Stochastic Gradient Descent

Pontus Giselsson

Finite sum problems

Finite sum problems

- Yesterday, you saw problems of the form

$$\text{minimize}\, f(x) + g(x)$$

  where
  - f is smooth (and potentially convex)
  - g is nonsmooth and convex
- Algorithm: proximal gradient method
- Sometimes there is additional structure, we will treat

$$\text{minimize} \underbrace{\sum_{i=1}^{N} f_i(x)}_{f(x)}$$

  where f is of finite sum form (and $g \equiv 0$)
- Can be solved by gradient method
- If N is large, stochastic gradient descent is often preferrable

Why finite sum?

Finite sum problems appear naturally, e.g., in supervised learning

What is supervised learning?

- Let (x, y) represent object and label pairs
  - Object $x \in \mathcal{X} \subseteq \mathbb{R}^n$
  - Label $y \in \mathcal{Y} \subseteq \mathbb{R}^K$
- Available: Labeled training data (training set) $\{(x_i, y_i)\}_{i=1}^N$
  - Data $x_i \in \mathbb{R}^n$, or *examples* (often n large)
  - Labels $y_i \in \mathbb{R}^K$, or *response variables* (often $K = 1$)

Objective: Find a model (function) $m(x)$:

- that takes data (example, object) x as input
- and predicts corresponding label (response variable) y

How?:

- learn m from training data, but should *generalize* to all (x, y)

Relation to optimization

Training the "machine" m consists in solving optimization problem

Regression vs Classification

There are two main types of supervised learning tasks:

- Regression:
  - Predicts quantities
  - Real-valued labels $y \in \mathcal{Y} = \mathbb{R}^K$ (will mainly consider $K = 1$)
- Classification:
  - Predicts class belonging
  - Finite number of class labels, e.g., $y \in \mathcal{Y} = \{1, 2, \ldots, k\}$

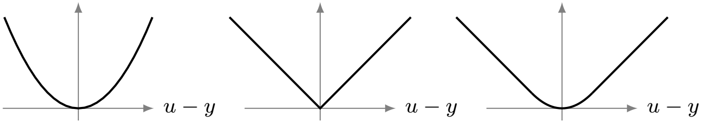
Regression training problem

- Objective: Find data model m such that for all (x, y):

$$m(x) - y \approx 0$$

- Let model output $u = m(x)$; Examples of data misfit losses

$$L(u, y) = \tfrac{1}{2}(u - y)^2$$
$$L(u, y) = |u - y|$$
$$L(u, y) = \begin{cases} \frac{1}{2}(u-y)^2 & \text{if } |u - v| \le c \\ c(|u - y| - c/2) & \text{else} \end{cases}$$



Square | 1-norm | Huber

- Training: find model m that minimizes sum of training set losses

$$\underset{m}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i), y_i)$$

Supervised learning – Least squares

- Parameterize model m and set a linear (affine) structure

$$m(x;\theta) = w^T x + b$$

  where $\theta = (w, b)$ are *parameters* (also called *weights*)

- Training: find model parameters that minimize training cost

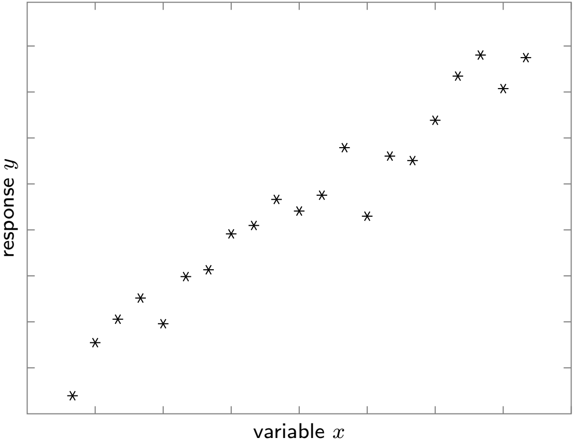
$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i;\theta), y_i) = \tfrac{1}{2} \sum_{i=1}^{N} (w^T x_i + b - y_i)^2$$

  (note: optimization over model *parameters* θ)

- Problem is convex in θ since $L(\cdot, y)$ convex and model affine
- Once trained, predict response of new input x as $\hat{y} = w^T x + b$

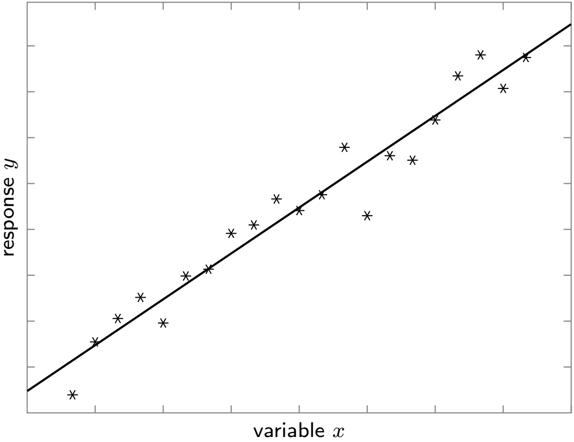
Example – Least squares

- Find affine function parameters that fit data:



Example – Least squares

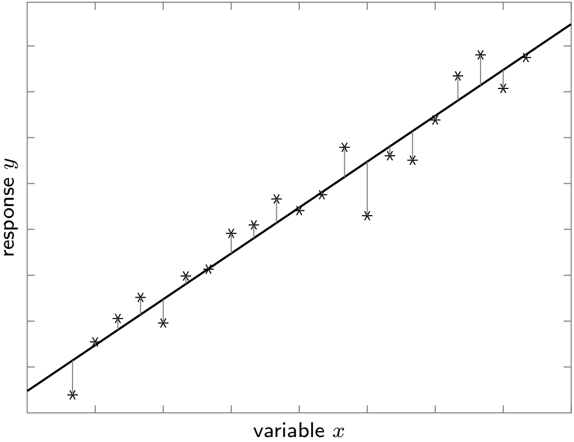
- Find affine function parameters that fit data:



- Data points (x, y) marked with ($*$), LS model $wx + b$ (——)

Example – Least squares

- Find affine function parameters that fit data:



- Data points (x, y) marked with ($*$), LS model $wx + b$ (——)
- Least squares finds affine function that minimizes squared distance

Binary classification

- Labels $y = 0$ or $y = 1$ (alternatively $y = -1$ or $y = 1$)
- Training problem

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i;\theta), y_i)$$

- Design loss L to train model parameters θ such that:
  - $m(x_i;\theta) < 0$ for pairs (x_i, y_i) where $y_i = 0$
  - $m(x_i;\theta) > 0$ for pairs (x_i, y_i) where $y_i = 1$
- Predict class belonging for new data points x with trained θ^*:
  - $m(x;\theta^*) < 0$ predict class $y = 0$
  - $m(x;\theta^*) > 0$ predict class $y = 1$

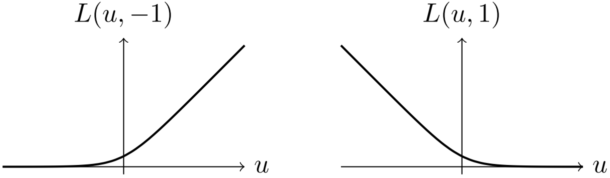
  objective is that this prediction is accurate on unseen data

Logistic regression

- Logistic regression uses:
  - affine parameterized model $m(x;\theta) = w^T x + b$ (where $\theta = (w, b)$)
  - loss function $L(u, y) = \log(1 + e^u) - yu$ (if labels $y = 0$, $y = 1$)
- Training problem, find model parameters by solving:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i;\theta), y_i) = \sum_{i=1}^{N} \Big(\log(1 + e^{x_i^T w + b}) - y_i(x_i^T w + b) \Big)$$

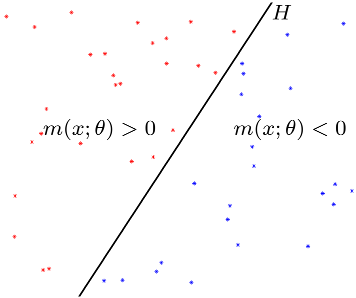
- Training problem convex in $\theta = (w, b)$ since:
  - model $m(x;\theta)$ is affine in θ
  - loss function $L(u, y)$ is convex in u



