EECS 545 - Machine Learning

Lecture 3: Convex Optimization + Probability/Stat Overview

Date: January 13, 2016

Instructor: Jacob Abernethy

In [1]:
```
from IPython.core.display import HTML, Image
from IPython.display import YouTubeVideo
from sympy import init_printing, Matrix, symbols, Rational
import sympy as sym
from warnings import filterwarnings
init_printing(use_latex = 'mathjax')
filterwarnings('ignore')

%pylab inline

import numpy as np
```

```
Populating the interactive namespace from numpy and matplotlib
```

Some important notes

- HW1 is out! Due January 25th at 11pm

- Homework will be submitted via *Gradescope*. Please see Piazza for precise instructions. Do it soon, not at the last minute!!

- There is an *optional* tutorial **this evening**, 5pm, in Dow 1013. Come see Daniel go over some tough problems.

- No class on Monday January 18, MLK day!

Python: We recommend Anacdona

- Anaconda is standalone Python distribution that includes all the most important scientific packages: *numpy, scipy, matplotlib, sympy, sklearn, etc.*
- Easy to install, available for OS X, Windows, Linux.
- Small warning: it's kind of large (250MB)

Some notes on using Python

- HW1 has only a very simple programming exercise, just as a warmup. We don't expect you to submit code this time
- This is a good time to start learning Python basics
- There are **a ton** of good places on the web to learn python, we'll post some
- Highly recommended: **ipython**; it's a much more user friendly terminal interface to python
- Even better: **jupyter notebook**, a web based interface. This is how I'm making these slides!

Checking if all is installed, and HelloWorld

If you got everything installed, this should run:

```python
# numpy is crucial for vectors, matrices, etc.
import numpy as np
# Lots of cool plotting tools with matplotlib
import matplotlib.pyplot as plt
# For later: scipy has a ton of stats tools
import scipy as sp
# For later: sklearn has many standard ML algs
import sklearn
# Here we go!
print("Hello World!")
```

More on learning python

- We will have one tutorial devoted to this
- If you're new to Python, go slow!
  - First learn the basics (lists, dicts, for loops, etc.)
  - Then spend a couple days playing with numpy
  - Then explore matplotlib
  - etc.
- Piazza = your friend. We have a designated python instructor (IA Ben Bray) who has lots of answers.

Lecture Cat #2

(credit to Johann for the suggestion)

Functions and Convexity

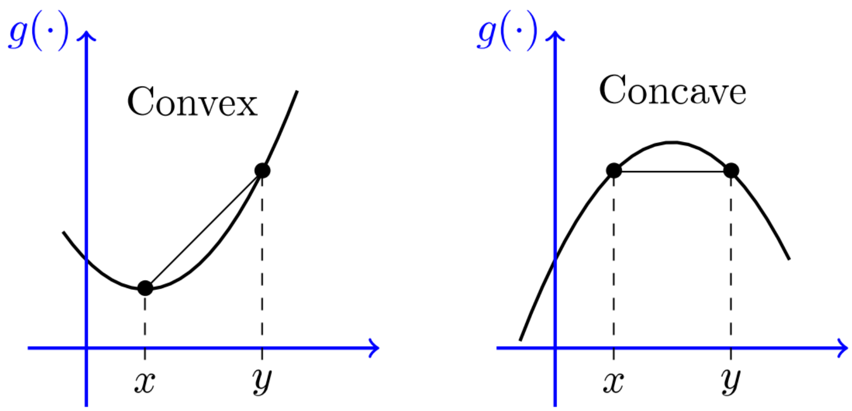
- Let f be a function mapping $\mathbb{R}^n \to \mathbb{R}$, and assume f is twice differentiable.
- The *gradient* and *hessian* of f, denoted $\nabla f(x)$ and $\nabla^2 f(x)$, are the vector an matrix functions:

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_1} \end{bmatrix} \qquad \nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

- Note: the hessian is always symmetric!

