# DC Programming: The Optimization Method You Never Knew You Had To Know

May 18, 2012

# 1 Introduction

## 1.1 What is DC Programming?

Recall that in the lectures on Support Vector Machines and Kernels, we repeatedly relied on the use of convex optimization to ensure that solutions existed and could be computed. As we shall see later in the lecture, there are many cases in which the assumption that the objective function and constraints are convex (or quasi-convex) are invalid, and so the methods on convex functions that we have developed in class prove insufficient.

To deal with these problems, we develop a theory of optimization for a superclass of convex functions, called DC - Difference of Convex - functions. We now define such functions formally.

**Definition 1.1.** Let $f$ be a real valued function mapping $\mathbb{R}^n$ to $\mathbb{R}$. Then $f$ is a *DC function* if there exist convex functions, $g, h : \mathbb{R}^n \to \mathbb{R}$ such that $f$ can be decomposed as the difference between $g$ and $h$:

$$f(\mathbf{x}) = g(\mathbf{x}) - h(\mathbf{x}) \ \ \forall \mathbf{x} \in \mathbb{R}^n$$

In the remainder of this lecture, we will discuss the solutions to the following - the DC Programming Problem (DCP):

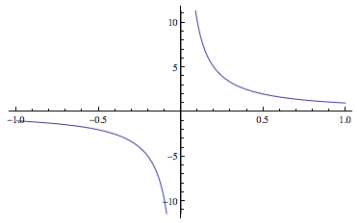
$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f_0(\mathbf{x}) \\ \text{subject to} \quad & f_i(\mathbf{x}) \leq 0, \ i = 1, \ldots, m. \end{aligned} \tag{1}$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ is a differentiable DC function for $i = 0, \ldots, m$.

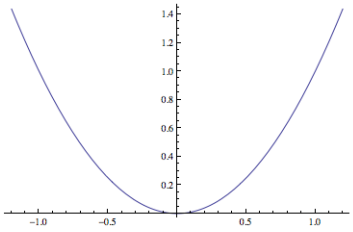
## 1.2 Some Intuition About DC Functions

Before we continue with the discussion of the solution to (1), we develop some intuition regarding DC functions. Recall that a function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if for every $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$ and every $\alpha \in [0, 1]$, $f(\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2) \leq \alpha f(\mathbf{x}_1) + (1-\alpha)f(\mathbf{x}_2)$. In particular, as you may recall, if $f$ is a twice-differential function, then it is convex if and only if its Hessian matrix is positive-semidefinite. To get a sense of what DC functions can look like, we will look at some common convex functions and the DC functions they can form.

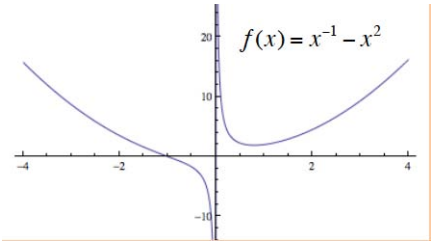
**Example 1.2.** Consider the convex functions $f_1(x) = \frac{1}{x}$ and $f_2(x) = x^2$.



(a) $f_1(x) = \frac{1}{x}$

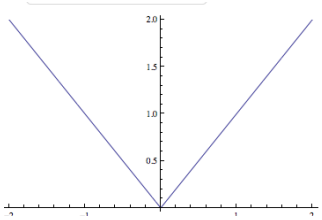


(b) $f_2(x) = x^2$

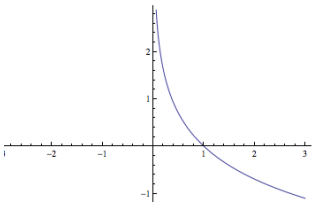



(c) $f = \frac{1}{x} - x^2$

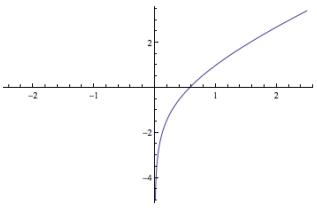
**Example 1.3.** Consider the convex functions $f_1(x) = \text{abs}(x)$ and $f_2(x) = -log(x)$.



