COMPUTER VISION - CS 527

Homework 1

Spring Semester

Collaborators

- Haozhe Zhang
- Felipe Montano-Campos

Part 1: Calculus

Problem 1.1

$$f(x, y) = 3x^4 - 4x^2y + y^2$$

Subject to $y = mx$. Hence we have to solve the following problem

$$\max_{x,y} f(x, y) \quad \text{subject to} \quad y = mx$$

If we plug in our restriction in our objective function we can define $\phi_m(x) = f(x, mx)$, so

$$\phi_m(x) = f(x, mx) = 3x^4 - 4x^2mx + m^2x^2 = 3x^4 - 4mx^3 + m^2x^2$$

Note that clearly the above function is twice differentiable with continuous second derivative in some ball centered at $x = 0$, and $x = 0$ is interior to the domain. Then, let's consider the first order condition, that is we take first order derivative with respect to x and get

$$\frac{\partial \phi_m}{\partial x} = 12x^3 - 12mx^2 + 2m^2x$$

Clearly, the above is 0 when $x = 0$. Then, we check the second order condition (Hessian), and get

$$\frac{\partial^2 \phi_m}{\partial x^2} = 36x^2 - 24mx + 2m^2$$

Note that the above is $2m^2 > 0$ when $x = 0$. This holds for any m since the square of any number is postive. Thus, in Hessian language it is definite negative so we indeed have a local strict minimum at $x = 0$.

Problem 1.2

The expression for the gradient of f is:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 12x^3 - 8xy \\ -4x^2 + 2y \end{bmatrix}$$

and the Hessian of f is:

$$H_f(\mathbf{x}) = \begin{bmatrix} 36x^2 - 8y & -8x \\ -8x & 2 \end{bmatrix}$$

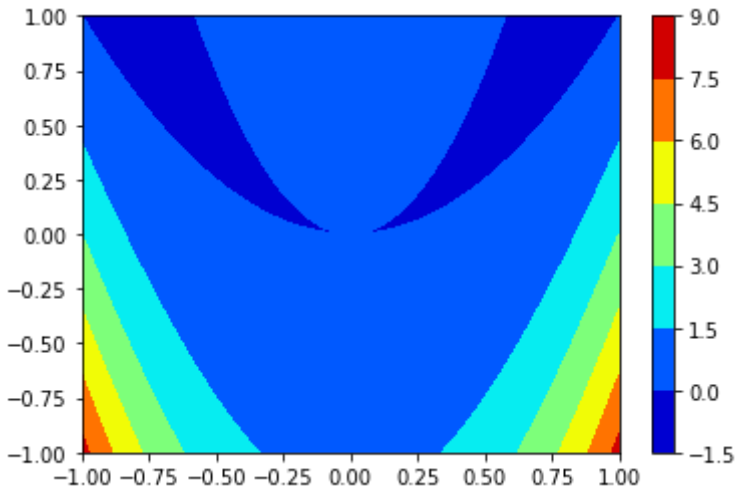
Problem 1.3

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
```

In [2]:
```
%matplotlib inline
# Defining the function
def foo(x, y):
    f = 3*x**4 - 4*x**2*y + y**2
    return f

# Creating the grid
x = np.linspace(-1, 1, 1001)
y = np.linspace(-1, 1, 1001)
X, Y = np.meshgrid(x, y)
```

In [3]:
```
# Here we compute the function values on that grid
F = foo(X, Y)
```

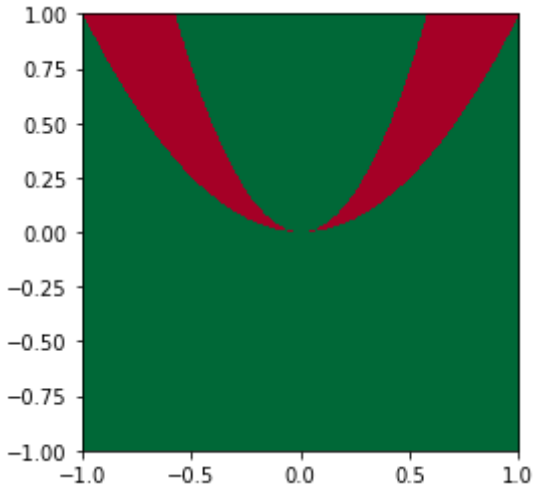
In [4]:
```
# We can initially get the following graph for several values of F

