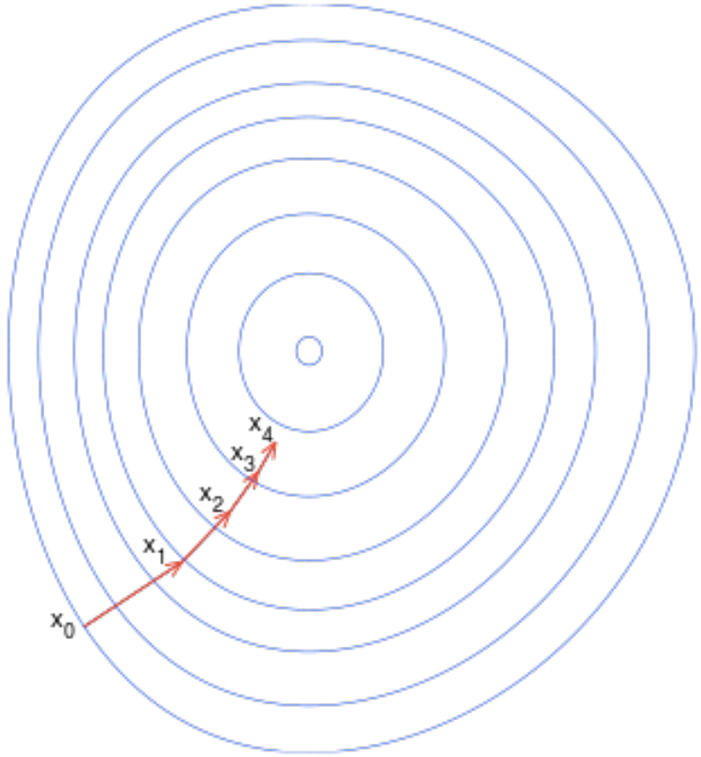
Stochastic Gradient Descent

CS 584: Big Data Analytics

Gradient Descent Recap

- Simplest and extremely popular
- Main Idea: take a step proportional to the negative of the gradient
- Easy to implement
- Each iteration is relatively cheap
- Can be slow to converge



Example: Linear Regression

- Optimization problem:

$$\min_w ||Xw - y||_2$$

- Closed form solution:

$$w^* = (X^\top X)^{-1} X^\top y$$

- Gradient update:

$$w^+ = w - \frac{1}{m} \sum_i (x_i^\top w - y_i) x_i$$

Requires an entire pass through the data!

Tackling Compute Problems: Scaling to Large n

- Streaming implementation
- Parallelize your batch algorithm
- Aggressively subsample the data
- Change algorithm or training method
  - Optimization is a surrogate for learning
  - Trade-off weaker optimization with more data

Tradeoffs of Large Scale Learning

- True (generalization) error is a function of approximation error, estimation error, and optimization error subject to number of training examples and computational time

$$\min_{\mathcal{F},\rho,n} \; \mathcal{E} = \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}} \quad \text{subject to} \begin{cases} n & \leq & n_{\max} \\ T(\mathcal{F},\rho,n) & \leq & T_{\max} \end{cases}$$

