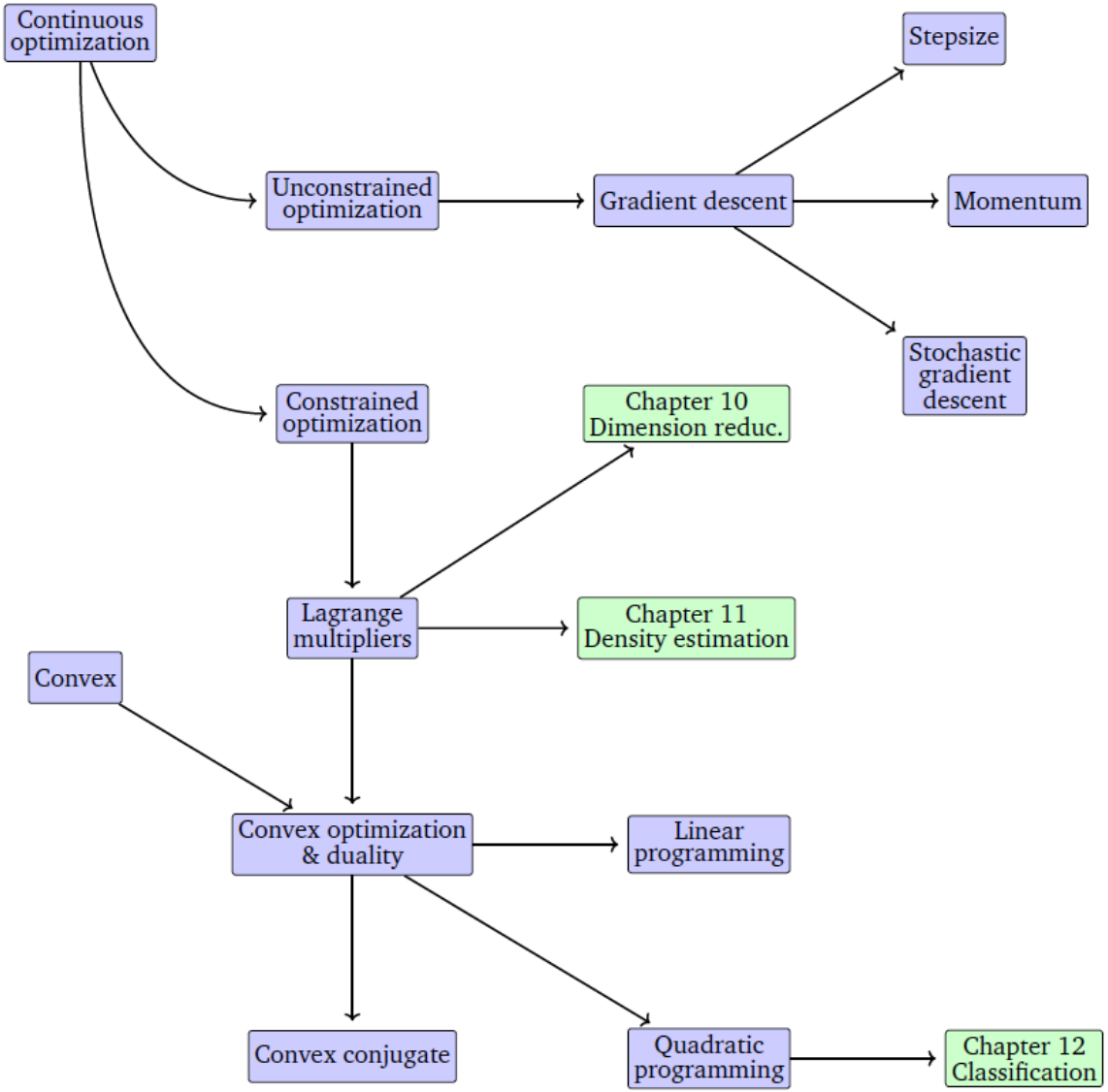
Continuous Optimization 1 of 2

Continuous Optimization

Typically in ML our models are guided by parameters, the questions we often have to face is:

- What does it mean to have a "good" set of parameters?
  - We typically have some type of cost function as our objective function.
  - In ML we mostly consider minimization as the goal.
- How do I find a good set of parameters?
  - Directly analytically obtainable
  - Not Directly analytically obtainable
  - What can we say about the form of the objective function.
    - ★ Do we have explicit access to the objective function?
    - ★ Is the objective function differentiable?
    - ★ Is the function convex?

