Deep Learning and Human Society

Department of Electrical Engineering
University of Notre Dame, USA

Lecture 26, 27, 28

Introduction

- Deep learning and human society are on a collision course. Deep learning lies at the heart of self-driving cars. It is essential for many of the convenient applications on our mobile devices. It makes possible voice assistants (Alexa, Siri) and we rely on it in helping us find directions to our destination.
- Deep learning appears regularly in reports that raise warnings regarding implicit racial bias in commercial AI products (?, ?), voice concern over the use of ChatGPT in education, and express astonishment over DALL-E 2 creating art from natural language descriptions.
- The simple fact is that deep learning applications mimic human behavior in a manner that appears to pass the Turing test (a.k.a. imitation game) under non-expert examiners. This fact should give us pause to consider how this technology might be used in lay society.

Introduction

- Let us consider a scenario using demographic and economic data on residents of geographically based communities.
- These community datasets are gathered and used to help a city or other potential service providers decide how to serve community residents.
- But ultimately the idea is that the city or provider is using these datasets to decide how to invest in services that improve public health and safety.
- To make our problem more concrete, we consider a single city composed of several geographically distinct communities.

Introduction

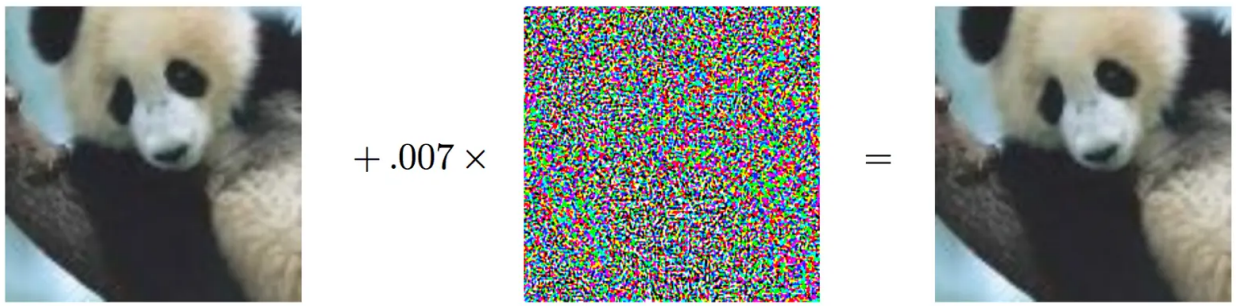
- Each community resident is characterized by a vector of *attributes*, a. Attribute vector components are either real-valued or categorical variables measuring that resident's measured attribute.
- The city uses the resident data to decide in which community it will place major infrastructure projects.
- These community datasets often contain the personal information of community residents. The private nature of this data raises a number of issues.
  - *Security:* One issue concerns the potential for external agents to either maliciously corrupt/alter the data to unfairly influence the city's decisions.
  - *Privacy:* Another issue concerns the unauthorized access of private/sensitive resident data.
  - *Equity:* We are also concerned with whether an ML algorithm used to make decisions regarding residents is "fair" with respect to the resident's gender, race, economic status.

Security - Adversarial Examples

- One security concern is with *adversarial examples*.
- These are perturbed data samples that are injected into a ML system to cause the system to make a false prediction or categorization.
- The maliciously perturbed data may be inputs to an inference version of the model, so the attacker can evade detection.
- These perturbed data samples can also be used during online ML training to compromise an existing model. This use of perturbed data is sometimes called data poisoning.

Security - Adversarial Examples

- Concern with adversarial examples was kicked off by a surprising discovery in (szegedy 2013) that found several ML models, including state-of-the-art deep networks are vulnerable to adversarial examples.
- That work found very slight perturbations of a correctly sampled input could trick the trained model into making the wrong classification.
- These adversarial examples could be generated by slightly perturbing a correctly sampled input in the training data.
- This example showed that neural network training may not select models whose performance is **robust** to variations in the input datasets.



Security - adversarial examples

- One way for addressing this issue is by generating adversarial examples and then adding those examples to the training data. There are several ways such adversarial examples can be generated. The following script shows a particularly simple example based on the MNIST dataset.
- We first train a highly accurate classifier on the MNIST dataset and then use it to create adversarial examples.

```
earlyStop = EarlyStopping(monitor='val_categorical_accuracy',
              min_delta=0, patience=10, verbose=0, mode='auto',
                          baseline=None, restore_best_weights=True)

mnist_model.fit(x_train, y_train, batch_size=128, epochs=100,
                 verbose=0, validation_data=(x_test, y_test),
                 callbacks=[earlyStop])

print(mnist_model.evaluate(x_train, y_train))
print(mnist_model.evaluate(x_test, y_test))

#1875/1875 #[==============================] - 11s #6ms/step
#   - loss: 0.1083 - categorical_accuracy: 0.9860
#    [0.10834111273288727,0.9860333204269409]
#
#313/313 [==============================] - 2s 6ms/step
#   - loss: 0.1168 - categorical_accuracy: 0.9845
#  [0.11677185446023941, 0.9845000505447388]
```

Security - Adversarial Example

We first try to create an adversarial example, by selecting an image in the data set that we want to perturb. We then create an image of pure noise and add it to the original image.

