

Stochastic Gradient Descent Tricks - By Léon Bottou

B. Neelima Srilakshmi

Introduction

- explains why SGD is a good learning algorithm when the training set is large
- provides useful recommendations



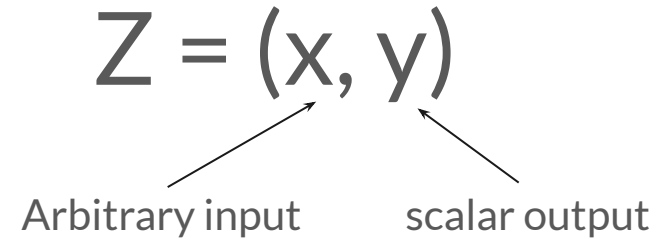


Basic supervised learning setup



Basic supervised learning setup

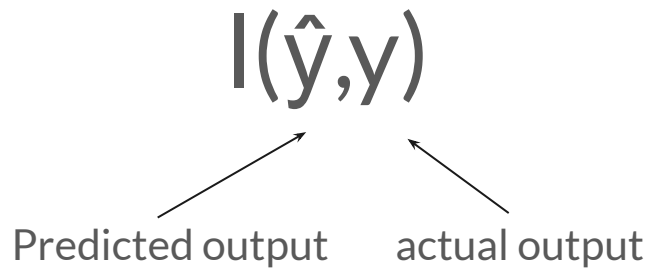
- $Z = (x, y)$





Basic supervised learning setup

- $z = (x, y)$
- Loss function, $l(\hat{y}, y)$





Basic supervised learning setup

- $z = (x, y)$
- Loss function, $l(\hat{y}, y)$
- Parameter, w



Basic supervised learning setup

- $z = (x, y)$
- Loss function, $l(\hat{y}, y)$
- Parameter, w
- Family of functions, F
- $f_w(x)$



Basic supervised learning setup

- $z = (x, y)$
- Loss function, $l(\hat{y}, y)$
- Parameter, w
- Family of functions, F
- $f_w(x)$
- Loss, $Q(z, w)$

$$Q(z, w) = l(f_w(x), y)$$



Basic supervised learning setup

- Laws of nature - average loss over unknown distribution

$dP(Z)$



Basic supervised learning setup

- Laws of nature - average loss over unknown distribution

$$dP(Z)$$

- average loss over known sample

$$z_1, \dots, z_n$$



Basic supervised learning setup

- Laws of nature - average loss over unknown distribution

$dP(Z)$

$$E(f) = \int \ell(f(x), y) dP(z)$$

Expected risk

- Average loss over known sample

z_1, \dots, z_n

$$E_n(f) = \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), y_i)$$

Empirical risk



Basic supervised learning setup

- Expected risk:
 - Generalization performance
 - Expected performance for future examples

$$E(f) = \int \ell(f(x), y) dP(z)$$

Expected risk

- Empirical risk:
 - Training performance

$$E_n(f) = \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), y_i)$$

Empirical risk



Gradient descent

- Empirical risk:
$$E_n(f) = \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), y_i)$$

- Gradient descent(GD):
$$w_{t+1} = w_t - \gamma \frac{1}{n} \sum_{i=1}^n \nabla_w Q(z_i, w_t)$$



Gradient descent

- Gradient descent(GD):
(γ = Scalar learning rate)

$$w_{t+1} = w_t - \gamma \frac{1}{n} \sum_{i=1}^n \nabla_w Q(z_i, w_t)$$

- Second order gradient descent(2GD):
(Γ = Positive definite matrix)

$$w_{t+1} = w_t - \Gamma_t \frac{1}{n} \sum_{i=1}^n \nabla_w Q(z_i, w_t)$$



Gradient descent

- Gradient descent(GD): Linear convergence
- Second order gradient descent(2GD): Quadratic convergence



Stochastic Gradient Descent

- GD:
$$w_{t+1} = w_t - \gamma \frac{1}{n} \sum_{i=1}^n \nabla_w Q(z_i, w_t)$$
- SGD:
$$w_{t+1} = w_t - \gamma_t \nabla_w Q(z_t, w_t)$$