Continuous Optimization: Mindmap



Source: M.P. Deisenroth *et al*, Mathematics for Machine Learning (First Edition)

Polynomial Example

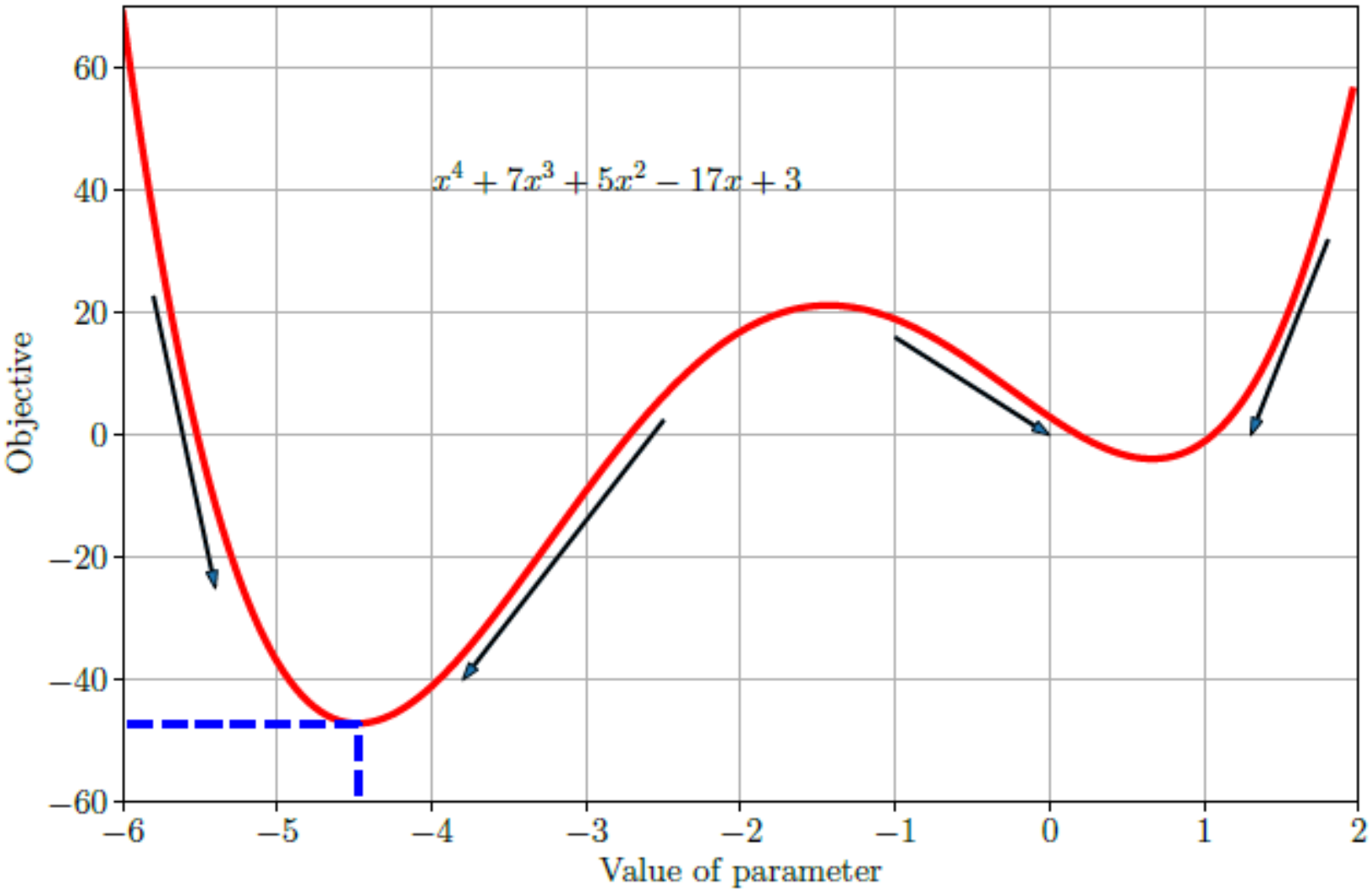
Let use consider a polynomial example.

$$I(x) = x^4 + 7x^3 + 5x^2 - 17x + 3 \tag{1}$$

If we are looking for the minimum we can follow the straight forward approach

- Find all stationary point by solving $\frac{d\,I(x)}{dx} = 0$
  - In this case $\frac{d\,I(x)}{dx} = 4x^3 + 21x^2 + 10x - 17 = 0$ is satisfied for $x \approx -4.4803,\ -1.4321,\ 0.66238$
- We can then check each of these points to determine if they are a minimum or maximum:
  - If $\frac{d^2\,I(x)}{d^2x}$ is greater than 0 for a given point it is a minimum.
  - If $\frac{d^2\,I(x)}{d^2x}$ is less than 0 for a given point it is a maximum.

