An Introduction to Optimization

Jerónimo Arenas-García, Jesús Cid Sueiro

Universidad Carlos III de Madrid

jeronimo.arenas@uc3m.es

September 12, 2018

Sources

This presentation is based on the following online resources:

- Convex Optimization Overview by Boid et al.
- Mathematical optimization with Scipy by Gael Varoquaux
- An Introduction to Gradient Descent by Faizan Shaikh

You can follow the following notebooks to learn more about optimization methods implementations in Python:

- Notebook on Gradient Descent for Optimization
- Notebook on Stochastic Gradient Descent (very important when cost function is based on sum over data)
- Tutorial Notebook providing an introduction to optimization with Scipy (intro.ipynb)

Contents

Mathematical optimization

Mathematical optimization consists in the minimization or maximization of an objective function, possibly subject to a set of constraints.

Definition

Minimize $\quad \mathbf{w}^* = \arg\min_{\mathbf{w}} f(\mathbf{w})$

Subject to $\quad f_i(\mathbf{w}) \leq 0, \ i = 1, \cdots, m \qquad$ (inequality constraints)

$\qquad\qquad\quad g_i(\mathbf{w}) = 0, \ i = 1, \cdots, p \qquad$ (equality constraints)

- $\mathbf{w} \in \Re^d$ is a set of parameters, and $\mathbf{w}^*$ is the optimal solution
- $f(\mathbf{w})$ is the objective or goal function
- Solution may exist only if the **feasible set** is not empty
- Box constraints can be considered as particular cases of inequality constraints

Mathematical optimization in machine learning

- In supervised Machine Learning we are typically given a set of data

$$\mathcal{D} = \left\{ \mathbf{x}^{(k)}, y^{(k)} \right\}_{k=1}^{K}$$

where labels $y^{(k)}$ represent the desired output or prediction associated to data vector $\mathbf{x}^{(k)}$.

- The problem can be formulated as that of learning a function that models the relation between $\mathbf{x}$ and y

$$y = h(\mathbf{x})$$

- In parametric methods, the function is parameterized with a set of parameters, $h_{\mathbf{w}}(\mathbf{x})$, and learning the function is equivalent to learning a set of parameters $\mathbf{w}^*$, i.e.,

$$h^*(\mathbf{x}) = h_{\mathbf{w}^*}(\mathbf{x})$$

Mathematical optimization in machine learning (II)

- A typical objective of machine learning methods is to pursue that the postulated model fits well the training data, what frequently results in cost functions similar to

$$f(\mathbf{w}) = \sum_{k=1}^{K} \ell(y^{(k)}, h_{\mathbf{w}}(\mathbf{x}^{(k)}))$$

  where $\ell(\cdot, \cdot)$ is an error function that measures discrepancy between the arguments.
- In this context, optimization methods can be used to minimize $f(\mathbf{w})$, and therefore to obtain a good prediction model.
- Different methods are characterized by different objective functions and possibly the presence of constraints.
- Futhermore, regularization is sometimes considered to prevent solutions that fit extremely well the training data but generalize poorly to new data.

Mathematical optimization in machine learning (III)

$$f(\mathbf{w}) = \sum_{k=1}^{K} \ell(y^{(k)}, h_{\mathbf{w}}(\mathbf{x}^{(k)})) \qquad [+R(\mathbf{w})]$$

- Evaluation and optimization of the cost function can get challenging because of
  - Availability of very large datasets (big data)
  - The shape of $f(\mathbf{w})$, with many local minima and maxima (e.g., in deep learning)
- In this sense, we can say that the renaissance of Machine Learning in the last 20-30 years has been possible, at least in part, thanks to the development of powerful and efficient optimization techniques, as well as the increasing availability of computing power.

Convex optimization

Most optimization problems are intractable. Convex optimization is a very important exception to this statement.

Definition: Convex optimization problem

$$
\begin{array}{lll}
\text{Minimize} & \mathbf{w}^* = \arg\min_{\mathbf{w}} f(\mathbf{w}) & \\
\text{Subject to} & f_i(\mathbf{w}) \leq 0, \; i = 1, \cdots, m & \text{(inequality constraints)} \\
 & \mathbf{Aw} = \mathbf{b} & \text{(equality constraints)}
\end{array}
$$

- Equality constraints are linear
- Functions $f(\mathbf{w})$ and $f_i(\mathbf{w})$ are convex

LP and QP optimization

Linear Programming and Quadratic Programming are particular cases of convex optimization

- LP: Objective function and inequality constraints are also linear
- QP: Objective function is quadratic (in the parameters) and inequality constraints are linear

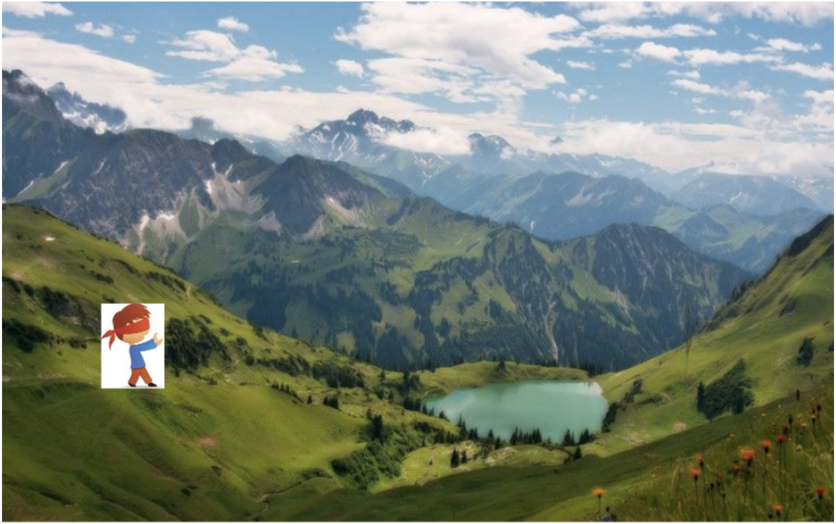
Specific methods are available for both cases that are more efficient than using a generic solver for convex optimization

