Homework 2

Spring 25, CS 442: Trustworthy Machine Learning
Due Friday Mar. 28th at 23:59 CT

Instructor: Han Zhao

Instructions for submission All the homework submissions should be typeset in LaTeX. For all the questions, please clearly justify each step in your derivations or proofs.

1 Statistical Parity and Equalized Odds [10 pts]

1.1 [5 pts]

Construct three binary random variables X, A and Y such that X is independent of A, but X is dependent of A given Y.

Solution:
Let X and A be independent binary random variables defined by

$$P(X=0) = P(X=1) = \frac{1}{2}, \quad P(A=0) = P(A=1) = \frac{1}{2}.$$

Define the binary random variable Y as

$$Y = \mathbf{1}_{\{X=A\}},$$

i.e.,

$$Y = \begin{cases} 1, & \text{if } X = A, \\ 0, & \text{if } X \neq A. \end{cases}$$

1.2 [5 pts]

In the course we have seen the following incompatibility theorem between statistical parity and equalized odds for a binary classification problem:

Theorem 1: Incompatibility Theorem

Assume that Y and A are binary random variables, then for any binary classifier $\widehat{Y}$, statistical parity and equalized odds are mutually exclusive unless $A \perp Y$ or $\widehat{Y} \perp Y$.

Give an example of a classification problem where the target variable Y can take three distinct values, and such that statistical parity and equalized odds are simultaneously achievable.

Solution:
Let the target variable be $Y \in \{0, 1, 2\}$ and the sensitive attribute be $A \in \{0, 1\}$. Assume that Y and A are independent, i.e.,

$$P(Y = y \mid A = 0) = P(Y = y \mid A = 1) \quad \text{for each } y \in \{0, 1, 2\}.$$

Now consider a perfect classifier defined by

$$\widehat{Y} = Y.$$

This satisfies the conditions of the problem because:

- For **equalized odds**, note that for each $y \in \{0, 1, 2\}$ and for each $a \in \{0, 1\}$,

$$P(\widehat{Y} = y \mid Y = y, A = a) = 1.$$

- For **statistical parity**, because Y (and hence $\widehat{Y}$) is independent of A,

$$P(\widehat{Y} = y \mid A = 0) = P(\widehat{Y} = y \mid A = 1) = P(Y = y).$$

Thus, both statistical parity and equalized odds are simultaneously achieved.

2 Asymmetric Binary Distinguishing Game [20 pts]

Consider a binary distinguishing game between Alice (A) and Bob (B) as follows. Let P, Q be two distributions that are known to both A and B. Every round, A will flip a biased coin S, such that $\Pr(S = \text{Head}) = p$ and $\Pr(S = \text{Tail}) = 1 - p$ for some constant $0 < p < 1$. Then, upon observing the outcome of the coin flipping, A will draw sample X from a distribution D as follows:

$$D = \begin{cases} P & \text{If } S = \text{Head} \\ Q & \text{If } S = \text{Tail} \end{cases}$$

The outcome of the coin flipping will not be revealed to B. Instead, A will show the drawn sample $X \sim D$ to B, and the goal of B is to guess from which distribution $(P$ or $Q)$ the sample X is drawn from.

2.1 Error of a Randomized Strategy [10 pts]

Let $\eta : \mathcal{X} \to [0, 1]$ be the conditional probability of B guess that the given sample X is drawn from P, i.e.,

$$\eta(X) := \Pr(B \text{ guesses } X \sim P \mid X).$$

Derive the expected error of this strategy as a function of η.

Solution:
The error occurs when:

- X is drawn from P (which happens with probability p) and Bob incorrectly guesses Q (with probability $1 - \eta(X)$).
- X is drawn from Q (which happens with probability $1 - p$) and Bob incorrectly guesses P (with probability $\eta(X)$).

Thus, the expected error is:

$$\begin{aligned}\mathrm{err}(\eta) &= p\,\mathbb{E}_{X\sim P}[1-\eta(X)] + (1-p)\,\mathbb{E}_{X\sim Q}[\eta(X)] \\ &= p\int_{\mathcal{X}} (1-\eta(x))P(x)\,dx + (1-p)\int_{\mathcal{X}} \eta(x)Q(x)\,dx.\end{aligned}$$

2.2 Optimal Strategy and Optimal Error Rate [10 pts]

Based on your answer from the last question, derive the optimal strategy η^* and the corresponding optimal error rate of B.

