# CS 446 / ECE 449 — Homework 3

*your NetID here*

Version 1.1

**Instructions.**

- Homework is due **Thursday, March 3, at noon CST**; no late homework accepted.
- Everyone must submit individually on Gradescope under `hw3` and `hw3code`. Problem parts are marked with [`hw3`] and [`hw3code`] to indicate where they are handed in.
- The "written" submission at `hw3` **must be typed**, and submitted in any format Gradescope accepts (to be safe, submit a PDF). You may use LaTeX, Markdown, Google Docs, MS Word, whatever you like; but it must be typed!
- When submitting at `hw3`, Gradescope will ask you to select pages for each problem; please do this precisely!
- Please make sure your NetID is clear and large on the first page of the homework.
- Your solution **must** be written in your own words. Please see the course webpage for full academic integrity information. Briefly, you may have high-level discussions with at most 3 classmates, whose NetIDs you should place on the first page of your solutions, and you should cite any external reference you use; despite all this, your solution must be written in your own words.
- We reserve the right to reduce the auto-graded score for `hw3code` if we detect funny business (e.g., your solution lacks any algorithm and hard-codes answers you obtained from someone else, or simply via trial-and-error with the autograder).
- Coding problems come with suggested "library routines"; we include these to reduce your time fishing around APIs, but you are free to use other APIs.
- When submitting to `hw3code`, upload `hw3.py`. Don't upload a zip file or additional files.

**Version history.**

1.0. Initial version.

1.1. Clarify to use SGD in Problem 1(c) and Problem 1(d).

## 1. ResNet.

In this problem, you will implement a simplified ResNet. You do not need to change arguments which are not mentioned here (but you of course could try and see what happens).

(a) [`hw3code`] Implement a class `Block`, which is a building block of ResNet. It is described in Figure 2 of He et al. (2016), but also as follows.

The input to `Block` is of shape $(N, C, H, W)$, where $N$ denotes the batch size, $C$ denotes the number of channels, and $H$ and $W$ are the height and width of each channel. For each data example $\boldsymbol{x}$ with shape $(C, H, W)$, the output of `block` is

$$\texttt{Block}(\boldsymbol{x}) = \sigma_r\left(\boldsymbol{x} + f(\boldsymbol{x})\right),$$

where $\sigma_r$ denotes the ReLU activation, and $f(\boldsymbol{x})$ also has shape $(C, H, W)$ and thus can be added to $\boldsymbol{x}$. In detail, $f$ contains the following layers.

i. A `Conv2d` with $C$ input channels, $C$ output channels, kernel size 3, stride 1, padding 1, and no bias term.
ii. A `BatchNorm2d` with $C$ features.
iii. A ReLU layer.
iv. Another `Conv2d` with the same arguments as i above.
v. Another `BatchNorm2d` with $C$ features.

Because $3 \times 3$ kernels and padding 1 are used, the convolutional layers do not change the shape of each channel. Moreover, the number of channels are also kept unchanged. Therefore $f(\boldsymbol{x})$ does have the same shape as $\boldsymbol{x}$.

Additional instructions are given in doscstrings in `hw3.py`.

**Library routines:** `torch.nn.Conv2d and torch.nn.BatchNorm2d.`

**Remark:** Use `bias=False` for the `Conv2d` layers.

(b) [`hw3code`] Implement a (shallow) `ResNet` consists of the following parts:

i. A `Conv2d` with 1 input channel, $C$ output channels, kernel size 3, stride 2, padding 1, and no bias term.
ii. A `BatchNorm2d` with $C$ features.
iii. A ReLU layer.
iv. A `MaxPool2d` with kernel size 2.
v. A `Block` with $C$ channels.
vi. An `AdaptiveAvgPool2d` which for each channel takes the average of all elements.
vii. A `Linear` with $C$ inputs and 10 outputs.

Additional instructions are given in doscstrings in `hw3.py`.

