1.

(i)

$$\hat{\theta} = \arg\min - \sum_{n=1}^{N} \{ y_n \log h_\theta(x_n) + (1-y_n)\log(1-h_\theta(x_n)) \}.$$

$$J(\theta) = \sum_{n=1}^{N} \{ y_n \log h_\theta(x_n) + (1-y_n)\log[1-h_\theta(x_n)] \}$$

$$= (\sum_{n=1}^{N} y_n x_n)^T \theta - \sum_{n=1}^{N} \log(1+e^{\theta^T x_n})$$

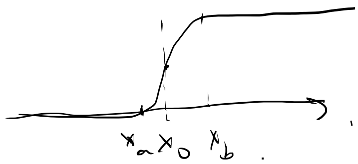
$$\nabla_\theta J(\theta) = (\sum_{n=1}^{N} y_n x_n) - \sum_{n=1}^{N} \frac{x_n \cdot e^{\theta^T x_n}}{1+e^{\theta^T x_n}}$$

$$= \sum_{n=1}^{N} x_n \cdot (y_n - \frac{e^{\theta^T x_n}}{1+e^{\theta^T x_n}})$$

$$= \sum_{n=1}^{N} x_n [y_n - h_\theta(x)] = 0.$$

If the two classes are linearly seperable, the optimal θ should switch from $y_n = 0$ to $y_n = 1$ instantly to satisfy the gradient being 0.

Assume x_0 is the switching point,



$$h_\theta(x_a) = \frac{1}{1+e^{-\theta^T(x_a-x_0)}}$$

$$h_\theta(x_b) = \frac{1}{1+e^{-\theta^T(x_b-x_0)}} \qquad \lim_{\substack{x_b\to x_0 \\ x_a \to x_0}} \frac{h_\theta(x_b) - h_\theta(x_a)}{x_b - x_a} \to \infty.$$

As the steepness of the curve goes to infinity, $\|\theta\| \to \infty$.

Hence $\|W\| \to \infty$ and $\|W_0\| \to \infty$.

ii) As shown in i).

$$\nabla_\theta J(\theta) = \sum_{n=1}^{N} x_n [y_n - h_\theta(x_n)] < 0,$$

Since the steepness of $h_\theta(x_n)$ only tends to ∞, but never reaches it, the gradient descent would never converges unless it reaches ∞, which is impossible.

iii) If we set a maximum steepness on the logistic curve, the gradient descent will stop. once it it reaches there,

Another way to counter this problem is to add regularization term in the loss function.

iv) We haven't met non-convergence issue with other linear classifiers, because if the data are linearly seperable, then we can invert the matrix and perform good'ol linear regression.

2. $J(\theta) = -\frac{1}{N} \sum_{n=1}^{N} \left\{ y_n \log h_\theta(x_n) + (1-y_n) \log(1-h_\theta(x_n)) \right\}$

where $h_\theta(x) = \dfrac{1}{1+\exp(-\theta^T x)}$. Show $J(\theta)$ convex. by

Its Hessian, & show it's PSD.

$$J(\theta) = -\frac{1}{N} \sum_{n=1}^{N} \Big\{ \underbrace{y_n \log\left(\frac{h_\theta(x_n)}{1-h_\theta(x_n)}\right)}_{①} + \underbrace{\log(1-h_\theta(x_n))}_{②} \Big\}.$$

① : $y_n \log\left(\dfrac{h_\theta(x_n)}{1-h_\theta(x_n)}\right)$

$$= y_n \log\left(\frac{\frac{1}{1+e^{-\theta^T x_n}}}{\frac{e^{-\theta^T x}}{1+e^{-\theta^T x}}}\right) = y_n \log(e^{\theta^T x_n}) = \underbrace{y_n \theta^T x_n}_{\text{linear}}.$$

thus the first term is convex.

② $\nabla_\theta[-\log(1-h_\theta(x))] = -\nabla_\theta\left[\log\left(1-\frac{1}{1+e^{-\theta^T x}}\right)\right]$

$$= -\nabla_\theta\left[\log \frac{e^{-\theta^T x}}{1-e^{-\theta^T x}}\right] = h_\theta(x)\, x,$$

$$\nabla_\theta^2[-\log(1-h_\theta(x))] = h_\theta(x)[1-h_\theta(x)]\, x x^T$$

For any $v \in \mathbb{R}^d$,

$$v^T \nabla_\theta^2 [-\log(1-h_\theta(x))] v = v^T [h_\theta(x)(1-h_\theta(x)) x x^T] v = h_\theta(x)(1-h_\theta(x)) \|v^T x\|^2 \geq 0$$

thus the Hessian is PSD., so $-\log(1-h_\theta(x))$ is convex in θ.