Polynomial Example



Source: M.P. Deisenroth *et al*, Mathematics for Machine Learning (First Edition)

Polynomial Example

The problem is that in practice, solving $\frac{d\,l(x)}{dx} = 0$ (or the more complex multivariate case) is often not possible.

- So we are left with a search problem.
- Which in ML is typically guided by the derivative of the function (in the case of first order methods)

Let us use our polynomial example:

- Say we start at $x_0 = -5.5$ then we see that the derivative is negative.
- This means that if we move to the "right" by a sufficiently small amount we will get a lower objective value.
  - Specifically, by $+\gamma \frac{d\,l}{dx}(x_0)$
  - Where γ is the step size.
- What if we started at $x_0 = 0$?

Optimization Using Gradient Descent

Let use assume that we are trying to solve

$$f : \mathbb{R}^d \to \mathbb{R} \tag{2}$$

$$\min_{\mathbf{x}} f(\mathbf{x}) \tag{3}$$

and that f is differentiable. Then for a starting point $\mathbf{x}_0 \in \mathbb{R}^d$

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma_i((\nabla f)(\mathbf{x}_i))^T \tag{4}$$

for a suitable step-size γ_i, the sequence $f(\mathbf{x}_0) \geq f(\mathbf{x}_1) \geq \ldots$ converges to a local minimum.

Gradient Descent: Example

Consider, $f : \mathbb{R}^2 \to \mathbb{R}$,

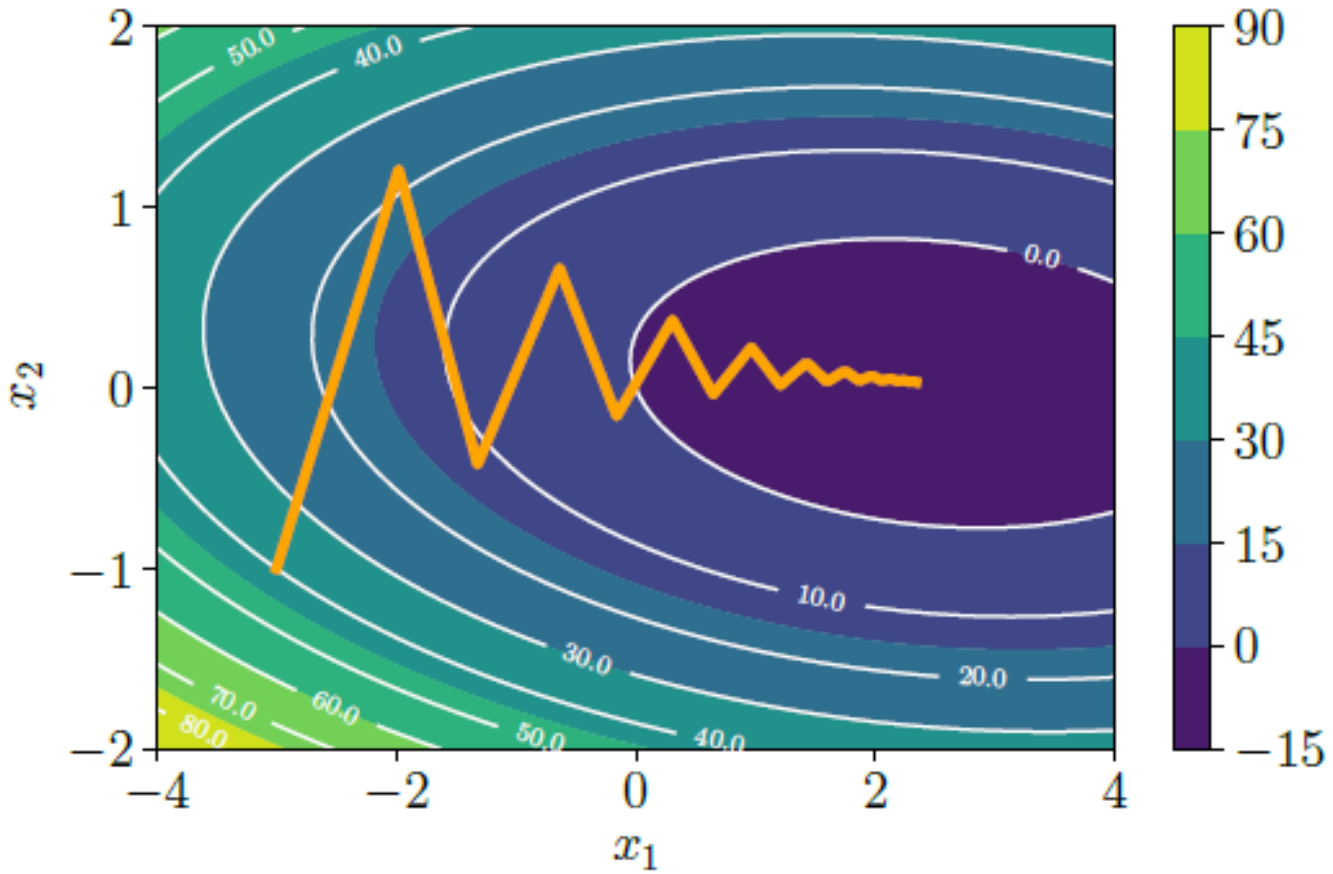
$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} \mathbf{x} - \begin{bmatrix} 5 & 3 \end{bmatrix} \mathbf{x} \tag{5}$$

