

# More Data, Less Work: SVM Training in Decreasing Runtime

(and also a few words about clustering)

Nati Srebro

Toyota Technological Institute—Chicago

*(an independent philanthropically-endowed academic computer science institute located on the University of Chicago campus)*

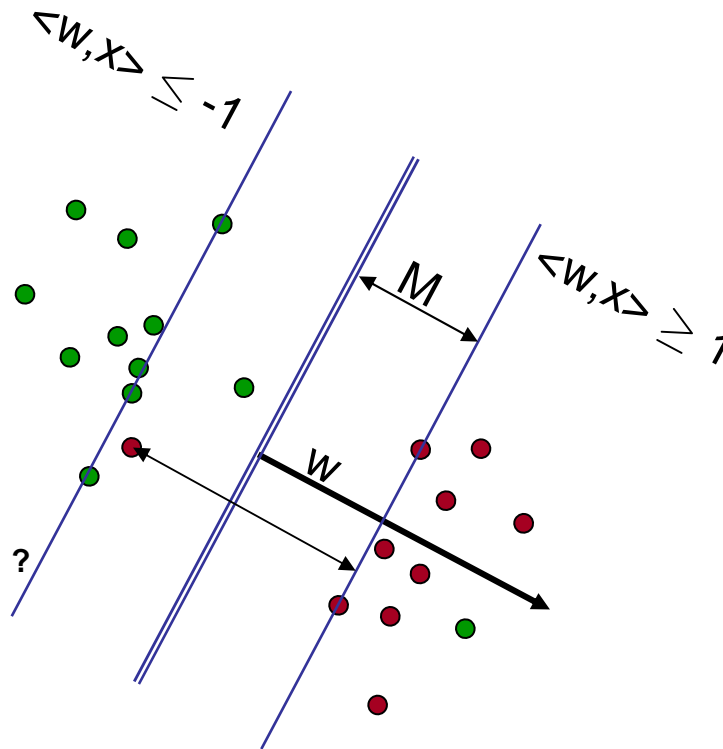
- SVM Optimization: Inverse Dependence on Training Set Size  
Shai Shalev-Shwartz (TTI), N Srebro, ICML'08 *best paper award*
- Pegasos: Primal Estimated sub-GrAdient SOLver for SVM  
Shai Shalev-Shwartz (TTI), Yoram Singer (Google), N Srebro, ICML'07
- An Investigation of Comp. and Informational Limits in Gaussian Mixture Clustering  
N Srebro, Greg Shakhnarovich (TTI), Sam Roweis (Google/Toronto), ICML'06

# Large Margin Linear Classification

aka  $L_2$ -regularized Linear Classification

aka Support Vector Machines

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



Margin:  $M = 1/|\mathbf{w}|$

Error:  $[1 - y \langle \mathbf{w}, \mathbf{x} \rangle]_+$

# SVM Training as an Optimization Problem

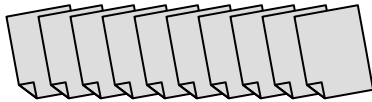
$$f(\mathbf{w}) = \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{n} \sum_{i=1}^n [1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle]_+$$

- IP method on dual (standard QP solver):  
 $O(n^4 \log \log(1/\epsilon))$
- Dual decomposition methods (e.g. SMO):  
 $O(n^2 d \log(1/\epsilon))$  [Platt 98][Joachims 98][Lin 02]
- Primal cutting plane method (SVMperf):  
 $O(nd / (\lambda\epsilon))$  [Joachims 06][Smola et al 08]

Runtime to get  $f(\mathbf{w}) \leq \min f(\mathbf{w}) + \epsilon$

# More Data $\Rightarrow$ More Work?

10k training examples

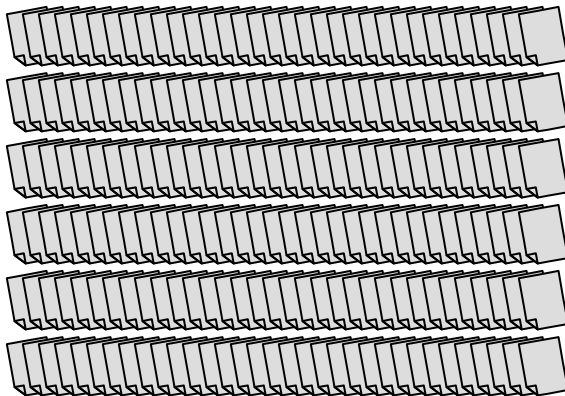


1 hour



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:

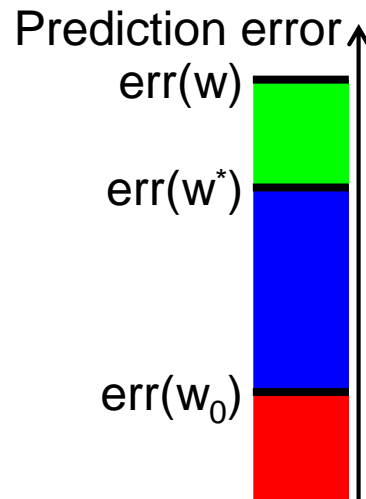
$$f(\mathbf{w}) = \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{n} \sum_{i=1}^n [1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle]_+$$

- True objective: prediction error

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

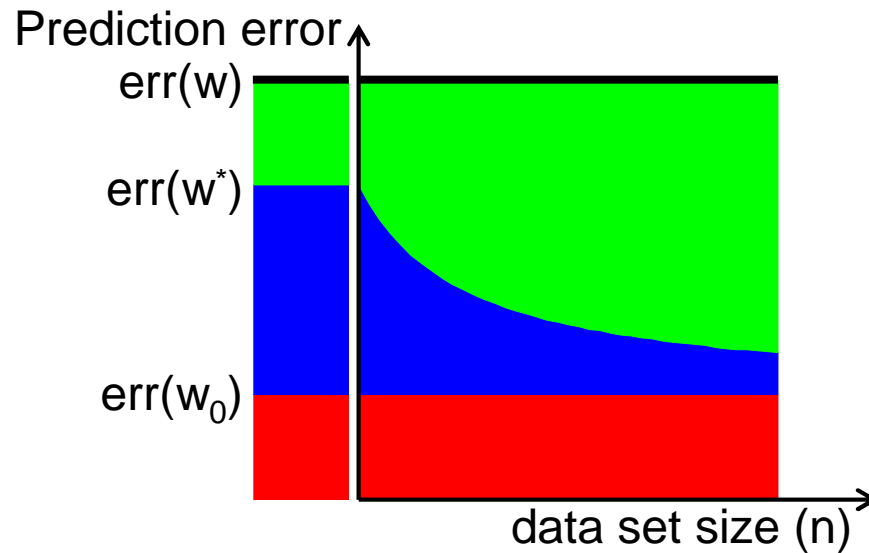
- Would like to understand computational cost in terms of:
- Increasing function of:
  - Desired generalization performance (i.e. as  $\text{err}(\mathbf{w})$  decreases)
  - Hardness of problem:  
margin, noise (unavoidable error)
- Decreasing function of available data set size




# Error Decomposition



- **Approximation error:**
  - Best error achievable by large-margin predictor
  - Error of population minimizer  
 $w_0 = \operatorname{argmin} E[f(w)] = \operatorname{argmin} \lambda|w|^2 + E[\operatorname{loss}(w)]$
- **Estimation error:**
  - Extra error due to replacing  $E[\operatorname{loss}]$  with empirical loss  
 $w^* = \operatorname{arg} \min f_n(w)$
- **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  $\Rightarrow$  fewer iterations 
  - But handling more data is expensive  
e.g. runtime of each iteration increases 
- PEGASOS (Primal Efficient Sub-Gradient Solver for SVMs)  
[Shalev-Shwartz Singer S 07]
  - Fixed runtime per iteration
  - Runtime to get fixed accuracy does not increase with n 

# PEGASOS: Stochastic (sub-)Gradient Descent

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

- Initialize  $\mathbf{w}=0$

- At each iteration  $t$ ,

with random data point  $(\mathbf{x}_i, y_i)$ :

$$\nabla = 2\lambda \mathbf{w} - \begin{cases} y_i \mathbf{x}_i & \text{if } y_i \langle \mathbf{w}, \mathbf{x}_i \rangle < 1 \\ 0 & \text{otherwise} \end{cases}$$

subgradient of  
 $\lambda \|\mathbf{w}\|^2 + [1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle]_+$

