Optimization for machine learning

Smooth optimization

R. Flamary

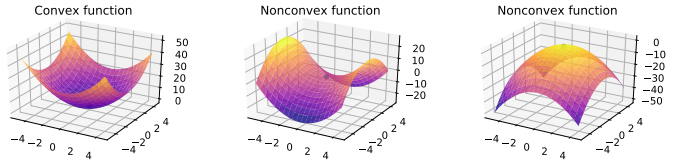
June 4, 2020

Full course overview

1. **Introduction to numerical optimization**
   - 1.1 Optimization problem formulation and principles
   - 1.2 Properties of optimization problems
   - 1.3 Machine learning as an optimization problem
2. **Constrained Optimization and Standard Optimization problems**
   - 2.1 Constraints, Lagrangian and KKT
   - 2.2 Linear Program (LP)
   - 2.3 Quadratic Program (QP)
   - 2.4 Other Classical problems (MIP,QCQP,SOCP,SDP)
3. **Smooth Optimization**
   - 3.1 Gradient descent
   - 3.2 Newton, quasi-Newton and Limited memory
   - 3.3 Stochastic Gradient Descent
4. **Non-smooth Optimization**
   - 4.1 Proximal operator and proximal methods
   - 4.2 Conditional gradient
5. **Conclusion**
   - 5.1 Other approaches (Coordinate descent, DC programming)
   - 5.2 Optimization problem decision tree
   - 5.3 References an toolboxes

Course overview

Smooth Optimization problem



Optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} \; F(\mathbf{x}), \tag{1}$$

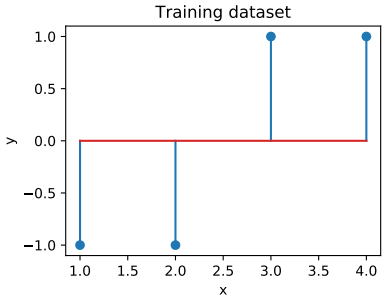
- F is smooth (at least differentiable).
- When F is convex $\mathbf{x}^\star$ is a solution of the problem if

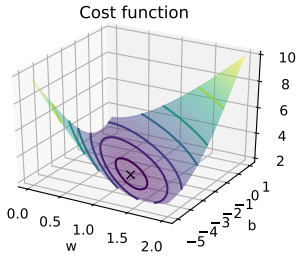
$$\nabla_{\mathbf{x}} F(\mathbf{x}^\star) = \mathbf{0}$$

- When F is non convex $\mathbf{x}^\star$ is a local minimizer of the problem if

$$\nabla_{\mathbf{x}} F(\mathbf{x}^\star) = \mathbf{0} \qquad \text{and} \quad \nabla^2_{\mathbf{x}} F(\mathbf{x}^\star) \succeq 0$$

Example optimization problem





1D Logistic regression

$$\min_{w,b} \quad \sum_{i=1}^{n} \log(1+\exp(-y_i(wx_i+b))) + \lambda\frac{w^2}{2}$$

- Linear model : $f(x) = wx + b$
- Training data (x_i, y_i) : $(1,-1),(2,-1),(3,1),(4,1)$.
- Problem solution for $\lambda = 1$: $\mathbf{x}^* = [w^\star, b^\star] = [0.96, -2.40]$
- Initialization : $\mathbf{x}^{(0)} = [1, -0.5]$.
- Complexity : Cost and gradient both $O(nd)$

Descent algorithm for smooth optimization

General iterative algorithm

1: Initialize $\mathbf{x}^{(0)}$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad \mathbf{d}^{(k)} \leftarrow$ Compute descent direction from $\mathbf{x}^{(k)}$
4: $\quad \rho^{(k)} \leftarrow$ Choose stepsize
5: $\quad \mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}$
6: **end for**

- $\mathbf{d}^{(k)} \in \mathbb{R}^n$ is a descent direction if $\boldsymbol{\nabla} F(\mathbf{x}^{(k)})^T \mathbf{d}^{(k)} < 0$.
- The conditions for convergence are discussed more in details in [Bertsekas, 1999, Nocedal and Wright, 2006].