from which we can calculate

$$\nabla f(\mathbf{x}) = \mathbf{x}^T \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} - \begin{bmatrix} 5 & 3 \end{bmatrix} \tag{6}$$

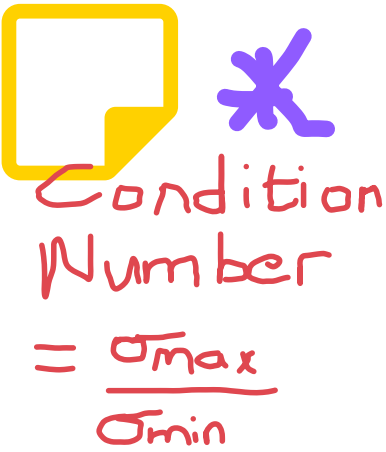
Gradient Descent: Example

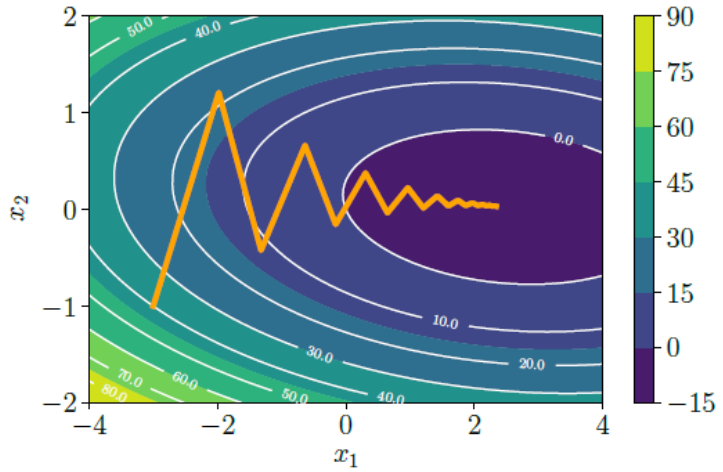
Starting at the initial location $\mathbf{x}_0 = \left[-3, -1\right]^T$, if we iteratively apply (4) using $\gamma = 0.085$ to obtain a sequence of estimates that converge to the minimum value



Source: M.P. Deisenroth *et al*, Mathematics for Machine Learning (First Edition)

Gradient Descent: Example





Source: M.P. Deisenroth *et al*, Mathematics for Machine Learning (First Edition)

- Gradient descent can be relatively slow close to the minimum:
  - Its asymptotic rate of convergence is inferior to many other methods.
  - For poorly conditioned convex problems, gradient descent increasingly "zigzags" as the gradients point nearly orthogonally to the shortest direction to a minimum point as can be seen in this figure.

Gradient Descent: Step-size

Choosing a good step-size is important in gradient descent.