Solution:
For each $x \in \mathcal{X}$, the contribution to the error is:

$$f(\eta(x)) = p\,P(x)\big(1-\eta(x)\big) + (1-p)\,Q(x)\,\eta(x).$$

Since $f(\eta(x))$ is linear in $\eta(x)$, the optimum is achieved at the endpoints. That is, the optimal strategy can be expressed like:

$$\eta^*(x) = \mathbf{1}\{p\,P(x) > (1-p)\,Q(x)\}.$$

The corresponding optimal error rate is then given by:

$$\mathrm{err}(\eta^*) = p\int_{\{x:p\,P(x)\le(1-p)\,Q(x)\}} P(x)\,dx + (1-p)\int_{\{x:p\,P(x)>(1-p)\,Q(x)\}} Q(x)\,dx.$$

3 Non-trivial Prediction of the Protected Attribute [10 pts]

Let the tuple (X, A, Y) be the random variables corresponding to input data, the protected attribute and the target variable, respectively. In many cases we can predict both Y and A from the same data X with reasonable accuracy. Suppose we have a classifier g to predict Y from X. Define the statistical disparity of g as

$$\Delta_{\mathrm{DP}}(g) := |\Pr_{A=0}(g(X)=1) - \Pr_{A=1}(g(X)=1)|,$$

where we use $\Pr_{A=a}(\cdot)$ to denote the conditional probability of an event conditioned on $A = a$. Clearly, if $\Delta_{\mathrm{DP}}(g) = 0$, then g satisfies the statistical parity condition. Show that there exists a classifier h to predict A from X such that the following error bound holds:

$$\varepsilon_{A=0}(h) + \varepsilon_{A=1}(h) \le 1 - \Delta_{\mathrm{DP}}(g),$$

where $\varepsilon_{A=a}(h) := \Pr_{A=a}[h(X) \ne a]$.

Solution:
Let, $p_0 = \Pr_{A=0}\big(g(X)=1\big)$ and $p_1 = \Pr_{A=1}\big(g(X)=1\big)$. Then by definition,

$$\Delta_{\mathrm{DP}}(g) = \big|\Pr_{A=0}\big(g(X)=1\big) - \Pr_{A=1}\big(g(X)=1\big)\big| = |\,p_0 - p_1\,|.$$

Without loss of generality, assume $p_1 \ \geq \ p_0$. Thus,

$$\Delta_{\text{DP}}(g) \ = \ p_1 \ - \ p_0.$$

Define a new classifier h that predicts the protected attribute A from X via

$$h(x) \ = \ \begin{cases} 1, & \text{if } g(x) = 1, \\ 0, & \text{otherwise.} \end{cases}$$

(If instead $p_0 > p_1$, simply swap the roles of 0 and 1 in the definition of h.)
Then,

$$\varepsilon_{A=0}(h) \ = \ \Pr_{A=0}\big[h(X) \neq 0\big] \ = \ \Pr_{A=0}\big[g(X) = 1\big] \ = \ p_0,$$

$$\varepsilon_{A=1}(h) \ = \ \Pr_{A=1}\big[h(X) \neq 1\big] \ = \ \Pr_{A=1}\big[g(X) = 0\big] \ = \ 1 - p_1.$$

Therefore,

$$\varepsilon_{A=0}(h) \ + \ \varepsilon_{A=1}(h) \ = \ p_0 \ + \ \big(1 - p_1\big) \ = \ 1 \ - \ \big(p_1 - p_0\big) \ = \ 1 \ - \ \Delta_{\text{DP}}(g).$$

This shows the existence of a classifier h predicting A from X such that

$$\varepsilon_{A=0}(h) \ + \ \varepsilon_{A=1}(h) \ \leq \ 1 \ - \ \Delta_{\text{DP}}(g),$$

4 Fair Representations [40 pts]

In this problem, we will show that fair representations whose distributions are conditionally aligned will not exacerbate the statistical disparity. Again, let the tuple (X, A, Y) be the random variables corresponding to input data, the protected attribute and the target variable, respectively. In this problem, we assume both A and Y to be binary variables.
Consider representations $Z = g(X)$ such that $Z \perp A \mid Y$. For a classifier $\widehat{Y} = h(Z)$ that acts on the representations Z, let $\Delta_{\text{DP}}(\widehat{Y}) := |\Pr_{A=0}(\widehat{Y} = 1) - \Pr_{A=1}(\widehat{Y} = 1)|$.

