Optimization

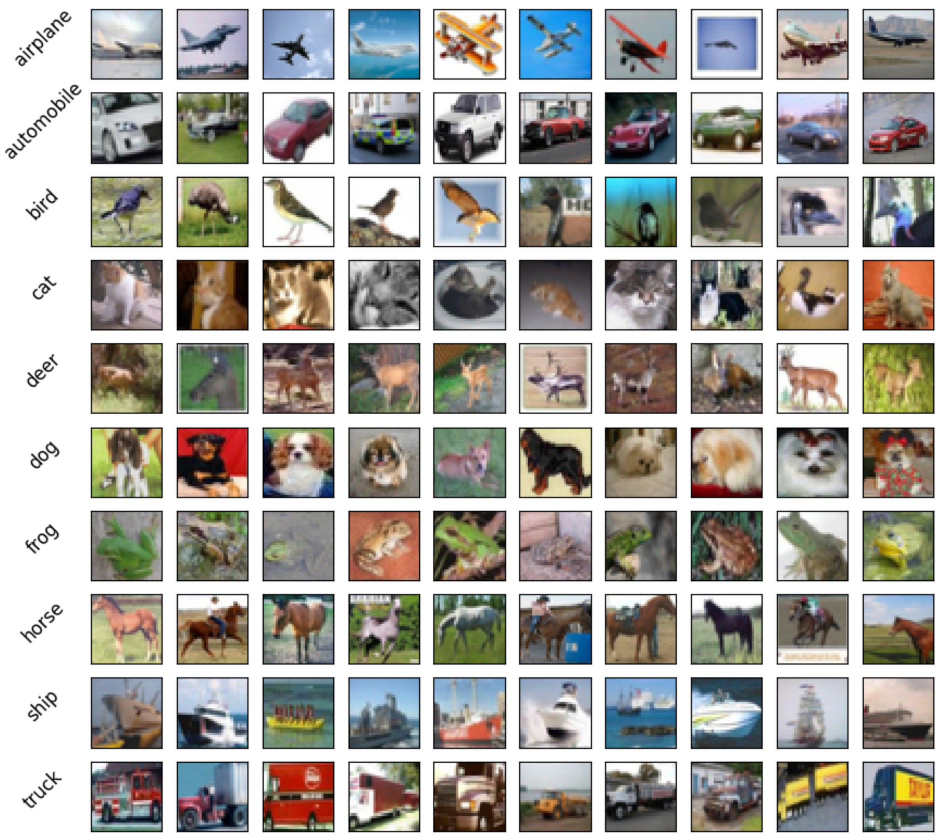
CoE197M/EE298M (Foundations of Machine Learning)

Rowel Atienza, Ph.D.

rowel@eee.upd.edu.ph

Reference: "Mathematics for Machine Learning". Copyright 2020 by Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong. Published by Cambridge University Press.

Model Optimization: Finding $\boldsymbol{\theta}$ that explains the dataset $\mathcal{D} = \{\boldsymbol{x}, \boldsymbol{y}\}$



Input : $p(\boldsymbol{x})$

ML Model

Output: $p(\boldsymbol{y}|\boldsymbol{x})$

Parameters : $p(\boldsymbol{\theta}|\boldsymbol{x}, \boldsymbol{y})$

What to optimize to find θ that explains $\mathcal{D}$?

By minimizing a metric, distance or loss function between the model prediction and ground truth labels:

$$L(\boldsymbol{\theta}) = L\left(\boldsymbol{y}^{true}, \boldsymbol{y}^{pred} \middle| \boldsymbol{\theta}\right) = d\left(\boldsymbol{y}^{true}, \boldsymbol{y}^{pred} \middle| \boldsymbol{\theta}\right)$$

Let $\boldsymbol{y}^{error} = \boldsymbol{y}^{true} - \boldsymbol{y}^{pred}$

Common Loss Functions

L1 Norm: For $\boldsymbol{y} \in \mathbb{R}^n$.

$$\|\boldsymbol{y}^{error}\|_1 = \sum_{i=1}^{n} |y_i^{error}|$$

e.g. n is the number of classes

Some cases use a factor $\frac{1}{n}$ to normalize L1

Mean Absolute Error (MAE):

$$MAE = \frac{1}{batch\_size} \sum_{b=1}^{batch\_size} \|\boldsymbol{y}^{error}\|_{1,b}$$

For one batch of samples in SGD.

Common Loss Functions

L2 Norm: For $\boldsymbol{L} \in \mathbb{R}^n$:

$$\|\boldsymbol{y}^{error}\|_2 = \sqrt{\sum_{i=1}^{n} \left(y_i^{error}\right)^2}$$

$$= \sqrt{\boldsymbol{y}^{error^T} \boldsymbol{y}^{error}}$$

e.g. n is the number of classes

Some cases use a factor $\frac{1}{n}$ to normalize L2

Mean Squared Error (MSE):

$$MSE = \frac{1}{batch\_size} \sum_{b=1}^{batch\_size} \sum_{i=1}^{n} \left(y_{i,b}^{error}\right)^2$$

For one batch of samples in SGD.

Common Loss Functions

Cross-Entropy:

$$CE = \langle \boldsymbol{y}^{true}, -\log \boldsymbol{y}^{pred} \rangle = -\int_a^b \boldsymbol{y}^{true} \log \boldsymbol{y}^{pred} \, d\boldsymbol{y}$$

Categorical Cross-Entropy:

$$CE = -\sum_{i=1}^{n} y_i^{true} \log y_i^{pred}$$

CE per batch

$$CE = \frac{1}{batch\_size} \sum_{b=1}^{batch\_size} CE_b$$

For one batch of samples in SGD.

Common Loss Functions

Binary Cross-Entropy (BCE):

$$BCE = -y_i^{true} \log y_i^{pred} - (1 - y_i^{true}) \log(1 - y_i^{pred})$$

BCE per batch

$$CE = \frac{1}{batch\_size} \sum_{b=1}^{batch\_size} BCE_b$$

For one batch of samples in SGD.

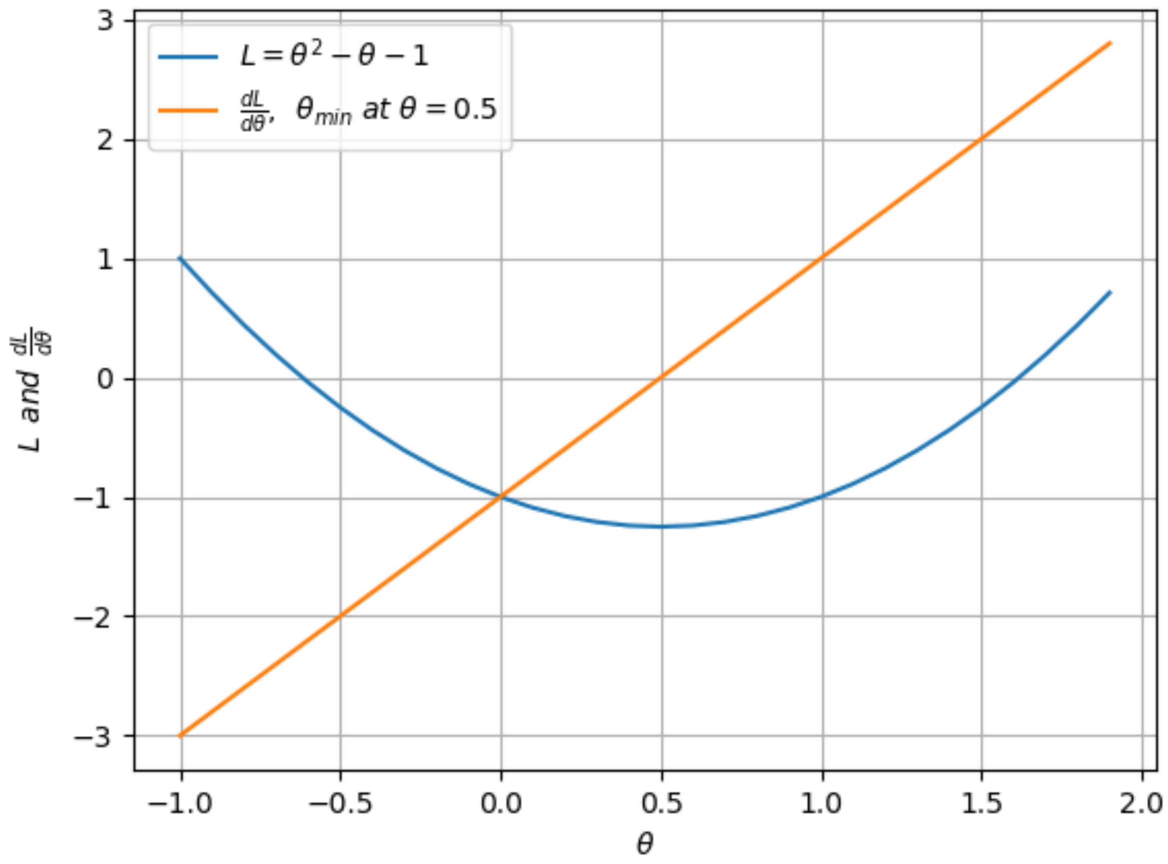
1-Min Loss Function : $L = \theta^2 - \theta - 1$