Prediction

- Use trained model m to predict label y for unseen data point x
- Since affine model $m(x;\theta) = w^T x + b$, prediction for x becomes:
  - If $w^T x + b < 0$, predict corresponding label $y = 0$
  - If $w^T x + b > 0$, predict corresponding label $y = 1$
  - If $w^T x + b = 0$, predict either $y = 0$ or $y = 1$
- A hyperplane (decision boundary) separates class predictions:

$$H := \{x : w^T x + b = 0\}$$



Multiclass logistic regression

- K classes in $\{1, \ldots, K\}$ and data/labels $(x, y) \in \mathcal{X} \times \mathcal{Y}$
- Labels: $y \in \mathcal{Y} = \{e_1, \ldots, e_K\}$ where $\{e_j\}$ coordinate basis
  - Example, $K = 5$ class 2: $y = e_2 = [0, 1, 0, 0, 0]^T$
- Use one model per class $m_j(x; \theta_j)$ for $j \in \{1, \ldots, K\}$
- Objective: Find $\theta = (\theta_1, \ldots, \theta_K)$ such that for all models j:
  - $m_j(x; \theta_j) \gg 0$, if label $y = e_j$ and $m_j(x; \theta_j) \ll 0$ if $y \neq e_j$
- Training problem loss function:

$$L(u, y) = \log\left(\sum_{j=1}^{K} e^{u_j}\right) - u^T y$$

where label y is a "one-hot" basis vector, is convex in u

Multiclass logistic regression – Training problem

- Affine data model $m(x;\theta) = w^T x + b$ with

$$w = [w_1, \ldots, w_K] \in \mathbb{R}^{n\times K}, \qquad b = [b_1, \ldots, b_K]^T \in \mathbb{R}^K$$

- One data model per class

$$m(x;\theta) = \begin{bmatrix} m_1(x;\theta_1) \\ \vdots \\ m_K(x;\theta_K) \end{bmatrix} = \begin{bmatrix} w_1^T x + b_1 \\ \vdots \\ w_K^T x + b_K \end{bmatrix}$$

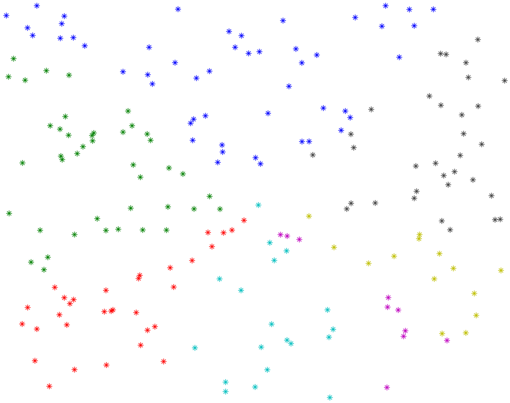
- Training problem:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} \log\left(\sum_{j=1}^{K} e^{w_j^T x_i + b_j}\right) - y^T(w^T x_i + b)$$

  where y is "one-hot" encoding of label
- Problem is convex since affine model is used

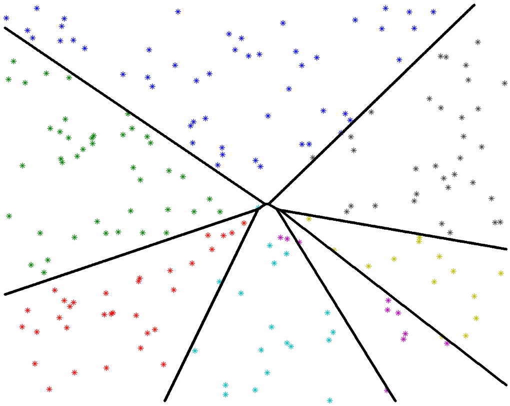
Example – Linearly separable data

- Problem with 7 classes



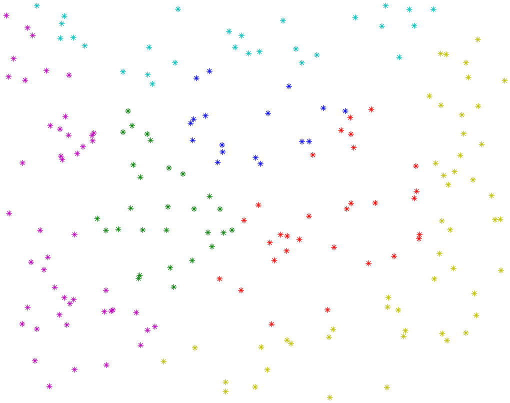
Example – Linearly separable data

- Problem with 7 classes and affine multiclass model



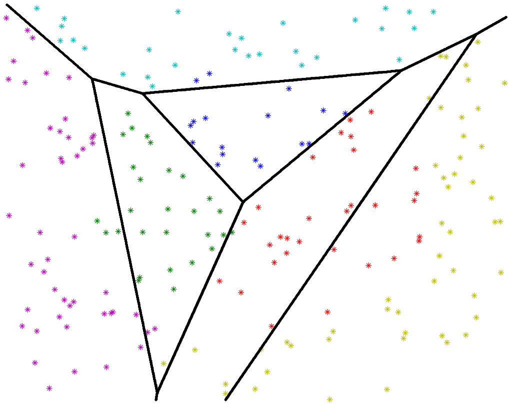
Example – Quadratically separable data

- Same data, new labels in 6 classes



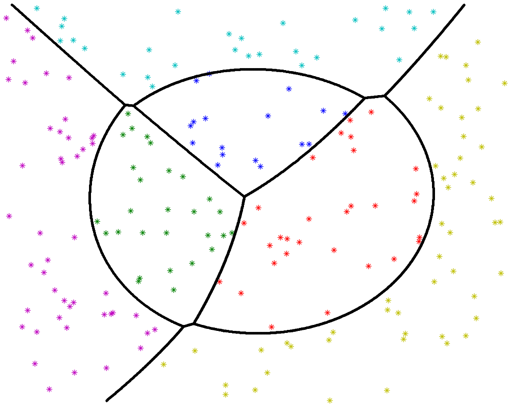
Example – Quadratically separable data

- Same data, new labels in 6 classes, affine model



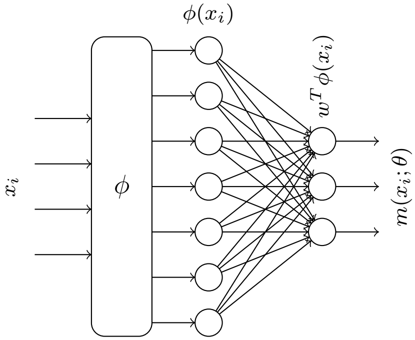
Example – Quadratically separable data

- Same data, new labels in 6 classes, quadratic model



Features

- Used quadratic features in last example
- Same procedure as before:
  - replace data vector x_i with feature vector $\phi(x_i)$
  - run classification method with feature vectors as inputs
- Model still affine in parameters, training problem still convex



Deep learning

- Can be used both for classification and regression
- Deep learning training problem is of the form

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i;\theta), y_i)$$

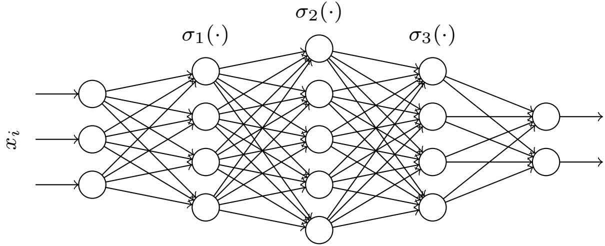
  where typically
  - $L(u, y) = \frac{1}{2}\|u - y\|_2^2$ is used for regression
  - $L(u, y) = \log\left(\sum_{j=1}^{K} e^{u_j}\right) - y^T u$ is used for K-class classification
- Difference to previous convex methods: *Nonlinear model* $m(x;\theta)$
  - Deep learning regression generalizes least squares
  - DL classification generalizes multiclass logistic regression
  - Nonlinear model makes training problem nonconvex

Deep learning – Model

- Nonlinear model of the following form is often used:

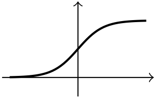
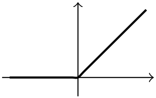
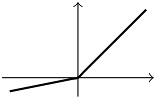
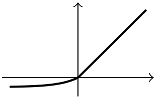
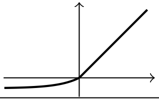
$$m(x;\theta) := W_n\sigma_{n-1}(W_{n-1}\sigma_{n-2}(\cdots(W_2\sigma_1(W_1x+b_1)+b_2)\cdots)+b_{n-1})+b_n,$$

- The σ_j are nonlinear and called activation functions
- Composition of nonlinear (σ_j) and affine ($W_j(\cdot)+b_j$) operations
- Each σ_j function constitutes a hidden layer in the model network
- Graphical representation with three hidden layers



- Why this structure?
  - (Assumed) universal function approximators
  - Efficient gradient computation using backpropagation (chain rule)