SGD

- Stochastic gradient descent(SGD):

$$w_{t+1} = w_t - \gamma_t \nabla_w Q(z_t, w_t)$$

- Second order stochastic gradient descent(2SGD):

$$w_{t+1} = w_t - \gamma_t \Gamma_t \nabla_w Q(z_t, w_t) .$$

Stochastic gradient algorithms for various learning systems

Loss	Stochastic gradient algorithm
Adaline [26] $Q_{\text{adaline}} = \frac{1}{2}(y - w^\top \Phi(x))^2$ Features $\Phi(x) \in \mathbb{R}^d$, Classes $y = \pm 1$	$w \leftarrow w + \gamma_t (y_t - w^\top \Phi(x_t)) \Phi(x_t)$
Perceptron [17] $Q_{\text{perceptron}} = \max\{0, -y w^\top \Phi(x)\}$ Features $\Phi(x) \in \mathbb{R}^d$, Classes $y = \pm 1$	$w \leftarrow w + \gamma_t \begin{cases} y_t \Phi(x_t) & \text{if } y_t w^\top \Phi(x_t) \leq 0 \\ 0 & \text{otherwise} \end{cases}$
K-Means [12] $Q_{\text{kmeans}} = \min_k \frac{1}{2}(z - w_k)^2$ Data $z \in \mathbb{R}^d$ Centroids $w_1 \dots w_k \in \mathbb{R}^d$ Counts $n_1 \dots n_k \in \mathbb{N}$, initially 0	$k^* = \arg \min_k (z_t - w_k)^2$ $n_{k^*} \leftarrow n_{k^*} + 1$ $w_{k^*} \leftarrow w_{k^*} + \frac{1}{n_{k^*}}(z_t - w_{k^*})$ (counts provide optimal learning rates!)
SVM [5] $Q_{\text{svm}} = \lambda w^2 + \max\{0, 1 - y w^\top \Phi(x)\}$ Features $\Phi(x) \in \mathbb{R}^d$, Classes $y = \pm 1$ Hyperparameter $\lambda > 0$	$w \leftarrow w - \gamma_t \begin{cases} \lambda w & \text{if } y_t w^\top \Phi(x_t) > 1, \\ \lambda w - y_t \Phi(x_t) & \text{otherwise.} \end{cases}$
Lasso [23] $Q_{\text{lasso}} = \lambda w _1 + \frac{1}{2}(y - w^\top \Phi(x))^2$ $w = (u_1 - v_1, \dots, u_d - v_d)$ Features $\Phi(x) \in \mathbb{R}^d$, Classes $y = \pm 1$ Hyperparameter $\lambda > 0$	$u_i \leftarrow [u_i - \gamma_t (\lambda - (y_t - w^\top \Phi(x_t)) \Phi_i(x_t))]_+$ $v_i \leftarrow [v_i - \gamma_t (\lambda + (y_t - w^\top \Phi(x_t)) \Phi_i(x_t))]_+$ with notation $[x]_+ = \max\{0, x\}$.



When to use Stochastic Gradient Descent

Limitations to statistical machine :

- Sample size
- Computing time



When to use Stochastic Gradient Descent

Limitations to statistical machine :

- Sample size
- Computing time

data sizes

speed of processors

**Use stochastic gradient descent
when training time is the bottleneck.**



The trade-offs of large scale learning

$f^* = \arg \min_f E(f)$ -> best possible prediction function

$f_F^* = \arg \min_{f \in F} E(f)$ -> best function in family F

$f_n = \arg \min_{f \in F} E_n(f)$ -> empirical optimum

f_n^{\sim} -> minimizes the objective

function with a predefined accuracy

$$\mathcal{E} = \underbrace{\mathbb{E}[E(f_{\mathcal{F}}^*) - E(f^*)]}_{\mathcal{E}_{\text{app}}} + \underbrace{\mathbb{E}[E(f_n) - E(f_{\mathcal{F}}^*)]}_{\mathcal{E}_{\text{est}}} + \underbrace{\mathbb{E}[E(\tilde{f}_n) - E(f_n)]}_{\mathcal{E}_{\text{opt}}}$$

- The approximation error
- The estimation error
- The optimization error



Two types of learning problems

Small-scale learning problems


Large-scale learning problems



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




Recommendations

- 
- **Randomly shuffle the training examples.**
 - **Use preconditioning techniques.**
 - **Monitor both the training cost and the validation error.**



Monitor both the training cost and the validation error.


