Large Scale Machine Learning with Stochastic Gradient Descent

Léon Bottou
leon@bottou.org

Microsoft (since June)

Summary

i. Learning with Stochastic Gradient Descent.

ii. The Tradeoffs of Large Scale Learning.

iii. Asymptotic Analysis.

iv. Learning with a Single Pass.

I. Learning with Stochastic Gradient Descent

Example

Binary classification

– Patterns x.
– Classes $y = \pm 1$.

Linear model

– Choose features: $\Phi(x) \in \mathbb{R}^d$
– Linear discriminant function: $f_w(x) = \text{sign}\left(w^\top \Phi(x)\right)$

SVM training

– Choose loss function

$$Q(x, y, w) = \ell(y, f_w(x)) \quad = \quad \text{(e.g.)} \quad \log\left(1 + e^{-y\, w^\top \Phi(x)}\right)$$

– Cannot minimize the expected risk $E(w) = \int Q(x, y, w)\, dP(x, y)$.

– Can compute the empirical risk $E_n(w) = \frac{1}{n} \sum_{i=1}^{n} Q(x_i, y_i, w)$.

Minimize L_2 regularized empirical risk

$$\min_w \frac{\lambda}{2}\|w\|^2 + \frac{1}{n} \sum_{i=1}^{n} Q(x_i, y_i, w)$$

Choosing λ is the same setting a constraint $\|w\|^2 < B$.

Batch versus Online

Batch: process all examples together (GD)

– Example: minimization by gradient descent

$$\text{Repeat: } w \leftarrow w - \gamma \left(\lambda w + \frac{1}{n} \sum_{i=1}^{n} \frac{\partial Q}{\partial w}(x_i, y_i, w) \right)$$

Online: process examples one by one (SGD)

– Example: minimization by stochastic gradient descent

$$\text{Repeat: (a) Pick random example } x_t, y_t$$
$$\text{(b) } w \leftarrow w - \gamma_t \left(\lambda w + \frac{\partial Q}{\partial w}(x_t, y_t, w) \right)$$

Second order optimization

Batch: (2GD)

– Example: Newton's algorithm

$$\text{Repeat: } w \leftarrow w - H^{-1} \left(\lambda w + \frac{1}{n} \sum_{i=1}^{n} \frac{\partial Q}{\partial w}(x_i, y_i, w) \right)$$

Online: (2SGD)

– Example: Second order stochastic gradient descent

Repeat: (a) Pick random example x_t, y_t

(b) $w \leftarrow w - \gamma_t H^{-1} \left(\lambda w + \frac{\partial Q}{\partial w}(x_t, y_t, w) \right)$

More SGD Algorithms

Adaline (Widrow and Hoff, 1960)

$$Q_{\text{adaline}} = \tfrac{1}{2}\big(y - w^\top \Phi(x)\big)^2$$
$$\Phi(x) \in \mathbb{R}^d, \;\; y = \pm 1$$

$$w \leftarrow w + \gamma_t \big(y_t - w^\top \Phi(x_t)\big)\, \Phi(x_t)$$

Perceptron (Rosenblatt, 1957)

$$Q_{\text{perceptron}} = \max\{0, -y\, w^\top \Phi(x)\}$$
$$\Phi(x) \in \mathbb{R}^d, \;\; y = \pm 1$$

$$w \leftarrow w + \gamma_t \begin{cases} y_t\, \Phi(x_t) & \text{if } y_t\, w^\top \Phi(x_t) \le 0 \\ 0 & \text{otherwise} \end{cases}$$

Multilayer perceptrons (Rumelhart et al., 1986) ...

SVM (Cortes and Vapnik, 1995) ...

Lasso (Tibshirani, 1996)

$$Q_{\text{lasso}} = \lambda |w|_1 + \tfrac{1}{2}\big(y - w^\top \Phi(x)\big)^2$$
$$w = (u_1 - v_1, \ldots, u_d - v_d)$$
$$\Phi(x) \in \mathbb{R}^d, \;\; y \in \mathbb{R}, \;\; \lambda > 0$$

$$u_i \leftarrow \Big[u_i - \gamma_t \big(\lambda - (y_t - w^\top \Phi(x_t))\Phi_i(x_t)\big)\Big]_+$$
$$v_i \leftarrow \Big[v_i - \gamma_t \big(\lambda + (y_t - w_t^\top \Phi(x_t))\Phi_i(x_t)\big)\Big]_+$$
with notation $[x]_+ = \max\{0, x\}$.

K-Means (MacQueen, 1967)

$$Q_{\text{kmeans}} = \min_k \tfrac{1}{2}(z - w_k)^2$$
$$z \in \mathbb{R}^d, \;\; w_1 \ldots w_k \in \mathbb{R}^d$$
$$n_1 \ldots n_k \in \mathbb{N}, \;\; \text{initially } 0$$

$$k^* = \arg\min_k (z_t - w_k)^2$$
$$n_{k^*} \leftarrow n_{k^*} + 1$$
$$w_{k^*} \leftarrow w_{k^*} + \tfrac{1}{n_{k^*}}(z_t - w_{k^*})$$

II. The Tradeoffs of Large Scale Learning

The Computational Problem

- Baseline large-scale learning algorithm

Randomly discarding data is the simplest way to handle large datasets.

  - What is the statistical benefit of processing more data?
  - What is the computational cost of processing more data?

- We need a theory that links Statistics and Computation!

– 1967: Vapnik's theory does not discuss computation.

– 1981: Valiant's learnability excludes exponential time algorithms, but (i) polynomial time already too slow, (ii) few actual results.