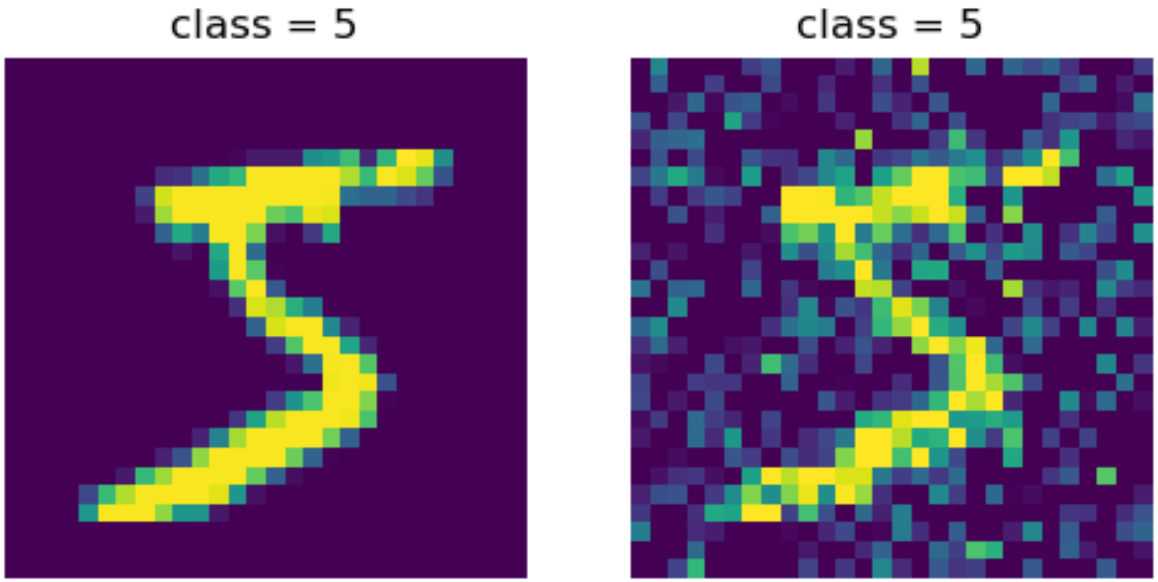
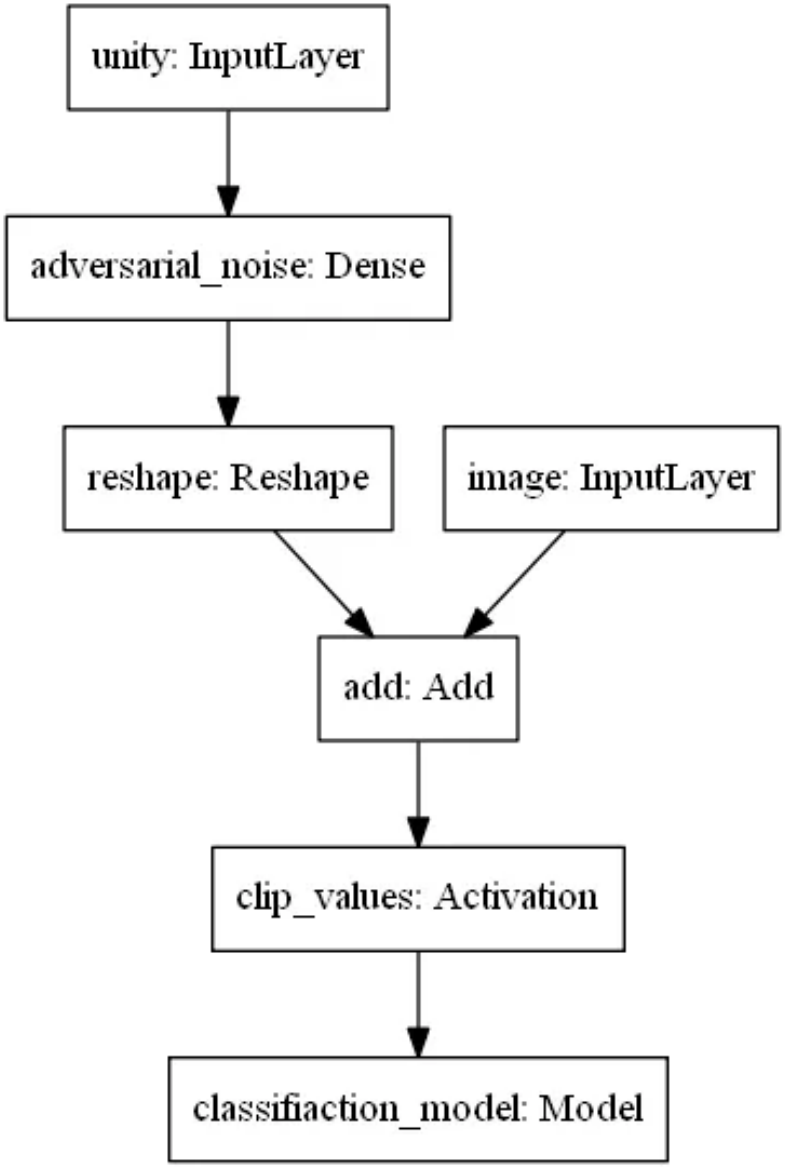


Figure: Selected image of the digit 5 and its noise perturbed version

Security - Adversarial Example

- Classifying the noisy image does not generate an incorrect classification. So we need to build a model to generate the adversarial example
- To generate a model capable of creating an adversarial example, we create a model with two inputs; selected image and noise.
- We retrain the noise image with a "mistargeted" loss. This means we freeze all weights except those in the noise layer.