Can be solved analytically by:

$$\frac{dL}{d\theta} = 2\theta - 1 = 0$$
$$\therefore \theta = \frac{1}{2}$$

Verify as (global) minimum:

$$\left.\frac{d^2L}{d\theta^2}\right|_{\theta=\frac{1}{2}} = \theta\Big|_{\theta=\frac{1}{2}} = \frac{1}{2} > 0$$



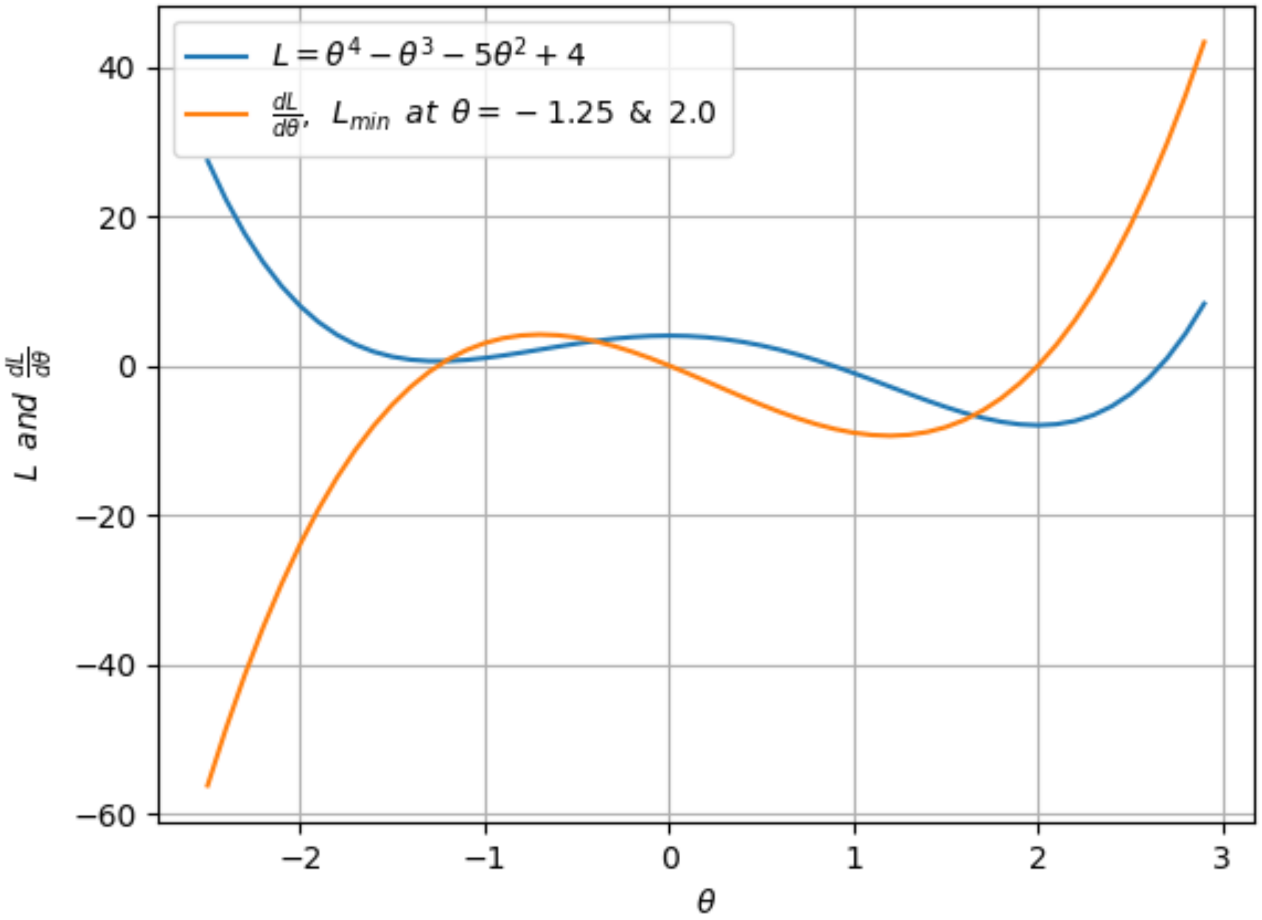
2-Mins Loss Function: $L = \theta^4 - \theta^3 - 5\theta^2 + 4$

$$\frac{dL}{d\theta} = 4\theta^3 - 3\theta^2 - 10\theta = 0$$

$$L_{min}\ at\ \theta = -1.25, 2.0$$

(Global) min at $\theta = 2.0$

$$\left.\frac{d^2L}{d\theta^2}\right|_{\theta=2} = 26 > 0$$



Issues

Many ML Models do not have simple loss functions as a function of parameters that can be solved analytically in closed form

Use a numerical solution that can be solved iteratively

Gradient Descent

Numerical Algorithm for Optimization

Gradient Descent

To find the minimum, adjust the parameters in the direction opposite the gradient (ie negative gradient) of the loss function:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \varepsilon\big(\nabla L(\boldsymbol{\theta})\big)^T$$

ε a small learning rate or step size: $\varepsilon \in (0, \infty)$ typically $\varepsilon \in [1e - 6{,}1.0]$

Gradient Descent on Loss Function : $L = \theta^2 - \theta - 1$

Gradient:

$$\frac{dL}{d\theta} = 2\theta - 1$$

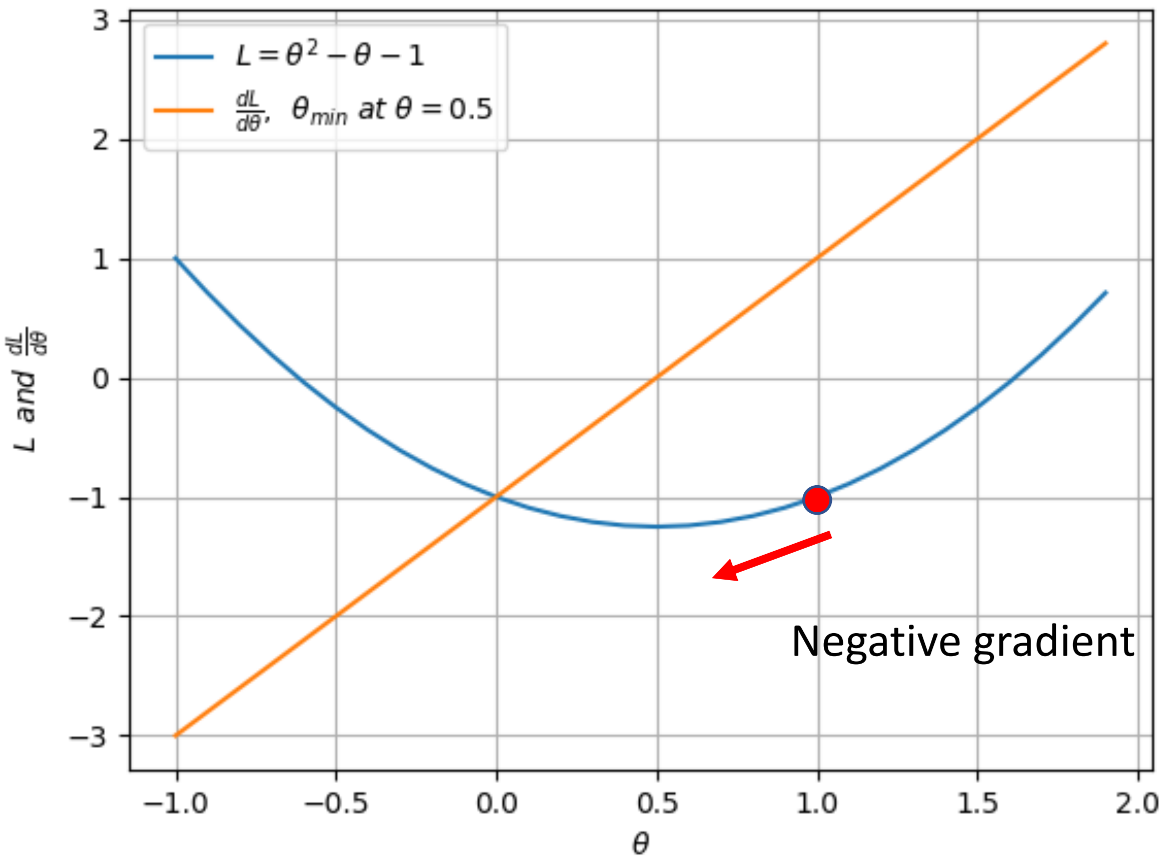
Let us say the initial state is $\theta_0 = 1$ & $\varepsilon = 0.1$

$$\frac{dL}{d\theta} = 1$$

$$\theta = 1 - 0.1(1) = 0.9$$

As we move down the bowl, we will eventually hit the minimum of L at:

$$\theta = 0.5$$



Effect of Learning Rate

Learning rate $\varepsilon = 0.01$

Gradient:

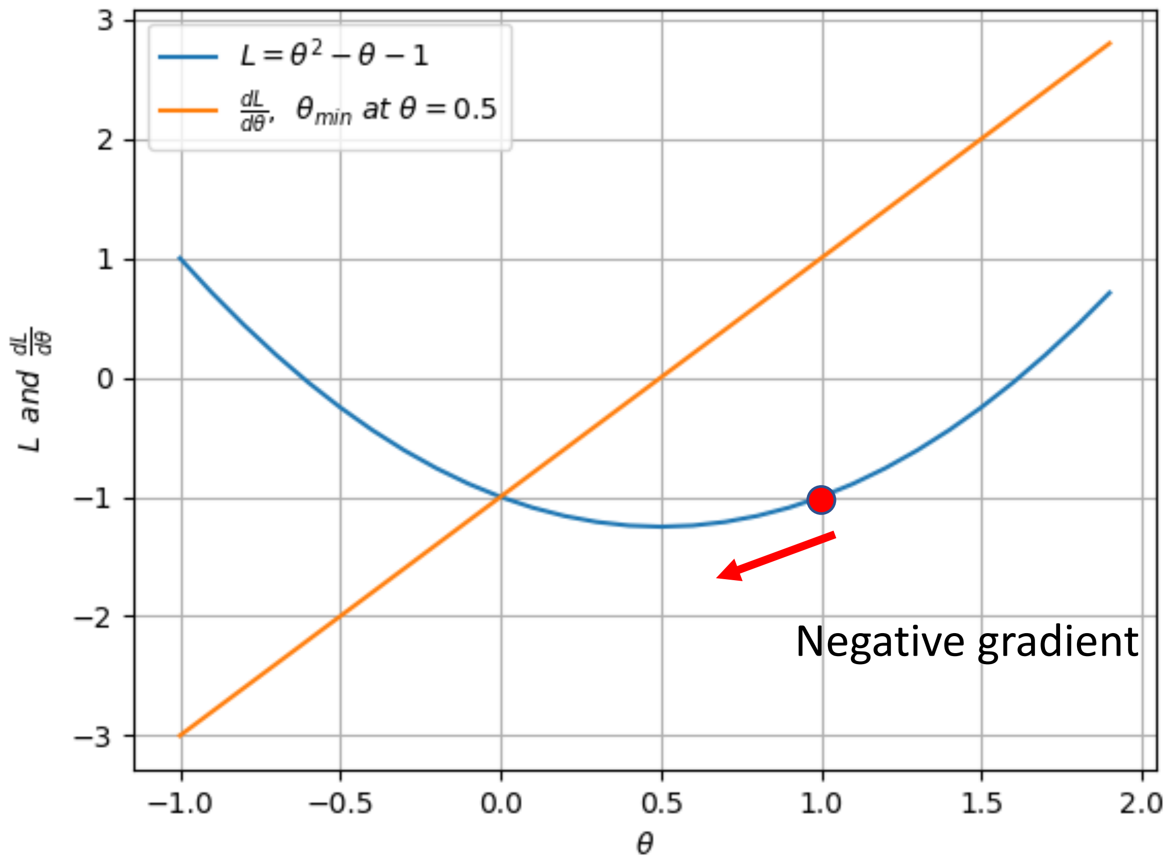
$$\frac{dL}{d\theta} = 2\theta - 1$$

Let us say $\theta_0 = 1$ & $\varepsilon = 0.01$

$$\frac{dL}{d\theta} = 1$$

$$\theta = 1 - 0.01(1) = 0.99$$

As we move down the bowl, we will eventually hit the minimum of L at: $\theta = 0.5$ but with a bigger number of steps



Learning rate $\varepsilon = 1$

Gradient:

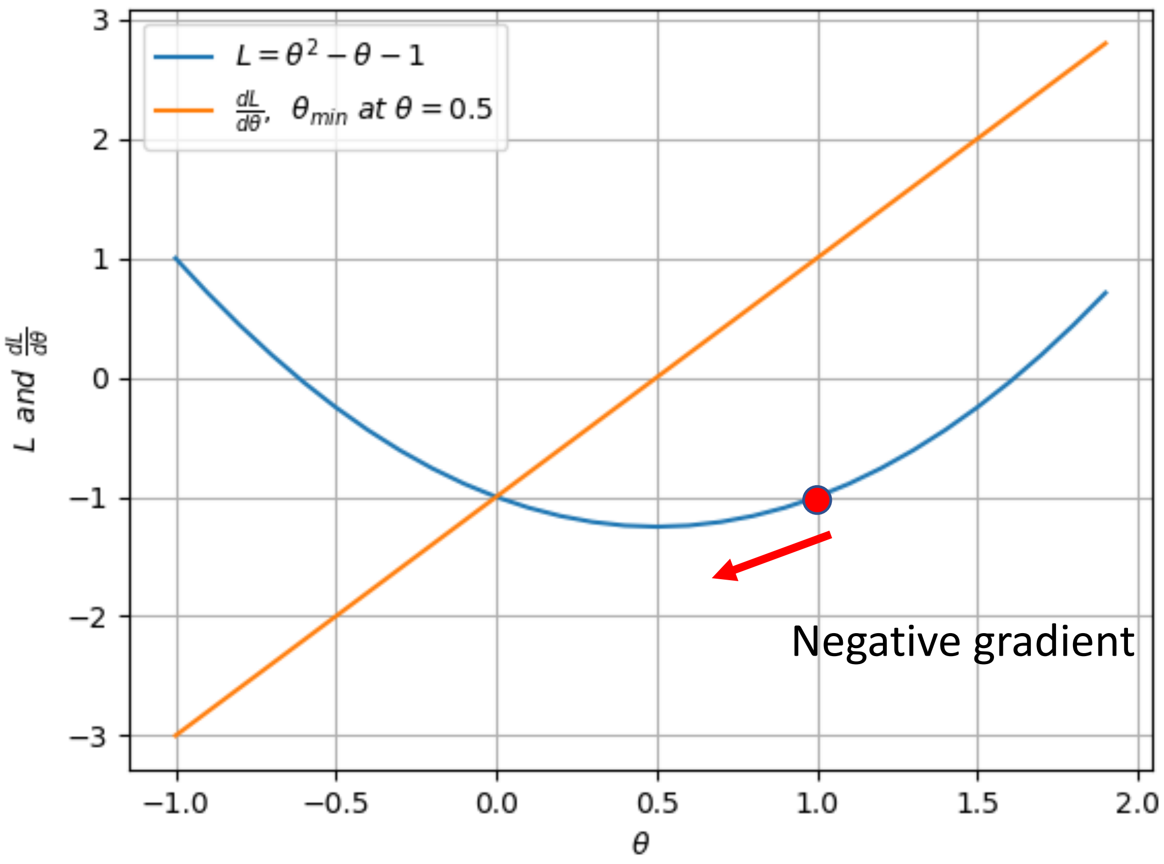
$$\frac{dL}{d\theta} = 2\theta - 1$$

Let us say $\theta_0 = 1$ & $\varepsilon = 1$

$$\frac{dL}{d\theta} = 1$$

$$\theta = 1 - 1(1) = 0$$

We will always miss the the minimum of L at: $\theta = 0.5$



Right Learning Rate to Overcome Local Minima

Learning rate $\varepsilon = 0.01$

Gradient:

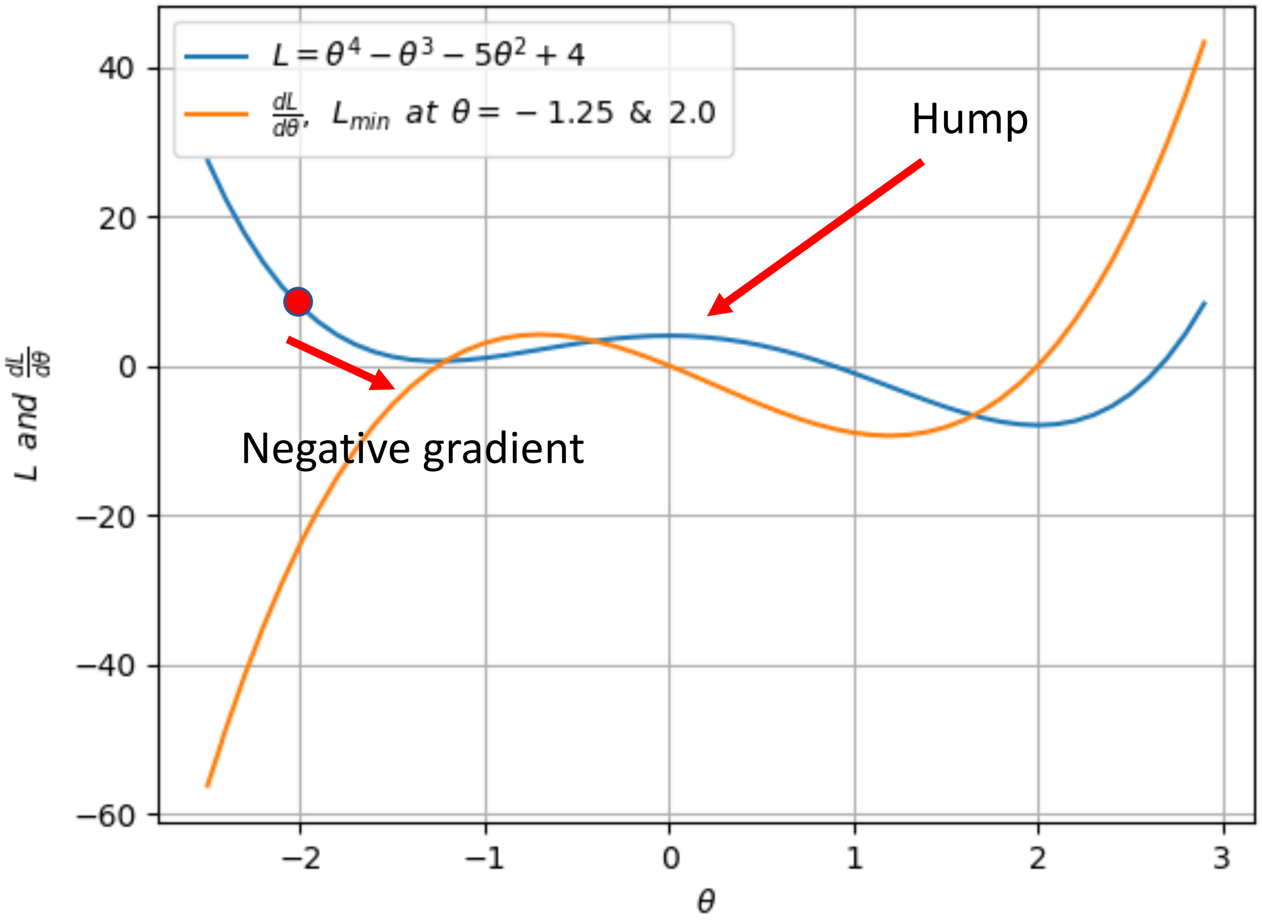
$$\frac{dL}{d\theta} = 4\theta^3 - 3\theta^2 - 10\theta$$

Let us say $\theta_0 = -2$ & $\varepsilon = 0.01$

$$\frac{dL}{d\theta} = -24$$

$$\theta = -2 - 0.01(-24) = -1.76$$

We get stuck in the local convex bowl and find minimum at $\theta = -1.25$. No way we can over the hump and discover a smaller minimum at $\theta = 2$.



Learning rate $\varepsilon = 0.1$

Gradient:

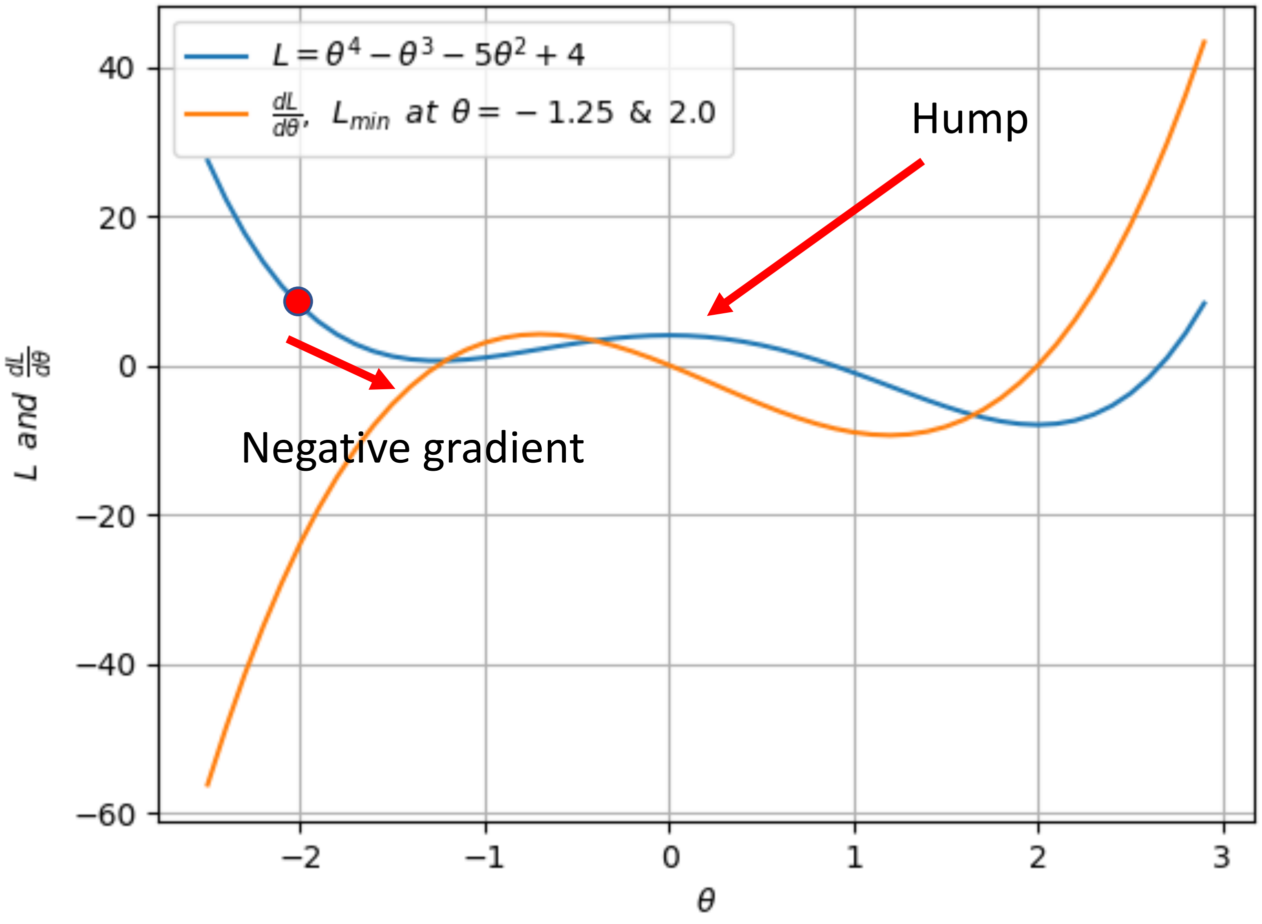
$$\frac{dL}{d\theta} = 4\theta^3 - 3\theta^2 - 10\theta$$

Let us say $\theta_0 = -2$ & $\varepsilon = 0.1$

$$\frac{dL}{d\theta} = -24$$

$$\theta = -2 - 0.1(-24) = 0.4$$

We went over the hump and may eventually discover the global minimum at $\theta = 2$.



