SVM Optimization: An Inverse Dependence on Data Set Size

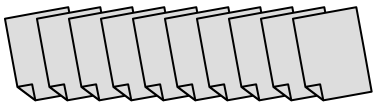
Shai Shalev-Shwartz Nati Srebro

Toyota Technological Institute—Chicago

(a philanthropically endowed academic computer science institute dedicated to basic research and graduate education in computer science)

More Data $\Rightarrow$ More Work?

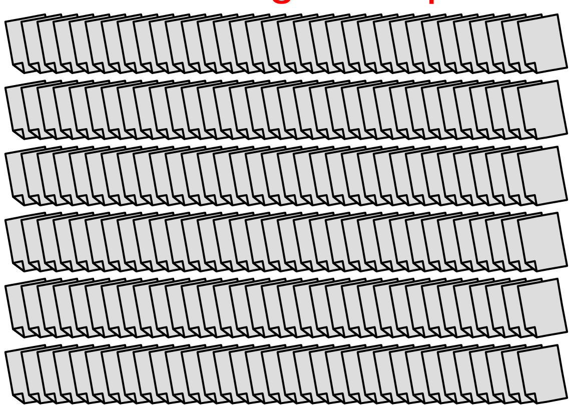
10k training examples



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:

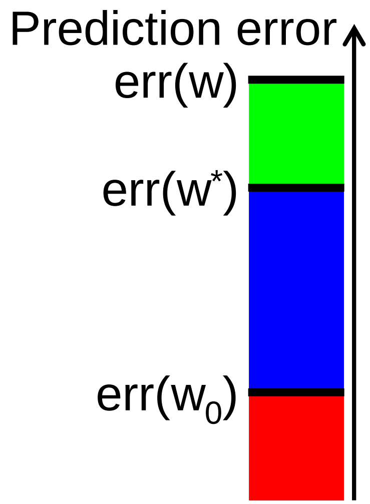
$$f(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n}[1 - y_i\langle\mathbf{w}, \mathbf{x}_i\rangle]_+$$

- True objective: prediction error

  err(w) = $\mathbf{E}_{x,y}$[error of ⟨w,x⟩ vs. y]

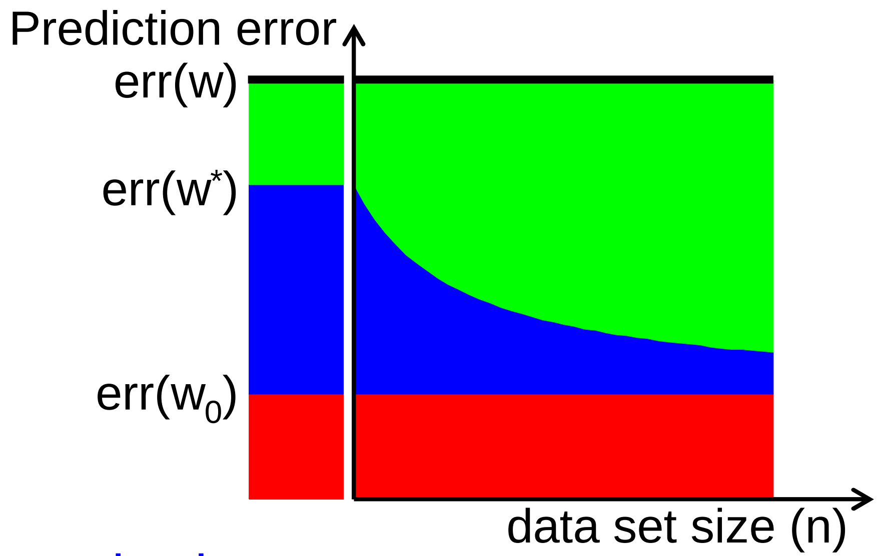
- Would like to understand computational cost in terms of:
- **Increasing** function of:
  - Desired generalization performance (i.e. as err(w) decreases)
  - Hardness of problem:
    margin, noise (unavoidable error)
- **Decreasing** function of available data set size