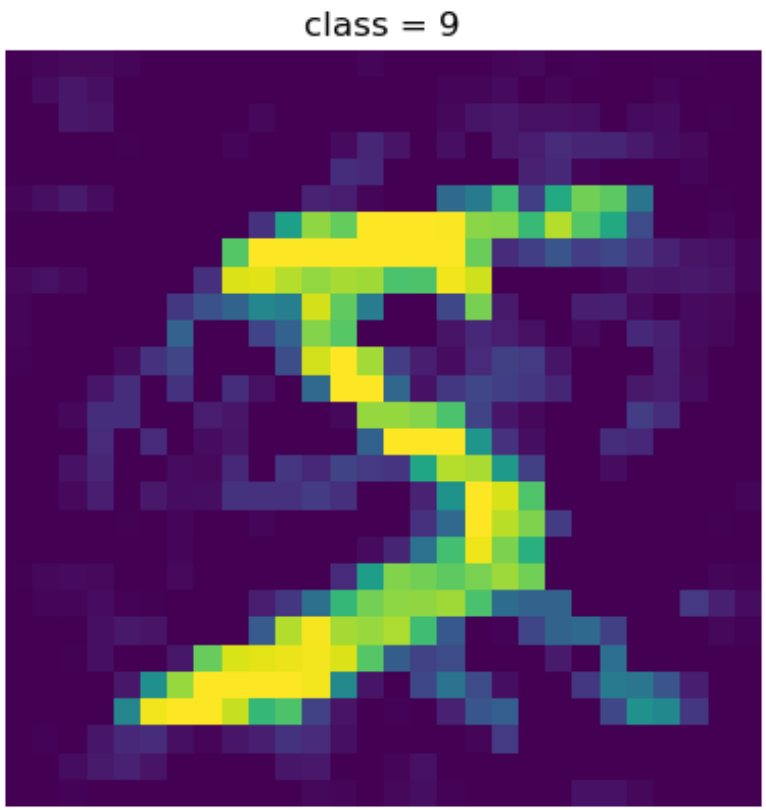


Security - adversarial example

```
regularizer = l2(0.01)
loss_function = 'categorical_crossentropy'
model = mnist_model

image = Input(shape=(28,28,1),name='image')
one   = Input(shape=(1,), name = 'unity')
noise = Dense(28*28,activation=None,use_bias=False,kernel_initializer='random_normal',
              kernel_regularizer=regularizer,name='adversarial_noise')(one)
noise = Reshape((28,28,1), name='reshape')(noise)

net = Add(name='add')([noise,image])
net = Activation('clip',name='clip_values')(net)

outputs = model(net)
adversarial_model = Model(inputs=[image,one], outputs=outputs)
adversarial_model.layers[-1].trainable = False

adversarial_model.compile(optimizer='nadam', loss=loss_function, metrics=[categorical_accura

adversarial_model.summary()

#target adversarial classification
target = 9                      #non-target
target_vector = np.zeros(10)
target_vector[target] = 1.

#train adversarial image
adversarial_model.fit(x={'image':img,'unity':np.ones(shape=(1,1))},
               y=target_vector.reshape(1,-1),epochs=10000,verbose=0,
                   callbacks=[checkpoint])
```

Security - adversarial example

The resulting perturbed image resulting in an incorrect classification of "9" was obtained by taking the original 5 image and adding the adversarial noise obtained during training of the adversarial model. This image indeed is classified by the MNIST model as 9, but if we look at what that image is in Fig. ??, we see little difference



Deep Learning with Differential Privacy

- Another security concern for community datasets regards the privacy of individual resident data when the entire dataset is being used to find out something about the community as a whole.
- The main concept used to address this issue with regards to databases is *differential privacy* [Dwork 2008].
- With regard to databases, differential privacy ensures that the removal or addition of a single database item does not substantially affect the outcome of any analysis or query.
- This provides a mathematically rigorous way to manage the fact that any query to a statistical database may disclose some bits of information about individual entries.

Deep Learning with Differential Privacy

- Database privacy concerns also appear in deep learning. The neural network is trained on a large dataset and for deep networks, the model's overparametrization means that some attributes of individual data entries may be disclosed by users of that model.
- These models should not disclose private information and one can develop algorithmic techniques for training that provide ϵ-differential guarantees for individual dataset entries. T
- Let us consider a *statistical database*. A statistic is a quantity computed from a sample. We suppose a trusted curator gathers sensitive information from a large number of residents (the sample) with the goal of learning (and releasing) statistics for the entire population.
- The problem is to release this statistical information without compromising the privacy of any individual resident.