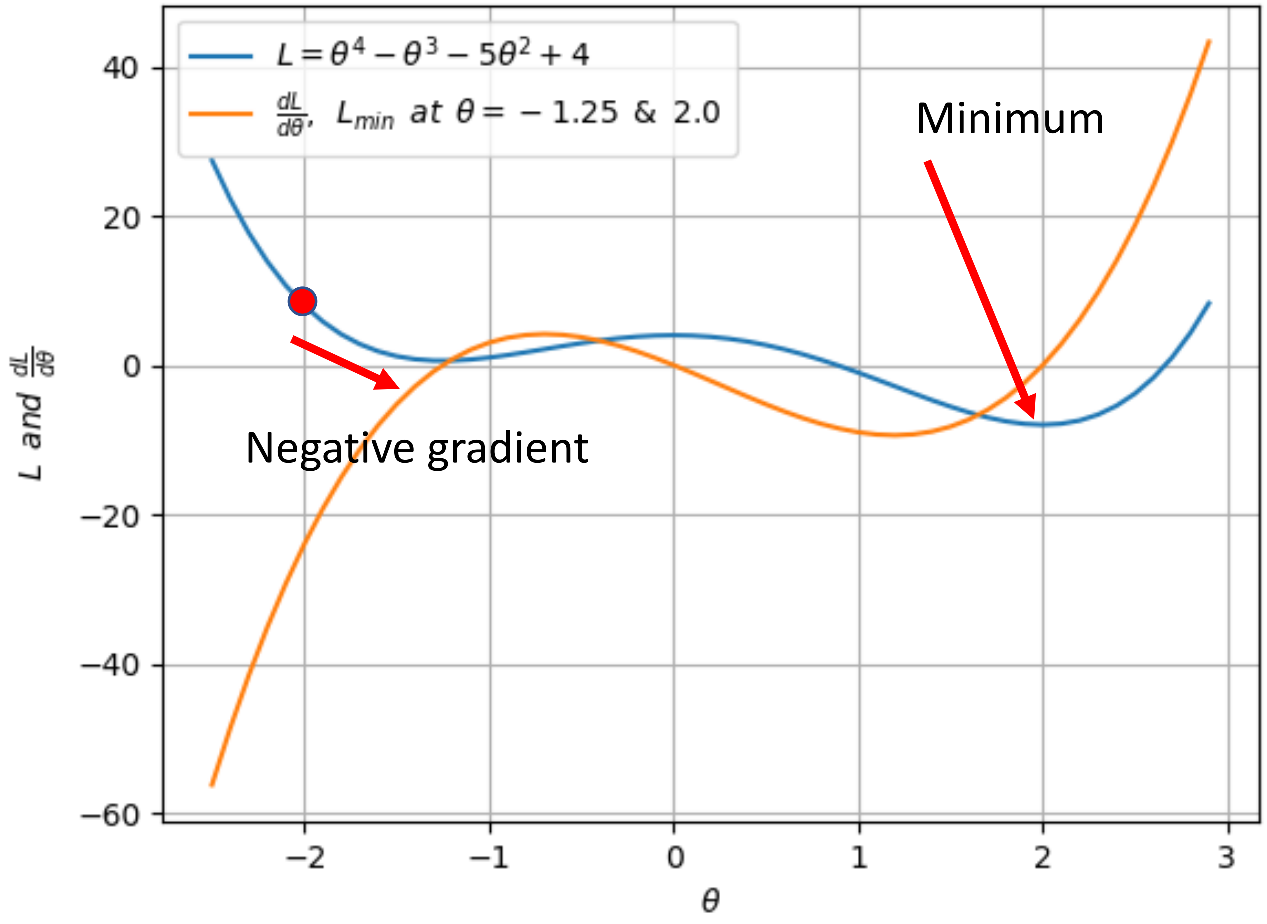
Learning Rate Scheduler

Decreasing Learning Rate Near the Minimum

10-step update at learning rate $\varepsilon = 0.1$

θ, Updated θ
-2.00, 0.40
0.40, 0.82
0.82, 1.63
1.63, 2.33
2.33, 1.24
1.24, 2.18
2.18, 1.64
1.64, 2.32
2.32, 1.25
1.25, 2.19

We will always miss the global minimum at $\theta = 2$.

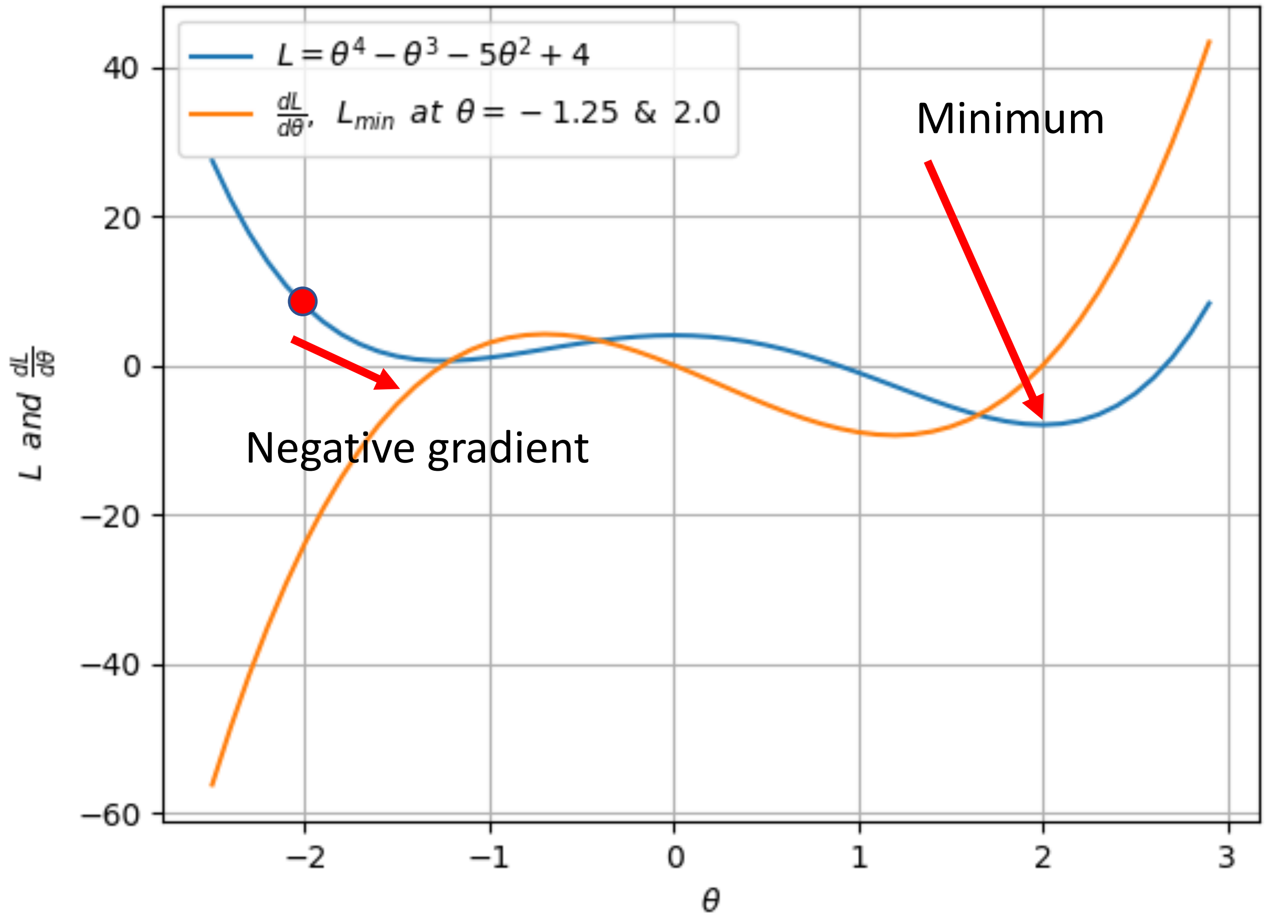


20-step update at learning rate $\varepsilon = \begin{cases} 0.005 & step > 15 \\ 0.01 & step > 5 \\ 0.1 & else \end{cases}$

θ, Updated θ
-2.00, 0.40
0.40, 0.82
0.82, 1.63
1.63, 2.33
2.33, 1.24
1.24, 2.18
2.18, 2.13
2.13, 2.09
2.09, 2.07
2.07, 2.05
2.05, 2.03

θ, Updated θ
2.03, 2.03
2.03, 2.02
2.02, 2.01
2.01, 2.01
2.01, 2.01
2.01, 2.01
2.01, 2.01
2.01, 2.00
2.00, 2.00

At step=18, we hit the minimum at $\theta = 2$.



