7

Continuous Optimization

Recall from Section 1.1.4 that training a machine learning model often boils down to finding a good set of parameters. The notion of "good" is determined by the objective function or the probabilistic model, which we will see examples of in the second part of this book. Given an objective function finding the best value is done using optimization algorithms. We will assume in this chapter that our objective function is differentiable (see Chapter 5), hence we have access to a gradient at each location in the space to help us find the optimum value. By convention most objective functions in machine learning are intended to be minimized, that is the best value is the minimum value. Intuitively finding the best value is like finding the valleys of the objective function, and the gradients point us uphill. The idea is to move downhill (opposite to the gradient) and hope to find the deepest point.

Since we consider data and models in $\mathbb{R}^D$ the optimization problems we face are *continuous* optimization problems, as opposed to *combinatorial* optimization problems for discrete variables.

global minimum

Consider the function in Figure 7.1. The function has a *global minimum* around the value $x = -4.5$ which has the objective function value of

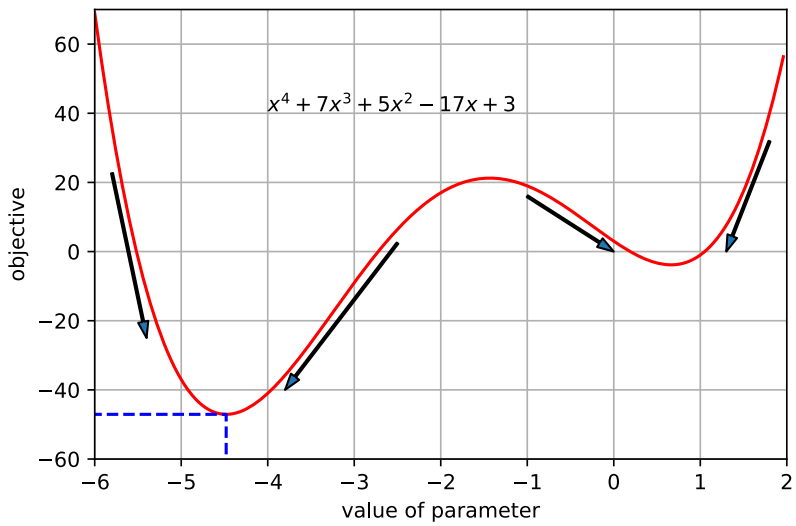

Figure 7.1 Example objective function. Gradients are indicated by arrows, and the global minimum is indicated by the dashed blue line.

Draft chapter (March 15, 2018) from "Mathematics for Machine Learning" ©2018 by Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. To be published by Cambridge University Press. Report errata and feedback to `mml-book.com`. Please do not post or distribute this file, please link to `mml-book.com`.

around -47. Since the function is "smooth" the gradients can be used to help find the minimum by indicating whether we should take a step to the right or left. This assumes that we are in the correct bowl, as there exists another *local minimum* around the value $x = 0.7$. Recall that we can solve for all the stationary points of a function by calculating its derivative and setting it to zero. Let

local minimum

Stationary points are points that have zero gradient.

$$\ell(x) = x^4 + 7x^3 + 5x^2 - 17x + 3. \tag{7.1}$$

Its gradient is given by

$$\frac{d\ell(x)}{dx} = 4x^3 + 21x^2 + 10x - 17. \tag{7.2}$$

Since this is a cubic equation, it has three solutions when set to zero. Two of them are minima and one is a maximum (around $x = -1.4$). Recall that to check whether a stationary point is a minimum or maximum we need to take the derivative a second time and check whether the second derivative is positive or negative at the stationary point.

$$\frac{d^2\ell(x)}{dx^2} = 12x^2 + 42x + 10 \tag{7.3}$$

By substituting our visually estimated values of $x = -4.5, -1.4, 0.7$ we will observe that as expected the middle point is a maximum $\left(\frac{d^2\ell(x)}{dx^2} < 0\right)$ and the other two stationary points are minimums.

Note that we have avoided analytically solving for values of x in the previous discussion, although for low order polynomials such as the above we could. In general, we are unable to find analytic solutions, and hence we need to start at some value, say $x_0 = -10$ and follow the gradient. The gradient indicates that we should go right, but not how far (this is called the step size). Furthermore, if we had started at the right side (e.g. $x_0 = 0$) the gradient would have led us to the wrong minimum. Figure 7.1 illustrates the fact that for $x > -1$, the gradient points towards the minimum on the right of the figure, which has a larger objective value.

In fact according to the Abel-Ruffini theorem, also known as Abel's impossibility theorem, there is in general no algebraic solution for polynomials of degree 5 or more.

We will see in Section 7.3 a class of functions called convex functions that do not exhibit this tricky dependency on the starting point of the optimization algorithm. For *convex functions* all local minima are global minimum. It turns out that many machine learning objective functions are designed such that they are convex, and we will see an example in Chapter 10.

convex functions

The discussion in this Chapter so far was about a one dimensional function, where we are able to visualize the ideas of gradients, descent directions and optimal values. In the rest of this chapter we develop the same ideas in high dimensions. Unfortunately we can only visualize the concepts in one dimension, but some concepts do not generalize directly to higher dimensions, therefore some care needs to be taken when reading.

7.1 Optimization using Gradient Descent

We now consider the problem of solving for the minimum of a real-valued function

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{7.4}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is an objective function that captures the machine learning problem at hand. We assume that our function f is differentiable, and we are unable to analytically find a solution in closed form.

Gradient descent is a first-order optimization algorithm. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient of the function at the current point. Recall from Chapter 5 that the gradient points in the direction of the steepest ascent and it is orthogonal to the contour lines of the function we wish to optimize.

We use the convention of row vectors for gradients.

Consider the quadratic function $f(x) = (x-1)^2$ in Figure 7.2. We begin our search for the minimum value at $x_0 = -0.7$, where the subscript 0 indicates the initial estimate. Observe that the gradient $\frac{d}{df(x)}x\Big|_{x=-0.7}$ points toward the right hand side. This means that we should take a step to the right (increasing x), which makes intuitive sense. However it is unclear how big a step to take, and the example algorithm overshoots the (unknown) minimum to $x_1 = 2.5$. Now the gradient points to the left, and we take a step to the left, and so on.

