Assignment #3

Due: 11:59pm EST, March 23rd, 2023

Homework 3: Bayesian Methods and Neural Networks

Introduction

This homework is about Bayesian methods and neural networks. You may want to consider the lecture notes from Feb 14th to 23rd (weeks 4 and 5). Here's an outline of the questions:

1. You'll explore the Bayesian paradigm and compare it with the frequentist paradigm for the Beta-Binomial conjugate pair.
2. You'll derive the backpropagation algorithm for a single-hidden-layer neural network for the binary classification task.
3. You'll write some code using the PyTorch library for an image classification task.
4. You'll consider the opportunities and limitations of ML applications and learn to anticipate possible exploits of these systems.

As always, please start early and ask questions on Ed!

Please type your solutions after the corresponding problems using this LaTeX template, and start each problem on a new page.

Please submit the **writeup PDF to the Gradescope assignment 'HW3'**. Remember to assign pages for each question. **You must include your plots in your writeup PDF.** The supplemental files will only be checked in special cases, e.g. honor code issues, etc.

Please submit your **LaTeX file and code files to the Gradescope assignment 'HW3 - Supplemental'**.

Problem 1 (Connecting Bayesian and Frequentist Approaches)

In this question, we will gain practice with Bayesian modeling and compare it with the frequentist paradigm.

In class, we discussed *Normal-Normal conjugacy.* Now we will turn to *Beta-Binomial conjugacy.* This model can be visualized in the following way.

You observe a fixed number N of coin flips (either heads or tails) of which Y (a random variable) are heads. You assume that these are drawn by flipping a coin with an unknown probability θ of landing heads. That is, we choose a **Binomial likelihood** $Y \sim \text{Bin}(N, \theta)$. The PMF of this distribution is given by

$$p(Y = y) = \binom{N}{y} \theta^y (1 - \theta)^{N-y}.$$

1. **Frequentist paradigm and MLE.** The (log) likelihood is all we need for frequentist inference. Derive the MLE estimate for θ given the observations $Y = y$. That is, find $\arg\max_\theta \log p(Y = y \mid \theta)$.

2. **Beta-Binomial conjugacy.** Under the Bayesian paradigm, we must specify a prior distribution for the unknown parameter θ. We choose a **Beta prior** $\theta \sim \text{Beta}(\alpha, \beta)$. The PDF of this distribution is given by

$$p(\theta) \propto \theta^{\alpha-1}(1 - \theta)^{\beta-1}.$$

When the prior and posterior belong to the same distribution family, we call the prior-and-likelihood pair a **conjugate pair.**

(a) Derive the mean, mode, and variance of the Beta distribution. That is, for $\theta \sim \text{Beta}(\alpha, \beta)$, derive

i. $\mathbb{E}[\theta]$. See hint. [a]

ii. $\arg\max_\theta p(\theta)$ when $\alpha > 1$ and $\beta > 1$. What happens otherwise? (Consider $p(0)$ and $p(1)$.)

iii. $\text{Var}(\theta) = \mathbb{E}[\theta^2] - (\mathbb{E}[\theta])^2$.

Qualitatively speaking, what does this distribution look like for different α and β? You can either plot this yourself or see its Wikipedia page after deriving the statistics above. What does $\text{Beta}(1, 1)$ correspond to?

(b) Show that the posterior $p(\theta \mid Y = y)$ is indeed Beta and derive its parameters. This proves that a Beta prior and a Binomial likelihood form a conjugate pair; in other words, the Beta distribution is a **conjugate prior** for the Binomial distribution. See hint.[b]

[a]As an alternative to taking the integral, you may want to use *reasoning by representation.* See example 8.5.2 of the Stat 110 textbook. If you do so, please explain the derivation in your own words!

[b]For convenience in calculation: Do you need to calculate the normalizing constant? Reuse your results from the previous part.

3. **Posterior mean and mode.** Often we wish to work with just a single point estimate of the posterior. Two commonly used point estimates are the *posterior mean* and the *posterior mode* (a.k.a. the maximum a posteriori (MAP) estimate).

   (a) Discuss the advantages and disadvantages of using posterior point estimates. Which of these are relevant for our Beta-Binomial conjugate pair? Consider the case when $\alpha, \beta < 1$.