Examples of activation functions

Name	$\sigma(u)$	Graph
Sigmoid	$\frac{1}{1+e^{-u}}$	
ReLU	$\max(u, 0)$	
LeakyReLU	$\max(u, \alpha u)$	
ELU	$\begin{cases} u & \text{if } u \geq 0 \\ \alpha(e^u - 1) & \text{else} \end{cases}$	
SELU	$\lambda \begin{cases} u & \text{if } u \geq 0 \\ \alpha(e^u - 1) & \text{else} \end{cases}$	

Learning features

- Used *prespecified* feature maps (or Kernels) in convex methods
- Deep learning instead *learns* feature map during training
  - Define parameter (weight) dependent feature vector:

    $$\phi(x;\theta) := \sigma_{n-1}(W_{n-1}\sigma_{n-2}(\cdots(W_2\sigma_1(W_1x+b_1)+b_2)\cdots)+b_{n-1})$$

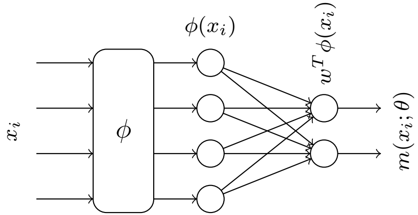
  - Model becomes $m(x;\theta) = W_n\phi(x;\theta) + b_n$
  - Inserted into training problem:

    $$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(W_n\phi(x_i;\theta) + b_n, y_i)$$

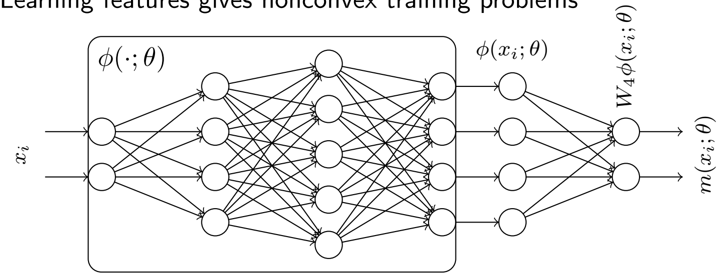
    same as before, but with learned (parameter-dependent) features
- Learning features at training makes training nonconvex

Learning features – Graphical representation

- Fixed features gives convex training problems



- Learning features gives nonconvex training problems



- Output of last activation function is feature vector

Deep learning training problem

- Training problem:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i; \theta), y_i)$$

  where typically
  - $L(u, y) = \frac{1}{2}\|u - y\|_2^2$ is used for regression
  - $L(u, y) = \log\left(\sum_{j=1}^{K} e^{u_j}\right) - y^T u$ is used for K-class classification
- Model $m(x; \theta)$ is nonlinear
  - Training problem becomes nonconvex
  - If activation functions are smooth, training problem is smooth

Proving convergence

Deterministic and stochastic algorithms

- We have deterministic algorithms

$$x_{k+1} = \mathcal{A}_k x_k$$

  that given initial x_0 will give the same sequence $(x_k)_{k\in\mathbb{N}}$
- We will also see stochastic algorithms that iterate

$$x_{k+1} = \mathcal{A}_k(\xi_k) x_k$$

  where ξ_k is a random variable that also decides the mapping
  - $(x_k)_{k\in\mathbb{N}}$ is a stochastic process of random variables
  - when running the algorithm, we evaluate ξ_k and get a realization
  - different realization $(x_k)_{k\in\mathbb{N}}$ every time even if started at same x_0
- Stochastic algorithms useful although problem is deterministic

Types of convergence

- Let $x^\star$ be solution to composite problem and $p^\star = f(x^\star) + g(x^\star)$
- We will see convergence of different quantities in different settings
- For deterministic algorithms that generate $(x_k)_{k\in\mathbb{N}}$, we will see
    - Sequence convergence: $x_k \to x^\star$
    - Function value convergence: $f(x_k) + g(x_k) \to p^\star$
    - If $g = 0$, gradient norm convergence: $\|\nabla f(x_k)\|_2 \to 0$
- Convergence is stronger as we go up the list
- First two common in convex setting, last in nonconvex

Convergence for stochastic algorithms

- Stochastic algorithms described by stochastic process $(x_k)_{k\in\mathbb{N}}$
- When algorithm is run, we get realization of stochastic process
- We analyze stochastic process and will see, e.g.,:
  - Expected sequence convergence: $\mathbb{E}[\|x_k - x^\star\|_2] \to 0$
  - Expected function value convergence: $\mathbb{E}[f(x_k) + g(x_k) - p^\star] \to 0$
  - If $g = 0$, expected gradient norm convergence: $\mathbb{E}[\|\nabla f(x_k)\|_2] \to 0$
- Says what happens with expected value of different quantities

What happens with algorithm realizations?

- We will conclude that expected value of some quantity, e.g.,:

$$\mathbb{E}[\|x_k - x^\star\|_2] \quad \text{or} \quad \mathbb{E}[f(x_k) + g(x_k) - p^\star] \quad \text{or} \quad \mathbb{E}[\|\nabla f(x_k)\|_2]$$

  converges to 0, where all quantities are nonnegative
- What happens with the actual algorithm realizations?
- We can make conclusions by the following result: If
  - $(Z_k)_{k\in\mathbb{N}}$ is a stochastic process with $Z_k \geq 0$
  - the expected value $\mathbb{E}[Z_k]$ converges to 0 as $k \to \infty$

  then realizations converge to 0 almost surely (with probability 1)
- That expected value of nonnegative quantity goes to 0 is strong

Convergence rates

- We have only talked about convergence, not convergence *rate*
- Rates indicate how fast (in iterations) algorithm reaches solution
- Typically divided into:
  - Sublinear rates
  - Linear rates (also called geometric rates)
  - Quadratic rates (or more generally superlinear rates)
- Sublinear rates slowest, quadratic rates fastest
- Linear rates further divided into Q-linear and R-linear
- Quadratic rates further divided into Q-quadratic and R-quadratic

Linear rates

- A Q-linear rate with factor $\rho \in [0, 1)$ can be:

$$f(x_{k+1}) + g(x_{k+1}) - p^\star \leq \rho(f(x_k) + g(x_k) - p^\star)$$
$$\mathbb{E}[\|x_{k+1} - x^\star\|_2] \leq \rho\mathbb{E}[\|x_k - x^\star\|_2]$$

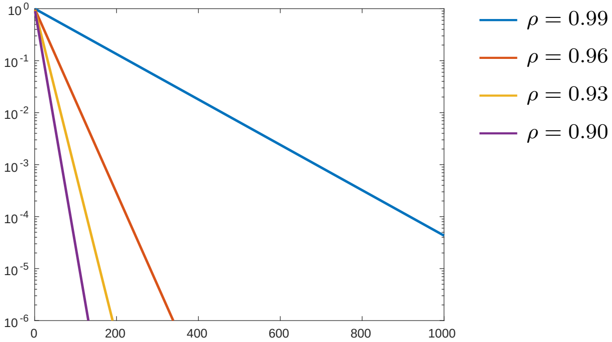
- An R-linear rate with factor $\rho \in [0, 1)$ and some $C > 0$ can be:

$$\|x_k - x^\star\|_2 \leq \rho^k C$$

  this is implied by Q-linear rate and has *exponential decrease*
- Linear rate is superlinear if $\rho = \rho_k$ and $\rho_k \to 0$ as $k \to \infty$
- Examples:
  - (Accelerated) proximal gradient with strongly convex cost
  - Randomized coordinate descent with strongly convex cost
  - BFGS has *local* superlinear with strongly convex cost
  - but SGD with strongly convex cost gives sublinear rate

Linear rates – Comparison

- Different rates in log-lin plot



- Called linear rate since linear in log-lin plot

Quadratic rates

- Q-quadratic rate with factor $\rho \in [0,1)$ can be:

$$f(x_{k+1}) + g(x_{k+1}) - p^\star \leq \rho(f(x_k) + g(x_k) - p^\star)^2$$
$$\|x_{k+1} - x^\star\|_2 \leq \rho\|x - x^\star\|_2^2$$

- R-quadratic rate with factor $\rho \in [0,1)$ and some $C > 0$ can be:

$$\|x_k - x^\star\|_2 \leq \rho^{2k} C$$

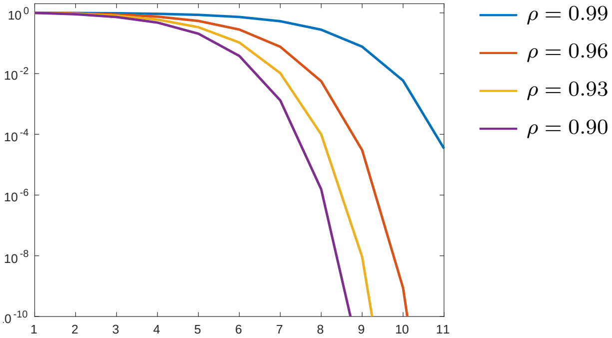
- Quadratic (ρ^{2k}) vs linear (ρ^k) rate with factor $\rho = 0.9$:

Quadratic	Linear
1.000000000000	1.000000000000
0.900000000000	0.900000000000
0.729000000000	0.810000000000
0.478296799000	0.729000000000
0.205891068000	0.656099945000
0.038152029400	0.590490005000
0.001310019380	0.531440964000
0.000001544535	0.478296936000
0.000000000002	0.430467270000

- Example: *Locally* for Newton's method with strongly convex cost

Quadratic rates – Comparison

- Different rates in log-lin scale



- Quadratic convergence is superlinear

Sublinear rates

- A rate is sublinear if it is slower than linear
- A sublinear rate can, for instance, be of the form

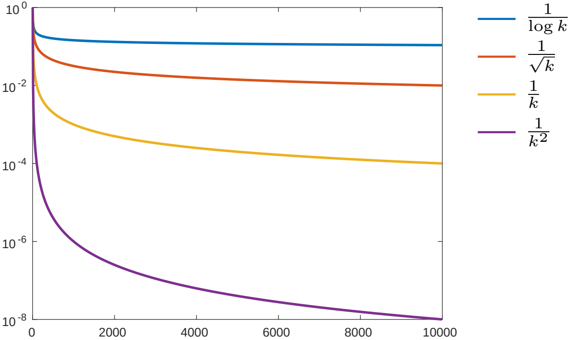
$$\begin{aligned} f(x_k) + g(x_k) - p^\star &\leq \tfrac{C}{\psi(k)} \\ \|x_{k+1} - x_k\|_2^2 &\leq \tfrac{C}{\psi(k)} \\ \min_{l=0,\ldots,k} \mathbb{E}[\|\nabla f(x_l)\|_2^2] &\leq \tfrac{C}{\psi(k)} \end{aligned}$$

  where $C > 0$ and ψ decides how fast it decreases, e.g.,
  - $\psi(k) = \log k$: Stochastic gradient descent $\gamma_k = c/k$
  - $\psi(k) = \sqrt{k}$: Stochastic gradient descent: optimal γ_k
  - $\psi(k) = k$: Proximal gradient, coordinate proximal gradient
  - $\psi(k) = k^2$: Accelerated proximal gradient method

  with improved rate further down the list
- We say that the rate is $O(\frac{1}{\psi(k)})$ for the different ψ
- To be sublinear ψ has slower than exponential growth as

Sublinear rates – Comparison

- Different rates on log-lin scale



- Many iterations may be needed for high accuracy

Proving convergence rates

- To prove a convergence rate typically requires
  - Using inequalities that describe problem class
  - Using algorithm definition equalities (or inclusions)
  - Combine these to a form so that convergence can be concluded
- Linear and quadratic rates proofs conceptually straightforward
- Sublinear rates implicit via a *Lyapunov inequality*

Proving linear or quadratic rates

- If we suspect linear or quadratic convergence for $V_k \geq 0$:

$$V_{k+1} \leq \rho V_k^p$$

where $\rho \in [0, 1)$ and $p = 1$ or $p = 2$ and V_k can, e.g., be

$$V_k = \|x_k - x^\star\|_2 \quad \text{or} \quad V_k = f(x_k) + g(x_k) - p^\star \quad \text{or} \quad V_k = \|\nabla f(x_k)\|_2$$

- Can prove by starting with V_{k+1} (or V_{k+1}^2) and continue using
  - function class inequalities
  - algorithm equalities
  - propeties of norms
  - $\ldots$

Sublinear convergence – Lyapunov inequality

- Assume we want to show sublinear convergence of some $R_k \geq 0$
- This typically requires finding a *Lyapunov inequality*:

$$V_{k+1} \leq V_k + W_k - R_k$$

  where

  - $(V_k)_{k\in\mathbb{N}}$, $(W_k)_{k\in\mathbb{N}}$, and $(R_k)_{k\in\mathbb{N}}$ are nonnegative real numbers
  - $(W_k)_{k\in\mathbb{N}}$ is summable, i.e., $\overline{W} := \sum_{k=1}^{\infty} W_k < \infty$
- Such a Lyapunov inequality can be found by using
  - function class inequalities
  - algorithm equalities
  - propeties of norms
  - $\ldots$

Lyapunov inequality consequences

- From the Lyapunov inequality:

$$V_{k+1} \leq V_k + W_k - R_k$$

  we can conclude that
  - V_k is nonincreasing if all $W_k = 0$
  - V_k converges as $k \to \infty$ (will not prove)
- Recursively applying the inequality for $l \in \{k, \ldots, 0\}$ gives

$$V_{k+1} \leq V_0 + \sum_{l=0}^{k} W_l - \sum_{l=0}^{k} R_l \leq V_0 + \overline{W} - \sum_{l=0}^{k} R_l$$

  where $\overline{W}$ is infinite sum of W_k, this implies

$$\sum_{l=0}^{k} R_l \leq V_0 - V_{k+1} + \sum_{l=0}^{k} W_l \leq V_0 + \sum_{l=0}^{k} W_l \leq V_0 + \overline{W}$$

  from which we can
  - conclude that $R_k \to 0$ as $k \to \infty$ since $R_k \geq 0$
  - derive sublinear rates of convergence for R_k towards 0

Concluding sublinear convergence

- Lyapunov inequality consequence restated

$$\sum_{l=0}^{k} R_l \leq V_0 + \sum_{l=0}^{k} W_l \leq V_0 + \overline{W}$$

- We can derive sublinear convergence for
  - Best R_k: $(k+1)\min_{l\in\{0,\ldots,k\}} R_l \leq \sum_{l=0}^{k} R_l$
  - Last R_k (if R_k decreasing): $(k+1)R_k \leq \sum_{l=0}^{k} R_l$
  - Average R_k: $\bar{R}_k = \frac{1}{k+1}\sum_{l=0}^{k} R_l$
- Let $\hat{R}_k$ be any of these quantities, and we have

$$\hat{R}_k \leq \frac{\sum_{l=0}^{k} R_l}{k+1} \leq \frac{V_0 + \overline{W}}{k+1}$$

which shows a $O(1/k)$ sublinear convergence

Deriving other than $O(1/k)$ convergence (1/3)

- Other rates can be derived from a modified Lyapunov inequality:

$$V_{k+1} \leq V_k + W_k - \lambda_k R_k$$

with $\lambda_k > 0$ when we are interested in convergence of R_k, then

$$\sum_{l=0}^{k} \lambda_l R_l \leq V_0 + \sum_{l=0}^{k} W_l \leq V_0 + \overline{W}$$

- To have $R_k \to 0$ as $k \to \infty$ we need $\sum_{l=0}^{\infty} \lambda_l = \infty$

Deriving other than $O(1/k)$ convergence (2/3)

- Restating the consequence: $\sum_{l=0}^{k} \lambda_l R_l \leq V_0 + \overline{W}$
- We can derive sublinear convergence for
  - Best R_k: $\min_{l \in \{0,\ldots,k\}} R_l \sum_{l=0}^{k} \lambda_l \leq \sum_{l=0}^{k} \lambda_l R_l$
  - Last R_k (if R_k decreasing): $R_k \sum_{l=0}^{k} \lambda_l \leq \sum_{l=0}^{k} \lambda_l R_l$
  - Weighted average R_k: $\bar{R}_k = \frac{1}{\sum_{l=0}^{k} \lambda_l} \sum_{l=0}^{k} \lambda_l R_l$
- Let $\hat{R}_k$ be any of these quantities, and we have

$$\hat{R}_k \leq \frac{\sum_{l=0}^{k} R_l}{\sum_{l=0}^{k} \lambda_l} \leq \frac{V_0 + \overline{W}}{\sum_{l=0}^{k} \lambda_l}$$

Deriving other than $O(1/k)$ convergence (3/3)

- How to get a rate out of:

$$\hat{R}_k \le \frac{V_0 + \overline{W}}{\sum_{l=0}^{k} \lambda_l}$$

- Assume $\psi(k) \le \sum_{l=0}^{k} \lambda_l$, then $\psi(k)$ decides rate:

$$\hat{R}_k \le \frac{\sum_{l=0}^{k} R_l}{\sum_{l=0}^{k} \lambda_l} \le \frac{V_0 + \overline{W}}{\psi(k)}$$

which gives a $O(\frac{1}{\psi(k)})$ rate

  - If $\lambda_k = c$ is constant: $\psi(k) = c(k+1)$ and we have $O(1/k)$ rate
  - If λ_k is decreasing: slower rate than $O(1/k)$
  - If λ_k is increasing: faster rate than $O(1/k)$