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

- **Theorem:** After at most  $\tilde{O}\left(\frac{1}{\delta \lambda \epsilon}\right)$  iterations,  $f(\mathbf{w}_{\text{PEGASOS}}) \leq \min_{\mathbf{w}} f(\mathbf{w}) + \epsilon$ , with probability  $\geq 1 - \delta$
- With  $d$ -dimensional (or  $d$ -sparse) features, each iteration takes time  $O(d)$
- **Conclusion:** Run-time required for PEGASOS to find  $\epsilon$  accurate solution with constant probability:

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

- Run-time does not depend on #examples



# Training Time (in seconds)

	Pegasos	SVM-Perf [Joachims06]	SVM-Light [Joachims]
Reuters CCAT (800K examples, 47k features)	<b>2</b>	77	20,075
Covertypes (581k examples, 54 features)	<b>6</b>	85	25,514
Physics ArXiv (62k examples, 100k features)	<b>2</b>	5	80

# Runtime Analyzis

Traditional

$$f(w) < f(w^*) + \epsilon_{\text{acc}}$$

Data Laden:

$$\text{err}(w) \leq \text{err}(w_0) + \epsilon$$

SMO

$$n^2 d \log(1/\epsilon_{\text{acc}})$$

$$d |w_0|^4 / \epsilon^4$$

SVMPperf

$$n d / (\lambda \epsilon_{\text{acc}})$$

$$d |w_0|^4 / \epsilon^4$$

PEGASOS

$$d / (\lambda \epsilon_{\text{acc}})$$

$$d |w_0|^2 / \epsilon^2$$

*(ignoring log-factors)*

large margin  $M = 1/|w_0|$

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

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

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

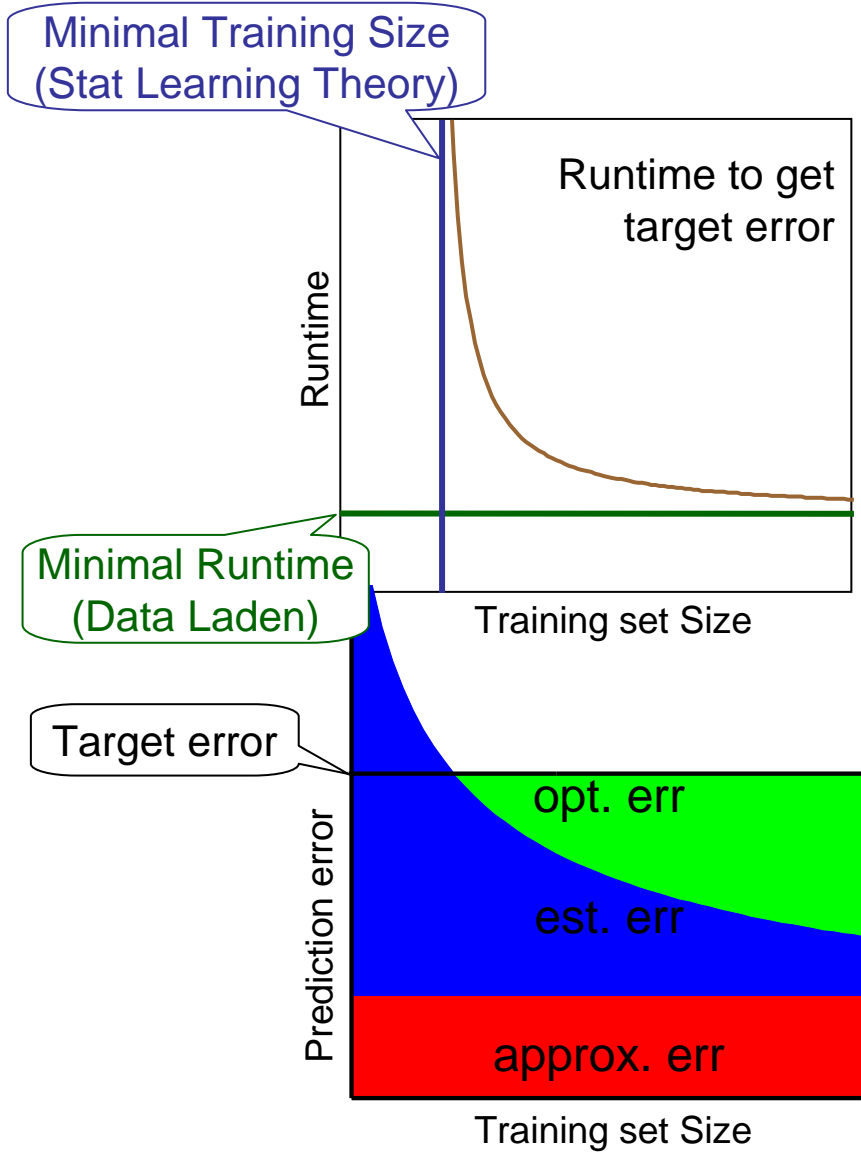
$$\epsilon_{\text{acc}} = O(\epsilon)$$

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

Unlimited data available, can  
choose working data-set size

**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:

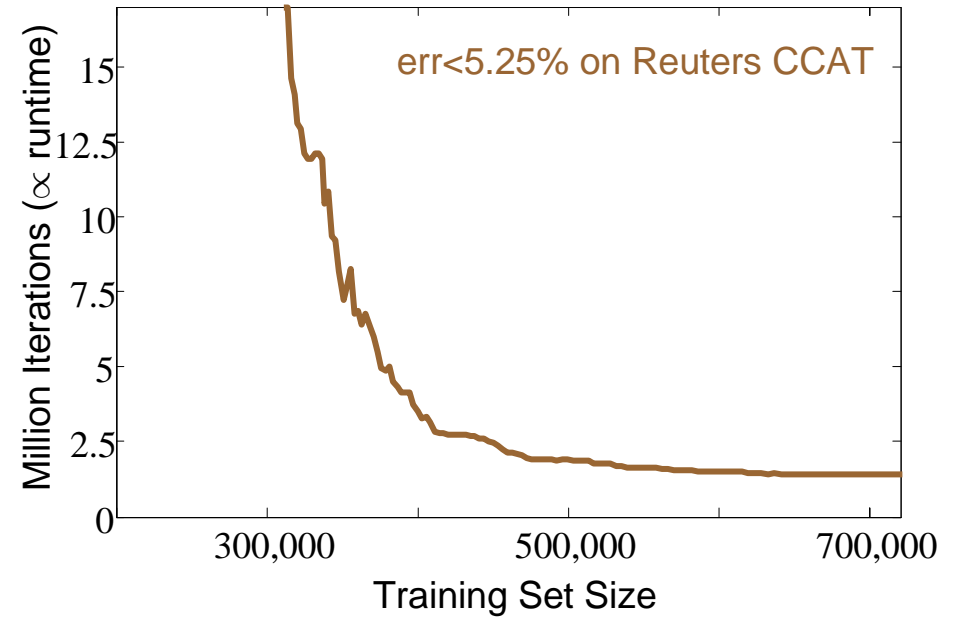
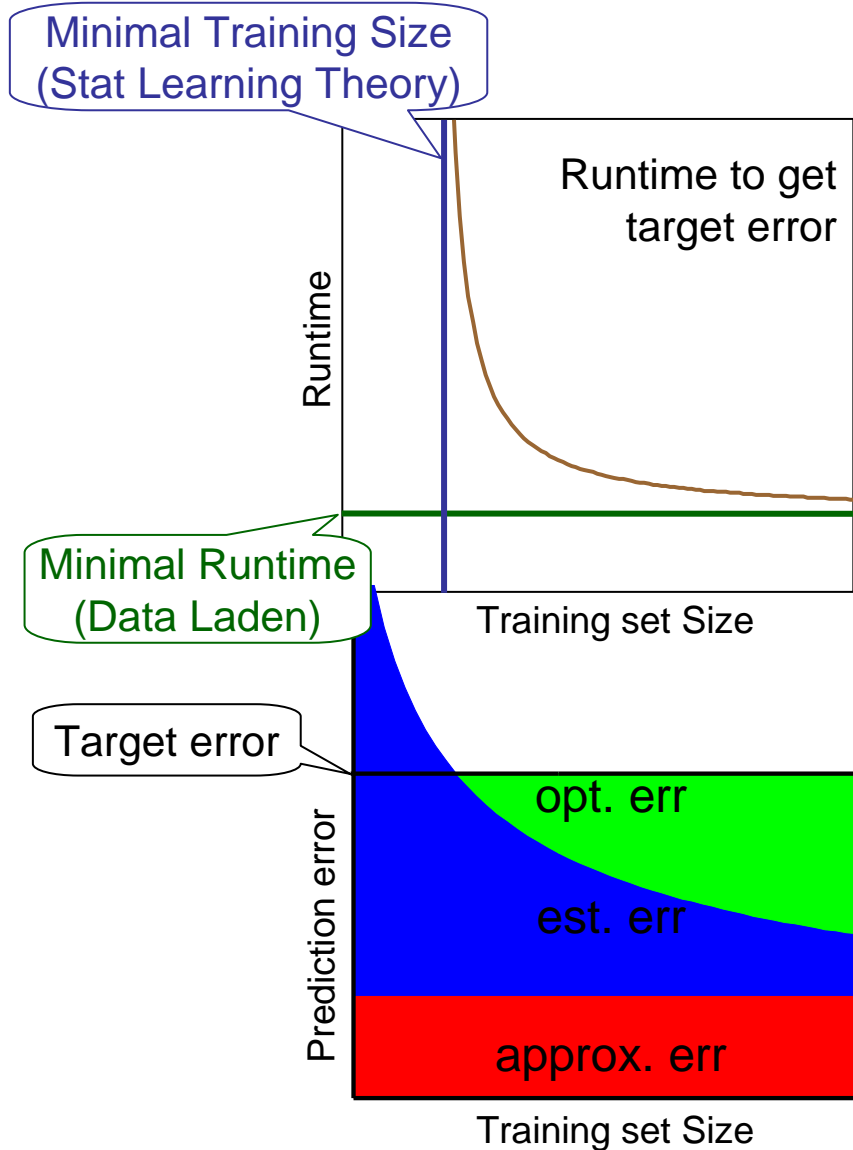
$$T = \tilde{\Omega} \left( \frac{d}{\left( \epsilon M - O\left(\frac{1}{\sqrt{n}}\right) \right)^2} \right)$$