Let us consider multivariate functions. Imagine a surface (described by the function $f(\boldsymbol{x})$) with a ball starting at a particular location $\boldsymbol{x}_0$. When the ball is released, it will move downhill in the direction of steepest descent. Gradient descent exploits the fact that $f(\boldsymbol{x}_0)$ decreases fastest if one moves from $\boldsymbol{x}_0$ in the direction of the negative gradient $-((\nabla f)(\boldsymbol{x}_0))^\top$ of f at $\boldsymbol{x}_0$. We assume in this book that the functions are differentiable, and refer the reader to more general settings in Section 7.4. Then, if

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 - \gamma((\nabla f)(\boldsymbol{x}_0))^\top \tag{7.5}$$

for a small *step size* $\gamma \geqslant 0$ then $f(\boldsymbol{x}_1) \leqslant f(\boldsymbol{x}_0)$. Note that we use the transpose for the gradient since otherwise the dimensions will not work out.

This observation allows us to define a simple gradient-descent algorithm: If we want to find a local optimum $f(\boldsymbol{x}_*)$ of a function $f : \mathbb{R}^n \to \mathbb{R},\ \boldsymbol{x} \mapsto f(\boldsymbol{x})$, we start with an initial guess $\boldsymbol{x}_0$ of the parameters we wish to optimize and then iterate according to

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i((\nabla f)(\boldsymbol{x}_i))^\top . \tag{7.6}$$

For suitable step size γ_i, the sequence $f(\boldsymbol{x}_0) \geqslant f(\boldsymbol{x}_1) \geqslant \ldots$ converges to a local minimum.

Remark. Gradient descent can be relatively slow close to the minimum:

©2018 Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. To be published by Cambridge University Press.

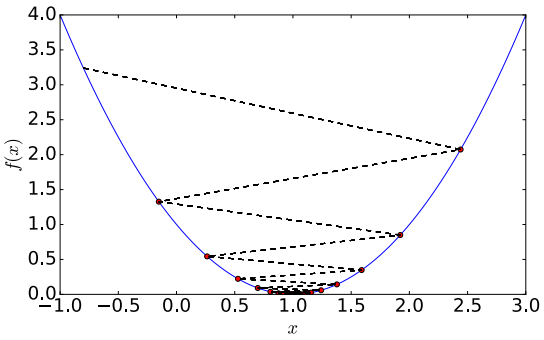

Figure 7.2 Gradient descent can lead to zigzagging and slow convergence.

Its asymptotic rate of convergence is inferior to many other methods. Using the ball rolling down the hill analogy, when the surface is a long thin valley the problem is poorly conditioned. For poorly conditioned convex problems, gradient descent increasingly 'zigzags' as the gradients point nearly orthogonally to the shortest direction to a minimum point, see Fig. 7.2. ♢

7.1.1 Stepsize

As mentioned earlier, choosing a good stepsize is important in gradient descent. If the stepsize is too small, gradient descent can be slow. If the stepsize is chosen too large, gradient descent can overshoot, fail to converge, or even diverge. We will discuss the use of momentum in the next section. It is a method that smoothes out erratic behavior of gradient updates and dampens oscillations.

The stepsize is also called the learning rate

Adaptive gradient methods rescale the stepsize at each iteration, depending on local properties of the function. There are two simple heuristics (Toussaint, 2012):

- When the function value increases after a gradient step, the step size was too large. Undo the step and decrease the stepsize.
- When the function value decreases the step could have been larger. Try to increase the stepsize.

Although the "undo" step seems to be a waste of resources, using this heuristic guarantees monotonic convergence.

Example (Solving a Linear Equation System)
When we solve linear equations of the form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, in practice we solve $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0}$ approximately by finding $\boldsymbol{x}_*$ that minimizes the the squared error

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2 = (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^\top (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \tag{7.7}$$

if we use the Euclidean norm. The gradient of (7.7) with respect to $\boldsymbol{x}$ is

$$\nabla_{\boldsymbol{x}} = 2(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^\top \boldsymbol{A}\,. \tag{7.8}$$

We can use this gradient directly in a gradient descent algorithm. However for this particular special case, it turns out that there is an analytic solution, which can be found by setting the gradient to zero. We can see that this analytic solution is given by $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. We will see more on solving squared error problems in Chapter 9.

Remark. When applied to the solution of linear systems of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ gradient descent may converge slowly. The speed of convergence of gradient descent is dependent on the condition number $\kappa = \frac{\sigma(\boldsymbol{A})_{\max}}{\sigma(\boldsymbol{A})_{\min}}$, which is the ratio of the maximum to the minimum singular value of $\boldsymbol{A}$. The condition number essentially measures the ratio of the most curved direction versus the least curved direction, which corresponds to our imagery that poorly conditioned problems are long thin valleys: they are very curved in one direction, but very flat in the other. Instead of directly solving $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, one could instead solve $\boldsymbol{P}^{-1}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) = 0$, where P is called the preconditioner. The goal is to design $\boldsymbol{P}^{-1}$ such that $\boldsymbol{P}^{-1}\boldsymbol{A}$ has a better condition number, but at the same time $\boldsymbol{P}^{-1}$ is easy to compute. For further information on gradient descent, pre-conditioning and convergence we refer to (Boyd and Vandenberghe, 2004, Chapter 9). ♢

7.1.2 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. Continuing the ball analogy, the momentum term emulates the phenomenon of a heavy ball which is reluctant to change directions. The idea is to have a gradient update with memory to implement a moving average. The momentum-based method remembers the update $\Delta\boldsymbol{x}_i$ at each iteration i and determines the next update as a linear combination of the current and previous gradients

Goh (2017) wrote an intuitive blog post on gradient descent with momentum.

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i((\nabla f)(\boldsymbol{x}_i))^\top + \alpha\Delta\boldsymbol{x}_i \tag{7.9}$$

$$\Delta\boldsymbol{x}_i = \boldsymbol{x}_i - \boldsymbol{x}_{i-1} = -\gamma_{i-1}((\nabla f)(\boldsymbol{x}_{i-1}))^\top\,, \tag{7.10}$$

where $\alpha \in [0, 1]$. Sometimes we will only know the gradient approximately. In such cases the momentum term is useful since it averages out different noisy estimates of the gradient. One particularly useful way to obtain an approximate gradient is using a stochastic approximation, which we discuss next.

©2018 Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. To be published by Cambridge University Press.

