{DCA}: \quad \rho(m,T,\lambda) \leq \frac{1}{T}\left(\frac{1}{\lambda}+m\right)$$

Conclusion: Relative performance depends on $\lambda m \overset{?}{<} \log(T)$





Combining SGD and DCA ?

- The above graphs raise the natural question: Can we somehow combine SGD and DCA ?
- Seemingly, this is impossible as SGD is a **primal** algorithm while DCA is a **dual** algorithm
- Interestingly, SGD can be viewed also as a dual algorithm, but with a dual function that changes along the optimization process
- This is an ongoing direction ...

Machine Learning analysis of DCA

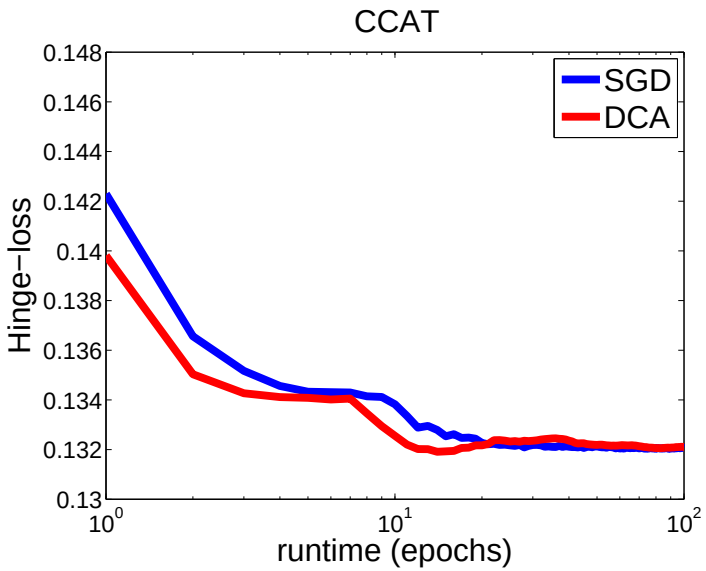
- So far, we compared SGD and DCA using the old way (ρ)
- But, what about runtime as a function of ϵ and B ?
- Similarly to previous derivation (and ignoring log terms)

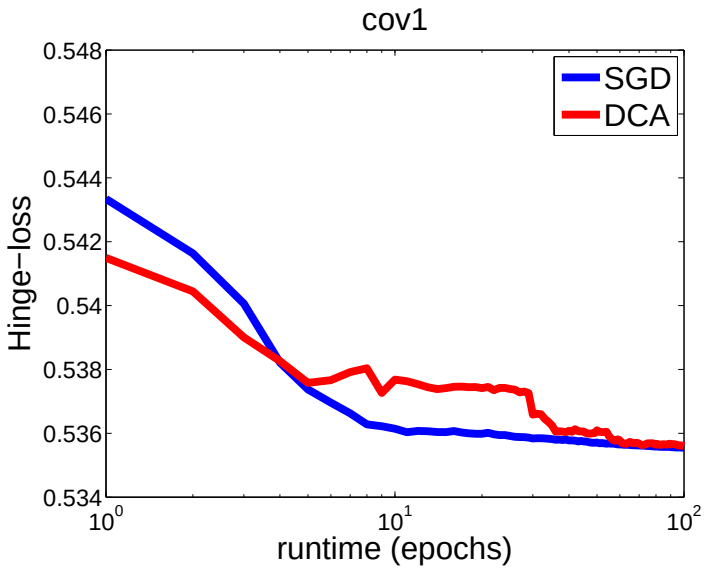
$$\text{SGD}: \quad T \leq \frac{B^2}{\epsilon^2}$$

$$\text{DCA}: \quad T \leq \frac{B^2}{\epsilon^{\mathbf{3}}}$$

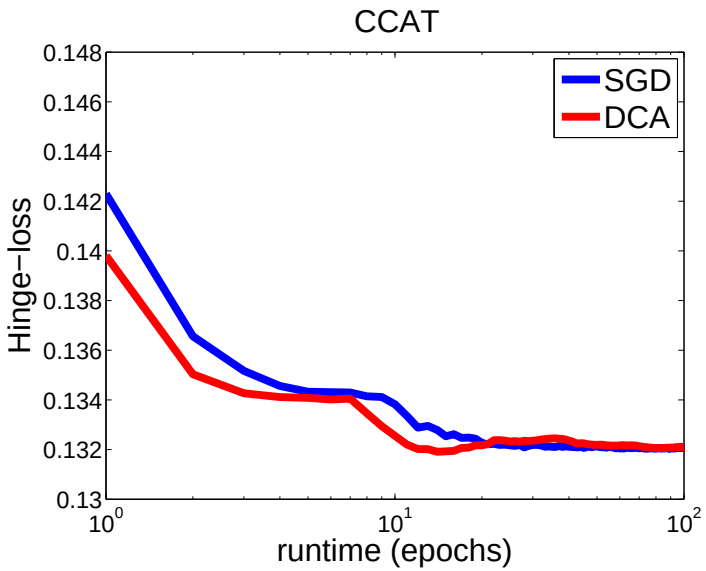
- Is this really the case ?