Convex functions

- We say that a function f is *convex* if, for any distinct pair of points x, y we have
$$f\left(\frac{x+y}{2}\right) \leq \frac{f(x)}{2} + \frac{f(y)}{2}$$



Fun facts about convex functions

- If f is differentiable, then f is convex iff f "lies above its linear approximation", i.e.:
$$f(x+y) \geq f(x) + \nabla_x f(x) \cdot y \qquad \text{for every } x, y$$
- If f is twice-differentiable, then the hessian is always positive semi-definite!
- This last one you will show on your homeowork :-)

Convex Sets

- $C \subseteq \mathbb{R}^n$ is **convex** if
  $$tx + (1 - t)y \in C$$
  for any $x, y \in C$ and $0 \leq t \leq 1$
- that is, a set is convex if the line connecting **any** two points in the set is entirely inside the set

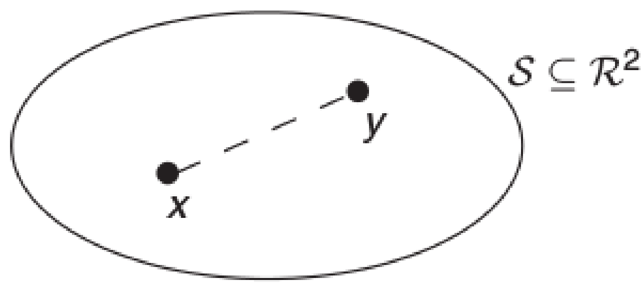
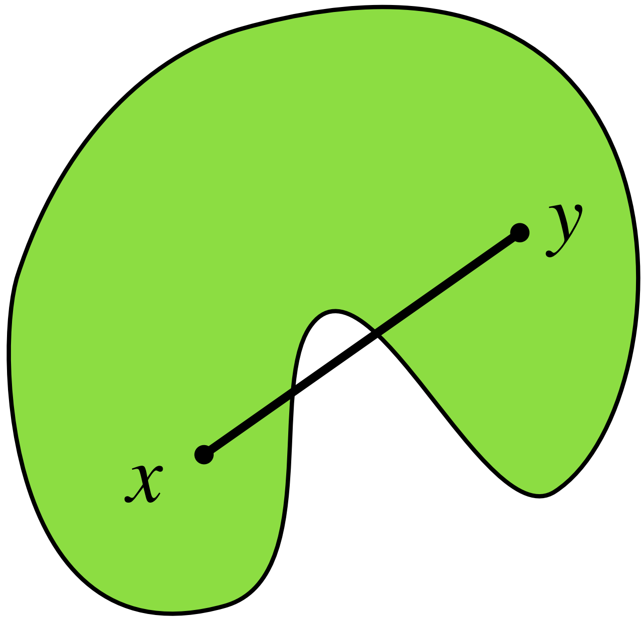


Figure 2.1 A convex set, $\mathcal{S} \subseteq \mathcal{R}^2$.

Not all sets are convex



The Most General Optimization Problem

Assume f is some function, and $C \subset \mathbb{R}^n$ is some set. The following is an *optimization problem*:

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

- How hard is it to find a solution that is (near-) optimal? This is one of the fundamental problems in Computer Science and Operations Research.
- A huge portion of ML relies on this task

A rough optimization hierarchy

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

- **[Really Easy]** $C = \mathbb{R}^n$ (i.e. problem is *unconstrained*), f is convex, f is differentiable, strictly convex, and "slowly-changing" gradients
- **[Easyish]** $C = \mathbb{R}^n$, f is convex
- **[Medium]** C is a convex set, f is convex
- **[Hard]** C is a convex set, f is non-convex
- **[REALLY Hard]** C is an arbitrary set, f is non-convex

