

15-780 – Graduate Artificial Intelligence: Optimization

J. Zico Kolter (this lecture) and Ariel Procaccia
Carnegie Mellon University
Spring 2018

Outline

Introduction to optimization

Types of optimization problems, convexity

Solving optimization problems

Logistics

Everyone who submitted HW0 and received a sufficient high score (not necessarily full credit), has been taken off the waitlist

Solutions to HW0 posted to Piazza (do not post these publicly anywhere)

Outline

Introduction to optimization

Types of optimization problems, convexity

Solving optimization problems

Continuous optimization

The problems we have seen so far (i.e., search) in class involve making decisions over a *discrete* space of choices

An amazing property:

	Discrete search	(Convex) optimization
Variables	Discrete	Continuous
# Solutions	Finite	Infinite
Solution complexity	Exponential	Polynomial

One of the most significant trends in AI in the past 15 years has been the integration of optimization methods throughout the field

Optimization definitions

We'll write optimization problems like this:

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

which should be interpreted to mean: we want to find the value of x that achieves the smallest possible value of $f(x)$, out of all points in \mathcal{C}

Important terms:

$x \in \mathbb{R}^n$ – optimization variable (vector with n real-valued entries)

$f: \mathbb{R}^n \rightarrow \mathbb{R}$ – optimization objective

$\mathcal{C} \subseteq \mathbb{R}^n$ – constraint set

$x^* \equiv \underset{x \in \mathcal{C}}{\operatorname{argmin}} f(x)$ – optimal *solution*

$f^* \equiv f(x^*) \equiv \min_{x \in \mathcal{C}} f(x)$ – optimal *objective*

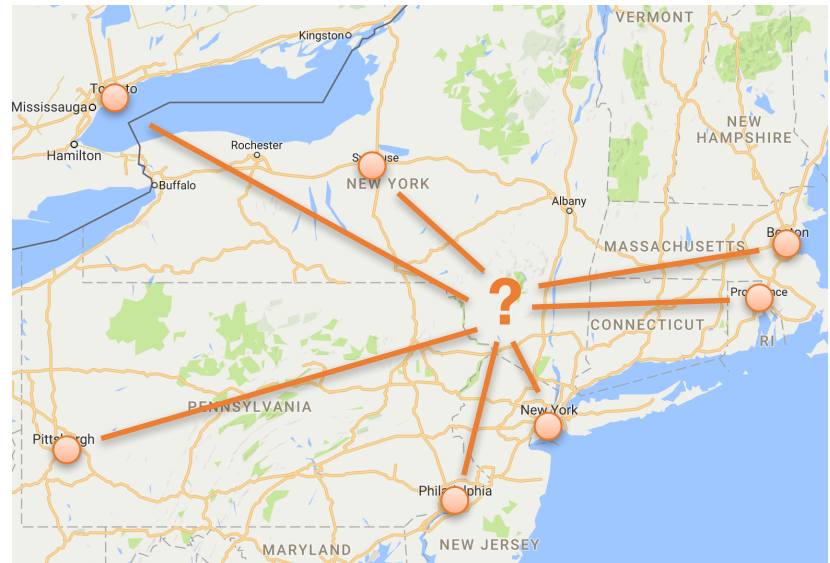
Example: Weber point

Given a collection of cities (assume on 2D plane) how can we find the location that minimizes the sum of distances to all cities?

Denote the locations of the cities as $y^{(1)}, \dots, y^{(m)}$

Write as the optimization problem:

$$\text{minimize}_x \sum_{i=1}^m \|x - y^{(i)}\|_2$$



Example: image deblurring



(a) Original image.



(b) Blurry, noisy image.



(c) Restored image.

Figure from (O'Connor and Vandenberghe, 2014)

Given corrupted image $Y \in \mathbb{R}^{m \times n}$, reconstruct image by solving optimization problem:

$$\underset{X}{\text{minimize}} \sum_{i,j} |Y_{ij} - (K * X)_{ij}| + \lambda \sum_{i,j} \left((X_{ij} - X_{i,j+1})^2 + (X_{i+1,j} - X_{ij})^2 \right)^{\frac{1}{2}}$$

where $K *$ denotes convolution with a blurring filter

Example: robot trajectory planning

Many robotic planning tasks are more complex than shortest path, e.g. have robot dynamics, require “smooth” controls

