Homework 3

Manyara Bonface Baraka - mbaraka

March 29, 2025

1 Dimensionality of k-Nearest Neighbors (k-NN)

- Curse of Dimensionality: Is the phenomenon that is experienced when the number of dimensions or features in a dataset increases, and the volume of the space increases so fast that the available data becomes sparse.

a) X is uniformly distributed on [0,1]. Predict the response at a test observation x using the trainng observation that are within 10% of the range of X closest to x.

Solution

Defining interval for different cases.

- If x is not too close to the edges that is $x \in [0.05, 0.95]$ then the interval is $[x - 0.05, x + 0.05]$
- From the example x = 0.6, then the interval is [0.55, 0.65]

$$0.65 - 0.55 = 0.1$$

- If $x \leq 0.05$ for instance 0.02, then the interval is $[0, 0.1]$

$$0.1 - 0 = 0.1$$

- if $x \geq 0.95$ for instance 0.98, then the interval is $[0.9, 1]$

$$1 - 0.9 = 0.1$$

Therefore, since the range of X is 1: **the interval is 0.1**

b) When d = 2, we have X_1 and X_2 that are uniformly distributed on [0,1]. Predict the response at a test observation x using the training observation that is within 10% of the range of X closest to x.

- Since the range for both the features is 0.1, this means:

$$\text{Area of selection } = 0.1 \times 0.1 = 0.01$$

- Since they are uniformly distributed:

$$1 \times 1 = 1$$

- Since the training points are uniformly distributed over the square therefore the available observation is:

$$\frac{\text{Area of selection}}{\text{Total area}} = \frac{0.01}{1} = 0.01 \quad \text{or } 1\%$$

c) When d = 100, we have $X_1, X_2, \ldots, X_{100}$ that are uniformly distributed on [0,1]. Each feature ranges in value from 0 to 1. Predict the response at a test observation x using the training observation that is within 10% of the range of X closest to x. What fraction of the available observations will we use to make the prediction?

- In d = 100, with the length of 0.1, the volume of the hypercube is:

$$0.1^{100} = 10^{-100}$$

d) Why does KNN struggle in high dimensions?

- The key observation from parts a-c is that:
  - **For *d = 1:*** The fraction of the available observations is 10%, this is because the training points are close to the test points.
  - **For *d = 2:*** The fraction of the available observations is 1%, we have fewer "nearby" points to rely on compared to d = 1.
  - **For *d = 100:*** The fraction of the available observations is 0.1^{100}, this is a very small number. Almost no training points are close to the test point.
- The reason this happens is due to the curse of Dimensionality:
  - Neighborhood becomes sparser as the number of dimensions increases.
  - The distance measures become less meaningful with the increase of dimension this is because in high dimensions the distance between any two points becomes almost the same.
  - Since KNN relies on Neighborhood if there are not enough nearby training points, the prediction is unreliable.

e) We wish to predict test observation by creating a d-Dimensionality hypercube centered around the test observation that contains an average of 10% of the training observation For d = 1, 2, 100, what is the length of each side of the hypercube? How does the answer change as d increases, and what does it imply for the accuracy of the k-nearest neighbors when d is large?

- If we need to contain 10% of all observations the volume should be 0.1
- For d-Dimensionalityal hypercube:

$$\text{volume} = (\text{Side length})^d = 0.1$$

  let the side length be s:

$$s^d = 0.1$$

$$s = 0.1^{1/d}$$

- Finding the side length for:
  - **For d = 1:** $s = 0.1^{1/1} = 0.1$
  - **For d = 2:** $s = 0.1^{1/2} = 0.316$
  - **For d = 100:** $s = 0.1^{1/100} = 0.977$
- As d increases the side length approaches 1, this indicates that the hypercube needs to grow in size to include 10% of the training data.
- For large d, the hypercube will be very large and will contain almost all the training data. This implies that the KNN will be less reliable as the number of dimensions increases.
- The accuracy of the KNN will decrease as the number of dimensions increases.

