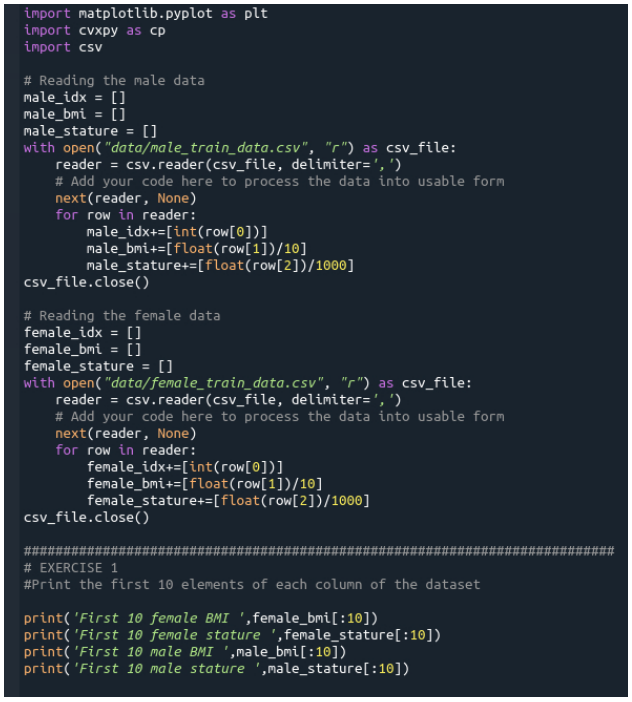
ECE 595: Homework 2

Manish Nagaraj, PUID: 0029904105

(Spring 2021)

Exercise 1 : Loading Data via Python

The code for loading the data is shown in Figure 1



```
import matplotlib.pyplot as plt
import cvxpy as cp
import csv

# Reading the male data
male_idx = []
male_bmi = []
male_stature = []
with open("data/male_train_data.csv", "r") as csv_file:
    reader = csv.reader(csv_file, delimiter=',')
    # Add your code here to process the data into usable form
    next(reader, None)
    for row in reader:
        male_idx+=[int(row[0])]
        male_bmi+=[float(row[1])/10]
        male_stature+=[float(row[2])/1000]
csv_file.close()

# Reading the female data
female_idx = []
female_bmi = []
female_stature = []
with open("data/female_train_data.csv", "r") as csv_file:
    reader = csv.reader(csv_file, delimiter=',')
    # Add your code here to process the data into usable form
    next(reader, None)
    for row in reader:
        female_idx+=[int(row[0])]
        female_bmi+=[float(row[1])/10]
        female_stature+=[float(row[2])/1000]
csv_file.close()

###########################################################################
# EXERCISE 1
#Print the first 10 elements of each column of the dataset

print('First 10 female BMI ',female_bmi[:10])
print('First 10 female stature ',female_stature[:10])
print('First 10 male BMI ',male_bmi[:10])
print('First 10 male stature ',male_stature[:10])
```

Figure 1: Code for loading the datasets

- The first 10 entries of female BMI : $[2.82, 2.21, 2.71, 2.81, 2.55, 2.3, 3.56, 3.11, 2.46, 4.3]$
- The first 10 entries of female stature : $[1.563, 1.716, 1.484, 1.651, 1.548, 1.665, 1.564, 1.676, 1.69, 1.704]$
- The first 10 entries of male BMI : $[3.0, 2.56, 2.42, 2.73, 2.59, 2.53, 2.27, 2.54, 3.41, 3.34]$
- The first 10 entries of male stature : $[1.679, 1.586, 1.773, 1.816, 1.809, 1.662, 1.829, 1.686, 1.761, 1.797]$

Exercise 2: Build a Linear Classifier via Optimization

(a)

$$\hat{\theta} = \underset{\theta \in \mathbf{R}^d}{\arg\min} \|y - X\theta\|^2$$