4.1 [10pts]

Show that for any classifier h that acts on the representations $Z = g(X)$, $\widehat{Y} = h(Z)$ satisfies equalized odds.
Solution:
Equalized odds requires that for each $y \in \{0, 1\}$ and $a \in \{0, 1\}$,

$$\Pr(\widehat{Y} = 1 \mid Y = y, \ A = a) \quad \text{is the same across both } a = 0 \text{ and } a = 1.$$

Since $Z \perp A \mid Y$, for each fixed $y \in \{0, 1\}$ and $a \in \{0, 1\}$,

$$\Pr(\widehat{Y} = 1 \mid Y = y, \ A = a) \ = \ \Pr\big[h(Z) = 1 \mid Y = y, \ A = a\big] \ = \ \Pr\big[h(Z) = 1 \mid Y = y\big],$$

where the last equality uses the conditional independence $Z \perp A \mid Y$ (and the fact that $\widehat{Y}$ depends only on Z). Hence it does not depend on a, and thus the true positive rates and false positive rates are the same for both $A = 0$ and $A = 1$. Therefore, $\widehat{Y}$ satisfies the equalized odds condition.

4.2 [20pts]

Define $\gamma_a := \Pr_{A=a}(Y = 0)$. Show that for any classifier h over Z, the following inequality holds:

$$\left|\Pr_{A=0}(\widehat{Y} = y) - \Pr_{A=1}(\widehat{Y} = y)\right| \leq |\gamma_0 - \gamma_1| \cdot \left(\Pr(\widehat{Y} = y \mid Y = 0) + \Pr(\widehat{Y} = y \mid Y = 1)\right), \forall y \in \{0, 1\}.$$

Solution:
For any group $A = a$, we have

$$\Pr_{A=a}(\widehat{Y} = y) = \Pr_{A=a}\big(\widehat{Y} = y, Y = 0\big) + \Pr_{A=a}\big(\widehat{Y} = y, Y = 1\big).$$

Because $Z \perp A \mid Y$, the probability $\Pr_{A=a}(\widehat{Y} = y \mid Y = y')$ does not depend on a; it only depends on y'. Denote $p_{y'} = \Pr(\widehat{Y} = y \mid Y = y')$. Then for $A = a$,

$$\Pr_{A=a}(\widehat{Y} = y) = \Pr_{A=a}(Y = 0) \cdot p_0 + \Pr_{A=a}(Y = 1) \cdot p_1.$$

Using the definition of $\gamma_a = \Pr_{A=a}(Y = 0)$, we get:

$$\Pr_{A=a}(\widehat{Y} = y) = p_0\,\gamma_a + p_1\,(1 - \gamma_a).$$

Hence the difference between groups $A = 0$ and $A = 1$ is:

$$\Pr_{A=0}(\widehat{Y} = y) - \Pr_{A=1}(\widehat{Y} = y) = \Big[p_0\,\gamma_0 + p_1\,(1 - \gamma_0)\Big] - \Big[p_0\,\gamma_1 + p_1\,(1 - \gamma_1)\Big].$$

$$= p_0\,(\gamma_0 - \gamma_1) + p_1\,\big[(1 - \gamma_0) - (1 - \gamma_1)\big] = (\gamma_0 - \gamma_1)\,\big[p_0 - p_1\big].$$

Taking absolute values,

$$\big|\Pr_{A=0}(\widehat{Y} = y) - \Pr_{A=1}(\widehat{Y} = y)\big| = |\gamma_0 - \gamma_1|\,\big|p_0 - p_1\big|.$$

But $p_0 = \Pr(\widehat{Y} = y \mid Y = 0)$ and $p_1 = \Pr(\widehat{Y} = y \mid Y = 1)$ imply

$$\big|p_0 - p_1\big| \leq \Pr(\widehat{Y} = y \mid Y = 0) + \Pr(\widehat{Y} = y \mid Y = 1).$$

Thus,

$$\big|\Pr_{A=0}(\widehat{Y} = y) - \Pr_{A=1}(\widehat{Y} = y)\big| \leq |\gamma_0 - \gamma_1| \cdot \Big(\Pr(\widehat{Y} = y \mid Y = 0) + \Pr(\widehat{Y} = y \mid Y = 1)\Big),$$

4.3 [10pts]

Prove that for any classifier $\widehat{Y} = h(Z)$, $\Delta_{\mathrm{DP}}(h \circ g) \leq \Delta_{\mathrm{BR}}$, where $\Delta_{\mathrm{BR}} := |\gamma_0 - \gamma_1|$ is the difference of base rates. Note: this proposition states that if a classifier satisfies equalized odds, then it will not exacerbate the statistical disparity of the optimal classifier.