(d) $f_1(x) = \text{abs}(x)$



(e) $f_2(x) = -log(x)$



(f) $f = \text{abs}(x) + log(x)$

Notice that while in these examples, the minimum is easy to find by inspection in the convex functions, it is less clear in the resulting DC function. Clearly, then, having DC functions as part of an optimization problem adds a level of complexity to the problem that we did not encounter in our dealings with convex functions. Fortunately, as we shall soon see, this complexity is not unsurpassable.

## 1.3 How Extensive Are These Functions?

### 1.3.1 Hartman

**Theorem 1.4.** *The three following formulations of a DC program are equivalent:*

1. *sup*$\{f(x) : x \in C\}, f, C$ *convex*
2. *inf*$\{g(x) - h(x) : x \in \mathbb{R}^n\}, g, h$ *convex*
3. *inf*$\{g(x) - h(x) : x \in C, f_1(x) - f_2(x) \leq 0\}, g, h, f_1, f_2, C$ *all convex.*

*Proof.*

- We will show how to go from formulation (1) to formulation (2). Define an indicator function to be:

$$I_C(x) = \begin{cases} 0 & \text{if } x \in C \\ \infty & \text{otherwise} \end{cases} \tag{2}$$

  Then $\sup\{f(x) : x \in C\} = \inf\{I_C(x) - f(x) : x \in \mathbb{R}^n\}$.

- We will show how to go from formulation (3) to formulation (1). We have that $\inf\{g(x) - h(x) : x \in C, f_1(x) - f_2(x) \leq 0\}, g, h, f_1, f_2, C$ all convex. Then we can write:

$$\alpha_t = \inf\{g(x) + t\max\{f_1(x), f_2(x)\} - h(x) - tf_2(x) : x \in C\}$$

  for some value of $t$ such that $\alpha = \alpha_{t'}$ for all $t' > t$. It can be shown that such a $t$ always exists.

- Finally, it is clear that (2) is a special case of (3). Thus, we have shown conversions (1) $\rightarrow$ (2) $\rightarrow$ (3) $\rightarrow$ (1), indicating that the three formulations are equivalent.

□

Next we demonstrate just how large the class of DC functions is.
**Theorem 1.5** (Hartman)**.** *A function $f$ is locally DC if there exists an $\varepsilon$-ball on which it is DC. Every function that is locally DC, is DC.*

**Proposition 1.6.** *Let $f_i$ be DC functions for $i = 1, \ldots, m$. Then the following are also DC:*

1. $\sum_i \lambda_i f_i(x)$, *for* $\lambda_i \in \mathbb{R}$
2. $max_i f_i(x)$

3. $min_i f_i(x)$
4. $\prod_i f_i(x)$
5. *$f_i$, twice-continuously differentiable*
6. *If $f$ is DC and $g$ is convex, then the composition $(g \circ f)$ is DC.*
7. *Every continuous function on a convex set, $C$ is the limit of a sequence of uniformly converging DC functions.*

# 2 Optimality Conditions

## 2.1 Duality

Before we can discuss the conditions for global and local optimality in the canonical DC Programming problem, we need to introduce some notions regarding duality in the DCP, and develop some intuition of what this gives us. To begin with this, we introduce conjugate functions and use them to demonstrate the relationship between the DCP and its dual.

**Definition 2.1.** Let $g : \mathbb{R}^n \to \mathbb{R}$. Then the conjugate function of $g(x)$ is

$$g^*(y) = \sup\{x^T y - g(x) : x \in \mathbb{R}^n\}$$

To understand what why conjugate functions are so important, we need just one more definition.
**Definition 2.2.** The *epigraph* of a function, $g : \mathbb{R}^n \to \mathbb{R}$ is the set of points lying on or on top of its graph:

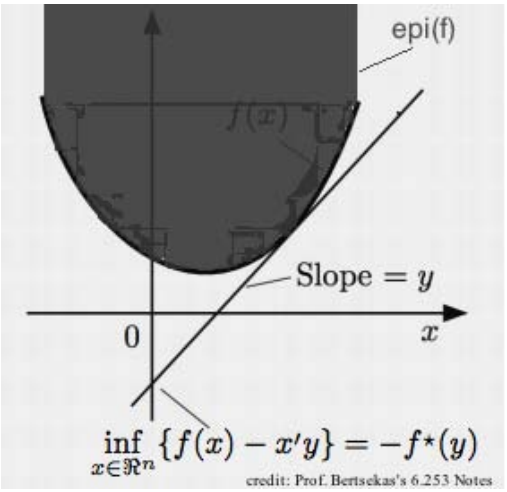
$$epi(g) = \{(x, t) \in \mathbb{R}^n \times \mathbb{R} : g(x) \leq t\}$$

Note that $g$ is convex if and only if $epi(g)$ is a convex set.

Having defined the epigraph, we can now give a geometric interpretation of the conjugate function: The conjugate function $g^*$ 'encloses' the convex hull of $epi(g)$ with $g$'s supporting hyperplanes. In particular, we can see that when $f$ is differentiable,

$$\frac{df}{dx} = \text{argsup}\{x^T y - g^*(y) : y \in \mathbb{R}^n\} = y(x)$$

That is, $y$ is a dual variable for $x$, and so can be interpreted as (approximately) the gradient (or slope in $\mathbb{R}^2$) of $f$ at $x$. This should be a familiar result from the convex analysis lecture.




Courtesy of Dimitri Bertsekas. Used with permission.

(g) A Convex fn $f(x)$ and the affine function meeting $f$ at $x$. The conjugate, $f^*(y)$ of $f(x)$ is the point of intersection.

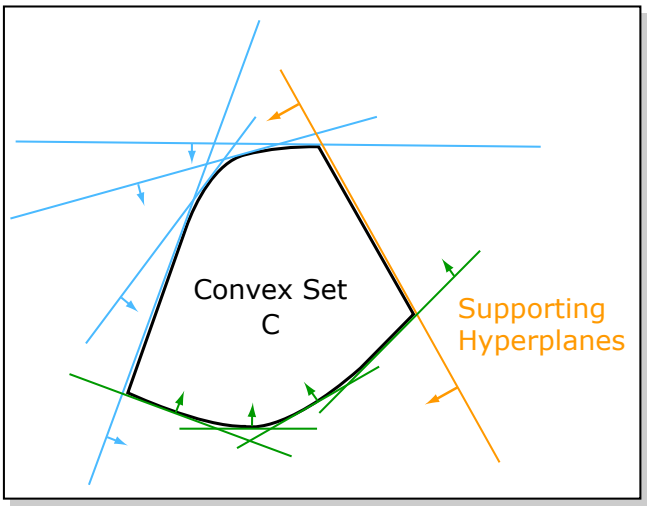



Image by MIT OpenCourseWare.

(h) A convex set being enclosed by supporting hyperplanes. If C is the epigraph of a function $f^*$, then the intersections of C with the hyperplanes is the set of values that $f^*$ takes.

**Theorem 2.3.** *Let $g : \mathbb{R}^n \to \mathbb{R}$ such that $g(x)$ is lower semi-continuous and convex on $\mathbb{R}^n$. Then*

$$g(x) = sup\{x^T y - g^*(y) : y \in \mathbb{R}^n\}$$

*where $g^*(y)$ is the conjugate of $g(x)$.*

We provide this theorem without proof, and omit further discussion of lower semi-continuity, but we can safely assume that the functions that we will deal with satisfy this. Note that this condition implies that $g^{**} = g$, that is, the conjugate of $g$ conjugate is $g$, which means that the definitions of $g$ and $g^*$ are symmetric. (What does this mean for minimizing $g$?)