Algorithms and variants (seen in this course)

- Steepest descent : $\mathbf{d}^{(k)} = -\nabla F(\mathbf{x}^{(k-1)})$
- Newton algorithm : $\mathbf{d}^{(k)} = -\nabla \mathbf{H}^{-1} \nabla F(\mathbf{x}^{(k-1)})$
- Quasi-Newton : $\mathbf{d}^{(k)} = -\nabla \hat{\mathbf{H}}^{-1} \nabla F(\mathbf{x}^{(k-1)})$.
- Stochastic Gradient Descent : $\mathbf{d}^{(k)} = -\nabla \hat{F}(\mathbf{x}^{(k-1)})$

Gradient descent algorithm

Gradient descent algorithm (steepest descent)

1: Initialize $\mathbf{x}^{(0)}$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad \mathbf{d}^{(k)} \leftarrow -\nabla F(\mathbf{x}^{(k)})$
4: $\quad \rho^{(k)} \leftarrow$ Choose stepsize
5: $\quad \mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}$
6: **end for**

For a step small enough, each iteration decreases the cost : $F(\mathbf{x}^{(k+1)}) \leq F(\mathbf{x}^{(k)})$

Convergence of gradient descent algorithm

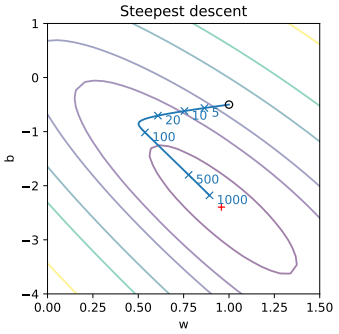
Sufficient conditions for convergence are the **Wolfe conditions**:

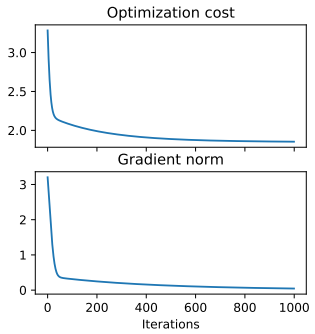
1. $F(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) \leq F(\mathbf{x}^{(k)}) + c_1\rho^{(k)}\nabla F(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)}$
2. $-\nabla F(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)})^T\mathbf{d}^{(k)} \leq -c_2\nabla F(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)}$

With $0 < c_1 < c_2 < 1$ (typically $c_1 = 10^{-4}, c_2 = 0.9$) The first condition is called the **Armijo rule** and the second the **curvature condition**.

More details on convergence in [Nocedal and Wright, 2006, Chapter 3] and [Bertsekas, 1999].

Example of steepest descent





Discussion

- Steepest descent with fixed step $\rho^{(k)} = 0.1$
- Slow convergence around the solution (small gradients).
- After 1000 iterations, still not converged.
- Complexity $\mathcal{O}(nd)$ per iteration.

Gradient descent as Majorization Minimization

Gradient Lispschitz function

A function F is gradient Lispchitz if there exists a constant K such that

$$\|\boldsymbol{\nabla} F_1(\mathbf{x}+\mathbf{p}) - \boldsymbol{\nabla} F_1(\mathbf{x})\| \leq K\|\mathbf{p}\|, \qquad \forall \mathbf{p} \in \mathbb{R}^n, \forall \mathbf{x} \in \mathbb{R}^n. \tag{2}$$

The constant K is called the Lipschitz constant of $\boldsymbol{\nabla} F$ (and $\|\nabla^2 F(\mathbf{x})\|^2$).
Note that if F is gradient Lispchitz, we have the following second order majorization of function F around $\mathbf{x}$, also called descent Lemma:

$$F(\mathbf{x}+\mathbf{p}) \leq F(\mathbf{x}) + \boldsymbol{\nabla} F(\mathbf{x})^T \mathbf{p} + \frac{K}{2}\|\mathbf{p}\|^2, \qquad \forall \mathbf{p} \in \mathbb{R}^n, \forall \mathbf{x} \in \mathbb{R}^n. \tag{3}$$

Gradient descent update as majorization minimization (MM)

At iteration k we can do a majorization of F around $\mathbf{x}^{(k)}$:

$$F(\mathbf{x}^{(k)}+\mathbf{p}) \leq F(\mathbf{x}^{(k)}) + \boldsymbol{\nabla} F(\mathbf{x}^{(k)})^T \mathbf{p} + \frac{K}{2}\|\mathbf{p}\|^2$$

Minimizing the equation above *w.r.t.* $\mathbf{p}$ leads to

$$\mathbf{p}^\star = -\frac{1}{K}\boldsymbol{\nabla} F(\mathbf{x}^{(k)})$$

which corresponds exatly to an update of gradient decsent with step $\rho = \frac{1}{K}$.