hw4

March 19, 2021

[1]:
```
%config IPCompleter.use_jedi = False
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import cvxpy as cp
import pandas as pd
np.set_printoptions(precision=4)

from pathlib import Path
fig_path = str(Path().absolute())+'/figures/hw4/'
print(fig_path)
data_path = str(Path().absolute())+'/hw4/data/'
print(data_path)
```

```
/home/zpyang/grad_courses/2021_spring/ece595_ml/figures/hw4/
/home/zpyang/grad_courses/2021_spring/ece595_ml/hw4/data/
```

1 Exercise 3

3 a)

$$
\begin{aligned}
J(\theta) &= -\frac{1}{N}\sum_{n=1}^{N}[y_n \cdot \log h_\theta(x_n) + (1-y_n)\cdot \log(1-h_\theta(x_n))] \\
&= -\frac{1}{N}\sum_{n=1}^{N}[y_n \cdot \log(\frac{h_\theta(x_n)}{1-h_\theta(x_n)}) + \log(1-h_\theta(x_n))] \\
&= -\frac{1}{N}\{\sum_{n=1}^{N}[y_n \cdot \log(\frac{h_\theta(x_n)}{1-h_\theta(x_n)})] + \sum_{n=1}^{N}\log(1-h_\theta(x_n))\}
\end{aligned}
$$

where $h_\theta(x) = \frac{1}{1+\exp\{-\theta^T x_n\}}$

Subbing $h_\theta(x)$ back into the loss function, we obtain

$$
\begin{aligned}
J(\theta) &= -\frac{1}{N}\{\sum_{n=1}^{N}[y_n \cdot \log(\frac{\frac{1}{1+\exp\{-\theta^T x_n\}}}{1-\frac{1}{1+\exp\{-\theta^T x_n\}}})] + \sum_{n=1}^{N}\log(1-\frac{1}{1+\exp\{-\theta^T x_n\}})\} \\
&= -\frac{1}{N}\{\sum_{n=1}^{N} y_n \log(e^{-\theta^T x_n}) + \sum_{n=1}^{N}\log(1-\frac{e^{\theta^T x_n}}{1+e^{\theta^T x_n}})\}
\end{aligned}
$$

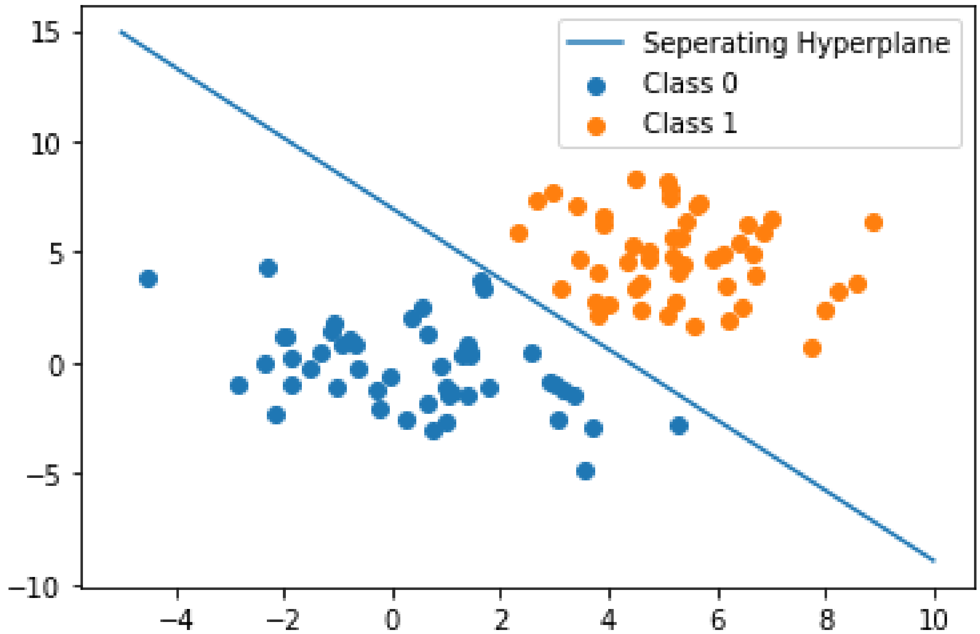
$$= -\frac{1}{N}\{\sum_{n=1}^{N} -y_n\theta^T x_n - \sum_{n=1}^{N} \log(1+e^{\theta^T x_n})\}$$
$$= -\frac{1}{N}\{(\sum_{n=1}^{N} -y_n x_n)^T\theta - \sum_{n=1}^{N} \log(1+e^{\theta^T x_n})\}$$