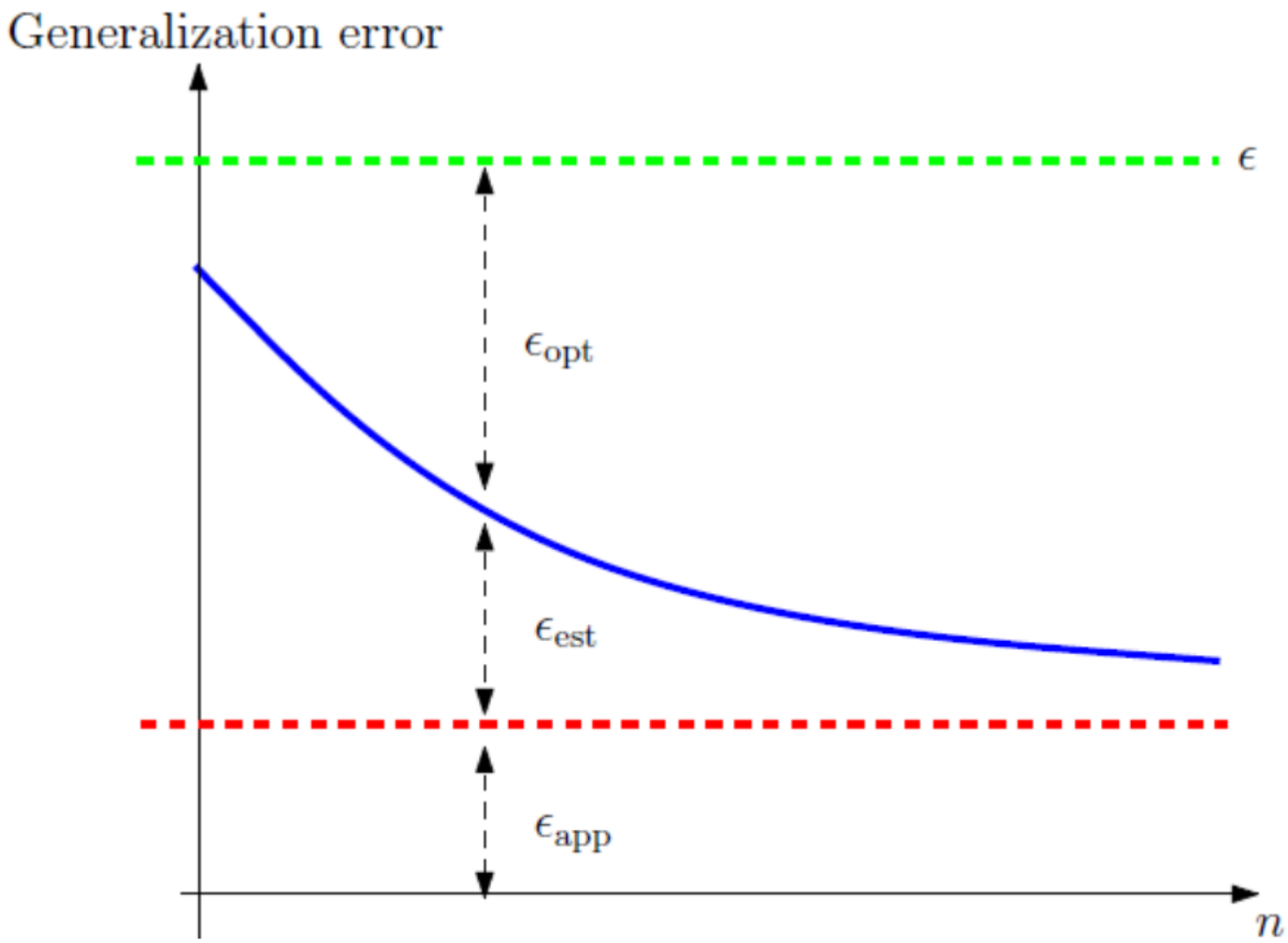
- Solution will depend on which budget constraint is active

Table 1: Typical variations when $\mathcal{F}$, n, and ρ increase.

		$\mathcal{F}$	n	ρ
$\mathcal{E}_{\text{app}}$	(approximation error)	↘		
$\mathcal{E}_{\text{est}}$	(estimation error)	↗	↘	
$\mathcal{E}_{\text{opt}}$	(optimization error)	⋯	⋯	↗
T	(computation time)	↗	↗	↘

Bottom and Bousquet (2011). The Tradeoffs of Large-Scale Learning. In Optimization for Machine Learning (pp. 351–368).

Minimizing Generalization Error



If $n \to \infty$,

then $\varepsilon_{\text{est}} \to 0$

For fixed generalization error, as number of samples increases, we can increase optimization tolerance

Talk by Aditya Menon, UCSD

Expected Risk vs Empirical Risk Minimization

Expected Risk

- Assume we know the ground truth distribution P(x,y)
- Expected risk associated with classification function

$$E(f_w) = \int L(f_w(x), y) dP(x, y)$$
$$= E[L(f_w(x), y)]$$

Empirical Risk

- Real world, ground truth distribution is not known
- Only empirical risk can be calculated for function

$$E_n(f_w) = \frac{1}{n} \sum_i L(f_w(x_i), y_i)$$

Gradient Descent Reformulated

$$w^+ = w - \gamma \frac{1}{n} \sum_i \nabla_w L(f_w(x_i), y_i) \qquad \nabla E_n(f_w)$$

learning rate or gain

- True gradient descent is a batch algorithm, slow but sure
- Under sufficient regularity assumptions, initial estimate is close to the optimal and gain is sufficiently small, there is linear convergence