We now demonstrate the relationship between DCP and its dual problem. First note the form of the conjugate function $f^*$ for $f \equiv g - h$:

$$(f(x))^* = ((g-h)(x))^* = \sup\{x^T y - (g-h)(x) : x \in \mathbb{R}^n\} = h^*(y) - g^*(y)$$

Let $\alpha$ be the optimum value to DCP.

$$\alpha = \inf\{g(x) - h(x) : x \in X\} = \inf\{g(x) - \sup\{x^T y - h^*(y) : y \in Y\} : x \in X\}$$

$$= \inf\{\inf\{g(x) - x^T y + h^*(y) : x \in X\} : y \in Y\}$$

$$= \inf\{h^*(y) - g^*(y) : y \in Y\}$$

Thus, we have that the optimal value to DCP is the same as the optimal value for its dual! Now that is symmetry! This means that we can solve either the primal or the dual problem and obtain the solution to both - the algorithm that we will employ to solve the DCP will crucially rely on this fact.

## 2.2 Global Optimality Conditions

**Definition 2.4.** Define an $\varepsilon$-subgradient of $g$ at $x^0$ to be

$$\partial_\varepsilon g(x^0) = \{y \in \mathbb{R}^n : g(x) - g(x^0) \geq (x - x^0)^T y - \varepsilon \ \ \forall x \in \mathbb{R}^n\}$$

Define a differential of $g$ at $x^0$ to be

$$\partial g(x^0) = \bigcap_{\varepsilon > 0} \partial_\varepsilon g(x^0)$$

Given these two definitions, we have the following conditions for global optimality:

**Theorem 2.5** (Generalized Kuhn-Tucker)**.** .
*Let $x^*$ be an optimal solution to the (primal) DCP. Then $\partial h(x^*) \subset \partial g(x^*)$.*
*Let $y^*$ be an optimal solution to the dual DCP. Then $\partial g^*(y^*) \subset \partial h^*(y^*)$*

*Proof.* This condition essentially follows from the equivalence of the primal and dual optima. We showed before that if $\alpha$ is the optimum value of the DCP, then

$$\alpha = \inf\{g(x) - h(x) : x \in X\} = \inf\{h^*(y) - g^*(y) : y \in Y\}$$

Then if $\alpha$ is finite, we must have that dom $g \subset$ dom $h$ and dom $h^* \subset$ dom $g^*$ where dom $g = \{x \in \mathbb{R}^n : g(x) < \infty\}$, the domain of $g$. That is, $h$ (respectively, $g^*$) is finite whenever $g$ (respectively, $h^*$) is finite. Note that we require this inclusion because we are *minimizing* the objective function, and so if there existed an $x \in \mathbb{R}^*$ such that $g(x) < \infty, h(x) = \infty$, then $g(x) - h(x)$ would be minimized at $x$, yielding an objective value of $-\infty$. Note also that this statement is not an in and only if statement, as we work under the convention that $\infty - \infty = \infty$.

Thus, we have that if $x^*$ is an optimum to the primal DCP, then $x^* \in$ dom $g$, and by weak duality,

$$g(x^*) - h(x^*) \leq h^*(y) - g^*(y), \ \ \forall y \in \text{dom } h^*$$

and so (for $x^* \in$ dom h) if $x^* \in \partial h(x^*)$, then

$$x^T y \geq h(x^*) + h^*(y) \geq g(x^*) + g^*(y)$$

where the first inequality is by definition of $\partial h(x^*)$ and the second inequality follows from the weak duality inequality presented. □

We can also think of this in terms of the interpretations of the dual problem as well. We have that if $g, h$ are differentiable, and so $\partial h(x^*) \neq \emptyset \neq \partial g(x^*)$, then $\partial h(x^*)$ is just the set of gradients of $h$ at $x^*$, and by equality of the primal and dual optima, it is the set of $y^*$ that optimize the dual problem. Thus, this optimality condition in terms of subdifferentials is analogous to the one we discussed in terms of domains.