Increases for smaller target error

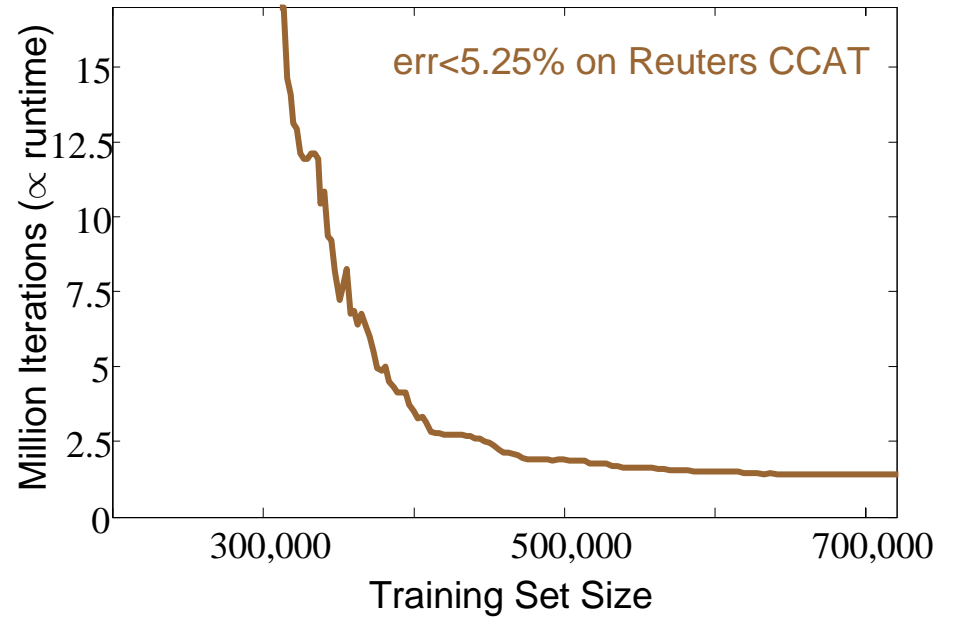
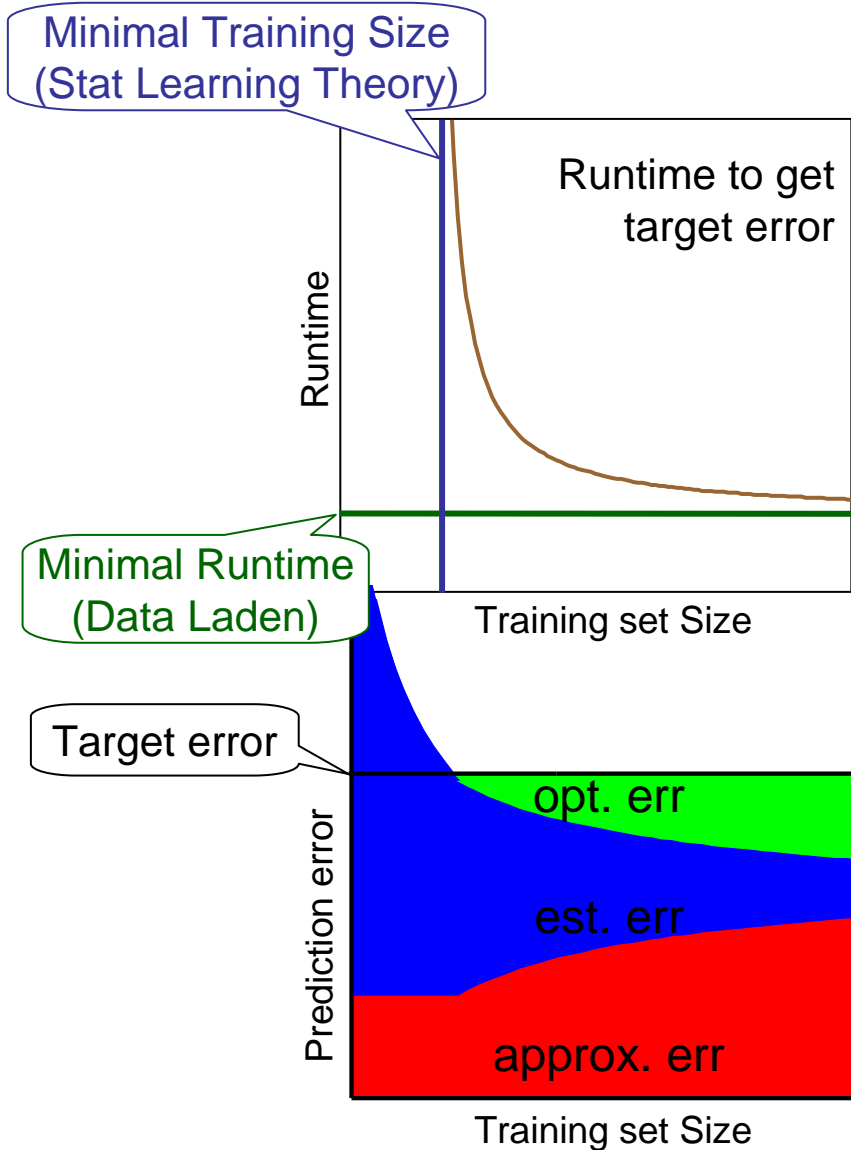
Increases for smaller margin

Decreases for larger data set

# Dependence on Data Set Size



# Dependence on Data Set Size



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

Larger  $\lambda \Rightarrow$

More regularization, less predictors allowed

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

Faster runtime  $T \propto 1/\lambda$

# Beyond PEGASOS

- PEGASOS (stochastic sub-gradient descent) effective for SVM with **linear kernel** (i.e. feature vectors given explicitly)
  - Relevant especially in text analysis, where feature vectors are sparse, very high dimensional, bags-of-words
- More generally: instead of explicit access of vectors  $x_i$ , only access to  $\langle x_i, x_j \rangle = K(i, j)$ 
  - Stochastic Sub-Gradient Descent applicable, but runtime to get fixed  $\epsilon_{\text{acc}}$  does increase linearly with  $n$
  - **Can we get similar behavior for general kernels?**
- **Can we more explicitly leverage excess data?**
  - Playing only on the decrease in estimation error, having a constant factor more samples than statistical limit gets us within constant factor of data-laden computational limit
- **Other machine learning problems...**