   (b) Using your results from part 2, write down

      i. the posterior mean estimate $\theta_{\text{post mean}} = \mathbb{E}[\theta \mid Y = y]$,

      ii. the posterior MAP estimate $\theta_{\text{MAP}} = \arg\max_\theta p(\theta \mid Y = y)$,

      iii. and the posterior variance $\text{Var}(\theta \mid Y = y) = \mathbb{E}[\theta^2 \mid Y = y] - (\mathbb{E}[\theta \mid Y = y])^2$.

   You shouldn't need any further derivations. That's the nice thing about conjugate priors!

4. **Prior-posterior connections.**

   (a) Explain in your own words how α and β affect the MAP estimate. How would you set α and β to reflect a prior belief that the coin is fair (i.e. shows heads and tails with equal probability)? (Be careful! See 2.a.ii.)

   (b) Now let's analyze the variances of our prior and posterior distributions. Consider the case when $\alpha = \beta$. (If you'd enjoy it, consider the general case for a better understanding.) A sentence or two for each point is fine.

      i. How does the variance of the prior relate to the variance of the posterior?

      ii. How might you use the prior variance to encode a stronger or weaker prior belief?

      iii. How does the posterior variance change as we observe more samples n?

5. **Analysis and connection to frequentism.**

   (a) Write a loss function $\ell(\theta) \in \mathbb{R}$ in terms of $\theta, y, n, \alpha, \beta$ such that minimizing ℓ is equivalent to calculating the MAP estimate, i.e. $\theta_{\text{MAP}} = \arg\min_\theta \ell(\theta)$. Your function should be a sum of:

      i. a mean-squared-error term (which should loosely resemble $(y - \hat{y})^2$)

      ii. a regularization term $g(\theta) = -a\theta + b\theta^2$ for some a, b.

   Can you interpret the regularization term?

   Hint: Work backwards from part 1 to derive the MSE term and from part 2.a.ii to get the regularization term. Watch out for the signs! For the interpretation, complete the square and then compare your expression with the prior mode you found in 2.a.ii.

   (b) What happens to both $\theta_{\text{post mean}}$ and θ_{MAP} as $n \to \infty$? Compare this to the MLE estimate. (Remember to account for the change in y.)

Solution:

1.1: Frequentist paradigm and MLE.

Deriving the MLE for a Binomial is very similar to finding the MLE for the Bernoulli (since the two are very related):

$$\begin{aligned}
logp(Y = y|\theta) &\propto ylog(\theta) + (N - y)log(1 - \theta) \\
\frac{d}{d\theta}logp(Y = y|\theta) &\propto \frac{y}{\theta} - \frac{N - y}{1 - \theta} = 0 \\
\Rightarrow \hat{\theta}_{MLE} &= \frac{y}{N}
\end{aligned}$$

TODO: do I need to check the second derivative to show that the function is convex?

1.2: Beta-binomial conjugacy.

TODO

(a(i)) From the story of the beta distribution (STAT110), we know that the parameters can be interpreted as the number of successes and the number of failures. Thus, we see that the mean (aka the average number of successes) a Beta(α, β) is $\frac{\alpha}{\alpha+\beta}$.

(a(ii)) We compute, when $\alpha > 1, \beta > 1$:

$$\begin{aligned}
\frac{d}{d\theta}p(\theta) &= (\alpha - 1)\theta^{\alpha-2}(1 - \theta)^{\beta-1} - \theta^{\alpha-1}(\beta - 1)(1 - \theta)^{\beta-2} = 0 \quad \text{by product rule} \\
&= \theta^{\alpha-2}(1 - \theta)^{\beta-2}\left((\alpha - 1)(1 - \theta) - \theta(\beta - 1)\right) \\
\Rightarrow \text{arg max}_\theta p(\theta) &= (\alpha - 1)(1 - \theta) - \theta(\beta - 1) = \alpha - 1 + \alpha - \alpha\theta - \theta\beta + \theta \\
&= \theta(-\alpha + 1 - \beta + 1) + (\alpha - 1) \\
\Rightarrow \hat{\theta}_{MLE} &= \frac{\alpha - 1}{\alpha + \beta - 2}
\end{aligned}$$

If $\alpha < 1$ or $\beta < 1$ then the distribution is convex so there isn't a unique or finite mode; the PDF goes to infinity as $\theta \to 0$ and $\theta \to 1$. TODO: add?