Linesearch methods

- For gradient Lischitz functions, a small enough step $\rho^{(k)} = \frac{1}{K}$ ensures a decrease of the cost but convergence is slow.
- We would like to select the "best" step at each iteration :

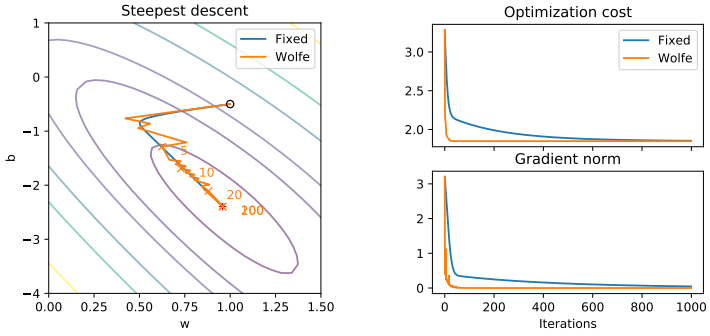
$$\rho^{\star(k)} = \arg\min_{\rho} \quad F(\mathbf{x}^{(k)} + \rho \mathbf{d}^{(k)})$$

- In practice one seeks for a step respecting the Wolfe conditions.
- `scipy.optimize.line_search` implements such a linesearch following [Nocedal and Wright, 2006, Sec 3.5].
- **Backtracking linesearch**

  Initialization of ρ and $0 < \tau < 1$.
  repeat
  $\rho \leftarrow \rho\tau$
  until $F(\mathbf{x}+\rho\mathbf{d}) < F(\mathbf{x}) + \rho c_1 \mathbf{d}^T \boldsymbol{\nabla} F(\mathbf{x})$

  At the end the Armijo rule is respected, since we select the first step the respects it, we usually suppose that the second condition is also respected.
- Note that linesearch can also be used for all gradient descent algorithms (newton, Quasi-newton)

Impact of line search



Discussion

- Linesearch speedup is important *w.r.t.* fixed step.
- Be careful of the number of function call (necessary for linesearch).
- Complexity $\mathcal{O}(knd)$ per iteration where k is the nb of function call.

Convergence of gradient descent and acceleration

Convergence speed of gradient descent

If function F is convex and differentiable and its gradient has a Lipschitz constant L, then the gradient descent with fixed step $\rho^{(k)} = \rho \leq \frac{1}{L}$ converges to a solution $\mathbf{x}^\star$ of the optimization problem with the following speed:

$$F(\mathbf{x}^{(k)}) - F(\mathbf{x}^\star) \leq \frac{\|\mathbf{x}^{(0)} - \mathbf{x}^\star\|^2}{2\rho k} \tag{4}$$

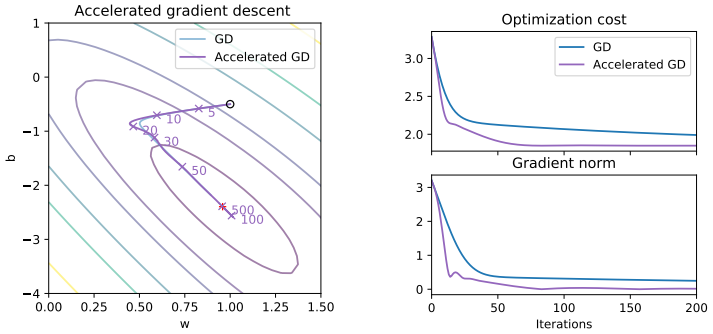
- We say the the gradient descent has a convergence $O(\frac{1}{k})$.
- When the function is strongly convex it has a linear convergence $O(e^{-k/\kappa})$
- Yuri Nesterov proposed acceleration procedures for convex functions [Nesterov, 1983, Nesterov, 2013] that has a $O(\frac{1}{k^2})$ convergence.

Accelerated gradient descent (AGD)

1: Initialize $\mathbf{x}^{(0)}, \mathbf{y}^{(0)} = \mathbf{x}^{(0)}$ and $\rho \leq \frac{1}{L}$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad \mathbf{y}^{(k)} \leftarrow \mathbf{x}^{(k)} + \frac{k-1}{k+2}(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)})$
4: $\quad \mathbf{x}^{(k+1)} \leftarrow \mathbf{y}^{(k)} + \rho \nabla F(\mathbf{x}^{(k)})$
5: **end for**

- Use momentum.
- Compute interpolated position $\mathbf{y}$.
- Do gradient update of $\mathbf{y}$.

