Introduction to the Language of Convex Optimization

Elan Frenkel

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-249
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-249.html

December 17, 2015

Acknowledgement

I'd like to thank Professor Rao and Professor Raghavendra for introducing and teaching me mind-boggling applications of this material, which then inspired me to go back and learn the details. I'd also like to thank them for taking the time to read this report.

I'd like to thank Professor El Ghaoui for teaching me a framework for understanding this material. I'd like to thank Professor Boyd and Professor VandenBerghe for writing a lucid text that I could refer to when I was confused. I'd also like to thank Professor Bartlett for providing clarity as to how convex optimization unifies machine learning concepts related to support vector machines.

Introduction To the Language Of Convex Optimization

Elan Frenkel

October 2015

Introduction

These notes were written as part of a Masters Project to help introduce computer science undergraduates to the world of convex optimization. Convex Optimization is a relatively new field that has seen many applications, but the math required to understand it is rarely taught at the undergraduate level. Another aim of this work was selfish– I wanted to synthesize and internalize some of the material that I have learned over the past 2 years for fear of otherwise losing it. I think I have succeeded in the latter goal.

The notes are aimed at introducing the mathematics required of *convex modeling*, rather than algorithms. Convexity is a universal phenomenon in tractable problems, but recognizing it is hard.

The material is largely synthesized from 3 sources– I make no claim to the novelty of content:

(1) Professor Boyd and Professor Vandenberghe's *Convex Optimization* [1](primary source for my first section on Convex Sets and Functions)
(2) Professor El Ghaoui's notes, hyper-textbook, and class for EE 227 A-B [2] (primary source for my second section on Convex Optimization, and in particular the historical anecdotes)
(3) Professor Bartlett lecture notes from CS 289 (primary source for applications in machine learning)

These sources provide go into much greater detail than I have. In particular, I have *italicised* technical terms. If the reader is unfamiliar with them, It would behoove her or him to search the term to learn more about it.

Acknowledgements

I'd like to thank Professor Rao and Professor Raghavendra for introducing and teaching me mind-boggling applications of this material, which then inspired me to go back and learn the details. I'd also like to thank them for taking the time to read this report.

I'd like to thank Professor El Ghaoui for teaching me a framework for understanding this material. I'd like to thank Professor Boyd and Professor VandenBerghe for writing a lucid text that I could refer to when I was confused. I'd also like to thank Professor Bartlett for providing clarity as to how convex optimization unifies machine learning concepts related to support vector machines.

Contents

1 Convex Sets and Convex Functions

1.1 Affine Sets

Before attempting to understand convex optimization it is worthwhile to familiarize and digest the mathematical language in which it operates.

A set C is an *affine set* if for all $x, y \in C$, $\theta \in R$, $\theta x + (1-\theta)y$ is also in C.

Geometrically, $\theta x + (1-\theta)y$ is the unique line containing both vectors x and y. In order to see this, note that by the geometry of vector addition, subtraction and scaling, certainly $y + \theta(x-y)$ is such a line.

An *affine combination* of points $v_1, v_2 ... v_n$ is $\theta_1 v_1 + \theta_2 v_2 + ... \theta_n v_n$ such that $\sum \theta_i = 1$.

A set is affine if and only if it contains every affine combinations of a finite subset of its points. The non trivial direction can be proved via induction.

Affine spaces are closely related to vector subspaces. In fact, they are simply translations of subspaces. If C is affine, and if $x_0 \in C$, then $C - x_o$ is a subspace.

pf: Let $v_1, v_2 \in C - x_0$. Then $\alpha v_1 + \beta v_2 + x0 = \alpha(v_1 + x_0) + \beta(v_2 + x_0) + (1 - \alpha - \beta)x_0$ is in C. Thus $\alpha v_1 + v_2$ is in $C - x_0$.

1.2 Convex Sets

Convex sets are a relaxation of affine sets. For every two points in a convex set, the line segment (instead of the entire line) is contained in that set. This relaxation allows for much more interesting geometries (to the human eye) than simple translations of subspaces.

Mathematically, this amounts to restricting θ to lie in $[0, 1]$. Specifically, a set C is a *convex set* if for all $x, y \in C$, $\theta \in [0, 1]$, $\theta x + (1 - \theta)y$ is also in C.

We define *convex combinations* in the same way as *affine combinations*, except that we add the restriction that $\theta \succeq 0$.(Here, we use the *generalized inequality* $\succeq$ on the vector θ to mean that every θ_i is ≥ 0.) Once again a set is convex if and only if it contains all convex combinations of any finite subset of its points. (Note: finite assumption can be dropped).

1.2.1 Operations that Preserve Convexity

The following operations preserve convexity:

The intersection of (a possibly infinite) set of convex sets
The *image* of a convex set under an affine function (a linear function plus an offset)
Similarly, the *inverse image* of a convex set under an affine function
The *projection* of a convex set onto some of its coordinates
The *sum* two convex sets
The *product* of two convex sets

As an example, consider a polyhedron : $\{x : Ax \succeq b, Cx = d\}$. It can be seen as the intersection of half planes and half spaces (which are easily proven to be convex). It can also be seen as the inverse image of the Cartesian product of the non-negative orthant and the origin under the affine function $(b - Ax, d - Cx)$ (and hence convex).

1.3 Cones

A set C is a *cone* if for all $x \in C$, $\theta \geq 0$, θx is also in C.

