

Line search methods for first and second order optimization

Ciprian-Mihai Ceașescu

Supervised by: Bogdan Alexe, University of Bucharest



FACULTATEA DE

MATEMATICĂ
ȘI INFORMATICĂ

June 26, 2019

Overview

- 1 Introduction
- 2 Convex and non-convex optimization
- 3 Optimisation methods
- 4 Line search methods
- 5 Results
- 6 Conclusions
- 7 Future work

Overview

- 1 Introduction
- 2 Convex and non-convex optimization
- 3 Optimisation methods
- 4 Line search methods
- 5 Results
- 6 Conclusions
- 7 Future work

Machine Learning (ML)

ML algorithms recognize the pattern and the structure of data. Methods:

- **supervised learning** - classification and regression;
- **unsupervised learning** - clustering and representation learning.

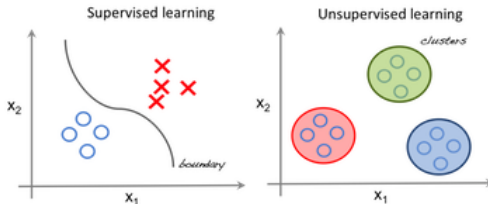


Figure: *Supervised and Unsupervised learning methods. Source: George Seif.*

Artificial Neuron

Proposed in 1943 by **Warren McCulloch** and **Walter Pitts**, with the following structure:

- **boolean inputs** and **weights**;
- **threshold**;
- **output**.

$$x_1, x_2, \dots, x_n \in \{0,1\}, \quad w_1, w_2, \dots, w_n \in \{-1,1\}, \quad \theta \in \mathbb{R}, \quad \hat{y} \in \{0,1\}$$

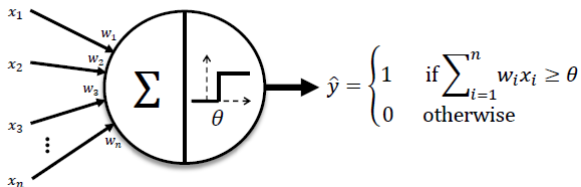


Figure: Artificial Neuron. Source: Data Mining Lectures - Sparktech.

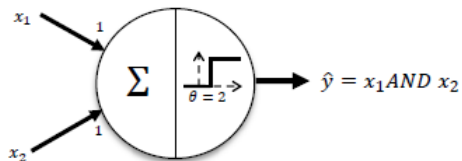


Figure: AND gate. Source: Data Mining Lectures - Sparktech.

Introduction

Artificial Neural Network (ANN)

ANN represents a nonlinear model extensively used in ML tasks. Layers:

- **input** - provide data;
- **hidden** - perform computational operations;
- **output** - perform computations and provide the results.

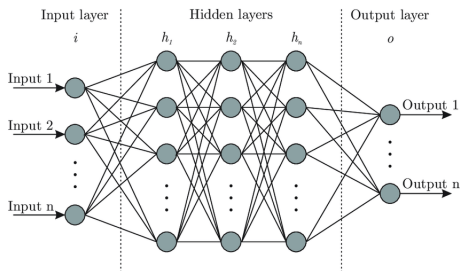


Figure: Artificial Neural Network. Source: Facundo Bre.

Introduction

Training process of ANN

Two main steps:

- **forward-propagation;**
- **backward-propagation** using **loss function**.

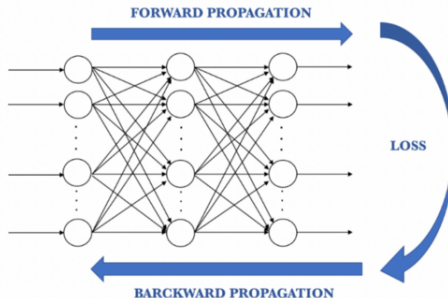


Figure: *Training process.* Source: Jordi Torres.

Loss function

Method of evaluating how well your algorithm models your dataset. Loss:

- **high** - predictions are totally off;
- **low** - predictions are pretty good.