Estimating ψ via integrals

- Assume that $\lambda_k = \phi(k)$, then $\psi(k) \le \sum_{l=0}^{k} \phi(l)$ and

$$\hat{R}_k \le \frac{\sum_{l=0}^{k} R_l}{\sum_{l=0}^{k} \phi(l)} \le \frac{V_0 + \overline{W}}{\psi(k)}$$

- To estimate ψ, we use the integral inequalities
  - for decreasing nonnegative ϕ:

$$\int_{t=0}^{k} \phi(t)dt + \phi(k) \le \sum_{l=0}^{k} \phi(l) \le \int_{t=0}^{k} \phi(t)dt + \phi(0)$$

  - for increasing nonnegative ϕ:

$$\int_{t=0}^{k} \phi(t)dt + \phi(0) \le \sum_{l=0}^{k} \phi(l) \le \int_{t=0}^{k} \phi(t)dt + \phi(k)$$

- Remove $\phi(k), \phi(0) \ge 0$ from the lower bounds and use estimate:

$$\psi(k) = \int_{t=0}^{k} \phi(t)dt \le \sum_{l=0}^{k} \phi(l)$$

Sublinear rate examples

- For Lyapunov inequality $V_{k+1} \leq V_k + W_k - \lambda_k R_k$, we get:

$$\hat{R}_k \leq \frac{V_0 + \overline{W}}{\psi(k)} \qquad \text{where} \qquad \lambda_k = \phi(k) \text{ and } \psi(k) = \int_{t=0}^{k} \phi(t)dt$$

- Let us quantify the rate ψ in a few examples:
  - Two examples that are slower than $O(1/k)$:
    - $\lambda_k = \phi(k) = c/(k+1)$ gives slow $O(\frac{1}{\log k})$ rate:

$$\psi(k) = \int_{t=0}^{k} \frac{c}{t+1} dt = c[\log(t+1)]_{t=0}^{k} = c\log(k+1)$$

    - $\lambda_k = \phi(k) = c/(k+1)^\alpha$ for $\alpha \in (0,1)$, gives faster $O(\frac{1}{k^{1-\alpha}})$ rate:

$$\psi(k) = \int_{t=0}^{k} \frac{c}{(t+1)^\alpha} dt = c[\tfrac{(t+1)^{1-\alpha}}{(1-\alpha)}]_{t=0}^{k} = \tfrac{c}{1-\alpha}((k+1)^{1-\alpha} - 1)$$

  - An example that is faster than $O(1/k)$
    - $\lambda_k = \phi(k) = c(k+1)$ gives $O(\frac{1}{k^2})$ rate:

$$\psi(k) = \int_{t=0}^{k} c(t+1)dt = c[\tfrac{1}{2}(t+1)^2]_{t=0}^{k} = \tfrac{c}{2}((k+1)^2 - 1)$$

Stochastic setting and law of total expectation

- In the stochastic setting, we analyze the stochastic process

$$x_{k+1} = \mathcal{A}_k(\xi_k)x_k$$

- We will look for inequalities of the form

$$\mathbb{E}[V_{k+1}|x_k] \leq \mathbb{E}[V_k|x_k] + \mathbb{E}[W_k|x_k] - \mathbb{E}[R_k|x_k]$$

  to see what happens in one step given x_k (but not given ξ_k)
- We use *law of total expectation* $\mathbb{E}[\mathbb{E}[X|Y]] = \mathbb{E}[X]$ to get

$$\mathbb{E}[V_{k+1}] \leq \mathbb{E}[V_k] + \mathbb{E}[W_k] - \mathbb{E}[R_k]$$

  which is a Lyapunov inequality
- We can draw rate conclusions, as we did before, now for $\mathbb{E}[R_k]$

Stochastic gradient descent

Proximal gradient method

- Proximal gradient method solves problems of the form

$$\operatorname*{minimize}_{x} f(x) + g(x)$$

where (at least in our analysis)

  - $f : \mathbb{R}^n \to \mathbb{R}$ is β-smooth (not necessarily convex)
  - $g : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is closed convex

- For large problems, gradient is expensive to compute
$\Rightarrow$ replace by unbiased stochastic approximation of gradient

Unbiased stochastic gradient approximation

- Stochastic gradient:
  - estimator $\widehat{\nabla} f(x)$ outputs $\mathbb{R}^n$-valued random variable
  - realization $\widetilde{\nabla} f(x) : \mathbb{R}^n \to \mathbb{R}^n$ outputs a realization in $\mathbb{R}^n$
- An unbiased stochastic gradient approximator $\widehat{\nabla} f$ satisfies

$$\mathbb{E}\widehat{\nabla} f(x) = \nabla f(x)$$

- If x is random variable (as in SGD) an unbiased estimator satisfies

$$\mathbb{E}[\widehat{\nabla} f(x)|x] = \nabla f(x)$$

Stochastic gradient descent (SGD)

- Consider SGD for solving $\operatorname{minimize}_x f(x)$
- The following iteration generates $(x_k)_{k\in\mathbb{N}}$ of *random* variables:

$$x_{k+1} = x_k - \gamma_k \widehat{\nabla} f(x_k)$$

  since $\widehat{\nabla} f$ outputs random $\mathbb{R}^n$-valued variables
- Stochastic gradient descent finds a *realization* of this sequence:

$$x_{k+1} = x_k - \gamma_k \widetilde{\nabla} f(x_k)$$

  where $(x_k)_{k\in\mathbb{N}}$ here is a realization which is different every time
- Sloppy in notation for when x_k is *random variable* vs *realization*
- Can be efficient if realizations $\widetilde{\nabla} f$ much cheaper than ∇f

Stochastic gradients – Finite sum problems

- Consider *finite sum problems* of the form

$$\underset{x}{\text{minimize}}\ \underbrace{\frac{1}{N}\left(\sum_{i=1}^{N} f_i(x)\right)}_{f(x)}$$

  where ($\frac{1}{N}$ is for convenience and)
  - all $f_i : \mathbb{R}^n \to \mathbb{R}$ are β_i-smooth (not necessarily convex)
  - $f : \mathbb{R}^n \to \mathbb{R}$ is β-smooth (not necessarily convex)
- Training problems of this form, where sum over training data
- Stochastic gradient: select f_i at random and take gradient step

Single function stochastic gradient

- Let I be a $\{1, \ldots, N\}$-valued random variable
- Let, as before, $\widehat{\nabla} f$ denote the stochastic gradient estimator
- Realization: let i be drawn from probability distribution of I

$$\widetilde{\nabla} f(x) = \nabla f_i(x)$$

where we will use uniform probability distribution

$$p_i = p(I = i) = \tfrac{1}{N}$$

- Stochastic gradient is unbiased:

$$\mathbb{E}[\widehat{\nabla} f(x)|x] = \sum_{i=1}^{N} p_i \nabla f_i(x) = \tfrac{1}{N} \sum_{i=1}^{N} \nabla f_i(x) = \nabla f(x)$$

Mini-batch stochastic gradient

- Let $\mathcal{B}$ be set of K-sample mini-batches to choose from:
  - Example: 2-sample mini-batches and $N = 4$:

$$\mathcal{B} = \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\}$$

  - Number of mini batches $\binom{N}{K}$, each item in $\binom{N-1}{K-1}$ batches
- Let $\mathbb{B}$ be $\mathcal{B}$-valued random variable
- Let, as before, $\widehat{\nabla} f$ denote stochastic gradient estimator
- Realization: let B be drawn from probability distribution of $\mathbb{B}$

$$\widetilde{\nabla} f(x) = \frac{1}{K} \sum_{i \in B} \nabla f_i(x)$$

where we will use uniform probability distribution

$$p_B = p(\mathbb{B} = B) = \frac{1}{|\mathcal{B}|}$$

- Stochastic gradient is unbiased:

$$\mathbb{E}\widehat{\nabla} f(x) = \frac{1}{\binom{N}{K}} \sum_{B \in \mathcal{B}} \frac{1}{K} \sum_{i \in B} \nabla f_i(x) = \frac{\binom{N-1}{K-1}}{\binom{N}{K} K} \sum_{i=1}^{N} \nabla f_i(x) = \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x) = \nabla f(x)$$

Stochastic gradient descent for finite sum problems

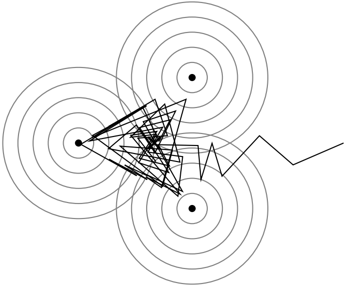
- The algorithm, choose $x_0 \in \mathbb{R}^n$ and iterate:
  1. Sample a mini-batch $B_k \in \mathcal{B}$ of indices uniformly (prob. $\frac{1}{|\mathcal{B}|}$)
  2. Run

$$x_{k+1} = x_k - \tfrac{\gamma_k}{|B_k|} \sum_{j \in B_k} \nabla f_j(x_k)$$

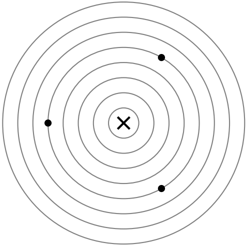
- Of course, can have $\mathcal{B} = \{1, \ldots, N\}$ and sample only one function
- Gives realization of underlying stochastic process
- How about convergence?