- If the step-size is too small, gradient descent can be slow.
- If the step-size is chosen too large, gradient descent can overshoot, fail to converge, or even diverge.

The second consideration is the optimal step size early in the search might be different than later in the search.

Gradient Descent: Step-size

Adaptive gradient methods rescale the step-size at each iteration, depending on local properties of the function. There are two simple heuristics that can be used together

- When the function value increases after a gradient step, the step-size was too large. Undo the step and decrease the step-size.
  - Although the "undo" step seems to be a waste of resources, using this heuristic guarantees monotonic convergence
- When the function value decreases the step could have been larger. Try to increase the step-size.

Gradient Descent With Momentum

Gradient descent with momentum (Rumelhart et al., 1986) is a method that introduces an additional term to remember what happened in the previous iteration.

- This memory dampens oscillations and smoothes out the gradient updates.
- The momentum-based method remembers the update $\Delta \mathbf{x}_i$ at each iteration i and determines the next update as a linear combination of the current and previous gradients.

Gradient Descent With Momentum

Specifically,

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma_i((\nabla f)(\mathbf{x}_i))^T + \alpha\Delta\mathbf{x}_i \tag{7}$$

$$\Delta\mathbf{x}_i = \mathbf{x}_i - \mathbf{x}_{i-1} = \alpha\Delta\mathbf{x}_{i-1} - \gamma_{i-1}((\nabla f)(\mathbf{x}_{i-1}))^T \tag{8}$$

where $\alpha \in [0, 1]$.

- GD with momentum is particularly useful if we only a noisy estimate of the gradient, as we average out over the two estimates.

Stochastic Gradient Descent

Stochastic gradient descent (often shortened as SGD) is a stochastic approximation of the gradient descent method for minimizing an objective function that is written as a sum of differentiable functions.

- The word stochastic here refers to the fact that we acknowledge that we do not know the gradient precisely, but instead only know a noisy approximation to it.
- By constraining the probability distribution of the approximate gradients, we can still theoretically guarantee that SGD will converge.

Stochastic Gradient Descent

In machine learning, given $n = 1, \ldots, N$ data points, we often consider objective functions that are the sum of the losses L_n incurred by each example n. Specifically,

$$L(\theta) = \sum_{n=1}^{N} L_n(\boldsymbol{\theta}) \tag{9}$$

where $\boldsymbol{\theta}$ is the vector of parameters of interest

- An example from ML is the negative loglikelihood, which is expressed as a sum over log-likelihoods of individual examples so that

$$L(\theta) = -\sum_{n=1}^{N} \log p(y_n|\mathbf{x}_n, \boldsymbol{\theta}) \tag{10}$$

where $\mathbf{x}_n \in \mathbb{R}^D$ are the training inputs, y_n are the training targets, and $\boldsymbol{\theta}$ are the parameters of the regression model.

Stochastic Gradient Descent

Using the full training data set used it is typically refer to as "batch" optimization method.

- A hold over from early computing, where you would have a big batch job in the evening.
- The main down side is:
  - Your parameter update step becomes very computationally heavy as it grows with your data set size

Stochastic Gradient Descent

With Stochastic Gradient Descent we use a randomly selected subset of the L_n term (we consider only a subset of the data) for mini-batch gradient descent.

- In the most extreme case you use individual random samples (online learning)

Why does SGD work?

- In short the partial sum is an unbias estimator of the true gradient.
- Note: When the step size rate decreases at an appropriate rate, and subject to relatively mild assumptions, stochastic gradient descent converges almost surely to local minimum (Bottou, 1998 [1]).

[1] Bottou, Léon. 1998. Online Algorithms and Stochastic Approximations. Pages 9–42 of: Online Learning and Neural Networks. Cambridge University Press.

Stochastic Gradient Descent

Picking a batch size:

- Large mini-batch sizes will provide accurate estimates of the gradient, reducing the variance in the parameter update.
  - Furthermore, large mini-batches take advantage of highly optimized matrix operations in vectorized implementations of the cost and gradient.
  - The reduction in variance leads to more stable convergence, but each gradient calculation will be more expensive.

Stochastic Gradient Descent

Picking a batch size:

- Small mini-batches are quick to estimate.
- If we keep the mini-batch size small, the noise in our gradient estimate will allow us to get out of some bad local optima, which we may otherwise get stuck in.
- If the batch size is too small we may have too much variance leading to "undoing" progress or divergent behavior