The optimization task is to reduce the value of the loss function, $J(\theta)$:

$$J(\theta) = \mathbb{E}_{(x,y) \sim \hat{p}_{data}} L(f(x; \theta), y), \quad (1)$$

L - loss function, $f(x; \theta)$ - predicted value for the x input, \hat{p}_{data} - empirical distribution.

Loss function

During the training process, the **empirical risk** is minimized:

$$\mathbb{E}_{x,y \sim \hat{p}_{data}(x,y)} [L(f(x; \theta), y)] = \frac{1}{m} \sum_{i=1}^m L(f(x^{(i)}; \theta), y^{(i)}), \quad (2)$$

m - number of samples, L - loss function, $f(x; \theta)$ - predicted value for the x input, \hat{p}_{data} - empirical distribution.

Overview

- 1 Introduction
- 2 Convex and non-convex optimization
- 3 Optimisation methods
- 4 Line search methods
- 5 Results
- 6 Conclusions
- 7 Future work

Convex and non-convex optimization

Convex problems

Defined as problems where:

- all constraints are convex functions;
- the objective is a **convex** function if minimizing, or a **concave** function if maximizing.

Linear functions are convex, so linear programming problems are convex problems.

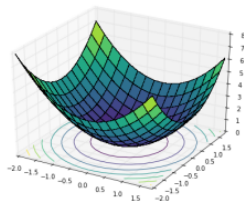


Figure: *Convex problem.* Source: Jan Mrkos.

Convex and non-convex problems

Non-convex problems

Defined as problems where:

- all constraints are non-convex functions;
- might exist multiple feasible regions and multiple locally optimal points within each region.

Neural Networks are non-convex problems.

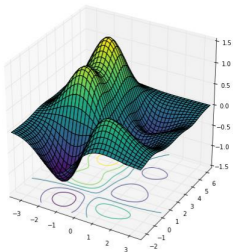


Figure: *Non-convex problem.* Source: Jan Mrkos.

Overview

- 1 Introduction
- 2 Convex and non-convex optimization
- 3 Optimisation methods**
- 4 Line search methods
- 5 Results
- 6 Conclusions
- 7 Future work

First Order methods

Algorithms that require **first-derivative** or **gradient** for the update step are defined as **first order** algorithms. The optimization methods iterate the following update:

$$x = x - \eta \nabla f(x),$$

where x are input data, $\nabla f(x)$ is the Jacobian matrix (square matrix of first-order partial derivatives of the loss function) and η the step size.

Second Order methods

Algorithms that require **second-derivative** for the update step are defined as **second order** algorithms. In the context of Deep Learning, these optimization method are based on Newton's method, which iterates the following update:

$$x = x - [Hf(x)]^{-1} \nabla f(x),$$

where x are input data, $Hf(x)$ is the Hessian matrix (square matrix of second-order partial derivatives of the loss function).

Challenges

Finding a good learning rate η aims to find a step length that is neither:

- too short;
- too long.

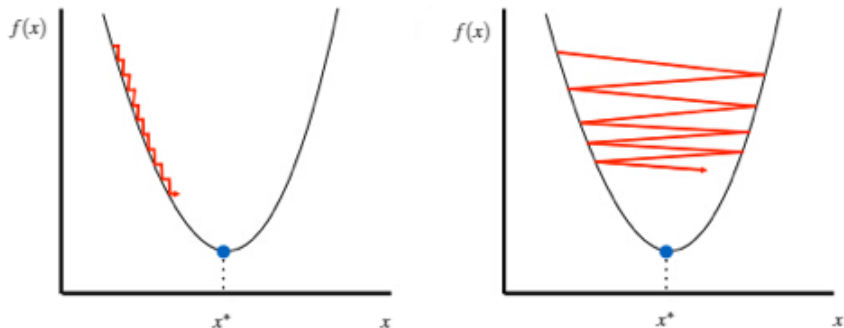


Figure: *Small learning rate (left) / Large learning rate (right).*

Overview

- 1 Introduction
- 2 Convex and non-convex optimization
- 3 Optimisation methods
- 4 Line search methods**
- 5 Results
- 6 Conclusions
- 7 Future work

General idea

An algorithm is called a **line search** method if:

- it searches for the minimum of a nonlinear function by selecting a reasonable **direction vector**;
- it will provide a function value closer to the absolute minimum when the direction vector is computed iteratively with a reasonable **step size**.