Deep Learning with Differential Privacy

- We consider an *interactive* setting in which the curator sits between the users and the database. Queries from the users and responses to these queries may be modified by the curator to protect the privacy of the residents.
- We define the notion of *differential privacy* in the context of this interactively curated statistical database. Intuitively, differential privacy ensure that the removal or addition of a single database item does not substantially affect the outcome of any statistical analysis.
- We can formalize this notion as follows. Think of the database as a data matrix whose rows represent the attribute vectors of individual residents in the community. We define a randomized mechanism (a.k.a. algorithm) $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ that takes a dataset $D \in \mathcal{D}$ and randomly maps it to a statistic in $\mathcal{R}$.
- We say two datasets $D, D' \in \mathcal{D}$ are adjacent if they differ by one entry. We say this mechanism satisfies (ϵ)-differential privacy if for any two adjacent datasets $D, D' \in \mathcal{D}$ and for any subset $S \subset \mathcal{R}$ we have

$$\Pr\{\mathcal{M}(D) \in S\} \leq e^{\epsilon} \Pr\{\mathcal{M}(D') \in S\}$$

Deep Learning with Differential Privacy

- Any mechanism that satisfies this condition should address all concerns that any resident may have about leaking their personal information.
- This condition says that if a resident removes his/her data from the dataset, no output and so no consequences arising from that output will become more or less likely for the individual.
- For example, if the database were used by an insurer to determine whether or not to insure a resident, then whether or not the resident is in the database would have a negligible impact on whether the resident gets insured.
- Achieving differential privacy means that we hide the presence or absence of a single individual.
- Consider the query "How many rows in the database satisfy property P?" The presence or absence of a single row can effect the answer by at most 1.
- So a differentially private mechanism for a query of this type can be designed by first computing the true answer and then adding random noise so that for any z, z' for which $|z - z'| = 1$, we have $\Pr\{z\} \leq e^{\epsilon}\Pr\{z'\}$.

Deep Learning with Differential Privacy

- To see why this is so, consider any feasible response, r. For any m if m is the true answer to the query and the response is r then the random noise must have value $r - m$.
- Similarly, if $m - 1$ is the true answer and the response is r, then the random noise must have value $r - m + 1$.
- So for the response to be generated in a differentially private manner it suffices for

$$e^{-\epsilon} \leq \frac{\Pr\{\text{noise} = r - m\}}{\Pr\{\text{noise} = r - m + 1\}} \leq e^{\epsilon}$$

Deep Learning and Differential Privacy

- For deep learning models, one seeks to protect the privacy of the training data. The neural network model is our "mechanism".
- One might attempt to do this by working with the final parameters after training. In general, however, one does not have useful tight bounds on how these weights vary with the training data.
- A more sophisticated approach aims to control the influence of the training data during the training process; specifically during the stochastic gradient descent (SGD) computation.
- These SGD algorithms (?, ?) train a model with parameters θ by minimizing the empirical loss function $L(\theta)$.
- Each step of the SGD computes the gradient $\nabla_\theta L(\theta, x_i)$ for a random subset of examples, clips the ℓ_2 norm of each gradient, compute the average, add noise to protect privacy, and takes a step in the opposite direction of this average noisy gradient.
- At the end we also need to compute the privacy loss of the mechanism based on the information maintained by the curator.

Statistical Fairness in Supervised Learning

- Fairness is a key consideration when machines use algorithms to make decisions.
- Fairness is defined with respect to population groups that are marginalized in a legal or societal manner as a result of demographic factors (gender, race, age) or socio-economic factors.
- In particular, it means that decisions or favorable outcomes provided to groups are independent of that group is marginalized or not.
- "Fairness", however, has a number of formal definitions. This section considers statistical measures of fairness.

Statistical Notions of Fairness

- Before introducing statistical fairness measures, we will first review the classification problem to establish notational conventions commonly used the ML fairness literature.
- Classification determines a plausible value for an unknown *target* Y given observed inputs, X. Typically the target Y and inputs, X are jointly distributed random variables.
- At the time of classification the value of the target variable is unknown, but we observe an input X and make a guess $\widehat{Y} = h(X)$ based on what we observed.
- The function h is called a *classifier* or *predictor*. The output of the classifier is called a *label* or *prediction*.

Statistical Notions of Fairness

- We choose a classifier h out of a model set $\mathcal{H}$ that has good classification accuracy.
- Formally, this means we select h so that $\Pr\left\{Y = \widehat{Y}\right\}$ is close to 1. But there may be other criteria that we wish to consider as constraints on this optimization problems. These criteria take a closer look at how well the classifier works on positive targets ($Y = 1$) and negative targets ($Y = 0$).

Event	Condition	Resulting notion
$\widehat{Y} = 1$	$Y = 1$	True positive rate (TPR)
$\widehat{Y} = 0$	$Y = 1$	False negative rate
$\widehat{Y} = 1$	$Y = 0$	False positive rate (FPR)
$\widehat{Y} = 0$	$Y = 0$	True negative rate

Table: Common classification criteria

Statistical Notions of Fairness

- Rather than simply trying to maximize the $\Pr\left\{Y = \widehat{Y}\right\}$, we consider problems where the cost of true positives, false positives, true negatives, and false negatives may be different.
- The problem of optimal classification is then to find a classifier that minimizes the cost in this weighted expectation over the entire population.
- Note that since the false negative rate is equals $1 - \mathrm{TPR}$ and the true negative rate equals $1 - \mathrm{FPR}$, we really only need to consider model that maximize the expected value,

$$\mathbb{E}_{X,Y}\left\{\mathrm{TPR} - \lambda \times \mathrm{FPR}\right\}$$

where λ is a weight on the cost of the FPR.