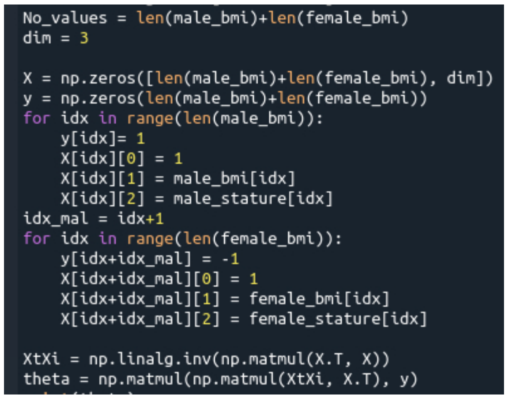
$$\hat{\theta} \text{ can be derived solving,} \quad \nabla_\theta \{\|y - X\theta\|^2\} = 0$$

$$\boxed{\hat{\theta} = \left(X^T X\right)^{-1} X^T y}$$

If $X^T X$ is not invertible, we can add regularizers such as the Ridge regularization or Lasso regularization.

(b)

The code used to solve the problem is shown in Figure 2. The solution is $\theta = \begin{bmatrix} -10.70 & -0.12 & 6.67 \end{bmatrix}$

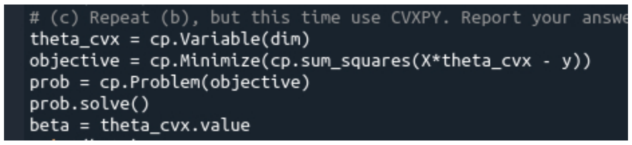


```
No_values = len(male_bmi)+len(female_bmi)
dim = 3

X = np.zeros([len(male_bmi)+len(female_bmi), dim])
y = np.zeros(len(male_bmi)+len(female_bmi))
for idx in range(len(male_bmi)):
    y[idx]= 1
    X[idx][0] = 1
    X[idx][1] = male_bmi[idx]
    X[idx][2] = male_stature[idx]
idx_mal = idx+1
for idx in range(len(female_bmi)):
    y[idx+idx_mal] = -1
    X[idx+idx_mal][0] = 1
    X[idx+idx_mal][1] = female_bmi[idx]
    X[idx+idx_mal][2] = female_stature[idx]

XtXi = np.linalg.inv(np.matmul(X.T, X))
theta = np.matmul(np.matmul(XtXi, X.T), y)
```

Figure 2: Code for solving the optimization using linalg

(c)

The CVXPY solution to the problem is shown in Figure 3. The solution is $\theta = \begin{bmatrix} -10.70 & -0.12 & 6.67 \end{bmatrix}$



```
# (c) Repeat (b), but this time use CVXPY. Report your answe
theta_cvx = cp.Variable(dim)
objective = cp.Minimize(cp.sum_squares(X*theta_cvx - y))
prob = cp.Problem(objective)
prob.solve()
beta = theta_cvx.value
```

Figure 3: Code for solving the optimization using CVXPY

(d)

$$\theta^{k+1} = \theta^k - \alpha^k \nabla \mathcal{E}_{train}(\theta^k)$$

$$\nabla \mathcal{E}_{train}(\theta^k) = \nabla_\theta(\|y - X\theta^k\|^2) = -2X^T(y - X\theta^k)$$

$$\boxed{\therefore \theta^{k+1} = \theta^k + 2\alpha^k X^T(y - X\theta^k)}$$

To find the optimal α^k using exact line search, we solve $\hat{\alpha}_k = \arg\min_\alpha \mathcal{E}(\theta^k + \alpha^k d)$. Where $d = \nabla \mathcal{E}_{train}(\theta^k)$

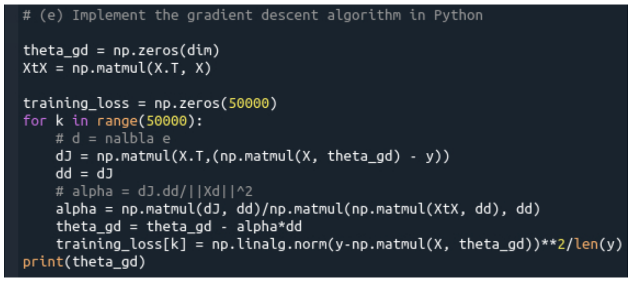
$$\frac{\delta}{\delta \alpha^k}\Big(\|y - X(\theta^k + \alpha^k d)\|\Big) = 0$$

$$-2\big(y - X(\theta^k + \alpha^k d)\big)^T X \cdot d = 0$$

$$\boxed{\therefore \alpha^k = \frac{(y - X\theta^k)^T \cdot X \cdot d}{\|X \cdot d\|^2}}$$