Error Decomposition



- Approximation error:
  - Best error achievable by large-margin predictor
  - Error of population minimizer
    $w_0 = \text{argmin } E[f(w)] = \text{argmin } \lambda|w|^2 + E_{x,y}[\text{loss}(\langle w,x \rangle;y)]$
- Estimation error:
  - Extra error due to replacing E[loss] with empirical loss
    $w^* = \text{arg min } f_n(w)$
- 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

$$f(\mathbf{w}) = \lambda\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\left[1 - y_i\langle\mathbf{w}, \mathbf{x}_i\rangle\right]_+$$

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

subgradient of
$\lambda|w|^2+[1-y_i<w,x_i>]_+$

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{1}{2\lambda t}\nabla$$

- Theorem: After at most $\tilde{O}\left(\frac{1}{\lambda\,\epsilon}\right)$ iterations, $E[f(w_{PEGASOS})] \leq \min_w f(w)+\varepsilon$
- With d-dimensional (or d-sparse) features, each iteration takes time O(d)
- **Conclusion**: Run-time required for PEGASOS to find ε accurate solution:

$$\tilde{O}\left(\frac{d}{\lambda\,\epsilon}\right)$$

- Run-time does not depend on #examples

Comparison of Runtime Guarantees

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

- Runtime to get ε_{acc}-accurate solution: $f(w) \leq \min f(w) + \varepsilon_{acc}$

PEGASOS	$d / (\lambda\, \varepsilon_{acc})$
SVMPerf	$n\, d / (\lambda\, \varepsilon_{acc})$
Dual Decomposition (SMO)	$n^2\, d \log(1/\varepsilon_{acc})$
Interior Point	$n^{3.5} \log(\log(1/\varepsilon_{acc}))$

(ignoring log-factors)

- 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

Comparison of Runtime Guarantees

large margin $M=1/|w_0|$

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

$$
\begin{aligned}
err(w) &= err(w_0) + \lambda(|w_0|^2-|w|^2) + E[f(w)]-E[f(w_0)] \\
&\leq err(w_0) + \lambda|w_0|^2 + 2(f(w)-f(w_0)) + O(1/(\lambda n)) \\
&\leq err(w_0) + \lambda|w_0|^2 + 2\varepsilon_{acc} + O(1/(\lambda n))
\end{aligned}
$$

$$\lambda|w_0|^2 \leq O(\varepsilon) \qquad 2\varepsilon_{acc} \leq O(\varepsilon) \qquad O(1/(\lambda n)) \leq O(\varepsilon)$$

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

Comparison of Runtime Guarantees

large margin $M=1/|w_0|$

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	$n^2\, d \log(1/\varepsilon_{acc})$
SVMPerf	$n\, d / (\lambda\, \varepsilon_{acc})$
PEGASOS	$d / (\lambda\, \varepsilon_{acc})$

(ignoring log-factors)