Statistical Notions of Fairness

- The optimal classifier often has the problem that it can be realized as a threshold test applied to a risk score (a.k.a. sufficient statistic).

$$r(x) = \Pr\{Y = 1 \,|\, X = x\}$$

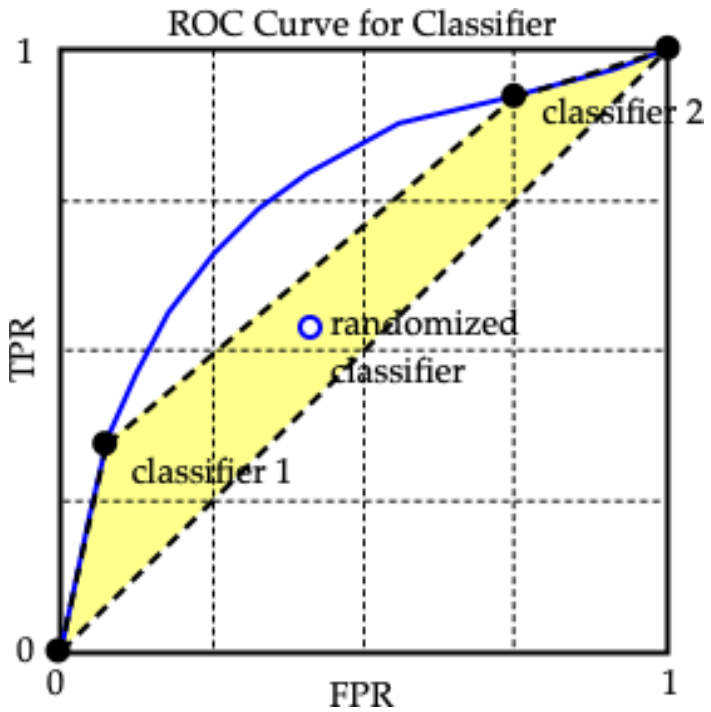
- This risk is, therefore, the posterior probability of outcome Y given X. In particular, if we seek to minimize the classification error $\Pr\left\{Y \neq \widehat{Y}\right\}$, then one can show that $\widehat{Y} = h(X)$ where

$$h(x) = \begin{cases} 1 & \text{if } r(x) = \Pr\{Y = 1 \,|\, X = x\} > 1/2 \\ 0 & \text{otherwise} \end{cases}$$

- In this case the threshold is $1/2$ for we assumed an equal cost for false positives and false negatives.

Statistical Notions of Fairness

- But if our weight λ is not equal to one, then $h(x)$ will select 1 with a different threshold level. Each choice of a threshold corresponds to a different weight λ on the FPR.
- By varying this threshold between 0 and 1, we trace out a curve in a two-dimensional space whose axes correspond to TPR and FPR. This curve is called the ROC curve (a.k.a. receiver operating characteristic)



Statistical Notions of Fairness

- "Discrimination" is concerned with socially relevant groups of people that have historically received unjust or systematically adverse treatment.
- U.S. law recognizes certain *protected categories* of people. These categories include race, sex, religion, disability status, and place of birth.
- We are interested in determining whether a classifier that was trained on input data, X is inherently discriminatory against of these protected categories, and if so what we can do to address the situation.
- "Fairness" is concerned with whether a protected category receives the same favorable outcomes under resource distribution as the unprotected category.
- In our problem we will use the random variable A to correspond to a protected $(A = 1)$ versus an unprotected group $(A = 0)$. The random variable, A is often called a *sensitive attribute*.

Statistical Notions of Fairness

- It is sometimes thought that removing or ignoring sensitive attributes in the training data will ensure the trained model's impartiality.
- This practice, however, is often ineffective and even harmful. The reason for this is that the other attributes in the training data that were not excluded may be positively correlated to the sensitive attribute. What this means is that when we train the model, then it will eventually learn labels that are highly correlated to the sensitive attribute.
- Statistical fairness criteria use statistical measures to determine whether the labels predicted by a learned classifier are discriminatory or not. Researchers have proposed dozens of different statistical fairness criteria, but we this section will confine its attention to two distinct measures known as demographic (statistical) parity and equal odds.

Demographic Parity

- This criteria requires that the outcome be statistically *independent* of the sensitive attribute, A.
- Formally this means

$$\Pr\left\{\widehat{Y} = 1 \,|\, A = a\right\} = \Pr\left\{\widehat{Y} = 1 \,|\, A = b\right\}$$

- This criteria also goes under the name of statistical parity or disparate impacts.
- In many cases, we relax this requirement for independence by allowing a positive slack, $\epsilon > 0$ such that

$$\left|\Pr\left\{\widehat{Y} = 1 \,|\, A = a\right\} - \Pr\left\{\widehat{Y} = 1 \,|\, A = b\right\}\right| \leq \epsilon$$

Demographic Parity

- Decisions and models based solely on "independence" (parity) can have undesirable outcomes.
- Consider, for example, a scenario in which a company hires applicants at the same rate from a protected and unprotected group. So the company's hiring practice satisfy demographics parity.
- But let us also assume that applicants in the unprotected group are select are selected more carefully than applicants from the protected group. This could lead to more "unqualified" hires being made from the protected group.
- This would have the unintended consequence of reinforcing pre-existing biases that hires from the protected group will always perform worse than hires from the unprotected group.