Statement

Given the function $f(x)$ and an initial x_k , to find a lower value of $f(x)$, the value of x_{k+1} is increased by the following iteration scheme:

$$x_{k+1} = x_k + \alpha_k p_k,$$

where $\alpha_k > 0$ is the **step length** and p_k defines the **step direction**.

Line search methods

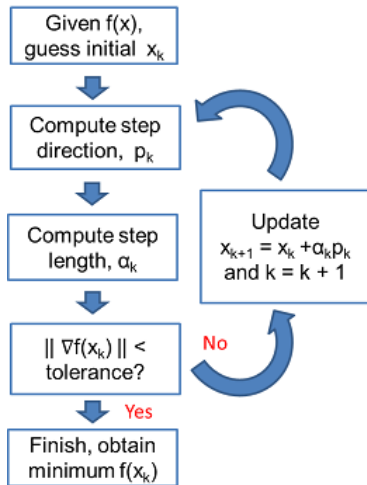


Figure: Line search algorithm flow chart. Source: Elizabeth Conger.

Methods

In our implementation we present three different line search methods:

- **backtracking;**
- **Goldstein;**
- **Weak Wolfe.**

The main idea of the algorithms: starting from an initial value α_k we update it until the conditions are met.

Backtracking

Sufficient decrease - given the loss, $f(x_k)$, and the direction, p_k , a good **step length**, α_k , satisfies the condition:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \beta \alpha_k g_k^T p_k, \quad (3)$$

where $0 < \beta < 1$ and $g_k = \nabla f(x)$.

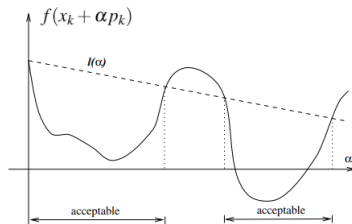


Figure: Sufficient decrease. Source: Nocedal and Wright.

Goldstein

Sufficient decrease and **control step size** - given the loss, $f(x_k)$, and the direction, p_k , a good **step length**, α_k , satisfies the condition:

$$f(x_k + \alpha_k p_k) \geq f(x_k) + (1 - \rho)\alpha_k g_k^T p_k, \quad (4)$$

where $0 < \rho < \frac{1}{2}$ and $g_k = \nabla f(x)$.

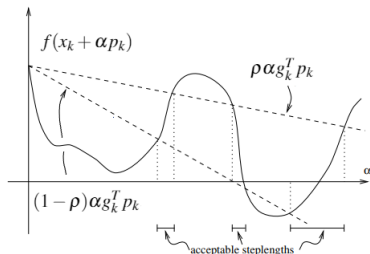


Figure: Control step size. Source: Nocedal and Wright.

Weak Wolfe

Sufficient decrease and **curvature condition** - given the loss, $f(x_k)$, and the direction, p_k , a good **step length**, α_k , satisfies the condition:

$$g_{k+1}^T p_k \geq \sigma g_k^T p_k, \quad (5)$$

where $0 < \rho < \frac{1}{2}$, $\sigma \in (\rho, 1)$ and $g_k = \nabla f(x)$.

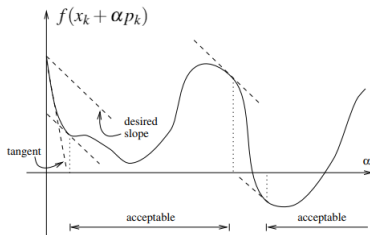


Figure: Curvature condition. Source: Nocedal and Wright.