Gradient Descent with Momentum

Accelerated Learning

Gradient Descent with Momentum

Rationale: Add contribution of past gradients to the current update to accelerate learning (ie faster convergence)

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \varepsilon\big(\nabla L(\boldsymbol{\theta}_i)\big)^T + \alpha g$$

$$g = \boldsymbol{\theta}_i - \boldsymbol{\theta}_{i-1}$$

g is the momentum term, $\alpha \in [0,1]$

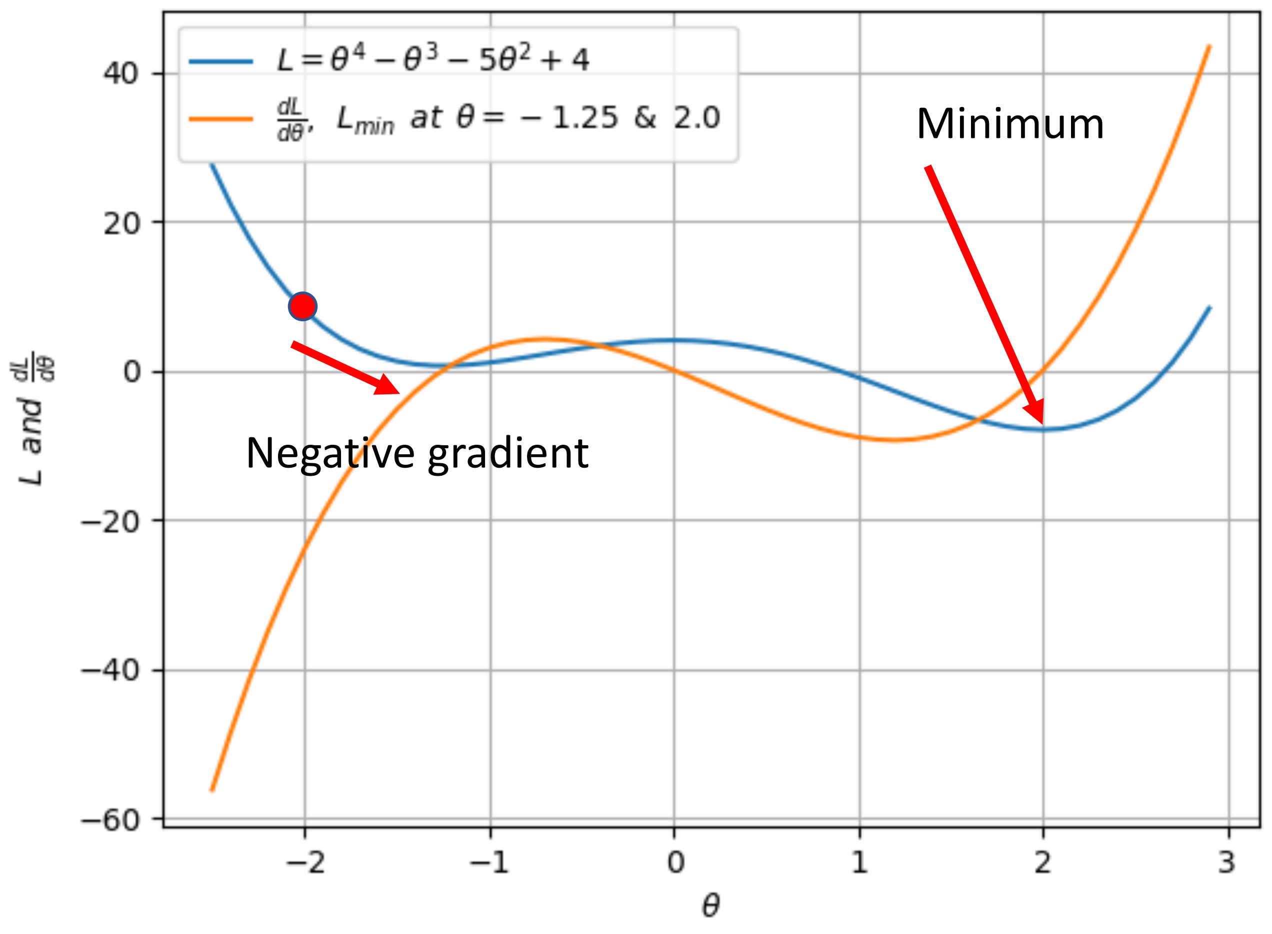
Gradient Descent with Momentum α=0.1

20-step update at learning rate $\varepsilon = \begin{cases} 0.005 & step > 15 \\ 0.01 & step > 5 \\ 0.1 & else \end{cases}$

θ, Updated θ
-2.00, 0.40
0.40, 1.06
1.06, 2.05
2.05, 2.01
2.01, 1.97
1.97, 2.04
2.04, 2.03
2.03, 2.02
2.02, 2.02
2.02, 2.01
2.01, 2.01

θ, Updated θ
2.01, 2.01
2.01, 2.00
2.00, 2.00
2.00, 2.00
2.00, 2.00
2.00, 2.00
2.00, 2.00
2.00, 2.00
2.00, 2.00

At step=12, we hit the minimum at $\theta = 2$.



```python
import numpy as np
import argparse


def gd(theta, lr=0.1, momentum=0.):
    grad = 4*theta**3 - 3*theta**2 - 10*theta
    theta = theta - lr*grad
    theta += momentum
    return theta

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--momentum',
                        default=False,
                        action='store_true',
                        help='use momentum')
    parser.add_argument('--schedule',
                        default=False,
                        action='store_true',
                        help='use learning rate schedule')
    parser.add_argument('--lr',
                        default=0.1,
                        type=float,
                        help='use learning rate schedule')
    args = parser.parse_args()
```

```python
    theta_0 = -2
    lr = args.lr
    momentum = 0
    alpha = 0.1
    for i in range(20):
        if args.schedule:
            if i > 15:
                lr = 0.005
            elif i > 5:
                lr = 0.01


        theta_1 = gd(theta_0, lr=lr,
                     momentum=alpha*momentum)
        print("%0.2f, %0.2f" %
              (theta_0, theta_1))

        if args.momentum:
            momentum = theta_1 - theta_0
        theta_0 = theta_1
```

Stochastic Gradient Descent

An Estimate of Gradient Descent

Stochastic Gradient Descent (SGD)

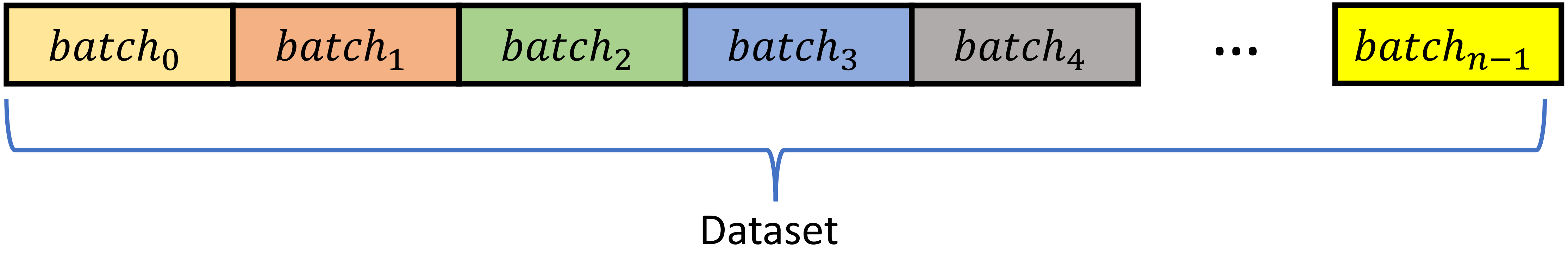
In gradient descent, we use the entire dataset $\mathcal{D} = \{\boldsymbol{x}, \boldsymbol{y}\}$ to estimate the gradient

If the dataset $\mathcal{D} = \{\boldsymbol{x}, \boldsymbol{y}\}$ is huge (eg hundreds or millions points), the computation is expensive

Estimating the gradient by a small sample of the dataset called *mini-batch* usually leads to a good approximation

Gradient Descent : Use entire dataset to compute $\nabla L(\boldsymbol{\theta}_i)$ for 1 update

Stochastic Gradient Descent : Use random samples from dataset (small mini-batches) to compute $\nabla L(\boldsymbol{\theta}_i)$ for n updates