2 Decision Trees

- Entropy is use to measure the impurity of a node in a decision tree.
- It helps us in to measure the uncertainty or randomness in a dataset. We use entropy for:
  - Decide whiche feature to split (higher reduction in entropy its a better split)
  - Evaluate how mixed/uncertain the dataset is (lower entropy leads to pure groups)
- Entropy is defined as:

$$H(S) = -\sum_{i=1}^{c} p_i \log_2 p_i f$$

  where p_i is the probability of class i in the dataset.
- High entropy means the dataset is mixed, while low entropy means the dataset is pure.
- Information Gain is the measure of the reduction in entropy after the dataset is split on an attribute.
- Information Gain is defined as:

IG = Entropy(before split) - Entropy(after split)

a) What is the entropy of Phone Usage? (Calculate the entropy in log_2 base and round to 4 decimal places)

Solution

- Count occurances:
  - Low: 7
  - Medium: 5
  - High: 3
  - Total Count: 15
- Calculate the Entropy of phone Usage:

  H(Phone Usage) = -(p_low * log2(p_low) + p_medium * log2(p_medium) + p_high * log2(p_high))

  where:
  - $p_{low} = \frac{7}{15}$
  - $p_{medium} = \frac{5}{15}$

- $p_{high} = \frac{3}{15}$

Substute:

$$= -(\frac{7}{15}\log_2(\frac{7}{15}) + \frac{5}{15}\log_2(\frac{5}{15}) + \frac{3}{15}\log_2(\frac{3}{15}))$$

$$\approx 1.5058$$

b) Find the information gain IG from each features (relationship status, age, eductaion levarianceel and income). Which features should be choosen as the root of th tree? Show calculations for the IG and explain the choise in a sentence. (Calculate entropy in log_2 base and round to 4 decimal places)

- Information gain measures how much the entropy is reduced when splitting the dataset.
- Calculated by:

  IG(Feature (Y)) = H(Phone Usage) - H(Phone Usage — Feature (Y))

Solution

- **relationship status:**
  - Count of relationship status:
    - Single: 5 Low, 0 Medium, 0 High **(5)**
    - Married: 0 Low, 1 Medium, 3 High **(4)**
    - In a relationship: 2 Low, 4 Medium, 0 High **(6)**
  - Calculate the conditional entropy for each relationship status:
    - Single:

$$H(PhoneUsage|Single) = -(\frac{5}{5}\log_2(\frac{5}{5}) + \frac{0}{5}\log_2(\frac{0}{5}) + \frac{0}{5}\log_2(\frac{0}{5}))$$

$$= 0$$

    - Married:

$$H(PhoneUsage|Married) = -(\frac{0}{4}\log_2(\frac{0}{4}) + \frac{1}{4}\log_2(\frac{1}{4}) + \frac{3}{4}\log_2(\frac{3}{4}))$$

$$\approx 0.8113$$

    - In a relationship:

$$H(PhoneUsage|Inarelationship) = -(\frac{2}{6}\log_2(\frac{2}{6}) + \frac{4}{6}\log_2(\frac{4}{6}) + \frac{0}{6}\log_2(\frac{0}{6}))$$

$$\approx 0.9183$$

- Calculate the weighted average of the conditional entropy:

$$H(PhoneUsage|RelationshipStatus) = \frac{5}{15}\times 0 + \frac{4}{15}\times 0.8113 + \frac{6}{15}\times 0.9183$$

$$\approx 0.5836$$

- Calculate the Information Gain:

$$IG(RelationshipStatus) = 1.5058 - 0.5836$$

$$\approx 0.9222$$

- **Age:**
  - Count of Age:
    - ≥ 25: 6 Low, 1 Medium, 2 High **(9)**
    - ≤ 25: 1 Low, 4 Medium, 1 High **(6)**
  - Calculate the conditional entropy for each age:
    - ≥ 25:

$$H(PhoneUsage| \geq 25) = -(\frac{6}{9}\log_2(\frac{6}{9}) + \frac{2}{9}\log_2(\frac{2}{9}) + \frac{1}{9}\log_2(\frac{1}{9}))$$

$$\approx 1.2244$$

    - ≤ 25:

$$H(PhoneUsage| \leq 25) = -(\frac{1}{6}\log_2(\frac{1}{6}) + \frac{1}{6}\log_2(\frac{1}{6}) + \frac{4}{6}\log_2(\frac{4}{6}))$$

$$\approx 1.2516$$

  - Calculate the weighted average of the conditional entropy:

$$H(PhoneUsage|Age) = \frac{9}{15}\times 1.2244 + \frac{6}{15}\times 1.2516$$

$$\approx 1.2350$$

  - Calculate the Information Gain:

$$IG(Age) = 1.5058 - 1.2350$$

$$\approx 0.2708$$

- **Education Level:**
  - Count of Education Level:
    - High School: 4 Low, 0 Medium, 0 High **(4)**
    - College: 0 Low, 5 Medium, 0 High **(5)**
    - University: 3 Low, 0 Medium,3 High **(6)**
  - Calculate the conditional entropy for each education level:

* High School:

$$H(PhoneUsage|HighSchool) = -(\frac{4}{4}\log_2(\frac{4}{4}) + \frac{0}{4}\log_2(\frac{0}{4}) + \frac{0}{4}\log_2(\frac{0}{4}))$$
$$= 0$$

* College:

$$H(PhoneUsage|College) = -(\frac{0}{5}\log_2(\frac{0}{5}) + \frac{5}{5}\log_2(\frac{5}{5}) + \frac{0}{5}\log_2(\frac{0}{5}))$$
$$= 0$$

* University:

$$H(PhoneUsage|University) = -(\frac{3}{6}\log_2(\frac{3}{6}) + \frac{0}{6}\log_2(\frac{0}{6}) + \frac{3}{6}\log_2(\frac{3}{6}))$$
$$\approx 1$$

– Calculate the weighted average of the conditional entropy:

$$H(PhoneUsage|EducationLevel) = \frac{4}{15} \times 0 + \frac{5}{15} \times 0 + \frac{6}{15} \times 1$$
$$\approx 0.4$$

– Calculate the Information Gain:

$$IG(EducationLevel) = 1.5058 - 0.4$$
$$\approx 1.1058$$

- **Income:**
  – Count of Income:
    * $\leq$ 50K: 3 Low, 2 Medium, 3 High **(8)**
    * $\geq$ 50K: 4 Low, 3 Medium, 0 High **(7)**
  – Calculate the conditional entropy for each income:
    * $\leq$ 50K:

$$H(PhoneUsage| \leq 50K) = -(\frac{3}{8}\log_2(\frac{3}{8}) + \frac{2}{8}\log_2(\frac{2}{8}) + \frac{3}{8}\log_2(\frac{3}{8}))$$
$$\approx 1.5613$$

    * $\geq$ 50K:

$$H(PhoneUsage| \geq 50K) = -(\frac{4}{7}\log_2(\frac{4}{7}) + \frac{3}{7}\log_2(\frac{3}{7}) + \frac{0}{7}\log_2(\frac{0}{7}))$$
$$\approx 0.9852$$

- Calculate the weighted average of the conditional entropy:

$$H(PhoneUsage|Income) = \frac{8}{15} \times 1.5613 + \frac{7}{15} \times 0.9852$$

$$\approx 1.2925$$

- Calculate the Information Gain:

$$IG(Income) = 1.5058 - 1.2925$$

$$\approx 0.2133$$

The features choice for Root Node: The feature with the highest information gan is (Education) with IG ≈ 1.1058. This feature will be chosen as the root of the tree because it provides the highest reduction in entropy when splitting the dataset.

c) Using the root node you have choosen determine the rest of the nodes in the decision tree for the abovariancee data. Draw the full decision tre i.e keep splittng the nodes untill further splits do not lead to any information gain (you may not need all the features for this). For each split, show your working on why you chose this feature based on information gain.

Solution

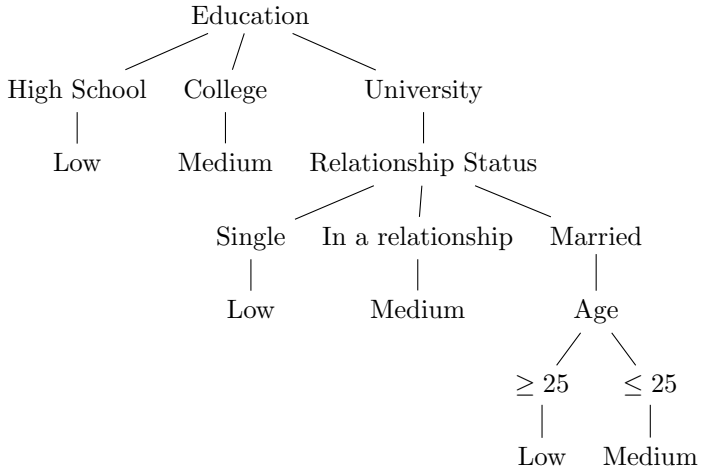
- Since we have chosen Education as the root node, we will split the dataset based on the values of Education.
- We will calculate the Information Gain for each feature at each node and choose the feature with the highest Information Gain.
- We will continue splitting the nodes until further splits do not lead to any Information Gain.
- The decision tree will look like this:
- **Root Node: Education**
  - **Branch1:High School**
    - Count of Phone Usage:
      - Low: 4
      - Medium: 0
      - High: 0
    - Since all the Phone Usage values are Low, this node is pure and we do not need to split further.
  - **Branch2: College**
    - Count of Phone Usage:
      - Low: 0
      - Medium: 5