Line search methods

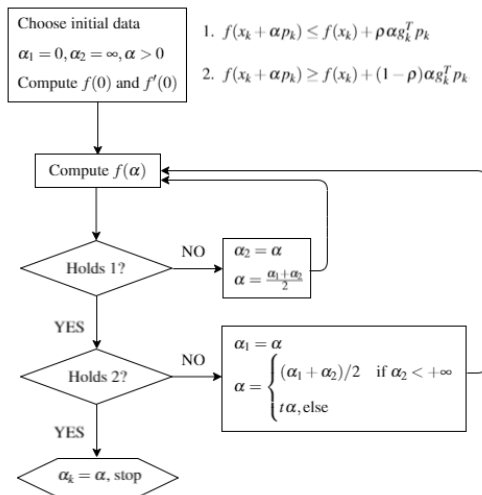


Figure: Goldstein algorithm. Source: Wenyu Sun and Ya-Xiang Yuan.

Line search methods

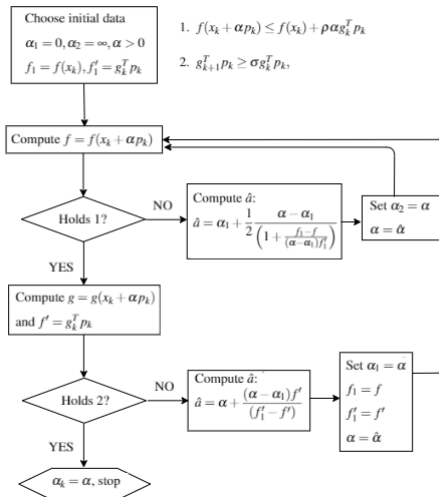


Figure: Weak Wolfe algorithm. Source: Wenyu Sun and Ya-Xiang Yuan.

Overview

- 1 Introduction
- 2 Convex and non-convex optimization
- 3 Optimisation methods
- 4 Line search methods
- 5 Results**
- 6 Conclusions
- 7 Future work

Results: Convex problem

Linear regression

Applies a linear transformation to the incoming data:

$$y = xA^T + b, \quad (6)$$

where A , b represent the parameters and x the input data.

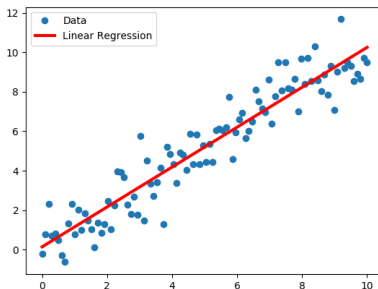


Figure: *Linear regression.*

Results: Non-convex problem

Convolutional Neural Network

Algorithm which takes as input an image and predicts its label, by applying several operations. It is most commonly applied to analyzing visual imagery.

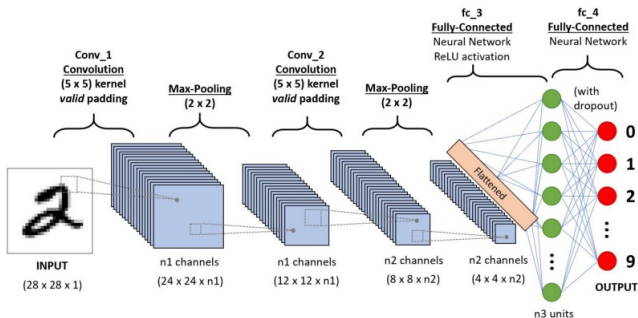


Figure: *Convolutional Neural Network.* Source: Sumit Saha.

Results: Non-convex problem

Residual Neural Network

ResNet prevents **vanishing gradients** problem by introducing **residual block**.