Convex optimization in practice

Once you have successfully obtained a convex optimization formulation of your problem, you can usually solve it numerically

- Using available libraries for many programming languages.
  - If problem is LP or QP, more efficient algorithms can be used.
  - Reliably solved by interior-point methods on single machine
  - Polynomial complexity
- For Big Data problems, the previous implementations are not practical. You need to implement your own code.
  - Gradient and Stochastic gradient methods
  - Admit parallel implementations
  - Frequently used in machine learning (e.g., deep learning)

Motivation of Gradient Descent



Source:
http://cs231n.stanford.edu/slides/winter1516_lecture3.pdf

Gradient Descent Algorithm

- Very simple method for **unconstrained** optimization
- Iterate updates of the solution in the direction of maximum decrement of the function

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \rho_n \nabla_{\mathbf{w}} f(\mathbf{w}) \mid_{\mathbf{w}=\mathbf{w}_n}$$

  where $\rho_n > 0$ is a learning step
- Selection of the learning step is critical, since for small values we suffer small convergence, but too large values can result in algorithm divergence
- Common choices are $\rho_n = \frac{1}{n}$ or $\rho_n = \frac{\alpha}{1+\beta n}$
- In general (non convex) problems only convergence to a local minimum is achieved
- Can be initialized multiple times, keeping the best solution

Stochastic Gradient Descent (SGD) Algorithm

- For very large training sets, repeated computation of $f(\mathbf{w})$ and $\nabla_{\mathbf{w}} f(\mathbf{w})$ gets prohibitive

$$\nabla_{\mathbf{w}} f(\mathbf{w}) = \sum_{k=1}^{K} \nabla_{\mathbf{w}} \ell(y^{(k)}, h_{\mathbf{w}}(\mathbf{x}^{(k)})) \qquad [+\nabla_{\mathbf{w}} R(\mathbf{w})]$$

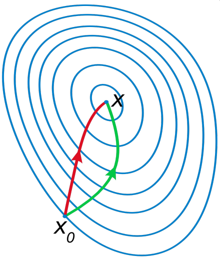
- In Stochastic Gradient Descent the gradient is approximated by a sum over subsets of the training samples, i.e.,

$$\hat{\nabla}_{\mathbf{w}} f(\mathbf{w}) = \sum_{k \in \mathcal{D}_n} \nabla_{\mathbf{w}} \ell(y^{(k)}, h_{\mathbf{w}}(\mathbf{x}^{(k)})) \qquad [+\nabla_{\mathbf{w}} R(\mathbf{w})]$$

  where $\mathcal{D}_n \subset \{1, \ldots, K\}$

- The computational cost of each iteration of SGD is much smaller than that of gradient descent, though it usually needs more iterations to converge.

Newton's Method Motivation



Source: Wikipedia

- The direction of fastest decrement does not in general point towards the minimum of the objective function
- Gradient Descent in such case can result in slow convergence and oscillations during the convergence
- Better updates can be obtained by analyzing the local curvature of the error function, but this requires evaluation of second derivatives

Newton's Method

- Approximation of $f(\mathbf{w})$ by its second order Taylor series expansion around $\mathbf{w}_n$

$$f(\mathbf{w}) = f(\mathbf{w}_n)+[\nabla_{\mathbf{w}} f(\mathbf{w}) \mid_{\mathbf{w}=\mathbf{w}_n}]^\top(\mathbf{w}-\mathbf{w}_n)+\frac{1}{2}(\mathbf{w}-\mathbf{w}_n)^\top \mathbf{H}(\mathbf{w}_n)(\mathbf{w}-\mathbf{w}_n)$$

- For the approximation, the minimum of $f(\mathbf{w})$ is given by

$$\mathbf{w}^\star = \mathbf{w}_n - [\mathbf{H}(\mathbf{w}_n)]^{-1}[\nabla_{\mathbf{w}} f(\mathbf{w}) \mid_{\mathbf{w}=\mathbf{w}_n}]$$

- Since the second order polynomial is just an approximation to $f(\mathbf{w})$ we can expect $\mathbf{w}^\star$ to be just an approximation to the optimum solution
- We can then iterate the procedure, i.e.,

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \rho_n[\mathbf{H}(\mathbf{w}_n)]^{-1}[\nabla_{\mathbf{w}} f(\mathbf{w}) \mid_{\mathbf{w}=\mathbf{w}_n}]$$

- The cost per iteration is larger than for GD, but we can expect that less iterations are necessary

Notebooks

In this course, you will just need to implement Gradient Descent for Logistic Regression. Thus, you are advised to study, at least, the first of the following selected notebooks

1. Notebook on Gradient Descent for Optimization
2. Notebook on Stochastic Gradient Descent (very important when cost function is based on sum over data)
3. Tutorial Notebook providing an introduction to optimization with Scipy (intro.ipynb)