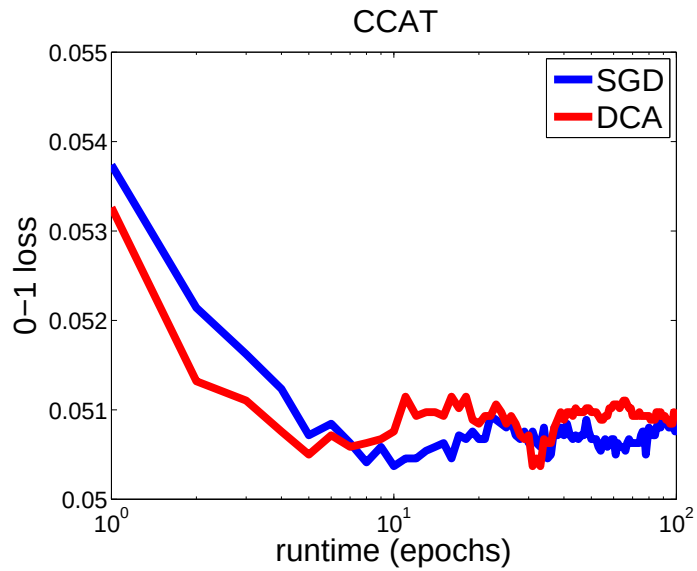
SGD vs. DCA – Machine Learning Perspective

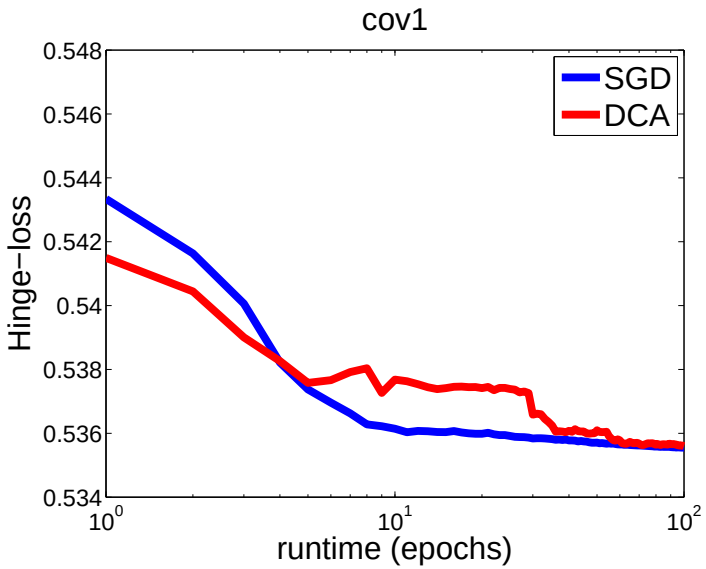


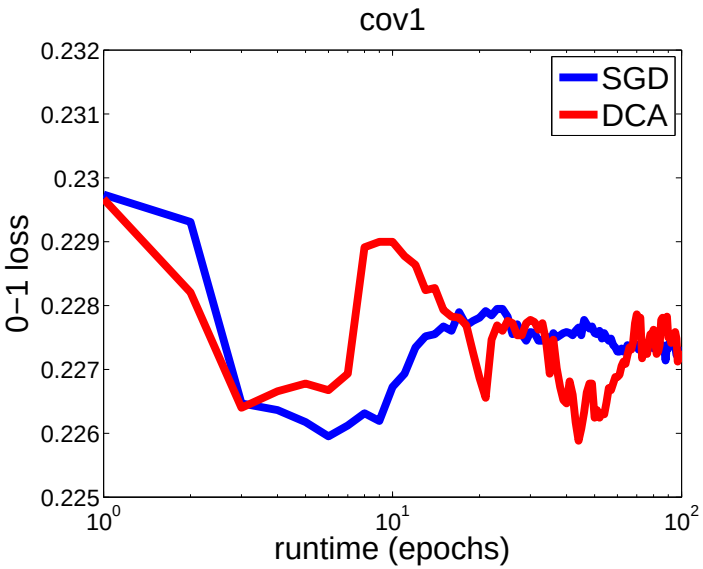


SGD vs. DCA – Machine Learning Perspective









Analysis of DCA revisited

- DCA analysis $T \leq \frac{1}{\lambda \rho} + \frac{m}{\rho}$
- First term is like in SGD while second term involves training set size. This is necessary since each dual variable has only $1/m$ effect on $\mathbf{w}$.
- However, a more delicate analysis is possible:

Theorem (DCA refined analysis)

If $T \geq m$ then with high probability at least one of the following holds true:

- *After a single epoch DCA satisfies* $L(\tilde{\mathbf{w}}) \leq \min_{\mathbf{w}:\|\mathbf{w}\| \leq B} L(\mathbf{w})$
- *DCA converges in time* $\rho \leq \frac{c}{T-m} \left(\frac{1}{\lambda} + \lambda m B^2 + B\sqrt{m} \right)$

The above theorem implies $T \leq O(B^2/\epsilon^2)$.

Discussion

- Bottou and Bousquet initiated a study of approximated optimization from the perspective of generalization error
- We further develop this idea
  - Regularized loss (like SVM)
  - Comparing algorithms based on runtime for achieving certain generalization error
  - Comparing algorithms in the data-laden regime
  - More data $\Rightarrow$ less work
- Two stochastic approaches are close to optimal
- Best methods are extremely simple :-)

Limitations and Open Problems

- Analysis is based on upper bounds of estimation and optimization error
- The online-to-batch analysis gives the same bounds for one epoch over the data (No theoretical explanation when we need more than one pass)
- We assume constant runtime for each inner product evaluation (holds for linear kernels). How to deal with non-linear kernels ?
  - Sampling ?
  - Smart selection (online learning on a budget ? Clustering ?)
- We assume λ is optimally chosen. Incorporating the runtime of tuning λ in the analysis ?
- Assumptions on distribution (e.g. Noise conditions) $\stackrel{?}{\Rightarrow}$ Better analysis
- A more general theory of optimization from a machine learning perspective