# Clustering

(by fitting a Gaussian mixture model)

- Clustering is hard in the worst-case
- Given LOTS of data (and enough separation):
  - Can efficiently recover true clustering  
[Dasgupta 99][Dasgupta Schulman 00][Arora Kannan 01][Vempala Wang 04]  
[Achlioptis McSherry 05][Kannan Salmasian Vempala 05]
  - EM works (empirically)
- With too little data, clustering is meaningless:
  - Even if we find the ML clustering, it has nothing to do with underlying distribution

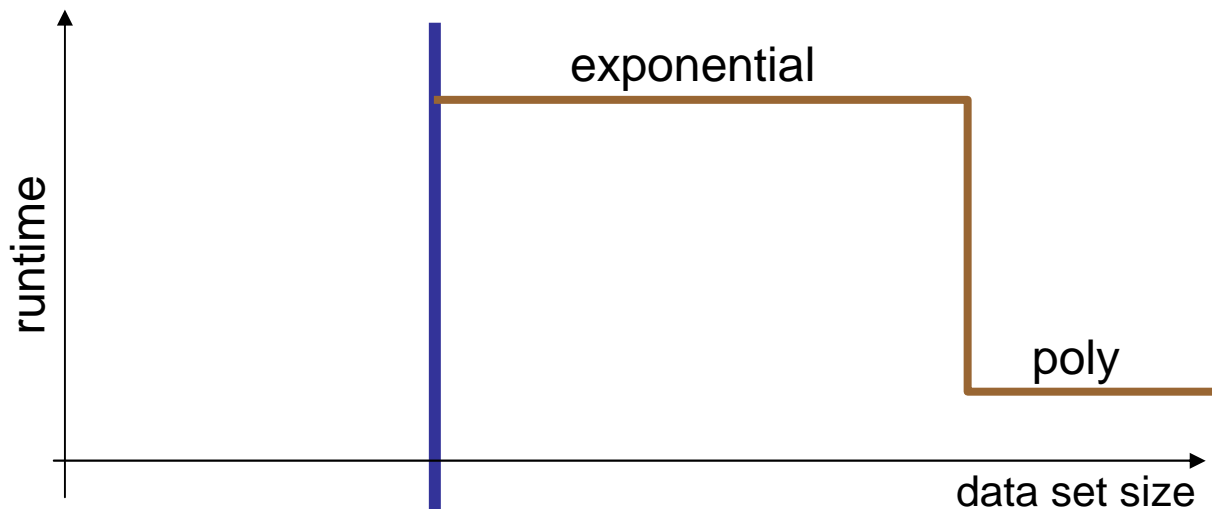
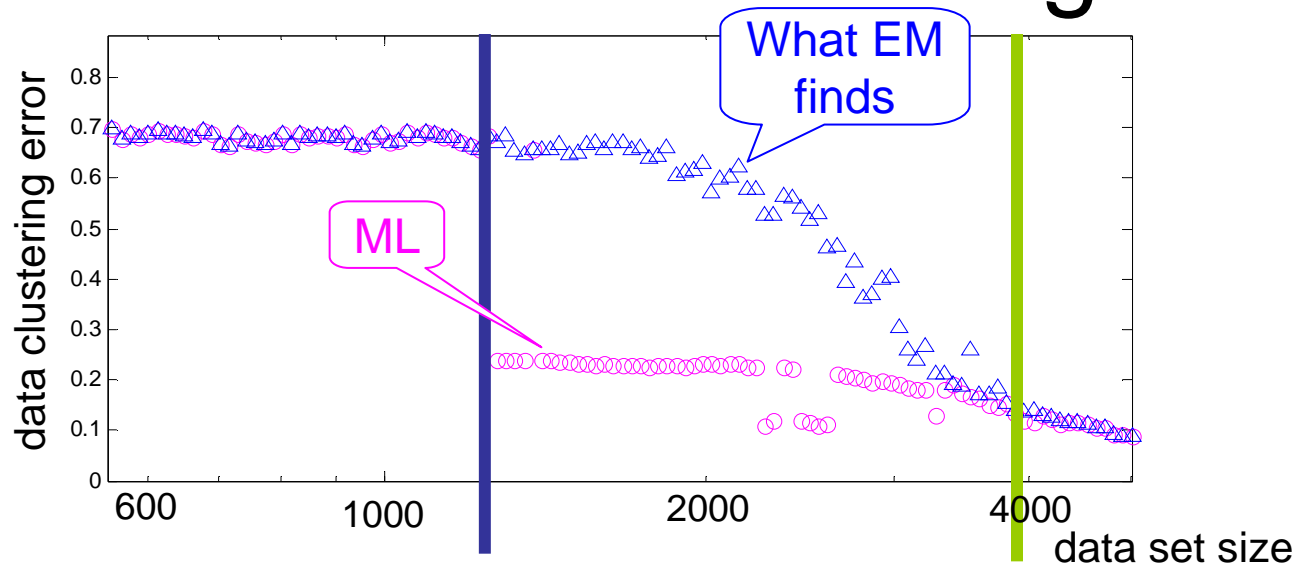
“Clustering isn’t hard—  
it’s either easy, or not interesting”

