Convex Optimization Overview

Zico Kolter

October 19, 2007

1 Introduction

Many situations arise in machine learning where we would like to ***optimize*** the value of some function. That is, given a function $f : \mathbb{R}^n \to \mathbb{R}$, we want to find $x \in \mathbb{R}^n$ that minimizes (or maximizes) $f(x)$. We have already seen several examples of optimization problems in class: least-squares, logistic regression, and support vector machines can all be framed as optimization problems.

It turns out that in the general case, finding the global optimum of a function can be a very difficult task. However, for a special class of optimization problems, known as ***convex optimization problems***, we can efficiently find the global solution in many cases. Here, "efficiently" has both practical and theoretical connotations: it means that we can solve many real-world problems in a reasonable amount of time, and it means that theoretically we can solve problems in time that depends only *polynomially* on the problem size.

The goal of these section notes and the accompanying lecture is to give a very brief overview of the field of convex optimization. Much of the material here (including some of the figures) is heavily based on the book *Convex Optimization* [1] by Stephen Boyd and Lieven Vandenberghe (available for free online), and EE364, a class taught here at Stanford by Stephen Boyd. If you are interested in pursuing convex optimization further, these are both excellent resources.

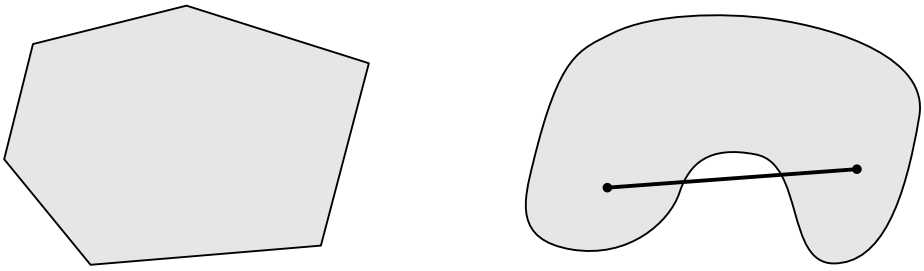
2 Convex Sets

We begin our look at convex optimization with the notion of a ***convex set***.

Definition 2.1 *A set C is convex if, for any $x, y \in C$ and $\theta \in \mathbb{R}$ with $0 \leq \theta \leq 1$,*

$$\theta x + (1 - \theta)y \in C.$$

Intuitively, this means that if we take any two elements in C, and draw a line segment between these two elements, then every point on that line segment also belongs to C. Figure 1 shows an example of one convex and one non-convex set. The point $\theta x + (1 - \theta)y$ is called a ***convex combination*** of the points x and y.



(a) (b)

Figure 1: Examples of a convex set (a) and a non-convex set (b).

2.1 Examples

- **All of $\mathbb{R}^n$.** It should be fairly obvious that given any $x, y \in \mathbb{R}^n$, $\theta x + (1-\theta)y \in \mathbb{R}^n$.

- **The non-negative orthant, $\mathbb{R}^n_+$.** The non-negative orthant consists of all vectors in $\mathbb{R}^n$ whose elements are all non-negative: $\mathbb{R}^n_+ = \{x : x_i \geq 0 \ \ \forall i = 1, \ldots, n\}$. To show that this is a convex set, simply note that given any $x, y \in \mathbb{R}^n_+$ and $0 \leq \theta \leq 1$,

$$(\theta x + (1-\theta)y)_i = \theta x_i + (1-\theta)y_i \geq 0 \ \ \forall i.$$

- **Norm balls.** Let $\|\cdot\|$ be some norm on $\mathbb{R}^n$ (e.g., the Euclidean norm, $\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$). Then the set $\{x : \|x\| \leq 1\}$ is a convex set. To see this, suppose $x, y \in \mathbb{R}^n$, with $\|x\| \leq 1, \|y\| \leq 1$, and $0 \leq \theta \leq 1$. Then

$$\|\theta x + (1-\theta)y\| \leq \|\theta x\| + \|(1-\theta)y\| = \theta\|x\| + (1-\theta)\|y\| \leq 1$$

  where we used the triangle inequality and the positive homogeneity of norms.

- **Affine subspaces and polyhedra.** Given a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^m$, an affine subspace is the set $\{x \in \mathbb{R}^n : Ax = b\}$ (note that this could possibly be empty if b is not in the range of A). Similarly, a polyhedron is the (again, possibly empty) set $\{x \in \mathbb{R}^n : Ax \preceq b\}$, where '$\preceq$' here denotes componentwise inequality (i.e., all the entries of Ax are less than or equal to their corresponding element in b).[1] To prove this, first consider $x, y \in \mathbb{R}^n$ such that $Ax = Ay = b$. Then for $0 \leq \theta \leq 1$,

$$A(\theta x + (1-\theta)y) = \theta Ax + (1-\theta)Ay = \theta b + (1-\theta)b = b.$$

  Similarly, for $x, y \in \mathbb{R}^n$ that satisfy $Ax \leq b$ and $Ay \leq b$ and $0 \leq \theta \leq 1$,

$$A(\theta x + (1-\theta)y) = \theta Ax + (1-\theta)Ay \leq \theta b + (1-\theta)b = b.$$

[1] Similarly, for two vectors $x, y \in \mathbb{R}^n$, $x \succeq y$ denotes that each element of X is greater than or equal to the corresponding element in b. Note that sometimes '$\leq$' and '$\geq$' are used in place of '$\preceq$' and '$\succeq$'; the meaning must be determined contextually (i.e., both sides of the inequality will be vectors).

- **Intersections of convex sets.** Suppose $C_1, C_2, \ldots, C_k$ are convex sets. Then their intersection
$$\bigcap_{i=1}^{k} C_i = \{x : x \in C_i \ \ \forall i = 1, \ldots, k\}$$
is also a convex set. To see this, consider $x, y \in \bigcap_{i=1}^{k} C_i$ and $0 \leq \theta \leq 1$. Then,
$$\theta x + (1-\theta)y \in C_i \ \ \forall i = 1, \ldots, k$$
by the definition of a convex set. Therefore
$$\theta x + (1-\theta)y \in \bigcap_{i=1}^{k} C_i.$$
Note, however, that the *union* of convex sets in general will not be convex.

- **Positive semidefinite matrices.** The set of all symmetric positive semidefinite matrices, often times called the *positive semidefinite cone* and denoted $\mathbb{S}_+^n$, is a convex set (in general, $\mathbb{S}^n \subset \mathbb{R}^{n \times n}$ denotes the set of symmetric $n \times n$ matrices). Recall that a matrix $A \in \mathbb{R}^{n \times n}$ is symmetric positive semidefinite if and only if $A = A^T$ and for all $x \in \mathbb{R}^n$, $x^T A x \geq 0$. Now consider two symmetric positive semidefinite matrices $A, B \in \mathbb{S}_+^n$ and $0 \leq \theta \leq 1$. Then for any $x \in \mathbb{R}^n$,
$$x^T(\theta A + (1-\theta)B)x = \theta x^T A x + (1-\theta) x^T B x \geq 0.$$
The same logic can be used to show that the sets of all positive definite, negative definite, and negative semidefinite matrices are each also convex.

3 Convex Functions

A central element in convex optimization is the notion of a ***convex function***.

Definition 3.1 *A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if its domain (denoted $\mathcal{D}(f)$) is a convex set, and if, for all $x, y \in \mathcal{D}(f)$ and $\theta \in \mathbb{R}$, $0 \leq \theta \leq 1$,*
$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y).$$

Intuitively, the way to think about this definition is that if we pick any two points on the graph of a convex function and draw a straight line between then, then the portion of the function between these two points will lie below this straight line. This situation is pictured in Figure 2.[2]

We say a function is ***strictly convex*** if Definition 3.1 holds with strict inequality for $x \neq y$ and $0 < \theta < 1$. We say that f is ***concave*** if $-f$ is convex, and likewise that f is ***strictly concave*** if $-f$ is strictly convex.

[2]Don't worry too much about the requirement that the domain of f be a convex set. This is just a technicality to ensure that $f(\theta x + (1-\theta)y)$ is actually defined (if $\mathcal{D}(f)$ were not convex, then it could be that $f(\theta x + (1-\theta)y)$ is undefined even though $x, y \in \mathcal{D}(f)$).