SGD – Example

- Let $c_1 + c_2 + c_3 = 0$
- Solve $\mathrm{minimize}_x(\frac{1}{2}(\|x - c_1\|_2^2 + \|x - c_2\|_2^2 + \|x - c_3\|_2^2) = \frac{3}{2}\|x\|_2^2 + c$
- Stochastic gradient method with $\gamma_k = 1/3$



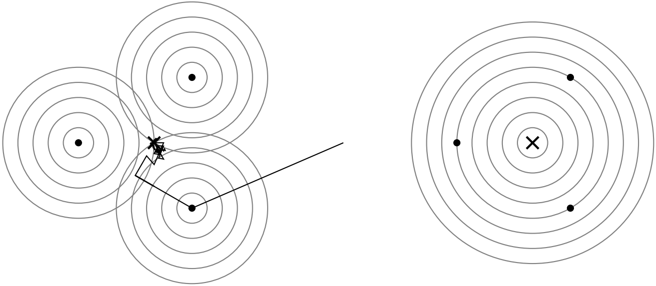
Levelsets of summands



Levelset of sum

SGD – Example

- Let $c_1 + c_2 + c_3 = 0$
- Solve $\text{minimize}_x(\frac{1}{2}(\|x - c_1\|_2^2 + \|x - c_2\|_2^2 + \|x - c_3\|_2^2) = \frac{3}{2}\|x\|_2^2 + c$
- Stochastic gradient method with $\gamma_k = 1/k$

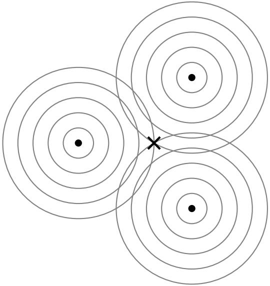


Levelsets of summands

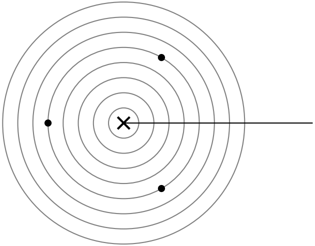
Levelset of sum

SGD – Example

- Let $c_1 + c_2 + c_3 = 0$
- Solve $\text{minimize}_x(\frac{1}{2}(\|x - c_1\|_2^2 + \|x - c_2\|_2^2 + \|x - c_3\|_2^2) = \frac{3}{2}\|x\|_2^2 + c$
- Gradient method with $\gamma_k = 1/3$



Levelsets of summands



Levelset of sum

- SGD will not converge for constant steps (unlike gradient method)

Fixed step-size SGD does not converge to solution

- We can at most hope for finding point $\bar{x}$ such that

$$0 = \nabla f(\bar{x})$$

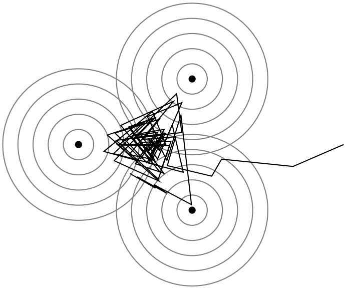
  i.e., the proximal gradient fixed-point characterization
- Assume x_k such that $0 = \nabla f(x_k)$
  - That $0 = \nabla f(x_k)$ does *not* imply $0 = \nabla f_i(x_k)$ for all f_i, hence

$$x_{k+1} = x_k - \gamma_k \nabla f_i(x_k) \neq x_k$$

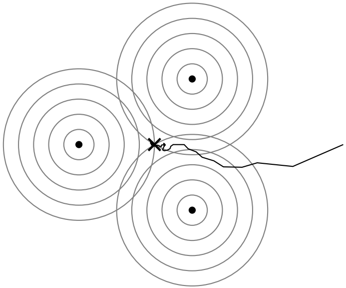
    i.e., will move away from prox-grad fixed-point for fixed $\gamma_k > 0$
  - Need diminishing step-size rule to hope for convergence

Polyak-Ruppert averaging

- Polyak-Ruppert averaging:
  - Output average of iterations instead of last iteration
- Example: SGD with constant steps and its average sequence



SGD with constant step-size



Average of SGD sequence

Nonconvex setting – Assumptions

- We consider problems of the form

$$\text{minimize}\, f(x)$$

- Assumptions:

(i) $f : \mathbb{R}^n \to \mathbb{R}$ is β-smooth, for all $x, y \in \mathbb{R}^n$:

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \tfrac{\beta}{2}\|y - x\|_2^2$$

(ii) Stochastic gradient of f is unbiased: $\mathbb{E}[\widehat{\nabla} f(x)|x] = \nabla f(x)$
(iii) Variance is bounded: $\mathbb{E}[\|\widehat{\nabla} f(x)\|_2^2|x] \leq \|\nabla f(x)\|_2^2 + M^2$
(iv) No nonsmooth term, i.e., $g = 0$
(v) A minimizer exists and $p^\star = \min_x f(x)$ is optimal value
(vi) Step-sizes satisfy $\sum_{k=1}^\infty \gamma_k = \infty$ and $\sum_{k=1}^\infty \gamma_k^2 < \infty$

- Comments:
  - (iii): variance is bounded by M^2 since

$$\begin{aligned}\mathbb{E}[\|\widehat{\nabla} f(x)\|_2^2|x] &= \mathrm{Var}[\|\widehat{\nabla} f(x)\|_2|x] + \|\mathbb{E}[\widehat{\nabla} f(x)|x]\|_2^2 \\ &= \mathrm{Var}[\|\widehat{\nabla} f(x)\|_2|x] + \|\nabla f(x)\|_2^2\end{aligned}$$

  - (iii): analysis is slightly simpler if assuming $\mathbb{E}[\|\widehat{\nabla} f(x)\|_2^2|x] \leq G$

Nonconvex setting – Analysis

- Upper bound on f in Assumption (i) gives

$$
\begin{aligned}
&\mathbb{E}[f(x_{k+1})|x_k] \\
&\qquad \leq \mathbb{E}[f(x_k) + \nabla f(x_k)^T(x_{k+1} - x_k) + \tfrac{\beta}{2}\|x_{k+1} - x_k\|_2^2 | x_k] \\
&\qquad = f(x_k) - \gamma_k \nabla f(x_k)^T \mathbb{E}[\widehat{\nabla} f(x_k)|x_k] + \tfrac{\beta\gamma_k^2}{2}\mathbb{E}[\|\widehat{\nabla} f(x_k)\|_2^2|x_k] \\
&\qquad \leq f(x_k) - \gamma_k \nabla f(x_k)^T \nabla f(x_k) + \tfrac{\beta\gamma_k^2}{2}(\|\nabla f(x_k)\|_2^2 + M^2) \\
&\qquad = f(x_k) - \gamma_k(1 - \tfrac{\beta\gamma_k}{2})\|\nabla f(x_k)\|_2^2 + \tfrac{\beta\gamma_k^2}{2}M^2
\end{aligned}
$$

- Let $\gamma_k \leq \frac{1}{\beta}$ (true for large enough k since γ_k summable):

$$\mathbb{E}[f(x_{k+1})|x_k] \leq f(x_k) - \tfrac{\gamma_k}{2}\|\nabla f(x_k)\|_2^2 + \tfrac{\beta\gamma_k^2}{2}M^2$$

- Subtracting $p^\star$ from both sides gives

$$\mathbb{E}[f(x_{k+1})|x_k] - p^\star \leq f(x_k) - p^\star - \tfrac{\gamma_k}{2}\|\nabla f(x_k)\|_2^2 + \tfrac{\beta\gamma_k^2}{2}M^2$$