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

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

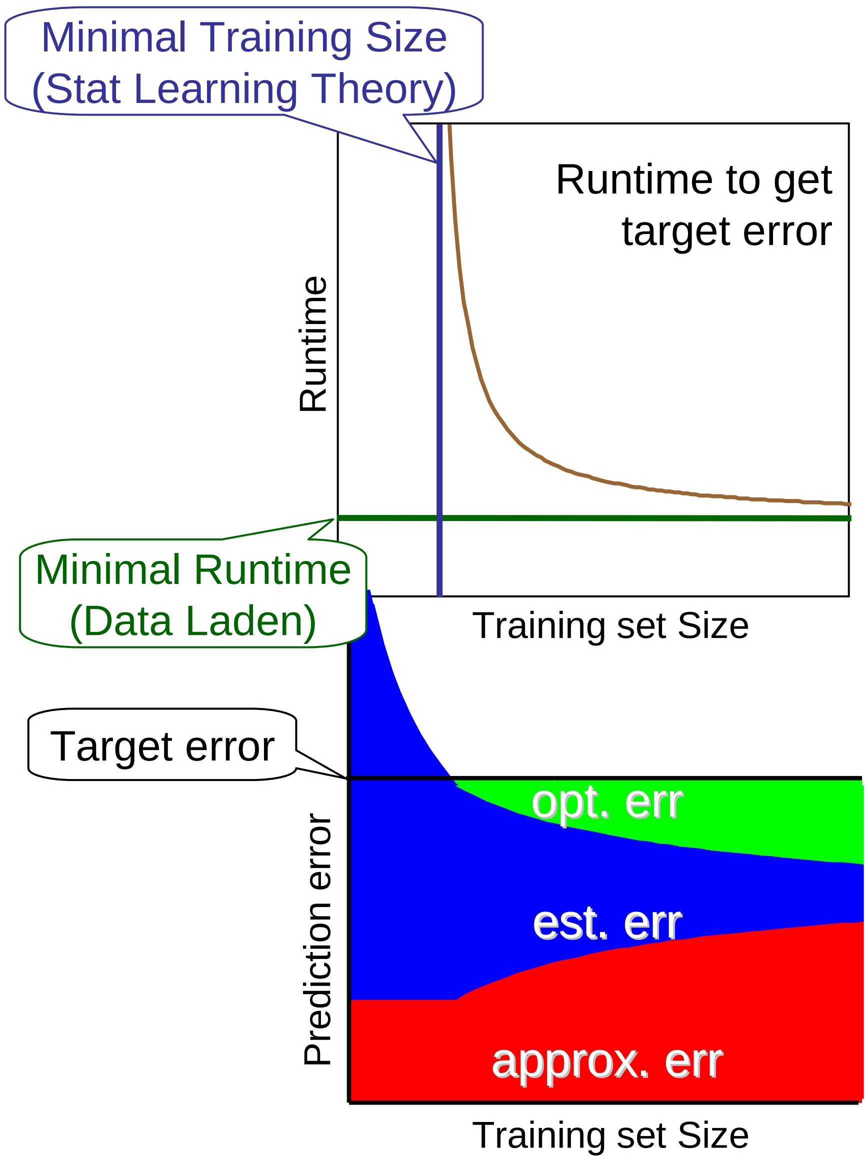
$\varepsilon_{acc} = O(\varepsilon)$

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

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 err(w_0)+ε with n training points:

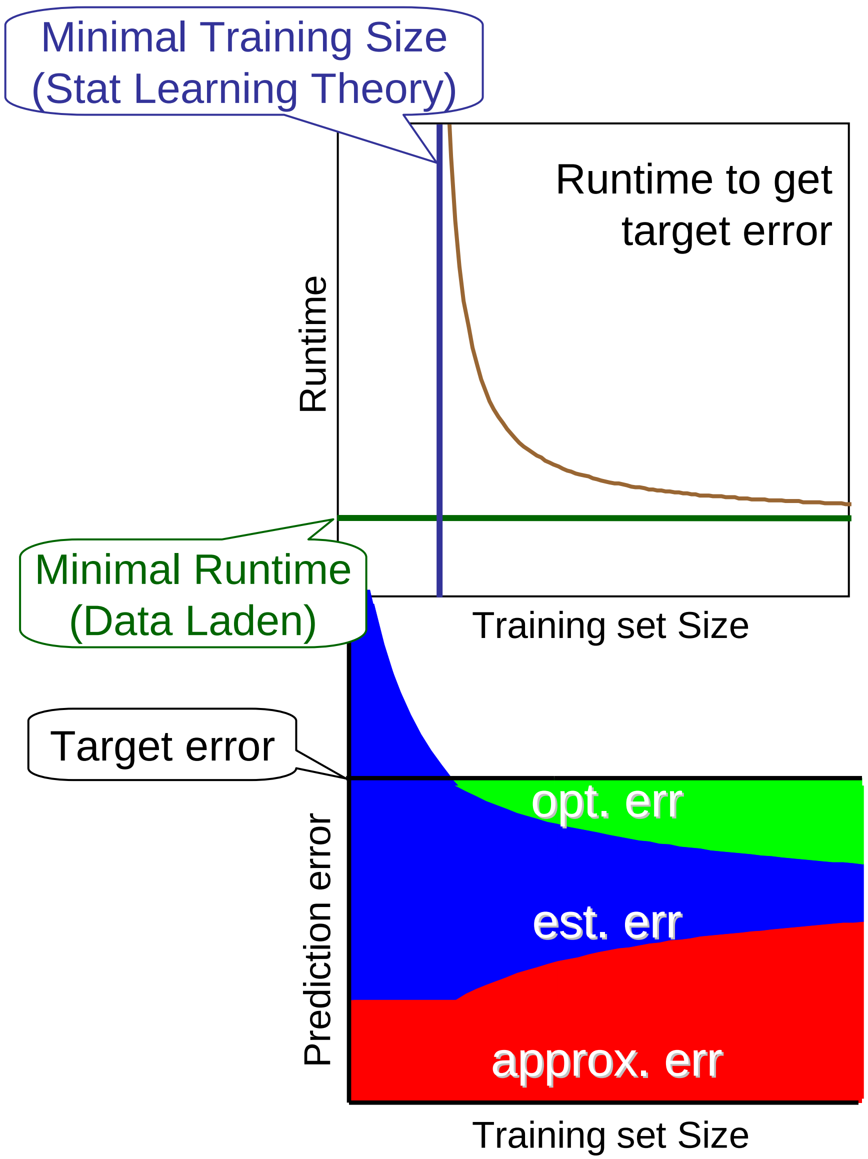
$$T = \tilde{\Omega}\left(\frac{d}{\left(\epsilon M - O\left(\frac{1}{\sqrt{n}}\right)\right)^2}\right)$$