We are mostly interested *convex cones*. Combining the two definitions, we see

that a set C is a *convex cone* if for all $x, y \in C$, $\theta_1, \theta_2 \geq 0$, $\theta_1 x + \theta_2 y$ is also in C.

A *proper cone* K is convex, closed (contains its limit points), solid (has a non-empty interior) , and pointed (if $x \in K$ and $-x \in K$ then $x = 0$).

1.4 Examples

1.4.1 Hyperplanes and Halfspaces

Although simple, the *hyperplane* has proven to be very useful in the fields of convex optimization and machine learning. A hyperplane divides our space in two, and as we shall later see, can separate two disjoint convex bodies. By taking a simple dot product, we can see which *halfspace* a given point lies on with respect to the *hyperplane*.

A *hyperplane* H is the affine set $x : a^t x = b$. The vector a is orthogonal to vectors that are spanned by the subspace $x : a^t x = 0$, which can also be thought of as $a^\perp$, the set of vectors perpendicular to a. Of less importance is the scalar b, which determines how far this subspace has been translated.

To see this, consider any $x_0 \in H$. Then H is the set $x : a^t x - b = 0 \equiv x : a^t(x - x_0) = 0$. The solution to this latter equation is the set $a^\perp + x_0$.

The shortest distance from the origin to the hyperplane H must be in the direction of a. (This can be proved using a right triangle and elementary geometry). Therefore, dist(H,0) $=\|ka\|$ for some scalar k. Since $a^t ka = b$, $k = \frac{b}{\|a\|^2}$. The distance from H to the origin is therefore $\frac{|b|}{\|a\|}$ in the direction $sign(b) * a$.

A *halfspace* is the set $x : a^t x \leq b$. It is convex but not affine. Geometrically, it is the set of points on one side of a hyperplane. If we assume b is positive (which we can always do by dividing by -1), a given point x_i lies on the halfspace containing the origin if and only if $a^t x \leq b$. (Note the technicality regarding the boundary).

1.4.2 The Positive Semi-Definite Cone

The *positive semi-definite cone* S^{n+} is a subset of the set of symmetric matrices S^n. A matrix A is positive semi-definite if and only if for all vectors v $v^t Av \geq 0$.

It is a convex cone. If $A, B \in S^{n+}$, $\alpha \geq 0$, then $v^t(\alpha_1 A + \alpha_2 B)v = \alpha_1 v^t Av + \alpha_2 v^t Bv \geq 0$. It is also a proper cone.

Another neat way to prove convexity is by showing that S_+^n intersection of infinitely many half spaces. Consider $\cap_{v \in R^n}\{X : X \in S^n \text{ and } v^t X v \geq 0\}$. For each v, the set of X that satisfy the inequality is a half space in $(n^2 - n)/2 + n$ variables.

A matrix A is called *positive definite* if the inequality above is *strict*, meaning $v^t A v > 0$.

1.5 The Spectral Decomposition Theorem

Let A be a real symmetric matrix. The *spectral decomposition theorem* says that there exists a decomposition such that $A = U^t D U$, where U is an *orthogonal matrix* and D is a *diagonal matrix*. The rows of U are precisely the *eigenvectors* of A, and the diagonal entries of D are the corresponding eigenvalues of A. The proof of this theorem is constructive and uses the *Gram-Shmidt Process*. Note that this theorem guarantees that every real symmetric matrix is *diagonalizable* , or put another way, has an *eigenbasis*.

An orthogonal matrix U has the property that $U^t U = I$. Seen as linear operators, orthogonal matrices preserve the dot product between vectors (and hence the norm of a vector): $< Ux, Uy >= xU^t Uy =< x, y >$. Geometrically, these operators represent rotations or reflections of the space.

A *diagonal* matrix is a matrix where all non-diagonal entries are 0. Seen as a linear operator, diagonal matrices simply stretch and shrink a vector along the coordinate axis.

The spectral decomposition gives us a way to visualize the action of any real symmetric matrix on a vector. First, the vector's coordinate system is oriented to the eigenbasis via U. Then, the vector is stretched and shrinked along certain axis proportional to the eigenvalues of the corresponding axis. (Remember: our axis now are our eigenvectors). Finally, the new vector's coordinates are rewritten in the original basis via the inverse of U, namely U^T.

1.5.1 Ellipsoids

(courtasy of Professor El Ghaoui)

Ellipsoids are geometric structures which can give clarity and insight to many topics in convex optimization and linear algebra.

An ellipsoid can be defined as the image of an affine transformation of the unit ball: $\{y : y = x_0 + A(x), \|x\| \leq 1, A \text{ non-singular}\}$.

Rewriting the above condition, we see that the ellipsoid can equivalently defined as $\{y : \|A^{-1}(y - x_0)\| \leq 1\} = \{y : (y - x_0)^T A^{-T} A^{-1}(y - x_0) \leq 1\}$. We can now define $B = A^{-T}A^{-1}$, which is trivially positive definite.(why?)

By taking the symmetric decomposition of $B = U^T DU$, and letting $y^* = U(y - x_0)$, we see that the ellipse is the set $\{y^* : y^{*T} D y^* \leq 1 \rightarrow \sum y_i^2 d_i \leq 1 \rightarrow \sum \frac{y_i^2}{(1/\sqrt{(d_i)})^2} \leq 1$.