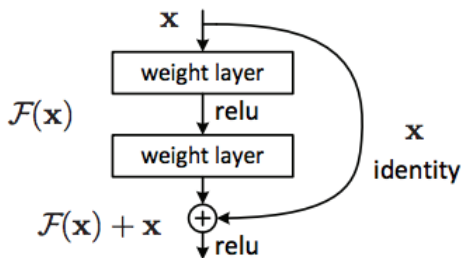


Figure: *Residual block.*

Results: Convex / MNIST / SGD

Model	Linear Regression
Dataset	MNIST 60k – 10k
Loss function	Cross Entropy
Optimizer	SGD with momentum 0.9
Epochs	30
Batch size	2048
Learning rate SGD	0.001
α_1 for SGD LS	0
α_2 for SGD LS	0.001

Results: Convex / MNIST / SGD

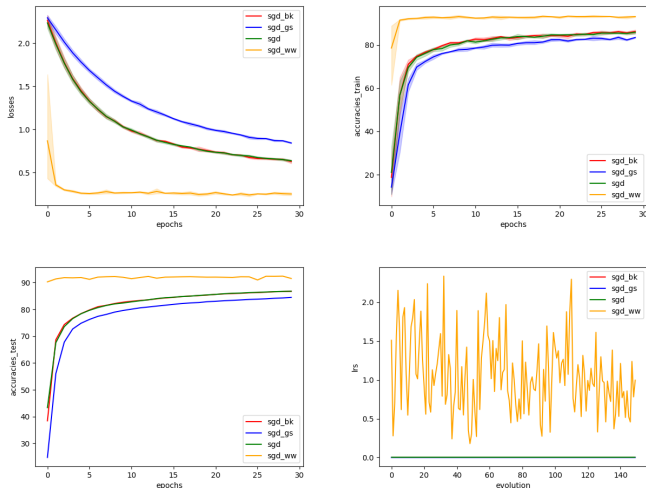


Figure: *Loss evolution / Train accuracy / Test accuracy / Learning rates.*

Results: Convex / MNIST / L-BFGS

Model	Linear Regression
Dataset	MNIST 60k – 10k
Loss function	Cross Entropy
Optimizer	L-BFGS
Epochs	30
Batch size	2048
Learning rate L-BFGS	0.001
α_1 for L-BFGS LS	0
α_2 for L-BFGS LS	0.001

Results: Convex / MNIST / L-BFGS

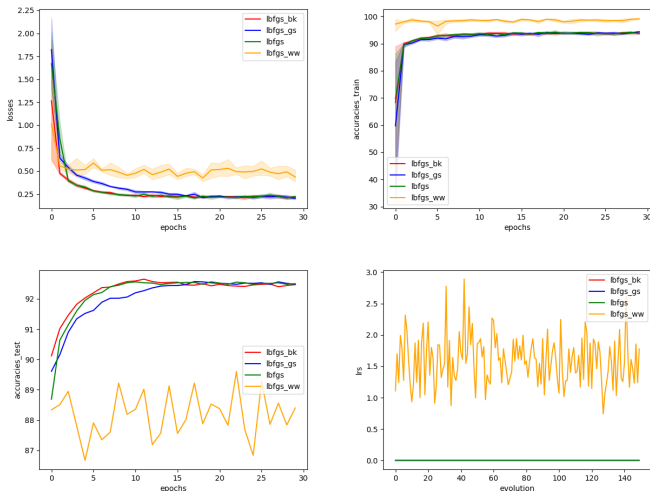


Figure: *Loss evolution / Train accuracy / Test accuracy / Learning rates.*

Results: Convex / MNIST / SGD & L-BFGS

Model	Linear Regression
Dataset	MNIST 60k – 10k
Loss function	Cross Entropy
Optimizer	L-BFGS
	SGD (mom 0)
	SGD (mom 0.5)
Epochs	30
Batch size	1024
Learning rate 1	0.01
Learning rate 2	0.1