7.1.3 Stochastic Gradient Descent

Computing the gradient can be very time consuming. However, often it is possible to find a "cheap" approximation of the gradient. Approximating the gradient is still useful as long as it points in roughly the same direction as the true gradient.

Stochastic gradient descent

Stochastic gradient descent (often shortened in 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.

In machine learning given $n = 1, \dots, N$ data points, we often consider objective functions which are the sum of the losses L_n incurred by each example n. In mathematical notation we have the form

$$L(\boldsymbol{\theta}) = \sum_{n=1}^{N} L_n(\boldsymbol{\theta}) \tag{7.11}$$

where $\boldsymbol{\theta}$ is the vector of parameters of interest, i.e., we want to find $\boldsymbol{\theta}$ that minimizes L. An example from regression (Chapter 9), is the negative log-likelihood, which is expressed as a sum over log-likelihoods of individual examples,

$$L(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}) \tag{7.12}$$

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

Standard gradient descent, as introduced previously, is a "batch" optimization method, i.e., optimization is performed using the full training set by updating the vector of parameters according to

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \gamma_i (\nabla L(\boldsymbol{\theta}_i))^\top = \boldsymbol{\theta}_i - \gamma_i \sum_{n=1}^{N} (\nabla L_n(\boldsymbol{\theta}_i))^\top \tag{7.13}$$

for a suitable stepsize parameter γ_i. Evaluating the sum-gradient may require expensive evaluations of the gradients from all individual functions L_n. When the training set is enormous and/or no simple formulas exist, evaluating the sums of gradients becomes very expensive.

Consider the term $\sum_{n=1}^{N} (\nabla L_n(\boldsymbol{\theta}_i))$ in (7.13) above: we can reduce the amount of computation by taking a sum over a smaller set of L_n. In contrast to batch gradient descent, which uses all L_n for $n = 1, \dots, N$, we randomly choose a subset of L_n for mini-batch gradient descent. In the extreme case, we randomly select only a single L_n to estimate the gradient.

Why should one consider using an approximate gradient? A major reason is practical implementation constraints, such as the size of CPU/GPU memory or limits on computational time. We can think of the size of the subset used to estimate the gradient in the same way that we thought of the size of a sample when estimating empirical means 6.4.1. In practice, it is good to keep the size of the mini-batch as large as possible. Large mini-batches reduce 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. However when we choose the mini-batch size, we need to make sure it fits into CPU/GPU memory. Typical mini-batch sizes are 64, 128, 256, 512, 1024, which depends on the way computer memory is laid out and accessed.

This often leads to more stable convergence since the gradient estimator is less noisy.

Remark. When the learning rate decreases at an appropriate rate, and subject to relatively mild assumptions, stochastic gradient descent converges almost surely to local minimum (Bottou, 1998). ♢

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.

Stochastic gradient descent is very effective in large-scale machine learning problems (Bottou et al., 2018), such as training deep neural networks on millions of images (Dean et al., 2012), topic models (Hoffman et al., 2013), reinforcement learning (Mnih et al., 2015) or training large-scale Gaussian process models (Hensman et al., 2013; Gal et al., 2014).

7.2 Constrained Optimization and Lagrange Multipliers

In the previous section, we considered the problem of solving for the minimum of a function

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{7.14}$$

where $f : \mathbb{R}^D \to \mathbb{R}$.

In this section we have additional constraints. That is for real valued functions $g_i : \mathbb{R}^D \to \mathbb{R}$ for $i = 1, \ldots, m$ we consider the constrained optimization problem

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \leqslant 0 \quad \text{for all} \quad i = 1, \ldots, m \end{aligned} \tag{7.15}$$

It is worth pointing out that the functions f and g_i could be non-convex in general, and we will consider the convex case in the next section.

One obvious, but not very practical, way of converting the constrained problem (7.15) into an unconstrained one is to use an indicator function

$$J(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \mathbf{1}(g_i(\boldsymbol{x})) \tag{7.16}$$

©2018 Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. To be published by Cambridge University Press.

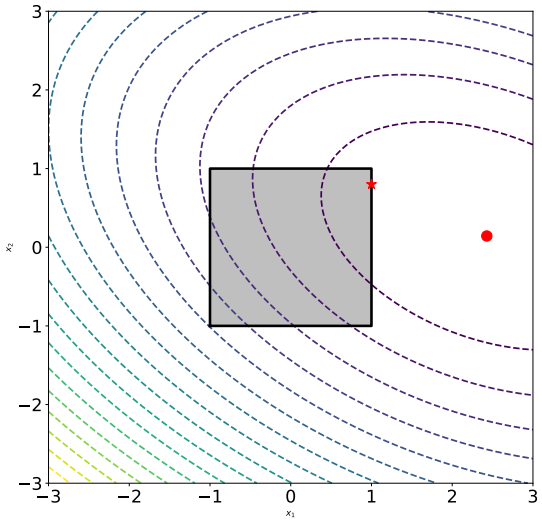

Figure 7.3 Illustration of constrained optimization. The unconstrained problem (indicated by the contour lines) has a minimum on the right side (indicated by the circle). The box constraints ($-1 \leqslant x \leqslant 1$ and $-1 \leqslant y \leqslant 1$) require that the optimal solution are within the box, resulting in an optimal value indicated by the star.

where $\mathbf{1}(z)$ is an infinite step function

$$\mathbf{1}(z) = \begin{cases} 0 & \text{if } z \leqslant 0 \\ \infty & \text{otherwise} \end{cases} . \tag{7.17}$$

This gives infinite penalty if the constraint is not satisfied, and hence would provide the same solution. However, this infinite step function is equally difficult to optimize. We can overcome this difficulty by introducing *Lagrange multipliers*. The idea of Lagrange multipliers is to replace the step function with a linear function.

Lagrange multipliers

We associate to problem (7.15) the *Lagrangian* by introducing the Lagrange multipliers λ_i corresponding to each inequality constraint respectively (Boyd and Vandenberghe, 2004, Chapter 4).

Lagrangian

$$\begin{aligned} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) &= f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x}) \\ &= f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top \boldsymbol{g}(\boldsymbol{x}) \end{aligned} \tag{7.18}$$

where in the last line we have a concatenated all constraints $g_i(\boldsymbol{x})$ into a vector $\boldsymbol{g}(\boldsymbol{x})$, and all the Lagrange multipliers into a vector $\boldsymbol{\lambda} \in \mathbb{R}^m$.