TODO: do I need to check the second derivative to show that the function is convex?

(a(iii)) We compute the variance as follows: TODO: why do I find variance particularly difficult? It's always the $\mathbb{E}[\theta^2]$ bit that I find difficult.

TODO: do the derivation

$$\begin{aligned}
\text{Var}(\theta) &= \mathbb{E}[\theta^2] - (\mathbb{E}[\theta])^2 \quad \text{by def var} \\
&= \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta - 1)}
\end{aligned}$$

(a cont'd) The distribution Beta$(1,1)$ = Unif$(0,1)$, because when $\alpha = \beta = 1$ the PDF becomes constant (no dependence on θ). The beta distribution is remarkably flexible depending on the setting of its two hyperparameters. TODO: add?

(b) Here we show that the posterior is a Beta$(\alpha + y, \beta + N - y)$:

$$\begin{aligned} p(\theta|Y) &\propto \theta^y (1-\theta)^{N-y} \theta^{\alpha-1}(1-\theta)^{\beta-1} \\ &= \theta^{y+\alpha-1}(1-\theta)^{\beta+N-y-1} \end{aligned}$$

Since the support for θ is $[0,1]$, we see that this matches the PDF for a Beta distribution where the update rule is $\alpha_N = \alpha + y$ and $\beta_N = \beta + N - y$.

1.3: Posterior mean and mode.

TODO

(a) Advantages: point estimates are easy to interpret, and are a single number. This can make them easy to work with, too.

Disadvantages: when we summarize a posterior distribution with a point estimate, we lose tons of information. Using a distribution instead of a single number lets us account for uncertainties and the randomness inherent to the process.

(b)

$$\begin{aligned} \theta_{\text{post mean}} = \mathbb{E}[\theta|Y=y] &= \frac{\alpha + y}{\alpha+\beta+N} \\ \theta_{\text{MAP}} = \arg\max_\theta p(\theta|Y=y) &= \frac{\alpha_N - 1}{\alpha_N + \beta_N - 2} = \frac{\alpha + y - 1}{\alpha+\beta+N-2} \\ \text{Var}(\theta|Y=y) &= \frac{\alpha_N\beta_N}{(\alpha_N+\beta_N)^2(\alpha_N+\beta_N-1)} = \frac{(\alpha+y)(\beta+N-y)}{(\alpha+\beta+N)^2(\alpha+\beta+N-1)} \end{aligned}$$

1.4: Prior-posterior connections.

TODO

(a) The larger α is, the stronger our prior belief is that 1 occurs more than 0, so our MAP moves towards 1. On the other hand, as $\beta \to \infty$, $\theta_{MAP} \to 0$. To reflect a prior belief that the coin is fair, then we can set $\alpha = \beta > 1$. TODO: why do I need to be careful of 2.a.ii?

(b.i) The variance of the prior is larger than the variance of the posterior, because we are adding data. TODO: how to show computationally?

(b.ii) To encode a weaker belief, we can choose parameters of α, β to make the prior variance larger. This basically shows that we are less certain in our belief.

(b.iii) As $N \to \infty$, the posterior variance goes to 0 because we have a N^3 term in the bottom and a N in the numerator. This makes sense, because if we had infinite data we expect our variance to shrink to 0.

1.5: Analysis and connection to frequentism.

TODO: OH.

(a) We write a loss function s.t. minimizing ℓ is equivalent to calculating the MAP estimate. Specifically, we write:

$$\ell(\theta|y, n, \alpha, \beta) = TODO.$$

(b) As $N \to \infty$, the posterior mean and the MAP both go to the MLE, which is $\frac{y}{N}$. This occurs because the constant terms (e.g. α, β) will drop out. This makes sense, because if we have infinite data then the prior shouldn't affect the outcome at all.

Problem 2 (Neural Networks)

In this problem, we will take a closer look at how gradients are calculated for backprop with a simple multi-layer perceptron (MLP). The MLP will consist of a first fully connected layer with a sigmoid activation, followed by a one-dimensional, second fully connected layer with a sigmoid activation to get a prediction for a binary classification problem. We use non-linear activation functions as the composition of linear functions is linear. Assume bias has not been merged. Let:

- $\mathbf{W}_1$ be the weights of the first layer, $\mathbf{b}_1$ be the bias of the first layer.
- $\mathbf{W}_2$ be the weights of the second layer, $\mathbf{b}_2$ be the bias of the second layer.