Thus we see that an ellipsoid's axes are aligned with the eigenvectors of B. The length of each axis is proportional to the respective inverse of the square root of the eigenvalue of B. Basic linear algebra facts tells us that B^{-1} is positive definite, has the same eigenvectors as B, and has eigenvalues which are the inverse of the eigenvalues of B. Furthermore, every positive definite matrix has a unique positive definite square root. Looking back, we defined A to be an arbitrary invertible matrix. But we could have defined it to be the unique positive definite square root of B^{-1}. Since each positive definite matrix results in a different ellipsoid (if we ignore the center x_0), the set of positive definite matrices are in one-to-one correspondence with the set of ellipsoids centered at the origin.

1.6 Generalized Inequalities

The mathematical abstraction (some would say nonsense) of a *generalized inequality* $\succeq$ induced by *proper cones* provides conceptual clarity for the theory of semi-definite programming as it relates to linear programming.

Given 2 elements A, B in some underlying space which contains a proper cone K, we say $A \succeq_K B$ if $A - B \in K$. We say $A \succ B$ if $A - B$ is in the interior of K.

This ordering satisfies many of the algebraic properties you would expect of an ordering, such as transitivity and reflexivity.

However, it is not necessarily a *total ordering*. Many elements in the underlying space may not be be comparable.

Rather than delving into these abstractions, however, we note that we will be using 2 generalized inequalities frequently:

(1) The generalized inequality induced by K = non-negative orthant. Given two vectors $x, y \in R^n$ we say $x \succeq_K y$ if $x_i \geq y_i$ for all i. Similarly, we say $x \succ_K y$ if $x_i > y_i$ for all i. This inequality between vectors is used so frequently that we drop the subscript K from now on.

(2) The generalized inequality induced by $K = S^n_+$, the set of semi-positive definite matrices. Given two matrices $X, Y \in S^n$, we say $X \succeq_K Y$ if $X - Y$ is positive semi-definite. The interior of S^n_+ is the set of positive definite matrices. Therefore, we say $X \succ_K Y$ if $X - Y$ is positive definite. Once again, we use this inequality between matrices so frequently that we normally drop the subscript K.

1.6.1 Minimum and Minimal Elements

Given a set S, and a generalized inequality $\succeq_K$, we can define the *minimum* element of the set, x to be the element such that for all $y \in S$ $y \succeq_K x$. Note that this requires x to be comparable to all other elements in the set, and therefore this element will often not exist.

We are also interested in *minimal* elements of a set S. An element x is minimal in S if for all $y \in S$, $x \succeq_K y$ if and only if $x = y$.

If K is the non-negative orthant in R^2, for instance, an element x is the minimum element in S if all other elements are above and to the right of it. An element is minimal if there are no elements to the buttom and left of it.

Ellipsoids give us geometric intuition on to the ordering of positive definite matrices. For each positive definite matrix X one can associate the ellipsoid $E_X : \{v : v^T X v \leq 1\}$. Then $X \succ Y$ if and only if $E_X \subset E_Y$.

As an example, suppose we take 2 positive definite matrices A, B. We define S to be the set of positive definite matrices $\{X : X \succeq A \text{ and } X \succeq B\}$. An element $C \in S$ is minimal if E_C is contained within E_A and E_B but no other ellipsoid can 'squish' between the containment (There does not exist D such that $E_C \subset E_D$ and $E_D \subset E_A, E_D \subset E_B$.) Drawing examples in R^2 should provide the insight that there could be many minimal elements in general, and hence there does not exist a minimum element of S in general.

1.7 The Separating Hyperplane Theorem

An important theoretical result for the theory of convex optimization is the following: Given 2 convex bodies X, Y, we can find a hyperplane $H(v) = a^T v - b$

that 'seperates' them, meaning $H(x)$ is non-negative for all $x \in X$, and $H(y)$ is non-positive for all $y \in Y$.

Proof:

Let $\alpha \in X$ and $\beta \in Y$ be chosen such that the $dist(x, y)$, $x \in X, y \in Y$ is minimized. Consider the vector $a = \alpha - \beta$. This will be the orthogonal vector for our hyperplane.

Now we need to figure out b. We want to translate our hyperplane so that it is midway between the segment between α and β. This amounts to projecting the midpoint of α and β onto a. Thus $b = \frac{(\alpha+\beta)}{2} * (\alpha - \beta) = (\|\alpha\|^2 - \|\beta\|^2)/2$.

Note that this is a very natural choice for a and b. Now we need to prove that $H(v) = a^T v - b$ separates X and Y.

We prove that for all $x \in X$, $H(x)$ is non-negative. The proof for $y \in Y$ is non-positive is similar.

Suppose for contradiction that there exists a q such that $q \in X$ and $H(q) < 0$. Now, $H(q) = a^T q - b = (\alpha - \beta)^T q - \frac{(\alpha-\beta)^T(\alpha+\beta)}{2} = (\alpha - \beta)^T (q - (\alpha + \beta)/2) = (\alpha - \beta)^T (q - \alpha + (\alpha - \beta)/2) = (\alpha - \beta)^T (q - \alpha) + \frac{1}{2}\|\alpha - \beta\|^2$.

Therefore, if H(q) is negative than $(\alpha - \beta)^T (q - \alpha)$ must be negative.