plt.contourf(X,Y,F,cmap="jet")
plt.colorbar()
plt.show()
```



In [5]:
```
# We need to transform data to plot as indicated in the
F[F >0] = 1
F[F <= 0 ] = -1
```

In [6]:
```
# Plotting
x_min = x[0]
x_max = x[-1]
y_min = y[0]
y_max = y[-1]
plt.imshow(F, origin = 'lower', extent =(x_min, x_max, y_min, y_max), cm
ap='RdYlGn')
```

Out[6]: <matplotlib.image.AxesImage at 0x1146e3ef0>



Problem 1.4

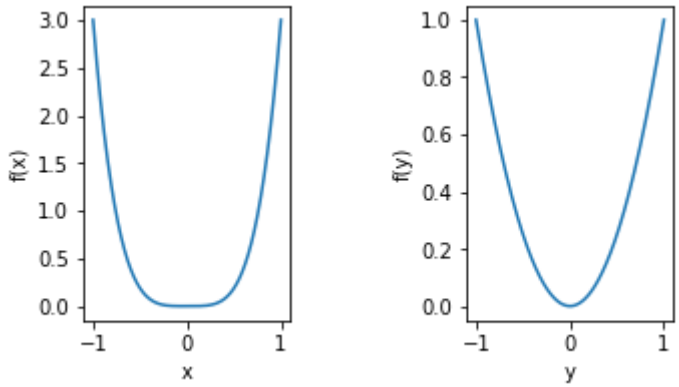
Following the hint, let's solve $f(x, y) = 0$ for y.

$$f(x, y) = 3x^4 - 4x^2y + y^2 = 0$$
$$y^2 - 4x^2y + 3x^4 = 0$$
$$(y - 3x^2)(y - x^2) = 0$$

Thus, from above we can observe that $f(x, y) \leq 0$, when $x^2 \leq y \leq 3x^2$. That is, $a(x) = x^2$ and $b(x) = 3x^2$.

Problem 1.5

In [7]:
```
f_x = foo(x, 0)
f_y = foo(0, y)
fig, axes = plt.subplots(1,2)
axes[0].plot(x, f_x)
axes[1].plot(y, f_y)
axes[0].set_xlabel('x')
axes[0].set_ylabel('f(x)')
axes[1].set_xlabel('y')
axes[1].set_ylabel('f(y)')
fig.tight_layout(pad=5.0)
```



Problem 1.6

First of all, let's notice the domain is $\mathbf{R}^2$, so every point is an interior point. Thus, if there is any extremas, they have to satisfy the first order condition, that is:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 12x^3 - 8xy \\ -4x^2 + 2y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

That is

$$12x^3 = 8xy$$
$$4x^2 = 2y$$

If $x = 0$, from the second equation we know it must be $y = 0$. If $x \neq 0$, then

$$12x^2 = 8y$$
$$4x^2 = 2y$$

This leads to $3x^2 = 4x^2$, which leads to $x = 0$. Thus there is no solution in this case. Therefore, the only possible candidate for extrema is $(0, 0)$. However, when $\mathbf{x} = \mathbf{0}$, we know the Hessian matrix is the following:

$$H_f(\mathbf{0}) = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix}$$

- First way of proof using Principal minors

Clearly, this matrix has all the principal minors equal to 0, so it's both positive and negative semi-definite. Thus, $\mathbf{0}$ cannot be local maximum or minimum. According to the definition, since the function is differentiable, then we have $\mathbf{x} = (0, 0)$ is a saddle point.

- Second way of proof directly using definition

Clearly, this matrix has only non-negative eigenvalues, so it's positive semi-definite and we can rule out the case of being a local maximum. Then, since from the previous question, we know if $x^2 < y < 3x^2$, we have $f(x, y) < 0$. Then, for any open ball around $\mathbf{0}$, we can always find $(x, 1.1x^2)$ (for x small enough) such that it is in the ball and the function value is negative. Since $f(0, 0) = 0$, it cannot be a local minimum either. According to the definition, since the function is differentiable, then we have $\mathbf{x} = (0, 0)$ is a saddle point.

Part 2: Linear Algebra

Problem 2.1

Since we know $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ are eigenvectors, then they has the property $\mathbf{Au} = \lambda\mathbf{u}$.

$$\mathbf{Au}_1 = \begin{bmatrix} 3 \\ a+b+c \\ d+e+f \end{bmatrix} = \begin{bmatrix} \lambda_1 \\ \lambda_1 \\ \lambda_1 \end{bmatrix} = \lambda_1 \mathbf{u_1}$$

$$\mathbf{Au}_2 = \begin{bmatrix} 0 \\ a-c \\ d-f \end{bmatrix} = \begin{bmatrix} \lambda_2 \\ 0 \\ -\lambda_2 \end{bmatrix} = \lambda_2 \mathbf{u_2}$$

$$\mathbf{Au}_1 = \begin{bmatrix} 0 \\ a-b \\ d-e \end{bmatrix} = \begin{bmatrix} \lambda_3 \\ -\lambda_3 \\ 0 \end{bmatrix} = \lambda_3 \mathbf{u_3}$$