Decomposition of the Error

$$E(\tilde{f}_n) - E(f^*) = E(f^*_{\mathcal{F}}) - E(f^*) \quad \text{Approximation error } (\mathcal{E}_{\text{app}})$$

$$+ E(f_n) - E(f^*_{\mathcal{F}}) \quad \text{Estimation error } (\mathcal{E}_{\text{est}})$$

$$+ E(\tilde{f}_n) - E(f_n) \quad \text{Optimization error } (\mathcal{E}_{\text{opt}})$$

Problem:

Choose $\mathcal{F}$, n, and ρ to make this as small as possible,

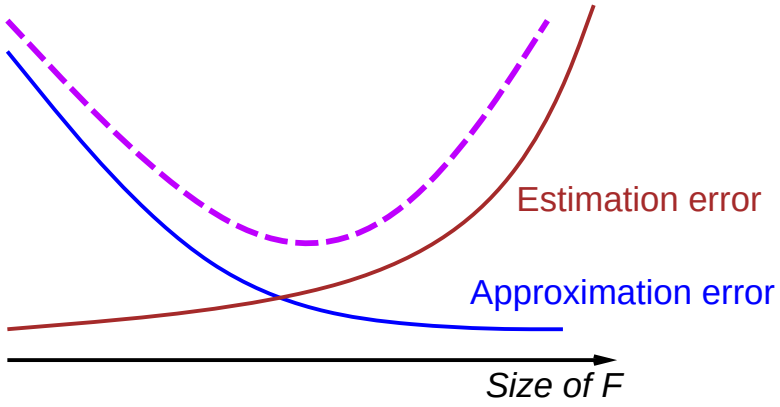
subject to budget constraints { max number of examples n / max computing time T

Note: choosing λ is the same as choosing $\mathcal{F}$.

Small-scale Learning

"The active budget constraint is the number of examples."

- To reduce the estimation error, take n as large as the budget allows.
- To reduce the optimization error to zero, take $\rho = 0$.
- We need to adjust the size of $\mathcal{F}$.



See Structural Risk Minimization (Vapnik 74) and later works.

Large-scale Learning

"The active budget constraint is the computing time."

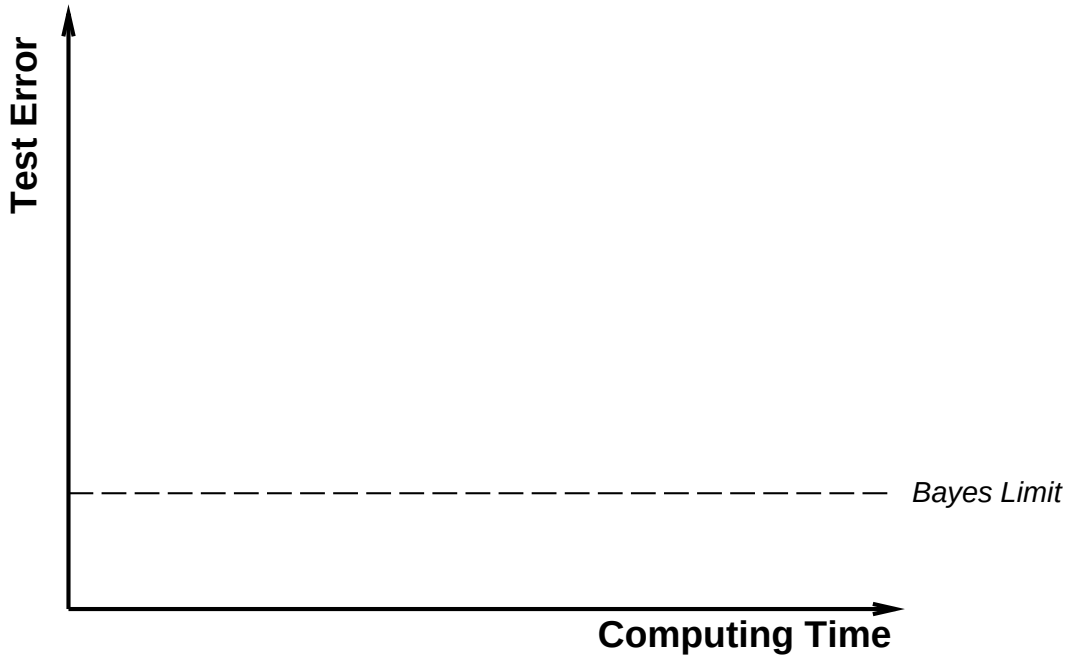
- More complicated tradeoffs.
  The computing time depends on the three variables: $\mathcal{F}$, n, and ρ.

- Example.
  If we choose ρ small, we decrease the optimization error. But we must also decrease $\mathcal{F}$ and/or n with adverse effects on the estimation and approximation errors.

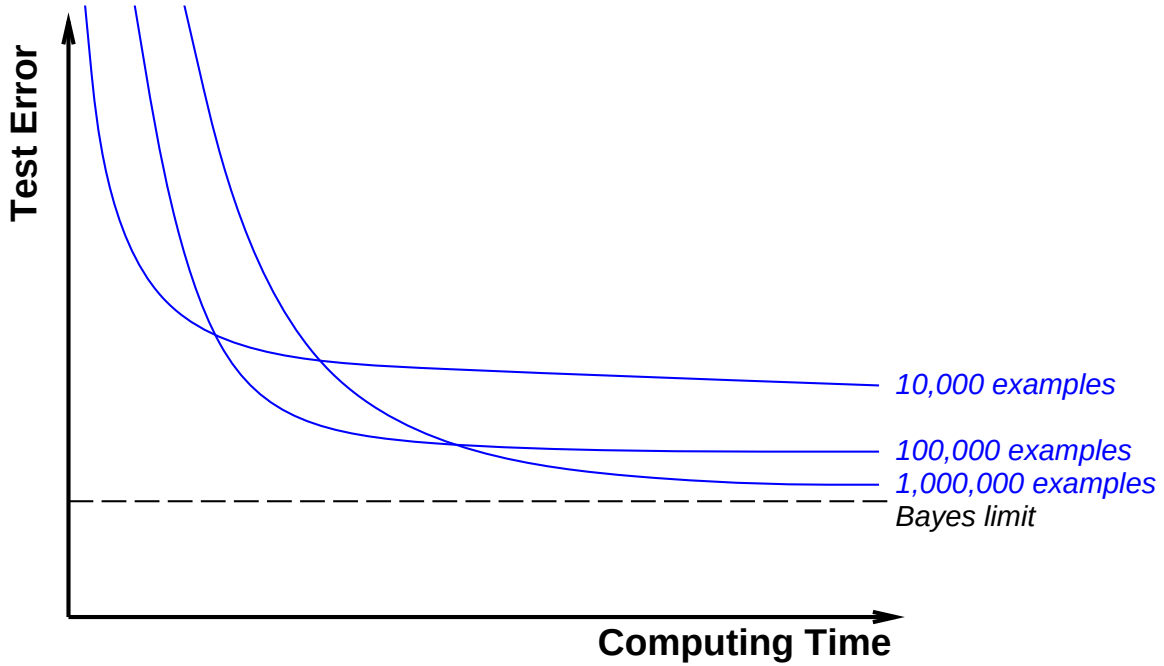
- The exact tradeoff depends on the optimization algorithm.