Consider the line segment between α and $q \equiv \alpha + t(q - \alpha)$. Now consider a small neighborhood, around α, and its distance to β. $\frac{d}{dt}\|\alpha + t(q-\alpha) - \beta\||_{t=0} = 2(\alpha - \beta)^T (q - \alpha)$.

Since this is precisely the quantity that we proved was negative earlier, this implies that there exists a point c on the line segment between α and q such that the $dist(c, \beta) < dist(\alpha, \beta)$. Moreover, since X is convex, c is in X! But this contradicts the fact that α and β were chosen such that the distance between X and Y is minimized. QED.

1.8 Convex Functions

A function f is said to be a *convex function* on a convex domain D if for all $x, y \in D$ and scalars $\{\lambda : 0 < \lambda < 1\}$, $f(\lambda x + (1 - \lambda y) \leq \lambda f(x) + (1 - \lambda) f(y)$.

Geometrically, a function is convex if for any two points in the domain, the segment between the output values on these two points lies above the graph of

the function.

A function f is said to be a *concavefunction* if $-f$ is convex. For the remainder of these notes, we will restrict our discussion to convex functions. It should be importantly noted that for each concept for convex functions and their optimization problems, there is a natural equivalent concept for concave functions. (Which can easily be derived from the fact from if f concave $\rightarrow -f$ convex.

1.8.1 The Epigraph Criterion

The *epigraph* of a function f from $R^n \rightarrow R$ is the set E in R^{n+1}, where $E = \{(x,t) : t > f(x)\}$.

The connection between convex functions and convex sets is provided by the epigraph. A function f is convex if and only if it's epigraph is a convex set.

Since the epigraph is a set, the geometric intuition of convex sets can be applied to the study of convex functions.

1.8.2 Sub-level Sets of a Convex Function

The *sub-level sets* of a function are defined as the sets $S_\alpha : \{x : f(x) \leq \alpha\}$.

The sub-level sets of a convex function are convex: Let $x, y \in S_\alpha$. Then $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) \leq \alpha$..

1.8.3 Jensens Inequality

Let f be a convex function.

The definition of convexity can be extended to convex combinations of points:

$f(\theta_1 x_1 + \theta_2 x_2 + .../ / \theta_i x_i) \leq \theta_1 f(x_1) + \theta_2 f(x_2) + ...\theta_i f(x_i)$ such that $\theta \geq 0$ and $\sum \theta_i = 1$.

This extends to infinite sums and integrals: $f(\int_D p(x)x) \leq \int_D p(x)f(x)$ where $\int_D p(x) = 1$.

If x is a random variable, this means that $f(E(x)) \leq E(f(x))$.

If x is fixed, and if r is a random vector with zero mean in R^N, then from the above equation we see that if f is convex, $E(f(x+r)) \geq f(x)$. This can be interpreted to mean that if a function is convex, then adding random (symmetric) noise cannot decrease the value of f on average.

These inequalities are all known as Jensen's inequality. They can be used to derive many famous inequalities .

For example, if we take $f(x) = -log(x)$, which is convex, (see the section on differentiable convex functions), and use $\theta = \frac{1}{2}$ in the standard definition of convexity, one can derive that the arithmetic mean is always greater than the geometric mean. (Try it!)

1.8.4 Examples

Norms

Let $\|\|_*$ be any norm on R^n. Then $\|\lambda x + (1-\lambda)y\|_* \leq \|\lambda x\|_* + \|(1-\lambda)y\|_* = \lambda\|x\|_* + (1-\lambda)\|y\|_*$. Thus, norms are always convex.

Max Function

Let $f(x) = max\{x_1, x_2 ... x_n\}$. Then $max(\lambda x + (1-\lambda)y \leq max(\lambda x) + max(1-\lambda)y = \lambda max x + (1-\lambda) max y$. Thus, the maximum element of a vector is convex.

1.9 Differentiable Convex Functions

If the function f is differentiable on a convex domain D then, we have other characterizations of convexity.

1.9.1 First Order Condition

The differential function f is convex on D if and only if the *first order taylor approximation* is a global under-estimator of the function on D:

$f(y) \geq f(x) + (y-x)^T * \bigtriangledown f$ for all $x, y \in D$

This property is part of the reason convex optimization is 'easy' (complexity

wise). Once you find a minimum, you can be assured that it is the global minimum.

Proof It is easy to see that a function is convex if and only if it is convex restricted to any line in its domain. Thus the first step is to prove the equivalence for scalar functions.

Let f be a function on R. For sufficiency, assume convexity : $f(x + t(y - x)) - (1 - t)f(x) \leq tf(y)$. Divide by t and take the limit as $t \to 0$ to get the result.

If f satisfies the first order condition, define $z = \theta x + (1 - \theta)y$. Then $f(x) \geq f(z) + f'(z)(x - z)$. Similarly, $f(y) \geq f(z) + f'(z)(y - z)$. Multiply the first equation by θ, the second by $1 - \theta$, and add them to prove that f is convex.

For the general case, for necessity simply restrict f to a line. For any $x, y \in D$ let $g(t) = f(tx+(1-t)y)$. If f is convex then so is g. Since g is a scalar function, we apply the result above and therefore the first order condition is met on g. Now apply the first order condition $g(1) \geq g(0) + g'(0)$ to get the general result.

For sufficiency in the general case, apply the first order condition to the points $x* = t_1x + (1 - t_1)y$ and $y* = t_2x + (1 - t_2)y$. A simple application of the chain rule and implies that $g(t_1) \geq g(t_2) + g'(t_2)(t_1 - t_2)$. This implies the convexity g by our 1-dimensional result, which in turn implies the convexity of f.

