Modern Optimization Techniques

2. Unconstrained Optimization / 2.2. Stochastic Gradient Descent

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute of Computer Science
University of Hildesheim, Germany

Syllabus

Mon. 30.10.	(0)	0. Overview
		1. Theory
Mon. 6.11.	(1)	1. Convex Sets and Functions
		2. Unconstrained Optimization
Mon. 13.11.	(2)	2.1 Gradient Descent
Mon. 20.11.	(3)	2.2 Stochastic Gradient Descent
Mon. 27.11.	(4)	2.3 Newton's Method
Mon. 4.12.	(5)	2.4 Quasi-Newton Methods
Mon. 11.12.	(6)	2.5 Subgradient Methods
Mon. 18.12.	(7)	2.6 Coordinate Descent
	—	— *Christmas Break* —
		3. Equality Constrained Optimization
Mon. 8.1.	(8)	3.1 Duality
Mon. 15.1.	(9)	3.2 Methods
		4. Inequality Constrained Optimization
Mon. 22.1.	(10)	4.1 Primal Methods
Mon. 29.1.	(11)	4.2 Barrier and Penalty Methods
Mon. 5.2.	(12)	4.3 Cutting Plane Methods

Outline

Outline

Unconstrained Convex Optimization

$$\arg\min_{x \in \operatorname{dom} f} f(x)$$

- $\operatorname{dom} f \subseteq \mathbb{R}^N$ is open (unconstrained optimization)
- f is convex

Stochastic Gradient

Gradient Descent makes use of the gradient

$$\nabla f(x)$$

Stochastic Gradient Descent: makes use of Stochastic Gradient only:

$$g(x) \sim p(g \in \mathbb{R}^N \mid x), \quad \mathbb{E}_p(g(x)) = \nabla f(x)$$

- for each point $x \in \mathbb{R}^N$:
  random variable over $\mathbb{R}^N$ with distribution p (conditional on x)
- on average yields the gradient (at each point)

Stochastic Gradient / Example: Big Sums

f is a "big sum":

$$f(x) = \frac{1}{C} \sum_{c=1}^{C} f_c(x)$$

$$\text{with } f_c \text{ convex}, \quad c = 1, \dots, C$$

g is the gradient of a random summand:

$$p(g \mid x) := \text{Unif}(\{\nabla f_c(x) \mid c = 1, \dots, C\})$$

Stochastic Gradient / Example: Least Squares

$$\min_{x \in \mathbb{R}^N} f(x) := ||Ax - b||_2^2$$

- will find solution for $Ax = b$ if there is any (then $||Ax - b||_2 = 0$)
- otherwise will find the x where the difference $Ax - b$ of left and right side is as small as possible (in the squared L2 norm)
- is a big sum:

$$f(x) := ||Ax - b||_2^2 = \sum_{m=1}^{M} ((Ax)_m - b_m)^2 = \sum_{m=1}^{M} (A_{m,.}x - b_m)^2$$

$$= \frac{1}{M} \sum_{m=1}^{M} f_m(x), \quad f_m(x) := M(A_{m,.}x - b_m)^2$$

- stochastic gradient g:
  - gradient for a random component m

Stochastic Gradient / Example: Supervised Learning

$$\min_{\theta \in \mathbb{R}^P} f(x) := \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, \hat{y}(x_n, \theta)) + \lambda ||\theta||_2^2$$

- where
  - $(x_n, y_n) \in \mathbb{R}^M \times \mathbb{R}^T$ are N training samples,
  - $\hat{y}$ is a parametrized model, e.g., logistic regression

    $$\hat{y}(x; \theta) := (1 + e^{-\theta^T x})^{-1}, \quad P := M, T := 1$$

  - ℓ is a loss, e.g., negative binomial loglikelihood:

    $$\ell(y, \hat{y}) := -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

  - $\lambda \in \mathbb{R}_0^+$ is the regularization weight.
- will find parametrization with best trade-off between low loss and low model complexity

Stochastic Gradient / Example: Supervised Learning (2/2)

$$\min_{\theta \in \mathbb{R}^P} f(x) := \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, \hat{y}(x_n, \theta)) + \lambda ||\theta||_2^2$$

- where
  - $(x_n, y_n) \in \mathbb{R}^M \times \mathbb{R}^T$ are N training samples,
  - ...
- is a big sum:

$$f(\theta) := \frac{1}{N} \sum_{n=1}^{N} f_n(\theta), \quad f_n(\theta) := \ell(y_n, \hat{y}(x_n, \theta)) + \lambda ||\theta||_2^2$$