**Library routines:** `torch.nn.Conv2d, torch.nn.BatchNorm2d, torch.nn.MaxPool2D, torch.nn.AdaptiveAvgPool2d and torch.nn.Linear.`

**Remark:** Use `bias=False` for the `Conv2d` layer.

(c) [`hw3`] Train your `ResNet` implemented in (b) with different choices $C \in \{1, 2, 4\}$ on digits data and draw the training error vs the test error curves. To make your life easier, we provide you with the starter code to load the digits data and draw the figures with different choices for $C$. Therefore, you only need to write the code to train your `ResNet` in function `plot_resnet_loss_1()`. Train your algorithms for 4000 epochs using SGD with mini batch size 128 and step size 0.1. See the docstrings in `hw3.py` for more details. Include the resulting plot in your written handin.

For full credit, in addition to including the six train and test curves, include at least one complete sentence describing how the train and test error (and in particular their gap) change with $C$, which itself corresponds to a notion of model complexity as discussed in lecture.

**Library routines:** `torch.nn.CrossEntropyLoss, torch.autograd.backward, torch.no_grad, torch.optim.Optimizer.zero_grad, torch.autograd.grad, torch.nn.Module.parameters.`

(d) [hw3] Train your ResNet implemented in (b) with $C = 64$ on digits data and draw the training error vs the test error curve. To make your life easier, we provide you with the starter code to load the digits data and draw the figures with $C = 64$. Therefore, you only need to write the code to train your ResNet in function plot_resnet_loss_2(). Train your algorithms for 4000 epochs using SGD with mini batch size 128 and step size 0.1. See the docstrings in hw3.py for more details. Notice that you can use the same implementation of training part in part (c). Include the resulting plot in your written handin.

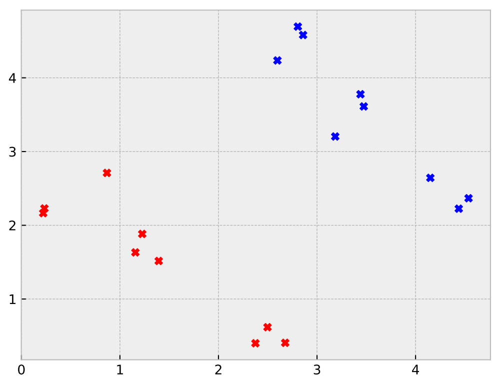
For full credit, additionally include at least one complete sentence comparing the train and test error with those in part (c).

**Library routines:** torch.nn.CrossEntropyLoss, torch.autograd.backward, torch.no_grad, torch.optim.Optimizer.zero_grad, torch.autograd.grad, torch.nn.Module.parameters.

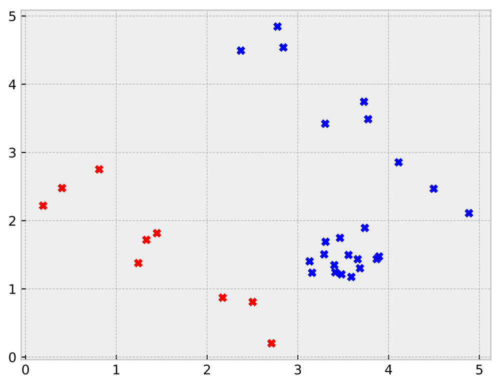
**Solution.**

## 2. Decision Trees and Nearest Neighbor.

Consider the training and testing data sets as given in Figures 1a and 1b for the sub-parts (a)-(c). For the sub-parts (d)-(f), refer to the training and testing data sets as given in Figures 2a and 2b. For problems related to decision trees, either draw the decision trees unambiguously on the figure (e.g., either with drawing software, or by taking a picture of a hand drawing) and include the modified diagrams in your submission, or use unambiguous pseudocode to specify the decision trees.



(a) Fig 1a: Training data set 1.

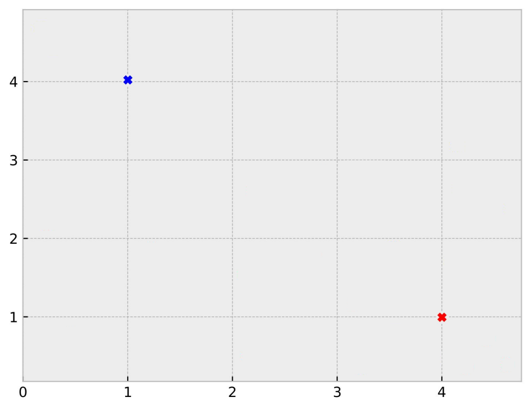


(b) Fig 1b: Testing data set 1.