The described architecture can be written mathematically as:

$$\hat{y} = \sigma(\mathbf{W}_2 \left[\sigma\left(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1\right)\right] + \mathbf{b}_2)$$

where $\hat{y}$ is a scalar output of the net when passing in the single datapoint $\mathbf{x}$ (represented as a column vector), the additions are element wise additions, and the sigmoid is an element wise sigmoid.

1. Let:
    - N be the number of datapoints we have
    - M be the dimensionality of the data
    - H be the size of the hidden dimension of the first layer. Here, hidden dimension is used to describe the dimension of the resulting value after going through the layer. Based on the problem description, the hidden dimension of the second layer should be 1.

    Write out the dimensionality of each of the parameters, and of the intermediate variables:

    $$\begin{aligned} \mathbf{a}_1 &= \mathbf{W}_1\mathbf{x} + \mathbf{b}_1, & \mathbf{z}_1 &= \sigma(\mathbf{a}_1) \\ a_2 &= \mathbf{W}_2\mathbf{z}_1 + \mathbf{b}_2, & \hat{y} = z_2 &= \sigma(a_2) \end{aligned}$$

    and make sure they work with the mathematical operations described above. Examining shapes is one of the key ways to debug your code, and can be done using .shape after any numpy array.

2. We will derive the gradients for each of the parameters, which can then be used along with gradient descent to find weights that improve our model's performance. For this question, assume there is only one datapoint $\mathbf{x}$, and that our loss is $L = -(y\log(\hat{y}) + (1-y)\log(1-\hat{y}))$. For all questions, the chain rule will be useful.
    (a) Find $\frac{\partial L}{\partial b_2}$.
    (b) Find $\frac{\partial L}{\partial W_2^h}$, where W_2^h represents the hth element of $\mathbf{W}_2$.
    (c) Find $\frac{\partial L}{\partial b_1^h}$, where b_1^h represents the hth element of $\mathbf{b}_1$. (*Hint: Note that only the hth element of $\mathbf{a}_1$ and $\mathbf{z}_1$ depend on b_1^h - this should help you with how to use the chain rule.)
    (d) Find $\frac{\partial L}{\partial W_1^{h,m}}$, where $W_1^{h,m}$ represents the element in row h, column m in $\mathbf{W}_1$.

Problem 2 (cont.)

3. We now explore an example of forward-mode auto-differentiation. Consider the following equation:

$$f(x_1, x_2) = \ln(\sin(x_1)) + x_1 \exp\{x_2\}$$

This equation can be split up using intermediate variables $v_1, \ldots, v_7$ as follows:

$$\begin{aligned} v_1 &= x_1 \\ v_2 &= \sin(v_1) \\ v_3 &= \ln(v_2) \\ v_4 &= x_2 \\ v_5 &= \exp\{v_4\} \\ v_6 &= v_1 v_5 \\ v_7 &= v_3 + v_6 \\ f(x_1, x_2) &= v_7 \end{aligned}$$

Splitting up the equation like this is very similar to what an auto-differentiation library would do. From these equations we can construct a *computational graph* where each node of the graph corresponds to an input, an intermediate variable, or the output.

   (a) Let $x_1 = \frac{\pi}{6}$ and $x_2 = 1$. Calculate the values of all the intermediate variables $v_1, \ldots v_7$ and $f(x_1, x_2)$.

   (b) Calculate the derivative of all of the intermediate variables $v_1, \ldots, v_7$ and f with respect to x_1 evaluated at $x_1 = \frac{\pi}{6}$ and $x_2 = 1$.