(e)

The code used to solve the gradient descent problem shown in Figure 4.
The solution is $\theta = \begin{bmatrix} -10.70 & -0.12 & 6.67 \end{bmatrix}$



```
# (e) Implement the gradient descent algorithm in Python

theta_gd = np.zeros(dim)
XtX = np.matmul(X.T, X)

training_loss = np.zeros(50000)
for k in range(50000):
    # d = nalbla e
    dJ = np.matmul(X.T,(np.matmul(X, theta_gd) - y))
    dd = dJ
    # alpha = dJ.dd/||Xd||^2
    alpha = np.matmul(dJ, dd)/np.matmul(np.matmul(XtX, dd), dd)
    theta_gd = theta_gd - alpha*dd
    training_loss[k] = np.linalg.norm(y-np.matmul(X, theta_gd))**2/len(y)
print(theta_gd)
```

Figure 4: Code for solving the gradient descent algorithm

(f)

The training loss of the gradient descent algorithm is shown in Figure 5.

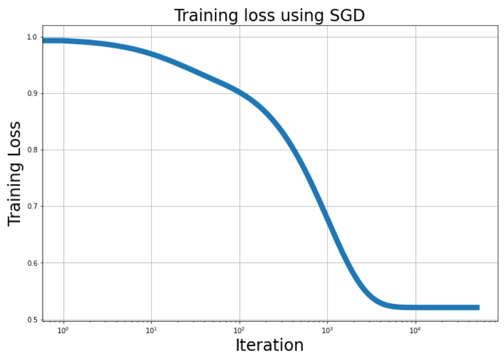
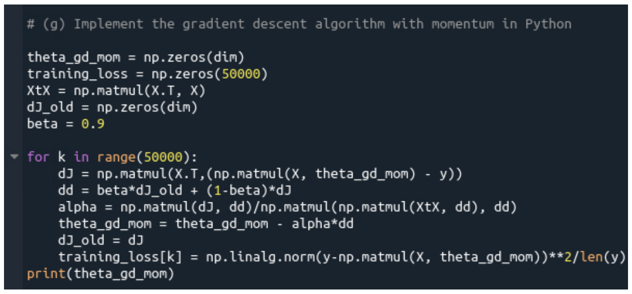


Figure 5: Gradient Descent Algorithm Training Loss

(g)

The code used to solve the gradient descent problem using momentum shown in Figure 6.



```
# (g) Implement the gradient descent algorithm with momentum in Python

theta_gd_mom = np.zeros(dim)
training_loss = np.zeros(50000)
XtX = np.matmul(X.T, X)
dJ_old = np.zeros(dim)
beta = 0.9

for k in range(50000):
    dJ = np.matmul(X.T,(np.matmul(X, theta_gd_mom) - y))
    dd = beta*dJ_old + (1-beta)*dJ
    alpha = np.matmul(dJ, dd)/np.matmul(np.matmul(XtX, dd), dd)
    theta_gd_mom = theta_gd_mom - alpha*dd
    dJ_old = dJ
    training_loss[k] = np.linalg.norm(y-np.matmul(X, theta_gd_mom))**2/len(y)
print(theta_gd_mom)
```

Figure 6: Code for solving the gradient descent algorithm

The solution is $\theta = \begin{bmatrix} -10.70 & -0.12 & 6.67 \end{bmatrix}$

(h)

The training loss of the gradient descent algorithm with momentum is shown in Figure 7.