Total expectation

- Taking total expectation gives Lyapunov inequality

$$\underbrace{\mathbb{E}[f(x_{k+1})] - p^\star}_{V_{k+1}} \leq \underbrace{\mathbb{E}[f(x_k)] - p^\star}_{V_k} - \underbrace{\tfrac{\gamma_k}{2}\mathbb{E}[\|\nabla f(x_k)\|_2^2]}_{R_k} + \underbrace{\tfrac{\beta\gamma_k^2}{2}M^2}_{W_k}$$

- Consequences:
  - $V_k = \mathbb{E}[f(x_k)] - p^\star$ converges (not necessarily to 0)
  - $\sum_{l=0}^k R_l \leq V_0 + \sum_{l=0}^k W_k$, which, when multiplied by 2 gives

$$\sum_{l=0}^{k} \gamma_l \mathbb{E}[\|\nabla f(x_l)\|_2^2] \leq 2(f(x_0) - p^\star) + \sum_{l=1}^{k} \gamma_l^2 \beta M^2$$

Minimum gradient bound tradeoff

- The Lyapunov inequality tells us that

$$\sum_{l=0}^{k} \gamma_l \mathbb{E}[\|\nabla f(x_l)\|_2^2] \leq 2(f(x_0) - p^\star) + \sum_{l=1}^{k} \gamma_l^2 \beta M^2$$

- Using that

$$\min_{l=0,\ldots,k} \mathbb{E}[\|\nabla f(x_l)\|_2^2] \sum_{l=0}^{k} \gamma_l \leq \sum_{l=0}^{k} \gamma_l \mathbb{E}[\|\nabla f(x_l)\|_2^2]$$

we conclude that the minimum gradient norm satisfies

$$\min_{l=0,\ldots,k} \mathbb{E}[\|\nabla f(x_l)\|_2^2] \leq \frac{2(f(x_0) - p^\star) + \sum_{l=0}^{k} \gamma_l^2 \beta M^2}{\sum_{l=0}^{k} \gamma_l}$$

where terms in the numerator:
  - $2(f(x_0) - p^\star)$ is due to initial suboptimality
  - $\sum_{l=0}^{k} \gamma_l^2 \beta M^2$ is due to noise in gradient estimates
    (if $M = 0$, use $\gamma_k = \frac{1}{\beta}$ to recover (proximal) gradient bound)

Minimum gradient convergence

- What conclusions can we draw from

$$\min_{l=0,\ldots,k} \mathbb{E}[\|\nabla f(x_l)\|_2^2] \leq \frac{2(f(x_0) - p^\star) + \sum_{l=0}^k \gamma_l^2 \beta M^2}{\sum_{l=0}^k \gamma_l}$$

- Let $C = \sum_{l=0}^\infty \gamma_l^2 < \infty$ (finite since $(\gamma_k^2)_{k\in\mathbb{N}}$ summable) then

$$\min_{l=0,\ldots,k} \mathbb{E}[\|\nabla f(x_l)\|_2^2] \leq \frac{2(f(x_0) - p^\star) + C\beta M^2}{\sum_{l=0}^k \gamma_l} \to 0$$

  as $k \to \infty$ since $(\gamma_k)_{k\in\mathbb{N}}$ is not summable
- Consequences:
  - Smallest expected value of gradient norm square converges to 0
  - We don't know what happens with latest expected value
  - Gradient converges to 0 for algorithm realizations almost surely

Convexity and strong convexity

- If we in addition assume convexity, we can show

$$R_k \leq \frac{\|x_0 - x^\star\|_2^2 + \sum_{l=0}^{k} \gamma_l^2 M^2}{2 \sum_{l=0}^{k} \gamma_l}$$

  where

$$R_k = \min_{l=0,\ldots,k} \mathbb{E}[f(x_k) - f(x^\star)] \qquad \text{or} \qquad R_k = \mathbb{E}[f(\bar{x}_k) - f(x^\star)]$$

  and $\bar{x}_k$ is an average of previous iterates
- Smallest or average function value converges to $f(x^\star)$
  - in expectation
  - for algorithm realizations with probility 1
  - no last iterate convergence bound
- Assumption: f smooth and strongly convex
  - Proximal gradient method achieves linear convergence
  - Stochastic gradient descent does not achieve linear convergence

Convergence results

- Convergence in nonconvex and convex settings are:

$$R_k \leq \frac{V_0 + D\sum_{l=0}^{k}\gamma_l^2}{b\sum_{l=0}^{k}\gamma_l}$$

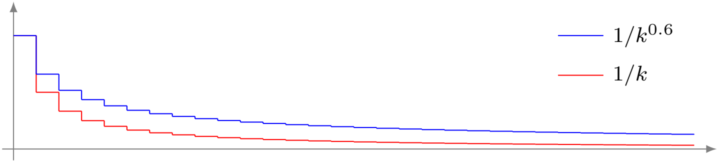
  for different V_0, D, and b and R_k
- Same dependance on step-size
- What step-sizes can we use and have convergence?

Step-size requirements

- We shift indices k and l by one to start algorithm with $k = 1$
- Step-sizes: $\sum_{l=1}^{\infty} \gamma_l^2 < \infty$ and $\sum_{l=1}^{\infty} \gamma_l = \infty$ make upper bound

$$R_k \leq \frac{V_1 + D \sum_{l=1}^{k} \gamma_l^2}{b \sum_{l=1}^{k} \gamma_l} \to 0$$

  as $k \to \infty$
- Step-size choices that satisfy assumptions:
  - $\gamma_k = c/k$ for some $c > 0$
  - $\gamma_k = c/k^\alpha$ for $\alpha \in (0.5, 1)$



Estimating rates via integrals

- For convergence need to verify $\sum_{l=1}^{\infty}\gamma_l = \infty$ and $\sum_{l=1}^{\infty}\gamma_l^2 < \infty$
- To estimate rates we need to estimate $\sum_{l=1}^{k}\gamma_l$ and $\sum_{l=1}^{k}\gamma_l^2$
- Assume $\gamma_l = \phi(l)$ with decreasing and nonnegative $\phi : \mathbb{R}_+ \to \mathbb{R}_+$
- Then we can estimate using integrals

$$\int_{t=1}^{k}\phi(t)dt + \phi(k) \leq \sum_{l=1}^{k}\phi(l) \leq \int_{t=1}^{k}\phi(t)dt + \phi(1)$$

- We can also remove $\phi(k)$ from lower bound to simplify

Estimating rates – Example $\gamma_k = c/k$

- Let $\gamma_k = \phi(k)$ with $\phi(k) = c/k$ and estimate the sum

$$\sum_{l=1}^{k} \gamma_l \geq \int_{t=1}^{k} \frac{c}{t} dt = c[\log(t)]_{t=1}^{k} = c\log(k)$$

  (which diverges as $k \to \infty$ as required) and the finite sum

$$\sum_{l=1}^{k} \gamma_l^2 \leq \int_{t=1}^{k} \frac{c^2}{t^2} dt + \phi(1)^2 = c^2[-1/t]_{t=1}^{k} + c^2 = c^2(2 - 1/k) \leq 2c^2$$

- We use these to arrive at the following rate when $\gamma_k = c/k$:

$$R_k \leq \frac{V_1 + D\sum_{l=1}^{k} \gamma_l^2}{b\sum_{l=1}^{k} \gamma_l} \leq \frac{V_1 + 2Dc^2}{bc\log k} = \frac{V_1/c + 2Dc}{b\log k}$$

  so we have $O(1/\log k)$ convergence, which is slow
- The constant c trades off the two constant terms V_1 and D

Estimating rates – Example $\gamma_k = c/k^\alpha$

- Let $\gamma_k = \phi(k)$ with $\phi(k) = c/k^\alpha$ and $\alpha \in (0.5, 1)$ and estimate

$$\sum_{l=1}^{k} \gamma_l \geq \int_{t=1}^{k} \frac{c}{t^\alpha} dt = c[\tfrac{t^{1-\alpha}}{1-\alpha}]_{t=1}^{k} = \tfrac{c}{1-\alpha}(k^{1-\alpha} - 1)$$

(which diverges as $k \to \infty$ since slower than $1/k$) and the sum

$$\sum_{l=1}^{k} \gamma_l^2 \leq \int_{t=1}^{k} \frac{c^2}{t^{2\alpha}} dt + \phi(1)^2 = c^2[\tfrac{t^{1-2\alpha}}{1-2\alpha}]_{t=1}^{k} + c^2 \leq \tfrac{c^2}{2\alpha-1} + c^2 =: c^2 C$$

where the last inequality holds since $\alpha > 0.5$

- We use these to arrive at the following rate when $\gamma_k = c/k^\alpha$:

$$R_k \leq \frac{V_1 + D\sum_{l=1}^{k} \gamma_l^2}{b\sum_{l=1}^{k} \gamma_l} \leq \frac{(1-\alpha)(V_1/c + DCc)}{b(k^{1-\alpha} - 1)}$$

so we have $O(1/k^{1-\alpha})$ rate with $\alpha \in (0.5, 1)$

- Rate improves with smaller α and $1/k^{1-\alpha} \to \sqrt{k}$ as $\alpha \to 0.5$