We now introduce the idea of Lagrangian duality. In general, duality in optimization is the idea of converting an optimization problem in one set of variables $\boldsymbol{x}$ (called the primal variables), into another optimization problem in a different set of variables $\boldsymbol{\lambda}$ (called the dual variables). We introduce two different approaches to duality: in this section we discuss

Lagrangian duality, and in Section 7.3.3 we discuss Legendre-Fenchel duality.

Theorem 7.1. *The problem in* (7.15)

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \leqslant 0 \quad \textit{for all} \quad i = 1, \dots, m \end{aligned}$$

is known as the primal problem, corresponding to the primal variables x. *The associated Lagrangian dual problem is given by* primal problem Lagrangian dual problem

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^m} \quad \mathcal{D}(\boldsymbol{\lambda}) \tag{7.19}$$

$$\text{subject to} \quad \boldsymbol{\lambda} \geqslant 0, \tag{7.20}$$

where $\boldsymbol{\lambda}$ *are the dual variables and* $\mathcal{D}(\boldsymbol{\lambda}) = \min_{\boldsymbol{x} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$.

Proof Recall that the difference between $J(\boldsymbol{x})$ in (7.16) and the Lagrangian in (7.18) is that we have relaxed the indicator function to a linear function. Therefore when $\lambda \geqslant 0$, the Lagrangian $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is a lower bound of $J(\boldsymbol{x})$. Hence the maximum of $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ with respect to $\boldsymbol{\lambda}$ is $J(\boldsymbol{x})$

$$J(\boldsymbol{x}) = \max_{\boldsymbol{\lambda} \geqslant 0} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}). \tag{7.21}$$

Recall that the original problem was minimising $J(\boldsymbol{x})$,

$$\min_{\boldsymbol{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geqslant \mathbf{0}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}). \tag{7.22}$$

By the minimax inequality (Boyd and Vandenberghe, 2004) it turns out that, for any function swapping the order of the minimum and maximum above results in a smaller value.

$$\min_{\boldsymbol{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geqslant \mathbf{0}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \geqslant \max_{\boldsymbol{\lambda} \geqslant \mathbf{0}} \min_{\boldsymbol{x} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \tag{7.23}$$

This is also known as *weak duality*. Note that the inner part of the right hand side is the dual objective function $\mathcal{D}(\boldsymbol{\lambda})$ and the theorem follows. □ weak duality

In contrast to the original optimization problem which has constraints, $\min_{\boldsymbol{x} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is an unconstrained optimization problem for a given value of $\boldsymbol{\lambda}$. If solving $\min_{\boldsymbol{x} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is easy, then the overall problem is easy to solve. The reason is that the outer problem (maximization over $\boldsymbol{\lambda}$) is a maximum over a set of affine functions, and hence is a concave function, even though $f(\cdot)$ and $g_i(\cdot)$ may be non-convex. The maximum of a concave function can be efficiently computed.

Assuming $f(\cdot)$ and $g_i(\cdot)$ are differentiable, we find the Lagrange dual problem by differentiating the Lagrangian with respect to $\boldsymbol{x}$ and setting the differential to zero and solving for the optimal value. We will discuss two concrete examples in Section 7.3.1 and 7.3.2, where $f(\cdot)$ and $g_i(\cdot)$ are convex.

©2018 Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. To be published by Cambridge University Press.

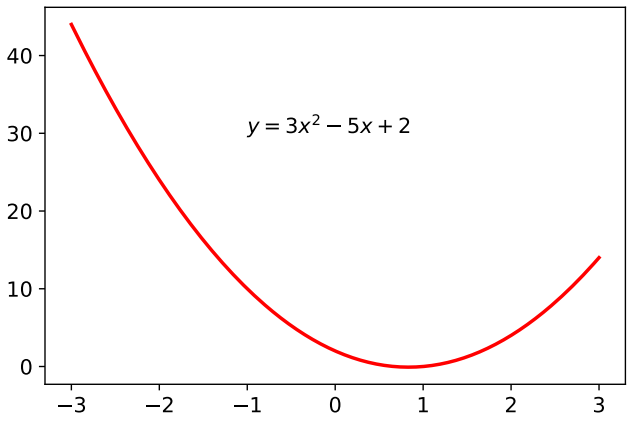

Figure 7.4 Example of a convex function.

Remark (Equality constraints). Consider (7.15) with additional equality constraints

$$\begin{aligned} &\min_{\boldsymbol{x}} f(\boldsymbol{x}) && (7.24)\\ &\text{subject to } g_i(\boldsymbol{x}) \leqslant 0 \quad \text{for all} \quad i = 1, \dots, m \\ &\qquad\qquad h_j(\boldsymbol{x}) = 0 \quad \text{for all} \quad j = 1, \dots, n \end{aligned}$$

We can model equality constraints by replacing them with two inequality constraints. That is for each equality constraint $h_j(\boldsymbol{x}) = 0$ we equivalently replace it by two constraints $h_j(\boldsymbol{x}) \leqslant 0$ and $h_j(\boldsymbol{x}) \geqslant 0$. It turns out that the resulting Lagrange multipliers are then unconstrained.

Therefore we constrain the Lagrange multipliers corresponding to the inequality constraints in (7.24) to be non-negative, and leave the Lagrange multipliers corresponding to the equality constraints unconstrained. ♢

7.3 Convex Optimization

We focus our attention of a particularly useful class of optimization problems, where we can guarantee global optimality. When $f(\cdot)$ is a convex function, and when the constraints involving $g(\cdot)$ and $h(\cdot)$ are convex sets, this is called a *convex optimization problem*. In this setting, we have *strong duality*: The optimal solution of the dual problem is the same as the optimal solution of the primal problem. The distinction between *convex functions* and *convex sets* are often not strictly presented in machine learning literature, but one can often infer the implied meaning from context.

convex optimization problem
strong duality
convex functions
convex sets

Convex functions are functions such that a straight line between any two points of the function lie above the function. Figure 7.1 shows a non-convex function and Figure 7.2 shows a convex function. Another convex function is shown in Figure 7.4.

Definition 7.2. A function $f : \mathbb{R}^D \to \mathbb{R}$ is *convex* if for all $\boldsymbol{x}, \boldsymbol{y}$ in the domain of f, and for any scalar θ with $0 \leqslant \theta \leqslant 1$, we have

$$f(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leqslant \theta f(\boldsymbol{x}) + (1-\theta)f(\boldsymbol{y}) \tag{7.25}$$

Technically, the domain of the function f must also be a convex set.