[2]:
```
# 3 b)

c0 = pd.read_csv(data_path+"class0.txt", delimiter='\s+', header=None).
 ↪to_numpy()
c1 = pd.read_csv(data_path+"class1.txt", delimiter='\s+', header=None).
 ↪to_numpy()

n0 = c0.shape[0]
n1 = c1.shape[0]
N = n0 + n1
x = np.vstack([c0,c1])
y = np.hstack([np.zeros(n0), np.ones(n1)]).reshape(-1,1)
X = np.hstack([x, np.ones((N,1))])

lambd = 0.0001

theta = cp.Variable((3,1))
loss = -cp.sum(cp.multiply(y, X @ theta)) \
        + cp.sum(cp.log_sum_exp(cp.hstack([np.zeros((N,1)), X @ theta]),␣
 ↪axis=1))
reg = cp.sum_squares(theta)
prob = cp.Problem(cp.Minimize(loss/N + lambd*reg))
prob.solve()
omega = theta.value

omega
```

[2]:
```
array([[  2.3786],
       [  1.4975],
       [-10.4365]])
```

[3]:
```
# 3 c)

x1 = np.linspace(-5,10,2)
x2 = -omega[0]/omega[1] * x1 - omega[2]/omega[1]

plt.figure
plt.scatter(c0[:,0], c0[:,1], label='Class 0')
plt.scatter(c1[:,0], c1[:,1], label='Class 1')
plt.plot(x1,x2, label='Seperating Hyperplane')
plt.legend()
```

[3]: <matplotlib.legend.Legend at 0x7f262525cf10>

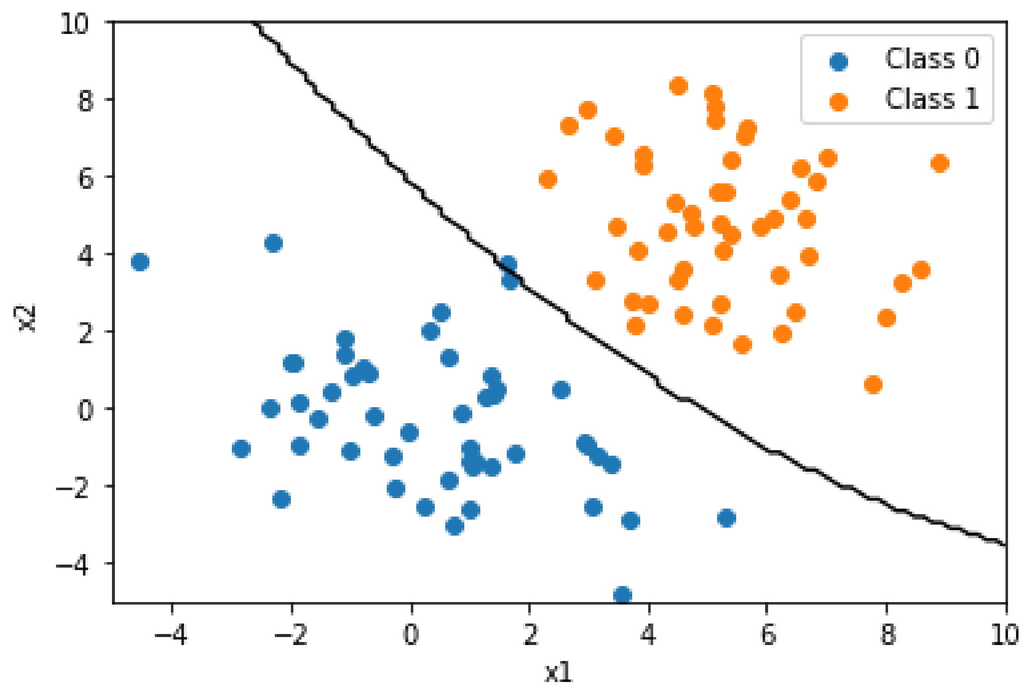


[4]:
```
# 3 d)
mu0 = c0.mean(0)
mu1 = c1.mean(0)

SIG0 = np.cov(c0.T)
SIG1 = np.cov(c1.T)

invSIG1 = np.linalg.inv(SIG1)
invSIG0 = np.linalg.inv(SIG0)
detSIG1 = np.linalg.det(SIG1)
detSIG0 = np.linalg.det(SIG0)

LOG0 = -0.5*np.log(detSIG0)
LOG1 = -0.5*np.log(detSIG1)
```