Increases for smaller target error

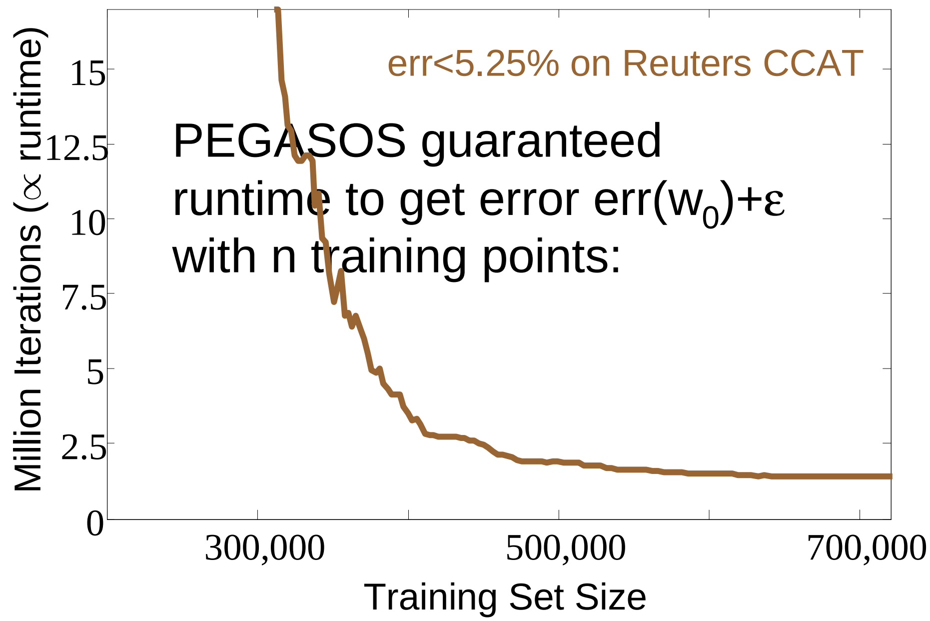
Increases for smaller margin

Decreases for larger data set

Dependence on Data Set Size



PEGASOS guaranteed runtime to get error $err(w_0)+\varepsilon$ with n training points:

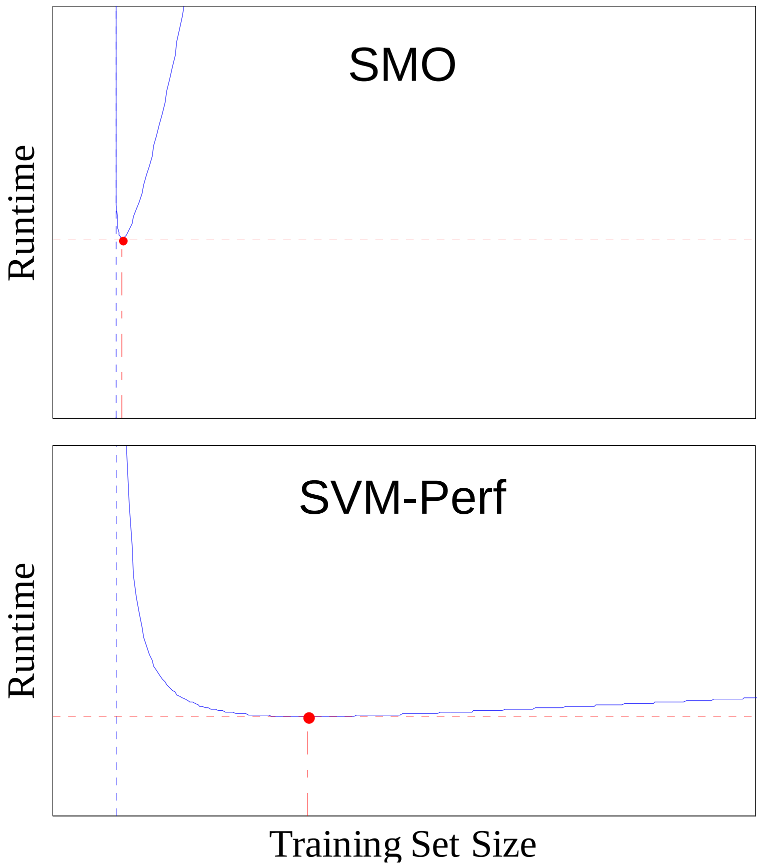


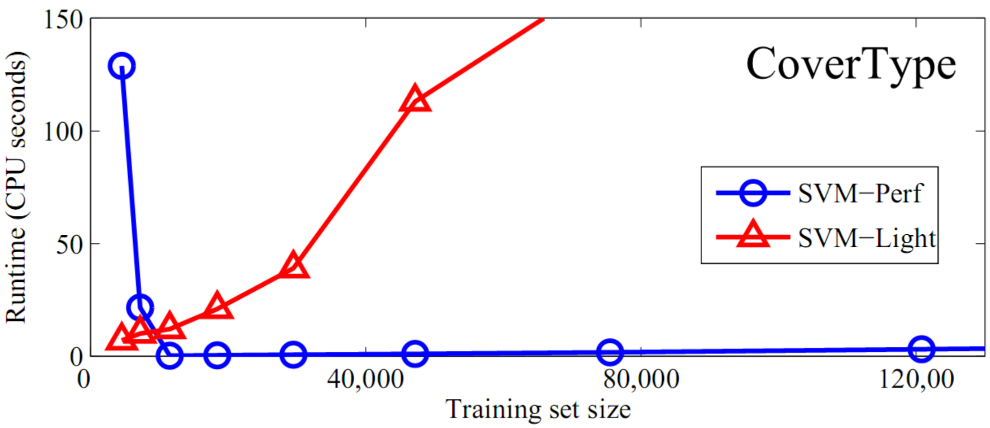
$err(w) \leq err(w_0) + \lambda|w_0|^2 + O(1/(\lambda n)) + O(d/(\lambda T))$

Increase λ as training size increases!