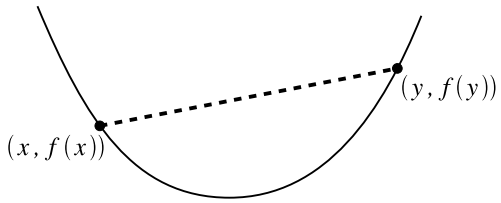


Figure 2: Graph of a convex function. By the definition of convex functions, the line connecting two points on the graph must lie above the function.

3.1 First Order Condition for Convexity

Suppose a function $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable (i.e., the gradient[3] $\nabla_x f(x)$ exists at all points x in the domain of f). Then f is convex if and only if $\mathcal{D}(f)$ is a convex set and for all $x, y \in \mathcal{D}(f)$,

$$f(y) \geq f(x) + \nabla_x f(x)^T (y - x).$$

The function $f(x) + \nabla_x f(x)^T (y - x)$ is called the ***first-order approximation*** to the function f at the point x. Intuitively, this can be thought of as approximating f with its tangent line at the point x. The first order condition for convexity says that f is convex if and only if the tangent line is a global underestimator of the function f. In other words, if we take our function and draw a tangent line at any point, then every point on this line will lie below the corresponding point on f.

Similar to the definition of convexity, f will be strictly convex if this holds with strict inequality, concave if the inequality is reversed, and strictly concave if the reverse inequality is strict.

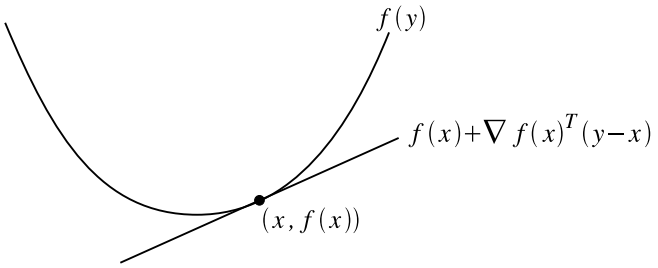


Figure 3: Illustration of the first-order condition for convexity.

[3]Recall that the gradient is defined as $\nabla_x f(x) \in \mathbb{R}^n, (\nabla_x f(x))_i = \frac{\partial f(x)}{\partial x_i}$. For a review on gradients and Hessians, see the previous section notes on linear algebra.

3.2 Second Order Condition for Convexity

Suppose a function $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable (i.e., the Hessian[4] $\nabla_x^2 f(x)$ is defined for all points x in the domain of f). Then f is convex if and only if $\mathcal{D}(f)$ is a convex set and its Hessian is positive semidefinite: i.e., for any $x \in \mathcal{D}(f)$,

$$\nabla_x^2 f(x) \succeq 0.$$

Here, the notation '$\succeq$' when used in conjunction with matrices refers to positive semidefiniteness, rather than componentwise inequality. [5] In one dimension, this is equivalent to the condition that the second derivative $f''(x)$ always be positive (i.e., the function always has positive curvature).

Again analogous to both the definition and first order conditions for convexity, f is strictly convex if its Hessian is positive definite, concave if the Hessian is negative semidefinite, and strictly concave if the Hessian is negative definite.

3.3 Jensen's Inequality

Suppose we start with the inequality in the basic definition of a convex function

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) \quad \text{for} \quad 0 \leq \theta \leq 1.$$

Using induction, this can be fairly easily extended to convex combinations of more than one point,

$$f\left(\sum_{i=1}^{k} \theta_i x_i\right) \leq \sum_{i=1}^{k} \theta_i f(x_i) \quad \text{for} \quad \sum_{i=1}^{k} \theta_i = 1,\ \theta_i \geq 0\ \ \forall i.$$

In fact, this can also be extended to infinite sums or integrals. In the latter case, the inequality can be written as

$$f\left(\int p(x)x dx\right) \leq \int p(x)f(x)dx \quad \text{for} \quad \int p(x)dx = 1,\ p(x) \geq 0\ \ \forall x.$$

Because $p(x)$ integrates to 1, it is common to consider it as a probability density, in which case the previous equation can be written in terms of expectations,

$$f(\mathbf{E}[x]) \leq \mathbf{E}[f(x)].$$

This last inequality is known as *Jensen's inequality*, and it will come up later in class.[6]

[4] Recall the Hessian is defined as $\nabla_x^2 f(x) \in \mathbb{R}^{n \times n}, (\nabla_x^2 f(x))_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$

[5] Similarly, for a symmetric matrix $X \in \mathbb{S}^n$, $X \preceq 0$ denotes that X is negative semidefinite. As with vector inequalities, '$\leq$' and '$\geq$' are sometimes used in place of '$\preceq$' and '$\succeq$'. Despite their notational similarity to vector inequalities, these concepts are very different; in particular, $X \succeq 0$ does not imply that $X_{ij} \geq 0$ for all i and j.

[6] In fact, all four of these equations are sometimes referred to as Jensen's inequality, due to the fact that they are all equivalent. However, for this class we will use the term to refer specifically to the last inequality presented here.

3.4 Sublevel Sets

Convex functions give rise to a particularly important type of convex set called an ***α-sublevel set***. Given a convex function $f : \mathbb{R}^n \to R$ and a real number $\alpha \in \mathbb{R}$, the α-sublevel set is defined as

$$\{x \in \mathcal{D}(f) : f(x) \leq \alpha\}.$$

In other words, the α-sublevel set is the set of all points x such that $f(x) \leq \alpha$.

To show that this is a convex set, consider any $x, y \in \mathcal{D}(f)$ such that $f(x) \leq \alpha$ and $f(y) \leq \alpha$. Then

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) \leq \theta\alpha + (1-\theta)\alpha = \alpha.$$

3.5 Examples

We begin with a few simple examples of convex functions of one variable, then move on to multivariate functions.

- **Exponential.** Let $f : \mathbb{R} \to \mathbb{R}$, $f(x) = e^{ax}$ for any $a \in \mathbb{R}$. To show f is convex, we can simply take the second derivative $f''(x) = a^2 e^{ax}$, which is positive for all x.

- **Negative logarithm.** Let $f : \mathbb{R} \to \mathbb{R}$, $f(x) = -\log x$ with domain $\mathcal{D}(f) = \mathbb{R}_{++}$ (here, $\mathbb{R}_{++}$ denotes the set of strictly positive real numbers, $\{x : x > 0\}$). Then $f''(x) = 1/x^2 > 0$ for all x.

- **Affine functions.** Let $f : \mathbb{R}^n \to \mathbb{R}$, $f(x) = b^T x + c$ for some $b \in \mathbb{R}^n$, $c \in \mathbb{R}$. In this case the Hessian, $\nabla_x^2 f(x) = 0$ for all x. Because the zero matrix is both positive semidefinite and negative semidefinite, f is both convex and concave. In fact, affine functions of this form are the *only* functions that are both convex and concave.

- **Quadratic functions.** Let $f : \mathbb{R}^n \to \mathbb{R}$, $f(x) = \frac{1}{2}x^T A x + b^T x + c$ for a symmetric matrix $A \in \mathbb{S}^n$, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. In our previous section notes on linear algebra, we showed the Hessian for this function is given by

  $$\nabla_x^2 f(x) = A.$$

  Therefore, the convexity or non-convexity of f is determined entirely by whether or not A is positive semidefinite: if A is positive semidefinite then the function is convex (and analogously for strictly convex, concave, strictly concave). If A is indefinite then f is neither convex nor concave.

  Note that the squared Euclidean norm $f(x) = \|x\|_2^2 = x^T x$ is a special case of quadratic functions where $A = I$, $b = 0$, $c = 0$, so it is therefore a strictly convex function.

- **Norms.** Let $f : \mathbb{R}^n \to \mathbb{R}$ be some norm on $\mathbb{R}^n$. Then by the triangle inequality and positive homogeneity of norms, for $x, y \in \mathbb{R}^n$, $0 \leq \theta \leq 1$,

$$f(\theta x + (1-\theta)y) \leq f(\theta x) + f((1-\theta)y) = \theta f(x) + (1-\theta)f(y).$$

  This is an example of a convex function where it is *not* possible to prove convexity based on the second or first order conditions, because norms are not generally differentiable everywhere (e.g., the 1-norm, $||x||_1 = \sum_{i=1}^n |x_i|$, is non-differentiable at all points where any x_i is equal to zero).