**Corollary 2.6.** *Let $\mathbb{P}$ and $\mathbb{D}$ be the solution sets of the primal and dual problems of the DCP, respectively. Then:*
*$x^* \in \mathbb{P}$ if and only if $\partial_\varepsilon h(x^*) \subset \partial_\varepsilon g(x^*)\ \forall \varepsilon > 0$.*
*$y^* \in \mathbb{D}$ if and only if $\partial_\varepsilon g^*(y^*) \subset \partial_\varepsilon h^*(y^*)\ \forall \varepsilon > 0$.*

**Theorem 2.7.** *Let $\mathbb{P}$ and $\mathbb{D}$ be the solution sets of the primal and dual problems of the DCP, respectively. Then:*

$$\bigcup\{\partial h(x) : x \in \mathbb{P}\} \subset \mathbb{D} \subset \textit{dom } h^*$$

*and*

$$\bigcup\{\partial g^*(y) : y \in \mathbb{D}\} \subset \mathbb{P} \subset \textit{dom } g$$

Note that this theorem implies that solving the primal DCP implies solving the dual DCP.

## 2.3 Local Optimality Conditions

We would like to construct an algorithm to find global optimal solutions based on the conditions discussed in the previous section. However, finding an algorithm that does this efficiently in general is an open problem, and most approaches are combinatorial, rather than convex-based, and so rely heavily on the formulation of a given problem, and are often inefficient. Thus, we present local optimality conditions, which (unlike the global optimality conditions) can be used to create a convex-based approach to local optimization. We present these theorems without proof as, although they are crucially important to solving DC programming

problems, their proofs do not add much more insight. Thus, we refer further investigation to either Hurst and Thoai or Tao and An.

**Theorem 2.8** (Sufficient Local Optimality Condition 1)**.** *Let $x^*$ be a point that admits a neighborhood $U(x)$ such that*

$$\partial h(x) \cap \partial g(x^*) \neq \emptyset \quad \forall x \in U(x) \cap dom\ g$$

*Then $x^*$ is a local minimizer of $g - h$.*

**Theorem 2.9** (Sufficient Local Optimality Condition 2: Strict Local Optimality)**.** *Let $int(S)$ refer to the interior of set $S$. Then if $x^* \in int(dom\ h)$ and $\partial h(x^*) \subset int(\partial g(x^*))$, then $x^*$ is a strict local minimizer of $g - h$.*

**Theorem 2.10** (DC Duality Transportation of a Local Minimizer)**.** *Let $x^* \in dom\ \partial h$ be a local minimizer of $g - h$ and let $y^* \in \partial h(x^*)$. Then if $g^*$ is differentiable at $y^*, y^*$ is a local minimizer of $h^* - g^*$. More generally, if $y^*$ satisfies Theorem 2.9, then $y^*$ is a local minimizer of $h^* - g^*$.*

# 3 Algorithms

As discussed earlier, the conditions for global optimality in DC programs do not wield efficient general algorithms. Thus, while there are a number of popular techniques - among them, branch-and-bound and cutting planes algorithms, we omit discussion of them and instead focus on the convex-based approach to local optimization. In fact, although there has not been an analytic result to justify this, according to the DC Programming literature, the local optimization approach often yields the global optimum, and a number of regularization and starting-point choosing methods exist to assist with incorporating the following local optimization algorithm to find the global optimum in different cases.

## 3.1 DCA-Convex Approach to Local Optimization

We now present an algorithm to find local optima for a general DC program. First, we offer the algorithm in raw form, then we will explain each step in an iteration, and finally, we will state a few results regarding the effectiveness and efficiency of the algorithm.

### 3.1.1 DCA

1. Choose $x_0 \in \text{dom } g$
2. **for** $k \in \mathbb{N}$ **do**:
3. choose $y_k \in \partial h(x_k)$
4. choose $x_{k+1} \in \partial g^*(y_k)$
5. **if** $\min\{|(x_{k+1} - x_l)_i|, |\ \frac{(x_{k+1}-x_l)_i}{(x_k)_i}|\} \leq \delta$:
6. then **return** $x_{k+1}$
7. **end if**
8. **end for**

### 3.1.2 DCA Explanation and Intuition

Let us go over each step of the DCA algorithm in more detail. The overarching method of the algorithm is to create two sequences of variables, $\{x_k\}_k, \{y_k\}_k$ so that $\{x_k\}$ converges to a local optimum of the primal problem, $x^*$, and $\{y_k\}$ converges to the local optimum of the dual problem, $y^*$. The key idea is to manipulate the symmetry of the primal and dual problem in order to follow a variation on the typical sub gradient-descent method used in convex optimization.