Results: Convex / MNIST / SGD & L-BFGS

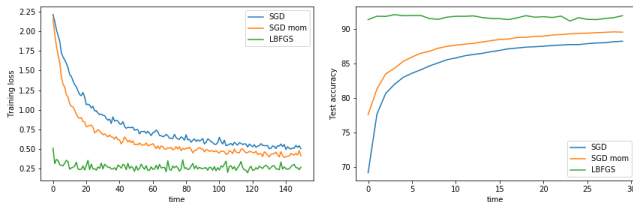


Figure: *Loss evolution / Test accuracy.*

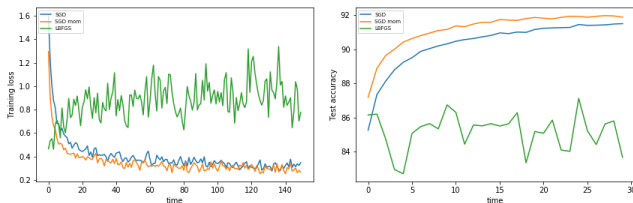


Figure: *Loss evolution / Test accuracy.*

Results: Non-convex / CIFAR-10 / SGD

Model	ResNet 20
Dataset	CIFAR-10 50k – 10k
Loss function	Cross Entropy
Optimizer	SGD with momentum 0.9
Epochs	100
Batch size	2048
Learning rate SGD	0.1 / 0.01 (ep. 80)
α_1 for SGD LS	0
α_2 for SGD LS	0.1

Results: Non-convex / CIFAR-10 / SGD

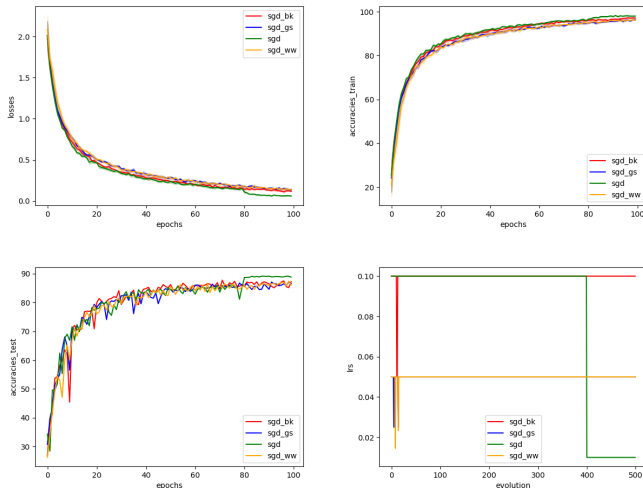


Figure: *Loss evolution / Train accuracy / Test accuracy / Learning rates.*

Results: Non-convex / CIFAR-10 / L-BFGS

Model	ResNet 20
Dataset	CIFAR-10 50k – 10k
Loss function	Cross Entropy
Optimizer	L-BFGS
Epochs	30
Batch size	2048
Learning rate L-BFGS	0.1
α_1 for L-BFGS LS	0
α_2 for L-BFGS LS	0.1

Results: Non-convex / CIFAR-10 / L-BFGS

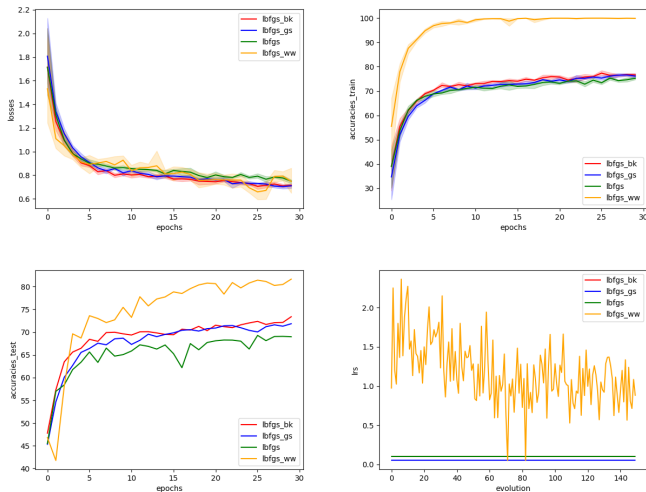


Figure: *Loss evolution / Train accuracy / Test accuracy / Learning rates.*

Results: Non-convex / CIFAR-10 / L-BFGS

Model	ResNet 20
Dataset	CIFAR-10 50k – 10k
Loss function	Cross Entropy
Optimizer	L-BFGS
Epochs	30
Batch size	2048
Learning rate L-BFGS	-
α_1 for L-BFGS LS	0
α_2 for L-BFGS LS	∞