[5]:
```
from tqdm.notebook import tqdm

x1 = np.linspace(-5,10,N)
x2 = np.linspace(-5,10,N)
X1,X2 = np.meshgrid(x1,x2)

grid = np.zeros((N,N))

for i in range(N):
```

```
    for j in range(N):
        xi = np.array([X1[i,j], X2[i,j]]).T
        k0 = -0.5*(xi - mu0).T @ invSIG0 @ (xi - mu0) + LOG0
        k1 = -0.5*(xi - mu1).T @ invSIG1 @ (xi - mu1) + LOG1
        if k1 > k0:
            grid[i,j] = 1
plt.figure()
plt.contour(X1,X2,grid, levels=[0.5], cmap='gray')
plt.scatter(c0[:,0], c0[:,1], label='Class 0')
plt.scatter(c1[:,0], c1[:,1], label='Class 1')
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend()
```

[5]: <matplotlib.legend.Legend at 0x7f262304c8b0>



2 Exercise 4

[6]:
```
# 4 a)

h = 1
K = np.zeros((N,N))
for i in range(N):
```

```
    for j in range(N):
        K[i,j] = np.exp(-np.sum((X[i,:]-X[j,:])**2)/h)

print('The 47th to 52th elements of the Kernel Matrix: \n', K[47:52, 47:52])
```

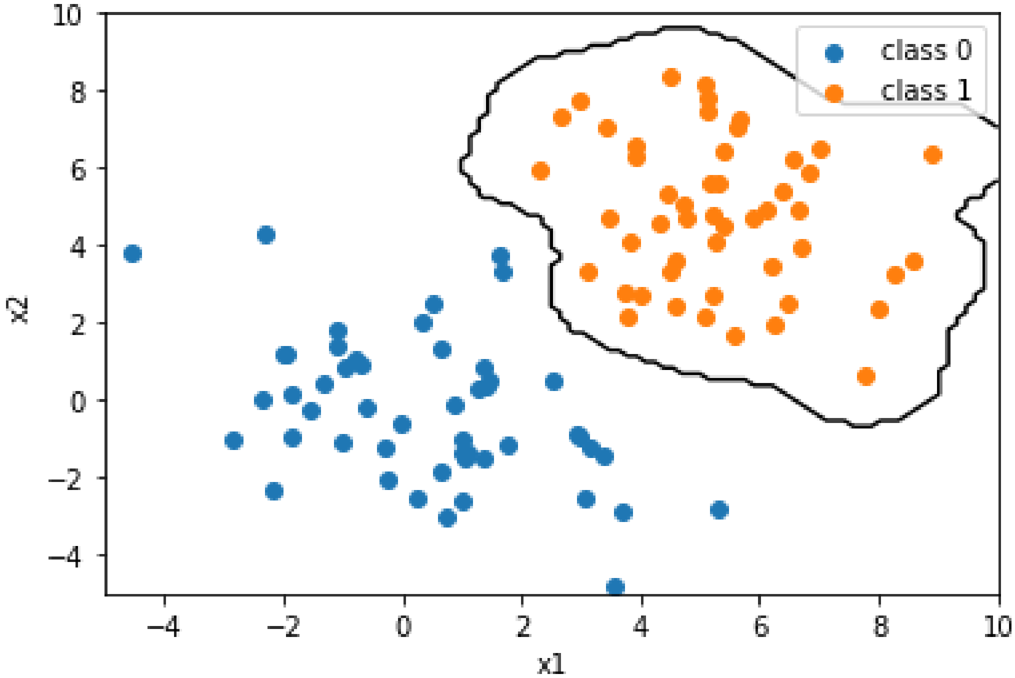
```
The 47th to 52th elements of the Kernel Matrix:
 [[1.0000e+00 5.0531e-25 6.0654e-20 4.6547e-29 4.0689e-17]
 [5.0531e-25 1.0000e+00 3.9593e-13 2.6936e-33 5.3878e-12]
 [6.0654e-20 3.9593e-13 1.0000e+00 2.3035e-65 3.7842e-34]
 [4.6547e-29 2.6936e-33 2.3035e-65 1.0000e+00 2.1628e-06]
 [4.0689e-17 5.3878e-12 3.7842e-34 2.1628e-06 1.0000e+00]]
```

[7]:
```
# 4 b)

alpha = cp.Variable((N,1))
loss = -cp.sum(cp.multiply(y, K @ alpha)) \
        + cp.sum(cp.log_sum_exp(cp.hstack([np.zeros((N,1)), K @ alpha]),
 ↪axis=1))
reg = cp.quad_form(alpha, K)
prob = cp.Problem(cp.Minimize(loss/N + lambd*reg))
prob.solve()
a = alpha.value
print('The first 2 elements of alpha: ', a[0:2].T)
```

```
The first 2 elements of alpha:  [[-0.9525 -1.2105]]
```