The constraints involving $g(\cdot)$ and $h(\cdot)$ above truncate functions at a scalar value, resulting in sets. Another relation between convex functions and convex sets is to consider the set obtained by "filling in" a convex function. A convex function is a bowl like object, and we imagine pouring water into it to fill it up. This resulting filled in set, called the epigraph of the convex function, is a convex set. Convex sets are sets such that a straight line connecting any two elements of the set lie inside the set. Figure 7.5 and Figure 7.6 illustrates convex and nonconvex sets respectively.

Figure 7.5 Example of a convex set

Figure 7.6 Example of a nonconvex set

Definition 7.3. A set C is *convex* if for any $x, y \in C$ and for any scalar θ with $0 \leqslant \theta \leqslant 1$, we have

$$\theta x + (1-\theta)y \in C \tag{7.26}$$

Remark. We can check that a function or set is convex from first principles by recalling the definitions. In practice we often rely on operations that preserve convexity to check that a particular function or set is convex. Although the details are vastly different, this is again the idea of closure that we introduced in Chapter 2 for vector spaces. ♢

Example

A nonnegative weighted sum of convex functions is convex. Observe that if f is a convex function, and $\alpha \geqslant 0$ is a nonnegative scalar, then the function αf is convex. We can see this by multiplying α to both sides of equation in Definition 7.2, and recalling that multiplying a nonnegative number does not change the inequality.

If f_1 and f_2 are convex functions, then we have by the definition

$$f_1(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leqslant \theta f_1(\boldsymbol{x}) + (1-\theta)f_1(\boldsymbol{y}) \tag{7.27}$$
$$f_2(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leqslant \theta f_2(\boldsymbol{x}) + (1-\theta)f_2(\boldsymbol{y}). \tag{7.28}$$

Summing up both sides gives us

$$\begin{aligned} &f_1(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) + f_2(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) \\ &\leqslant \theta f_1(\boldsymbol{x}) + (1-\theta)f_1(\boldsymbol{y}) + \theta f_2(\boldsymbol{x}) + (1-\theta)f_2(\boldsymbol{y}) \end{aligned} \tag{7.29}$$

where the right hand side can be rearranged to

$$\theta(f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})) + (1-\theta)(f_1(\boldsymbol{y}) + f_2(\boldsymbol{y})) \tag{7.30}$$

completing the proof that the sum of convex functions is convex.

Combining the two facts above, we see that $\alpha f_1(\boldsymbol{x}) + \beta f_2(\boldsymbol{x})$ is convex for $\alpha, \beta \geqslant 0$. This closure property can be extended using a similar argument for nonnegative weighted sums of more than two convex functions.

©2018 Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. To be published by Cambridge University Press.

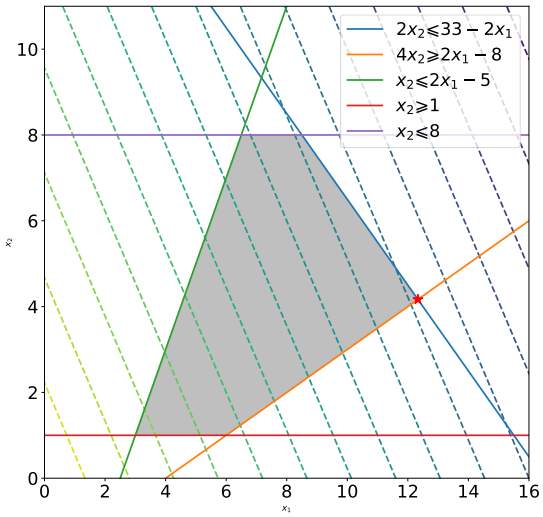

Figure 7.7 Illustration of a linear program. The unconstrained problem (indicated by the contour lines) has a minimum on the right side. The optimal value given the constraints are shown by the star.

Remark. The inequality defining convex functions, see 7.25, is sometimes called *Jensen's inequality*. In fact a whole class of inequalities for taking nonnegative weighted sums of convex functions are all called Jensen's inequality. ♢

Jensen's inequality

7.3.1 Linear Programming

Consider the special case when all the functions above are linear, that is

$$\min_{\boldsymbol{x} \in \mathbb{R}^d} \boldsymbol{c}^\top \boldsymbol{x} \tag{7.31}$$
$$\text{subject to } \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times d}$ and $\boldsymbol{b} \in \mathbb{R}^m$. This is known as a *linear program*. It has d variables and m linear constraints.

Linear programs are one of the most widely used approaches in industry.

Example

An example of a linear program is illustrated in Figure 7.7, which has two variables. The objective function is linear, resulting in linear contour lines. The constraint set in standard form is translated into the legend. The optimal value must lie in the shaded (feasible) region, and is indicated by the star.

$$\min_{\boldsymbol{x}\in\mathbb{R}^2} -\begin{bmatrix}5\\3\end{bmatrix}^\top \begin{bmatrix}x_1\\x_2\end{bmatrix} \tag{7.32}$$

$$\text{subject to } \begin{bmatrix}2 & 2\\2 & -4\\-2 & 1\\0 & -1\\0 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} \leqslant \begin{bmatrix}33\\8\\5\\-1\\8\end{bmatrix} \tag{7.33}$$

The Lagrangian is given by

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

where $\boldsymbol{\lambda}\in\mathbb{R}^m$ is the vector of non-negative Lagrange multipliers. It is easier to see what is going on by rearranging the terms corresponding to $\boldsymbol{x}$.

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

Taking the derivative of $\mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda})$ with respect to $\boldsymbol{x}$ and setting it to zero gives us

$$\boldsymbol{c}+\boldsymbol{A}^\top\boldsymbol{\lambda} = \boldsymbol{0}.$$

Therefore the dual Lagrangian is $\mathcal{D}(\boldsymbol{\lambda}) = -\boldsymbol{\lambda}^\top\boldsymbol{b}$. Recall we would like to maximize $\mathcal{D}(\boldsymbol{\lambda})$. This is traditionally presented as minimising the negative of it. In addition to the constraint due to the derivative of $\mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda})$ being zero, we also have the fact that $\boldsymbol{\lambda}\geqslant\boldsymbol{0}$, resulting in the following dual optimization problem

$$\begin{aligned}\min_{\boldsymbol{\lambda}\in\mathbb{R}^m} \boldsymbol{b}^\top\boldsymbol{\lambda} \\ \text{subject to } \boldsymbol{c}+\boldsymbol{A}^\top\boldsymbol{\lambda} = \boldsymbol{0} \\ \boldsymbol{\lambda}\geqslant\boldsymbol{0}.\end{aligned} \tag{7.34}$$

This is also a linear program, but with m variables. We have the choice of solving the primal (7.31) or the dual (7.34) program depending on whether m or d is larger. Recall that d is the number of variables and m is the number of constraints in the primal linear program.