Solution:
By **4.1**, the classifier $\widehat{Y}$ satisfies equalized odds. Hence from the derivation in **4.2**, for $y = 1$ we get

$$\big|\Pr_{A=0}(\widehat{Y} = 1) - \Pr_{A=1}(\widehat{Y} = 1)\big| = |\gamma_0 - \gamma_1| \cdot \big|\Pr(\widehat{Y} = 1 \mid Y = 0) - \Pr(\widehat{Y} = 1 \mid Y = 1)\big|.$$

Because $\big|\Pr(\widehat{Y} = 1 \mid Y = 0) - \Pr(\widehat{Y} = 1 \mid Y = 1)\big| \leq 1$, it follows that

$$\Delta_{\mathrm{DP}}(\widehat{Y}) = \big|\Pr_{A=0}(\widehat{Y} = 1) - \Pr_{A=1}(\widehat{Y} = 1)\big| \leq |\gamma_0 - \gamma_1| = \Delta_{\mathrm{BR}}.$$

Therefore, the statistical disparity of $\widehat{Y}$ cannot exceed the difference in base rates.

5 Coding [20 pts]

In this problem, you need to **1)** train a model to approximately achieve *statistical parity* using adversarial training (as introduced in Lecture 8), and **2)** train a model to approximately achieve *equalized odds* using conditional learning (as introduced in Lecture 10).
First, you need to download and unzip "cs442_hw2_code.zip". This folder consists of one subfolder and five files.

- The "data" subfolder includes two datasets—Adult and COMPAS.
- The "dataset.py" file processes these two datasets. Specifically, on the Adult dataset, "sex" is selected as the protected attribute while "income" is selected as the target attribute. On the COMPAS dataset, "African_American" is selected as the protected attribute while "Two_yr_Recidivism" is selected as the target attribute.
- The "models.py" file contains the model to be trained with adversarial training (called "FairNet") and the model to be trained with conditional learning (called "CFairNet").
- The "utils.py" file contains functions to compute errors.
- The "main_adult.py" file is the main file to run to get the performance on the Adult dataset. Once you implement the following functions, you can simply run:

  ```
  python main_adult.py -m fair
  python main_adult.py -m cfair-eo
  ```

  to get different errors. You can further specify "-u" to change the weight of adversarial classification loss. (Please specify as 0.1, 1, and 10, respectively.)
- The "main_compas.py" file is the main file to run to get the performance on the COMPAS dataset. Once you implement the following functions, you can simply run:

  ```
  python main_compas.py -m fair
  python main_compas.py -m cfair-eo
  ```

  to get different errors. You can further specify "-u" to change the weight of adversarial classification loss. (Please specify as 0.1, 1, and 10, respectively.)

For this problem, you will need to implement these functions following the corresponding docstrings.

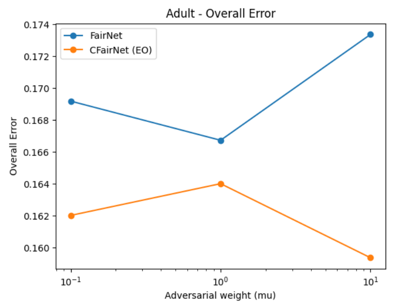
a. In "models.py" file, please implement the "forward" function and the "backward" function in the GradReverse Class.

b. In "models.py" file, please implement the "forward" function in the FairNet Class.

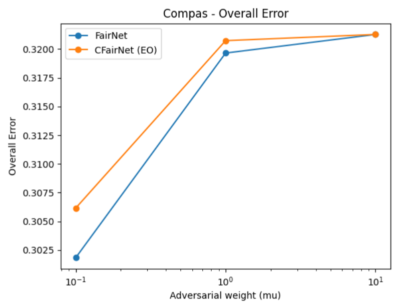
c. In "models.py" file, please implement the "forward" function in the CFairNet Class.