Stochastic Optimization Motivation

- Information is redundant amongst samples
- Sufficient samples means we can afford more frequent, noisy updates
- Never-ending stream means we should not wait for all data
- Tracking non-stationary data means that the target is moving

Stochastic Optimization

- Idea: Estimate function and gradient from a small, current subsample of your data and with enough iterations and data, you will converge to the true minimum

- Pro: Better for large datasets and often faster convergence

- Con: Hard to reach high accuracy

- Con: Best classical methods can’t handle stochastic approximation

- Con: Theoretical definitions for convergence not as well-defined

Stochastic Gradient Descent (SGD)

- Randomized gradient estimate to minimize the function using a single randomly picked example

$$\text{Instead of } \nabla f, \text{use } \tilde{\nabla} f, \text{where } E[\tilde{\nabla} f] = \nabla f$$

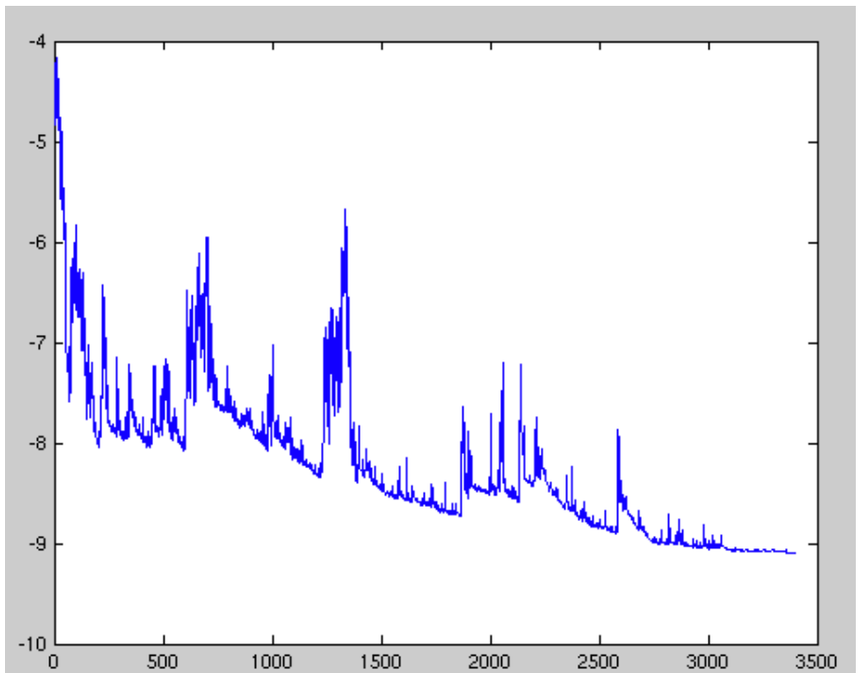
- The resulting update is of the form:

$$w^+ = w - \gamma \nabla_w L(f_w(x_i, y_i))$$

- Although random noise is introduced, it behaves like gradient descent in its expectation

SGD Algorithm

Randomly initialize parameter w and learning rate γ
while *Not Converged* **do**
  Randomly shuffle examples in training set
  for $i = 1, \cdots, N$ **do**
    $w^+ = w - \gamma \nabla_w L(f_w(x_i, y_i))$
  end
end



The Benefits of SGD

- Gradient is easy to calculate (“instantaneous”)
- Less prone to local minima
- Small memory footprint
- Get to a reasonable solution quickly
- Works for non-stationary environments as well as online settings
- Can be used for more complex models and error surfaces

Importance of Learning Rate

- Learning rate has a large impact on convergence
  - Too small —> too slow
  - Too large —> oscillatory and may even diverge
- Should learning rate be fixed or adaptive?
- Is convergence necessary?
  - Non-stationary: convergence may not be required or desired
  - Stationary: learning rate should decrease with time
    - Robbins-Monroe sequence is adequate $\gamma_t = \frac{1}{t}$

Mini-batch Stochastic Gradient Descent

- Rather than using a single point, use a random subset where the size is less than the original data size

$$w^+ = w - \gamma \frac{1}{|S_k|} \sum_{i \in S_k} \nabla_w L(f_w(x_i, y_i)), \text{where } S_k \subseteq [n]$$

- Like the single random sample, the full gradient is approximated via an unbiased noisy estimate

- Random subset reduces the variance by a factor of $1/|S_k|$, but is also $|S_k|$ times more expensive