4. **Extra Credit (Hard):** Consider two neural networks f_1 and f_2 for binary classification. They each take in inputs $x \in \mathbb{R}^2$ and output a prediction $\hat{y} \in [0, 1]$. f_1 consists of a single hidden layer with 4 nodes, each with a ReLU activation function. These nodes are connected to a single sigmoid output node. Thus f_1 has the following form:

$$f_1(x) = \sigma\left(W_2[ReLU(W_1 x + b_1)] + b_2\right)$$

f_2 consists of 2 hidden layers, each with 2 ReLU activated nodes. Just as in f_1, the nodes of the final layer are connected to a single sigmoid output node. Thus f_2 has the following form:

$$f_2(x) = \sigma(W_3[ReLU(W_2[ReLU(W_1 x + b_1)] + b_2)] + b_3)$$

We leave finding the shapes of the weight and bias vectors up to you, noting that by convention x should be considered a column vector with 2 elements.

Draw a classification boundary that f_2 can express but f_1 cannot and argue why f_2 can express the boundary but f_1 cannot.

Solution:

2.1:

Assuming the normal convention where $\vec{x}$ has shape Mx1 (column vector), then we have:

- Shape of $\vec{a}_1, \vec{b}_1, \vec{z}_1$: Hx1
- Shape of $\mathbf{W_1}$: HxM
- Shape of $\mathbf{W_2}$: 1xH
- Shape of $a_2, b_2, z_2, \hat{y}$: 1x1 (scalar)

2.2:

(a)

$$\begin{aligned}
\frac{\partial L}{\partial b_2} &= \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_2} \cdot \frac{\partial a_2}{\partial b_2} \\
&= \left(-\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}} \right) \cdot 1 \cdot \sigma(a_2)(1-\sigma(a_2)) \cdot 1 \\
&= \left(-\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}} \right) \cdot \sigma(a_2)(1-\sigma(a_2))
\end{aligned}$$

(b)

$$\begin{aligned}
\frac{\partial L}{\partial W_2^h} &= \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_2} \cdot \frac{\partial a_2}{\partial W_2^h} \\
&= \left(-\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}} \right) \cdot \sigma(a_2)(1-\sigma(a_2)) \cdot \frac{\partial a_2}{\partial W_2^h} \quad \text{from part (a)} \\
&= \left(-\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}} \right) \cdot \sigma(a_2)(1-\sigma(a_2)) \cdot z_1^h \quad \text{bc only the } h\text{th comp of } z_1 \text{ contributes}
\end{aligned}$$

(c)

TODO: ask in OH/EdStem about things like the $\frac{\partial a_2}{\partial z_1^h}$ terms. How should we write things like this out?

$$\begin{aligned}\frac{\partial L}{\partial W_2^h} &= \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_1^h} \cdot \frac{\partial z_1^h}{\partial a_1^h} \cdot \frac{\partial a_1^h}{\partial b_1^h} \\ &= \left(-\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right) \cdot \sigma(a_2)(1-\sigma(a_2)) \cdot \frac{\partial a_2}{\partial z_1^h} \cdot \frac{\partial z_1^h}{\partial a_1^h} \cdot \frac{\partial a_1^h}{\partial b_1^h} \quad \text{from part (a)} \\ &= \left(-\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right) \cdot \sigma(a_2)(1-\sigma(a_2)) \cdot \left(W_2^T\right)^h \cdot \sigma(a_1^h)(1-\sigma(a_1^h)) \cdot 1 \\ &= \left(-\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right) \cdot \sigma(a_2)(1-\sigma(a_2)) \cdot \left(W_2^T\right)^h \cdot \sigma(a_1^h)(1-\sigma(a_1^h))\end{aligned}$$

where anything c^h means the h-th element of the vector c.

(d)

$$\begin{aligned}\frac{\partial L}{\partial W_1^{h,m}} &= \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_1^h} \cdot \frac{\partial z_1^h}{\partial a_1^h} \cdot \frac{\partial a_1^h}{\partial W_1^{h,m}} \\ &= \left(-\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right) \cdot \sigma(a_2)(1-\sigma(a_2)) \cdot \left(W_2^T\right)^h \cdot \sigma(a_1^h)(1-\sigma(a_1^h)) \cdot \frac{\partial a_1^h}{\partial W_1^{h,m}} \quad \text{from part of (c)'s calculations} \\ &= \left(-\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right) \cdot \sigma(a_2)(1-\sigma(a_2)) \cdot \left(W_2^T\right)^h \cdot \sigma(a_1^h)(1-\sigma(a_1^h)) \cdot x^m\end{aligned}$$

where anything c^h means the h-th element of the vector c and x^m is the m-th element of the x vector.