- stochastic gradient g:
  - gradient for a random sample n

Stochastic Gradient Descent

- the very same as Gradient Descent
- but use stochastic gradient $g(x)$ instead of exact gradient $\nabla f(x)$ in each step

min-sgd$(f, p, x^{(0)}, \mu, K)$:
  for $k := 1, \dots, K$:
    draw $g^{(k-1)} \sim p(g \mid x)$
    $\Delta x^{(k-1)} := -g^{(k-1)}$
    $\mu^{(k-1)} := \mu(f, x^{(k-1)}, \Delta x^{(k-1)})$
    $x^{(k)} := x^{(k-1)} + \mu^{(k-1)} \Delta x^{(k-1)}$
    if **converged**$(\dots)$:
      return $\mathbf{x}^{(k)}$
  raise exception "not converged in K iterations"

where

- p (distribution of the) stochastic gradient of f

Stochastic Gradient Descent / For Big Sums

min-sgd$((f_c)_{c=1,\ldots,C},\ (\nabla f_c)_{c=1,\ldots,C},\ x^{(0)},\ \mu,\ K)$:
  for $k := 1, \ldots, K$:
    draw $c^{(k-1)} \sim \text{Unif}(1, \ldots, C)$
    $g^{(k-1)} := \nabla f_{c^{(k-1)}}(x^{(k-1)})$
    $\Delta x^{(k-1)} := -g^{(k-1)}$
    $\mu^{(k-1)} := \mu(f, x^{(k-1)}, \Delta x^{(k-1)})$
    $x^{(k)} := x^{(k-1)} + \mu^{(k-1)} \Delta x^{(k-1)}$
    if **converged**$(\ldots)$:
      return $\mathbf{x}^{(k)}$
  raise exception "not converged in K iterations"

where

- $(f_c)_{c=1,\ldots,C}$ objective function summands, $f := \frac{1}{C} \sum_{c=1}^{C} f_c$
- $(\nabla f_c)_{c=1,\ldots,C}$ gradients of the objective function summands

SGD / For Big Sums / Epochs

$\text{min-sgd}((f_c)_{c=1,\ldots,C},\ (\nabla f_c)_{c=1,\ldots,C},\ x^{(0)},\ \mu,\ K)$:
$\quad \mathcal{C} := (1, 2, \ldots, C)$
$\quad x^{(0,C)} := x^{(0)}$
$\quad$ for $k := 1, \ldots, K$:
$\qquad$ randomly shuffle $\mathcal{C}$
$\qquad x^{(k,0)} := x^{(k-1,C)}$
$\qquad$ for $i = 1, \ldots, C$:
$\qquad\quad g^{(k,i-1)} := \nabla f_{c_i}(x^{(k,i-1)})$
$\qquad\quad \Delta x^{(k,i-1)} := -g^{(k,i-1)}$
$\qquad\quad \mu^{(k,i-1)} := \mu(f, x^{(k,i-1)}, \Delta x^{(k,i-1)})$
$\qquad\quad x^{(k,i)} := x^{(k,i-1)} + \mu^{(k,i-1)} \Delta x^{(k,i-1)}$
$\quad$ return $x^{(K,C)}$

where

- $(f_c)_{c=1,\ldots,C}$ objective function summands, $f := \frac{1}{C}\sum_{c=1}^{C} f_c$
- ...
- K number of epochs

Convergence of SGD

Theorem (Convergence of SGD)

If

(i) *f is strongly convex ($||\nabla^2 f(x)|| \succeq mI, m \in \mathbb{R}^+$),*

(ii) *the expected squared norm of its stochastic gradient g is uniformly bounded ($\exists G \in \mathbb{R}_0^+ \; \forall x : \mathbb{E}(||g(x)||^2) \leq G^2$) and*

(iii) *the steplength $\mu^{(k)} := \frac{1}{m(k+1)}$ is used, then*

$$\mathbb{E}_p(||x^{(k)} - x^*||^2) \leq \frac{1}{k+1} \max\{||x^{(0)} - x^*||^2, \frac{G^2}{m^2}\}$$