- We can compare optimization algorithms rigorously.

Test Error versus Learning Time

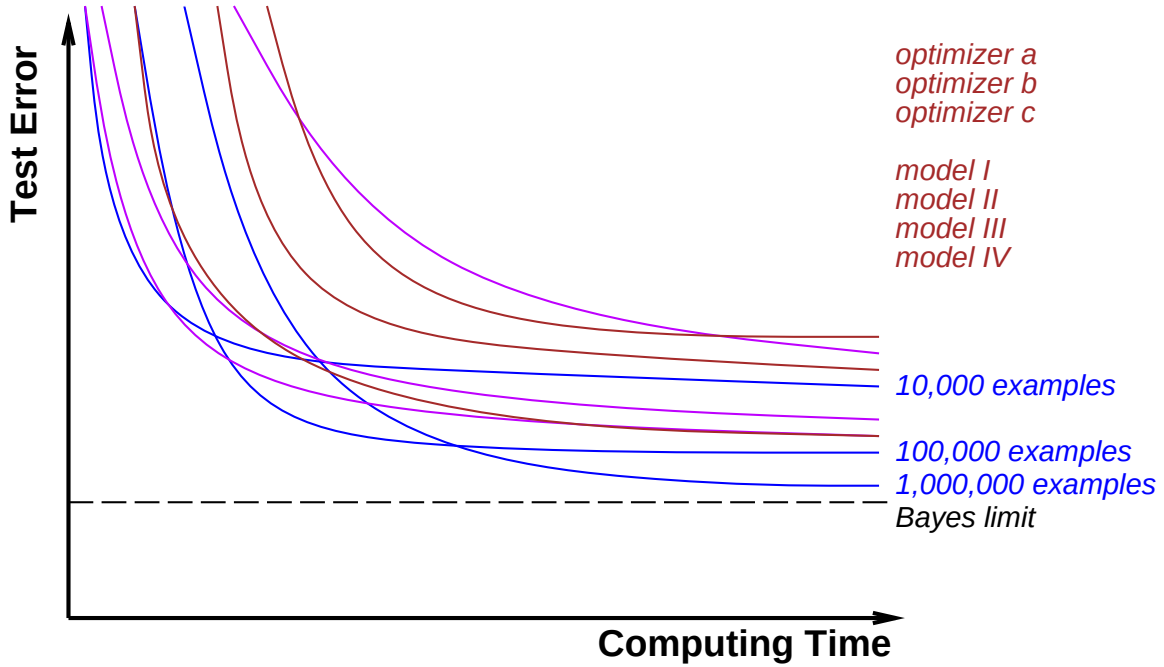


Test Error versus Learning Time



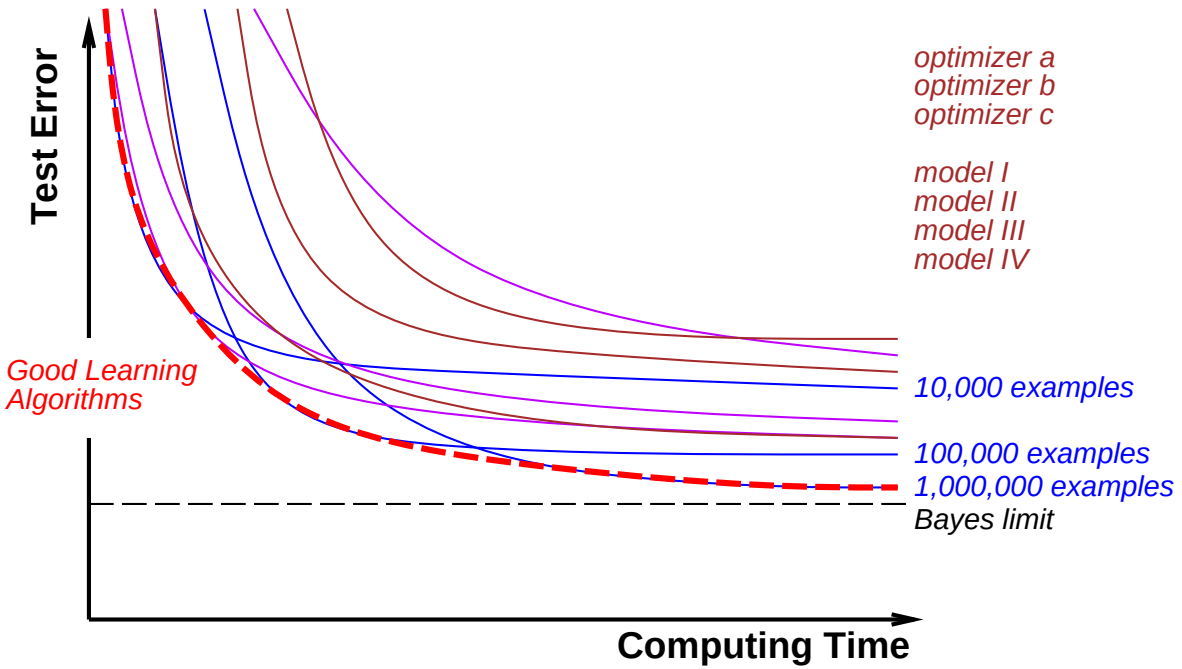
Vary the number of examples. . .

Test Error versus Learning Time



Vary the number of examples, the statistical models, the algorithms,. . .

Test Error versus Learning Time



Not all combinations are equal.

Let's compare the red curve for different optimization algorithms.

III. Asymptotic Analysis

Asymptotic Analysis

$$E(\tilde{f}_n) - E(f^*) = \mathcal{E} = \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}}$$

Asymptotic Analysis

All three errors must decrease with comparable rates.

Forcing one of the errors to decrease much faster
- would require additional computing efforts,
- but would not significantly improve the test error.

Statistics

Asymptotics of the statistical components of the error

– Thanks to refined uniform convergence arguments

$$\mathcal{E} \;=\; \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}} \;\sim\; \mathcal{E}_{\text{app}} \;+\; \left(\frac{\log n}{n}\right)^{\alpha} \;+\; \rho$$

with exponent $\frac{1}{2} \leq \alpha \leq 1$.