From the above we can infer that $\lambda_1 = 3$, and $\lambda_2 = \lambda_3 = 0$. Thus,

$$a = b = c$$
$$d = e = f$$
$$a+b+c = 3$$
$$d+e+f = 3$$

So we have $a = b = c = d = e = f = 1$.

Problem 2.2

- rank(A)

Let's find the row echelon form first by substracting the third row from the first row, which is

$$\begin{bmatrix} 1 & 2 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Clearly, from the above we know there are only two linearly independent columns, i.e. the first and the third, thus $rank(A) = 2$

- $dim(R(A))$ and a basis B($R(A)$) for the range $R(A)$

Note that the range is just the span of all the column vectors. $dim(R(A)) = 2$, and a basis of the range would be $\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ since elementary row and column operations do not affect the dependence relations between the column vectores.

- $dim(R(A^T))$ and a basis for the row space

Note that

$$A^T = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 2 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

and the row echelon form of the above is

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

From the row echelon form, we can tell that $dim(N(A^T)) = 2$ and a basis would be $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ since elementary row and column operations do not affect the dependence relations between the column vectores.

- $dim(N(A))$ and a basis $B(N(A))$ for the null space $N(A)$

Since the null space is defined to be $\{\mathbf{x}|A\mathbf{x} = 0\}$, then we can solve for a general form of $\mathbf{x}$, that is

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -2x_3 - x_4 \\ -x_3 \\ x_3 \\ x_4 \end{bmatrix} = x_3 \begin{bmatrix} -2 \\ -1 \\ 1 \\ 0 \end{bmatrix} + x_4 \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Thus, clealry, $dim(N(A)) = 2$ and a basis would be $\begin{bmatrix} -2 \\ -1 \\ 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$.

- $dim(N(A^T))$ and a basis for the left null space.

Since the left null space is defined to be $\{\mathbf{x}|A^T\mathbf{x} = 0\}$, then we can solve for a general form of $\mathbf{x}$, that is

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -x_3 \\ 0 \\ x_3 \end{bmatrix} = x_3 \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Thus, clealry, $dim(N(A)) = 1$ and a basis would be $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$.

Part 3: Probability

Problem 3.1

We have to calculate the conditional probability. Let's define:

- A = Both senators from NC part of the committe
- B = At least one of the senators of NC is already in the committee

Hence we need to calculate the following probability:

$$P(A|B) = \frac{P(A \wedge B)}{P(B)} = \frac{P(A)}{P(B)}$$

Let's first calculate $P(A)$:

$$P(A) = \frac{\binom{98}{48}}{\binom{100}{50}}$$

Now let's calculate $P(B)$, note that $P(B)$ equals to the the probabilit of 1-Prob of that no senator from NC is in commitee.

$$P(B) = 1 - \frac{\binom{98}{50}}{\binom{100}{50}}$$

So the conditional probability is:

$$P(A|B) = \frac{\frac{\binom{98}{48}}{\binom{100}{50}}}{1 - \frac{\binom{98}{50}}{\binom{100}{50}}}$$

Then, we use python to calculate this:

In [8]:
```
import scipy.special
```

In [9]:
```
a = scipy.special.comb(98,48, exact = True)
b = scipy.special.comb(100,50, exact = True)
c = scipy.special.comb(98,50, exact = True)