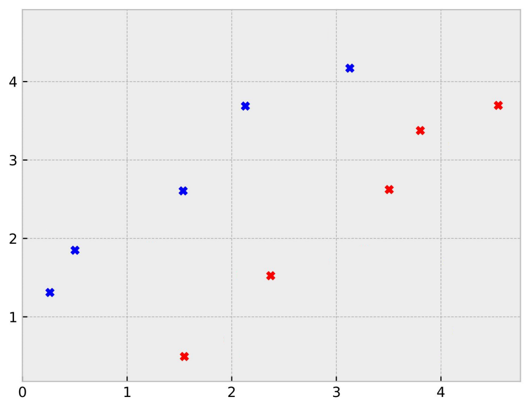
(a) [hw3] Define in pseudocode or draw (as above) a decision tree of depth one with integral and axis-aligned decision boundaries which achieves error at most $\frac{1}{6}$ on training data set 1 (Figure 1a). **Note:** "integral and axis-aligned" means the decision tree consists of splitting rules of the form $[x_1 \geq 5]$, $[x_2 < 3]$, and so on.

(b) [hw3] Define in pseudocode or draw (as above) a decision tree (of any depth) with integral and axis-aligned decision boundaries which achieves zero error on training data set 1 (Figure 1a).

(c) [hw3] Define in pseudocode or draw (as above) a decision tree (of any depth) with integral and axis-aligned decision boundaries which achieves zero error on training data set 1 (Figure 1a) but has error at least $\frac{1}{4}$ on testing data set 1 (Figure 1b).



(a) Fig 2a: Training data set 2.



(b) Fig 2b: Testing data set 2.

(d) [hw3] Define in pseudocode or draw (as above) a decision tree with integral and axis-aligned decision boundaries with at most two splits, which achieves zero error on training data set 2 (Figure 2a) and calculate its error on testing data set 2 (Figure 2b).

(e) [hw3] Construct and draw a 1-nearest-neighbor classifier using training data set 2 (Figure 2a). Then copy over that classifier to the corresponding testing data set 2 (Figure 2b). As discussed in class, the training error will be zero; what is the test error on testing data set 2 (Figure 2b)? For full points, include both figures and at least one complete sentence stating the test error.

(f) [hw3] Comparing the result of the decision tree from part (d) and the result of the 1-nn classifier from part (e). Which one has a smaller training error? Which one has a smaller test error? Which algorithm is more suitable here? (In case that both algorithms have the same error, state that they have the same error.)

**Solution.**

## 3. Robustness of the Majority Vote Classifier.

The purpose of this problem is to further investigate the behavior of the majority vote classifier (*see slides 5-7 of lecture 12*) using Hoeffding's inequality (*see slide 7 of lecture 12, and for more background, slide 14 of lecture 13*). Simplified versions of Hoeffding's inequality are as follows.

**Theorem 1.** *Given independent random variables $(Z_1, \dots, Z_k)$ with $Z_i \in [0,1]$,*

$$\Pr\left[\sum_{i=1}^{k} Z_i \geq \sum_{i=1}^{k} \mathbb{E}[Z_i] + k\epsilon\right] \leq \exp\left(-2k\epsilon^2\right), \tag{1}$$

*and*

$$\Pr\left[\sum_{i=1}^{k} Z_i \leq \sum_{i=1}^{k} \mathbb{E}[Z_i] - k\epsilon\right] \leq \exp\left(-2k\epsilon^2\right). \tag{2}$$

In this problem we have an odd number $n$ of classifiers $\{f_1, \dots, f_n\}$ and only consider their behavior on a fixed data example $(\boldsymbol{x}, y)$; by classifier we mean $f_i(\boldsymbol{x}) \in \{\pm 1\}$. Define the majority vote classifer $\textsc{Maj}$ as

$$\textsc{Maj}(\boldsymbol{x}; f_1, \dots, f_n) := 2 \cdot \mathbb{1}\left[\sum_{i=1}^{n} f_i(\boldsymbol{x}) \geq 0\right] - 1 = \begin{cases} +1 & \sum_{i=1}^{n} f_i(\boldsymbol{x}) > 0, \\ -1 & \sum_{i=1}^{n} f_i(\boldsymbol{x}) < 0, \end{cases}$$

where we will not need to worry about ties since $n$ is odd.