7.3.2 Quadratic Programming

Consider when the objective function is a convex quadratic function, and the constraints are affine,

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{2}\boldsymbol{x}^\top\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^\top\boldsymbol{x} \tag{7.35}$$

©2018 Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. To be published by Cambridge University Press.

$$\text{subject to } \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times d}$, $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{c} \in \mathbb{R}^d$. The square symmetric matrix $\boldsymbol{Q} \in \mathbb{R}^{d\times d}$ is positive definite, and therefore the objective function is convex. This is known as a *quadratic program*. Observe that it has d variables and m linear constraints.

Example

An example of a quadratic program is illustrated in Figure 7.3, which has two variables. The objective function is quadratic with a positive semidefinite matrix $\boldsymbol{Q}$, resulting in elliptical contour lines. The optimal value must lie in the shaded (feasible) region, and is indicated by the star.

$$\min_{\boldsymbol{x}\in\mathbb{R}^2} \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^\top \begin{bmatrix}2 & 1\\1 & 4\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} + \begin{bmatrix}5\\3\end{bmatrix}^\top \begin{bmatrix}x_1\\x_2\end{bmatrix} \tag{7.36}$$

$$\text{subject to } \begin{bmatrix}1 & 0\\-1 & 0\\0 & 1\\0 & -1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} \leqslant \begin{bmatrix}1\\1\\1\\1\end{bmatrix} \tag{7.37}$$

The Lagrangian is given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda}) &= \frac{1}{2}\boldsymbol{x}^\top\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^\top\boldsymbol{x} + \boldsymbol{\lambda}^\top(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})\\ &= \frac{1}{2}\boldsymbol{x}^\top\boldsymbol{Q}\boldsymbol{x} + (\boldsymbol{c}+\boldsymbol{A}^\top\boldsymbol{\lambda})^\top\boldsymbol{x} - \boldsymbol{\lambda}^\top\boldsymbol{b}\end{aligned}$$

where again we have rearranged the terms. Taking the derivative of $\mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda})$ with respect to $\boldsymbol{x}$ and setting it to zero gives

$$\boldsymbol{Q}\boldsymbol{x} + (\boldsymbol{c}+\boldsymbol{A}^\top\boldsymbol{\lambda}) = 0$$

Assuming that $\boldsymbol{Q}$ is invertible, we get

$$\boldsymbol{x} = -\boldsymbol{Q}^{-1}(\boldsymbol{c}+\boldsymbol{A}^\top\boldsymbol{\lambda}) \tag{7.38}$$

Substituting (7.38) into the primal Lagrangian $\mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda})$ we get the dual Lagrangian

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

Therefore the dual optimization problem is given by

$$\min_{\boldsymbol{\lambda}\in\mathbb{R}^m} \frac{1}{2}(\boldsymbol{c}+\boldsymbol{A}^\top\boldsymbol{\lambda})\boldsymbol{Q}^{-1}(\boldsymbol{c}+\boldsymbol{A}^\top\boldsymbol{\lambda}) + \boldsymbol{\lambda}^\top\boldsymbol{b} \tag{7.39}$$

$$\text{subject to } \boldsymbol{\lambda} \geqslant 0. \tag{7.40}$$

We will see an application of Quadratic Programming in machine learning in Chapter 10.

7.3.3 Legendre-Fenchel Transform and Convex Conjugate

Let us revisit the idea of duality, which we saw in Section 7.2, without considering constraints. One useful fact about convex sets is that a convex set can be equivalently described by its supporting hyperplanes. A hyperplane is called a *supporting hyperplane* of a convex set if it intersects the convex set and the convex set is contained on just one side of it. Recall that for a convex function, we can fill it up to obtain the epigraph which is a convex set. Therefore we can also describe convex functions in terms of their supporting hyperplanes. Furthermore observe that the supporting hyperplane just touches the convex function, and is in fact the tangent to the function at that point. And recall that the tangent of a function $f(\boldsymbol{x})$ at a given point $\boldsymbol{x}_0$ is the evaluation of the gradient of that function at that point $\frac{d^2 f(\boldsymbol{x})}{d\boldsymbol{x}^2}\Big|_{\boldsymbol{x}=\boldsymbol{x}_0}$. In summary, because convex sets can be equivalently described by its supporting hyperplanes, convex functions can be equivalently described by a function of their gradient. The *Legendre transform* formalizes this concept .

supporting hyperplane

Legendre transform

We begin with the most general definition which unfortunately has a counterintuitive form, and look at special cases to try to relate the definition to the intuition above. The *Legendre-Fenchel transform* is a transformation (in the sense of a Fourier transform) from a convex differentiable function $f(\boldsymbol{x})$ to a function that depends on the tangents $s(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$. It is worth stressing that this is a transformation of the function $f(\cdot)$ and not the variable $\boldsymbol{x}$ or the function evaluated at a value. The Legendre-Fenchel transform is also known as the convex conjugate (for reasons we will see soon) and is closely related to duality.

Physics students are often introduced to the Legendre transform as relating the Lagrangian and the Hamiltonian in classical mechanics.

Legendre-Fenchel transform

Definition 7.4. The *convex conjugate* of a function $f : \mathbb{R}^D \to \mathbb{R}$ is a function f^* defined by

convex conjugate

$$f^*(\boldsymbol{s}) = \sup_{\boldsymbol{x}\in\mathbb{R}^D} \langle \boldsymbol{s}, \boldsymbol{x}\rangle - f(\boldsymbol{x}) \tag{7.41}$$

Note that the convex conjugate definition above does not need the function f to be convex nor differentiable. In the definition above, we have used a general inner product (Section 3.2) but in the rest of this section we will consider the standard dot product between finite dimensional vectors ($\langle \boldsymbol{s}, \boldsymbol{x}\rangle = \boldsymbol{s}^\top \boldsymbol{x}$) to avoid too many technical details.