Example of Accelerated Gradient Descent



Discussion

- both GD and AGD use fixed step $\rho^{(k)} = 0.1$.
- Acceleration speedup is important *w.r.t.* steepest descent step.
- The momentum due the the Nesterov acceleration can be seen in the trajectory.
- Complexity $\mathcal{O}(nd)$ per iteration when no line search.

Quadratic approximation of the function

- Gradient descent is equivalent to Majorization Minimization when the function in approximated locally by its aupper bound:

$$F(\mathbf{x}+\mathbf{p}) \approx F(\mathbf{x}^{(k)}) + \boldsymbol{\nabla} F(\mathbf{x}^{(k)})^T \mathbf{p} + \frac{K}{2}\|\mathbf{p}\|^2$$

- Where K is the Lipschitz constant of the gradient. K is also an upper bond on the eigenvalues of the Hessian matrix ($\|\nabla^2 F(\mathbf{x})\| \leq K$).
- A better local approximation of the function is:

$$F(\mathbf{x}+\mathbf{p}) \approx F(\mathbf{x}^{(k)}) + \boldsymbol{\nabla} F(\mathbf{x}^{(k)})^T \mathbf{p} + \frac{1}{2}\mathbf{p}^T \mathbf{H} \mathbf{p}$$

  where $\mathbf{H} = \boldsymbol{\nabla}^2 F(\mathbf{x}^{(k)})$ is the Hessian matrix in $\mathbf{x}^{(k)}$.
- Minimizing the approximation above *w.r.t.* $\mathbf{p}$ leads to the following solution:

$$\mathbf{p}^\star = -\mathbf{H}^{-1} \boldsymbol{\nabla} F(\mathbf{x}^{(k)})$$

- Note that the approximation above is not a majorization so the update may not decrease the loss.

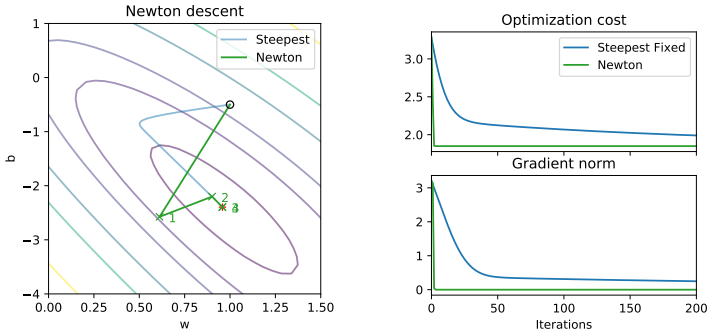
Newton Method

Algorithm of the Newton method

1: Initialize $\mathbf{x}^{(0)}$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad \mathbf{g}^{(k)}, \mathbf{H}^{(k)} \leftarrow$ Compute gradient $\boldsymbol{\nabla} F(\mathbf{x}^{(k)})$ and Hessian matrix $\boldsymbol{\nabla}^2 F(\mathbf{x}^{(k)})$
4: $\quad \mathbf{p}^{(k)} \leftarrow$ Solve linear system $\mathbf{H}^{(k)}\mathbf{p} = -\mathbf{g}^{(k)}$.
5: $\quad \mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} + \mathbf{p}^{(k)}$
6: **end for**

- Requires the resolution of a size d linear system at each iteration ($O(d^3)$).
- Newton method has a quadratic $O(e^{-2^k})$ convergence speed.
- Can also be used with linesearch to ensure cost decrease.
- If F is quadratic, convergence un 1 iteration.
- **Levenberg-Marquardt Modification** : use $\tilde{\mathbf{H}} = \mathbf{H} + \lambda \mathbf{I}$
- Allows to interpolate between Newton ($\lambda = 0$) and gradient descent with small step (large λ).

Example of Newton method



Discussion

- No linesearch, step is 1.
- Very fast convergence (converged in 4 iterations).
- When initial point is far from the solution firsts steps can increase the cost.
- Complexity $\mathcal{O}(nd + d^3)$ per iteration when no line search.