2.3: Forward-mode auto-differentiation example

TODO **(a)** Letting $x_1 = \pi/6, \quad x_2 = 1$, we calculate the intermediate values:

$$\begin{aligned} v_1 &= \frac{\pi}{6} \\ v_2 &= \sin(\pi/6) = 0.5 \\ v_3 &= \ln(1/2) = -\ln(2) \approx -0.693 \\ v_4 &= 1 \\ v_5 &= e^1 = e \\ v_6 &= x1 \cdot e = \frac{\pi}{6} e \approx 1.423 \\ v_7 &= v_3 + v_6 = -\ln(2) + \frac{\pi}{6} e \approx 0.730 \end{aligned}$$

(b) We calculate all the derivatives of intermediate variables evaluated at $x_1 = \pi/6, \quad x_2 = 1$:

TODO: OH/Edstem. Ask: do I need to use partial derivatives for v_6 and v_7?

$$
\begin{aligned}
\frac{dv_1}{dx_1} &= 1 & &= 1 \\
\frac{dv_2}{dv_1} &= \cos(v_1) & &= \cos\left(\frac{\pi}{6}\right) = \frac{\sqrt{3}}{2} \approx 0.866 \\
\frac{dv_3}{dv_2} &= \frac{1}{v_2} & &= 2 \\
\frac{dv_4}{dx_2} &= 1 & &= 1 \\
\frac{dv_5}{dv_4} &= \exp\{v_4\} & &= 1 \\
\frac{\partial v_6}{\partial v_1} &= v_5 & &= e \\
\frac{\partial v_6}{\partial v_5} &= v_1 & &= \frac{\pi}{6} \\
\frac{\partial v_7}{\partial v_3} &= 1 & &= 1 \\
\frac{\partial v_7}{\partial v_6} &= 1 & &= 1 \\
\frac{df(x_1, x_2)}{dv_7} &= 1 & &= 1
\end{aligned}
$$

2.4: Extra credit.

TODO

Problem 3 (Modern Deep Learning Tools: PyTorch)

In this problem, you will learn how to use PyTorch. This machine learning library is massively popular and used heavily throughout industry and research.

1. In `T3_P3.ipynb` you will implement an MLP for image classification from scratch. Paste your code solutions below and include a final graph of your training progress. Also submit your completed `T3_P3.ipynb` file.
2. Discuss what trends you see with your plot (train/test loss and train/test accuracy).

Out of Distribution (OOD) Analysis: Now, let's evaluate the usefulness of the predictive uncertainties of our model for test data that are dissimilar to our training data. These test data points are called out of distribution (OOD) points. Just as in Homework 2, we want the predictive uncertainties from our models to help us distinguish in-distribution test data (test data that are similar to data on which we trained our model) and OOD test data. Again, in many safety-critical applications of ML, we want human-experts to override model decisions if the model is operating on extremely unfamiliar data.

3. Report both the in and out distribution test accuracies of your model. In a couple of sentences, discuss what you notice about these accuracies.

You will recieve no points for code not included below.

You will recieve no points for code using built-in APIs from the `torch.nn` library.

Solution:

3.1: T

ODO

3.2: T

ODO

3.3: T

ODO

Plot:

Code:

```
n_inputs = 'not implemented'
n_hiddens = 'not implemented'
n_outputs = 'not implemented'

W1 = 'not implemented'
b1 = 'not implemented'
W2 = 'not implemented'
b2 = 'not implemented'
```

```python
def relu(x):
    'not implemented'



def softmax(x):
    'not implemented'



def net(X):
  'not implemented'



def cross_entropy(y_hat, y):
  'not implemented'



def sgd(params, lr=0.1):
  'not implemented'



def train(net, params, train_iter, loss_func=cross_entropy, updater=sgd):
  'not implemented'
```

Problem 4 (Impact Question: Testing security of neural networks deployed in autonomous vehicles and suggesting policy recommendations (9 points))

The learning goal of the impact questions of this homework is three-fold:

1. Get trained in adversarial thinking to be able to anticipate risks and possible exploits when designing Machine Learning applications
2. Understand opportunities and limitations to safety and security of Machine Learning applications
3. Learn to put yourself in the shoes of policymakers who are in charge of ensuring safety of real-world Machine Learning applications.