- **Nonnegative weighted sums of convex functions.** Let $f_1, f_2, \ldots, f_k$ be convex functions and $w_1, w_2, \ldots, w_k$ be nonnegative real numbers. Then

$$f(x) = \sum_{i=1}^k w_i f_i(x)$$

  is a convex function, since

$$\begin{aligned} f(\theta x + (1-\theta)y) &= \sum_{i=1}^k w_i f_i(\theta x + (1-\theta)y) \\ &\leq \sum_{i=1}^k w_i(\theta f_i(x) + (1-\theta)f_i(y)) \\ &= \theta \sum_{i=1}^k w_i f_i(x) + (1-\theta)\sum_{i=1}^k w_i f_i(y) \\ &= \theta f(x) + (1-\theta)f(x). \end{aligned}$$

4 Convex Optimization Problems

Armed with the definitions of convex functions and sets, we are now equipped to consider ***convex optimization problems***. Formally, a convex optimization problem in an optimization problem of the form

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in C \end{array}$$

where f is a convex function, C is a convex set, and x is the optimization variable. However, since this can be a little bit vague, we often write it often written as

$$\begin{array}{lll} \text{minimize} & f(x) & \\ \text{subject to} & g_i(x) \leq 0, & i = 1, \ldots, m \\ & h_i(x) = 0, & i = 1, \ldots, p \end{array}$$

where f is a convex function, g_i are convex functions, and h_i are affine functions, and x is the optimization variable.

Is it imporant to note the direction of these inequalities: a convex function g_i must be *less* than zero. This is because the 0-sublevel set of g_i is a convex set, so the feasible region, which is the intersection of many convex sets, is also convex (recall that affine subspaces are convex sets as well). If we were to require that $g_i \geq 0$ for some convex g_i, the feasible region would no longer be a convex set, and the algorithms we apply for solving these problems would not longer be guaranteed to find the global optimum. Also notice that only affine functions are allowed to be equality constraints. Intuitively, you can think of this as being due to the fact that an equality constraint is equivalent to the two inequalities $h_i \leq 0$ and $h_i \geq 0$. However, these will both be valid constraints if and only if h_i is both convex and concave, i.e., h_i must be affine.

The ***optimal value*** of an optimization problem is denoted $p^\star$ (or sometimes $f^\star$) and is equal to the minimum possible value of the objective function in the feasible region[7]

$$p^\star = \min\{f(x) : g_i(x) \leq 0, i = 1, \ldots, m, h_i(x) = 0, i = 1, \ldots, p\}.$$

We allow $p^\star$ to take on the values $+\infty$ and $-\infty$ when the problem is either *infeasible* (the feasible region is empty) or *unbounded below* (there exists feasible points such that $f(x) \to -\infty$), respectively. We say that $x^\star$ is an ***optimal point*** if $f(x^\star) = p^\star$. Note that there can be more than one optimal point, even when the optimal value is finite.

4.1 Global Optimality in Convex Problems

Before stating the result of global optimality in convex problems, let us formally define the concepts of local optima and global optima. Intuitively, a feasible point is called ***locally optimal*** if there are no "nearby" feasible points that have a lower objective value. Similarly, a feasible point is called ***globally optimal*** if there are no feasible points at all that have a lower objective value. To formalize this a little bit more, we give the following two definitions.

Definition 4.1 *A point x is locally optimal if it is feasible (i.e., it satisfies the constraints of the optimization problem) and if there exists some $R > 0$ such that all feasible points z with $\|x - z\|_2 \leq R$, satisfy $f(x) \leq f(z)$.*

Definition 4.2 *A point x is globally optimal if it is feasible and for all feasible points z, $f(x) \leq f(z)$.*

We now come to the crucial element of convex optimization problems, from which they derive most of their utility. The key idea is that ***for a convex optimization problem all locally optimal points are globally optimal***.

Let's give a quick proof of this property by contradiction. Suppose that x is a locally optimal point which is not globally optimal, i.e., there exists a feasible point y such that

[7]Math majors might note that the min appearing below should more correctly be an inf. We won't worry about such technicalities here, and use min for simplicity.

$f(x) > f(y)$. By the definition of local optimality, there exist no feasible points z such that $\|x - z\|_2 \leq R$ and $f(z) < f(x)$. But now suppose we choose the point

$$z = \theta y + (1 - \theta)x \quad \text{with} \quad \theta = \frac{R}{2\|x - y\|_2}.$$

Then

$$\begin{aligned} \|x - z\|_2 &= \left\| x - \left(\frac{R}{2\|x - y\|_2} y + \left(1 - \frac{R}{2\|x - y\|_2} \right) x \right) \right\|_2 \\ &= \left\| \frac{R}{2\|x - y\|_2} (x - y) \right\|_2 \\ &= R/2 \leq R. \end{aligned}$$

In addition, by the convexity of f we have

$$f(z) = f(\theta y + (1 - \theta)x) \leq \theta f(y) + (1 - \theta) f(x) < f(x).$$

Furthermore, since the feasible set is a convex set, and since x and y are both feasible $z = \theta y + (1 - \theta)$ will be feasible as well. Therefore, z is a feasible point, with $\|x - z\|_2 < R$ and $f(z) < f(x)$. This contradicts our assumption, showing that x cannot be locally optimal.

4.2 Special Cases of Convex Problems

For a variety of reasons, it is often times convenient to consider special cases of the general convex programming formulation. For these special cases we can often devise extremely efficient algorithms that can solve very large problems, and because of this you will probably see these special cases referred to any time people use convex optimization techniques.

- **Linear Programming.** We say that a convex optimization problem is a ***linear program*** (LP) if both the objective function f and inequality constraints g_i are affine functions. In other words, these problems have the form

$$\begin{aligned} \text{minimize} \quad & c^T x + d \\ \text{subject to} \quad & Gx \preceq h \\ & Ax = b \end{aligned}$$

  where $x \in \mathbb{R}^n$ is the optimization variable, $c \in \mathbb{R}^n$, $d \in \mathbb{R}$, $G \in \mathbb{R}^{m \times n}$, $h \in \mathbb{R}^m$, $A \in \mathbb{R}^{p \times n}$, $b \in \mathbb{R}^p$ are defined by the problem, and '$\preceq$' denotes elementwise inequality.

- **Quadratic Programming.** We say that a convex optimization problem is a ***quadratic program*** (QP) if the inequality constraints g_i are still all affine, but if the objective function f is a convex quadratic function. In other words, these problems have the form,

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2} x^T P x + c^T x + d \\ \text{subject to} \quad & Gx \preceq h \\ & Ax = b \end{aligned}$$

where again $x \in \mathbb{R}^n$ is the optimization variable, $c \in \mathbb{R}^n$, $d \in \mathbb{R}$, $G \in \mathbb{R}^{m \times n}$, $h \in \mathbb{R}^m$, $A \in \mathbb{R}^{p \times n}$, $b \in \mathbb{R}^p$ are defined by the problem, but we also have $P \in \mathbb{S}^n_+$, a symmetric positive semidefinite matrix.

- **Quadratically Constrained Quadratic Programming.** We say that a convex optimization problem is a ***quadratically constrained quadratic program*** (QCQP) if both the objective f and the inequality constraints g_i are convex quadratic functions,

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + c^T x + d \\ \text{subject to} & \frac{1}{2}x^T Q_i x + r_i^T x + s_i \leq 0, \quad i = 1, \ldots, m \\ & Ax = b \end{array}$$

  where, as before, $x \in \mathbb{R}^n$ is the optimization variable, $c \in \mathbb{R}^n$, $d \in \mathbb{R}$, $A \in \mathbb{R}^{p \times n}$, $b \in \mathbb{R}^p$, $P \in \mathbb{S}^n_+$, but we also have $Q_i \in \mathbb{S}^n_+$, $r_i \in \mathbb{R}^n$, $s_i \in \mathbb{R}$, for $i = 1, \ldots, m$.