Convergence of SGD / Proof

$$f(x^*) - f(x) \geq \nabla f(x)^T (x^* - x) + \frac{m}{2}||x^* - x||^2 \quad \text{str. conv. (i)}$$

$$f(x) - f(x^*) \geq \nabla f(x^*)^T (x - x^*) + \frac{m}{2}||x - x^*||^2 = \frac{m}{2}||x^* - x||^2$$

summing both yields

$$0 \geq \nabla f(x)^T (x^* - x) + m||x^* - x||^2$$

$$\nabla f(x)^T (x - x^*) \geq m||x^* - x||^2 \qquad (1)$$

$$\begin{aligned}
&\mathbb{E}(||x^{(k)} - x^*||^2) \\
&= \mathbb{E}(||x^{(k-1)} - \mu^{(k-1)} g^{(k-1)} - x^*||^2) \\
&= \mathbb{E}(||x^{(k-1)} - x^*||^2) - 2\mu^{(k-1)} \mathbb{E}((g^{(k-1)})^T (x^{(k-1)} - x^*)) + (\mu^{(k-1)})^2 \mathbb{E}(||g^{k-1}||^2) \\
&= \mathbb{E}(||x^{(k-1)} - x^*||^2) - 2\mu^{(k-1)} \mathbb{E}(\nabla f(x^{(k-1)})^T (x^{(k-1)} - x^*)) + (\mu^{(k-1)})^2 \mathbb{E}(||g^{k-1}|| \\
&\overset{(ii),(1)}{\leq} \mathbb{E}(||x^{(k-1)} - x^*||^2) - 2\mu^{(k-1)} m \mathbb{E}(||x^* - x^{(k-1)}||^2) + (\mu^{(k-1)})^2 G^2 \\
&= (1 - 2\mu^{(k-1)} m) \mathbb{E}(||x^{(k-1)} - x^*||^2) + (\mu^{(k-1)})^2 G^2 \qquad (2)
\end{aligned}$$

Convergence of SGD / Proof (2/2)

induction over k: $k := 0$:

$$||x^{(0)} - x^*||^2 \leq \frac{1}{1} L, \quad L := \max\{||x^{(0)} - x^*||^2, \frac{G^2}{m^2}\}$$

$k > 0$:

$$\begin{aligned}
\mathbb{E}(||x^{(k)} - x^*||^2) &\overset{(2)}{\leq} (1 - 2\mu^{(k-1)} m)\mathbb{E}(||x^{(k-1)} - x^*||^2) + (\mu^{(k-1)})^2 G^2 \\
&\overset{(iii)}{=} (1 - \frac{2}{k})\mathbb{E}(||x^{(k-1)} - x^*||^2) + \frac{G^2}{m^2 k^2} \\
&\overset{\text{ind.hyp.}}{\leq} (1 - \frac{2}{k})\frac{1}{k} L + \frac{G^2}{m^2 k^2} \\
&\overset{\text{def. } L}{\leq} (1 - \frac{2}{k})\frac{1}{k} L + \frac{1}{k^2} L \\
&= \frac{k-2}{k^2} L + \frac{1}{k^2} L = \frac{k-1}{k^2} L \leq \frac{1}{k+1} L
\end{aligned}$$

Outline

Choosing the step size for SGD

- The step size μ is a crucial parameter to be tuned
- Given the low cost of the SGD update, using line search for the step size is a bad choice
- Possible alternatives:
  - Fixed step size
  - Armijo principle
  - Bold-Driver
  - Adagrad

Example: Body Fat prediction

We want to estimate the percentage of body fat based on various attributes:

- Age (years)
- Weight (lbs)
- Height (inches)
- Neck circumference (cm)
- Chest circumference (cm)
- Abdomen 2 circumference (cm)
- Hip circumference (cm)
- Thigh circumference (cm)
- Knee circumference (cm)
- ...

http://lib.stat.cmu.edu/datasets/bodyfat

Example: Body Fat prediction

The data is represented it as:

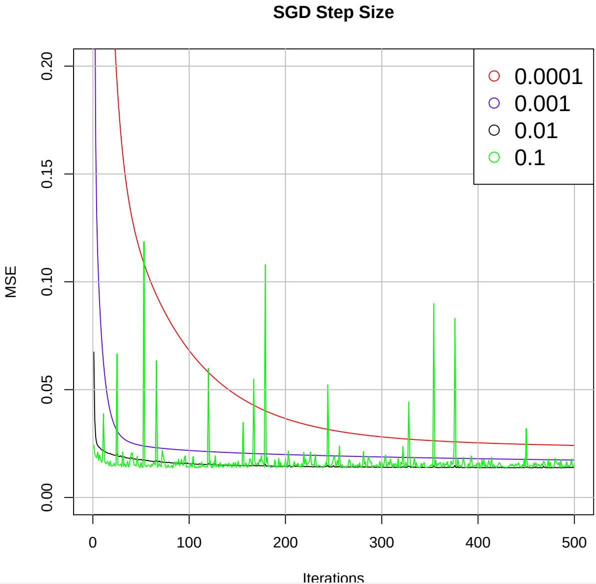
$$A = \begin{pmatrix} 1 & a_{1,1} & a_{1,2} & \dots & a_{1,M} \\ 1 & a_{2,1} & a_{2,2} & \dots & a_{2,M} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_{N,1} & a_{N,2} & \dots & a_{N,M} \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

with $N = 252$, $M = 14$

We can model the percentage of body fat y
as a linear combination of the body measurements with parameters $\mathbf{x}$:

$$\hat{y}_n = \mathbf{x}^T \mathbf{a_n} = x_0 1 + x_1 a_{n,1} + x_2 a_{n,2} + \ldots + x_M a_{n,M}$$

SGD - Fixed Step Size on the Body Fat dataset



Bold Driver Heuristic

- The Bold Driver Heuristic makes the assumption that smaller step sizes are needed when closer to the optimum
- It adjusts the step size based on the value of $f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k-1)})$
- If the value of $f(\mathbf{x})$ grows, the step size must decrease
- If the value of $f(\mathbf{x})$ decreases, the step size can be larger for faster convergence

Bold Driver Heuristic - Update Rule

We need to define

- an **increase factor** $\mu^+ > 1$, e.g. $\mu^+ := 1.05$, and
- a **decay factor** $\mu^- \in (0, 1)$, e.g., $\mu^- := 0.5$.

Step size update rule:

- adapt stepsize only **once after each epoch**, not for every (inner) iteration.
- Cycle through the whole data and update the parameters
- Evaluate the objective function $f(\mathbf{x}^{(k)})$
- if $f(\mathbf{x}^{(k)}) < f(\mathbf{x}^{(k-1)})$ then $\mu \rightarrow \mu^+\mu$
- else $f(\mathbf{x}^{(k)}) > f(\mathbf{x}^{(k-1)})$ then $\mu \rightarrow \mu^-\mu$
- different from the bold driver heuristics for batch gradient descent, there is no way to evaluate $f(x + \mu\Delta x)$ for different μ.
    - stepsize μ is adapted once after the step has been done

Bold Driver

```
stepsize-bd(μ, f_new, f_old, μ⁺, μ⁻):
    if f_new < f_old
        μ := μ⁺ μ
    else
        μ := μ⁻ μ
    return μ
```

where

- μ stepsize of last update
- $f_{\text{new}}, f_{\text{old}} = f(x^k), f(x^{k-1})$ function values before and after the last update
- μ^+, μ^- stepsize increase and decay factors

Considerations

- Works well for a range of problems
- The initial μ just needs to be large enough
- μ^+ and μ^- have to be adjusted to the problem at hand
- May lead to faster convergence

AdaGrad

- Adagrad adjusts the step size **individually for each variable** to be optimized
- It uses information about the past gradients
- Leads to faster convergence
- Less sensitive to the choice of the step size

AdaGrad - Update Rule

We have

$$f(\mathbf{x}) = \sum_{m=1}^{M} f_m(\mathbf{x})$$

Update rule:

- Update stepsize for every inner iteration
- Pick a random instance $m \sim \text{Uniform}(1, M)$
- Compute the gradient $\nabla_{\mathbf{x}} f_m(\mathbf{x})$
- Update the gradient history $\mathbf{h} := \mathbf{h} + \nabla_{\mathbf{x}} f_m(\mathbf{x}) \circ \nabla_{\mathbf{x}} f_m(\mathbf{x})$
- The step size for variable $\mathbf{x}_n$ is $\mu_n := \frac{\mu_0}{\sqrt{h_n}}$
- Update

$$\mathbf{x}^{\text{next}} := \mathbf{x} - \mu \circ \nabla_{\mathbf{x}} f_m(\mathbf{x})$$

$$\text{i.e., } \mathbf{x}_n^{\text{next}} := \mathbf{x}_n - \frac{\mu_0}{\sqrt{h_n}} (\nabla_{\mathbf{x}} f_m(\mathbf{x}))_n$$

$\circ$ denotes the elementwise product.