Asymptotically effective large scale learning

– Must choose $\mathcal{F}$, n, and ρ such that

$$\mathcal{E} \;\sim\; \mathcal{E}_{\text{app}} \;\sim\; \mathcal{E}_{\text{est}} \;\sim\; \mathcal{E}_{\text{opt}} \;\sim\; \left(\frac{\log n}{n}\right)^{\alpha} \;\sim\; \rho \;.$$

What about optimization times?

Statistics and Computation

	GD	2GD	SGD	2SGD
Time per iteration :	n	n	1	1
Iters to accuracy ρ :	$\log \frac{1}{\rho}$	$\log\log \frac{1}{\rho}$	$\frac{1}{\rho}$	$\frac{1}{\rho}$
Time to accuracy ρ :	$n \log \frac{1}{\rho}$	$n \log\log \frac{1}{\rho}$	$\frac{1}{\rho}$	$\frac{1}{\rho}$
Time to error ε :	$\frac{1}{\varepsilon^{1/\alpha}} \log^2 \frac{1}{\varepsilon}$	$\frac{1}{\varepsilon^{1/\alpha}} \log \frac{1}{\varepsilon} \log\log \frac{1}{\varepsilon}$	$\frac{1}{\varepsilon}$	$\frac{1}{\varepsilon}$

– 2GD optimizes much faster than GD.
– SGD optimization speed is catastrophic.
– SGD learns faster than both GD and 2GD.
– 2SGD only changes the constants.

Experiment: Text Categorization

Dataset

– Reuters RCV1 document corpus.

– 781,265 training examples, 23,149 testing examples.

Task

– Recognizing documents of category `CCAT`.

– 47,152 TF-IDF features.

– Linear SVM.

Same setup as (Joachims, 2006) and (Shalev-Schwartz et al., 2007) using plain SGD.

Experiment: Text Categorization

- **Results: Hinge-loss SVM**

  $Q(x, y, w) = \max\{0, 1 - yw^\top \Phi(x)\} \quad \lambda = 0.0001$

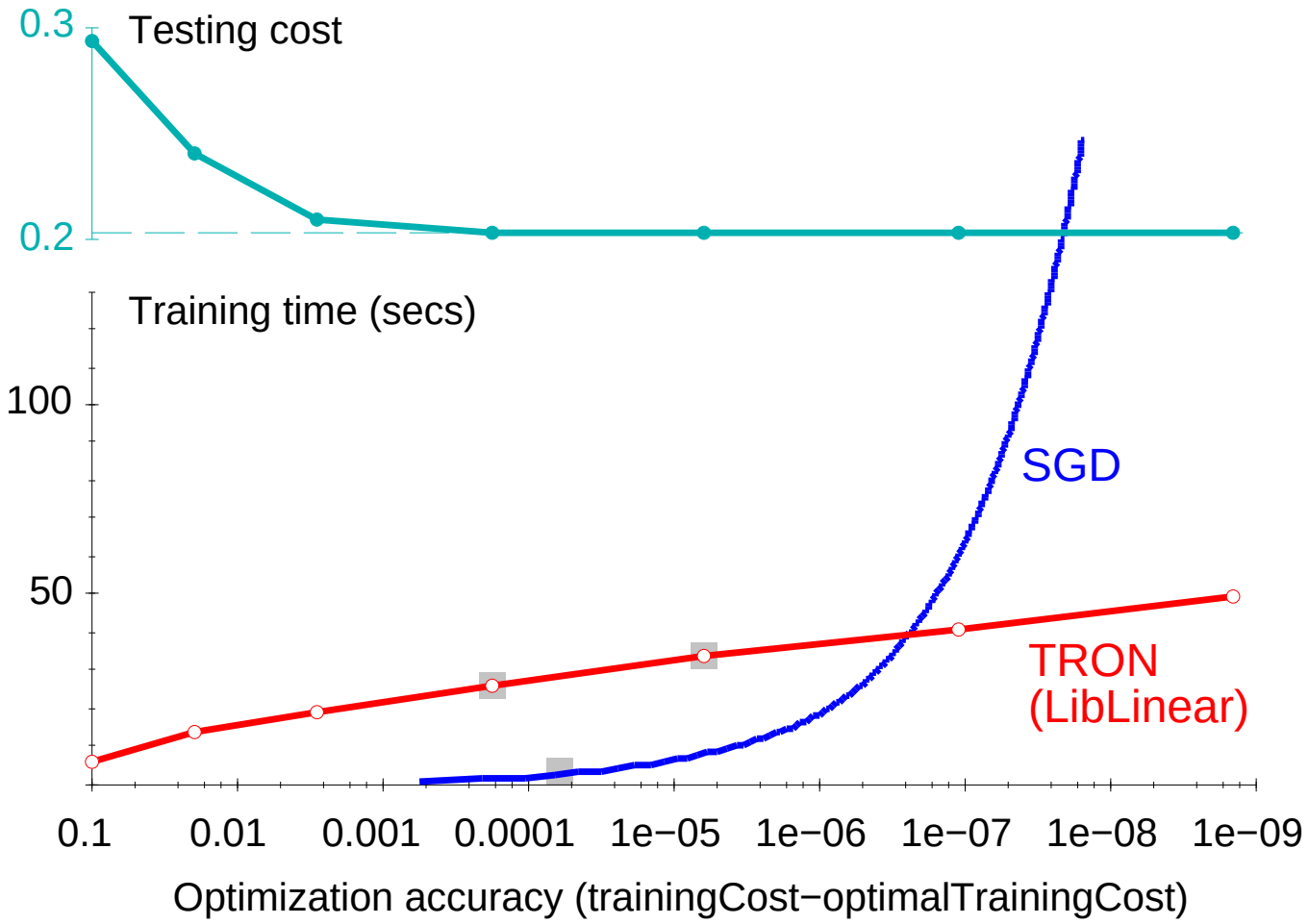
	Training Time	Primal cost	Test Error
SVMLight	23,642 secs	*0.2275*	*6.02%*
SVMPerf	66 secs	0.2278	6.03%
SGD	1.4 secs	*0.2275*	*6.02%*