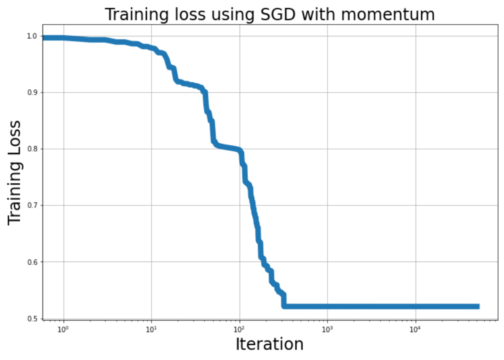


Figure 7: Gradient Descent Algorithm Training Loss using Momentum

Exercise 3: Visualization and Testing

(a)

The scatter plot of the two classes along with the decision boundary is shown in Figure 8.

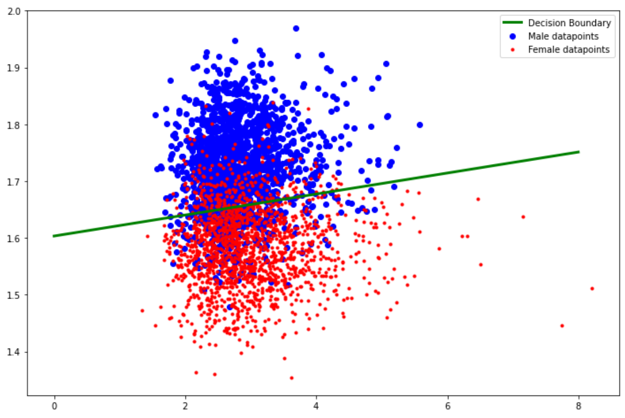


Figure 8: Decision Boundary

(b)

1. The percentage False Alarm of classifying a male is 14.17%
2. The percentage Type 2 error of classifying a male is 17.96%
3. The pecision of the classifier is 0.8526 and the recall of the classifier is 0.8203.

Exercise 4: Regularization

(a)

The solution to the ridge regression optimization is shown in Figure 9.

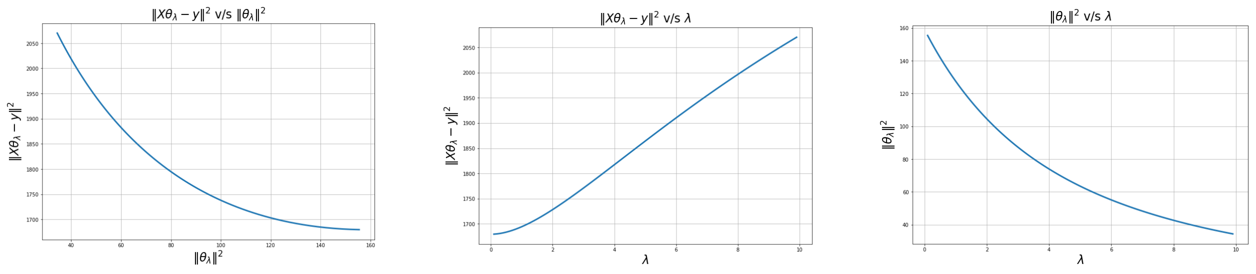


Figure 9: Solution to the ridge regression problem

(b)

(i) The Lagragian for eqns (4) and (5) is

$$\mathcal{L}\Big(\hat{\theta}_\alpha, \gamma_\alpha\Big) = \|X\hat{\theta_\alpha} - y\|^2 + \gamma_\alpha\big(\alpha - \|\hat{\theta_\alpha}\|^2\big)$$
$$\mathcal{L}\Big(\hat{\theta}_\epsilon, \gamma_\epsilon\Big) = \|\hat{\theta}_\epsilon\|^2 + \gamma_\epsilon\big(\epsilon - \|X\hat{\theta}_\epsilon - y\|^2\big)$$

(ii) The KKT conditions for eqn (4) are

$$\nabla_{\hat{\theta_\alpha}}\mathcal{L}\Big(\hat{\theta}^*_\alpha, \gamma^*_\alpha\Big) = 2\Big(X^T(X\hat{\theta}^*_\alpha - y) + \gamma^*_\alpha\hat{\theta}^*_\alpha\Big) = 0$$
$$g_i(\hat{\theta^*}_\alpha) = \alpha - \|\hat{\theta}^*_\alpha\|^2 \geq 0$$
$$\gamma^*_\alpha \geq 0$$
$$\gamma^*_\alpha\big(\alpha - \|\hat{\theta}^*_\alpha\|^2\big) = 0$$