Common to formulate planning problem as an optimization task

Robot state x_t and control inputs u_t

$$\begin{aligned} & \underset{x_{1:T}, u_{1:T-1}}{\text{minimize}} && \sum_{i=1}^T \|u_t\|_2^2 \\ & \text{subject to} && x_{t+1} = f_{\text{dynamics}}(x_t, u_t) \\ & && x_t \in \text{FreeSpace}, \forall t \\ & && x_1 = x_{\text{init}}, x_T = x_{\text{goal}} \end{aligned}$$

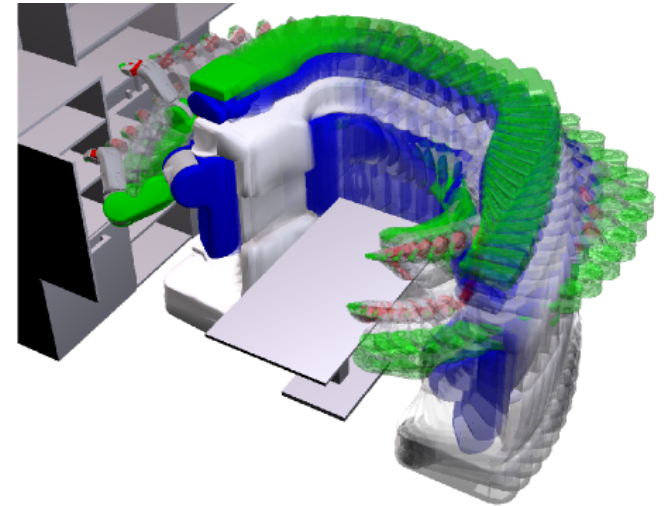


Figure from (Schulman et al., 2014)

Example: machine learning

As we will see in much more detail shortly, virtually all (supervised) machine learning algorithms boil down to solving an optimization problem

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^m \ell(h_{\theta}(x^{(i)}), y^{(i)})$$

Where $x^{(i)} \in \mathcal{X}$ are inputs, $y^{(i)} \in \mathcal{Y}$ are outputs, ℓ is a loss function, and h_{θ} is a hypothesis function parameterized by θ , which are the parameters of the model we are optimizing over

Much more on this soon

The benefit of optimization

One of the key benefits of looking at problems in AI as optimization problems: we separate out the *definition* of the problem from the *method for solving it*

For many classes of problems, there are off-the-shelf solvers that will let you solve even large, complex problems, once you have put them in the right form

Outline

Introduction to optimization

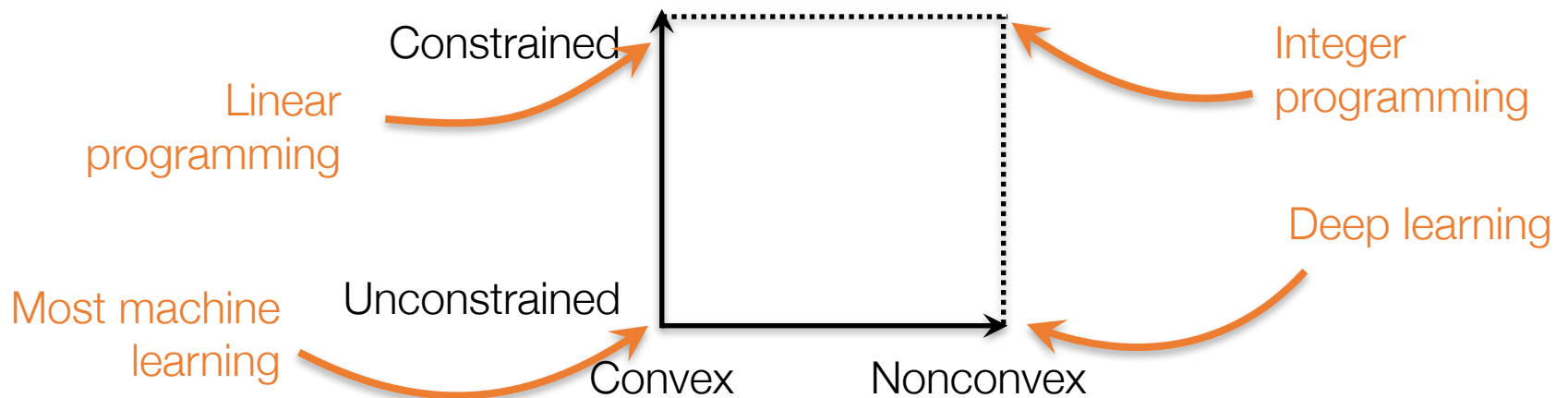
Types of optimization problems, convexity