- **Results: Log-Loss SVM**

  $Q(x, y, w) = \log(1 + \exp(-yw^\top \Phi(x))) \quad \lambda = 0.00001$

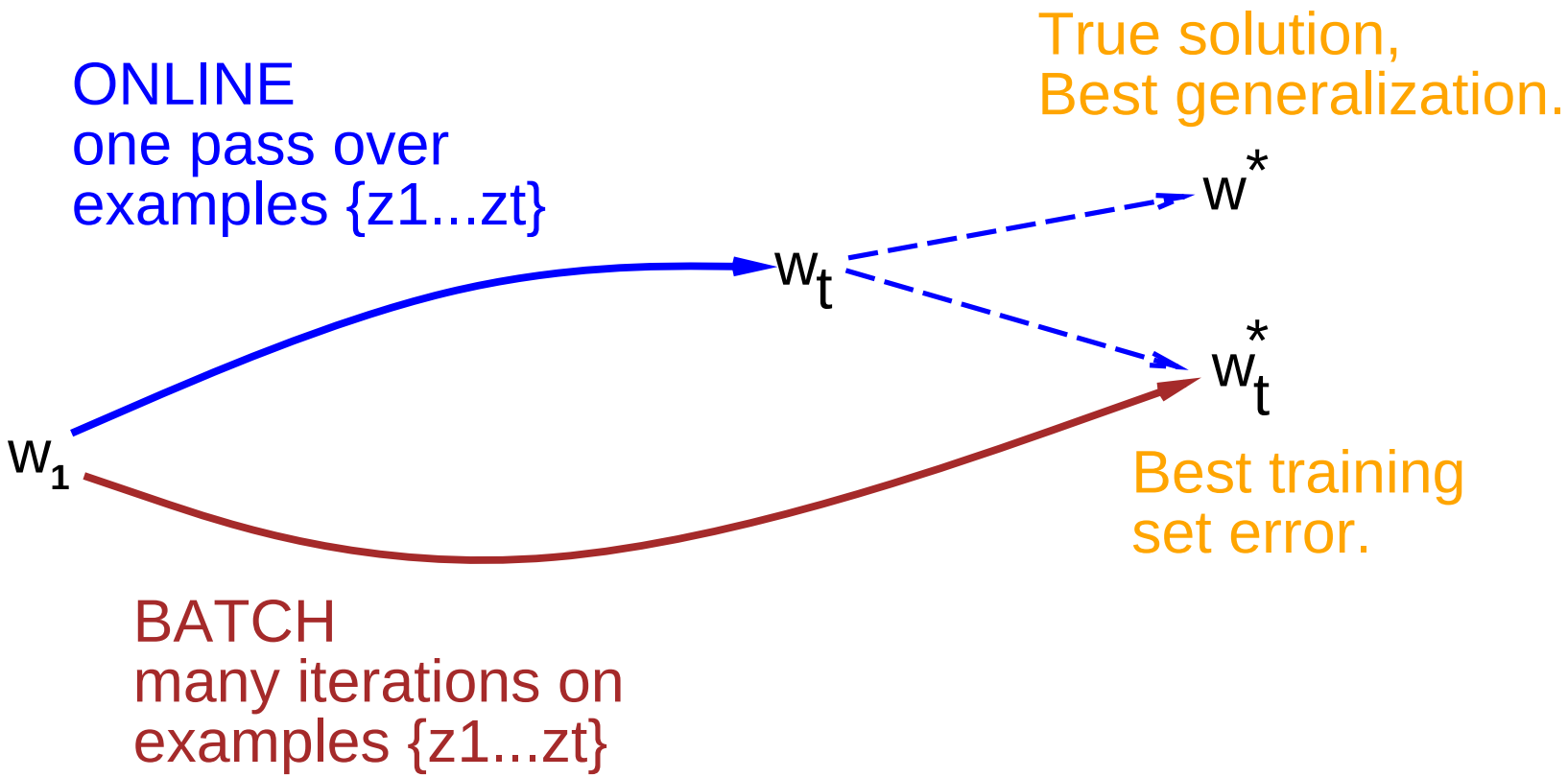
	Training Time	Primal cost	Test Error
TRON(LibLinear, $\varepsilon = 0.01$)	30 secs	0.18907	*5.68%*
TRON(LibLinear, $\varepsilon = 0.001$)	44 secs	*0.18890*	5.70%
SGD	2.3 secs	0.18893	*5.66%*

The Wall


0.3
Testing cost
0.2
Training time (secs)
100
50
SGD
TRON
(LibLinear)
0.1
0.01
0.001
0.0001
1e−05
1e−06
1e−07
1e−08
1e−09
Optimization accuracy (trainingCost−optimalTrainingCost)

IV. Learning with a Single Pass

Batch and online paths

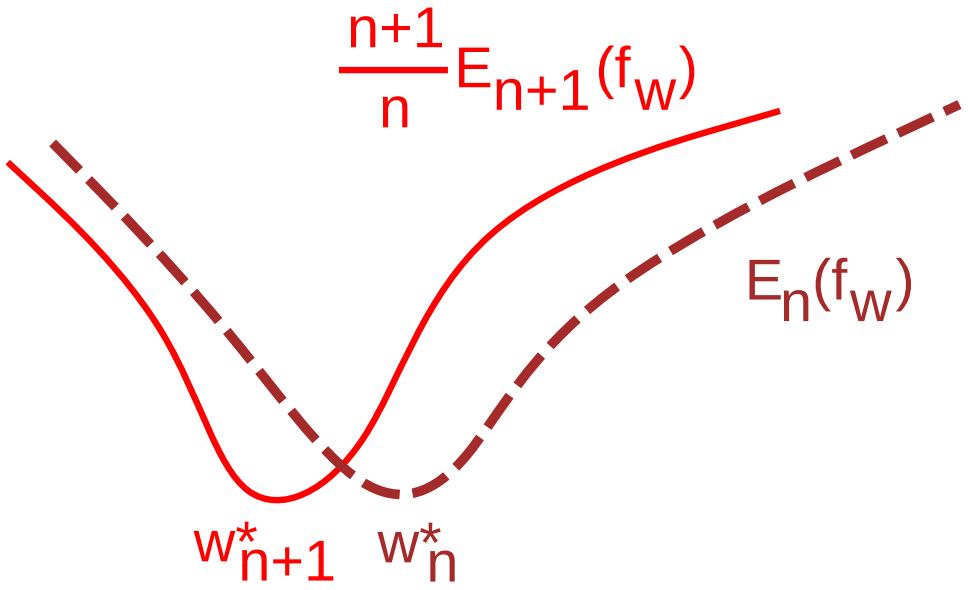


Effect of one Additional Example (i)

Compare

$$w_n^* = \arg\min_w E_n(f_w)$$

$$w_{n+1}^* = \arg\min_w E_{n+1}(f_w) = \arg\min_w \left[E_n(f_w) + \frac{1}{n} \ell\big(f_w(x_{n+1}), y_{n+1}\big) \right]$$



Effect of one Additional Example (ii)

- **First Order Calculation**

$$w_{n+1}^* = w_n^* - \frac{1}{n} H_{n+1}^{-1} \frac{\partial \ell\big(f_{w_n}(x_n), y_n\big)}{\partial w} + \mathcal{O}\left(\frac{1}{n^2}\right)$$

where H_{n+1} is the empirical Hessian on $n+1$ examples.