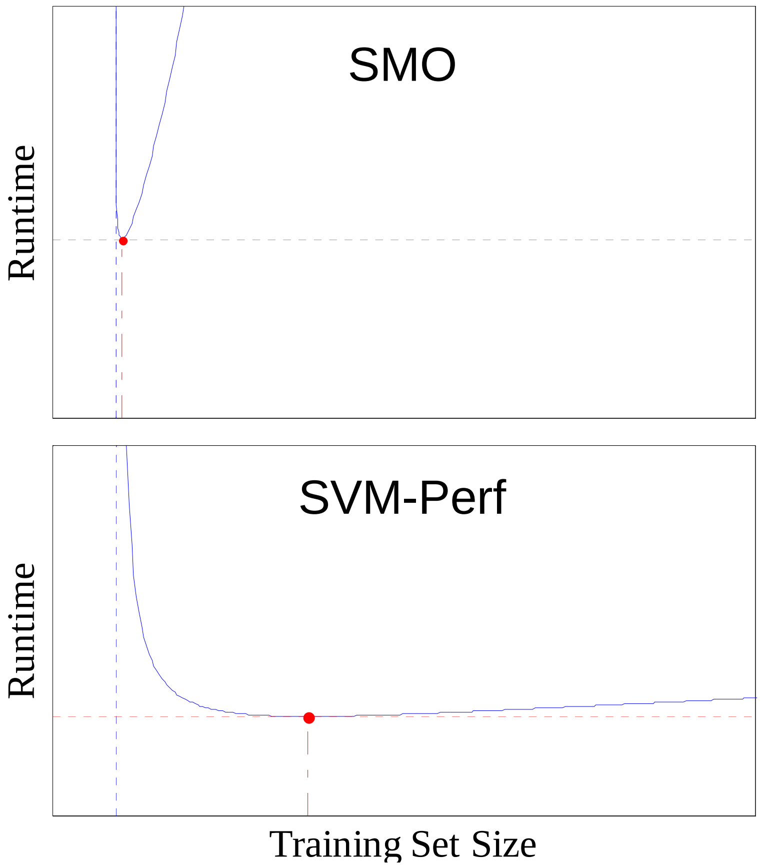
- More regularization, less predictors allowed
- Larger approximation error $err(w_0)+\lambda|w_0|^2$
- Faster runtime $T \propto 1/\lambda$

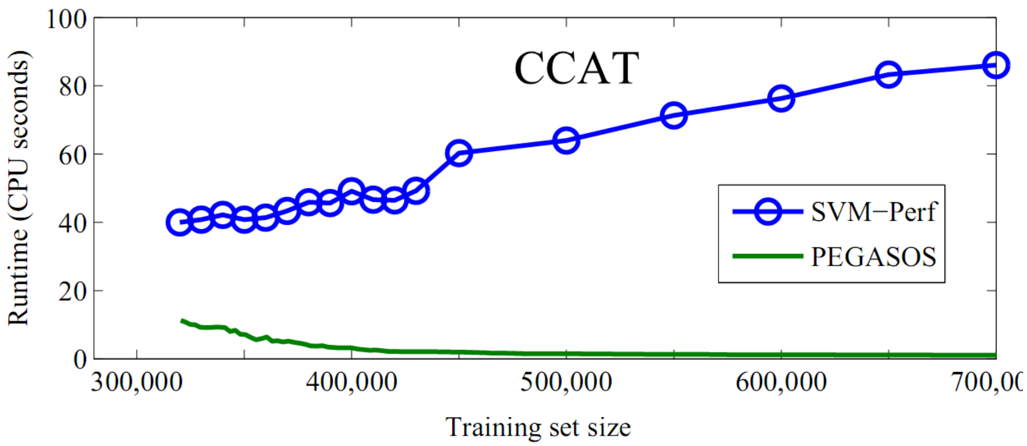
Dependence on Data Set Size: Traditional Optimization Approaches





Dependence on Data Set Size: Traditional Optimization Approaches



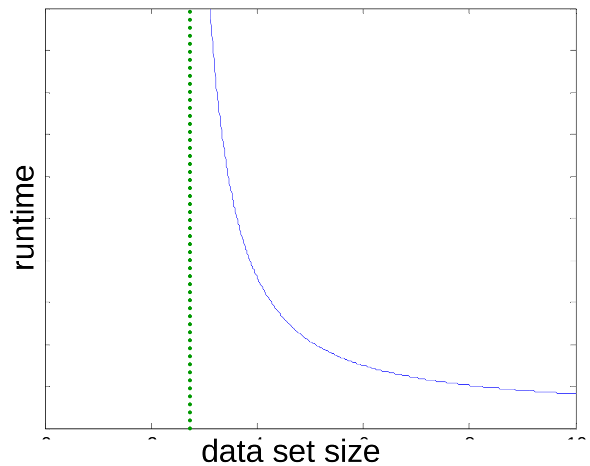


Beyond PEGASOS

- 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
- **Kernalized SVMs** (i.e. given access to a non-linear kernel):
  - Stochastic sub-gradient descent applicable, but runtime to get fixed ε_{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, const $\times$ minimum-sample-complexity is enough to get to const $\times$ 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]