The KKT conditions for eqn (5) are

$$\nabla_{\hat{\theta}_\epsilon}\mathcal{L}\Big(\hat{\theta^*}_\epsilon, \gamma^*_\epsilon\Big) = 2\hat{\theta}^*_\epsilon + 2\gamma^*_\epsilon\Big(X^T(X\hat{\theta}^*_\epsilon - y)\Big) = 0$$
$$g_i(\hat{\theta}^*_\epsilon) = \epsilon - \|X\hat{\theta}^*_\epsilon - y\|^2 \geq 0$$
$$\gamma^*_\epsilon \geq 0$$
$$\gamma^*_\epsilon\Big(\epsilon - \|X\hat{\theta}^*_\epsilon - y\|^2\Big) = 0$$

(iii) Solving (3)

$$\nabla_\theta(\|X\hat{\theta}_\lambda - y\|^2 + \lambda\|\hat{\theta}_\lambda\|^2) = 0$$
$$2X^T(X\hat{\theta_\lambda} - y) + 2\lambda\hat{\theta}_\lambda = 0$$
$$\boxed{\hat{\theta_\lambda} = (X^TX + \lambda)^{-1}X^Ty}$$

Using this in the KKT conditions for eqn (4) are

$$X^T(X\hat{\theta_\lambda} - y) + \gamma_\alpha\hat{\theta_\lambda} = 0$$
$$\boxed{\gamma_\alpha = X^T(y - X\hat{\theta}_\lambda)(\hat{\theta}_\lambda)^{-1}}$$
$$\gamma_\alpha\big(\alpha - \|\hat{\theta_\lambda}\|^2\big) = 0 \implies \boxed{\alpha = \|\hat{\theta_\lambda}\|^2}$$

(iv) Using the solution of (3) and the KKT conditions for eqn (5) are

$$\hat{\theta_\epsilon} + \gamma_\epsilon\Big(X^T(X\hat{\theta_\epsilon} - y)\Big) = 0$$

$$\boxed{\gamma_\epsilon = (y - X\theta_\lambda)^{-1}(X^T)^{-1}\theta_\lambda}$$

$$\gamma_\epsilon\Big(\epsilon - \|X\hat{\theta_\lambda} - y\|^2\Big) = 0 \implies \boxed{\epsilon = \|X\hat{\theta_\lambda} - y\|^2}$$

(v) We can not claim $\hat{\theta_\lambda}$ is the solution from just the KKT conditions. To do this, we need to prove convexity of the optimization objective. We can prove the convexity of eqn (3) by showing that the hessian of the objective is positive semi-definite.

$$\textbf{.i.e.,}\quad \nabla^2_{\theta_\lambda}\Big(\|X\hat{\theta}_\lambda - y\|^2 + \lambda\|\hat{\theta}_\lambda\|^2\Big) \geq 0$$

$$\nabla^2_{\theta_\lambda}\Big(\|X\hat{\theta}_\lambda - y\|^2 + \lambda\|\hat{\theta}_\lambda\|^2\Big) = 2X^TX + 2\lambda I$$

$$= U(S + \lambda I)U^T \quad \text{where } X^TX = USU^T$$

$$\because diag(S + \lambda I) \geq 0, \quad U(S + \lambda I)U^T \geq 0$$

$$\boxed{\therefore \nabla^2_{\theta_\lambda}\Big(\|X\hat{\theta}_\lambda - y\|^2 + \lambda\|\hat{\theta}_\lambda\|^2\Big) \geq 0}$$

Hence the KKT conditions along with the convexity help show both the necessary and sufficient conditions of finding the solution.

Exercise 5: Project: Check Point 2

The project report with check point 2 is attached below.

ECE595 Project - Exploring Learning with Noisy Labels

Manish Nagaraj [1]

Abstract

This is checkpoints 1 and 2 of the project.