1. Zip once through the shuffled training set and perform the stochastic gradient descent updates (4).
2. With an additional loop over the training set, compute the training cost. Training cost here means the criterion that the algorithm seeks to optimize. You can take advantage of the loop to compute other metrics, but the training cost is the one to watch
3. With an additional loop over the validation set, to compute the validation set error. Error here means the performance measure of interest, such as the classification error. You can also take advantage of this loop to cheaply compute other metrics.

- 
- Randomly shuffle the training examples.
 - Use preconditioning techniques.
 - Monitor both the training cost and the validation error.
 - Check the gradients using finite differences.



Check the gradients using finite differences.

1. Pick an example z .
2. Compute the loss $Q(z, w)$ for the current w .
3. Compute the gradient $g = \nabla_w Q(z, w)$.
4. Apply a slight perturbation $w' = w + \delta$. For instance, change a single weight by a small increment, or use $\delta = -\gamma g$ with γ small enough.
5. Compute the new loss $Q(z, w')$ and verify that $Q(z, w') \approx Q(z, w) + \delta g$.

- 
- Randomly shuffle the training examples.
 - Use preconditioning techniques.
 - Monitor both the training cost and the validation error.
 - Check the gradients using finite differences.
 - Experiment with the learning rates γ_t using a small sample of the training set.



Linear Models with L2 Regularization

The training objective:

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

SGD:

$$w_{t+1} = w_t - \gamma_t \nabla_w Q(z_t, w_t)$$

L2 Regularization:

$$w_{t+1} = (1 - \gamma_t \lambda) w_t - \gamma_t y_t x_t \ell'(y_t w_t x_t)$$

SVM

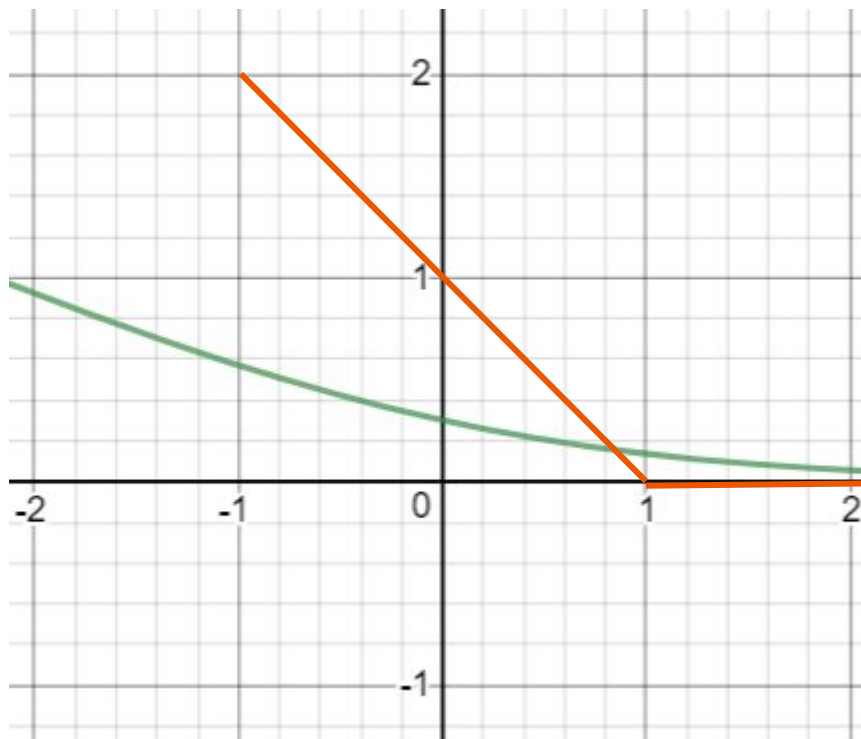
Loss $l(m)$

Hinge loss —

$$l(m) = \max\{0, 1 - m\}$$

Log-loss —

$$l(m) = \log(1 + e^{-m})$$





Sparsity

The stochastic gradient update:

$$w_{t+1} = (1 - \gamma_t \lambda) w_t - \gamma_t y_t x_t \ell'(y_t w_t x_t)$$

Leverage the sparsity of the training examples $\{x_t\}$.