        - High: 0
    - Since all the Phone Usage values are Medium, this node is pure and we do not need to split further.
  - **Branch3: University**
    - Count of Phone Usage:
      - Low: 3
      - Medium: 0
      - High: 3
    - The branch is impure,we need to split it further.
    - To considet the next split, from the information gain we have for each features within university:
      - **Relationship Status:** = 0.9222
      - **Age:** = 0.2708
      - **Income:** = 0.2133
    - The feature with the highest Information Gain is **Relationship Status**.

- **Splitting on Relationship status within University**
  - **Branch1: Single**
    - Count of Phone Usage:
      - Low: 5
      - Medium: 0
      - High: 0
    - Since all the Phone Usage values are Low, this node is pure and we do not need to split further.
  - **Branch2: In a relationship**
    - Count of Phone Usage:
      - Low: 2
      - Medium: 4
      - High: 0
    - Since all the Phone Usage values are Medium, this node is pure and we do not need to split further.
  - **Branch3: Married**
    - Count of Phone Usage:
      - Low: 0
      - Medium: 1
      - High: 3
    - The branch is impure, we need to split it further.
    - To consider the next split, from the information gain we have for each features within Married:
      - **Age:** = 0.9183

- **Income:** = 0.8113
  - The feature with the highest Information Gain is **Age**.

- The tree will be:



3 Random Forest

a) Build 3 decision trees using two features out of the three of each tree. Use the information gain to decide the feature to split on .Use majoity voting if all the samples in a leaf node do not have the same label.

Solution

- **Entropy of the Getting Sick**
  - Count of Getting Sick:
    - Yes: 4
    - No: 4
  - Calculate the Entropy of Getting Sick:

$$H(GettingSick) = -(\frac{4}{8}\log_2(\frac{4}{8}) + \frac{4}{8}\log_2(\frac{4}{8}))$$
$$= 1$$

- **Information Gain for each feature:**
  - **Age(A):**
    - Count of Age:
      - **Old** 3 yes, 3 no
      - **Young** 1 yes, 1 no
    - Calculate the conditional entropy for each age:
      - **Old:**

$$H(GettingSick|Old) = -(\frac{3}{6}\log_2(\frac{3}{6}) + \frac{3}{6}\log_2(\frac{3}{6}))$$
$$= 1$$

      - **Young:**

$$H(GettingSick|Young) = -(\frac{1}{2}\log_2(\frac{1}{2}) + \frac{1}{2}\log_2(\frac{1}{2}))$$
$$= 1$$

    - Calculate the weighted average of the conditional entropy:

$$H(GettingSick|Age) = \frac{6}{8} \times 1 + \frac{2}{8} \times 1$$
$$= 1$$

    - Calculate the Information Gain:

$$IG(Age) = 1 - 1$$
$$= 0$$

- **Social Distance(S):**
  - Count of Social Distance:
    - **Yes** 0 yes, 4 no
    - **No** 4 yes, 0 no
  - Calculate the conditional entropy for each social distance:
    - **Yes:**

$$H(GettingSick|Yes) = -(\frac{0}{4}\log_2(\frac{0}{4}) + \frac{4}{4}\log_2(\frac{4}{4}))$$

$$= 0$$

    - **No:**

$$H(GettingSick|No) = -(\frac{4}{4}\log_2(\frac{4}{4}) + \frac{0}{4}\log_2(\frac{0}{4}))$$

$$= 0$$

  - Calculate the weighted average of the conditional entropy:

$$H(GettingSick|SocialDistance) = \frac{4}{8} \times 0 + \frac{4}{8} \times 0$$

$$= 0$$

  - Calculate the Information Gain:

$$IG(SocialDistance) = 1 - 0$$

$$= 1$$

- **Handwashing (H):**
  - Count of Handwashing:
    - **Yes** 1 yes, 3 no
    - **No** 3 yes, 1 no
  - Calculate the conditional entropy for each handwashing:
    - **Yes:**