Quasi-Newton and BFGS

Quasi Newton method [Dennis and Moré, 1977]

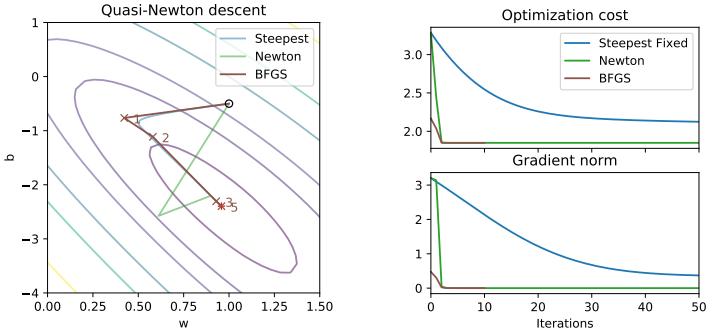
1: Initialize $\mathbf{x}^{(0)}$, $\hat{\mathbf{B}}^{(0)} = \sigma\mathbf{I}$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad \mathbf{p}^{(k)} \leftarrow -\rho^{(k)}\mathbf{B}^{(k)}\boldsymbol{\nabla}F(\mathbf{x}^{(k)})$ with $\rho^{(k)}$ satisfying the Wolfe conditions.
4: $\quad \mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} + \mathbf{p}^{(k)}$
5: $\quad \mathbf{y}^{(k)} \leftarrow \boldsymbol{\nabla}F(\mathbf{x}^{(k+1)}) - \boldsymbol{\nabla}F(\mathbf{x}^{(k)})$
6: $\quad \mathbf{B}^{(k+1)} \leftarrow$ Update $\mathbf{B}^{(k)}$ using previous gradients $\mathbf{y}^{(k)}$ and step $\mathbf{p}^{(k)}$.
7: **end for**

- The problem with Newton: Solving the linear equations is $O(d^3)$
- Principle: estimate and update an inverse matrix approximation $\mathbf{B}$ with efficient $O(d^2)$ updates (Sherman-Morrison formula).
- Most common update strategy is BFGS (Broyden–Fletcher–Goldfarb–Shanno):

$$\mathbf{B}^{(k+1)} = \left(\mathbf{I} - \frac{\mathbf{p}\mathbf{y}^T}{\mathbf{y}^T\mathbf{p}}\right)\mathbf{B}^{(k)}\left(\mathbf{I} - \frac{\mathbf{y}\mathbf{p}^T}{\mathbf{y}^T\mathbf{p}}\right) + \frac{\mathbf{p}\mathbf{p}^T}{\mathbf{y}^T\mathbf{p}}$$

  where $\mathbf{p}$ and $\mathbf{y}$ are expressed without the $^{(k)}$ index.
- Other update strategy include : Broyden, SFP, SR1
- Convergence speed is super-linear (faster than GD, slower than Newton).
- Implemented in `scipy.optimize.minimize` with `method='BFGS'`.

Example of BFGS



Discussion

- Very fast convergence (converged in 11 iterations, VS 4 for Newton).
- First step is Steepest descent (because $\mathbf{B}^{(0)} = \sigma\mathbf{I}$)
- Complexity $\mathcal{O}(nd + d^2)$ per iteration when no line search.

Limited Memory BFGS (L-BFGS)

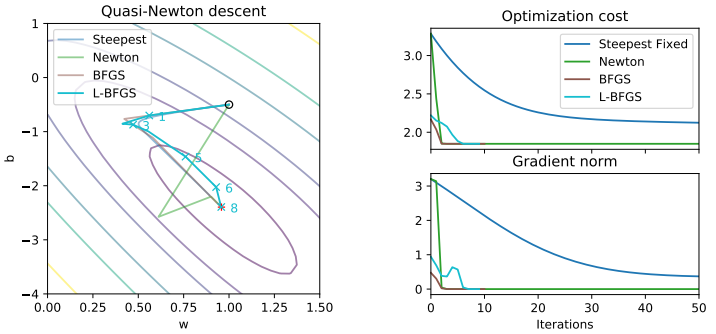
L-BFGS method [Liu and Nocedal, 1989]

- The problem with BFGS: the inverse hessian matrix $\mathbf{B}$ is $O(n^2)$ in memory.
- Limited Memory BFGS only store the last $m \leq d$ updates of $\mathbf{B}$ ($\mathbf{p}$ and $\mathbf{y}$).
- Usually $m < 10$ so memory complexity is $O(d)$.
- Compute the descent direction $\mathbf{B}^{(k)}\boldsymbol{\nabla}F(\mathbf{x}^{(k)})$ recursively.
- Considered one of the most efficient solver for optimization problems maximizing entropy [Malouf, 2002] (our example).
- Implemented in `scipy.optimize.minimize` with `method='L-BFGS-B'`.