SGD

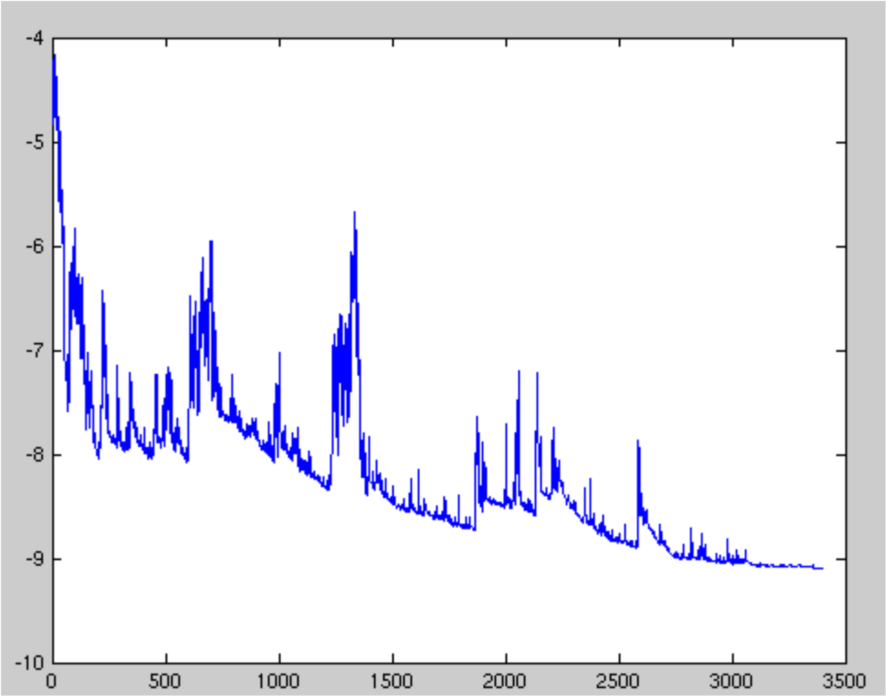
Pros

Many gradient updates – faster convergence

Can be done in parallel

Cons

Since it is a mini-batch, SGD is susceptible to noise



Wikipedia

Visualization

Navigating the loss surface

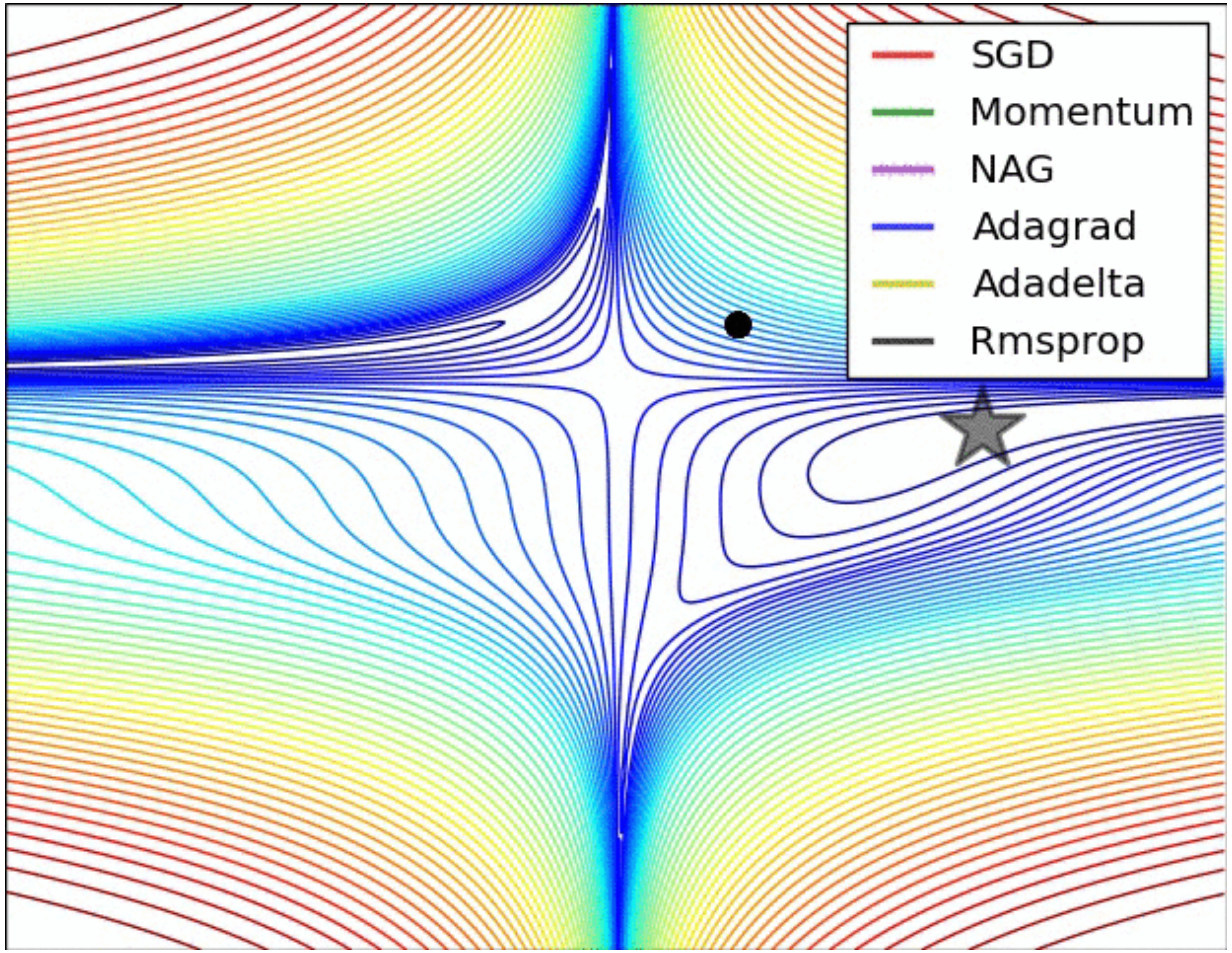


Image: Alec Radford

Navigating a saddle point

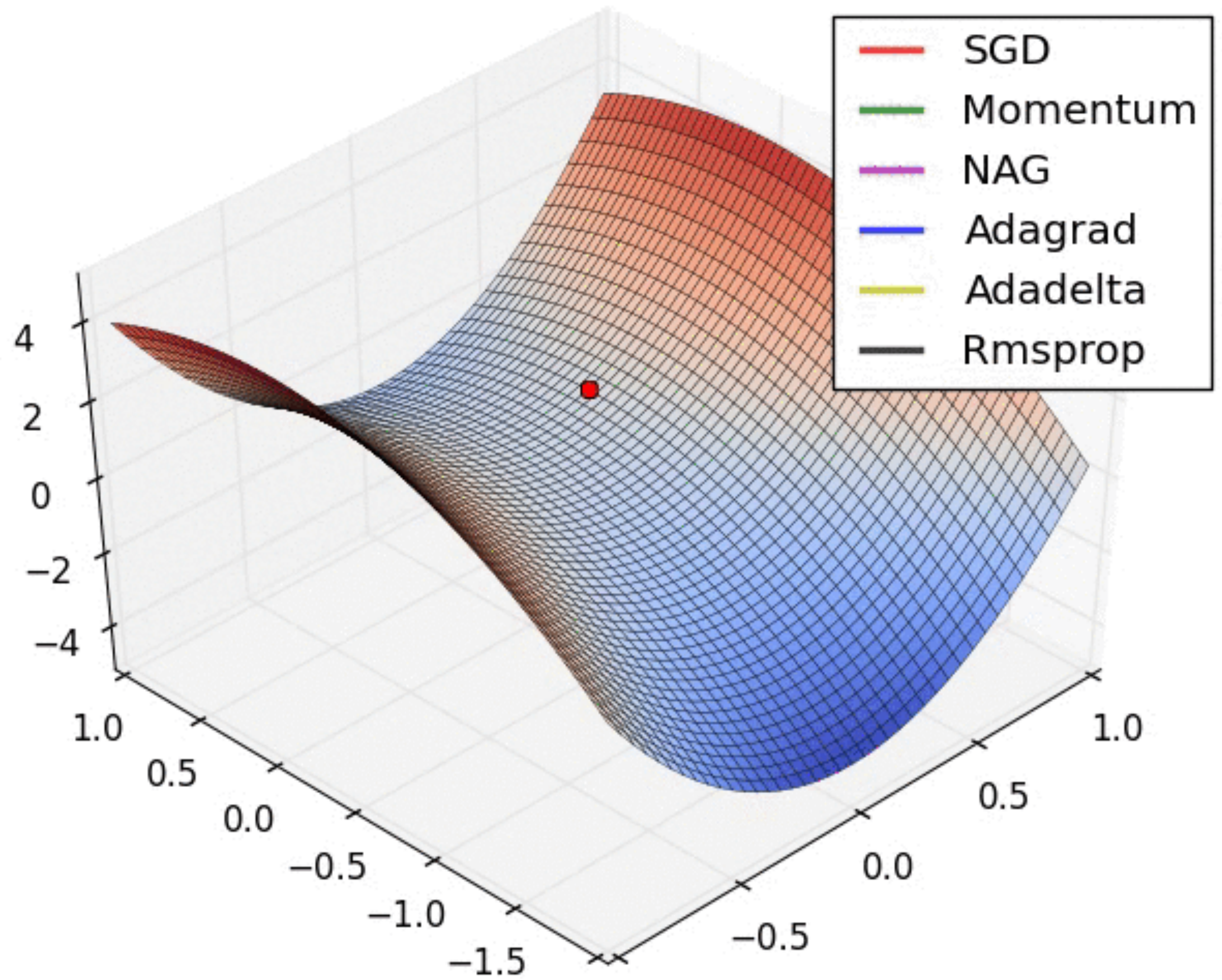


Image: Alec Radford

Moving Forward

Assignment on Backpro Due Nov 16 12mn

5 more lectures (we might have lec on Nov 30 – a holiday)

Need to plan on final project proposal and presentation in Dec

Constrained Optimization

Constrained Optimization

Unconstrained Optimization, $f : \mathbb{R}^D \to \mathbb{R}$:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$

For example, $\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$.