Example: Regularized Logistic Regression

- Optimization problem:

$$\min \frac{1}{n} \sum_i \left(-y_i x_i^\top \beta + \log(1 + e^{x_i^\top \beta}) \right) + \frac{\lambda}{2} ||\beta||_2^2$$

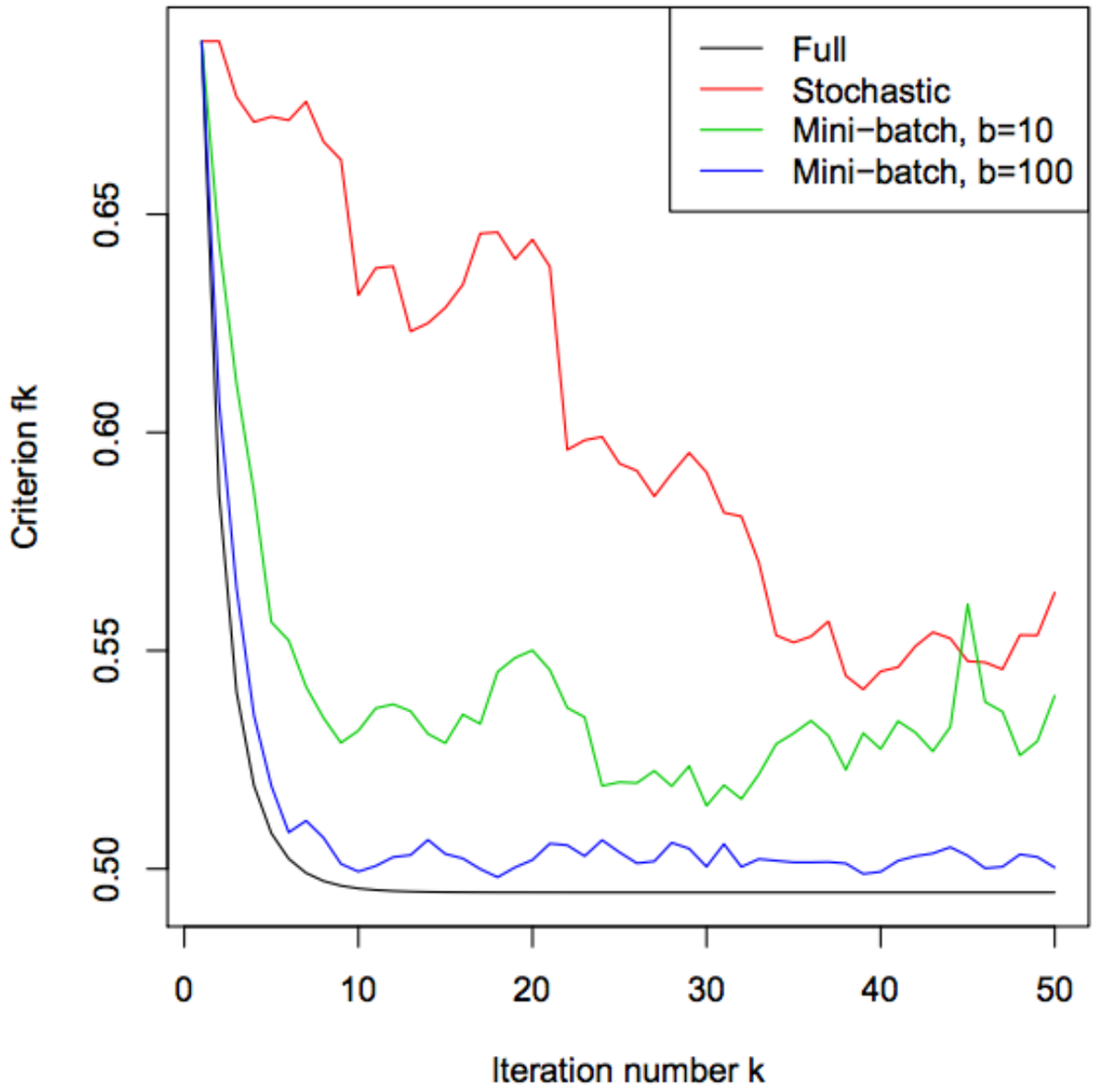
- Gradient computation:

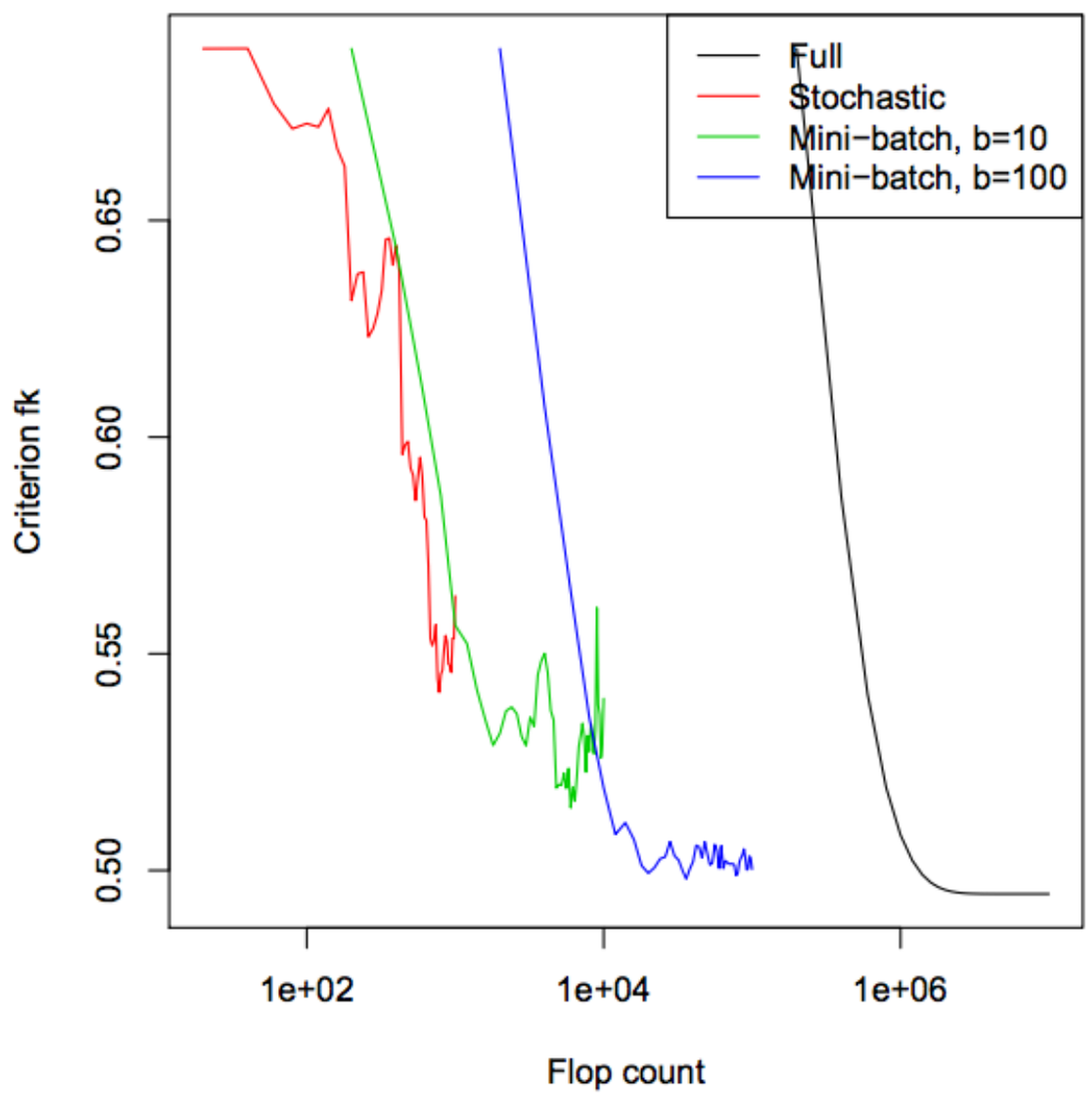
$$\nabla f(\beta) = \sum_i (y_i - p_i(\beta)) x_i + \lambda \beta$$

- Update costs:
  - Batch: O(nd)
  - Stochastic: O(d)
  - Mini-batch: O($|S_k|$d)

Batch is doable if n is moderate but not when n is huge

Example: n=10,000, d=20





Iterations make better progress as mini-batch size is larger but also takes more computation time

http://stat.cmu.edu/~ryantibs/convexopt/lectures/25-fast-stochastic.pdf

SGD Updates for Various Systems

Table 1. Stochastic gradient algorithms for various learning systems.