Let us now consider each step individually.

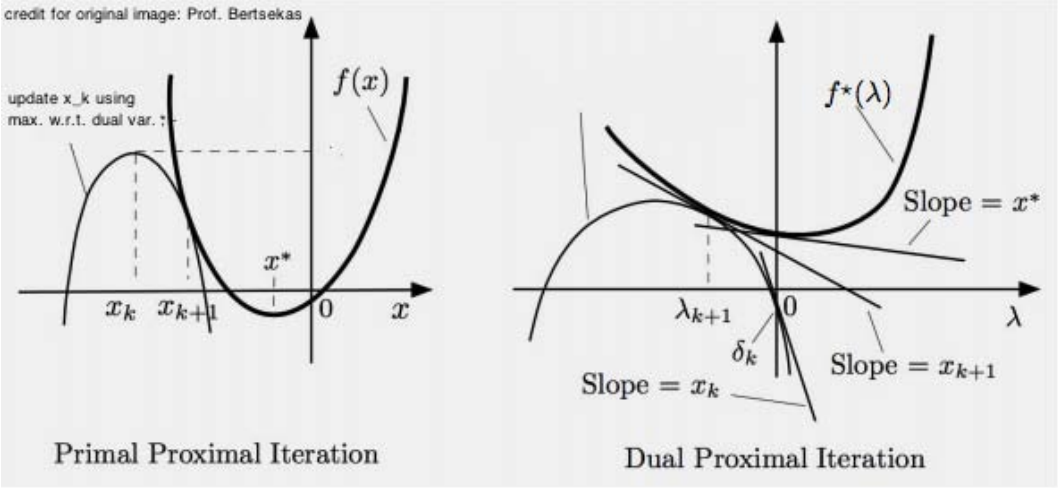
- Choose $x_0 \in \text{dom } g$:
  Since we are utilizing a descent approach, the convergence of the algorithm is independent of the starting point of the sequences that the algorithm creates. Thus, we can instantiate the algorithm with an arbitrary choice of $x_0$ so long as it is feasible.
- Choose $y_k \in \partial h(x_k)$:
  We have that $\partial h(x_k) = \arg\min\{h^*(y) - g^*(y_{k-1}) - x_k^T(y - y_{k-1}) : y \in \mathbb{R}^n\}$. Moreover, since this is a minimization over $y$, we hold $y_{k-1}, x_k$ constant, and so we have that $\partial h(x_k) = \arg\max\{x_k^T y - h^*(y) : y \in \mathbb{R}^n\}$. Computing this, however is just an exercise in convex optimization, since by the local optimality conditions, $x_k$ is (approximately) a subgradient of $h^* - g^*$, and so serves a role similar to that in a typical subgradient descent algorithm, which we solve quickly and efficiently. Since we maximize given $x_k$ (which improves together with $y_{k-1}$), we guarantee that $(h^* - g^*)(y_k - y_{k-1}) \leq 0\ \forall k \in \mathbb{N}$, and

due to the symmetry of duality, that $y_k$ converges to a critical point of $(h^* - g^*)$, i.e., a local minimizer.

- Choose $x_{k+1} \in \partial g^*(y_k)$:
  Given the symmetry of the primal and dual problem, this is entirely symmetric to the step finding $y_k$. Thus, we choose $x_{k+1} \in \arg\max\{x^T y_k - g(x) : x \in \mathbb{R}^n\}$.

- **If** $\min\{|(x_{k+1} - x_l)_i|, |\frac{(x_{k+1}-x_l)_i}{(x_k)_i}|\} \leq \delta$: then **return** $x_{k+1}$:
  Although we can guarantee convergence in the infinite limit of $k$, complete convergence may take a long time, and so we approximate an optimal solution within a predetermined bound, $\delta$. Once the solution (the change in $x_k$ or $y_k$) is small enough, we terminate the algorithm and return the optimal value for $x_{k+1}$ - recall that solving for $x_k$ is equivalent to solving for $y_k$.

