# Lecture 11

## Math Foundations Team

**BITS** Pilani
Pilani | Dubai | Goa | Hyderabad

# Introduction

- We will take a deeper look at some challenges in non-linear optimization, continuing on from the previous lecture.
- First we will look at the problem of different levels of curvature in different directions.
- We will need to figure out strategies to deal with the above situations to design a good optimization algorithm.
- Second we will look into how to solve constrained optimization problem.

# Examples of high curvature surfaces

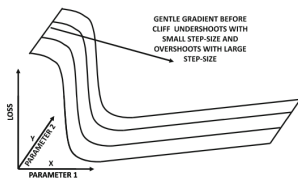- Two examples of high curvature surfaces are cliffs and valleys.



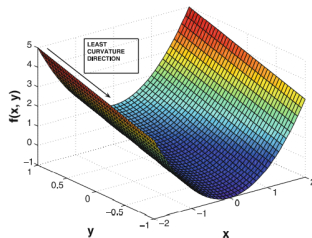Figure 5.3: An example of a cliff in the loss surface

(a)



Figure 5.4: The curvature effect in valleys

(b)

# Difficult topologies

- In Figure (a), the partial derivative with respect to $x$ changes drastically as we go along the axis of $x$.

- A modest learning rate will cause minimal reduction in the value of the objective function in the gently sloping regions.

- The same modest learning rate in the high-sloping regions will cause us to overshoot the optimal value in those regions.

- In Figure (b), there is gentle slope along the $y$-direction and a U-shape in the $x$-direction.

- The gradient descent method will bounce violently along the steep sides of the valley while not making much progress along the $x$-axis.

# Gradient descent with momentum

- Gradient descent may be slow if the curvature of the function is such that the gradient descent steps hop between the walls of the valley of contours and approaches the optimum slowly.

- If we endow the optimization procedure with memory, we can improve convergence.

- We use an additional term in the step-update to remember what happened in the previous iteration, so that we can dampen oscillations and speed up convergence.

- This is a momentum term - the name momentum comes from a comparison to a rolling ball whose direction becomes more and more difficult to change as its velocity increases.

# Gradient descent with momentum

The gradient descent with momentum has the following iteration

$$x_{i+1} = x_i - \alpha_i((\nabla f)(x_i))^T + v_i$$

- $v_i = \beta(x_i - x_{i-1})$ and $v_0 = 0$
- $\beta \in [0, 1]$ is referred to as the momentum parameter or the friction parameter.
- Momentum-based methods attack the issues of flat-regions, cliffs and valleys by emphasizing medium-term to long-term directions of consistent movement.

# Momentum-based learning

- An aggregated measure of feedback is used to reinforce movement along certain directions and speed up gradient descent.

- Momentum-based learning accelerates gradient descent since the algorithm moves quicker in the direction of the optimal solution.

- The useless sideways oscillations as they get cancelled out during the averaging process.

- The momentum term is useful where only approximate gradient is known as the momentum term averages out the noisy estimates of the gradient.

▶ The concept of momentum can be illustrated by a marble rolling down a hill that has a number of "local" distortions like potholes, ditches etc. The momentum of the marble causes it to navigate local distortions and emerge out of them.
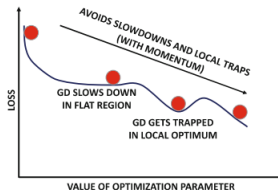


Figure 5.5: Effect of momentum in navigating complex loss surfaces. The annotation "GD" indicates pure gradient descent without momentum. Momentum helps the optimization process retain speed in flat regions of the loss surface and avoid local optima

# Momentum-based learning

Momentum increases the relative component of the gradient in the correct direction
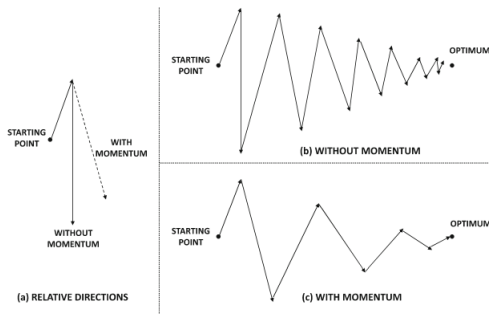


Figure 5.6: Effect of momentum in smoothing zigzag updates

# AdaGrad

- AdaGrad algorithm keep track of the aggregated squared magnitude of the partial derivative with respect to each parameter.

- At each iteration of Adagrad

$$A_i \leftarrow A_i + (\frac{\partial J}{\partial w_i})^2$$

$$w_i \leftarrow w_i - \frac{\alpha}{\sqrt{A_i}} \frac{\partial J}{\partial w_i}, \quad \forall i$$

- $A_i$ measures only the historical magnitude of the gradient rather than the sign. To avoid ill-conditioning, $\epsilon = 10^{-8}$ can be added to $A_i$.