1. Checkpoint 1

1.1. Author Details

- **Name:** Manish Nagaraj
- **Major:** Electrical and Computer Engineering
- **Level:** Ph.D.

1.2. Details of Project

This project aims to study the limitations of the methods proposed in the paper referenced in the title (Ren et al., 2018b) using the CleanLab ML framework (Northcutt et al., 2019). Further, it also aims in identifying and implementing any algorithms or techniques that will overcome these limitations.

1.2.1. Datasets

For the purpose of evaluating existing algorithms as well as potential methods proposed, we will be using **CIFAR10**, **MNIST**, **FMNIST** datasets.

1.3. Reference Codes

We will be using the following code sources available as a reference to help implement the project.

1. CleanLab Framework
2. Learning to Learn by Gradient Descent

2. Checkpoint 2

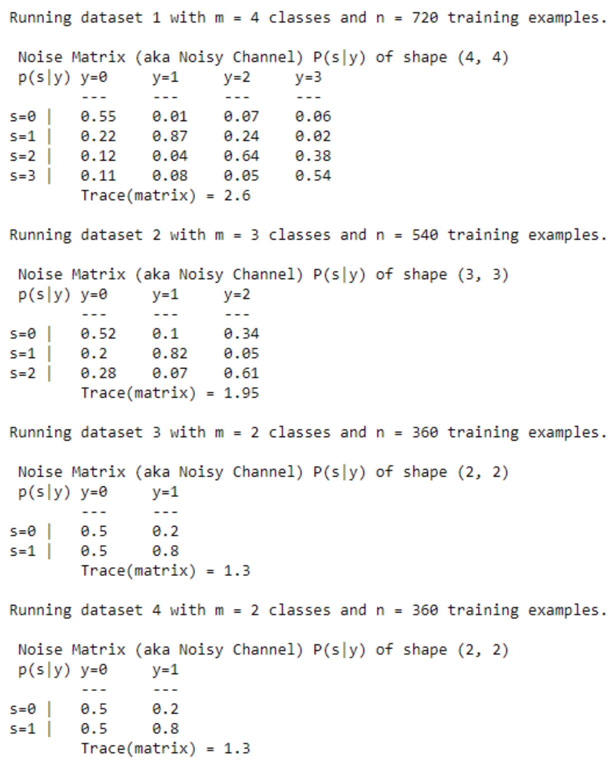
2.1. Checkpoint 2 Baseline

The baseline shows the performance of standard multi-class and binary classifiers in the presence of noisy labels.

[1]Department of Electrical and Computer Engineering, Purdue University, USA. Correspondence to: Manish Nagaraj <mnagara@purdue.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

The noise matrices that are used in the datasets are represented in the Figure 1. The s represents the observed noisy labels and the y represents the true labels. If the Trace of the matrix is 4, then there are no noisy values and if the trace is 0, then there all labels are noisy.



```
Running dataset 1 with m = 4 classes and n = 720 training examples.

 Noise Matrix (aka Noisy Channel) P(s|y) of shape (4, 4)
 p(s|y) y=0     y=1     y=2     y=3
        ---     ---     ---     ---
s=0 |   0.55    0.01    0.07    0.06
s=1 |   0.22    0.87    0.24    0.02
s=2 |   0.12    0.04    0.64    0.38
s=3 |   0.11    0.08    0.05    0.54
        Trace(matrix) = 2.6

Running dataset 2 with m = 3 classes and n = 540 training examples.

 Noise Matrix (aka Noisy Channel) P(s|y) of shape (3, 3)
 p(s|y) y=0     y=1     y=2
        ---     ---     ---
s=0 |   0.52    0.1     0.34
s=1 |   0.2     0.82    0.05
s=2 |   0.28    0.07    0.61
        Trace(matrix) = 1.95

Running dataset 3 with m = 2 classes and n = 360 training examples.

 Noise Matrix (aka Noisy Channel) P(s|y) of shape (2, 2)
 p(s|y) y=0     y=1
        ---     ---
s=0 |   0.5     0.2
s=1 |   0.5     0.8
        Trace(matrix) = 1.3

Running dataset 4 with m = 2 classes and n = 360 training examples.

 Noise Matrix (aka Noisy Channel) P(s|y) of shape (2, 2)
 p(s|y) y=0     y=1
        ---     ---
s=0 |   0.5     0.2
s=1 |   0.5     0.8
        Trace(matrix) = 1.3
```

Figure 1. $\mathbf{P}(s|y)$ Noise Matrices of the datasets used.