To understand the above definition in a geometric fashion, consider an nice simple one dimensional convex and differentiable function, for example $f(x) = x^2$. Note that since we are looking at a one dimensional problem, hyperplanes reduce to a line. Consider a line $y = sx + c$. Recall that we are able to describe convex functions by their supporting hyper-

This derivation is easiest to understand by drawing the reasoning as it progresses.

©2018 Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. To be published by Cambridge University Press.

planes, so let us try to describe this function $f(x)$ by its supporting lines. Fix the gradient of the line $s \in \mathbb{R}$ and for each point $(x_0, f(x_0))$ on the graph of f, find the minimum value of c such that the line still intersects $(x_0, f(x_0))$. Note that the minimum value of c is the place where a line with slope s "just touches" the function $f(x) = x^2$. The line passing through $(x_0, f(x_0))$ with gradient s is given by

$$y - f(x_0) = s(x - x_0) \tag{7.42}$$

The y-intercept of this line is $-sx_0 + f(x_0)$. The minimum of c for which $y = sx + c$ intersects with the graph of f is therefore

$$\inf_{x_0} -sx_0 + f(x_0). \tag{7.43}$$

The convex conjugate above is by convention defined to be the negative of this. The reasoning in this paragraph did not rely on the fact that we chose a one dimensional convex and differentiable function, and holds for $f : \mathbb{R}^D \to \mathbb{R}$ which is nonconvex and non differentiable.

The classical Legendre transform is defined on convex differentiable functions in $\mathbb{R}^D$

However convex differentiable functions such as the example $f(x) = x^2$ is a nice special case, where there is no need for the supremum, and there is a one to one correspondence between a function and its Legendre transform. For a convex differentiable function, we know that at x_0 the tangent touches $f(x_0)$, therefore

$$f(x_0) = sx_0 + c \tag{7.44}$$

Rearranging to get an expression for $-c$

$$-c = sx_0 - f(x_0). \tag{7.45}$$

Note that $-c$ changes with x_0 and therefore with s, which is why we can think of it as a function of s, which we call $f^*(s)$.

$$f^*(s) = sx - f(x). \tag{7.46}$$

The conjugate function has nice properties, for example for convex functions, applying the Legendre transform again gets us back to the original function. In the same way that the slope of $f(x)$ is s, the slope of $f^*(s)$ is x. The following two examples show common uses of convex conjugates in machine learning.

Example

To illustrate the application of convex conjugates, consider the quadratic function based on a positive definite matrix $\boldsymbol{K} \in \mathbb{R}^{n \times n}$. We denote the primal variable to be $\boldsymbol{y} \in \mathbb{R}^n$ and the dual variable to be $\boldsymbol{\alpha} \in \mathbb{R}^n$.

$$f(\boldsymbol{y}) = \frac{\lambda}{2} \boldsymbol{y}^\top \boldsymbol{K}^{-1} \boldsymbol{y} \tag{7.47}$$

Applying Definition 7.4, we obtain the function

$$f^*(\boldsymbol{\alpha}) = \sup_{\boldsymbol{y}\in\mathbb{R}^n} \langle \boldsymbol{y}, \boldsymbol{\alpha}\rangle - \frac{\lambda}{2}\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y}. \tag{7.48}$$

Observe that the function is differentiable, and hence we can find the maximum by taking the derivative and with respect to $\boldsymbol{y}$ setting it to zero.

$$\frac{\partial\left[\langle \boldsymbol{y}, \boldsymbol{\alpha}\rangle - \frac{\lambda}{2}\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y}\right]}{\partial \boldsymbol{y}} = (\boldsymbol{\alpha} - \lambda \boldsymbol{K}^{-1}\boldsymbol{y})^\top \tag{7.49}$$

and hence when the gradient is zero we have $\boldsymbol{y} = \frac{1}{\lambda}\boldsymbol{K}\boldsymbol{\alpha}$. Substituting into (7.48) yields

$$f^*(\boldsymbol{\alpha}) = \frac{1}{\lambda}\boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{\alpha} - \frac{\lambda}{2}\left(\frac{1}{\lambda}\boldsymbol{K}\boldsymbol{\alpha}\right)^\top \boldsymbol{K}^{-1}\left(\frac{1}{\lambda}\boldsymbol{K}\boldsymbol{\alpha}\right) \tag{7.50}$$

$$= \frac{1}{2\lambda}\boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{\alpha}\,. \tag{7.51}$$

Example

In machine learning we often use sums of functions, for example the objective function of the training set includes a sum of the losses for each example in the training set. In the following, we derive the convex conjugate of a sum of losses $\ell(t)$, where $\ell : \mathbb{R} \to \mathbb{R}$. This also illustrates the application of the convex conjugate to the vector case. Let $\mathcal{L}(\boldsymbol{t}) = \sum_{i=1}^n \ell_i(t_i)$,

$$\mathcal{L}^*(\boldsymbol{z}) = \sup_{\boldsymbol{t}\in\mathbb{R}^n} \langle \boldsymbol{z}, \boldsymbol{t}\rangle - \sum_{i=1}^n \ell_i(t_i) \tag{7.52}$$

$$= \sup_{\boldsymbol{t}\in\mathbb{R}^n} \sum_{i=1}^n z_i t_i - \ell_i(t_i) \qquad \text{definition of dot product} \tag{7.53}$$

$$= \sum_{i=1}^n \sup_{\boldsymbol{t}\in\mathbb{R}^n} z_i t_i - \ell_i(t_i) \tag{7.54}$$

$$= \sum_{i=1}^n \ell_i^*(z_i) \qquad \text{definition of conjugate} \tag{7.55}$$

Recall that in Section 7.2 we derived a dual optimization problem using Lagrange multipliers. Furthermore for convex optimization problems we have strong duality, that is the solutions of the primal and dual problem match. The Fenchel-Legendre transform described here also can be used to derive a dual optimization problem. Furthermore then the function is

©2018 Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. To be published by Cambridge University Press.

convex and differentiable, the supremum is unique. To further investigate the relation between these two approaches, let us consider a linear equality constrained convex optimization problem.