1.9.2 The Second Order Condition

If a function f if twice differentiable, then it is convex if and only if the hessian matrix is semi-positive definite: $\bigtriangledown^2 f \geq 0$.

Proof Idea and Hints: The structure of the proof is the same as the first order condition. First you prove equivalence to the first order condition in the scalar case. Expanding the Taylor series one can see that the first order criterion implies the second order criterion. In the reverse direction, one can use the fact that a non-negative second derivative implies a non-decreasing first derivative. Then you restrict the function to a line to prove the general case.

1.9.3 Examples

Negative Entropy

Let $f = xlogx$. Then $f'' = 1/x$ which is positive on the domain R^++. There-

fore the negative entropy function is convex on its domain.

Log Sum Exp

Consider $f(x) = log \sum e^{x_i}$, which is often used as a differentiable approximation of the max function, since $max(x) \leq f(x) \leq max(x) + log(n)$.

Let $v = (e^{x_1}, e^{x_2}..., e^{x_n})$. One could check that $\bigtriangledown^2 f = \frac{1}{(1^T v)^2} * ((1^T v) diag(v) - vv^T)$ Given a vector y, we must show that $y^T \bigtriangledown^2 fy \geq 0$. But this can be proven by using the Cauchy Swartz inequality $a^T abTb \geq (a^T b)^2$ with $a_i = y_i \sqrt{v_i}$ and $b_i = \sqrt{v_i}$.

1.10 Operations That Preserve Convexity

(Professor El Ghaoui's synthesis)

(1) The non-negative weighted sum of convex functions is convex

(2) The composition with an affine function is convex. Meaning, if $f(x)$ is convex, then so is $f(Ax + b)$

(3) The pointwise maximum of a family of convex functions is convex. Namely, if f_i is convex for all i, then $f(x) = max_i f_i(x)$ is convex.

Example: Largest Singular Value of a Matrix:

The function which take the largest singular value of a matrix: $f(A) \rightarrow max \|Ax\|_2 : \|x\|_2 = 1$ is convex because it the pointwise maximum (indexed by $x : \|x\| = 1$) of the convex function $\|\|_2$ composed with the affine function $A \rightarrow Ax$.

WARNING: This example often causes confusion because the function f is of the matrix A and not the variable x.

(4) The partial minimization of a convex function is convex. If $f(x, y)$ is convex in x and y (note that these can both be vectors of any length), then $f(x) = min_y f(x, y)$ is convex.

(5) The composition of 2 convex functions is **not** always convex. However, if h, from $R^k \rightarrow R$ is convex and non-decreasing in each argument, and $g_i : i = 1..k$ are all convex, then $f(x) = h(g_1(x), g_2(x)...g_i(x))$ is convex.

2 Convex Optimization

(Onwards is largely taken from Professor El Ghaoui's textbook and notes, and class).

In this section we describe the general form of a convex optimization problem. The form is important because most problems in this form can be solved efficiently.

Let f_i be convex for all i. A *convex optimization problem* is a problem in the following form:

Minimize $f_0(x)$
Subject To The Following Constraints:
$f_i(x) \leq 0$ for $i = 1...m$
$Ax = b$

The *feasible set* is the set of points X such that the constraints are not violated. Note that in a convex optimization problem the feasible set is convex: it is the intersection of sub-level sets of convex functions and the hyperplane $Ax = b$.

The *objective function* is the function f_0.

Although the form may seem restrictive, it is actually quite general. There are many standard tricks to bring a wide variety of problems into this form. We will encounter some.

2.0.1 Optimality

A point x^* is said to be locally optimal if there exists a radius $R > 0$ such that x^* is feasible and attains the minimum on $f_0$for all points $x : \|x^* - x\| \leq R$.

An important property for convex optimization problems is that any feasible local optimum must be a global optimum:

Proof: Let x^* be a local optimum. Let y be another feasible point. We can assume $f_0(y) \leq \infty$, otherwise we are done. Since the feasible set is convex, the entire line segment between x^* and y is feasible. Let $z = \alpha y + (1 - \alpha)x^*$ be a point on the segment. Now since f_0 is convex, $f_0(z) \leq \alpha f_0(y) + (1-\alpha)f_0(x^*) \rightarrow f_0(z) - f_0(x^*) \leq \alpha(f_0(y) - f_0(x^*))$. Since x^* is a local minimum, there exists a z such that $f_0(z) - f_0(x^*)$ is positive. But this implies that $f_0(y) - f_0(x^*)$ is also positive.

Note that although every local optimum is a global optimum, we may have more than one optimum. Of course, as one can easily show, the set of optimal points is convex !

Differentiable objective functions

If the objective function f_0 is differentiable we can further characterize optimal points.

We know that $f_0(y) \geq f_0(x^*) + \bigtriangledown f_0(x^*)(y - x^*)$ This implies that x^* is optimal if and only if $\bigtriangledown f_0(x^*)(y - x^*) \geq 0$.

A *supporting hyperplane* $av = b$ for a set is a hyperplane in which all the points in the set lie on one side of it. (When you plug a point $v \in S$ into $av - b$ you always either get purely non-negative numbers or purely non-positive numbers.)