- If the gradient takes values +100 and -100 alternatively, $A_i$ will be large and the update step along the parameter in question will be small.

- If the gradient takes a value 0.1 consistently, $A_i$ will not be as large as before and the update step will be comparatively larger.

- With the passage of time absolute movements along all components will slow down because $A_i$ is monotonically increasing with time.

- AdaGrad suffers from the problem of not making much progress after a while. The fact that $A_i$ is aggregated over the entire history of partial derivatives may make the method stale.

# RMSProp

- At each iteration of RMSProp

$$A_i \leftarrow \rho A_i + (1 - \rho)(\frac{\partial J}{\partial w_i})^2$$

$$w_i \leftarrow w_i - \frac{\alpha}{\sqrt{A_i}} \frac{\partial J}{\partial w_i}, \ \ \forall i$$

- It uses exponential averaging. scaling factor $A_i$ does not constantly increase. Note that $\rho \in (0, 1)$.
- In RMSProp the importance of ancient gradients decays exponentially with time as the gradient from $t$ steps before is weighted by $\rho^t$.

# Adam Algorithm

- At each iteration of Adam Algorithm

$$A_i \leftarrow \rho A_i + (1 - \rho)(\frac{\partial J}{\partial w_i})^2$$

$$F_i \leftarrow \rho_f F_i + (1 - \rho_f)\frac{\partial J}{\partial w_i}$$

$$w_i \leftarrow w_i - \alpha_t \frac{F_i}{\sqrt{A_i}}, \quad \forall i$$

- $\alpha_t = \alpha \frac{\sqrt{1 - \rho^t}}{1 - \rho_f^t}$,
- $\rho \in (0, 1)$ and $\rho_f$ is called decay parameter.

# Adam

- Adam incorporates momentum into the update.
- Adam addresses the initialization bias present in RMSProp. Both $F_i$ and $A_i$ are initialized to zero which causes bias in early iterations.
- The two quantities are affected differently which accounts for the equation for $\alpha_t$.
- As $t \to \infty$, $\rho^t \to 0$, $\rho_f^t \to 0$ and $\alpha_t \to \alpha$ since $\rho, \rho_f \in (0, 1)$.
- The default suggested value for $\rho_f$ and $\rho$ are 0.9 and 0.999 respectively.

# Constrained optimization and Lagrange multipliers

- Consider the following problem: $\min_{\boldsymbol{x}} f(\boldsymbol{x}), f : \mathbb{R}^D \to \mathbb{R}$, subject to additional constraints - so we are looking at a minimization problem except that the set of all $\boldsymbol{x}$ over which minimization is performed is not all of $\mathbb{R}^D$.

- The constrained problem becomes
  $\min_{\boldsymbol{x}} f(\boldsymbol{x})$
  subject to $g_i(\boldsymbol{x}) \leq 0 \ \forall i 1, 2, \ldots m$.

- If we convert constrained optimization problem into an unconstrained one, we can use techniques we studied

- We construct $J(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{i=m} \mathbf{1}(g_i(\boldsymbol{x}))$, where $\mathbf{1}(z) = 0$ for $z \leq 0$ and $\mathbf{1}(z) = \infty$ for $z > 0$.

# Constrained optimization and Lagrange multipliers

- ▶ The formulation of $J(\boldsymbol{x})$ in the previous slide ensures that the optimal solution to the unconstrained problem is the same as the constrained problem.

- ▶ The step-function is also difficult to optimize. Hence, we replace the step-function by a linear function using Lagrange multipliers.

- ▶ We create the Lagrangian of the given constrained optimization problem as follows:
  $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{i=m} \lambda_i g_i(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x})$, where $\lambda_i \geq 0$ for all $i$.

- The primal problem is min $f(\boldsymbol{x})$ subject to $g_i(\boldsymbol{x}) \leq 0, 1 \leq i \leq m$. Optimization is performed over the primal variables $\boldsymbol{x}$.

- The associated Lagrangian dual problem is $\max_{\boldsymbol{\lambda} \in \mathbb{R}^m} \mathfrak{D}(\boldsymbol{\lambda})$ subject to $\boldsymbol{\lambda} \geq 0$ where $\boldsymbol{\lambda}$ are dual variables.

- $\mathfrak{D}(\boldsymbol{\lambda}) = \min_{\boldsymbol{x} \in \mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$.

- The following minimax inequality holds over two arguments $\boldsymbol{x}, \boldsymbol{y}$: $\max_{\boldsymbol{y}} \min_{\boldsymbol{x}} \phi(\boldsymbol{x}, \boldsymbol{y}) \leq \min_{\boldsymbol{x}} \max_{\boldsymbol{y}} \phi(\boldsymbol{x}, \boldsymbol{y})$.