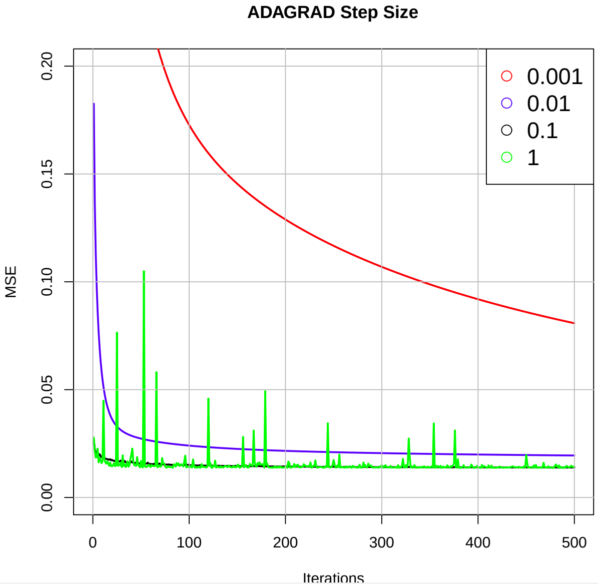
AdaGrad

stepsize-adagrad(g, h, μ_0):
  $h := h + g \circ g$
  $\mu_n := \mu_0 / \sqrt{h_n}$ for $n = 1, \ldots, N$
  return (μ, h)

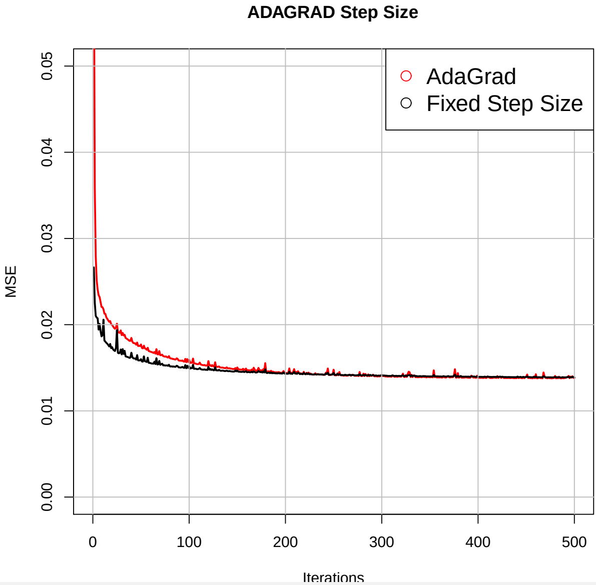
where

- returns a vector of stepsizes, one for each variable
- $g = \nabla f_m(x)$ current (stochastic) gradient
- h past gradient size history
- μ_0 initial stepsize

AdaGrad Step Size



AdaGrad vs Fixed Step Size



Outline

Practical Example: Household Spending

Suppose we have the following data about different households:

- Number of workers in the household (a_1)
- Household composition (a_2)
- Region (a_3)
- Gross normal weekly household income (a_4)
- **Weekly household spending** (y)

We want to creat a model of the weekly household spending

Practical Example: Household Spending

If we have data about m households, we can represent it as:

$$A_{m,n} = \begin{pmatrix} 1 & a_{1,2} & \dots & a_{1,n} \\ 1 & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & a_{m,2} & \dots & a_{m,n} \end{pmatrix} \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

We can model the household consumption is a linear combination of the household features with parameters $\mathbf{x}$:

$$\hat{y}_i = \mathbf{x}^T \mathbf{a_i} = x_0 1 + x_1 a_{i,1} + x_2 a_{i,2} + x_3 a_{i,3} + x_4 a_{i,4}$$

Least Square Problem Revisited

The following least square problem

$$\text{minimize} \quad ||A\mathbf{x} - \mathbf{y}||_2^2$$

Can be rewritten as

$$\text{minimize} \quad \sum_{i=1}^{m} (\mathbf{x}^T \mathbf{a_i} - y_i)^2$$

$$A_{m,n} = \begin{pmatrix} 1 & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 1 & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_{m,1} & a_{m,2} & a_{m,3} & a_{m,4} \end{pmatrix} \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

The Gradient Descent update rule

For the problem

$$\text{minimize} \quad \sum_{i=1}^{m} (\mathbf{x}^T \mathbf{a_i} - y_i)^2$$

The the gradient $\nabla f(\mathbf{x})$ of the objective function is:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = 2 \sum_{i=1}^{m} (\mathbf{x}^T \mathbf{a_i} - y_i) \mathbf{a_i}$$