5.1 [10pts]

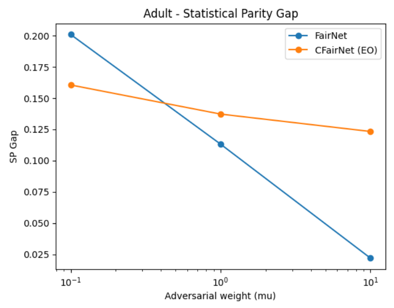
Please plot three figures for each dataset. In these figures, x-axis is the weight of adversarial classification loss (*i.e.*, 0.1, 1, and 10), while the y-axis is a) the overall predicted error ($\Pr(\hat{Y} \neq Y)$), b) the statistical parity gap ($|\Pr(\hat{Y}=1 \mid A=0) - \Pr(\hat{Y}=1 \mid A=1)|$), and c) the equalized odds gap ($0.5 * |\Pr(\hat{Y}=1 \mid A=0, Y=0) - \Pr(\hat{Y}=1 \mid A=1, Y=0)| + 0.5 * |\Pr(\hat{Y}=1 \mid A=0, Y=1) - \Pr(\hat{Y}=1 \mid A=1, Y=1)|$), respectively.



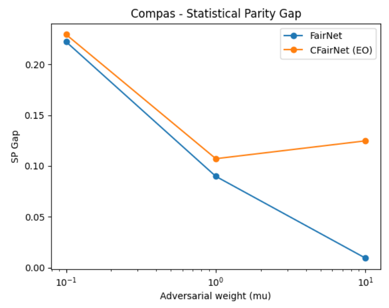
(a) Adult - Overall Error



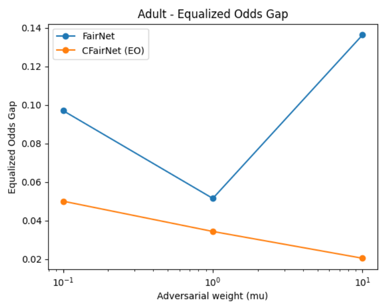
(b) Compas - Overall Error



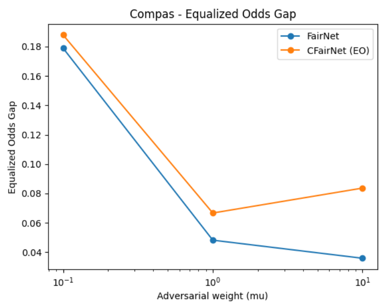
(c) Adult - Statistical Parity Gap



(d) Compas - Statistical Parity Gap



(e) Adult - Equalized Odds Gap



(f) Compas - Equalized Odds Gap

Figure 1: Fairness and Performance Metrics Comparison for Adult and Compas Datasets

5.2 [5pts]

Perform an analysis on the results (*e.g.*, What are the relationships between the weight of adversarial classification loss and the above notions? What are the relationships between the overall predicted error and the other two fairness notions?)
Solution:
When μ is small, the classifier emphasizes predictive accuracy over fairness. Hence, overall error tends to be smaller, but fairness metrics are larger. As a result, the overall classification error increases slightly, but the statistical parity gap and equalized odds gap decrease. This illustrates a fairness-accuracy tradeoff. Moreover, CFairNet often yields a lower equalized odds gap than FairNet, as it directly enforces conditional fairness.

5.3 [5pts]

Show your implemented codes.

Implemented Code for (a):

```
class GradReverse(Function):
    """
    Implement the gradient reversal layer for the convenience of domain adaptatic
    The forward part is the identity function while the backward part is the nega
    """

    @staticmethod
    def forward(ctx, x):
        ### YOUR CODES HERE ###
        return x

    @staticmethod
    def backward(ctx, grad_output):
        ### YOUR CODES HERE ###
        return -grad_output
```