Refining the step-size analysis

- Have not assumed $\sum_{l=1}^{\infty} \gamma_l^2$ finite for general convergence bound

$$R_k \le \frac{V_1 + D\sum_{l=1}^{k} \gamma_l^2}{b\sum_{l=1}^{k} \gamma_l}$$

- We can divide the sum into two parts

$$R_k \le \frac{V_1}{b\sum_{l=1}^{k} \gamma_l} + \frac{D}{b\frac{\sum_{l=1}^{k} \gamma_l}{\sum_{l=1}^{k} \gamma_l^2}}$$

- So $R_k \to 0$ if $\sum_{l=1}^{k} \gamma_l \to \infty$ and $\frac{\sum_{l=1}^{k} \gamma_l}{\sum_{l=1}^{k} \gamma_l^2} \to \infty$
(don't need $\sum_{l=1}^{k} \gamma_l^2 < \infty$ for $R_k \to 0$)

Refined step-size analysis interpretation

- Let $\psi_1(k) = \sum_{l=1}^{k} \gamma_l$ and $\psi_2(k) = \frac{\sum_{l=1}^{k} \gamma_l}{\sum_{l=1}^{k} \gamma_l^2}$ and restate bound:

$$R_k \leq \frac{V_1}{b\psi_1(k)} + \frac{D}{b\psi_2(k)}$$

- ψ_1 decides how fast V_1 $(f(x_k) - p^\star$ or $\|x_k - x^\star\|_2)$ is supressed
- ψ_2 decides how fast D is supressed, where D can be
  - G^2 if assumption $\mathbb{E}[\|\widehat{\nabla} f(x)\|_2^2 | x] \leq G^2$
  - M^2 if assumption $\mathbb{E}[\|\widehat{\nabla} f(x)\|_2^2 | x] \leq \|\nabla f(x)\|_2^2 + M^2$
- There is a tradeoff between supressing these quantities
- For previous step-size choices, ψ_1 is slower
- Will present step-sizes where ψ_2 is slower
- Actual convergence very much dependent on constants V_1 and D

Estimating rates – Example $\gamma_k = c/\sqrt{k}$

- We know from before that

$$\sum_{l=1}^{k} \gamma_l = \sum_{l=1}^{k} c/k^{0.5} \geq 2c(\sqrt{k} - 1) \approx 2c\sqrt{k}$$

and that the sum of step-sizes does not converge, but satisfies

$$\sum_{l=1}^{k} \gamma_l^2 \leq \sum_{l=1}^{k} c^2/k = c^2 \log(k)$$

- Since $\sum_{l=1}^{k} \gamma_l / \sum_{l=1}^{k} \gamma_l^2$ converges, also R_k converges as

$$R_k \leq \frac{V_1}{2bc\sqrt{k}} + \frac{Dc}{b\frac{\sqrt{k}}{\log k}}$$

with rate $O(\log k/\sqrt{k})$

Estimating rates – Example $\gamma_k = c/k^\alpha$

- Let now $\alpha \in (0, 0.5)$ for which γ_k is not square summable
- We know form before that

$$\sum_{l=1}^{k} \gamma_l \geq \tfrac{c}{1-\alpha}(k^{1-\alpha} - 1)$$

  and the squared sum does not converge, but satisfies

$$\sum_{l=1}^{k} \gamma_l^2 \leq c^2[\tfrac{t^{1-2\alpha}}{1-2\alpha}]_{t=1}^{k} + c^2 = \tfrac{c^2}{1-2\alpha}(k^{1-2\alpha} - 1) + c^2 = \tfrac{c^2}{1-2\alpha}(k^{1-2\alpha} - 2\alpha)$$

- We use these to arrive at the following rate when $\gamma_k = c/k^\alpha$:

$$R_k \leq \frac{(1-\alpha)V_1}{2bc(k^{1-\alpha} - 1)} + \frac{(1-\alpha)Dc}{b(1-2\alpha)\frac{k^{1-\alpha}-1}{k^{1-2\alpha}-2\alpha}}$$

  with rate (ignoring constant terms) is worst of

$$O(1/k^{1-\alpha}) \qquad \text{and} \qquad O(1/k^{1-\alpha}/k^{1-2\alpha}) = O(1/k^\alpha)$$

  which is the latter since $\alpha \in (0, 0.5)$
- Rate improves with larger α and $k^\alpha \to \sqrt{k}$ as $\alpha \to 0.5$

How about fixed step-size

- Algorithms run in practice a finite number of iterations K
- What happens with fixed-step size scheme after K steps?
- We fix $\gamma_k = \bar{\gamma} = \theta/\sqrt{K}$ with $\theta > 0$ to be the same for all k
- Our convergence result says:

$$R_K \leq \frac{V_1 + D\sum_{l=1}^{K}\gamma_l^2}{b\sum_{l=1}^{K}\gamma_l} = \frac{V_1 + DK\bar{\gamma}^2}{bK\bar{\gamma}} = \frac{V_1 + D\theta^2}{b\sqrt{K}\theta}$$

- Comments:
  - get $\sqrt{K}$ convergence rate until iteration K
  - but R_k will not converge to 0 as $k \to \infty$
  - that $\gamma_k = \theta/\sqrt{K}$ holds for every fixed step-size for some θ
  - actual convergence very much dependent on V_1, D, and θ

Rate comparison

Setting	Gradient	Stochastic gradient $\gamma_k = 1/k^\alpha$			
		$\alpha = 1$	$\alpha \in (0,5,1)$	$\alpha = 0.5$	$\alpha \in (0, 0.5)$
Nonconvex	$O(\frac{1}{k})$	$O(\frac{1}{\log k})$	$O(\frac{1}{k^{1-\alpha}})$	$O(\frac{\log k}{\sqrt{k}})$	$O(\frac{1}{k^\alpha})$
Convex	$O(\frac{1}{k})$	$O(\frac{1}{\log k})$	$O(\frac{1}{k^{1-\alpha}})$	$O(\frac{\log k}{\sqrt{k}})$	$O(\frac{1}{k^\alpha})$
Strongly convex	linear	sublinear	sublinear	sublinear	sublinear

- Stochastic gradient descent slower in all settings
- However, every iteration in stochastic gradient descent cheaper

Finite sum comparison

- We consider

$$\text{minimize} \sum_{i=1}^{N} f_i(x)$$

  where N is large and use one f_i for each stochastic gradient
- N iterations of stochastic gradient is at cost of 1 full gradient
- Progress after k epochs (stochastic) vs k iterations (full):

Setting	**Gradient**	**Stochastic gradient** $\gamma_k = 1/k^\alpha$			
		$\alpha = 1$	$\alpha \in (0,5,1)$	$\alpha = 0.5$	$\alpha \in (0, 0.5)$
Nonconvex	$O(\frac{1}{k})$	$O(\frac{1}{\log Nk})$	$O(\frac{1}{(Nk)^{1-\alpha}})$	$O(\frac{\log Nk}{\sqrt{Nk}})$	$O(\frac{1}{(Nk)^{\alpha}})$
Convex	$O(\frac{1}{k})$	$O(\frac{1}{\log Nk})$	$O(\frac{1}{(Nk)^{1-\alpha}})$	$O(\frac{\log Nk}{\sqrt{Nk}})$	$O(\frac{1}{(Nk)^{\alpha}})$

Finite sum comparison – Quantification

- Assume that finite sum of N equals 10 million summands
- Computational budget is that we run $k = 10$ iterations/epochs
- Replacing ordo expressions with numbers:

Setting	**Gradient**	**Stochastic gradient** $\gamma_k = 1/k^\alpha$			
		$\alpha = 1$	$\alpha = 0.75$	$\alpha = 0.5$	$\alpha = 0.25$
Nonconvex	0.1	0.054	0.01	0.0018	0.01
Convex	0.1	0.054	0.01	0.0018	0.01

- Stochastic gives better ordo-rates (but constants are worse)
- Significant difference within stochastic methods, $\gamma_k = \frac{c}{\sqrt{k}}$ best
- Actual performance depends a lot on relation between constants

Thanks for your attention

- Most slides from *Optimization for Learning* at Lund University

  `https://canvas.education.lu.se/courses/7714`

- Short “flipped classroom” style videos available for many topics

  `http://www.control.lth.se/fileadmin/control/Education/EngineeringProgram/FRTN50/VideoPlatform/VideoLecturePlatform.html`