- **Compare with Second Order Stochastic Gradient Descent**

$$w_{t+1} = w_t - \frac{1}{t} H^{-1} \frac{\partial \ell\big(f_{w_t}(x_n), y_n\big)}{\partial w}$$

- **Could they converge with the same speed?**

- **C_2 assumptions $\Longrightarrow$ Accurate speed estimates.**

Speed of Scaled Stochastic Gradient

- Study $w_{t+1} = w_t - \frac{1}{t} B_t \frac{\partial \ell\left(f_{w_t}(x_n), y_n\right)}{\partial w} + \mathcal{O}\left(\frac{1}{t^2}\right)$ with $B_t \to B \succ 0$, $BH \succ I/2$.
- Establish convergence a.s. via quasi-martingales (see Bottou, 1991, 1998).
- Let $U_t = H\,(w_t - w^*)\,(w_t - w^*)'$. Observe $E(f_{w_t}) - E(f_{w^*}) = \mathrm{tr}(U_t) + o(\mathrm{tr}(U_t))$
- Derive $\mathbb{E}_t(U_{t+1}) = \left[I - \frac{2BH}{t} + o\left(\frac{1}{t}\right)\right] U_t + \frac{HBGB}{t^2} + o\left(\frac{1}{t^2}\right)$ where G is the Fisher matrix.
- Lemma: study real sequence $u_{t+1} = \left(1 + \frac{\alpha}{t} + o\left(\frac{1}{t}\right)\right) u_t + \frac{\beta}{t^2} + o\left(\frac{1}{t^2}\right)$.
  - When $\alpha > 1$ show $u_t = \frac{\beta}{\alpha - 1} \frac{1}{t} + o\left(\frac{1}{t}\right)$ (nasty proof!).
  - When $\alpha < 1$ show $u_t \sim t^{-\alpha}$ (up to log factors).
- Bracket $\mathbb{E}(\mathrm{tr}(U_{t+1}))$ between two such sequences and conclude:

$$\frac{\mathrm{tr}(HBGB)}{2\lambda_{BH}^{\max} - 1} \frac{1}{t} + o\left(\frac{1}{t}\right) \le \mathbb{E}\left[E(f_{w_t}) - E(f_{w^*})\right] \le \frac{\mathrm{tr}(HBGB)}{2\lambda_{BH}^{\min} - 1} \frac{1}{t} + o\left(\frac{1}{t}\right)$$

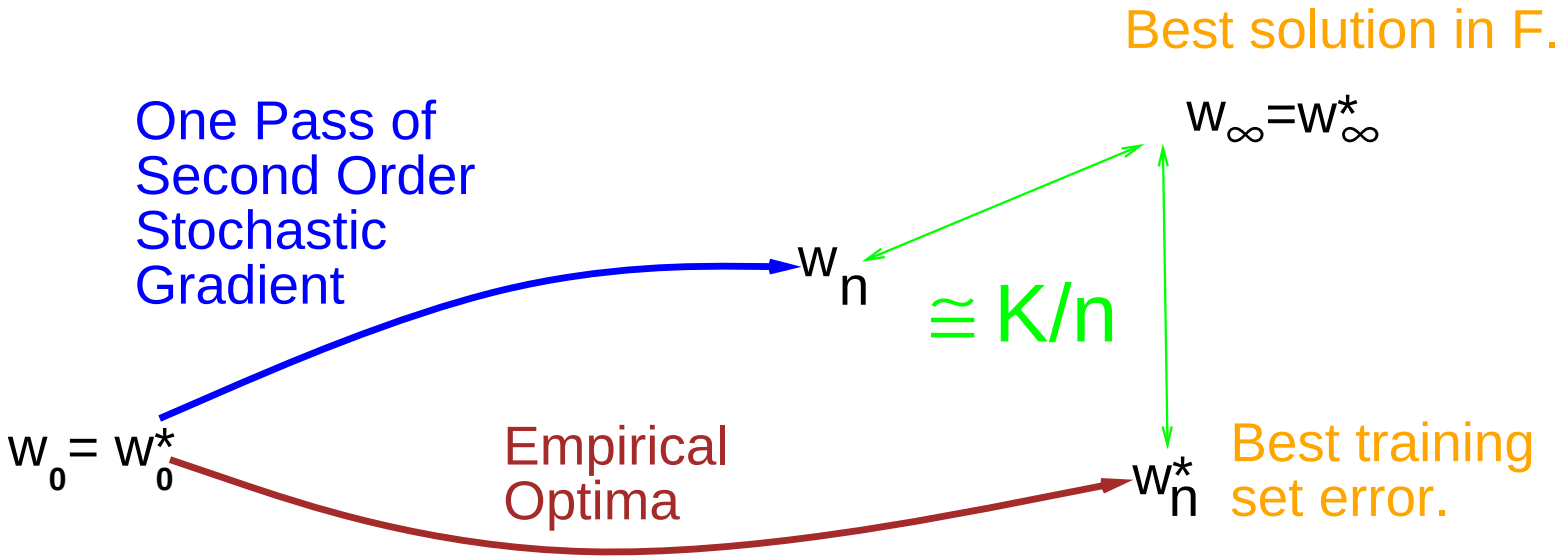
- Interesting special cases: $B = I/\lambda_H^{\min}$ and $B = H^{-1}$.

Asymptotic Efficiency of Second Order SGD.