# Minimax inequality

- Why is this inequality true?
- Assume that $x, y$: $\max_y \min_x \phi(x, y) = \phi(x_A, y_A)$ and $\min_x \max_y \phi(x, y) = \phi(x_B, y_B)$.
- Fixing $y$ at $y_A$ we see that the inner operation on the left hand side of the minimax inequality is a min operation over $x$ and returns $x_A$. Thus we have $\phi(x_A, y_A) \leq \phi(x_B, y_A)$.
- Fixing $x$ at $x_B$ we see that the inner operation on the right hand side of the minimax inequality is a max operation over $y$ and returns $y_B$. Thus we have $\phi(x_B, y_B) \geq \phi(x_B, y_A)$.
- From the above we conclude that $\phi(x_B, y_B) \geq \phi(x_A, y_A)$.

- The difference between $J(\boldsymbol{x})$ and the Lagrangian $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is that the indicator function is relaxed to a linear function.

- When $\boldsymbol{\lambda} \geq 0$, the Lagrangian $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is a lower bound on $J(\boldsymbol{x})$.

- The maximum of $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ with respect to $\boldsymbol{\lambda}$ is $J(\boldsymbol{x})$ - if the point $\boldsymbol{x}$ satisfies all the constraints $g_i(\boldsymbol{x}) \leq 0$, then the maximum of the Lagrangian is obtained at $\boldsymbol{\lambda} = 0$ and it is equal to $J(\boldsymbol{x})$.

- If one or more constraints is violated such that $g_i(\boldsymbol{x}) > 0$, then the associated Lagrangian coefficient $\lambda_i$ can be taken to be $\infty$ so as to equal $J(\boldsymbol{x})$.

- From the previous slide, we have $J(\boldsymbol{x}) = \max_{\boldsymbol{\lambda} \geq 0} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$.

- Our original constrained optimization problem boiled down to minimizing $J(\boldsymbol{x})$, in other words we are looking at $\min_{\boldsymbol{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geq 0} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$

- Using the minimax inequality we see that $\min_{\boldsymbol{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geq 0} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \geq \max_{\boldsymbol{\lambda} \geq 0} \min_{\boldsymbol{x} \in \mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$.

- This is known as weak duality. The inner part of the right hand side of the inequality is $\mathfrak{D}(\boldsymbol{\lambda})$.

# Lagrangian formulation

- In contrast to the original formulation $\mathfrak{D}(\boldsymbol{\lambda}) = \min_{\boldsymbol{x} \in \mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is an unconstrained optimization problem for a given value of $\boldsymbol{\lambda}$.

- We observe that $\mathfrak{D}(\boldsymbol{\lambda}) = \min_{\boldsymbol{x} \in \mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is a point-wise minimum of affine functions and hence $\mathfrak{D}(\boldsymbol{\lambda})$ is concave even though $f()$ and $g()$ may be nonconvex.

- We have obtained a Lagrangian formulation for a constrained optimization problem where the constraints are inequalities. What happens when some constraints are equalities?

- Suppose the problem is $\min_{\boldsymbol{x}} f(\boldsymbol{x})$ subject to $g_i(\boldsymbol{x}) \leq 0$ for all $1 \leq i \leq m$ and $h_j(\boldsymbol{x}) = 0$ for $1 \leq j \leq n$.

- We model the equality constraint $h_j(\boldsymbol{x}) = 0$ with two inequality constraints $h_j(\boldsymbol{x}) \geq 0$ and $h_j(\boldsymbol{x}) \leq 0$.

- The resulting Lagrange multipliers are then unconstrained.

- The Lagrange multipliers for the original inequality constraints are non-negative while those corresponding to the equality constraints are unconstrained.

# Convex optimization

- We are interested in a class of optimization problems where we can guarantee global optimality.
- When $f()$, the objective function, is a convex function and $g()$ and $h()$ are convex functions, we have a convex optimization problem.
- In this setting we have strong duality - the optimal solution of the primal problem is equal to the optimal solution of the dual problem.
- What is a convex function?

# Convex function

- First we need to know what is a convex set. A set $C$ is a convex set if for any $x, y \in C$, $\theta x + (1 - \theta)y \in C$.

- For any two points lying in the convex set, a line joining them lies entirely in the convex set.

- Let a function $f : \mathbb{R}^d \to R$ be a function whose domain is a convex set $C$.