$$H(GettingSick|Yes) = -(\frac{1}{4}\log_2(\frac{1}{4}) + \frac{3}{4}\log_2(\frac{3}{4}))$$

$$\approx 0.8113$$

    - **No:**

$$H(GettingSick|No) = -(\frac{3}{4}\log_2(\frac{3}{4}) + \frac{1}{4}\log_2(\frac{1}{4}))$$

$$\approx 0.8113$$

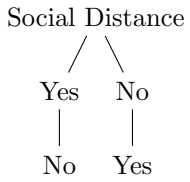
* Calculate the weighted average of the conditional entropy:

$$H(GettingSick|Handwashing) = \frac{4}{8} \times 0.8113 + \frac{4}{8} \times 0.8113$$
$$\approx 0.8113$$

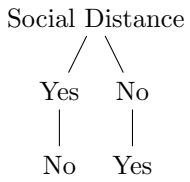
* Calculate the Information Gain:

$$IG(Handwashing) = 1 - 0.8113$$
$$\approx 0.1887$$

- **Tree 1: Social Distance (1) and Handwashing(0.8113)**
  - The feature with the highest Information Gain is **Social Distance**.
  - The tree will be:



- **Tree 2: Age(0) and Social Distance(1)**
  - The feature with the highest Information Gain is **Social Distance**.
  - The tree will be:



- **Tree 3: Age(0) and Handwashing(0.8113)**
  - The feature with the highest Information Gain is **Handwashing**.
  - The tree will be:

b) Given a new data point with the reatures (A = old, S=yes, H = no), use the trees you learned from part (a) to predict whether this person is sick.

Solution

- **Using Tree 1: Social Distance and Handwashing**
  - The new data point has the features: Social Distance = Yes, Handwashing = No.
  - The tree will predict that the person is **No**.
- **Using Tree 2: Age and Social Distance**
  - The new data point has the features: Social Distance = Yes.
  - The tree will predict that the person is **No**.
- **Using Tree 3: Age and Handwashing**
  - The new data point has the features: Handwashing = No.
  - The tree will predict that the person is **Yes**.

Final prediction using majoity voting will be: **No**

c) Briefly in 1-2 sentences comment on the advantage of random forests over decision tress from the perspective of bias-variance tradeoff.

Solution

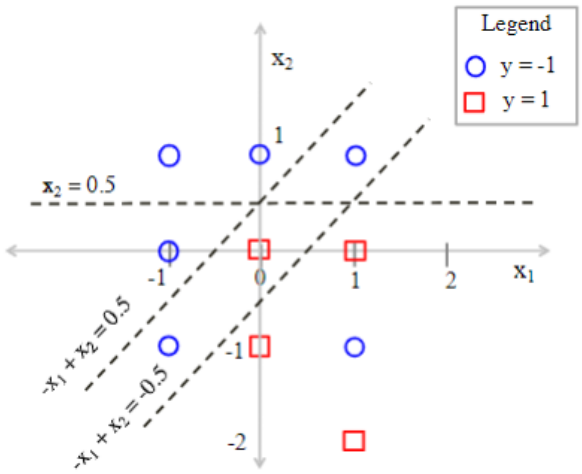
- Random forest are better than decision trees in terms of bias-variance tradeoff because they reduce overfitting and improve the model generalization. They generally reduce variance by averaging multiple decision trees that is through ensemble learning, this improves the overal performance of the model.

4 Adaboost

- Unlike random forest, Adaboost is an ensemble method that works by combining multiple weak learners to create a strong learner.
- Adaboost works by assigning weights to each training example and adjusting the weights of the incorrectly classified dataset.
Consider the training dataset with N = 10 points and 2-D featuress x = (x1, x2) as shown in the figure above. In this problem we will use binary linear decision boundary as the base classier and perfom two iterations of Adaboost.

a) Starting with equal weights $w\_1(n) = 1/10$ for all $N = 10$ points, which of the decision boundaries shown below gives the lowest error ϵ_1? Select one of the answers:

- Predict $y = 1$ if $x2 \leq 0.5$
- Predict $y = 1$ if $-x1 + x2 \leq -0.5$
- Predict $y = 1$ if $-x1 + x2 \leq 0.5$



Solution

- Option 1: correct missclassificationerror = 3/10 = 0.3
- Option 2: correct missclassificationerror = 2/10 = 0.2
- Option 3: correct missclassificationerror = 3/10 = 0.3

The decision boundary that gives the lowest error is **Predict** $y = 1$ **if** $-x1 + x2 \leq -0.5$ - **b**

b) Compute the error ϵ_1 and the contribution β_1 of the dicision boundary chosen in part (a) above.