Prompt: You are the Director of Machine Learning of the US Department of Transportation (a federal US government agency).

The Secretary of the US Department of Transportation declares security of Machine Learning applications deployed in autonomous vehicles as one of the priorities of the agency. You are tasked to assess the security of Machine Learning systems deployed in autonomous vehicles and develop policy recommendations for the US Department of Transportation.

Context: Tesla employs Neural Networks for perception and control tasks: For example semantic segmentation, object detection and monocular depth estimation is performed by Neural Networks on images captured with the car's camera system to identify road signs, traffic lights, pedestrians, cars or other traffic related individuals, vehicles, and objects. The full build autopilot consists of more than 48 networks which must identify high-risk scenarios and provide robust predictions to ensure safety.

Moreover, beyond cameras Tesla also uses additional sensor systems such as LiDAR or ultrasonic sensors. Yet, Tesla's engineering and design approaches are still iterated and their software gets updated.

Link to a demo video: `https://tesla-cdn.thron.com/static/NGSLYL_network_XZCUMR.mp4`

Please answer the questions below by using concise language (350 - 700 words in total). Bullet points are appropriate.

Questions:

1. **Adversarial thinking:** List and explain 3 options how you could attack the Neural Network deployed in a self-driving car to make it crash. In particular, explain the impact of the attack on the statistical properties of input data and predictions of the Neural Network. (3 points)
   (a) **Hardware adversarial attack:** List and explain one attack which targets the hardware system of an autonomous vehicle or its physical surround.
   (b) **Software adversarial attack:** List and explain one attack which targets the software system of an autonomous vehicle.
   (c) **Social engineering:** List and explain one attack which relies on social engineering to make an autonomous vehicle crash.
2. **Safeguards:** For each of the 3 attack options that you listed above, suggest and explain one possible solution that could safeguard the Neural Network deployed in Tesla's autopilot. (3 points)
3. **Policy recommendation:** The Secretary of the US Department of Transportation asks you to develop one policy recommendation on how to ensure sufficient security of deployed neural networks in autonomous vehicles.
   (a) **Recommendation:** List and explain one policy recommendation that the US Department of Transportation should implement. (1 point)
   (b) **Benefit:** List and explain one benefit of your chosen recommendation. (1 point)
   (c) **Drawback:** List and explain one drawback of your chosen recommendation. (1 point)

Solution:

4.1: Adversarial thinking.

TODO

(a) An example of a hardware adversarial attack would be spraying paint or otherwise covering the camera of the self-driving car, or by placing tape over stop signs to make them look rectangular and to remove the word "STOP." Making the camera unable to function (especially if done after the car was already in motion) would make the car unable to follow the rules of the road and avoid objects like cars or people. Physically making the stop sign unrecognizable would also likely lead to a crash from the autonomous vehicle hitting another car at a 4-way stop, because the rectangular stop sign would be pretty OOD compared to the input data used to train the model.

(b) An example of a software adversarial attack (assuming we nefariously gain access to the model prior to deployment) would be to (TODO: what is this?)

Another idea would be painting new lane markings that would lead the autopiloted car into oncoming traffic.

(c) An example of a social engineering attack could be if images/video used to train the neural network were purposefully polluted with mislabeled data or incorrect "appropriate actions" (e.g. making red mean go and green mean stop at a stoplight).

4.2: Safeguards.

TODO

To protect against someone physically disabling the camera, Tesla could make the autopilot feature automatically turn off if any of its sensors are below some threshold of 'optimum' (e.g. 95%). This would allow for specs of dirt but not for disabling issues.

To protect against issues like weirdly made-up stop signs or paint on the road, the NN needs to be trained to recognize odd/abnormal markings. This would involve augmenting the training dataset to include such issues as construction, adversarial actors changing road signs, etc.

To protect against mislabeled or otherwise inappropriate data, the engineers need to actually examine the training dataset images. This is especially important for classification or ID tasks that involve high-risk or high-impact decisions (e.g. it's less important to know the meaning of one obscure sign than it is to identify when a human or car is within striking distance).

4.3: Policy recommendation.

TODO

(a)

(b)

(c)

Name

Collaborators and Resources

Whom did you work with, and did you use any resources beyond cs181-textbook and your notes?

Calibration

Approximately how long did this homework take you to complete (in hours)?