To demonstrate the utility of Theorem 1 in analyzing $\textsc{Maj}$, suppose that $\Pr[f_i(\boldsymbol{x}) = y] = p > 1/2$ independently for each $i$. Then, by defining a random variable $Z_i := \mathbb{1}[f_i(\boldsymbol{x}) \neq y]$ and noting $\mathbb{E}[Z_i] = 1 - p$,

$$\begin{aligned} \Pr[\textsc{Maj}(\boldsymbol{x}; f_1, \dots, f_n) \neq y] &= \Pr\left[\sum_{i=1}^{n} \mathbb{1}[f_i(\boldsymbol{x}) \neq y] \geq \frac{n}{2}\right] \\ &= \Pr\left[\sum_{i=1}^{n} Z_i \geq n(1-p) - \frac{n}{2} + np\right] \\ &= \Pr\left[\sum_{i=1}^{n} Z_i \geq n\mathbb{E}[Z_1] + n(p - 1/2)\right] \\ &\leq \exp\left(-2n(p-1/2)^2\right). \end{aligned}$$

The purpose of this problem is to study the behavior of $\textsc{Maj}(\boldsymbol{x})$ when not all of the classifiers $\{f_1, \dots, f_n\}$ are independent.

(a) [hw3] Assume $n$ is divisible by 7 and $5n/7$ is odd, and that of the $n$ classifiers $\{f_1, \dots, f_n\}$, now only the first $5n/7$ of them (i.e., $\{f_1, \dots, f_{5n/7}\}$) have independent errors on $\boldsymbol{x}$. Specifically, $\Pr[f_i(\boldsymbol{x}) = y] = p := 4/5$ for classifiers $\{f_1, \dots, f_{5n/7}\}$. By contrast, we make no assumption on the other $2n/7$ classifiers (i.e., $\{f_{5n/7+1}, \dots, f_n\}$) and their errors. Now use Hoeffding's inequality to show that

$$\Pr\left[\sum_{i=1}^{5n/7} \mathbb{1}[f_i(\boldsymbol{x}) \neq y] \geq \frac{3n}{14}\right] \leq \exp\left(-\frac{n}{70}\right).$$

(b) [hw3] Continuing from (a), further show that the majority vote classifier over all $n$ classifiers is still good, specifically showing

$$\Pr\left[\textsc{Maj}(\boldsymbol{x}; f_1, \dots, f_n) \neq y\right] \leq \exp\left(-\frac{n}{70}\right).$$

**For full points:** You need to derive the inequality $\Pr\left[\textsc{Maj}(\boldsymbol{x}; f_1, \ldots, f_n) \neq y\right] \leq \exp(-n/70)$ rigorously for ANY possible behavior of the $\frac{2n}{7}$ arbitrary classifiers.

(c) [hw3] Is the probability of correctly classifying $\boldsymbol{x}$ reasonably good in part (b) for large $n$? Do you have any interesting observations? Any answer which contains at least one complete sentence will receive full credit.

(d) [hw3] Now suppose that $n$ is divisible by 5 and $3n/5$ is odd, but now only first $3n/5$ of the classifiers (i.e., $\{f_1, \ldots, f_{3n/5}\}$) have independent errors, and are correct with probability $\Pr[f_i(\boldsymbol{x}) = y] = p := 2/3$. Use Hoeffding's inequality to show that

$$\Pr\left[\sum_{i=1}^{3n/5} \mathbb{1}[f_i(\boldsymbol{x}) \neq y] \leq \frac{n}{10}\right] \leq \exp\left(-\frac{n}{30}\right).$$

(e) [hw3] Continuing from (d), describe malicious behavior for the remaining $2n/5$ classifiers so that

$$\Pr\left[\textsc{Maj}(\boldsymbol{x}; f_1, \ldots, f_n) = y\right] \leq \exp\left(-\frac{n}{30}\right).$$

**For full points:** Describe the malicious behavior of the arbitrary classifiers AND derive the inequality $\Pr\left[\textsc{Maj}(\boldsymbol{x}; f_1, \ldots, f_n) = y\right] \leq \exp(-n/30)$.

(f) [hw3] Comparing the results from part (b) and part (e), do you have any observation? Any answer which contains at least one complete sentence will receive full credit.

**Solution.**

## References

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.