L-BFGS variants

- **L-BFGS-B** [Zhu et al., 1997] : Allows to solve smooth optimization problem with box constraints (implemented in scipy)
- **OWL-QN** [Andrew and Gao, 2007] : A variant of L-BFGS dedicated to solve smooth problems with L1 regularization.

Example of L-BFGS



Discussion

- Very fast convergence (nearly as fast as BFGS).
- But requires linesearch.
- First step is Steepest descent (because $\mathbf{B}^{(0)} = \sigma\mathbf{I}$).
- Complexity $\mathcal{O}(nd + md)$ per iteration when no line search.

Optimization in machine learning

Optimization problem

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{w}) \tag{5}$$

- Standard ML problem (supervised or unsupervised learning).
- d is the number of parameter in the model, n the number of training samples.
- Can handle both ERM and regularized learning:
  - Empirical Risk Minimization : $f_i(\mathbf{w}) = (y_i - \mathbf{x}_i^T\mathbf{w})^2$
  - Regularization : $f_i(\mathbf{w}) = (y_i - \mathbf{x}_i^T\mathbf{w})^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$
- Gradient of F is: $\nabla_{\mathbf{w}} F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\mathbf{w}} f_i(\mathbf{w})$

Large sale optimization

- Both n and d can be very large.
- Computation of F and ∇F is $O(nd)$.
- Dataset may not fit in memory.

$\Rightarrow$ Stochastic Gradient Descent.

Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) algorithm

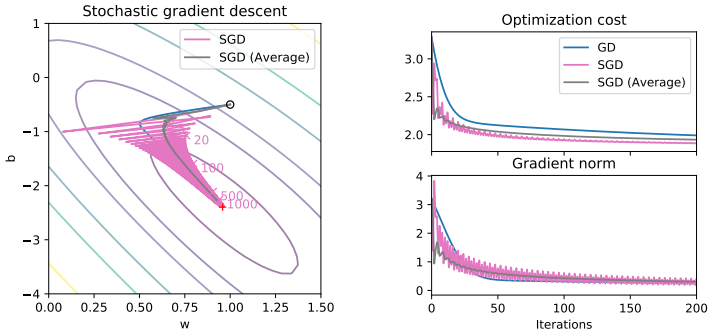
1: Initialize $\mathbf{x}^{(0)}$
2: **for** $k = 0, 1, 2, \dots$ **do**
3: $\quad i^{(k)} \leftarrow$ randomly pick an index $i \in \{1, \dots, n\}$
4: $\quad \mathbf{d}^{(k)} \leftarrow -\nabla_{\mathbf{x}} f_{i^{(k)}}(\mathbf{x}^{(k)})$
5: $\quad \mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}$
6: **end for**

- $\mathbf{d}^{(k)} \in \mathbb{R}^n$ is an approximation of the full gradient.
- Iteration complexity is $O(d)$ VS $O(nd)$ for GD.
- Polyak-Ruppert averaging : $\bar{\mathbf{x}}^{(k)} = \frac{1}{k+1}\sum_{u=0}^{k} \mathbf{x}^{(u)}$
- **Convergence speed** (*e.g.* $\rho^{(k)} = \frac{1}{\sqrt{k}}$) [Nemirovski et al., 2009]

$$E[F(\bar{\mathbf{x}}^{(k)}) - F(\mathbf{x}^\star)] = \begin{cases} O(\frac{1}{\sqrt{k}}) & \text{for } F \text{ convex} \\ O(\frac{1}{k}) & \text{for } F \text{ strongly convex} \end{cases}$$

- A function F is strongly convex if : $\nabla^2 F(\mathbf{x}) \succeq l$ for $l > 0$.

Example of stochastic gradient descent



Discussion

- Decreasing step size : $\rho^{(k)} = \frac{1}{\sqrt{k}}$
- Slow convergence (especially $\bar{\mathbf{x}}^{(k)}$)
- One GD iter $\equiv 4$ SGD iter (since $n = 4$).
- Complexity $\mathcal{O}(d)$ per iteration.