"Empirical optima" **"Second-order SGD"**

$$n\,\mathbb{E}\big[E(f_{w_n^*}) - E(f_{\mathcal{F}})\big] = \lim_{t\to\infty} t\,\mathbb{E}\big[E(f_{w_t}) - E(f_{\mathcal{F}})\big]$$

$$\lim_{n\to\infty} n\,\mathbb{E}\big[\|w_\infty^* - w_n^*\|^2\big] = \lim_{t\to\infty} t\,\mathbb{E}\big[\|w_\infty - w_t\|^2\big]$$

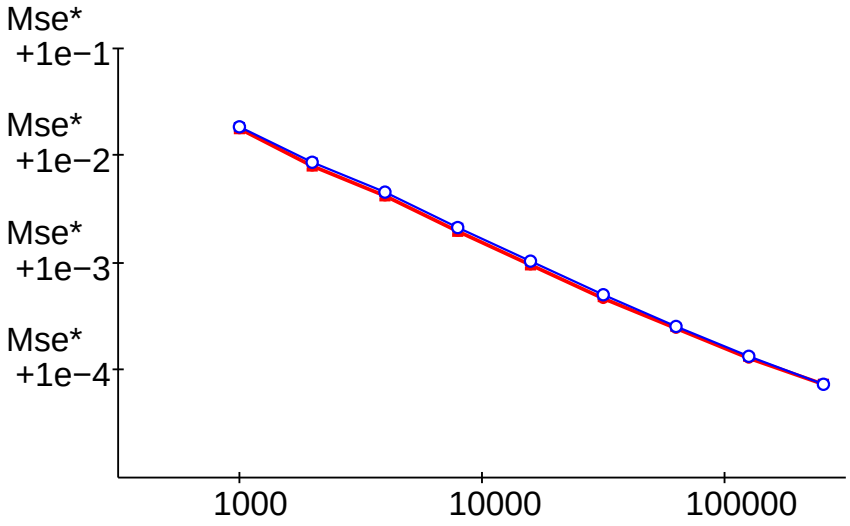


(Fabian, 1973; Murata & Amari, 1998; Bottou & LeCun, 2003).

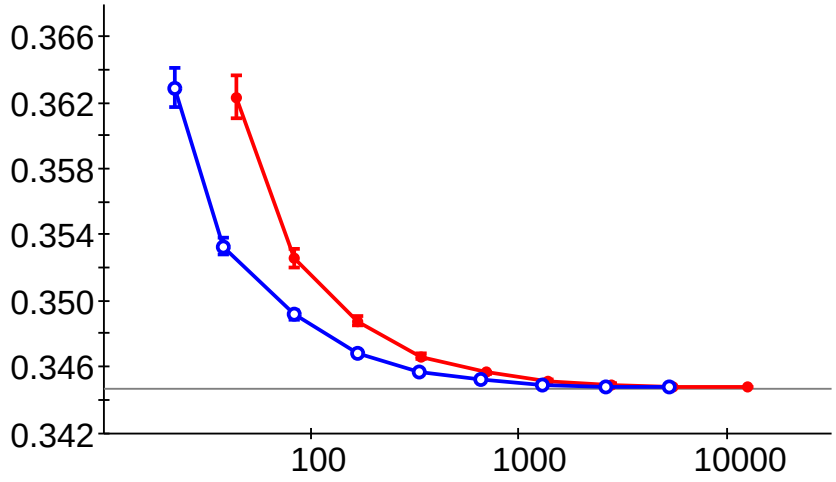
Optimal Learning in One Pass

A Single Pass of Second Order Stochastic Gradient generalizes as well as the Empirical Optimum.

Experiments on synthetic data



Number of examples



Milliseconds

Unfortunate Issues

Unfortunate theoretical issue

– How long to "reach" the asymptotic regime?

– One-pass learning speed regime may not be reached in one pass...

Unfortunate practical issue

– Second order SGD is rarely feasible.

  – estimate and store $d \times d$ matrix H^{-1}.

  – multiply the gradient for each example by this matrix H^{-1}.

Solutions

Limited storage approximations of H^{-1}

– Diagonal Gauss-Newton (Becker and Lecun, 1989)
– Low rank approximation [oLBFGS], (Schraudolph et al., 2007)
– Diagonal approximation [SGDQN], (Bordes et al., 2009)

Averaged stochastic gradient

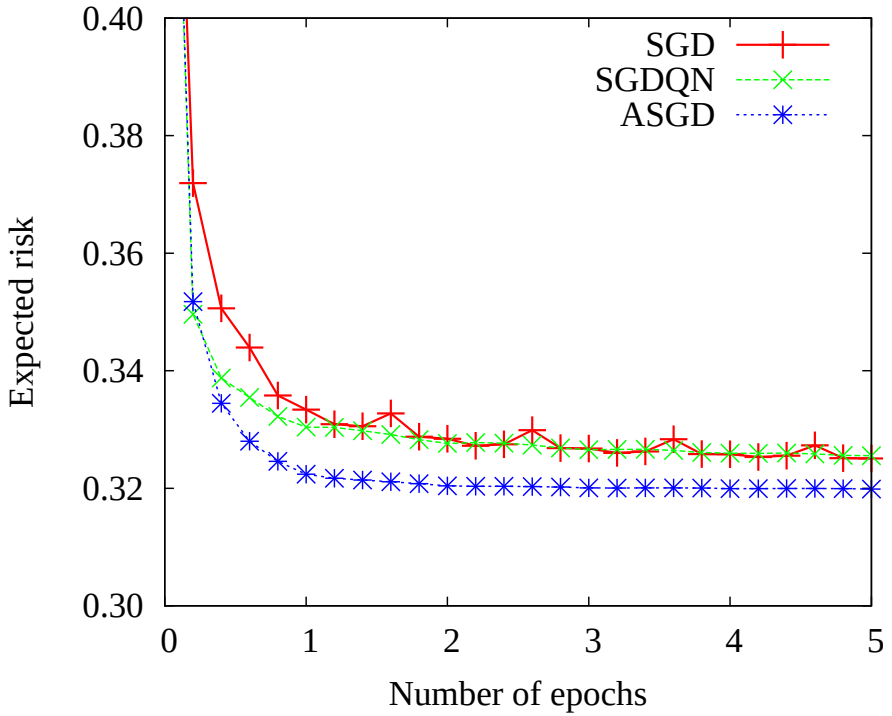
– Perform SGD with slowly decreasing gains, e.g. $\gamma_t \sim t^{-0.75}$.
– Compute averages $\bar{w}_{t+1} = \frac{t}{t+1}\bar{w}_t + \frac{1}{t}w_{t+1}$
– Same asymptotic speed as 2SGD (Polyak and Juditsky, 92)
– Can take a while to "reach" the asymptotic regime.