# Effect of “Signal Strength”





# Computational and Information Limits in Clustering



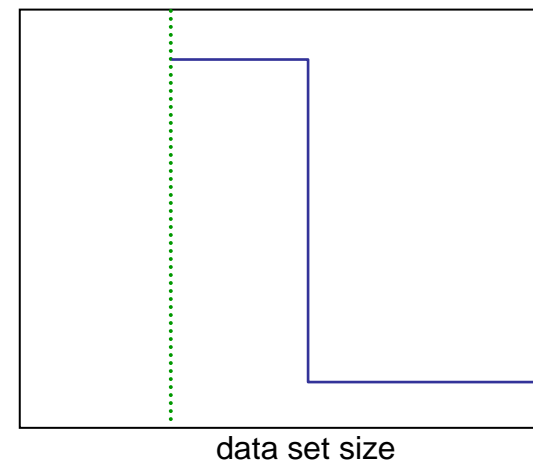
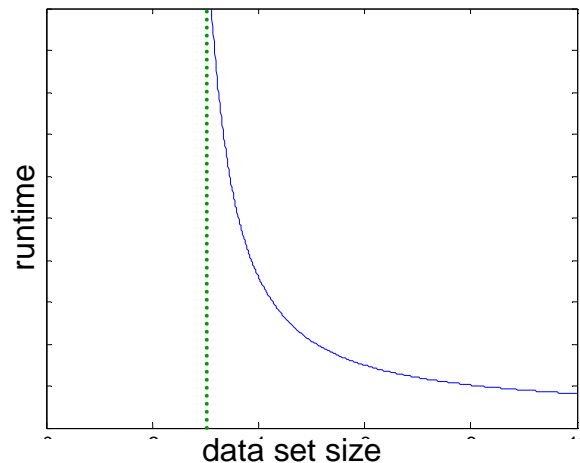
k=16 clusters in d=512 dimensions, sep=4 $\sigma$

# Informational Cost of Tractability?

- Gaussian Mixture Clustering
- Learning structure of dependency networks
  - Hard to find optimal (ML) structure in the worst case [Srebro 01]
  - Polynomial-time algorithms for the large-sample limit [Checheta Guestrin 07]
- Graph partitioning (correlation clustering)
  - Hard in the worst case
  - Easy for large graphs with a “nice” partitions [McSherry 03]
- Finding cliques in random graphs
- Planted Noisy MAX-SAT

# More Data $\Rightarrow$ Less Work

- Required runtime:
  - **increases** with complexity of the answer (separation, decision boundary)
  - **increases** with desired accuracy
  - **decreases** with amount of available data
- PEGASOS (stochastic sub-gradient descent for SVMs):
  - Runtime to get fixed optimization accuracy doesn't depend on  $n$
  - Best performance in data-laden regime
  - Runtime **decreases** as more data is available



SVM Optimization: Inverse Dependence on Training Set Size [ICML'08]

Pegasos: Primal Estimated sub-GrAdient SOLver for SVM [ICML'07]

An Investigation of Comp. and Inf. Limits in Gaussian Mixture Clustering [ICML'06]