- **Semidefinite Programming.** This last example is a bit more complex than the previous ones, so don't worry if it doesn't make much sense at first. However, semidefinite programming is become more and more prevalent in many different areas of machine learning research, so you might encounter these at some point, and it is good to have an idea of what they are. We say that a convex optimization problem is a ***semidefinite program*** (SDP) if it is of the form

$$\begin{array}{ll} \text{minimize} & \text{tr}(CX) \\ \text{subject to} & \text{tr}(A_i X) = b_i, \quad i = 1, \ldots, p \\ & X \succeq 0 \end{array}$$

  where the symmetric matrix $X \in \mathbb{S}^n$ is the optimization variable, the symmetric matrices $C, A_1, \ldots, A_p \in \mathbb{S}^n$ are defined by the problem, and the constraint $X \succeq 0$ means that we are constraining X to be positive semidefinite. This looks a bit different than the problems we have seen previously, since the optimization variable is now a matrix instead of a vector. If you are curious as to why such a formulation might be useful, you should look into a more advanced course or book on convex optimization.

It should be fairly obvious from the definitions that quadratic programs are more general than linear programs (since a linear program is just a special case of a quadratic program where $P = 0$), and likewise that quadratically constrained quadratic programs are more general than quadratic programs. However, what is not obvious at all is that semidefinite programs are in fact more general than all the previous types. That is, any quadratically constrained quadratic program (and hence any quadratic program or linear program) can be expressed as a semidefinte program. We won't discuss this relationship further in this document, but this might give you just a small idea as to why semidefinite programming could be useful.

4.3 Examples

Now that we've covered plenty of the boring math and formalisms behind convex optimization, we can finally get to the fun part: using these techniques to solve actual problems. We've already encountered a few such optimization problems in class, and in nearly every field, there is a good chance that someone has tried to apply convex optimization to solve some problem.

- **Support Vector Machines.** One of the most prevalent applications of convex optimization methods in machine learning is the support vector machine classifier. As discussed in class, finding the support vector classifier (in the case with slack variables) can be formulated as the optimization problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^m \xi_i \\ \text{subject to} & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \ldots, m \\ & \xi_i \geq 0, \qquad\qquad\qquad\quad i = 1, \ldots, m \end{array}$$

with optimization variables $w \in \mathbb{R}^n$, $\xi \in \mathbb{R}^m$, $b \in \mathbb{R}$, and where $C \in \mathbb{R}$ and $x^{(i)}, y^{(i)}, i = 1, \ldots m$ are defined by the problem. This is an example of a quadratic program, which we try to put the problem into the form described in the previous section. In particular, if define $k = m + n + 1$, let the optimization variable be

$$x \in \mathbb{R}^k \equiv \begin{bmatrix} w \\ \xi \\ b \end{bmatrix}$$

and define the matrices

$$P \in \mathbb{R}^{k \times k} = \begin{bmatrix} I & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad c \in \mathbb{R}^k = \begin{bmatrix} 0 \\ C \cdot \mathbf{1} \\ 0 \end{bmatrix},$$

$$G \in \mathbb{R}^{2m \times k} = \begin{bmatrix} -\text{diag}(y)X & -I & -y \\ 0 & -I & 0 \end{bmatrix}, \quad h \in \mathbb{R}^{2m} = \begin{bmatrix} -\mathbf{1} \\ 0 \end{bmatrix}$$

where I is the identity, $\mathbf{1}$ is the vector of all ones, and X and y are defined as in class,

$$X \in \mathbb{R}^{m \times n} = \begin{bmatrix} x^{(1)^T} \\ x^{(2)^T} \\ \vdots \\ x^{(m)^T} \end{bmatrix}, \quad y \in \mathbb{R}^m = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}.$$

You should try to convince yourself that the quadratic program described in the previous section, when using these matrices defined above, is equivalent to the SVM optimization problem. In reality, it is fairly easy to see that there the SVM optimization problem has a quadratic objective and linear constraints, so we typically don't need to put it into standard form to "prove" that it is a QP, and would only do so if we are using an off-the-shelf solver that requires the input to be in standard form.

- **Constrained least squares.** In class we have also considered the least squares problem, where we want to minimize $\|Ax - b\|_2^2$ for some matrix $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. As we saw, this particular problem can actually be solved analytically via the normal equations. However, suppose that we also want to constrain the entries in the solution x to lie within some predefined ranges. In other words, suppose we weanted to solve the optimization problem,

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|Ax - b\|_2^2 \\ \text{subject to} & l \preceq x \preceq u \end{array}$$

  with optimization variable x and problem data $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $l \in \mathbb{R}^n$, and $u \in \mathbb{R}^n$. This might seem like a fairly simple additional constraint, but it turns out that there will no longer be an analytical solution. However, you should be able to convince yourself that this optimization problem is a quadratic program, with matrices defined by

$$P \in \mathbb{R}^{n \times n} = \frac{1}{2}A^T A, \;\; c \in \mathbb{R}^n = -b^T A, \;\; d \in \mathbb{R} = \frac{1}{2}b^T b,$$
$$G \in \mathbb{R}^{2n \times 2n} = \begin{bmatrix} -I & 0 \\ 0 & I \end{bmatrix}, \;\; h \in \mathbb{R}^{2n} = \begin{bmatrix} -l \\ u \end{bmatrix}.$$

- **Maximum Likelihood for Logistic Regression.** For homework one, you were required to show that the log-likelihood of the data in a logistic model was concave. This log likehood under such a model is

$$\ell(\theta) = \sum_{i=1}^{n} \left\{ y^{(i)} \ln g(\theta^T x^{(i)}) + (1 - y^{(i)}) \ln(1 - g(\theta^T x^{(i)})) \right\}$$

  where $g(z)$ denotes the logistic function $g(z) = 1/(1 + e^{-z})$. Finding the maximum likelihood estimate is then a task of maximizing the log-likelihood (or equivalently, minimizing the negative log-likelihood, a convex function), i.e.,

$$\begin{array}{ll} \text{minimize} & -\ell(\theta) \end{array}$$

  with optimization variable $\theta \in \mathbb{R}^n$ and no constraints.

  Unlike the previous two examples, it turns out that it is not so easy to put this problem into a "standard" form optimization problem. Nevertheless, you've seen on the homework that the fact that ℓ is a concave function means that you can very efficiently find the global solution using an algorithm such as Newton's method.

References

[1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge UP, 2004. Online: `http://www.stanford.edu/~boyd/cvxbook/`

Convex Optimization Overview (cnt'd)

Chuong B. Do

October 26, 2007

1 Recap

During last week's section, we began our study of ***convex optimization***, the study of mathematical optimization problems of the form,

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \ldots, m, \\ & h_i(x) = 0, \quad i = 1, \ldots, p, \end{array} \tag{1}$$

where $x \in \mathbb{R}^n$ is the optimization variable, $f : \mathbb{R}^n \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R}$ are convex functions, and $h_i : \mathbb{R}^n \to \mathbb{R}$ are affine functions. In a convex optimization problem, the convexity of both the objective function f and the feasible region (i.e., the set of x's satisfying all constraints) allows us to conclude that any feasible locally optimal point must also be globally optimal. This fact provides the key intuition for why convex optimization problems can in general be solved efficiently.

In these lecture notes, we continue our foray into the field of convex optimization. In particular, we will introduce the theory of Lagrange duality for convex optimization problems with inequality and equality constraints. We will also discuss generic yet efficient algorithms for solving convex optimization problems, and then briefly mention directions for further exploration.

2 Duality

To explain the fundamental ideas behind duality theory, we start with a motivating example based on CS 229 homework grading. We prove a simple weak duality result in this setting, and then relate it to duality in optimization. We then discuss strong duality and the KKT optimality conditions.

2.1 A motivating example: CS 229 homework grading

In CS 229, students must complete four homeworks throughout the quarter, each consisting of five questions apiece. Suppose that during one year that the course is offered, the TAs

decide to economize on their work load for the quarter by grading only one problem on each submitted problem set. Nevertheless, they also require that every student submit an attempted solution to every problem (a requirement which, if violated, would lead to automatic failure of the course).

Because they are extremely cold-hearted[1], the TAs always try to ensure that the students lose as many points as possible; if the TAs grade a problem that the student did not attempt, the number of points lost is set to $+\infty$ to denote automatic failure in the course. Conversely, each student in the course seeks to minimize the number of points lost on his or her assignments, and thus must decide on a strategy—i.e., an allocation of time to problems—that minimizes the number of points lost on the assignment.