Loss	Stochastic gradient algorithm
Adaline [26] $Q_{\text{adaline}} = \frac{1}{2}\big(y - w^\top \Phi(x)\big)^2$ Features $\Phi(x) \in \mathbb{R}^d$, Classes $y = \pm 1$	$w \leftarrow w + \gamma_t \big(y_t - w^\top \Phi(x_t)\big)\, \Phi(x_t)$
Perceptron [17] $Q_{\text{perceptron}} = \max\{0, -y\, w^\top \Phi(x)\}$ Features $\Phi(x) \in \mathbb{R}^d$, Classes $y = \pm 1$	$w \leftarrow w + \gamma_t \begin{cases} y_t\, \Phi(x_t) & \text{if } y_t\, w^\top \Phi(x_t) \le 0 \\ 0 & \text{otherwise} \end{cases}$
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] $Q_{\text{lasso}} = \lambda \lvert w \rvert_1 + \frac{1}{2}\big(y - w^\top \Phi(x)\big)^2$ $w = (u_1 - v_1, \dots, u_d - v_d)$ Features $\Phi(x) \in \mathbb{R}^d$, Classes $y = \pm 1$ Hyperparameter $\lambda > 0$	$u_i \leftarrow \big[u_i - \gamma_t\big(\lambda - (y_t - w^\top \Phi(x_t))\Phi_i(x_t)\big)\big]_+$ $v_i \leftarrow \big[v_i - \gamma_t\big(\lambda + (y_t - w^\top \Phi(x_t))\Phi_i(x_t)\big)\big]_+$ with notation $[x]_+ = \max\{0, x\}$.

Bottou, L. (2012). Stochastic Gradient Descent Tricks. Neural Networks Tricks of the Trade.

Asymptotic Analysis of GD and SGD

Table 2. Asymptotic equivalents for various optimization algorithms: gradient descent (GD, eq. 2), second order gradient descent (2GD, eq. 3), stochastic gradient descent (SGD, eq. 4), and second order stochastic gradient descent (2SGD, eq. 5). Although they are the worst optimization algorithms, SGD and 2SGD achieve the fastest convergence speed on the expected risk. They differ only by constant factors not shown in this table, such as condition numbers and weight vector dimension.

	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 :	$\frac{1}{\varepsilon^{1/\alpha}} \log^2 \frac{1}{\varepsilon}$	$\frac{1}{\varepsilon^{1/\alpha}} \log \frac{1}{\varepsilon} \log\log \frac{1}{\varepsilon}$	$1/\varepsilon$	$1/\varepsilon$

Bottou, L. (2012). Stochastic Gradient Descent Tricks. Neural Networks Tricks of the Trade.

SGD Recommendations

- Randomly shuffle training examples
  - Although theory says you should randomly pick examples, it is easier to make a pass through your training set sequentially
  - Shuffling before each iteration eliminates the effect of order
- Monitor both training cost and validation error
  - Set aside samples for a decent validation set
  - Compute the objective on the training set and validation set (expensive but better than overfitting or wasting computation)

Bottou, L. (2012). Stochastic Gradient Descent Tricks. Neural Networks Tricks of the Trade.

SGD Recommendations (2)

- Check gradient using finite differences
  - If computation is slightly incorrect can yield erratic and slow algorithm
  - Verify your code by slightly perturbing the parameter and inspecting differences between the two gradients
- Experiment with the learning rates using small sample of training set
  - SGD convergence rates are independent from sample size
  - Use traditional optimization algorithms as a reference point

SGD Recommendations (3)

- Leverage sparsity of the training examples
  - For very high-dimensional vectors with few non zero coefficients, you only need to update the weight coefficients corresponding to nonzero pattern in x
- Use learning rates of the form $\gamma_t = \gamma_0(1 + \gamma_0 \lambda t)^{-1}$
  - Allows you to start from reasonable learning rates determined by testing on a small sample
  - Works well in most situations if the initial point is slightly smaller than best value observed in training sample

Some Resources for SGD

- Francis Bach's talk in 2012: http://www.ann.jussieu.fr/~plc/bach2012.pdf

- Stochastic Gradient Methods Workshop: http://yaroslavvb.blogspot.com/2014/03/stochastic-gradient-methods-2014.html

- Python implementation in scikit-learn: http://scikit-learn.org/stable/modules/sgd.html

- iPython notebook for implementing GD and SGD in Python: https://github.com/dtnewman/gradient_descent/blob/master/stochastic_gradient_descent.ipynb