Optimization without constraints

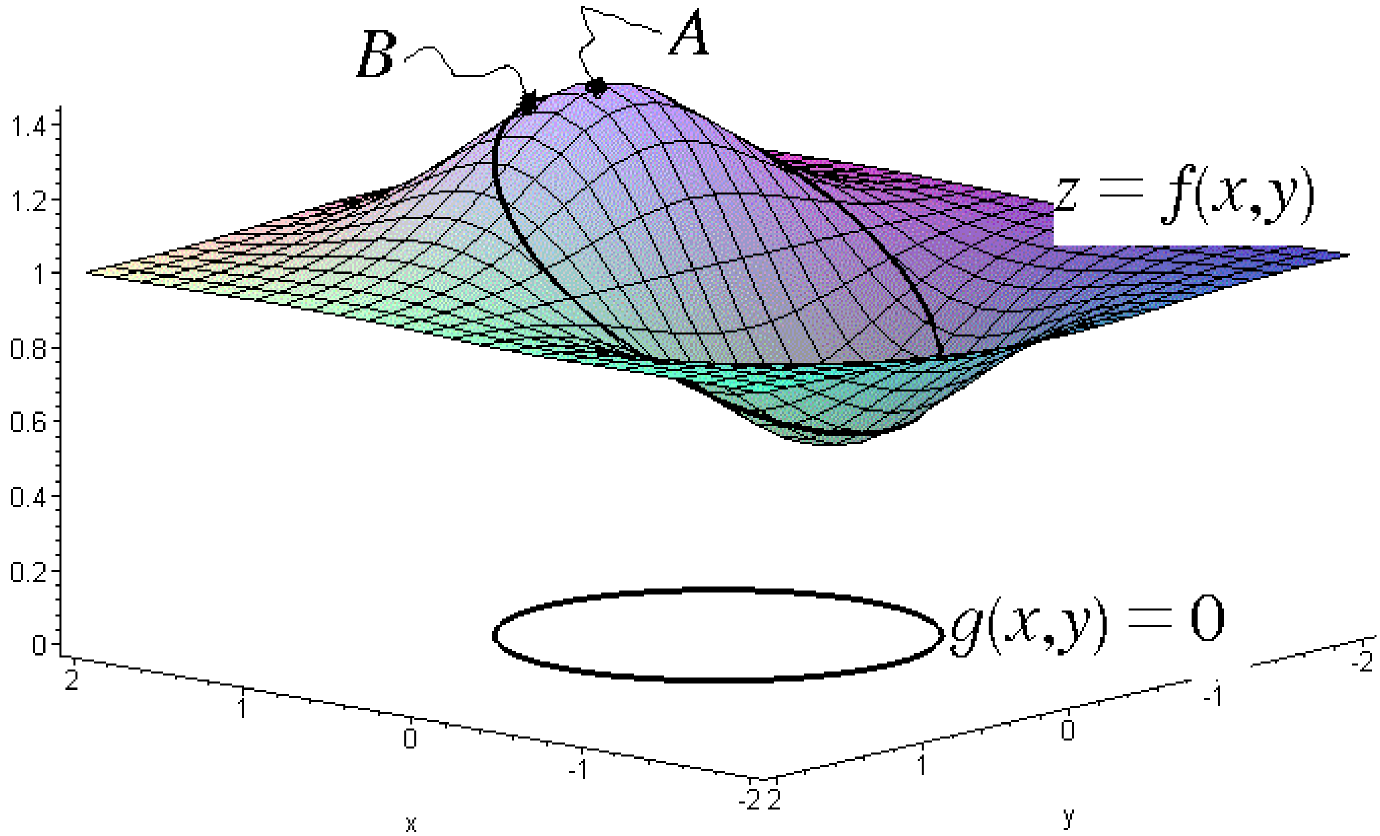
$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & x \in \mathbb{R}^n \end{aligned}$$

- This problem tends to be easier than constrained optimization
- We just need to find an x such that $\nabla f(x) = \vec{0}$
- Techniques like *gradient descent* or *Newton's method* work in this setting. (More on this later)

Optimization with constraints

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \le 0, \quad i = 1, \ldots, m \end{aligned}$$

- Here the set $C := \{x : g_i(x) \le 0, i = 1, \ldots, m\}$
- C is convex as long as all $g_i(x)$ convex
- The solution of this optimization may occur in the *interior* of C, in which case the optimal x will have $\nabla f(x) = 0$
- But what if the solution occurs on the *boundary* of C?



A Quick Overview of Lagrange Duality

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

- Here we need to work with the *Lagrangian*:

$$L(x, \boldsymbol{\lambda}) := f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$

- The vector $\boldsymbol{\lambda} \in \mathbb{R}^m$ are *dual variables*
- For fixed $\boldsymbol{\lambda}$, we now solve $\nabla_x L(x, \boldsymbol{\lambda}) = \mathbf{0}$

A Quick Overview of Lagrange Duality

$$\text{Lagrangian:} \quad L(x, \boldsymbol{\lambda}) := f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$

- Assume, for every fixed λ, we found x_λ such that

$$\nabla_x L(x_\lambda, \boldsymbol{\lambda}) = \nabla f(x_\lambda) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x_\lambda) = \mathbf{0}$$

- Now we have what is called the *dual function*,

$$h(\boldsymbol{\lambda}) := \inf_{x \in \mathbb{R}^n} L(x, \boldsymbol{\lambda}) = L(x_\lambda, \boldsymbol{\lambda})$$

The Lagrange Dual Problem

- What did we do here? We took one optimization problem:

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

- And then we got another optimization problem:

$$d^* := \begin{array}{ll} \text{maximize} & h(\boldsymbol{\lambda}) \\ \text{subject to} & \lambda_i \geq 0, \quad i = 1, \ldots, m \end{array}$$

- Sometimes this *dual problem* is easier to solve
- We always have *weak duality*: $p^* \geq d^*$
- Under nice conditions, we get *strong duality*: $p^* = d^*$

Recommended reading:

- Free online!
- Chapter 5 covers duality