Thus, we can say that if x^* is optimal, $\bigtriangledown f(x^*)$ is a supporting hyperplane to the feasible set.

Some Special Cases:

(1) If the function f_0 is differentiable and there are no constraints, the optimality condition reduces to $\bigtriangledown f(x^*) = 0$, which confirms what we learned in calculus class.

(2) If the only constraints are linear equality constraints $Ax = b$, then we can use linear algebra to find a neat characterization:

Let x^* be optimal and y be another feasible point. Note that since both y and x^* satisfy $Ax = b$, we know that the set of feasible y can be written as $x^* + N(A)$, where $N(A)$ is the nullspace of A.This implies that the optimality condition can be written as for all $u \in N(A)$, $\bigtriangledown f_0(x^*)^T u = 0$. By the fundamental theorem of algebra, the $N(A)$ is orthogonal to the range of A^T. Therefore, the optimality is assured if there exists a v such that $A^T v = \bigtriangledown(f_0(x^*))$.

In summary, optimality is assured if x^* satisfies $Ax = b$ and there exists a v such that $A^T v = \bigtriangledown f_0(x^*))$

2.1 Linear Programming

2.1.1 A Little History: An International Effort

The first type of convex optimization problem to be implemented efficiently was Linear Programming. George Dantzig, an American mathematical scientist, invented the *simplex* method in 1949 which solved most linear programs efficiently in practice. However in the worst case exponential behaviour was possible. In 1979, Leonid Khachiyan, a Soviet of Armenian descent, discovered the ellipsoid method for solving all linear programs in polynomial time. This method was theoretically valid but hard to use in practice due to numerical issues. In 1984 Narendra Karmarkar, an Indian mathematician made the final breakthrough by developing an *interior point method* that was both theoretically efficient and empirically robust in practice.

2.1.2 A standard form

A linear program is simply a convex optimization problem in which all f_i are affine functions. Namely:

min $a^T x$
subject to:
$Ax = b$
$Bx \leq c$

Some off-the-shelf algorithms use the following 'standard' form :

minimize $a^T x$ such that
$Bx = d$
$x \succeq 0$

To go from the first form to standard form we introduce non-negative slack variables s_i and transform the inequality constraint to equality:
$Bx + s = c$.

Then we introduce non-negative. x^+, x^- so that $x = x^+ - x^-$. The transformed problem looks like:

min $a^T x^+ - a^T x^-$
subject to:
$Ax^+ - Ax^- = b$
$Bx + s = c$
$x^+ \succeq 0, x^- \succeq 0, s \succeq 0$

The final step is to create an appropriate mega matrix (composed of A, B, I , mega vector (composed of x^+, x^-, s), and new objective (doubling a and padding with 0's) to get it into the compact standard form described earlier. I will spare the reader the gory details.

Bottom Line: Linear programming is very versatile. One can take inequalities or equalities of affine functions in the constraint. One can also maximize or minimize the objective (since affine functions are both convex and concave). As long as all of constraints are linear, we can always bring the problem back to standard form.

2.1.3 Machine Learning Application: Training a Binary Classifier

The problem of binary classification is fundamental to machine learning. Using a linear classifier (a separating hyperplane) is a historical technique dating back to the perception algorithm. The theory of convex optimization has given machinery to view separating hyperplanes in a principled fashion. We start with the simplest of models, which leads to a linear program. We will revisit this example as our machinery expands.

Consider the problem of training a separating hyperplane on training set of points X, where each point x_i has a label $y_i \in \{+1, -1\}$. If we know the points are separable, than there exists a hyperplane $w^T x + b$ which separates the points. Given the points, our problem is to find w and b.

Our conditions give us the affine inequality constraints:

$y_i(w^T x_i + b) \geq 0$ for each (x_i, y_i)

This is called an *unconstrained linear program* since there is no objective function. (We can simply replace the objective function with the constant function in standard software packages, which is trivially affine).

Once we have 'learned' w and b via our training examples, we can predict a new data point x^* label $y^* = sign(w^T x^* + b)$.

The preceding example is contrived in the sense that we assumed our training data is separable. In practice, there is usually no such reason that our data will be linearly separable, and therefore our training data need not be linearly separable.

Thus, we aim to minimize the amount of mis-classifications by the hyperplane.

This leads to the minimization of the objective function $\sum I_{y_i(w^T x_i+b)}$. Here, $I_(r)$ is the function which is 0 if $r \geq 0$ and 1 if $r < 0$.

Of course, this objective function is not convex, let alone linear, so we will have to upper bound by the use of a 'hinge function' : $(1-v)_+ = max(0, 1-v)$, yielding the following objective:

$min \sum (1 - y_i(w^T x_i + b))_+$

In order to transform this into a linear program, we introduce slack variables z_i:

$min \sum z_i$
such that
$z_i \geq (1 - y_i(w^T x_i + b))$
$z_i \geq 0$

It is worthwhile to convince yourself that both problems are equivalent. (What value does z_i take at optimum?)

This is a good start, but in order to find the better hyperlanes we will need more sophisticated machinery.

2.2 Conic Optimization

'I think frugality drives innovation, just like other constraints do. One of the only ways to get out of a tight box is to invent your way out' - Jeff Bazos

2.2.1 A Little More History

(Story via Professor El Ghaoui's Lecture. Forgive my recall errors)
After the success of linear programming in the 1950's many researchers set out to gradually expand the optimization frontier. Most researchers believed that the next frontier was in nonlinear programming.