Solution

- The error is given by:

$$\epsilon_1 = \sum_{i=1}^{N} w_1(i) \times \text{missclassification error}$$

$$= \frac{1}{10} \times 2 = 0.2$$

- The contribution is given by:

$$\beta_1 = \frac{1}{2} \log_2(\frac{1-\epsilon_1}{\epsilon_1})$$

$$= \frac{1}{2} \log_2(\frac{1-0.2}{0.2})$$

$$\approx 1$$

c) Compute the updated and normalized weights $w_2(n)$ for each of the data points as follows.

n	Coordinates	$w_2(n)$
1.	$(-1, 1)$	
2.	$(0, 1)$	
3.	$(1, 1)$	
4.	$(-1, 0)$	
5.	$(0, 0)$	
6.	$(1, 0)$	
7.	$(-1, -1)$	
8.	$(0, -1)$	
9.	$(1, -1)$	
10.	$(1, -2)$	

Table 2

Solution

The formular for updating weights is:

$$w_{t+1}(n) \propto w_t(n) \cdot 2^{-\beta_t y_n h_t(x_n)}$$

where:

- $w\_t(n)$ is the weight of sample n at iteration t.
- $beta_t$ the weight of weak classifier = 1
- y_n true label of the samples (1 or -1)
- $h_t(x_n)$ prediction of the weak classifier for the sample.
- Calculating the unnormalised weights:
  - For correctly classified points(where $y_n = h_t(x_n)$)

$$w_2(n) = w_1(n) \times 2^{-1} = \frac{1}{10} \times \frac{1}{2} = \frac{1}{20}$$

  - For misclassified points (where $y_n \neq h_t(x_n)$)

$$w_2(n) = w_1(n) \times 2^{1} = \frac{1}{10} \times 2 = \frac{2}{10} = \frac{1}{5}$$

- identify correct and incorrectly classifications using the chosen weak classifier from part (a) $y = 1$ if $-x_1 + x_2 \leq -0.5$

Index	Coordinates (x_1, x_2)	True Label y_n	Classifier Prediction $h_1(x_n)$	Correct?	Weight Update
1	$(-1, 1)$	-1	-1	correct	$\frac{1}{20}$
2	$(0, 1)$	-1	-1	correct	$\frac{1}{20}$
3	$(1, 1)$	-1	-1	correct	$\frac{1}{20}$
4	$(-1, 0)$	-1	-1	correct	$\frac{1}{20}$
5	$(0, 0)$	1	-1	incorrect	$\frac{1}{5}$
6	$(1, 0)$	1	1	correct	$\frac{1}{20}$
7	$(-1, -1)$	-1	-1	correct	$\frac{1}{20}$
8	$(0, -1)$	1	1	correct	$\frac{1}{20}$
9	$(1, -1)$	-1	1	incorrect	$\frac{1}{5}$
10	$(1, -2)$	1	1	correct	$\frac{1}{20}$

- Normalizing the weights $w_2(n)$:

$$\sum_{i=1}^{N} w_2(i) = \frac{1}{20} \times 8 + \frac{1}{5} \times 2 = \frac{8}{20} + \frac{4}{20} = \frac{12}{20} = \frac{4}{5}$$

$$w_2(n) = \frac{w_2(n)}{\sum_{i=1}^{N} w_2(i)}$$

– For correctly classified points:

$$w_2(n) = \frac{\frac{1}{20}}{\frac{4}{5}} = \frac{1}{20} \times \frac{5}{4} = \frac{1}{16}$$

– For misclassified points:

$$w_2(n) = \frac{\frac{1}{5}}{\frac{4}{5}} = \frac{1}{5} \times \frac{5}{4} = \frac{1}{4}$$

- Finalized normalized weights $w_2(n)$ table with index cordinates and the normalized new weights $w_2(n)$.