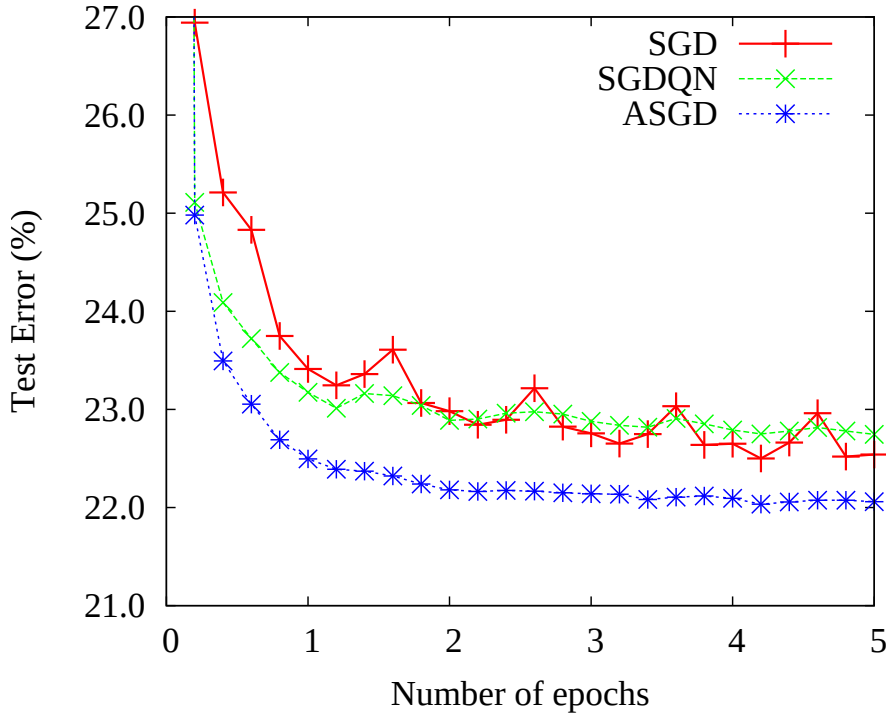
Experiment: ALPHA dataset

– From the 2008 Pascal Large Scale Learning Challenge.

– Loss: $Q(x, y, w) = \Big(\max\{0, 1 - y\, w^\top x\}\Big)^2$.

– SGD, SGDQN: $\gamma_t = \gamma_0(1 + \gamma_0 \lambda t)^{-1}$. ASGD: $\gamma_t = \gamma_0(1 + \gamma_0 \lambda t)^{-0.75}$

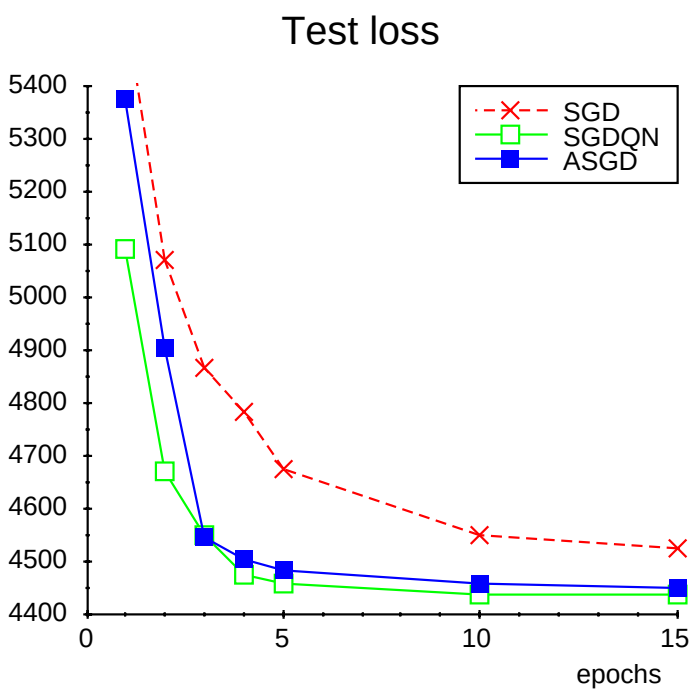


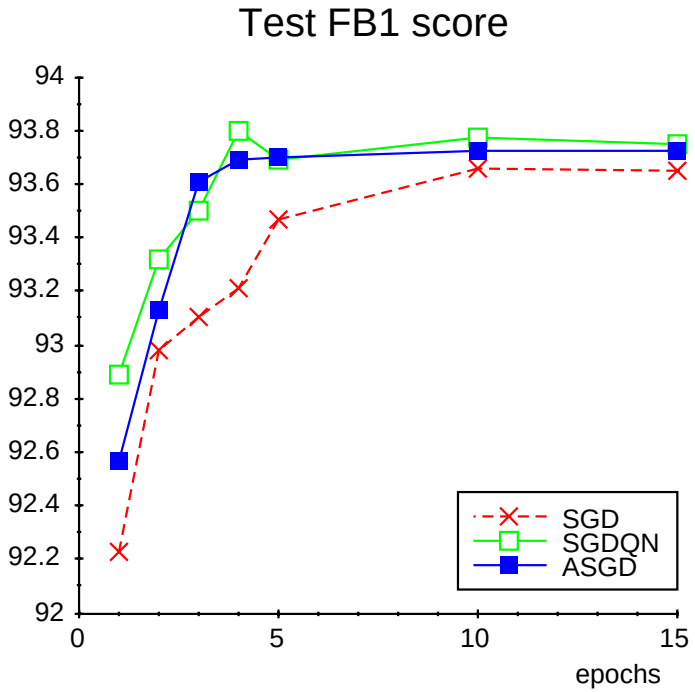


ASGD nearly reaches the optimal expected risk after a single pass.

Experiment: Conditional Random Field

– CRF for the CONLL 2000 Chunking task.
– 1.7M parameters. 107,000 training segments.





SGDQN more attractive than ASGD.
Training times: 500s (SGD), 150s (ASGD), 75s (SGDQN).
Standard LBFGS optimizer needs 72 minutes.

V. Conclusions

Conclusions

– Good optimization algorithm $\neq$ good learning algorithm.

– SGD is a poor optimization algorithm.

– SGD is a good learning algorithm for large scale problems.

– SGD variants can learn in a single pass (given enough data)