Example

Let $f(\boldsymbol{y})$ and $g(\boldsymbol{x})$ be convex functions, and $\boldsymbol{A}$ a real matrix of appropriate dimensions such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$. Then

$$\min_{\boldsymbol{x}} f(\boldsymbol{A}\boldsymbol{x}) + g(\boldsymbol{x}) = \min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{y}} f(\boldsymbol{y}) + g(\boldsymbol{x}). \tag{7.56}$$

By introducing the Lagrange multiplier $\boldsymbol{u}$ for the constraints $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$,

$$\min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{y}} f(\boldsymbol{y}) + g(\boldsymbol{x}) = \min_{\boldsymbol{x},\boldsymbol{y}} \max_{\boldsymbol{u}} f(\boldsymbol{y}) + g(\boldsymbol{x}) + (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})^\top \boldsymbol{u} \tag{7.57}$$

$$= \max_{\boldsymbol{u}} \min_{\boldsymbol{x},\boldsymbol{y}} f(\boldsymbol{y}) + g(\boldsymbol{x}) + (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})^\top \boldsymbol{u} \tag{7.58}$$

where the last step of swapping max and min is due to the fact that $f(\boldsymbol{y})$ and $g(\boldsymbol{x})$ are convex functions. By splitting up the dot product term and collecting $\boldsymbol{x}$ and $\boldsymbol{y}$,

$$\max_{\boldsymbol{u}} \min_{\boldsymbol{x},\boldsymbol{y}} f(\boldsymbol{y}) + g(\boldsymbol{x}) + (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})^\top \boldsymbol{u} \tag{7.59}$$

$$= \max_{\boldsymbol{u}} \left[\min_{\boldsymbol{y}} -\boldsymbol{y}^\top \boldsymbol{u} + f(\boldsymbol{y})\right] + \left[\min_{\boldsymbol{x}} (\boldsymbol{A}\boldsymbol{x})^\top \boldsymbol{u} + g(\boldsymbol{x})\right] \tag{7.60}$$

$$= \max_{\boldsymbol{u}} \left[\min_{\boldsymbol{y}} -\boldsymbol{y}^\top \boldsymbol{u} + f(\boldsymbol{y})\right] + \left[\min_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{u} + g(\boldsymbol{x})\right] \tag{7.61}$$

For general inner products, $\boldsymbol{A}^\top$ is replaced by the adjoint $\boldsymbol{A}^*$.

Recall the convex conjugate (Definition 7.4) and the fact that dot products are symmetric,

$$\max_{\boldsymbol{u}} \left[\min_{\boldsymbol{y}} -\boldsymbol{y}^\top \boldsymbol{u} + f(\boldsymbol{y})\right] + \left[\min_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{u} + g(\boldsymbol{x})\right] \tag{7.62}$$

$$= \max_{\boldsymbol{u}} -f^*(\boldsymbol{u}) - g^*(-\boldsymbol{A}^\top \boldsymbol{u}). \tag{7.63}$$

Therefore we have shown that

$$\min_{\boldsymbol{x}} f(\boldsymbol{A}\boldsymbol{x}) + g(\boldsymbol{x}) = \max_{\boldsymbol{u}} -f^*(\boldsymbol{u}) - g^*(-\boldsymbol{A}^\top \boldsymbol{u}). \tag{7.64}$$

The Legendre-Fenchel conjugate turns out to be quite useful for machine learning problems that can be expressed as convex optimization problems. In particular for convex loss functions that apply independently to each example, the conjugate loss is a convenient way to derive a dual problem. We will see such an example in Chapter 10.

7.4 Further Reading

Continuous optimization is an active area of research, and we do not try to provide a comprehensive account of recent advances.

From a gradient descent perspective, there are two major weaknesses which each have their own set of literature. The first challenge is the fact that gradient descent is a first order algorithm, and does not use information about the curvature of the surface. When there are long valleys, the gradient points perpendicularly to the direction of interest. Conjugate gradient methods avoid the issues faced by gradient descent by taking previous directions into account (Shewchuk, 1994). Second order methods such as Newton methods use the Hessian to provide information about the curvature. Quasi-Newton methods such as L-BFGS try to use cheaper computational methods to approximate the Hessian (Nocedal and Wright, 2006).

The second challenge are non-differentiable functions. Gradient methods are not well defined when there are kinks in the function. In these cases, *subgradient methods* can be used (Shor, 1985). For further information and algorithms for optimizing non-differentiable functions, we refer to the book by Bertsekas (1999).

Modern applications of machine learning often mean that the size of datasets prohibit the use of batch gradient descent, and hence stochastic gradient descent is the current workhorse of large scale machine learning methods. Recent surveys of the literature include (Hazan, 2015; Bottou et al., 2018).

For duality and convex optimization, the book by Boyd and Vandenberghe (Boyd and Vandenberghe, 2004) includes lectures and slides online. A more mathematical treatment is provided by Bertsekas (2009). Convex optimization is based upon convex analysis, and the reader interested in more foundational results about convex functions is referred to Hiriart-Urruty and Lemaréchal (2001); Rockafellar (1970); Borwein and Lewis (2006). Legendre-Fenchel transforms are also covered in the above books on convex analysis, but more beginner friendly presentations are available at Zia et al. (2009); Gonçalves (2014).

Exercises

7.1 Consider the univariate function

$$f(x) = x^3 + 2x^2 + 5x - 3.$$

Find its stationary points and indicate whether they are maximum, minimum or saddle points.

7.2 Consider the update equation for stochastic gradient descent (Equation (7.13)). Write down the update when we use a mini-batch size of one.

7.3 Express the following optimization problem as a standard linear program in

©2018 Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. To be published by Cambridge University Press.

matrix notation

$$\max_{\boldsymbol{x}\in\mathbb{R}^2,\xi\in\mathbb{R}} \boldsymbol{p}^\top \boldsymbol{x} + \xi$$

4039 subject to the constraints that $\xi \geqslant 0$, $x_0 \leqslant 0$ and $x_1 \leqslant 3$.

7.4 The hinge loss (which is the loss used by the Support Vector Machine) is given by

$$L(\alpha) = \max\{0, 1-\alpha\}$$

If we are interested in applying gradient methods such as L-BFGS, and do not want to resort to subgradient methods, we need to smooth the kink in the hinge loss. Compute the convex conjugate of the hinge loss $L^*(\beta)$ where β is the dual variable. Add a ℓ_2 proximal term, and compute the conjugate of the resulting function

$$L^*(\beta) + \frac{\gamma}{2}\beta^2$$

4040 where γ is a given hyperparameter.