Equal Odds

- This criterion attempts to address the problem described above for demographic parity.
- In a typical classification problem, there is a difference between accepting an individual with a positive target versus accepting one with a negative target.
- The target variable, Y, may be viewed as providing a sense of the individual's *merit*.
- So the equal odds criteria attempts to ensure that the likelihood of a *qualified* person in the protected group receiving a favorable outcome is equal to the likelihood of a *qualified* person in the unprotected group receiving a favorable outcome.
- This is sometimes called a *separation* criterion and may be seen as being equivalent to the following

$$\begin{aligned} \Pr\left\{\widehat{Y} = 1 \,|\, Y = 1, A = a\right\} &= \Pr\left\{\widehat{Y} = 1 \,|\, Y = 1, A = b\right\} \\ \Pr\left\{\widehat{Y} = 1 \,|\, Y = 0, A = a\right\} &= \Pr\left\{\widehat{Y} = 1 \,|\, Y = 0, A = b\right\} \end{aligned}$$

This criteria is sometimes referred to as equalized odds or equal opportunity.

Equal Odds

- The idea of basing fairness on this "separation" principle has been controversial. This is particularly true when positive outcomes from an optimal classifier vary between the protected and unprotected group.
- Enforcing equality in this case may lead to worse classifier performance which leads many to question whether this is really fair?
- One response to that criticism is to focus on the cost of misclassification. In particular, if the protected group has been historically marginalized, the cost of denying resources to qualified people in this group may be seen as having a much higher societal cost.
- In this case, one can argue that sacrificing some degree of optimality to address historical biases may be in the better long-term interests of the community.

Optimal Classifier UCI Dataset

- We will evaluate these two criteria (demographic parity and equalized odds) for a classifier that was trained to maximize the likelihood of classifying a city resident as being either a high (income greater than 50k/year) or low (income less than 50k/year) wage earner.
- This optimal classifier will then be used as the baseline for developing ML methods that improve that model's fairness.
- The dataset we use to explore this is the UCI adult dataset.
- This dataset was derived from US census data and contains demographic information from several thousand individuals.

Optimal Classifier UCI Dataset

- We will take the UCI dataset and modify it to remove irrelevant or biased attributes (fnlwgt and gender). We remove gender because we are concerned with fairness w.r.t. gender.
- We will treat the variable `income` as the training target, so it too is dropped from the input dataset
- We need to modify the dataset to take care of missing values.
- Certain categorical features will be encoded as integers
- Other categorical features will be one-hot encoded.

Optimal Classifier UCI Adult Dataset

- The curated dataframe now has 88 columns, with the last one being the target variable. So we put the first 87 columns in the input data `X` and the last column in the target `y`.
- From the curated dataset, we now split the data 80% training and 20% testing. The training data is then split 80/20 between p-training and validation.
- This script uses `sklearn`'s `fit_transform` method to rescale the training inputs.
- We then declare a dense sequential model with two hidden layers of 16 and then 8 nodes. The output layer has a single node.
- So when compiling the model, we specify a binary crossentropy loss function and use the standard Adam optimizer.

Optimal Classifier UCI Adult Dataset

```
X = dataframe.iloc[:, 0:87].values
y  = dataframe.iloc[:, 8u].values

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
X_ptrain, X_val, y_ptrain, y_val = train_test_split(
                 X_train, y_train, test_size = .2)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_ptrain = sc.fit_transform(X_ptrain)
X_val    = sc.fit_transform(X_val)
X_test   = sc.transform(X_test)

import keras
from keras.models import Sequential
from keras.layers import Dense

classifier = Sequential()
classifier.add(Dense(units=16,
        activation = "relu", input_dim = 87))
classifier.add(Dense(units=8, activation = "relu")
classifier.add(Dense(units=1, activation = "sigmoid")

classifier.compile(optimizer = 'adam',
          loss = "binary_crossentropy",
          metrics = ["accuracy"])
```

Optimal Classifier UCI Adult Dataset

- I trained this model for 20 epochs with minibatches of 256. As usual, I save the history of training and validation loss/accuracy at each epoch so I could generate the training curves and saved the model with the best validation loss. The training curve in Fig. ?? suggests that the model was successfully trained with the best model achieving an accuracy of 85%.