The struggle between student and TAs can be summarized in a matrix $A = (a_{ij}) \in \mathbb{R}^{n \times m}$, whose columns correspond to different problems that the TAs might grade, and whose rows correspond to different strategies for time allocation that the student might use for the problem set. For example, consider the following matrix,

$$A = \begin{bmatrix} 5 & 5 & 5 & 5 & 5 \\ 8 & 8 & 1 & 8 & 8 \\ +\infty & +\infty & +\infty & 0 & +\infty \end{bmatrix},$$

Here, the student must decide between three strategies (corresponding to the three rows of the matrix, A):

- $i = 1$: she invests an equal effort into all five problems and hence loses at most 5 points on each problem,
- $i = 2$: she invests more time into problem 3 than the other four problems, and
- $i = 3$: she skips four problems in order to guarantee no points lost on problem 4.

Similarly, the TAs must decide between five strategies ($j \in \{1, 2, 3, 4, 5\}$) corresponding to the choice of problem graded.

If the student is forced to submit the homework without knowing the TAs choice of problem to be graded, and if the TAs are allowed to decide which problem to grade after having seen the student's problem set, then the number of points she loses will be:

$$p^* = \min_i \max_j a_{ij} \qquad (= 5 \text{ in the example above}) \tag{P}$$

where the order of the minimization and maximization reflect that for each fixed student time allocation strategy i, the TAs will have the opportunity to choose the worst scoring problem $\max_j a_{ij}$ to grade. However, if the TAs announce beforehand which homework problem will be graded, then the the number of points lost will be:

$$d^* = \max_j \min_i a_{ij} \qquad (= 0 \text{ in the example above}) \tag{D}$$

where this time, for each possible announced homework problem j to be graded, the student will have the opportunity to choose the optimal time allocation strategy, $\min_i a_{ij}$, which loses

[1] Clearly, this is a fictional example. The CS 229 TAs want you to succeed. Really, we do.

her the fewest points. Here, (P) is called the ***primal*** optimization problem whereas (D) is called the ***dual*** optimization problem. Rows containing $+\infty$ values correspond to strategies where the student has flagrantly violated the TAs demand that all problems be attempted; for reasons, which will become clear later, we refer to these rows as being ***primal-infeasible***.

In the example, the value of the dual problem is lower than that of the primal problem, i.e., $d^* = 0 < 5 = p^*$. This intuitively makes sense: the second player in this adversarial game has the advantage of knowing his/her opponent's strategy. This principle, however, holds more generally:

Theorem 2.1 (Weak duality)**.** *For any matrix $A = (a_{ij}) \in \mathbb{R}^{m \times n}$, it is always the case that*

$$\max_j \min_i a_{ij} = d^* \leq p^* = \min_i \max_j a_{ij}.$$

Proof. Let (i_d, j_d) be the row and column associated with d^*, and let (i_p, j_p) be the row and column associated with p^*. We have,

$$d^* = a_{i_d j_d} \leq a_{i_p j_d} \leq a_{i_p j_p} = p^*.$$

Here, the first inequality follows from the fact that $a_{i_d j_d}$ is the smallest element in the j_dth column (i.e., i_d was the strategy chosen by the student after the TAs chose problem j_d, and hence, it must correspond to the fewest points lost in that column). Similarly, the second inequality follow from the fact that $a_{i_p j_p}$ is the largest element in the i_pth row (i.e., j_p was the problem chosen by the TAs after the student picked strategy i_p, so it must correspond to the most points lost in that row). □

2.2 Duality in optimization

The task of constrained optimization, it turns out, relates closely with the adversarial game described in the previous section. To see the connection, first recall our original optimization problem,

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \ldots, m, \\ & h_i(x) = 0, \quad i = 1, \ldots, p. \end{array}$$

Define the ***generalized Lagrangian*** to be

$$\mathcal{L}(x, \lambda, \nu) := f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{i=1}^{p} \nu_i h_i(x).$$

Here, the variables λ and ν are called the the ***dual variables*** (or ***Lagrange multipliers***). Analogously, the variables x are known as the ***primal variables***.

The correspondence between primal/dual optimization and game playing can be pictured informally using an infinite matrix whose rows are indexed by $x \in \mathbb{R}^n$ and whose columns

are indexed by $(\lambda, \nu) \in \mathbb{R}^m_+ \times \mathbb{R}^p$ (i.e., $\lambda_i \geq 0$, for $i = 1, \ldots, m$). In particular, we have

$$A = \begin{bmatrix} \ddots & \vdots & \iddots \\ \cdots & \mathcal{L}(x, \lambda, \nu) & \cdots \\ \iddots & \vdots & \ddots \end{bmatrix}$$

Here, the "student" manipulates the primal variables x in order to minimize the Lagrangian $\mathcal{L}(x, \lambda, \nu)$ while the "TAs" manipulate the dual variables (λ, ν) in order to maximize the Lagrangian.

To see the relationship between this game and the original optimization problem, we formulate the following ***primal*** problem:

$$\begin{aligned} p^* &= \min_x \max_{\lambda, \nu : \lambda_i \geq 0} \quad \mathcal{L}(x, \lambda, \nu) \\ &= \min_x \quad \theta_P(x) \end{aligned} \tag{P'}$$

where $\theta_P(x) := \max_{\lambda, \nu : \lambda_i \geq 0} \mathcal{L}(x, \lambda, \nu)$. Computing p^* is equivalent to our original convex optimization primal in the following sense: for any candidate solution x,

- if $g_i(x) > 0$ for some $i \in \{1, \ldots, m\}$, then setting $\lambda_i = \infty$ gives $\theta_P(x) = \infty$.
- if $h_i(x) \neq 0$ for some $i \in \{1, \ldots, m\}$, then setting $\lambda_i = \infty \cdot \text{Sign}(h_i(x))$ gives $\theta_P(x) = \infty$.
- if x is feasible (i.e., x obeys all the constraints of our original optimization problem), then $\theta_P(x) = f(x)$, where the maximum is obtained, for example, by setting all of the λ_i's and ν_i's to zero.

Intuitively then, $\theta_P(x)$ behaves conceptually like an "unconstrained" version of the original constrained optimization problem in which the infeasible region of f is "carved away" by forcing $\theta_P(x) = \infty$ for any infeasible x; thus, only points in the feasible region are left as candidate minimizers. This idea of using penalties to ensure that minimizers stay in the feasible region will come up later when talk about barrier algorithms for convex optimization.

By analogy to the CS 229 grading example, we can form the following ***dual*** problem:

$$\begin{aligned} d^* &= \max_{\lambda, \nu : \lambda_i \geq 0} \min_x \quad \mathcal{L}(x, \lambda, \nu) \\ &= \max_{\lambda, \nu : \lambda_i \geq 0} \quad \theta_D(\lambda, \nu) \end{aligned} \tag{D'}$$

where $\theta_D(\lambda, \nu) := \min_x \mathcal{L}(x, \lambda, \nu)$. Dual problems can often be easier to solve than their corresponding primal problems. In the case of SVMs, for instance, SMO is a dual optimization algorithm which considers joint optimization of pairs of dual variables. Its simple form derives largely from the simplicity of the dual objective and the simplicity of the corresponding constraints on the dual variables. Primal-based SVM solutions are indeed possible, but when the number of training examples is large and the kernel matrix K of inner products $K_{ij} = K(x^{(i)}, x^{(j)})$ is large, dual-based optimization can be considerably more efficient.

Using an argument essentially identical to that presented in Theorem (2.1), we can show that in this setting, we again have $d^* \leq p^*$. This is the property of ***weak duality*** for general optimization problems. Weak duality can be particularly useful in the design of optimization algorithms. For example, suppose that during the course of an optimization algorithm we have a candidate primal solution x and dual-feasible vector (λ, ν) such that $\theta_P(x) - \theta_D(\lambda, \nu) \leq \epsilon$. From weak duality, we have that

$$\theta_D(\lambda, \nu) \leq d^* \leq p^* \leq \theta_P(x),$$

implying that x and (λ, ν) must be ϵ-optimal (i.e., their objective functions differ by no more than ϵ from the objective functions of the true optima x^* and (λ^*, ν^*).

In practice, the dual objective $\theta_D(\lambda, \nu)$ can often be found in closed form, thus allowing the dual problem (D') to depend only on the dual variables λ and ν. When the Lagrangian is differentiable with respect to x, then a closed-form for $\theta_D(\lambda, \nu)$ can often be found by setting the gradient of the Lagrangian to zero, so as to ensure that the Lagrangian is minimized with respect to x.[2] An example derivation of the dual problem for the L_1 soft-margin SVM is shown in the Appendix.