The Gradient Descent update rule is then:

$$\mathbf{x} \to \mathbf{x} - \mu \left(2 \sum_{i=1}^{m} (\mathbf{x}^T \mathbf{a_i} - y_i) \mathbf{a_i} \right)$$

The Gradient Descent update rule

We need to "see" all the data before updating **x**

$$\mathbf{x} \rightarrow \mathbf{x} - \mu \left(2 \sum_{i=1}^{m} (\mathbf{x}^T \mathbf{a_i} - y_i) \mathbf{a_i} \right)$$

Can we make any progress before iterating over all the data?

Decomposing the objective function

The objective function

$$f(\mathbf{x}) = \sum_{i=1}^{m} (\mathbf{x}^T \mathbf{a_i} - y_i)^2$$

Can be expressed as a function of the objective on each data point $(\mathbf{a}, y)$:

$$f_i(\mathbf{x}) = (\mathbf{x}^T \mathbf{a_i} - y_i)^2$$

So that

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x})$$

A simpler update rule

Now that we have

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x})$$

We can define the following update rule

- Pick a random instance $i \sim \text{Uniform}(1, m)$
- Update $\mathbf{x}$

$$\mathbf{x} \rightarrow \mathbf{x} + \mu \left(-\nabla_{\mathbf{x}} f_i(\mathbf{x})\right)$$

Stochastic Gradient Descent (SGD)

procedure StochasticGradiendDescent
input: f, μ
  Get initial point $\mathbf{x}$
  repeat
    for $i \in 1, \ldots, m$ **do**
      $\mathbf{x} \rightarrow \mathbf{x} - \mu \nabla f_i(\mathbf{x})$
    end for
  until convergence
  return $\mathbf{x}$, $f(\mathbf{x})$
end procedure

SGD and Least Squares

We have

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x})$$

with

$$f_i(\mathbf{x}) = (\mathbf{x}^T \mathbf{a_i} - y_i)^2$$

The update rule is

$$\nabla_{\mathbf{x}} f_i(\mathbf{x}) = 2(\mathbf{x}^T \mathbf{a_i} - y_i)\mathbf{a_i}$$

$$\mathbf{x} \rightarrow \mathbf{x} - \mu \left(2(\mathbf{x}^T \mathbf{a_i} - y_i)\mathbf{a_i} \right)$$

SGD vs. GD

procedure SGD
  input: f, μ
  Get initial point $\mathbf{x}$
  repeat
    for $i \in 1, \ldots, m$ **do**
      $\mathbf{x} \rightarrow \mathbf{x} - \mu \nabla f_i(\mathbf{x})$
    end for
  until convergence
  return $\mathbf{x}$, $f(\mathbf{x})$
end procedure

procedure GradientDescent
  input: f
  Get initial point $\mathbf{x}$
  repeat
    Get Step Size μ
    $\mathbf{x} := \mathbf{x} - \mu \nabla f(\mathbf{x})$
  until convergence
  return $\mathbf{x}$, $f(\mathbf{x})$
end procedure

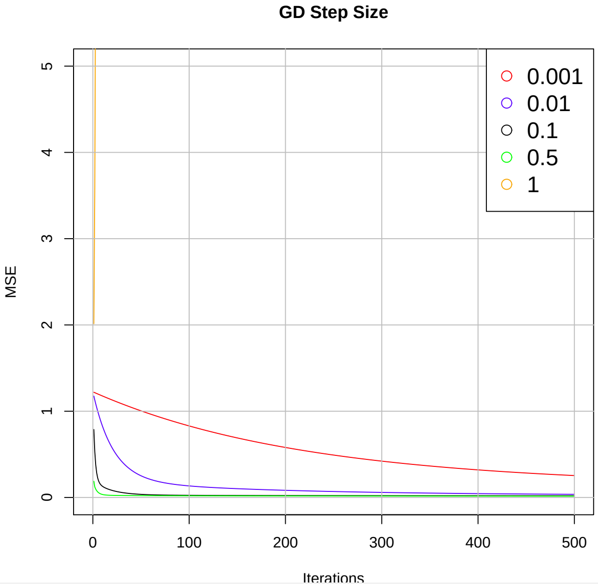
SGD vs. GD - Least Squares

procedure SGD
input: f, μ
  Get initial point $\mathbf{x}$
  repeat
    for $i \in 1, \ldots, m$ **do**
$$\mathbf{x} \rightarrow \mathbf{x} - \mu \left(2(\mathbf{x}^T \mathbf{a_i} - y_i)\mathbf{a_i} \right)$$
    end for
  until convergence
  return $\mathbf{x}$, $f(\mathbf{x})$
end procedure