Constrained Optimization, $f : \mathbb{R}^D \to \mathbb{R}$:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$

$$subject\ to \quad g_i(\boldsymbol{x}) \leq 0 \quad i = 1,2,\ldots,m$$

Lagrange Multiplier

Merging the constraint and target function:

$$\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x})$$

$$\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T g(\boldsymbol{x})$$

$\lambda_i \geq 0$ are called Lagrange Multipliers

Note that $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ can be non-convex functions

Lagrange Duality

Duality: convert the optimization in one set of variables (e.g. $\boldsymbol{x}$) called primal variables into another set of variables (e.g. $\boldsymbol{\lambda}$) called dual variables

Constrained Optimization, $f : \mathbb{R}^D \to \mathbb{R}$:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$

$$subject\ to \quad g_i(\boldsymbol{x}) \leq 0 \quad i = 1,2,\ldots,m$$

is called the primal problem corresponding to primal variable $\boldsymbol{x}$

Lagrange Duality

The corresponding dual problem:

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^m} \mathfrak{D}(\boldsymbol{\lambda})$$

$$subject\ to \quad \boldsymbol{\lambda} \geq 0$$

Where the dual variable is $\boldsymbol{\lambda}$ and $\mathfrak{D}(\boldsymbol{\lambda}) = \min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x})$

MinMax Inequality

For any function with 2 arguments $\varphi(\boldsymbol{x}, \boldsymbol{y})$, the maximin is less than the minimax:

$$\max_{\boldsymbol{y}} \min_{\boldsymbol{x}} \varphi(\boldsymbol{x}, \boldsymbol{y}) \leq \min_{\boldsymbol{x}} \max_{\boldsymbol{y}} \varphi(\boldsymbol{x}, \boldsymbol{y})$$

Weak Duality

In Lagrange Multiplier, the objective is:

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda} \geq 0} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$$

Since $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is a lower bound of $J(\boldsymbol{x})$:

$$J(\boldsymbol{x}) = \max_{\boldsymbol{\lambda} \geq 0} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$$

Weak Duality

Using minmax inequality, we arrive at weak duality:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \max_{\boldsymbol{\lambda}\geq 0} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \geq \max_{\boldsymbol{\lambda}\geq 0} \min_{\boldsymbol{x}\in\mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$$

Modified Constrained Optimization

Equality Constrained Optimization, $f : \mathbb{R}^D \to \mathbb{R}$:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$

$$subject\ to \quad g_i(\boldsymbol{x}) \leq 0 \quad i = 1,2, \dots, m$$
$$and \quad h_j(\boldsymbol{x}) = 0 \quad i = 1,2, \dots, n$$

Convex Optimization

Global optimization guarantee

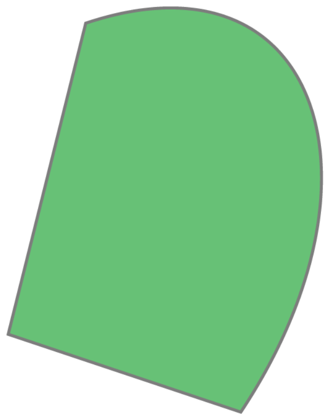
Convex Set

A set $\mathcal{C}$ is a convex set if for any $x, y \in \mathcal{C}$ and for any scalar $0 \leq \theta \leq 1$:

$$\theta x + (1 - \theta)y \in \mathcal{C}$$

Straight line connecting 2 elements are in the set

Figure 7.5 Example of a convex set.



Convex Function: Jensen's Inequality

Let $f : \mathbb{R}^D \rightarrow \mathbb{R}$ be a function whose domain in a convex set. The function is a convex function if for all $\boldsymbol{x}, \boldsymbol{y}$ in the the domain of f and for any scalar $0 \leq \theta \leq 1$:

$$f(\theta \boldsymbol{x} + (1 - \theta)\boldsymbol{y}) \leq \theta f(\boldsymbol{x}) + (1 - \theta) f(\boldsymbol{x})$$

A concave function is the negative of a convex function

Epigraph

Imagine filling up a bowl (a convex function) with water, the resulting filled in set is called an epigraph

Convexity

A function f is convex if and only if 2 points x and y, it holds that:

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} f(\boldsymbol{x})^T (\boldsymbol{y} - \boldsymbol{x})$$

If the Hessian, $\nabla_{\boldsymbol{x}}^2 f$, exists then it is positive semidefinite.

Example: $f(x) = x \log x$

For $\theta = 0.5$, $x = 2$ and $y = 4$ prove:

$$f(\theta \boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leq \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{x})$$

Alternatively, prove:

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} f(\boldsymbol{x})^T (\boldsymbol{y} - \boldsymbol{x})$$

Linear Programming

Linear Programming

All functions are linear. Primal Problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}) = \min_{\boldsymbol{x}\in\mathbb{R}^d} \boldsymbol{c}^T\boldsymbol{x}$$

$$\textit{subject to} \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$$

$\boldsymbol{A} \in \mathbb{R}^{m\times d}, \boldsymbol{b} \in \mathbb{R}^{m}, \boldsymbol{c} \in \mathbb{R}^{\boldsymbol{d}}$

Lagrangian

$$\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{c}^T \boldsymbol{x} + \boldsymbol{\lambda}^T (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$$

$$\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = (\boldsymbol{c} + \boldsymbol{A}^T \boldsymbol{\lambda})^T \boldsymbol{x} - \boldsymbol{\lambda}^T \boldsymbol{b}$$

Where $\boldsymbol{\lambda} \in \mathbb{R}^m$ are the Lagrange multipliers.

$$\frac{d\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})}{d\boldsymbol{x}} = \boldsymbol{c} + \boldsymbol{A}^T \boldsymbol{\lambda} = \boldsymbol{0}$$

Substituting the zero term $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$, the dual is $\mathfrak{D}(\boldsymbol{\lambda}) = -\boldsymbol{\lambda}^T \boldsymbol{b}$

Dual Optimization

Dual Problem:

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^d} \mathfrak{D}(\boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda} \in \mathbb{R}^d} -\boldsymbol{\lambda}^T \boldsymbol{b}$$

$$subject\ to \quad \boldsymbol{c} + \boldsymbol{A}^T \boldsymbol{\lambda} = \boldsymbol{0}$$

$$\boldsymbol{\lambda} \geq \boldsymbol{0}$$

Quadratic Programming

Quadratic Programming

Convex quadratic objective function. Primal Problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}) = \min_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x}$$

$$subject\ to \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$$

$\boldsymbol{A} \in \mathbb{R}^{m\times d}$, $\boldsymbol{b} \in \mathbb{R}^{m}$, $\boldsymbol{c} \in \mathbb{R}^{\boldsymbol{d}}$

$\boldsymbol{Q} \in \mathbb{R}^{d\times d}$ is symmetric positive definite square matrix

Lagrangian

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

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

Lagrangian

Where $\boldsymbol{\lambda} \in \mathbb{R}^m$ are the Lagrange multipliers.

$$\frac{d\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})}{d\boldsymbol{x}} = \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda} = \boldsymbol{0}$$

Assuming $\boldsymbol{Q}$ is invertible:

$$\boldsymbol{x} = -\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})$$

Lagrangian

Substituting into the primal problem, the dual and its optimization problem is:

$$\mathfrak{D}(\boldsymbol{\lambda}) = -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})^T\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}) - \boldsymbol{\lambda}^T\boldsymbol{b}$$

$$\max_{\boldsymbol{\lambda}\in\mathbb{R}^d} \mathfrak{D}(\boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda}\in\mathbb{R}^d} -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})^T\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}) - \boldsymbol{\lambda}^T\boldsymbol{b}$$

$$\boldsymbol{\lambda} \geq \boldsymbol{0}$$

Convex Conjugate

Convex Conjugate

The convex conjugate of a function $f : \mathbb{R}^D \rightarrow \mathbb{R}$ is a function f^* defined by:

$$f^*(\boldsymbol{s}) = \sup_{\boldsymbol{x} \in \mathbb{R}^D} \left(\langle \boldsymbol{s}, \boldsymbol{x} \rangle - f(\boldsymbol{x}) \right)$$

Convex Conjugate

The convex conjugate of a function $f : \mathbb{R}^D \to \mathbb{R}$ is a function f^* defined by:

$$f^*(\boldsymbol{s}) = \sup_{\boldsymbol{x} \in \mathbb{R}^D} \left(\langle \boldsymbol{s}, \boldsymbol{x} \rangle - f(\boldsymbol{x}) \right)$$

$\langle \boldsymbol{s}, \boldsymbol{x} \rangle$ can be a dot product.