Accelerated stochastic gradients

Stochastic Average Gradient (SAG) [Roux et al., 2012]

1: Initialize $\mathbf{x}^{(0)}, \mathbf{y}_i = \mathbf{0}\ \forall i$
2: **for** $k = 0, 1, 2, \dots$ **do**
3: $\quad i^{(k)} \leftarrow$ randomly pick an index $i \in \{1, \dots, n\}$
4: $\quad \mathbf{y}_{i^{(k)}} \leftarrow \nabla_{\mathbf{x}} f_{i^{(k)}}(\mathbf{x})$
5: $\quad \mathbf{d}^{(k)} \leftarrow -\frac{1}{n}\sum_i \mathbf{y}_i$
6: $\quad \mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}$
7: **end for**

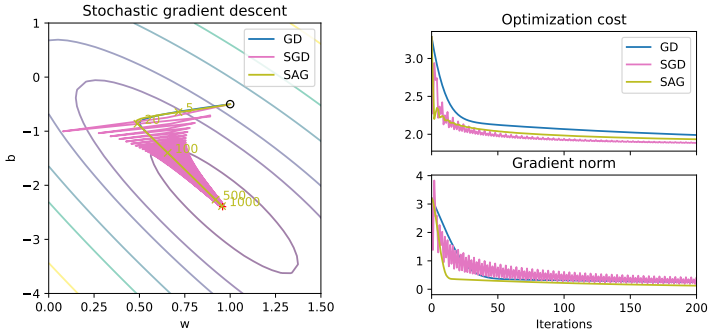
- Keep in memory all previous computed gradients $\mathbf{y}$, update only for sample $i^{(k)}$.
- Iteration is $O(d)$, memory space is $O(nd)$ (same size as data).
- **Convergence speed** [Roux et al., 2012]

$$E[F(\bar{\mathbf{x}}^{(k)}) - F(\mathbf{x}^\star)] = \begin{cases} O(\frac{1}{k}) & \text{for } F \text{ convex} \\ O(e^{-Ck}) & \text{for } F \text{ strongly convex} \end{cases}$$

Other accelerations

- **Stochastic Variance Reduced Gradient (SVRG)** : [Johnson and Zhang, 2013]
- **SAGA**: Better constant than SAG + proximal operators [Defazio et al., 2014]

Example of Stochastic Average Gradient (SAG)



Discussion

- Constant step size : $\rho^{(k)} = 0.1$
- Fast convergence because the problem is strongly convex..
- One GD iter $\equiv 4$ SGD iter (since $n = 4$).
- SAG complexity $\mathcal{O}(d)$ per iteration (but $O(nd)$ in memory).

SGD in machine learning

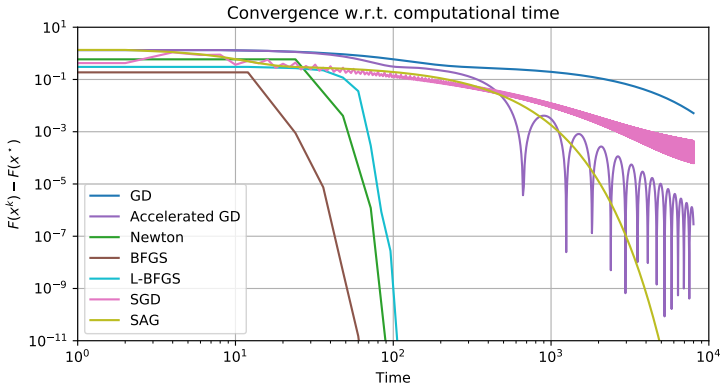
Large scale optimization

- Used for training linear and non-linear models on very large datasets.
- Sate of the art algorithm for linear SVM, logistic regression, least square.
- Use minibatches (compute stochastic gradient on multiple samples).
- Classification (SVM,Logistic) : `sklearn.linear_model.SGDClassifier`.
- Regression (lesta square, huber) : `sklearn.linear_model.SGDRegressor`.

Training Neural Networks with SGD

- Usually use fixed step (against theory).
- Use early stopping as regularization (avoid overfitting).
- Works very well on continuous, nonconvex problems but not very well understood.
- Several momentum averaging and adaptive step size strategies:
  - RMSPROP [Tieleman and Hinton, 2012].
  - Adaptive gradient step ADAGRAD [Duchi et al., 2011].
  - Adaptive Moment estimation ADAM [Kingma and Ba, 2014].