Solving optimization problems

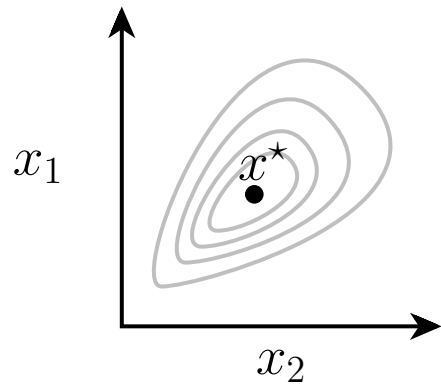
Classes of optimization problems

Many different names for types of optimization problems: linear programming, quadratic programming, nonlinear programming, semidefinite programming, integer programming, geometric programming, mixed linear binary integer programming (the list goes on and on, can all get a bit confusing)

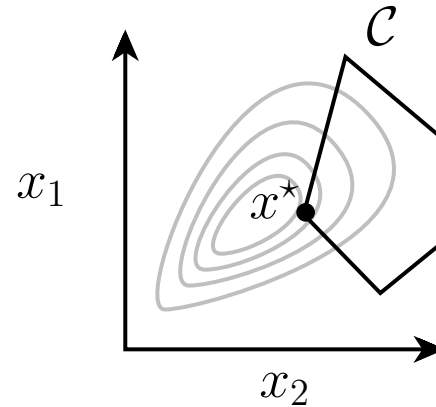
We're instead going to focus on two dimensions: convex vs. nonconvex and constrained vs. unconstrained



Constrained vs. unconstrained



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



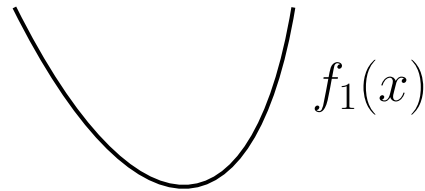
$$\underset{x}{\text{minimize}} \quad f(x)$$
$$\text{subject to } x \in \mathcal{C}$$

In unconstrained optimization, every point $x \in \mathbb{R}^n$ is feasible, so singular focus is on minimizing $f(x)$

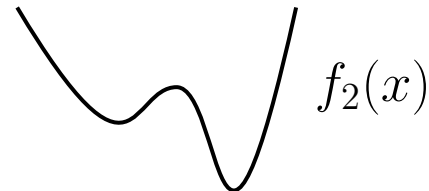
In contrast, for constrained optimization, it may be difficult to even *find* a point $x \in \mathcal{C}$

Often leads to very different methods for optimization (more next lecture)

Convex vs. nonconvex optimization



Convex function



Nonconvex function

Originally, researchers distinguished between linear (easy) and nonlinear (hard) problems

But in 80s and 90s, it became clear that this wasn't the right distinction, key difference is between convex and nonconvex problems

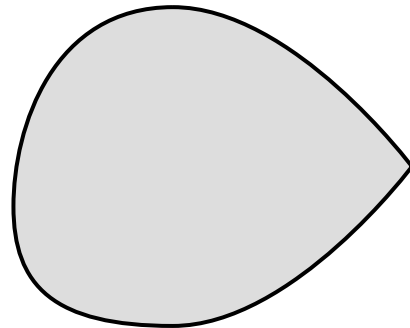
Convex problem:

$$\begin{aligned} & \underset{x}{\text{minimize}} && f(x) \\ & \text{subject to} && x \in \mathcal{C} \end{aligned}$$

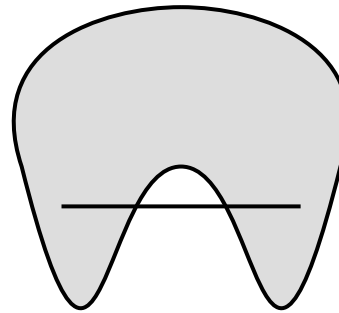
Where f is a convex function and \mathcal{C} is a convex set

Convex sets

A set \mathcal{C} is convex if, for any $x, y \in \mathcal{C}$ and $0 \leq \theta \leq 1$
$$\theta x + (1 - \theta)y \in \mathcal{C}$$



Convex set



Nonconvex set

Examples:

All points $\mathcal{C} = \mathbb{R}^n$

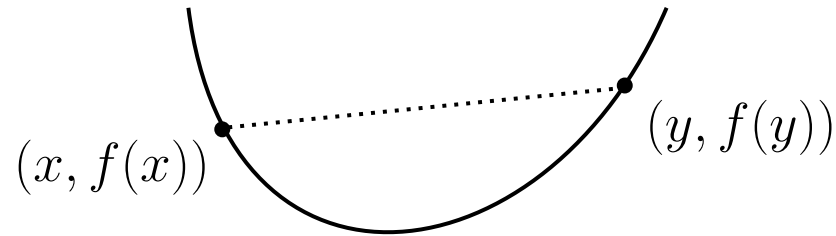
Intervals $\mathcal{C} = \{x \in \mathbb{R}^n \mid l \leq x \leq u\}$ (elementwise inequality)

Linear equalities $\mathcal{C} = \{x \in \mathbb{R}^n \mid Ax = b\}$ (for $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$)

Intersection of convex sets $\mathcal{C} = \bigcap_{i=1}^m \mathcal{C}_i$

Convex functions

A function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is convex if, for any $x, y \in \mathbb{R}^n$ and $0 \leq \theta \leq 1$

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


Convex functions “curve upwards” (or at least not downwards)

If f is convex then $-f$ is concave

If f is both convex and concave, it is affine, must be of form:

$$f(x) = \sum_{i=1}^n a_i x_i + b$$

Examples of convex functions

Exponential: $f(x) = \exp(ax)$, $a \in \mathbb{R}$

Negative logarithm: $f(x) = -\log x$, with domain $x > 0$

Squared Euclidean norm: $f(x) = \|x\|_2^2 \equiv x^T x \equiv \sum_{i=1}^n x_i^2$

Euclidean norm: $f(x) = \|x\|_2$

Convex function of affine function $f(x) = g(Ax + b)$, g convex

Non-negative weighted sum of convex functions

$$f(x) = \sum_{i=1}^m w_i f_i(x), \quad w_i \geq 0, f_i \text{ convex}$$

Poll: convex sets and functions

Which of the following functions or sets are convex

1. A union of two convex sets $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$
2. The set $\{x \in \mathbb{R}^2 \mid x \geq 0, x_1 x_2 \geq 1\}$
3. The function $f: \mathbb{R}_+^2 \rightarrow \mathbb{R}, f(x) = x_1 x_2$ with domain $x > 0$
4. The function $f: \mathbb{R}^2 \rightarrow \mathbb{R}, f(x) = x_1^2 + x_2^2 + x_1 x_2$

Convex optimization

The key aspect of convex optimization problems that make them tractable is that *all local optima are global optima*

Definition: a point x is globally optimal if x is feasible and there is no feasible y such that $f(y) < f(x)$

Definition: a point x is locally optimal if x is feasible and there is some $R > 0$ such that for all feasible y with $\|x - y\|_2 \leq R$, $f(x) \leq f(y)$

Theorem: for a convex optimization problem all locally optimal points are globally optimal

Proof of global optimality

Proof: Given a locally optimal x (with optimality radius R), and suppose there exists some feasible y such that $f(y) < f(x)$

Now consider the point

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

1) Since $x, y \in \mathcal{C}$ (feasible set), we also have $z \in \mathcal{C}$ (by convexity of \mathcal{C})

2) Furthermore, since f is convex:

$$f(z) = f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y) < f(x) \quad \text{and}$$
$$\|x - z\|_2 = \left\| x - \left(1 - \frac{R}{2\|x - y\|_2}\right)x + \frac{R}{2\|x - y\|_2}y \right\|_2 = \left\| \frac{R(x - y)}{2\|x - y\|_2} \right\|_2 = \frac{R}{2}$$

Thus, z is feasible, within radius R of x , and has lower objective value, a contradiction of supposed local optimality of x