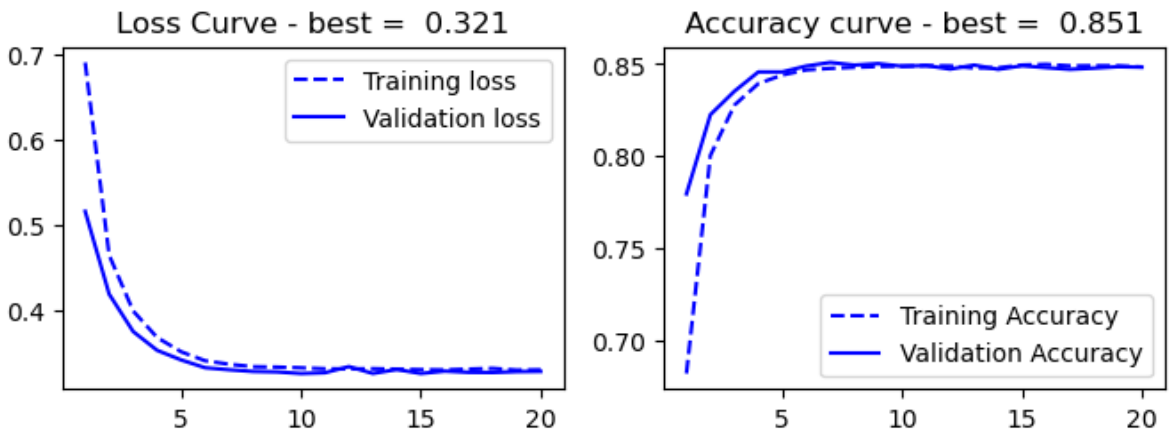
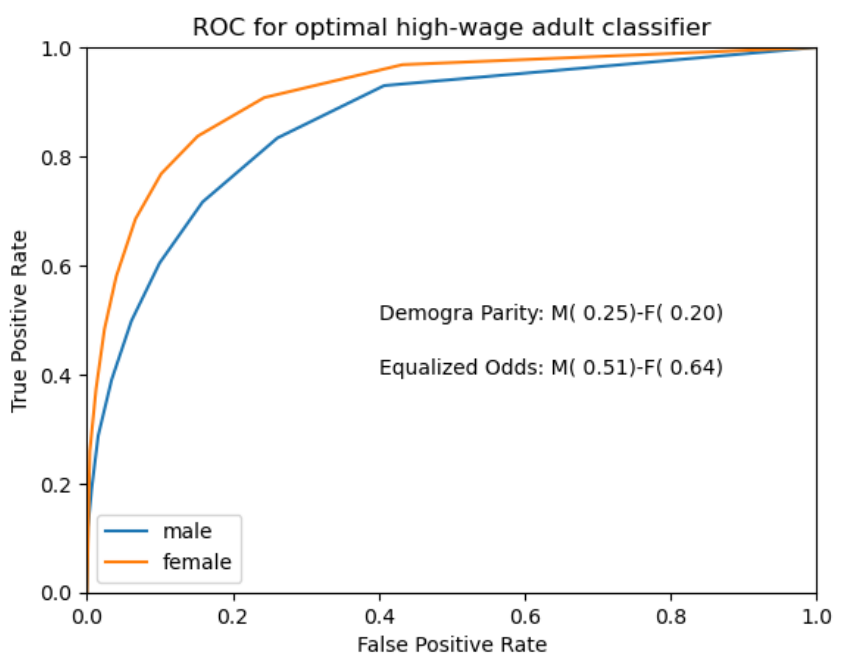


Figure: Training curves for model trained on adult UCI dataset

Optimal Classifier UCI Adult Dataset

- The "optimal" classifier was trained to maximize its accuracy. But we would like to obtain a more nuanced view of classifier performance with regard to the sensitive attribute of "gender".
- In particular, we compute the optimal classifier's equal opportunity and demographics parity metrics for male/female groups.
- We also varied the prediction threshold between 0 and 1 and plotted the male and female ROC curves.

```
ROC_male = np.empty(shape = (11,2))
ROC_female = np.empty(shape=(11,2))
ROC_male[0] = [1,1]
ROC_male[10] = [0,0]
ROC_female[0] = [1,1]
ROC_female[10] = [0,0]

for k in range(9):
    thresh = float((k+1)/10)
    male_positive = (y_pred_male > thresh).astype("float64").reshape((len(y_pred_male)))
    female_positive = (y_pred_female > thresh).astype("float64").reshape((len(y_pred_female;

    indx_male = np.where(y_male==1.)
    true_male_positive = male_positive[indx_male].astype("float64")
    indx_female = np.where(y_female==1.)
    true_female_positive = female_positive[indx_female].astype("float64")

    TPR_male = np.sum(true_male_positive)/len(true_male_positive)
    TPR_female = np.sum(true_female_positive)/len(true_female_positive)

    male_negative = (y_pred_male < thresh).astype("float64").reshape((len(y_pred_male)))
    female_negative = (y_pred_female < thresh).astype("float64").reshape((len(y_pred_female;

    indx_male = np.where(y_male==0.)
    false_male_positive = male_negative[indx_male].astype("float64")
    indx_female = np.where(y_female==0.)
    false_female_positive = female_negative[indx_female].astype("float64")

    FPR_male = 1-np.sum(false_male_positive)/len(false_male_positive)
    FPR_female = 1-np.sum(false_female_positive)/len(false_female_positive)

    ROC_male[k+1] = [TPR_male,FPR_male]
    ROC_female[k+1] = [TPR_female, FPR_female]


plt.plot(ROC_male[:,1],ROC_male[:,0],label="male")
plt.plot(ROC_female[:,1],ROC_female[:,0],label="female")
plt.xlim(0,1)
plt.ylim(0,1)
plt.legend()
plt.title('ROC for optimal high-wage adult classifier')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')

plt.text(.4,.5,f"Demogra Parity: M({male_DP: .2f})-F({female_DP: .2f})")
plt.text(.4,.4,f"Equalized Odds: M({male_EO: .2f})-F({female_EO: .2f})")
```



Addressing Discrimination in Optimal Adult Classifier

There are three ways one can address discrimination in the optimal classifier.