2.3 Strong duality

For any primal/dual optimization problems, weak duality will always hold. In some cases, however, the inequality $d^* \leq p^*$ may be replaced with equality, i.e., $d^* = p^*$; this latter condition is known as ***strong duality***. Strong duality does not hold in general. When it does however, the lower-bound property described in the previous section provide a useful termination criterion for optimization algorithms. In particular, we can design algorithms which simultaneously optimize both the primal and dual problems. Once the candidate solutions x of the primal problem and (λ, ν) of the dual problem obey $\theta_P(x) - \theta_D(\lambda, \nu) \leq \epsilon$, then we know that both solutions are ϵ-accurate. This is guaranteed to happen provided our optimization algorithm works properly, since strong duality guarantees that the optimal primal and dual values are equal.

Conditions which guarantee strong duality for convex optimization problems are known as ***constraint qualifications***. The most commonly invoked constraint qualification, for example, is ***Slater's condition***:

Theorem 2.2. *Consider a convex optimization problem of the form* (1)*, whose corresponding primal and dual problems are given by* (P') *and* (D')*. If there exists a primal feasible x for*

[2] Often, differentiating the Lagrangian with respect to x leads to the generation of additional requirements on dual variables that must hold at any fixed point of the Lagrangian with respect to x. When these constraints are not satisfied, one can show that the Lagrangian is unbounded below (i.e., $\theta_D(\lambda, \nu) = -\infty$).

Since such points are clearly not optimal solutions for the dual problem, we can simply exclude them from the domain of the dual problem altogether by adding the derived constraints to the existing constraints of the dual problem. An example of this is the derived constraint, $\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$, in the SVM formulation. This procedure of incorporating derived constraints into the dual problem is known as ***making dual constraints explicit*** (see [1], page 224).

which each inequality constraint is strictly satisfied (i.e., $g_i(x) < 0$), then $d^* = p^*$.[3]

The proof of this theorem is beyond the scope of this course. We will, however, point out its application to the soft-margin SVMs described in class. Recall that soft-margin SVMs were found by solving

$$\begin{array}{ll} \underset{w,b,\xi}{\text{minimize}} & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \ldots, m, \\ & \xi_i \geq 0, \qquad\qquad\qquad\quad i = 1, \ldots, m. \end{array}$$

Slater's condition applies provided we can find at least one primal feasible setting of w, b, and ξ where all inequalities are strict. It is easy to verify that $w = \mathbf{0}$, $b = 0$, $\xi = 2 \cdot \mathbf{1}$ satisfies these conditions (where $\mathbf{0}$ and $\mathbf{1}$ denote the vector of all 0's and all 1's, respectively), since

$$y^{(i)}(w^T x^{(i)} + b) = y^{(i)}(\mathbf{0}^T x^{(i)} + 0) = 0 > -1 = 1 - 2 = 1 - \xi_i, \qquad i = 1, \ldots, m,$$

and the remaining m inequalities are trivially strictly satisfied. Hence, strong duality holds, so the optimal values of the primal and dual soft-margin SVM problems will be equal.

2.4 The KKT conditions

In the case of differentiable unconstrained convex optimization problems, setting the gradient to "zero" provides a simple means for identifying candidate local optima. For constrained convex programming, do similar criteria exist for characterizing the optima of primal/dual optimization problems? The answer, it turns out, is provided by a set of requirements known as the ***Karush-Kuhn-Tucker (KKT) necessary and sufficient conditions*** (see [1], pages 242-244).

Suppose that the constraint functions $g_1, \ldots, g_m, h_1, \ldots, h_p$ are not only convex (the h_i's must be affine) but also differentiable.

Theorem 2.3. *If $\tilde{x}$ is primal feasible and $(\tilde{\lambda}, \tilde{\nu})$ are dual feasible, and if*

$$\nabla_x \mathcal{L}(\tilde{x}, \tilde{\lambda}, \tilde{\nu}) = \mathbf{0}, \tag{KKT1}$$

$$\tilde{\lambda}_i g_i(\tilde{x}) = 0, \qquad i = 1, \ldots, m, \tag{KKT2}$$

then $\tilde{x}$ is primal optimal, $(\tilde{\lambda}, \tilde{\nu})$ are dual optimal, and strong duality holds.

Theorem 2.4. *If Slater's condition holds, then conditions of Theorem 2.3 are necessary for any (x^*, λ^*, ν^*) such that x^* is primal optimal and (λ^*, ν^*) are dual feasible.*

[3] One can actually show a more general version of Slater's inequality, which requires only strict satisfaction of non-affine inequality constraints (but allowing affine inequalities to be satisfied with equality). See [1], page 226.

(KKT1) is the standard gradient stationarity condition found for unconstrained differentiable optimization problems. The set of inequalities corresponding to (KKT2) are known as the ***KKT complementarity (or complementary slackness) conditions***. In particular, if x^* is primal optimal and (λ^*, ν^*) is dual optimal, then (KKT2) implies that

$$\begin{aligned} \lambda_i^* > 0 \quad &\Rightarrow \quad g_i(x^*) = 0 \\ g_i(x^*) < 0 \quad &\Rightarrow \quad \lambda_i^* = 0 \end{aligned}$$

That is, whenever λ_i^* is greater than zero, its corresponding inequality constraint must be tight; conversely, any strictly satisfied inequality must have have λ_i^* equal to zero. Thus, we can interpret the dual variables λ_i^* as measuring the "importance" of a particular constraint in characterizing the optimal point.

This interpretation provides an intuitive explanation for the difference between hard-margin and soft-margin SVMs. Recall the dual problems for a hard-margin SVM:

$$\begin{aligned} \underset{\alpha,\beta}{\text{maximize}} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_i y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle \\ \text{subject to} \quad & \alpha_i \geq 0, & i = 1, \ldots, m, \\ & \sum_{i=1}^m \alpha_i y^{(i)} = 0, \end{aligned} \tag{2}$$

and the L_1 soft-margin SVM:

$$\begin{aligned} \underset{\alpha,\beta}{\text{maximize}} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_i y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, & i = 1, \ldots, m, \\ & \sum_{i=1}^m \alpha_i y^{(i)} = 0. \end{aligned} \tag{3}$$

Note that the only difference in the soft-margin formulation is the introduction of upper bounds on the dual variables α_i. Effectively, this upper bound constraint limits the influence of any single primal inequality constraint (i.e., any single training example) on the decision boundary, leading to improved robustness for the L_1 soft-margin model.

What consequences do the KKT conditions have for practical optimization algorithms? When Slater's conditions hold, then the KKT conditions are both necessary and sufficient for primal/dual optimality of a candidate primal solution $\tilde{x}$ and a corresponding dual solution $(\tilde{\lambda}, \tilde{\nu})$. Therefore, many optimization algorithms work by trying to guarantee that the KKT conditions are satisfied; the SMO algorithm, for instance, works by iteratively identifying Lagrange multipliers for which the corresponding KKT conditions are unsatisfied and then "fixing" KKT complementarity.[4]

[4]See [1], pages 244-245 for an example of an optimization problem where the KKT conditions can be solved directly, thus skipping the need for primal/dual optimization altogether.

3 Algorithms for convex optimization

Thus far, we have talked about convex optimization problems and their properties. But how does one solve a convex optimization problem in practice? In this section, we describe a generic strategy for solving convex optimization problems known as the ***interior-point*** method. This method combines a safe-guarded variant of Newton's algorithm with a "barrier" technique for enforcing inequality constraints.

3.1 Unconstrained optimization

We consider first the problem of unconstrained optimization, i.e.,

$$\underset{x}{\text{minimize}} \quad f(x).$$

In Newton's algorithm for unconstrained optimization, we consider the Taylor approximation $\tilde{f}$ of the function f, centered at the current iterate x_t. Discarding terms of higher order than two, we have

$$\tilde{f}(x) = f(x_t) + \nabla_x f(x_t)^T (x - x_t) + \frac{1}{2}(x - x_t)\nabla_x^2 f(x_t)(x - x_t).$$

To minimize $\tilde{f}(x)$, we can set its gradient to zero. In particular, if x_{nt} denotes the minimum of $\tilde{f}(x)$, then

$$\begin{aligned}
\nabla_x f(x_t) + \nabla_x^2 f(x_t)(x_{\text{nt}} - x_t) &= 0 \\
\nabla_x^2 f(x_t)(x_{\text{nt}} - x_t) &= -\nabla_x f(x_t) \\
x_{\text{nt}} - x_t &= -\nabla_x^2 f(x_t)^{-1} \nabla_x f(x_t) \\
x_{\text{nt}} &= x_t - \nabla_x^2 f(x_t)^{-1} \nabla_x f(x_t)
\end{aligned}$$

assuming $\nabla_x^2 f(x_t)^T$ is positive definite (and hence, full rank). This, of course, is the standard Newton algorithm for unconstrained minimization.