Outline

Introduction to optimization

Types of optimization problems, convexity

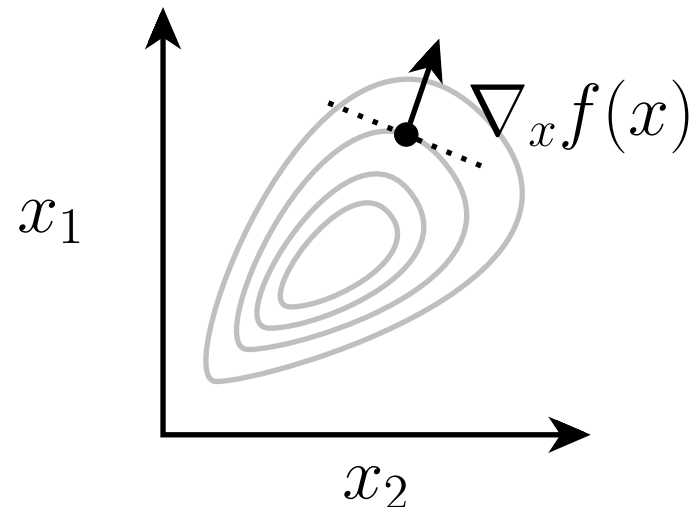
Solving optimization problems

The gradient

A key concept in solving optimization problems is the notation of the gradient of a function (multi-variate analogue of derivative)

For $f: \mathbb{R}^n \rightarrow \mathbb{R}$, gradient is defined as vector of partial derivatives

$$\nabla_x f(x) \in \mathbb{R}^n = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$$



Points in “steepest direction” of increase in function f

Gradient descent

Gradient motivates a simple algorithm for minimizing $f(x)$: take small steps in the direction of the negative gradient

Algorithm: Gradient Descent

Given:

Function f , initial point x_0 , step size $\alpha > 0$

Initialize:

$$x \leftarrow x_0$$

Repeat until convergence:

$$x \leftarrow x - \alpha \nabla_x f(x)$$

“Convergence” can be defined in a number of ways

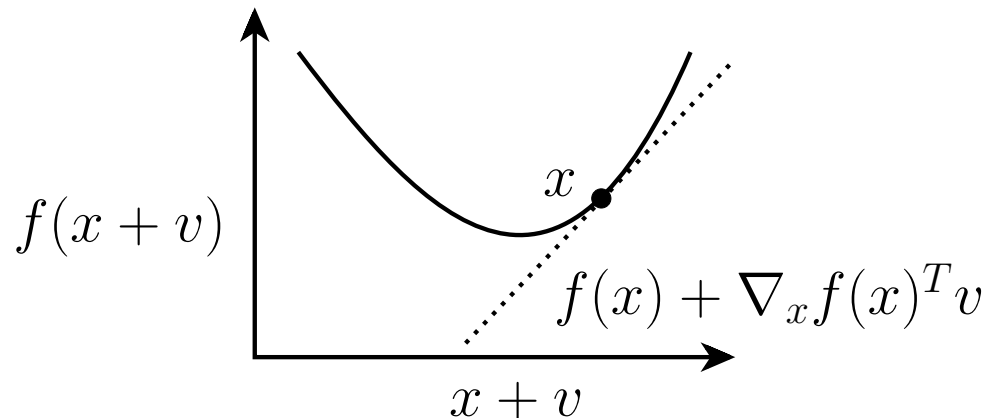
Gradient descent works

Theorem: For differentiable f and small enough α , at any point x that is not a (local) minimum

$$f(x - \alpha \nabla_x f(x)) < f(x)$$

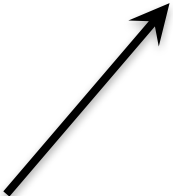
i.e., gradient descent algorithm will decrease the objective

Proof: Any differentiable function f can be written in terms of its *Taylor expansion*: $f(x + v) = f(x) + \nabla_x f(x)^T v + O(\|v\|_2^2)$



Gradient descent works (cont)

Choosing $v = -\alpha \nabla_x f(x)$, we have

$$\begin{aligned} f(x - \alpha \nabla_x f(x)) &= f(x) - \alpha \nabla_x f(x)^T \nabla_x f(x) + O(\|\alpha \nabla_x f(x)\|_2^2) \\ &\leq f(x) - \alpha \|\nabla_x f(x)\|_2^2 + C \|\alpha \nabla_x f(x)\|_2^2 \\ &= f(x) - (\alpha - \alpha^2 C) \|\nabla_x f(x)\|_2^2 \\ &< f(x) \quad (\text{for } \alpha < 1/C \text{ and } \|\nabla_x f(x)\|_2^2 > 0) \end{aligned}$$


(Watch out: a bit of subtlety of this line, only holds for small $\alpha \nabla_x f(x)$)

We are guaranteed to have $\|\nabla_x f(x)\|_2^2 > 0$ except at optima ■

Works for both convex and non-convex functions, but with convex functions guaranteed to find global optimum