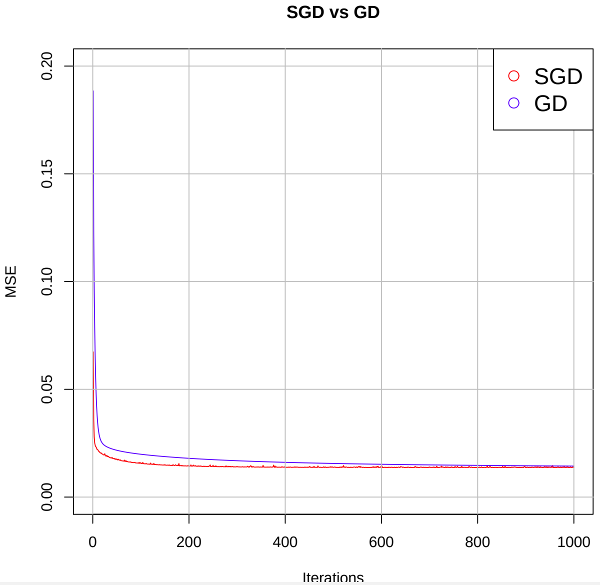
procedure GD
input: f
  Get initial point $\mathbf{x}$
  repeat
    Get Step Size μ
$$\mathbf{x} \rightarrow \mathbf{x} - \mu \left(2 \sum_{i=1}^{m} (\mathbf{x}^T \mathbf{a_i} - y_i)\mathbf{a_i} \right)$$
  until convergence
  return $\mathbf{x}$, $f(\mathbf{x})$
end procedure

Outline

GD Step Size


GD Step Size
MSE
0
1
2
3
4
5
0
100
200
300
400
500
Iterations
0.001
0.01
0.1
0.5
1

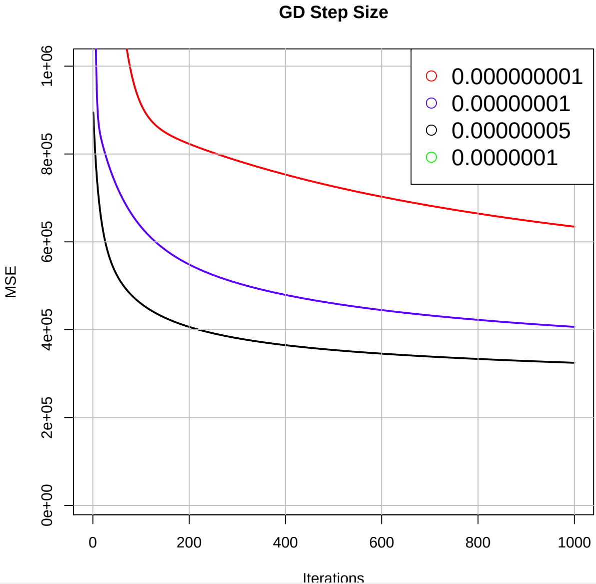
SGD vs GD - Body Fat Dataset



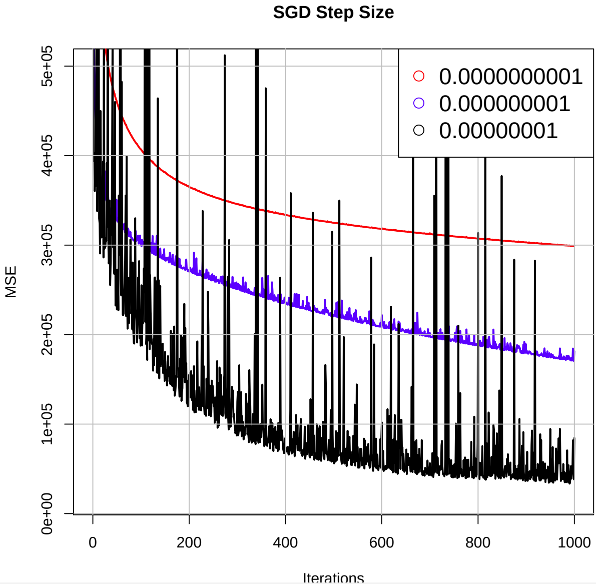
Year Prediction Data Set

- Least Squares Problem
- Prediction of the release year of a song from audio features
- 90 features
- Experiments done on a subset of 1000 instances of the data