In America, where resources were widely available, scientists explored heuristics to tackle non-linearity's. Unfortunately, as soon as you introduce non-linear equality constraints, the problem becomes non-convex and very hard to solve.

In the Soviet Union, academics were severely limited. They were often restricted to both the quantity of publications and number of pages they could

write. Perhaps this is one of the reason Soviet mathematicians concentrated on the theoretical underpinnings to convex optimization.

Consider the linear program in 'off the shelf' standard form:

$mina^T x$
subject to:
$Bx = d$
$x \succeq 0$

As alluded to earlier, one way to generalize this model is to allow for non-linearities in the equality constraints. This was tried but was unsuccessful as the problems became non-convex and intractable rather easily.

The other way to generalize is to modify the 'structural constraints' on x. Instead of requiring x to lie in R_+, we can consider other regions. The regions that turn out to work are convex cones!

This is precisely what the Soviet mathematicians Nesterov and Nemiroski developed in the 1980's. They developed a theory for a class of cones which are tractable.

What was learned from this is that what made linear programming 'easy' was not the fact that it was linear (although this helped), but that the problems were convex.

In 1994, Nesterov and Nemiroski published a landmark paper on polynomial interior point methods for non-linear convex optimization problems.

2.2.2 Tractable Cones

Consider the conic programming model:

$mina^T x$
subject to: $Ax = b$
$x \in C$

Where C is a convex cone.

We have the following tractable conic models:

(1) $C = R^N_+$. This obviously reduces to linear programming.

(2) Let $v \in R^N$. Then $C = Q^N \equiv \{(v,t) : \|v\|_2 \leq t\}$.

This is called the *second order cone*. If $v \in R^2$, for example, we get the familiar 'ice cream cone' in R^3. Note that the cross sections are circles.

(3)$C = S_+^N$, the set of positive semi-definite matrices.

Since the direct product of convex cones is a convex cone, we can mix and match as we like. For instance, if $x \in R^4$, we could require that $x_1 \geq 0$, and $\|(x_2, x_3)\| \leq x_4$. This new cone is the result of the direct product of R_+ and Q^2.

Each of the cones are nested in one another. We can recover the positive orthant R_+^N, we take the direct product of Q n times, while plugging in $x = 0$.

To show that the semi-positive definite cone is a generalization of the second order cone, consider the inequality $\|y\|_2 \leq t$, which can be represented as :

$$Y = \begin{pmatrix} t & y_1 & \cdots & y_n \\ y_1 & t & 0 \cdots & 0 \cdots \\ \vdots & 0 \cdots & t & 0 \cdots \\ y_n & 0 \cdots & 0 \cdots & t \end{pmatrix} \succeq 0$$

To prove the equivalency, we must invoke the *schur complement lemma*,, which states that given a symmetric matrix X partitioned into blocks A, B, C, with C positive definite:

$X = \begin{pmatrix} A & B \\ B & C \end{pmatrix}$, then X is semi-positive definite if and only if $A - B^T C^{-1} B^T$ is semi-positive definite.

Clearly our matrix Y satisfies the condition for X. But then the schur complement condition reads Y is semi-positive definite if and only if $t^2 \geq \|y\|^2$, which reduces to the second order cone constraint.

2.3 Second Order Cone Programming

While the conic form presented in the preceding is concise and theoretically pleasing, in practice the standard form for second order programming has a different form as to enable more immediate versatility:

min $c^T x$
subject to: $Ax = b$
$\|C_i x + d_i\| \leq e_i^T x + f_i$ for all i

We basically allow the minmization of an affine function subject to affine equality constraints and second order cone constraints, where the variables are now affine functions as well.

Like linear programming, this form can be converted into the form seen earlier through standard tricks.

2.3.1 Rotated Second Order Cone Constraints

A rotated second order cone constraint , where $x \in R^N$, and $y, z \in R_+$is of the following form:

$\{(x, y, z) : \|x\|_2 \leq yz.$

Note that these are equivalent to a second order cone constraint, since:

$\|x\|_2 \leq yz \iff \|(2x, y - z)\|_2 \leq y + z.$

2.3.2 Reductions to Linear and Quadratic Programming

First we note that if we have a non-linear objective function f_0 we can always replace it with a slack variable t, minimize t and add the constraint $f_0(x) \leq t$.

Thus we need only focus on constraints to see the generality of the model.

Consider the second order cone constraint $\|Ax + b\|_2 \leq c^T x + d$. If we let $A = 0, b = 0$ then we recover linear programming.

If we let $c = 0$, and square both sides of the inequality we get $x^T A^T Ax + 2(Ax)^T b \leq d - \|b\|^2$. Thus we can choose A, b, d appropriately to get constraints of the form:

$x^T Ex + f^x \leq g$, where E is semi-positive definite. (Recall that every semi positive matrix can be factored.)

An optimization problem with constraints of this form is called *quadratic programming.*

2.3.3 Revisiting Machine Learning: Separating Hyperplanes with Maximum Margin

(this section is synthesized from Peter Bartlett's Course on Machine Learning)

Recall the problem of training a separating hyperplane on a set of training data (x_i, y_i) where each $y_i \in \{-1, 1\}$. We previously found a linear program which solved the problem in the case that the data was assumed to be linearly separated, and then we solved it in the more likely case that the data is not guaranteed to be separable.