While Newton's method converges quickly if given an initial point near the minimum, for points far from the minimum, Newton's method can sometimes diverge (as you may have discovered in problem 1 of Problem Set #1 if you picked an unfortunate initial point!). A simple fix for this behavior is to use a ***line-search*** procedure. Define the search direction d to be,

$$d := \nabla_x^2 f(x_t)^{-1} \nabla_x f(x_t).$$

A line-search procedure is an algorithm for finding an appropriate step size $\gamma \geq 0$ such that the iteration

$$x_{t+1} = x_t - \gamma \cdot d$$

will ensure that the function f decreases by a sufficient amount (relative to the size of the step taken) during each iteration.

One simple yet effective method for doing this is called a ***backtracking line search***. In this method, one initially sets γ to 1 and then iteratively reduces γ by a multiplicative factor β until $f(x_t + \gamma \cdot d)$ is sufficiently smaller than $f(x_t)$:

Backtracking line-search

- Choose $\alpha \in (0, 0.5)$, $\beta \in (0, 1)$.
- Set $\gamma \leftarrow 1$.
- While $f(x_t + \gamma \cdot d) > f(x_t) + \gamma \cdot \alpha \nabla_x f(x_t)^T d$, do $\gamma \leftarrow \beta\gamma$.
- Return γ.

Since the function f is known to decrease locally near x_t in the direction of d, such a step will be found, provided γ is small enough. For more details, see [1], pages 464-466.

In order to use Newton's method, one must be able to compute and invert the Hessian matrix $\nabla_x^2 f(x_t)$, or equivalently, compute the search direction d indirectly without forming the Hessian. For some problems, the number of primal variables x is sufficiently large that computing the Hessian can be very difficult. In many cases, this can be dealt with by clever use of linear algebra. In other cases, however, we can resort to other nonlinear minimization schemes, such as ***quasi-Newton*** methods, which initially behave like gradient descent but gradually construct approximations of the inverse Hessian based on the gradients observed throughout the course of the optimization.[5] Alternatively, ***nonlinear conjugate gradient*** schemes (which augment the standard conjugate gradient (CG) algorithm for solving linear least squares systems with a line-search) provide another generic blackbox tool for multivariable function minimization which is simple to implement, yet highly effective in practice.[6]

3.2 Inequality-constrained optimization

Using our tools for unconstrained optimization described in the previous section, we now tackle the (slightly) harder problem of constrained optimization. For simplicity, we consider convex optimization problems without equality constraints[7], i.e., problems of the form,

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \ldots, m. \end{array}$$

[5]For more information on Quasi-Newton methods, the standard reference is Jorge Nocedal and Stephen J. Wright's textbook, *Numerical Optimization*.

[6]For an excellent tutorial on the conjugate gradient method, see Jonathan Shewchuk's tutorial, available at: `http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf`

[7]In practice, there are many of ways of dealing with equality constraints. Sometimes, we can eliminate equality constraints by either reparameterizing of the original primal problem, or converting to the dual problem. A more general strategy is to rely on equality-constrained variants of Newton's algorithms which ensure that the equality constraints are satisfied at every iteration of the optimization. For a more complete treatment of this topic, see [1], Chapter 10.

We will also assume knowledge of a feasible starting point x_0 which satisfies all of our constraints with strict inequality (as needed for Slater's condition to hold).[8]

Recall that in our discussion of the Lagrangian-based formulation of the primal problem,

$$\min_x \max_{\lambda:\lambda_i \geq 0} \quad \mathcal{L}(x, \lambda).$$

we stated that the inner maximization, $\max_{\lambda:\lambda_i \geq 0} \mathcal{L}(x, \lambda)$, was constructed in such a way that the infeasible region of f was "carved away", leaving only points in the feasible region as candidate minima. The same idea of using penalties to ensure that minimizers stay in the feasible region is the basis of ***barrier***-based optimization. Specifically, if $B(z)$ is the barrier function

$$B(z) = \begin{cases} 0 & z < 0 \\ \infty & z \geq 0, \end{cases}$$

then the primal problem is equivalent to

$$\min_x \quad f(x) + \sum_{i=1}^{m} B(g_i(x)). \tag{4}$$

When $g_i(x) < 0$, the objective of the problem is simply $f(x)$; infeasible points are "carved away" using the barrier function $B(z)$.

While conceptually correct, optimization using the straight barrier function $B(x)$ is numerically difficult. To ameliorate this, the ***log-barrier*** optimization algorithm approximates the solution to (4) by solving the unconstrained problem,

$$\underset{x}{\text{minimize}} \quad f(x) - \frac{1}{t} \sum_{i=1}^{m} \log(-g_i(x)).$$

for some fixed $t > 0$. Here, the function $-(1/t)\log(-z) \approx B(z)$, and the accuracy of the approximation increases as $t \to \infty$. Rather than using a large value of t in order to obtain a good approximation, however, the log-barrier algorithm works by solving a sequence of unconstrained optimization problems, increasing t each time, and using the solution of the previous unconstrained optimization problem as the initial point for the next unconstrained optimization. Furthermore, at each point in the algorithm, the primal solution points stay strictly in the interior of the feasible region:

[8] For more information on finding feasible starting points for barrier algorithms, see [1], pages 579-585. For inequality-problems where the primal problem is feasible but not strictly feasible, ***primal-dual interior point*** methods are applicable, also described in [1], pages 609-615.

Log-barrier optimization

- Choose $\mu > 1$, $t > 0$.
- $x \leftarrow x_0$.
- Repeat until convergence:
  (a) Compute $x' = \min_x f(x) - \frac{1}{t}\sum_{i=1}^{m} \log(-g_i(x))$ using x as the initial point.
  (b) $t \leftarrow \mu \cdot t$, $x \leftarrow x'$.

One might expect that as t increases, the difficulty of solving each unconstrained minimization problem also increases due to numerical issues or ill-conditioning of the optimization problem. Surprisingly, Nesterov and Nemirovski showed in 1994 that this is not the case for certain types of barrier functions, including the log-barrier; in particular, by using an appropriate barrier function, one obtains a general convex optimization algorithm which takes time polynomial in the dimensionality of the optimization variables and the desired accuracy!

4 Directions for further exploration

In many real-world tasks, 90% of the challenge involves figuring out how to write an optimization problem in a convex form. Once the correct form has been found, a number of pre-existing software packages for convex optimization have been well-tuned to handle different specific types of optimization problems. The following constitute a small sample of the available tools:

- commerical packages: CPLEX, MOSEK
- MATLAB-based: CVX, Optimization Toolbox (linprog, quadprog), SeDuMi
- libraries: CVXOPT (Python), GLPK (C), COIN-OR (C)
- SVMs: LIBSVM, SVM-light
- machine learning: Weka (Java)

In particular, we specifically point out CVX as an easy-to-use generic tool for solving convex optimization problems easily using MATLAB, and CVXOPT as a powerful Python-based library which runs independently of MATLAB.[9] If you're interested in looking at some of the other packages listed above, they are easy to find with a web search. In short, if you need a specific convex optimization algorithm, pre-existing software packages provide a rapid way to prototype your idea without having to deal with the numerical trickiness of implementing your own complete convex optimization routines.

[9] CVX is available at `http://www.stanford.edu/~boyd/cvx` and CVXOPT is available at `http://www.ee.ucla.edu/~vandenbe/cvxopt/`.

Also, if you find this material fascinating, make sure to check out Stephen Boyd's class, EE364: Convex Optimization I, which will be offered during the Winter Quarter. The textbook for the class (listed as [1] in the References) has a wealth of information about convex optimization and is available for browsing online.

