1. Using your class notes, prepare a 1-2 paragraph summary of what we talked about in class in the last week. I do not want just a bulleted list of topics, I want you to use complete sentences and establish context (Why is what we have learned relevant? How does it connect with other things you have learned here or in other classes?). The more insight you give, the better.

2. You have a large amount of money $M$ that you are going to gamble on a horse race. You want to be smart about it, though.

   There are $N$ horses running in the race. You will divide up your money to place a bet of $x_i$ on each of them. Clearly,
   \[
   \sum_{i=1}^{N} x_i = M.
   \]

   As with any parimutuel betting scenario, if horse $i$ wins, the payout to you is proportional to the amount you bet on horse $i$ versus what everybody else (the “public”) bet on horse $i$. If you wager $x_i$ on horse $i$ and the public wagers $s_i$ then
   \[
   \text{payout if horse } i \text{ wins} = C \cdot \left( \text{total amount of money bet on all horses} \right) \cdot \frac{x_i}{x_i + s_i}
   \]
   \[
   = C \cdot \left( M + \sum_{i=1}^{N} s_i \right) \frac{x_i}{x_i + s_i}.
   \]

   The constant $C$ above is less than 1, and represents the fact that the track takes a cut of all the bets (the “vigorish” or ”vig” is $1 - C$). A typical value of $C$ might be 0.8 or 0.9.

   The reason you are betting is that you have two pieces of key knowledge about this race. First, you know the actual probability $p_i$ that horse $i$ will win. Second, you know $s_i$, the amount that the public will end up placing on horse $i$.

   (a) With your knowledge of the probabilities $p$ and public money $s_i$, write down a convex optimization program answer will tell you how much to bet on each horse to maximize your expected return. (In the end, you should be maximizing a concave function over a convex set.)

   (b) Using Fenchel duality, show how this expected payout can be computed by solving an optimization program in one variable. (Hint: look at the resource allocation example in the notes.) All of the relevant functions are given to you here, so you can (and should) compute their conjugates explicitly.

   (c) Show how the primal optimal solution (the best $x_i$) can be recovered from the (single variable) dual solution.

   (d) Here are the track odds right before closing$^1$:

$^1$This is from the 2017 San Diego Handicap.
1. Accelerate  8-1
2. El Huerfano  12-1
3. Arrogate  1-5
4. Donworth  10-1
5. Cat Burglar  10-1
6. Dalmore  20-1

Saying horse $i$ has track odds $A$-$B$ simply means that a bet of $B$ will yield $A$ in profit (and a total return of $A + B$) should horse $i$ win. From the payout equation above, this tells us that if horse $i$ has odds $A$-$B$, then\footnote{Note that this payout equation represents the payoffs that would be made taking into account the bets that have been made up till now, but before taking into account any bets you will make at the last minute.}

$$A + B = C \left( \sum_{i=1}^{N} s_i \right) \frac{B}{s_i}.$$ 

Note that by examining these odds, you can infer what the vig $(1 - C)$. If you know the total amount the public has wagered you can also determine the $s_i$’s.

Suppose that the public has wagered a total of $1$ million on this race.

You happen to know that the true probabilities are

1. Accelerate  0.20
2. El Huerfano  0.05
3. Arrogate  0.40
4. Donworth  0.15
5. Cat Burglar  0.15
6. Dalmore  0.05

You have $500,000. How much do you bet on each horse?

(e) Accelerate won the race.\footnote{That actually happened.} How did you make out? (You should calculate your return using the value of $C$ and the $s_i$’s determined in the previous problem.)

3. (B& V 9.5) Recall that when using backtracking to select a step size to move from $x_0$ in direction $d$, we start with $t = 1$, and then iterative decrease $t$ by factor of $\beta < 1$ until

$$f(x_0 + td) - f(x_0) \leq \alpha t \langle d, \nabla f(x_0) \rangle,$$

where $0 < \alpha < 1/2$ is some user-defined parameter. Show that if $f$ is strongly convex,

$$mI \preceq \nabla^2 f(x) \preceq MI, \quad \text{for all } x \in \mathbb{R}^N,$$

and $d$ is a descent direction, then the stopping condition holds when

$$t \leq \frac{-\langle d, \nabla f(x_0) \rangle}{M \|d\|^2_2}.$$ 

Use this to derive an upper bound on the number of backtracking iterations.
4. We have presented gradient descent as a basic method for solving smooth unconstrained problems. In this problem we will explore its use in solving nonsmooth constrained problems, specifically linear programs.

(a) Implement gradient descent: write a MATLAB function

```matlab
function xstar = gd(f, gradf, x0, tol)
```

that takes a function handle for evaluating $f : \mathbb{R}^N \to \mathbb{R}$, a function handle for evaluating $\nabla f : \mathbb{R}^N \to \mathbb{R}^N$, a starting point $x_0 \in \mathbb{R}^N$, and a tolerance so that you terminate when $\|\nabla f(x^{(k)})\|_2^2 \leq \text{tol}$. For now, you can hard-code the backtracking parameters as $\alpha = 0.001$ and $\beta = 0.8$. You might also want to cap the total number of iteration at something reasonable (say 10,000).

(b) Consider the linear program in $\mathbb{R}^2$,

$$
\min_x \langle x, c \rangle \quad \text{subject to} \quad \langle x, a_m \rangle \leq b_m
$$

where

$$
c = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad a_m = \begin{bmatrix} \cos(m\pi/3) \\ \sin(m\pi/3) \end{bmatrix}, \quad b_m = 1, \quad m = 1, 2, \ldots, 6.
$$

Sketch the feasible region in the plane — the region where all six linear constraints hold. Using MATLAB, create and image of $\langle x, c \rangle$ over the feasible region. Where is the minimizer $x^{\star}$?

(c) Now consider the smooth, unconstrained problem

$$
\min_x \langle x, c \rangle - \frac{1}{\tau} \sum_{m=1}^{6} \log(b_m - \langle x, a_m \rangle).
$$

Using MATLAB, create an image of the functional over the feasible region for $\tau = 1$. What is happening near the boundary?

(d) Solve the program above using gradient descent for $\tau = 1, 10, 100, 500, 1000$ with $\text{tol}=1e-4$ and starting at the origin, $x_0 = 0$. Make a figure of the feasible region, then put an 'x' where your solution landed for these different $\tau$. Note how many iterations it took for each $\tau$.

(e) Do the same, but use the solution for $\tau = 1$ as the starting point for $\tau = 10$ then use that solution for $\tau = 100$, etc. How many total iterations does it take to get the answer for $\tau = 1000$?