Figure 2 shows the baseline operation of standard binary and multi class classifiers on datasets that are generated. Figure 3 shows the operation of the same classifiers on the datasets with noisy labels. The label errors are circled in green. Figure 4 shows the performance of the classifiers when the inbuilt Learning with Noisy Labels function of CLEANLAB package is used. In each subfigure, the accuracy scores on a test set are shown as decimal values. The leftmost values in black are the test accuracies of the classifiers trained with perfect labels. The middle values in value are the accuracies are the test accuracies of the classifiers trained with noisy

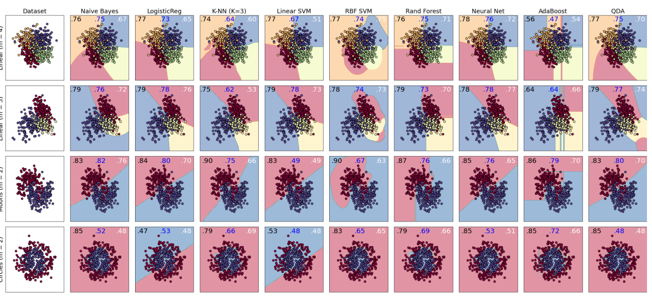


Figure 2. Baseline performance of classifiers with no noisy labels

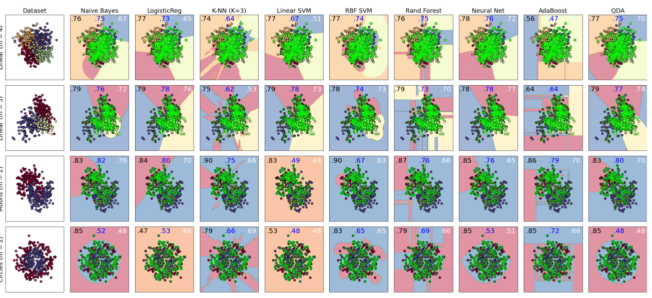


Figure 3. Baseline performance of classifiers with noisy labels (regular training)

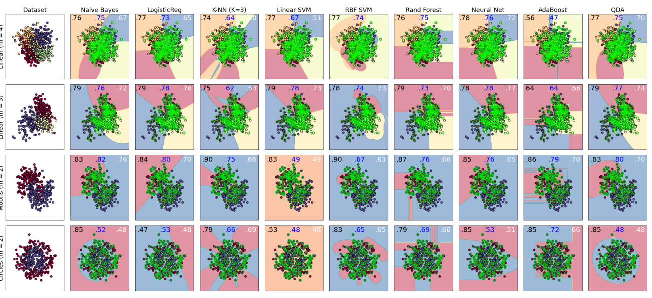


Figure 4. Performance of classifiers with noisy labels with CLEANLAB's inbuilt function

labels using CLEANLAB's inbuilt function. The rightmost values in white are the accuracies of the classifiers trained with noisy labels and no algorithm to rectify the noise.

References

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. Learning to learn by gradient descent by gradient descent. *arXiv preprint arXiv:1606.04474*, 2016.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

Northcutt, C. G., Wu, T., and Chuang, I. L. Learning with confident examples: Rank pruning for robust classification with noisy labels. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence*, UAI'17. AUAI Press, 2017. URL `http://auai.org/uai2017/proceedings/papers/35.pdf`.

Northcutt, C. G., Jiang, L., and Chuang, I. L. Confident learning: Estimating uncertainty in dataset labels, 2019.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. 2016.

Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., and Zemel, R. S. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018a.

Ren, M., Zeng, W., Yang, B., and Urtasun, R. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pp. 4334–4343. PMLR, 2018b.