References

[1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge UP, 2004. Online: `http://www.stanford.edu/~boyd/cvxbook/`

Appendix: The soft-margin SVM

To see the primal/dual action in practice, we derive the dual of the soft-margin SVM primal presented in class, and corresponding KKT complementarity conditions. We have,

$$\begin{aligned} \underset{w,b,\xi}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1,\ldots,m, \\ & \xi_i \geq 0, \qquad\qquad\qquad\quad i = 1,\ldots,m. \end{aligned}$$

First, we put this into our standard form, with "≤ 0" inequality constraints and no equality constraints. That is,

$$\begin{aligned} \underset{w,b,\xi}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} \quad & 1 - \xi_i - y^{(i)}(w^T x^{(i)} + b) \leq 0, \quad i = 1,\ldots,m, \\ & -\xi_i \leq 0, \qquad\qquad\qquad\quad i = 1,\ldots,m. \end{aligned}$$

Next, we form the generalized Lagrangian,[10]

$$\mathcal{L}(w,b,\xi,\alpha,\beta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i + \sum_{i=1}^{m}\alpha_i(1 - \xi_i - y^{(i)}(w^T x^{(i)} + b)) - \sum_{i=1}^{m}\beta_i\xi_i,$$

which gives the primal and dual optimization problems:

$$\max_{\alpha,\beta:\alpha_i\geq 0,\beta_i\geq 0} \quad \theta_D(\alpha,\beta) \qquad \text{where } \theta_D(\alpha,\beta) := \min_{w,b,\xi} \quad \mathcal{L}(w,b,\xi,\alpha,\beta), \tag{SVM-D}$$

$$\min_{w,b,\xi} \quad \theta_P(w,b,\xi) \qquad \text{where } \theta_P(w,b,\xi) := \max_{\alpha,\beta:\alpha_i\geq 0,\beta_i\geq 0} \quad \mathcal{L}(w,b,\xi,\alpha,\beta). \tag{SVM-P}$$

To get the dual problem in the form shown in the lecture notes, however, we still have a little more work to do. In particular,

[10] Here, it is important to note that (w,b,ξ) collectively play the role of the x primal variables. Similarly, (α,β) collectively play the role of the λ dual variables used for inequality constraints. There are no "ν" dual variables here since there are no affine constraints in this problem.

1. **Eliminating the primal variables.** To eliminate the primal variables from the dual problem, we compute $\theta_D(\alpha, \beta)$ by noticing that

$$\theta_D(\alpha, \beta) = \min_{w,b,\xi} \quad \mathcal{L}(w, b, \xi, \alpha, \beta)$$

is an unconstrained optimization problem, where the objective function $\mathcal{L}(w, b, \xi, \alpha, \beta)$ is differentiable. Therefore, for any fixed (α, β), if $(\hat{w}, \hat{b}, \hat{\xi})$ minimize the Lagrangian, it must be the case that

$$\nabla_w \mathcal{L}(\hat{w}, \hat{b}, \hat{\xi}, \alpha, \beta) = \hat{w} - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} = 0 \tag{5}$$

$$\frac{\partial}{\partial b} \mathcal{L}(\hat{w}, \hat{b}, \hat{\xi}, \alpha, \beta) = -\sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \tag{6}$$

$$\frac{\partial}{\partial \xi_i} \mathcal{L}(\hat{w}, \hat{b}, \hat{\xi}, \alpha, \beta) = C - \alpha_i - \beta_i = 0. \tag{7}$$

Adding (6) and (7) to the constraints of our dual optimizaton problem, we obtain,

$$\begin{aligned}
\theta_D(\alpha, \beta) &= \mathcal{L}(\hat{w}, \hat{b}, \hat{\xi}) \\
&= \frac{1}{2}\|\hat{w}\|^2 + C\sum_{i=1}^{m} \hat{\xi}_i + \sum_{i=1}^{m} \alpha_i (1 - \hat{\xi}_i - y^{(i)}(\hat{w}^T x^{(i)} + \hat{b})) - \sum_{i=1}^{m} \beta_i \hat{\xi}_i \\
&= \frac{1}{2}\|\hat{w}\|^2 + C\sum_{i=1}^{m} \hat{\xi}_i + \sum_{i=1}^{m} \alpha_i (1 - \hat{\xi}_i - y^{(i)}(\hat{w}^T x^{(i)})) - \sum_{i=1}^{m} \beta_i \hat{\xi}_i \\
&= \frac{1}{2}\|\hat{w}\|^2 + \sum_{i=1}^{m} \alpha_i (1 - y^{(i)}(\hat{w}^T x^{(i)})).
\end{aligned}$$

where the first equality follows from the optimality of $(\hat{w}, \hat{b}, \hat{\xi})$ for fixed (α, β), the second equality uses the definition of the generalized Lagrangian, and the third and fourth equalities follow from (6) and (7), respectively. Finally, to use (5), observe that

$$\begin{aligned}
\frac{1}{2}\|\hat{w}\|^2 + \sum_{i=1}^{m} \alpha_i (1 - y^{(i)}(\hat{w}^T x^{(i)})) &= \sum_{i=1}^{m} \alpha_i + \frac{1}{2}\|\hat{w}\|^2 - \hat{w}^T \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \\
&= \sum_{i=1}^{m} \alpha_i + \frac{1}{2}\|\hat{w}\|^2 - \|\hat{w}\|^2 \\
&= \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\|\hat{w}\|^2 \\
&= \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_i y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle.
\end{aligned}$$

Therefore, our dual problem (with no more primal variables) is simply

$$\begin{array}{ll} \underset{\alpha,\beta}{\text{maximize}} & \displaystyle\sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_i y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle \\ \text{subject to} & \alpha_i \geq 0, \qquad\qquad i = 1, \ldots, m, \\ & \beta_i \geq 0, \qquad\qquad i = 1, \ldots, m, \\ & \alpha_i + \beta_i = C, \qquad\qquad i = 1, \ldots, m, \\ & \displaystyle\sum_{i=1}^{m} \alpha_i y^{(i)} = 0. \end{array}$$

2. **KKT complementary.** KKT complementarity requires that for any primal optimal (w^*, b^*, ξ^*) and dual optimal (α^*, β^*),

$$\begin{aligned} \alpha_i^*(1 - \xi_i^* - y^{(i)}(w^{*T} x^{(i)} + b^*)) &= 0 \\ \beta_i^* \xi_i^* &= 0 \end{aligned}$$

for $i = 1, \ldots, m$. From the first condition, we see that if $\alpha_i > 0$, then in order for the product to be zero, then $1 - \xi_i^* - y^{(i)}(w^{*T} x^{(i)} + b^*) = 0$. It follows that

$$y^{(i)}(w^{*T} x^{(i)} + b^*) \leq 1$$

since $\xi^* \geq 0$ by primal feasibility. Similarly, if $\beta_i^* > 0$, then $\xi_i^* = 0$ to ensure complementarity. From the primal constraint, $y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i$, it follows that

$$y^{(i)}(w^{*T} x^{(i)} + b^*) \geq 1.$$

Finally, since $\beta_i^* > 0$ is equivalent to $\alpha_i^* < C$ (since $\alpha^* + \beta_i^* = C$), we can summarize the KKT conditions as follows:

$$\begin{aligned} \alpha_i^* = 0 \quad &\Rightarrow \quad y^{(i)}(w^{*T} x^{(i)} + b^*) \geq 1, \\ 0 < \alpha_i^* < C \quad &\Rightarrow \quad y^{(i)}(w^{*T} x^{(i)} + b^*) = 1, \\ \alpha_i^* = C \quad &\Rightarrow \quad y^{(i)}(w^{*T} x^{(i)} + b^*) \leq 1. \end{aligned}$$

3. **Simplification.** We can tidy up our dual problem slightly by observing that each pair of constraints of the form

$$\beta_i \geq 0 \qquad\qquad \alpha_i + \beta_i = C$$

is equivalent to the single constraint, $\alpha_i \leq C$; that is, if we solve the optimization problem

$$\begin{array}{ll} \underset{\alpha,\beta}{\text{maximize}} & \displaystyle\sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_i y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle \\ \text{subject to} & 0 \leq \alpha_i \leq C, \qquad\qquad i = 1, \ldots, m, \\ & \displaystyle\sum_{i=1}^{m} \alpha_i y^{(i)} = 0. \end{array} \tag{8}$$

and subsequently set $\beta_i = C - \alpha_i$, then it follows that (α, β) will be optimal for the previous dual problem above. This last form, indeed, is the form of the soft-margin SVM dual given in the lecture notes.