- *Pre-processing:* [Kamiran 2012, Calmon 2017] This method transforms the input dataset, X to remove any correlation of the data and the sensitive attribute.
- *In-training:* [Zafar 2017, Kamishima 2012, Sattiegeri 2019] These methods involve adding a fairness constraint into the training of the model. This can either be done. by treating the fairness constraint as a regularization kernel or through adversarial training.
- *Post-processing:* [Hardt 2016, Pleiss 2017] This method adjusts a learned classifier so it is uncorrelated with the sensitive attribute. These methods usually involve the development of a randomized model

Summary

- Deep learning is becoming a major force in the everyday lives of members of human communities. Recent applications involving Generative and Transformer models appear to pass the Turing test to non-expert examiners. ML algorithms are being trained to use demographic and economic data of community residents to decide on how and who should receive services. Since resident data is often "private", we need to address three major concerns; security, privacy, and fairness.
- We examined security with respect to the ease with which adversarial examples can be created for trained neural networks. These adversarial examples can be used by malicious agents or they can be used to expand a model's training set and thereby improve model robustness to maliciously perturbed inputs.
- The concern with adversarial examples was kicked off by a surprising discovery in (?, ?) that found several ML models, including state-of-the-art deep networks are vulnerable to adversarial examples. That work found very slight perturbations of a correctly sampled input could trick the trained model into making the wrong classification.

Summary

- Adversarial examples are generated by taking a known input sample and perturbing it so it is classified incorrectly. That small perturbation may not necessarily be i.i.d. random noise being added to the picture. But it can be noisy patterns with particular correlations.
- We generate an adversarial example from an *adversary model*. This model takes two inputs; the test input (image) that we wish to perturb and a "noise" input pattern. The noisy input is then passed through an optimal classifier and we compute a loss which is mistargeted by being the difference between the prediction and the "desired" incorrect classification. We freeze all model weights except those on the noisy input pattern. We then use backpropagation to find the input pattern that minimizes the mistargeted loss. This method generates a specific noise pattern that misclassifies the specified input.
- Another security concerns regards the privacy of individual resident data. This chapter reviewed the Dwork's differential privacy concept. Differential privacy is defined with respect to databases and it ensures that the removal or addition of a single database item does not substantially impact the outcome of any statistical query to the database.

Summary

- This database privacy notion can be applied to neural networks since the deep learning model is trained on a large dataset so that the neural network becomes the mechanism a dataset curator uses to generate a response to an input query. Differential privacy basically involves adding noise to the input samples during training.
- Fairness is a key consideration when ML algorithms are used to make decisions about who gets and does not get services. There are certain categories of residents who have legal protected status based on gender, race, age, etc. Fairness essentially means that outcomes should be equally distributed between those in and out of the protected group. Statistical fairness interprets this equality in terms of conditional probabilities.
- There are two basic notions of fairness considered in this chapter. These are based on the "independence" of outcomes with respect to the group's protected attribute. The other is based on the notion of separation. Demographic (statistical) parity is an example of a fairness criterion based on independence. Equal odds is an example of a fairness criterion based upon separating out those individuals qualified to receive a favorable outcome.

Summary

- Demographic parity requires that the outcome be statistically *independent* of the sensitive attribute, A. Formally this means

$$\Pr\left\{\widehat{Y} = 1 \,|\, A = a\right\} = \Pr\left\{\widehat{Y} = 1 \,|\, A = b\right\}$$

  This criteria also goes under the name of statistical parity or disparate impacts.

- In many cases, we relax this requirement for independence by allowing a positive slack, $\epsilon > 0$ such that

$$\left|\Pr\left\{\widehat{Y} = 1 \,|\, A = a\right\} - \Pr\left\{\widehat{Y} = 1 \,|\, A = b\right\}\right| \leq \epsilon$$

- Decisions and models based solely on "independence" (parity) can have undesirable consequences by allowing more "unqualified" residents in the protected group to receive benefits than those in "qualified" residents.

Summary

- Equal Odds may be seen as being equivalent to the following

$$\begin{aligned} \Pr\left\{\widehat{Y}=1 \,|\, Y=1, A=a\right\} &= \Pr\left\{\widehat{Y}=1 \,|\, Y=1, A=b\right\} \\ \Pr\left\{\widehat{Y}=1 \,|\, Y=0, A=a\right\} &= \Pr\left\{\widehat{Y}=1 \,|\, Y=0, A=b\right\} \end{aligned}$$

  where the target Y is seen as a surrogate for an individual's "merit" to receive a positive outcome.

- The idea of basing fairness on this "separation" principle has been controversial. Enforcing equality in this case may lead to worse classifier performance which leads many to question whether this is really fair?

Summary

- There are three ways one can address discrimination in the optimal classifier. These methods are categorized as
  - *Pre-processing:* This method transforms the input dataset, X to remove any correlation of the data and the sensitive attribute.
  - *In-training:* These methods involving adding a fairness constraint into the training of the model. This can either be done. by treating the fairness constraint as a regularization kernel or through adversarial training.
  - *Post-processing:* This method adjusts a learned classifier so it is uncorrelated with the sensitive attribute. These methods usually involve the development of a randomized model.