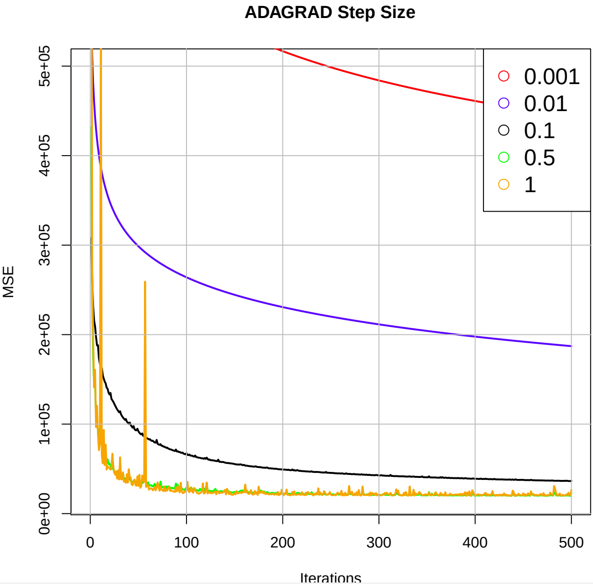
GD Step Size - Year Prediction



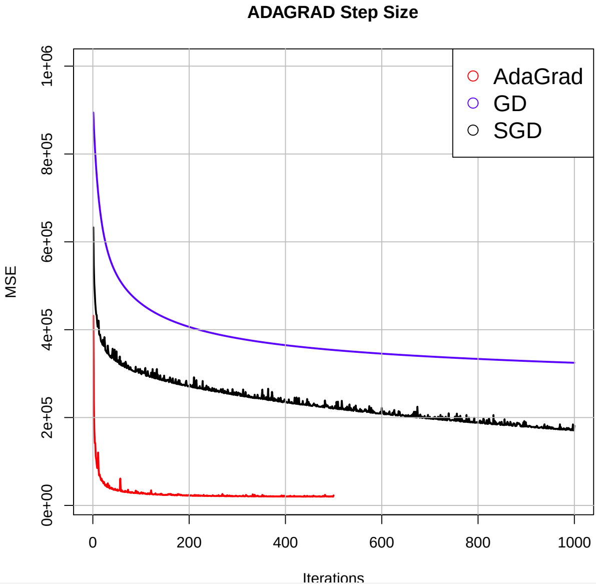
SGD Step Size - Year Prediction



AdaGrad Step Size - Year Prediction



AdaGrad vs SGD vs GD - Year Prediction



Summary

- Stochastic Gradient Descent (SGD) is like Gradient Descent,
  - but instead of the exact gradient uses
    just a random vector called stochastic gradient
    - with expectation of the true/exact gradient.

- stochastic gradients occur naturally when the objective is a big sum
  - then the gradient of a uniformly random component is a stochastic gradient
  - e.g., objectives for most machine learning problems are big sums over instance-wise losses (and regularization terms).

- SGD converges with a rate of $1/k$ in the number of steps k.

Summary (2/2)

- Step length and convergence critera have to be adapted
  - to aggregate over several update steps, e.g., an epoche
  - cannot test for different step lengths (like backtracking)
- Bold driver steplength control:
  - update per epoche based on additional function evaluation.
- Adagrad steplength control:
  - individual steplength for each variable
  - $1/\sum g^2$ for past gradients.

Further Readings

- SGD is not covered in Boyd and Vandenberghe [2004].
- Leon Bottou, Frank E. Curtis, Jorge Nocedal (2016): *Stochastic Gradient Methods for Large-Scale Machine Learning*, ICML 2016 Tutorial, `http://users.iems.northwestern.edu/~nocedal/ICML`
- Francis Bach (2013): *Stochastic gradient methods for machine learning*, Microsoft Machine Learning Summit 2013, `http://research.microsoft.com/en-us/um/cambridge/events/mls2013/downloads/stochastic_gradient.pdf`
- for the convergence proof:
  Ji Liu (2014), Notes "Stochastic Gradient Descent",
  `http://www.cs.rochester.edu/~jliu/CSC-576-2014fall.html`

References I

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge Univ Press, 2004.