Poll: modified gradient descent

Consider an alternative version of gradient descent, where instead of choosing an update $x - \alpha \nabla_x f(x)$, we chose some other direction $x + \alpha v$ where v has a negative inner product with the gradient

$$\nabla_x f(x)^T v < 0$$

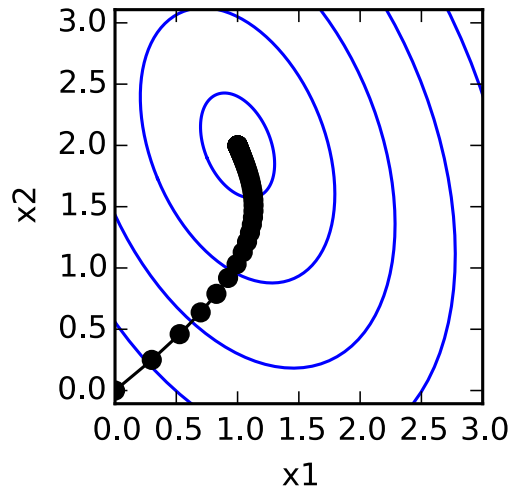
Will this update, for suitably chosen α , still decrease the objective?

1. No, not necessarily (for either convex or nonconvex functions)
2. Only for convex functions
3. Only for nonconvex functions
4. Yes, for both convex and nonconvex functions

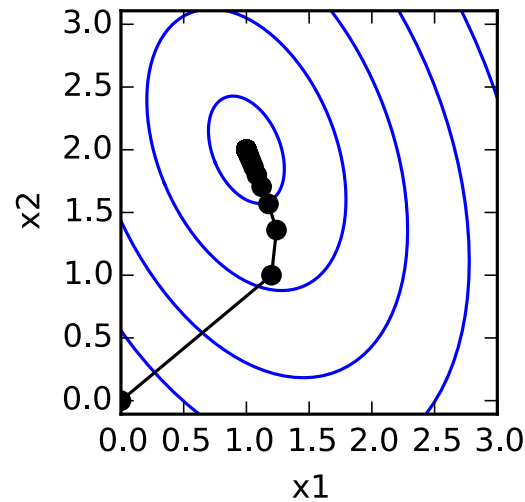
Gradient descent in practice

Choice of α matters a lot in practice:

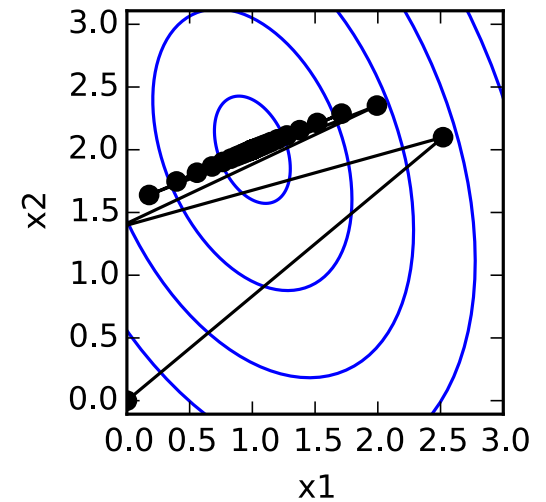
$$\underset{x}{\text{minimize}} \quad 2x_1^2 + x_2^2 + x_1x_2 - 6x_1 - 5x_2$$



$\alpha = 0.05$



$\alpha = 0.2$



$\alpha = 0.42$

Dealing with constraints, non-differentiability

For settings where we can easily project points onto the constraint set \mathcal{C} , can use a simple generalization called *projected gradient descent*

$$\text{Repeat: } x \leftarrow P_{\mathcal{C}}(x - \alpha \nabla_x f(x))$$

If f is not differentiable, but continuous, it still has what is called a *subgradient*, can replace gradient with subgradient in all cases (but theory/practice of convergence is quite different)

Optimization in practice

We won't discuss this too much yet, but one of the beautiful properties of optimization problems is that there exists a wealth of tools that can solve them using very simple notation

Example: solving Weber point problem using cvxpy (<http://cvxpy.org>)

```
import numpy as np
import cvxpy as cp

n,m = (5,10)
y = np.random.randn(n,m)
x = cp.Variable(n)
f = sum(cp.norm2(x - y[:,i]) for i in range(m))
cp.Problem(cp.Minimize(f), []).solve()
```