Implemented Code for (b):

```
class FairNet(nn.Module):
    """
    Multi-layer perceptron with adversarial training for fairness.
    """

    def __init__(self, configs):
        super(FairNet, self).__init__()
        self.input_dim = configs["input_dim"]
        self.num_classes = configs["num_classes"]
        self.num_hidden_layers = len(configs["hidden_layers"])
        self.num_neurons = [self.input_dim] + configs["hidden_layers"]
        # Parameters of hidden, fully-connected layers, feature learning componen
        self.hiddens = nn.ModuleList([nn.Linear(self.num_neurons[i], self.num_neu
                                      for i in range(self.num_hidden_layers)])
        # Parameter of the final softmax classification layer.
        self.softmax = nn.Linear(self.num_neurons[-1], configs["num_classes"])
        # Parameter of the adversary classification layer.
        self.num_adversaries = [self.num_neurons[-1]] + configs["adversary_layers
        self.num_adversaries_layers = len(configs["adversary_layers"])
        self.adversaries = nn.ModuleList([nn.Linear(self.num_adversaries[i], self
                                          for i in range(self.num_adversaries_lay
        self.sensitive_cls = nn.Linear(self.num_adversaries[-1], 2)

    def forward(self, inputs):
        """
        The feature extractor is specified by self.hiddens.
        The label predictor is specified by self.softmax.
        The adversarial discriminator is specified by self.adversaries, followed

        You need to return two things:
        1) The first thing is the log of the predicted probabilities (rather than
        2) The second thing is the log of the predicted probabilities (rather tha

        Notice:
        For both the label predictor and the adversarial discriminator, we apply

        """
        ### YOUR CODES HERE ###
        h_relu = inputs
        for hidden in self.hiddens:
            h_relu = F.relu(hidden(h_relu))
        logprobs_y = F.log_softmax(self.softmax(h_relu), dim=1)

        adv_relu = grad_reverse(h_relu)
        for adversary in self.adversaries:
            adv_relu = F.relu(adversary(adv_relu))
        logprobs_a = F.log_softmax(self.sensitive_cls(adv_relu), dim=1)

        return logprobs_y, logprobs_a

    def inference(self, inputs):
        h_relu = inputs
        for hidden in self.hiddens:
            h_relu = F.relu(hidden(h_relu))
        # Classification probability.
        logprobs = F.log_softmax(self.softmax(h_relu), dim=1)
        return logprobs
```

Implemented Code for (c):

```
class FairNet(nn.Module):
    """
    Multi-layer perceptron with adversarial training for fairness.
    """

    def __init__(self, configs):
        super(FairNet, self).__init__()
        self.input_dim = configs["input_dim"]
        self.num_classes = configs["num_classes"]
        self.num_hidden_layers = len(configs["hidden_layers"])
        self.num_neurons = [self.input_dim] + configs["hidden_layers"]
        # Parameters of hidden, fully-connected layers, feature learning componen
        self.hiddens = nn.ModuleList([nn.Linear(self.num_neurons[i], self.num_neu
                                      for i in range(self.num_hidden_layers)])
        # Parameter of the final softmax classification layer.
        self.softmax = nn.Linear(self.num_neurons[-1], configs["num_classes"])
        # Parameter of the adversary classification layer.
        self.num_adversaries = [self.num_neurons[-1]] + configs["adversary_layers
        self.num_adversaries_layers = len(configs["adversary_layers"])
        self.adversaries = nn.ModuleList([nn.Linear(self.num_adversaries[i], self
                                          for i in range(self.num_adversaries_lay
        self.sensitive_cls = nn.Linear(self.num_adversaries[-1], 2)

    def forward(self, inputs):
        """
        The feature extractor is specified by self.hiddens.
        The label predictor is specified by self.softmax.
        The adversarial discriminator is specified by self.adversaries, followed

        You need to return two things:
        1) The first thing is the log of the predicted probabilities (rather thar
        2) The second thing is the log of the predicted probabilities (rather tha

        Notice:
        For both the label predictor and the adversarial discriminator, we apply

        """
        ### YOUR CODES HERE ###
        h_relu = inputs
        for hidden in self.hiddens:
            h_relu = F.relu(hidden(h_relu))
        logprobs_y = F.log_softmax(self.softmax(h_relu), dim=1)
        adv_relu = grad_reverse(h_relu)
        for adversary in self.adversaries:
            adv_relu = F.relu(adversary(adv_relu))
        logprobs_a = F.log_softmax(self.sensitive_cls(adv_relu), dim=1)

        return logprobs_y, logprobs_a

    def inference(self, inputs):
        h_relu = inputs
        for hidden in self.hiddens:
            h_relu = F.relu(hidden(h_relu))
        # Classification probability.
        logprobs = F.log_softmax(self.softmax(h_relu), dim=1)
        return logprobs
```