p = a/(b-c)


print('p = {:.3f}'.format(p))
```

```
p = 0.329
```

Problem 3.2

For standard family with two kids, there are three cases.

- Both kids are boys.
- Both kids are girls.
- One is a boy and the other is a girl.
- One is a girl and the other is a boy.

Conditioning on the fact that one child is a boy, we can rule out the second case. Then using the law of total probability we have:

$$\Pr(B|B) = \Pr(B|BB) * \Pr(BB) + \Pr(B|BG) * \Pr(BG) + \Pr(B|GB) * \Pr(GB)$$

Since all families are choosen uniformly at random (all the 3 families have the probability $1/3$), we get the following result

In [10]:
```
p = 1*(1/3) + (1/2)*(1/3) + (1/2)*(1/3)
print('p = {:.3f}'.format(p))
```

```
p = 0.667
```

Problem 3.3

Following the hint, the maximum number of tosses needed to now for sure whether the event occurred is 10. This is because if there are less than 6 heads in the first 10 tosses, it means there has to be at least 5 tails thus we know the event did not occur or, if there are more than 6 heads in the first 10 tosses, it means there has to be less than 5 tails thus we know that the event occured. Then, among the 10 tosses we have the following cases for the event to occur:

- No tails: $\binom{10}{0}$
- One tail: $\binom{10}{1}$
- Two tails: $\binom{10}{2}$
- Three tails: $\binom{10}{3}$
- Four tails: $\binom{10}{4}$

We can calculate this using python

In [11]:
```
a = scipy.special.comb(10,0, exact = True)
b = scipy.special.comb(10,1, exact = True)
c = scipy.special.comb(10,2, exact = True)
d = scipy.special.comb(10,3, exact = True)
e = scipy.special.comb(10,4, exact = True)

print(a,b,c,d,e)
```

```
1 10 45 120 210
```

Hence, we have:

- No tails $\binom{10}{0} = 1$
- One tail $\binom{10}{1} = 10$
- Two tails $\binom{10}{2} = 45$
- Three tails $\binom{10}{3} = 120$
- Four tails $\binom{10}{4} = 210$

Then the totol cases of at least 6 heads is gonna be $1 + 10 + 45 + 120 + 210 = 386$. Also, the total cases would be $2^{10} = 1024$, so $p = \frac{386}{1024}$

In [12]:
```
a = 1 +10 + 45 +120 + 210
b = 2**10
p = a/b
print('p={:.3f}'.format(p))
```

```
p=0.377
```

Part 4: Programming with Images

In [13]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color
from tqdm import tqdm_notebook as tqdm

def read_gray(filename):
    image = io.imread(filename)
    if len(image.shape) == 3:  # Image has three bands, and is likely a
 color image
        image = color.rgb2gray(image)  # Convert to gray, with float val
ues between 0 and 1
        image = np.around(255 * image).astype(np.uint8)  # Scale and rec
ast to unsigned bytes
    return image

def write_gray(image, filename):
    io.imsave(filename, image)

def show_gray(image):
    plt.imshow(image, cmap='gray')
    plt.axis('off')  # Delete axis ticks
```

Problem 4.1

Consider the following function that performs median filtering on gray images and handles boundary pixels with the shrinking-window method described under Processing Near Image Boundaries in the instructions

In [14]:
```
def median_filter(image, h):
    """
    This function implements the median filter algorithm.
    """
    assert len(image.shape) == 2, 'Image must be a 2D array'
    assert image.dtype ==np.uint8, 'Pixel values must be bytes'
    assert isinstance(h, int) and h >= 0, 'h must be a non-negative integer'
    shape = image.shape
    medfilter = np.zeros(shape)
    for i in range(shape[0]):
        for j in range(shape[1]):
            left_indx_row = max(0, i-h) #Change
            right_indx_row = min(i+h+1, shape[0])
            down_indx_col = max(0, j-h)
            up_indx_col = min(j+h+1, shape[1]) #Change
            scan_block = image[left_indx_row:right_indx_row, down_indx_col:up_indx_col]
            med = np.median(scan_block)
            medfilter[i,j] = med
    return(medfilter)
```

In [15]:
```
image = read_gray("locomotive_corrupt.png")
show_gray(image)
```

In [ ]:
```
image_list =[image]
for h in tqdm([1, 3, 10]):
    new_image = median_filter(image, h)
    image_list.append(new_image)
```

In [ ]:
```
H = [0, 1, 3, 10]
for i in range(4):
    plt.subplot(2, 2, i+1).set_title('h = {:.0f}'.format(H[i]))
    show_gray(image_list[i])
```

Problem 4.2

Comment on advantages and disadvantages of median filtering (where does it help and where does it hurt the most), and on possible criteria for setting h .

ADVANTAGES

- It is very easy to implement and it successfully eliminate the noise in the images. The key element is that the median is very robust to outliers, so the salt and peper noise goes away, compared to other methods like the mean, for instance.
- It has an $O(nlogn)$ time complexity.

DIS-ADVANTAGES

- Even though we can eliminate all the noise of the image using the median filter, it degrades the quality of the image. Since, we are manipulating every pixel value of the image

Optimal Criteria for h

- As seen in our results, h should not be too big neither too small. One on hand ff it is too big the quality of the image is vastly degraded. On the other hand, if it is too small we might not remove all the salt and peper noise in the image