However, in general, there may be many hyperplanes which satisfy the linear program equally well. How do we distinguish between them?

One idea is to try to maximize the *margin* of the separating hyperplane. The margin is simply the distance of the hyperplane to the point closest to it. It can be empirically validated that this works well.

First, we will simplify our analysis by ridding ourselves of the offset term in our hyperplane, by appending 1 to each data point.

Once again, let us consider the simple case in which the data is separable. Even more, we will assume that the data is *strictly* separable, which simply means that there exists a normal vector θ and $\delta > 0$ such that $y_i \theta^T x_i \geq \delta$ for all i.

We can always scale θ so that $y_i \theta^T x_i \geq 1$, where this equation holds with equality for some x_j (which is the point closest to the hyperplane).

The distance from x_j is precisely $\frac{y_j \theta^T x_j}{\|\theta\|} = \frac{1}{\theta}$. This is the quantity we want to maximize, which is the same as minimizing θ.

Thus, training a hyperplane with maximum margin for separable data leads to the following optimization problem:

min $\|\theta\|$
subject to:
$y_i \theta^T x_i \geq 1$ for all i

If we introduce the slack variable t, we can rewrite this problem as a second order cone program:

min t
subject to:
$\|\theta\| \leq t$
$y_i \theta^T x_i \geq 1$ for all i

For the case of non-separable data, we must relax the constraint. We allow the constraints to be met within ϵ_i and punish the objective for using ϵ_i:

min $\|\theta\| + C \sum \epsilon_i$
subject to :
$y_i \theta^T x_i \geq 1 - \epsilon_i$ for all i
$\epsilon_i \geq 0$.

In practice, the constant term C is a hyper-parameter that is used to adjust the trade off between a large margin vs. mis-classifications.

Once again this can be turned into a second order program by using a slack variable t.

2.4 Positive Semi-Definite Programming

The most general conic program is an SDP (Semi-Definite-Program), where the structural constraint is replaced by the semi-positive cone. This is currently the state of the art in convex optimization. Indeed, we have seen that the second order cone is a subset of the semi-positive definite cone, and that both linear and quadratic programs can be formulated as second order cone program.

2.4.1 A Standard Form

Recall the conic standard form:

min $c^T x$
such that:
$Ax = B$
$x \in K$, where K is a convex cone.

If we Take $K = S_+^N$, then each x_i becomes a element of the matrix.

Recall also that we are allowed to take the direct product of cones ($S_+^N X S_+^N$). This can always be reduced to the case of a single cone S_+^{2N}. Specifically, $X_1, X_2 \in S_+^N$, if and only if $diag(X_1, X_2) \in S_+^{2N}$.

Consider a single affine equality constraint $ax = b$ (where $x_i \in X \in S^N$).

This constraint can be rewritten as Trace$(AX) = b$ for an appropriate matrix b. Note that Trace(XY) is the standard dot product on S^N, so we write it as $< X, Y >$. This allows us to conceptualize an SDP in terms of the matrix variable X:

min $< CX >$
subject to:
$< A_i, X >= b_i$ for all i
$X \succeq 0$

2.4.2 Quadratic Boolean Optimization

SDP's have shown to be invaluable tools to approximate *boolean* problems– problems where the decision variable x is restricted to lie in $\{0, 1\}^N$.

For example, consider the problem of finding the maximum cut in a graph. We would like to find two edge sets, E_a and E_b, such that the sum of the edge weights w_{ij} between E_a and E_b is maximized. This problem is well known to be NP-Complete. It can be posed as the following optimization problem:

max $\frac{1}{2} \sum w_i j(1 - x_i x_j)$
such that:
$x \in \{-1, 1\}^N$

Notice that there are two troubling qualities of this problem: we are *maximizing* a quadratic function and we are restricting our variables to lie in a obviously non-convex discrete space.

Many people tried finding a non-trivial approximation to this problem without success. Finally, in 1995, Goemans and Williamson found an SDP relaxation and proved that it achieves a solution within 15 percent of optimal.

Here we consider the general problem of maximizing a quadratic subject to boolean constraints (which approximates the max-cut problem as a special case):

max $x^T Ax$
subject to:
$x \in \{0, 1\}^N$

Consider the optimization problem :

max $< AX >$
subject to:

$X \succeq 0$
$X_i i = 1$ for all i
$\text{rank}(X) = 1$

The problems are equivalent. Indeed, by the spectral decomposition theorem X can be written as a sum of dyads $\sum \lambda_i v_i v_i^T$, where v_i is an eigenvector and λ_i is its eigenvalue. Since X is rank 1, this implies that this sum only has 1 term, and hence X must be of the form vv^T. Together with the fact that $X_{ii} = 1$ for all i, this implies that $v \in \{-1, 1\}$.

Furthermore, since $x^T Ax = \text{Trace}(Ax^T x) = < AX >$, we see that the objective are the same as well. This is rather surprising, as in one formulation we have a quadratic constraint and in one we have a linear constraint.

If we eliminate the rank constraint, we get an SDP. Indeed, the objective is linear, we have linear equality constraints, and the matrix variable X is restricted to be in the semi-positive definite cone. This is called the *rank relaxation*.

Let α be the true solution value. Let β be the approximation value. In 1996, Nesterov showed that this approximation satisfies: $\frac{\beta - \alpha}{\beta} \leq \frac{\pi}{2} - 1$