Figure 1: An example for the Proximal Point subgradient descent method. This has been shown to be equivalent to a regularized version of DCA, and so offers valuable intuition into how DCA works.




Courtesy of Dimitri Bertsekas. Used with permission.

### 3.1.3 Well-Definition and Convergence

Here we present a few results regarding the effectiveness and efficiency of the DCA algorithm. We will give the results without proof, and direct anyone interested in further delving into this matter to Thoai.

**Lemma 3.1.** *The sequences* $\{x_k\}$ *and* $\{y_k\}$ *are well defined if and only if*

$$dom\ \partial g \subset dom\ \partial h \qquad dom\ \partial h^* \subset dom\ \partial g^*$$

**Lemma 3.2.** *Let* $h$ *be a lower semi-continuous function on* $\mathbb{R}^n$ *and* $\{x_k\}$ *be a sequence of elements in* $\mathbb{R}^n$ *such that (i)* $x_k \to x^*$*; (ii) There exists a bounded*

*sequence* $\{y_k\}$ *such that* $y_k \in \partial h(x_k)$*; (iii)* $\partial h(x^*) \neq \emptyset$*. Then*

$$\lim_{k \to \infty} h(x_k) = h(x^*)$$

# 4 Applications to Machine Learning

The literature has references to many uses of DC Programming in Operations Research, Machine Learning, and Economics.

One interesting use of DC Programming is discussed in the 2006-paper "A DC-Programming Algorithm for Kernel Selection". In this paper, the authors discuss a greedy algorithm to *learn* a kernel from a convex hull of basic kernels. While this approach had been popularized before, it was limited to a finite set of basic kernels. The authors comment that the limitation was due to the non-convexity of a critical maximization involved in conducting the learning, but find that the optimization problem can be formulated as a DC Program. In particular, the objective function used to weigh basic kernels is DC as the limit of DC functions.

Another interesting use of DC Programming is discussed in the 2008-paper "A DC programming approach for feature selection in support vector machines learning". Here, DC Programming is employed in an SVM algorithm that attempts to choose optimally representative features in data while constructing an SVM classifier simultaneously. The authors equate this problem to minimizing a zero-norm function over step-$k$ feature vectors. Using the DC-decomposition displayed below, the authors employ the DCA algorithm to find local minima and applied it to ten datasets - some of which were particularly sparse - to find that the DCA algorithm created consistently good classifiers that often had the highest correctness rate among the tested classifiers (including standard SVMs, for example). Despite this, DCA consistently used less features than the standard SVM and other classifiers, and as a result, was more efficient and used less CPU capacity than many of the other commonplace classifiers. Thus, all in all, DCA proved a greatly attractive algorithm for classifying data, and particularly excellent for very large and sparse datasets.

# 5 Conclusion

As we can see, DC Programming is quite young. Although the oldest result presented in this lecture dates back to the 1950s (Hartman's), many of the results and algorithms discussed here were developed in the 1990s, and their applications are still very new to the scientific community. However, as can be seen from the examples that we presented, DC Programming has tremendous potential to expand and expedite many of the algorithms and techniques that are central to Machine Learning, as well as other fields. Thus, it is likely that the near future will bring many more algorithms inspired by DCA and DCA-related approaches as well as the various combinatoric global approaches that are used to solve DC Programming problems, but were not discussed here.

# References


[1] H. A. L. T. H. M. L. V. V. Nguyen and T. P. Dinh. A dc programming approach for feature selection in support vector machines learning. *Advances in Data Analysis and Classification*, 2(3):259–278, 2008.

[2] A. A. R. H. C. A. M. M. Pontil. A dc-programming algorithm for kernel selection. *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

[3] P. D. TAO and L. T. H. AN. Convex analysis approach to d. c. programming: Theory, algorithms and applications. *ACTA MATHEMATICA VIETNAMICA*, 22(1):289–355, 1997.

[4] R. H. N. Thoai. Dc programming: An overview. *Journal of Optimization Theory and Application*, 193(1):1–43, October 1999.

15.097 Prediction: Machine Learning and Statistics
Spring 2012