Example convergence of GD methods



Discussion

- Comparison of all methods as a function of computational time.
- Rank of methods (fastest left):
  BFGS $\approx$ Newton $\approx$ L-BFGS $<$ SAG $<$ AGD $<$ SGD $<$ GD
- Very small dataset, so SGD/SAG do not have an advantage ($n = 4, d = 2$).

Running time complexity of GD methods

Method	Iteration	Convergence	Running time
GD	nd	$1/k$	dn/ϵ
AGD	nd	$1/k^2$	$dn/\sqrt{\epsilon}$
Newton	$nd^2 + d^3$	$\exp(-2^k)$	$d(nd + d^2)\log\log(1/\epsilon)$
BFGS	$nd + d^2$	$< \exp(-k)$	$< d(n + d)\log(1/\epsilon)$
L-BFGS	$nd + md$	$< \exp(-k)$	$< d(n + m)\log(1/\epsilon)$
SGD	d	$1/\sqrt{k}$	d/ϵ^2
SAG	d	$1/k$	$d\sqrt{n}/\epsilon$

- For a convex function F, with K Lisphitz gradients.
- Running time for reaching ϵ optimality is provided.
- When F is l strongly convex we have the following running times:
  - GD : $O(dnc\log(1/\epsilon))$
  - AGD : $O(dn\sqrt{c}\log(1/\epsilon))$
  - SGD : $O(dc\log(1/\epsilon))$
  - SAG : $O(d(n + c)\log(1/\epsilon))$

  with $c = \frac{K}{l}$ the condition number of the problem and $\nabla^2 F(\mathbf{x}) \succeq l$.

References I

[Andrew and Gao, 2007] Andrew, G. and Gao, J. (2007).
Scalable training of l 1-regularized log-linear models.
In *Proceedings of the 24th international conference on Machine learning*, pages 33–40.

[Bertsekas, 1999] Bertsekas, D. P. (1999).
Nonlinear programming.

[Defazio et al., 2014] Defazio, A., Bach, F., and Lacoste-Julien, S. (2014).
Saga: A fast incremental gradient method with support for non-strongly convex composite objectives.
In *Advances in neural information processing systems*, pages 1646–1654.

[Dennis and Moré, 1977] Dennis, Jr, J. E. and Moré, J. J. (1977).
Quasi-newton methods, motivation and theory.
SIAM review, 19(1):46–89.

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011).
Adaptive subgradient methods for online learning and stochastic optimization.
Journal of machine learning research, 12(Jul):2121–2159.

[Johnson and Zhang, 2013] Johnson, R. and Zhang, T. (2013).
Accelerating stochastic gradient descent using predictive variance reduction.
In *Advances in neural information processing systems*, pages 315–323.

References II

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014).
Adam: A method for stochastic optimization.
arXiv preprint arXiv:1412.6980.

[Liu and Nocedal, 1989] Liu, D. C. and Nocedal, J. (1989).
On the limited memory bfgs method for large scale optimization.
Mathematical programming, 45(1-3):503–528.

[Malouf, 2002] Malouf, R. (2002).
A comparison of algorithms for maximum entropy parameter estimation.
In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–7.
Association for Computational Linguistics.

[Nemirovski et al., 2009] Nemirovski, A., Juditsky, A., Lan, G., and Shapiro, A. (2009).
Robust stochastic approximation approach to stochastic programming.
SIAM Journal on optimization, 19(4):1574–1609.

[Nesterov, 2013] Nesterov, Y. (2013).
Introductory lectures on convex optimization: A basic course, volume 87.
Springer Science & Business Media.

References III

[Nesterov, 1983] Nesterov, Y. E. (1983).
A method for solving the convex programming problem with convergence rate o (1/k^ 2).
In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. (2006).
Numerical optimization.
Springer Science & Business Media.

[Roux et al., 2012] Roux, N. L., Schmidt, M., and Bach, F. R. (2012).
A stochastic gradient method with an exponential convergence _rate for finite training sets.
In *Advances in neural information processing systems*, pages 2663–2671.

[Tieleman and Hinton, 2012] Tieleman, T. and Hinton, G. (2012).
Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.
COURSERA: Neural networks for machine learning, 4(2):26–31.

[Zhu et al., 1997] Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1997).
Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization.
ACM Transactions on Mathematical Software (TOMS), 23(4):550–560.