- Represent w_t as a product $s_t W_t$ where $s_t \in \mathbb{R}$.

$$g_t = \ell'(y_t s_t W_t x_t),$$

$$s_{t+1} = (1 - \gamma_t \lambda) s_t,$$

$$W_{t+1} = W_t - \gamma_t y_t g_t x_t / s_{t+1}$$



Learning rates

$$\gamma_t = (\lambda t)^{-1}$$

$$\gamma_t = \gamma_0 (1 + \gamma_0 \lambda t)^{-1}$$

$\gamma_0 \rightarrow$ Which value to select



Averaged Stochastic Gradient Descent

- Performs normal SGD
- Computes average

$$\bar{w}_t = \frac{1}{t - t_0} \sum_{i=t_0+1}^t w_t$$



Recursive formula:

$$g_t = \ell'(y_t s_t W_t x_t),$$

$$s_{t+1} = (1 - \gamma_t \lambda) s_t$$

$$w_t = s_t W_t$$

$$W_{t+1} = W_t - \gamma_t y_t x_t g_t / s_{t+1}$$

$$\bar{w}_t = (A_t + \alpha_t W_t) / \beta_t$$

$$A_{t+1} = A_t + \gamma_t \alpha_t y_t x_t g_t / s_{t+1}$$

$$\beta_{t+1} = \beta_t / (1 - \mu_t)$$

$$\alpha_{t+1} = \alpha_t + \mu_t \beta_{t+1} s_{t+1}$$



ASGD

vs

2SGD

- Computational Cost

ASGD

vs

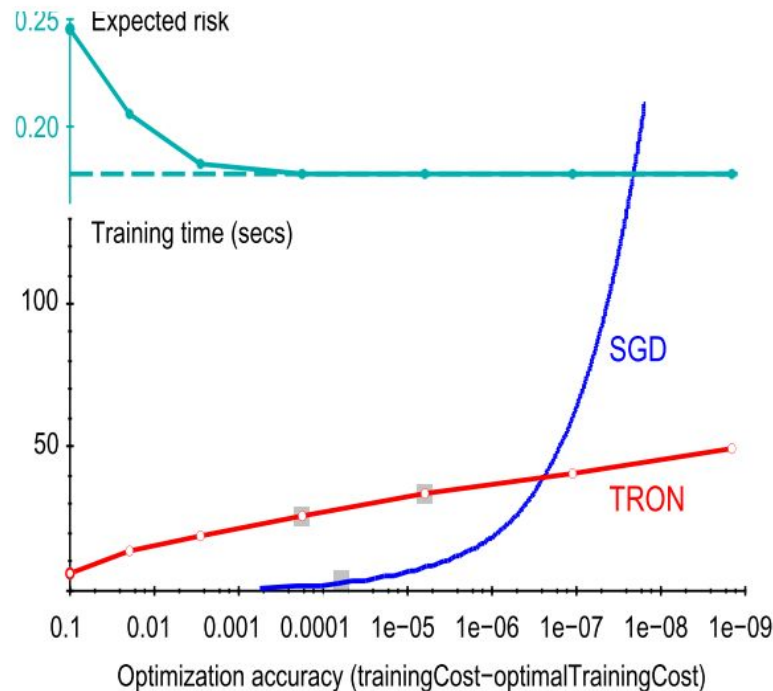
SGD

- Time taken to speed up

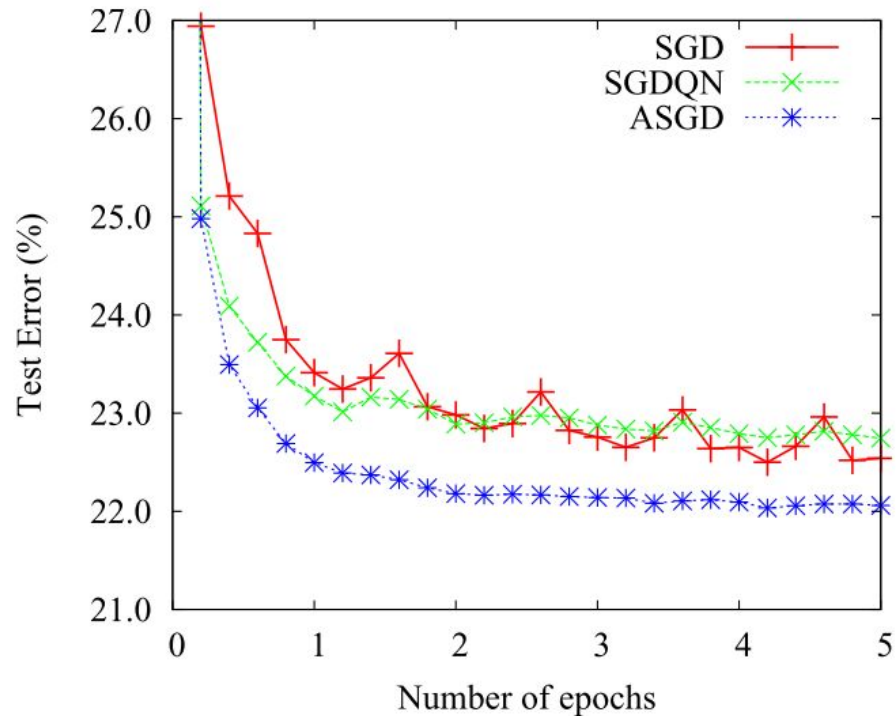
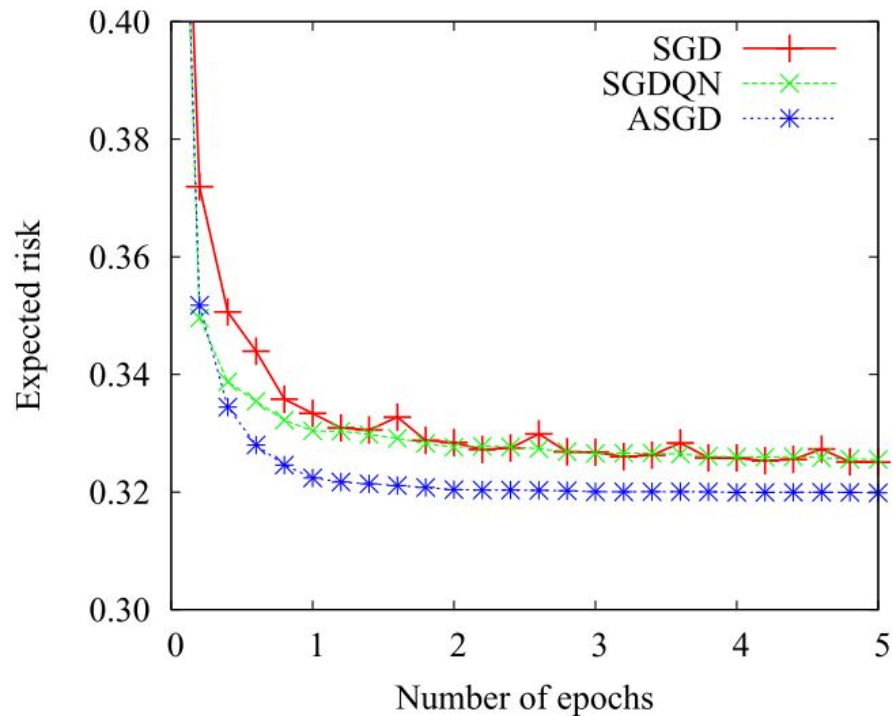
Experiments

A linear SVM trained for the recognition of the CCAT category in the RCV1 dataset [10] using both the hinge loss and the log loss

Algorithm	Time	Test Error
<i>Hinge loss SVM, $\lambda = 10^{-4}$.</i>		
SVMLIGHT	23,642 s.	6.02 %
SVMPELF	66 s.	6.03 %
SGD	1.4 s.	6.02 %
<i>Log loss SVM, $\lambda = 10^{-5}$.</i>		
TRON (-e0.01)	30 s.	5.68 %
TRON (-e0.001)	44 s.	5.70 %
SGD	2.3 s.	5.66 %