[8]:
```
from numpy.matlib import repmat
out = np.zeros((N,N))
for i in range(N):
    for j in range(N):
        data = repmat(np.array([x1[j], x2[i], 1]).reshape((1,3)), N, 1)
        phi = np.exp(-np.sum((X-data)**2, axis=1)/h)
        out[i,j] = np.dot(phi.T, a)

plt.figure()
plt.scatter(c0[:,0], c0[:,1], label='class 0')
plt.scatter(c1[:,0], c1[:,1], label='class 1')
plt.contour(x1, x2, out>0.5, levels=[0.5], cmap='gray')
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend()
```

[8]:
```
<matplotlib.legend.Legend at 0x7f2620d97130>
```


class 0
class 1
x1
x2
10
8
6
4
2
0
−2
−4
−4
−2
0
2
4
6
8
10

ECE595 ML Project Report - Checkpoint 1
Efficacy of Noise2Noise on Different Types of Noise

Zhanpeng Yang, MSAAE [1]

Abstract

This is an abstract for this project report, blah-blah blah

1. Intorduction

I plan to pursue topic one in the list for this project to explore self-supervised learning for image denoising. (Lehtinen et al., 2018). Some dataset is available on GitHub [2] as well as Kodak database [3] My background is in Autonomy and Control for unmanned aerial systems (UAS), where self-supervised learning has lots of potential to make control decisions. By conducting research and experiment on this topic, specifically in determining the efficacy of denoising effect in dynamic scenes, I hope to gain more insight in integration of deep learning and control's theory.

From my understanding of the paper(Lehtinen et al., 2018), as long as the noises are Gaussian, then we can recover the denoised image by training on noisy images, as the noiseless image would be the mean of the noisy images.

My current research in UAVs has led me to interact with race drone pilots whom use analog video transmitters to gain low latency first person view(FPV) from their high speed drones. Current competitor to the analog devices are the digital video transmission device from DJI using a compression algorithm to achieve low latency video streaming. The digital solution provides much better image quality, however, is more expensive compared to the analog solution.

The analog solution has lots of noise in the streamed videos due to radio interference. Denoising these videos would be an interesting extension to the current work done by the noise2noise method.

[1]School of Aeronautics and Astronautics, Purdue University, West Lafayette, Indiana, USA.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

[2]https://github.com/NVlabs/noise2noise

[3]http://r0k.us/graphics/kodak/

2. Homework 4 Project Update

2.1. Hypothesis

My hypothesis is that denoising dynamic scenes can be done with noise2noise method. Since videos are just images played at very fast speed. To denoise a video, we would simply denoise each frame and piece them back together. However, the quality of the final video may not be as good as a clean video due to motion blurs.

2.2. Verification Methods

Due to hardware limitation, aka GPU, I would work on videos with 360p to 480p resolution. This is also the maximum resolution of most commercial analog FPV transmitter.

For a simple proof of concept, I would artificially introduce noise to the frames captured using a smart phone or GoPro. Train the neural net on these frames, and evaluate its performance on Gaussian noise as well. However, one important metric I would also evaluate is the processing time of denoising each frame. This would give me an insight in the latency of denoising videos in real time.

If time permits, I would look into the performance comparison between neural nets of different sizes, and picking the optimally sized neural net.

2.3. Timeline

March 20 - March 27	Rewrite a simple training code with tensorflow and gather sample videos
March 28 - April 2	Train neural net and debug
April 3 - April 9	Evaluate performance and draft report
April 10 - April 30	Review report and conduct further research

Table 1. Project Timeline

References

Chen, C., Chen, Q., Xu, J., and Koltun, V. Learning to See in the Dark. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3291–3300, 5 2018. URL http://arxiv.org/abs/1805.01934.

Chi, Y., Gnanasambandam, A., Koltun, V., and Chan, S. H. Dynamic Low-light Imaging with Quanta Image Sensors. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12366 LNCS:122–138, 7 2020. URL http://arxiv.org/abs/2007.08614.

Gnanasambandam, A. and Chan, S. H. Image Classification in the Dark using Quanta Image Sensors. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12353 LNCS:484–501, 6 2020. URL http://arxiv.org/abs/2006.02026.

Lehtinen, J., Munkberg, J., Hasselgren, J., Laine, S., Karras, T., Aittala, M., and Aila, T. Noise2noise: Learning image restoration without clean data. Technical report, 2018. URL https://arxiv.org/pdf/1803.04189.pdf.