Results: Non-convex / CIFAR-10 / L-BFGS

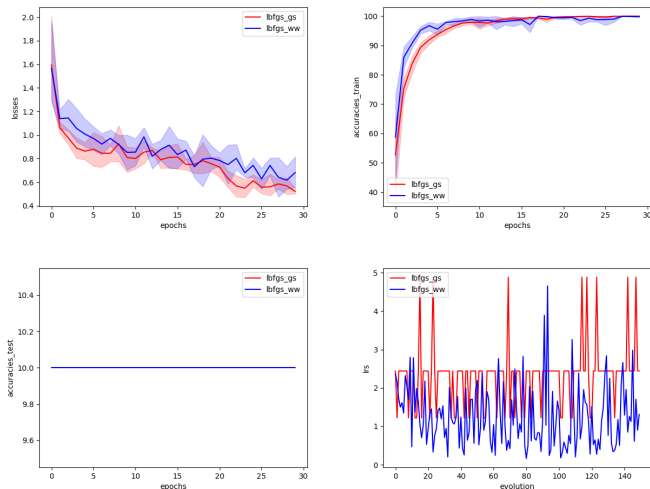


Figure: *Loss evolution / Train accuracy / Test accuracy / Learning rates.*

Overview

- 1 Introduction
- 2 Convex and non-convex optimization
- 3 Optimisation methods
- 4 Line search methods
- 5 Results
- 6 Conclusions**
- 7 Future work

Searching for Learning Rates

- good approach in training neural networks;
- need to be careful of overfitting the model, in order to generalize;
- Weak Wolfe is the best method, because the conditions are stronger, but the process is computational costly;
- method for detecting the right values for α_1 and α_2 produce better results (Nocedal and Wright).

Overview

- 1 Introduction
- 2 Convex and non-convex optimization
- 3 Optimisation methods
- 4 Line search methods
- 5 Results
- 6 Conclusions
- 7 Future work**

Open questions

- do line search strategies help with SGD on regular batch size and on large batch size?
- do the MNIST results from the previous experiments (convex problem) generalize to other architectures (e.g. AlexNet, VGG, LeNet) and datasets (CIFAR-10/100, ImageNet)?
- how large is the update of the model from an epoch to another (implement a progress measure based on the gradients)?
- how can we parallelize line search methods in order to perform faster?

Questions?

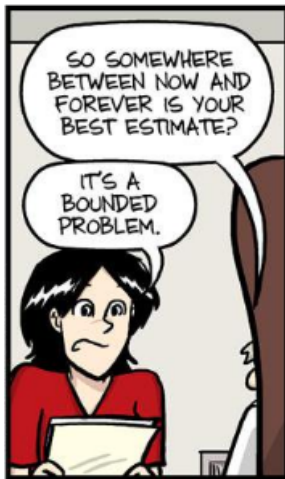








Figure: *Source: PHD Comics.*

Bibliography I

-  Ian Goodfellow, Yoshua Bengio, and Aaron Courville.
Deep Learning.
-  Stephen Boyd and Lieven Vandenberghe.
Convex Optimization.
-  Prateek Jain and Purushottam Kar.
Non-convex Optimization for Machine Learning.
-  Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton.
Imagenet classification with deep convolutional neural networks.
-  Wenyu Sun and Ya-Xiang Yuan.
Optimization Theory and Methods Nonlinear Programming.
-  Jorge Nocedal and Stephen J. Wright.
Numerical Optimization.

Thank you!



Line search methods for first and second order optimization

Ciprian-Mihai Ceașescu

Supervised by: Bogdan Alexe, University of Bucharest



FACULTATEA DE

MATEMATICĂ
ȘI INFORMATICĂ

June 26, 2019