- The function is a convex function if for any $\boldsymbol{x}, \boldsymbol{y} \in C$, $f(\theta \boldsymbol{x} + (1 - \theta)\boldsymbol{y}) \leq \theta f(\boldsymbol{x}) + (1 - \theta)f(\boldsymbol{y})$

- Another way of looking at a convex function is to use the gradient: for any two points $\boldsymbol{x}$ and $\boldsymbol{y}$, we have $f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} f(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x})$.

# Example

- The negative entropy, a useful function in Machine Learning, is convex: $f(x) = x\log_2 x$ for $x > 0$.

- First let us check if $f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y)$. Take $x = 2$, $y = 4$, and $\theta = 0.5$ to get $f(0.5 * 2 + 0.5 * 4) = f(3) = 3\log_2 3 \approx 4.75$. Then $\theta f(2) + (1 - \theta)f(4) = 0.5 * 2\log_2 2 + 0.5 * 4\log_2 4 = \log_2 32 = 5$. Therefore the convexity criterion is satisfied for these two points.

- Let us now use the gradient criterion. We have $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \log_2 x + x\frac{1}{x\log_e 2}$. $f(2) + \nabla f(2) * (4 - 2) = 2\log_2 2 + (\log_2 2 + \frac{1}{\log_e 2} * 2 \approx 6.9$. We see that $f(4) = 4\log_2 4 = 8$ which shows that the gradient criterion is also satisfied.

- ▶ Let us look at a convex optimization problem where the objective function and constraints are all linear.

- ▶ Such a convex optimization problem is called a linear programming problem.

- ▶ We can express a linear programming problem as $\min_{\boldsymbol{x}} \boldsymbol{c}^T \boldsymbol{x}$ subject to $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ where $\boldsymbol{A} \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^{m \times 1}$.

- ▶ The Lagrangian $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is given by $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{c}^T \boldsymbol{x} + \boldsymbol{\lambda}^T (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$ where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is the vector of non-negative Lagrangian multipliers.

- ▶ We can rewrite the Lagrangian as $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = (\boldsymbol{c} + \boldsymbol{A}^T \boldsymbol{\lambda})^T \boldsymbol{x} - \boldsymbol{\lambda}^T \boldsymbol{b}$.

- Taking the derivative of the Lagrangian with respect to $\boldsymbol{x}$ and setting it to zero we get $\boldsymbol{c} + \boldsymbol{A}^T \boldsymbol{\lambda} = 0$.

- Since $\mathfrak{D}(\lambda) = \min_{\boldsymbol{x} \in \mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$, plugging in the above equation gives $\mathfrak{D}(\boldsymbol{\lambda}) = -\boldsymbol{\lambda}^T \boldsymbol{b}$.

- We would like to maximize $\mathfrak{D}(\boldsymbol{\lambda})$, subject to the constraint $\boldsymbol{\lambda} \geq 0$.

- Thus we end up with the following problem:

$$max_{\boldsymbol{\lambda} \in \mathbb{R}^m} - \boldsymbol{\lambda}^T \boldsymbol{b}$$
$$\text{subject to } \boldsymbol{c} + \boldsymbol{A}^T \boldsymbol{\lambda} = 0$$
$$\boldsymbol{\lambda} \geq 0$$

# Linear programming

- We can solve the original primal linear program or the dual one - the optimum in each case is the same.
- The primal linear program is in $d$ variables but the dual is in $m$ variables, where $m$ is the number of constraints in the original primal program.
- We choose to solve the primal or dual based on which of $m$ or $d$ is smaller.

▶ We now consider the case of a quadratic objective function subject to affine constraints:

$$min_{\mathbf{x} \in \mathbb{R}^d} \frac{1}{2}\mathbf{x}^T \mathbf{Q}\mathbf{x} + \mathbf{c}^T \mathbf{x} \text{ subject to}$$

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}$$

▶ Here $\mathbf{A} \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m, c \in \mathbb{R}^d$

# Quadratic programming

- The Lagrangian $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is given by
$\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$.

- Rearranging the above we have
$\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + (\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})^T\boldsymbol{x} - \boldsymbol{\lambda}^T\boldsymbol{b}$

- Taking the derivative of $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ and setting it equal to zero gives $\boldsymbol{Q}\boldsymbol{x} + (\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}) = 0$.

- If we take $\boldsymbol{Q}$ to be invertible, we have $\boldsymbol{x} = -\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})$.

- Plugging this value of $\boldsymbol{x}$ into $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ gives us
$\mathfrak{D}(\boldsymbol{\lambda}) = -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}) - \boldsymbol{\lambda}^T\boldsymbol{b}$.

- This gives us the dual optimization problem:
$\max_{\boldsymbol{\lambda} \in \mathbb{R}^m} -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}) - \boldsymbol{\lambda}^T\boldsymbol{b}$ subject to $\boldsymbol{\lambda} \geq \boldsymbol{0}$.