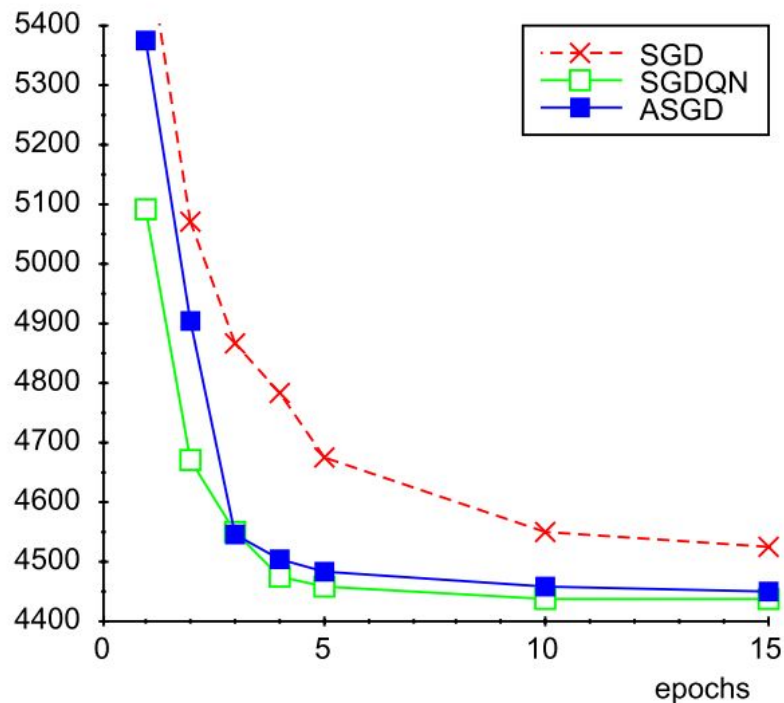


A linear model trained on the ALPHA task of the 2008 Pascal Large Scale Learning Challenge using the squared hinge loss

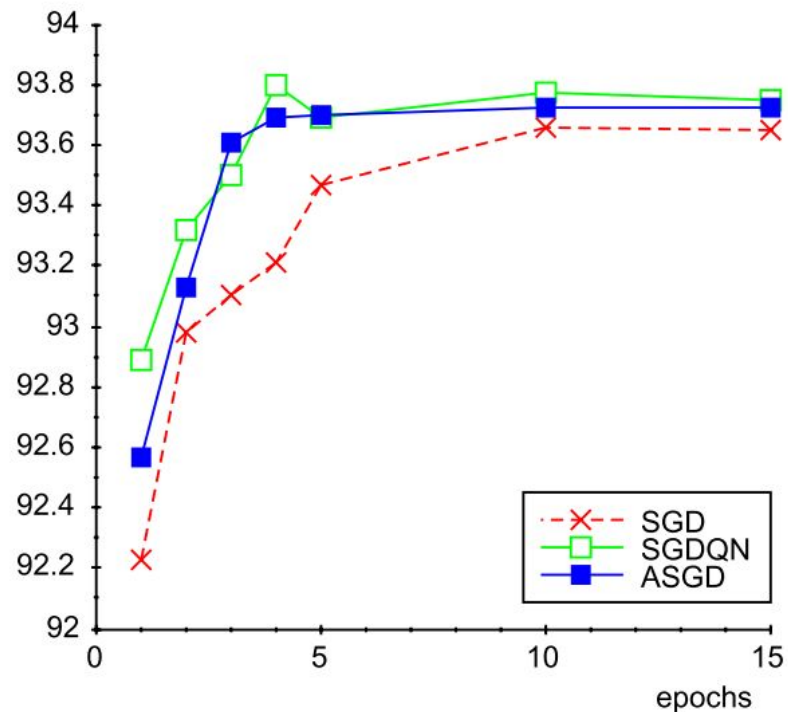


A CRF trained on the CONLL 2000 Chunking task

Test loss



Test FB1 score



Conclusion

Stochastic gradient descent and its variants are versatile techniques that have proven invaluable as a learning algorithms for large datasets.

Advice:

- perform small-scale experiments with subsets of the training data
- pay a ruthless attention to the correctness of the gradient computation





References

1. <https://metamug.com/article/groovy/stochastic-gradient-descent-tutorial-code-by-andrew-ng.html>
2. <https://leon.bottou.org/projects/sgd>