Index	Coordinates (x_1, x_2)	Normalized Weight $w_2(n)$
1	$(-1, 1)$	$\frac{1}{16}$
2	$(0, 1)$	$\frac{1}{16}$
3	$(1, 1)$	$\frac{1}{16}$
4	$(-1, 0)$	$\frac{1}{16}$
5	$(0, 0)$	$\frac{1}{4}$
6	$(1, 0)$	$\frac{1}{16}$
7	$(-1, -1)$	$\frac{1}{16}$
8	$(0, -1)$	$\frac{1}{16}$
9	$(1, -1)$	$\frac{1}{4}$
10	$(1, -2)$	$\frac{1}{16}$

d) Suppose you are given one more weak classifier that predicts $y = 1$ if $1.5x_1 + x_2 \leq 0.5$ as shown in the figure below. Compute the contribution β_2 of this classifier for the updated set of weights $w_2(n)$. Use the approximation $log\_23 \approx 1.6$

Solution

To find β_t:

$$\beta_t = \frac{1}{2} log_2 \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

From part c $\epsilon_2 = \frac{1}{4}$

Substitute:

$$\frac{1}{2} log_2 \left(\frac{1 - \frac{1}{4}}{\frac{1}{4}} \right)$$

$$\beta_2 = \frac{1}{2} \log_2 \left(\frac{1 - \frac{1}{4}}{\frac{1}{4}} \right)$$

$$= \frac{1}{2} \log_2(3)$$

$$\approx \frac{1}{2} \times 1.6$$
$$= 0.8$$

e) Combine the new classifier with the classifier that you obtained in part (a). Specify the label $y \in \{-1, 1\}$ predicted by this combined classifier for each data point. Do you observe any change in the prediction, compared to the prediction from only using part (a)'s classifier?

Solution

The two classifiers are:

- From part(a): $h_1(x) = 1$ if $-x_1 + x_2 \leq -0.5$; otherwise = -1
- From part(d): $h_2(x) = 1$ if $1.5x_1 + x_2 \leq 0.5$; otherwise = -1

The weights of the classifiers are:

- $\beta_1 = 1$
- $\beta_2 = 0.8$

The combine classifier $H(x)$

$$H(x) = sign(h_1(x) + 0.8h_2(x))$$

There is no change in prediction for data point 9a0, this could mean either

Index	Coordinates (x_1, x_2)	True Label y_n	$h_1(x)$	$h_2(x)$	$h_1(x) + 0.8h_2(x)$	$H(x)$
1	$(-1, 1)$	-1	-1	1	-0.2	-1
2	$(0, 1)$	-1	-1	-1	-1.8	-1
3	$(1, 1)$	-1	-1	-1	-1.8	-1
4	$(-1, 0)$	-1	-1	1	-0.2	-1
5	$(0, 0)$	1	-1	1	-0.2	-1
6	$(1, 0)$	1	1	-1	0.2	1
7	$(-1, -1)$	-1	-1	1	-0.2	-1
8	$(0, -1)$	1	1	1	1.8	1
9	$(1, -1)$	-1	1	1	1.8	1
10	$(1, -2)$	1	1	1	1.8	1

the second classifier is not working correctly.

f) If we continue adding more classifiers, each with error less than 0.5, how does the training error of the combined
classifier $f_T(x)$ change? Explain in 1- 2 sentences.

Solution

Adding more classifiers will tend to reduce the training error of the combine classifier since each new classifier will focus
on the misclassified points from the previous iteration hence leading to the improved accuracy, however if too many classifiers
are added it might lead to overfitting.

5 Neural Networks

Below is a deep network with inputs x_1, x_2. The internal nodes and activation functions are as shown in the figure and equations below. All variables are scalar values, and $\exp(x)$ refers to the function e^x. The activation functions of the nodes h_1, h_2, h_3 are ReLU (i.e. $r_1 = \max(h_1, 0)$ etc.), for node s_1 we have $s_1 = \max(r_2, r_3)$ and for the other nodes:

$$y_1 = \frac{\exp(r_1)}{\exp(r_1) + \exp(s_1)}, \quad y_2 = \frac{\exp(s_1)}{\exp(r_1) + \exp(s_1)}, \quad z = y_1 + y_2.$$

a) **Forward Propagation** Now, given:

$$x_1 = 1, \quad x_2 = -2, \quad w_{11} = 6, \quad w_{12} = 2, \quad w_{21} = 4, \quad w_{22} = 7, \quad w_{31} = 5, \quad w_{32} = 1,$$

compute the values of the internal nodes (shown in the table below). You may leave e in your answer.

h_1	h_2	h_3	r_1	r_2	r_3	s_1	y_1	y_2	z

Solution

- Calculate h_1, h_2, h_3:

$$h_1 = w_{11}x_1 + w_{12}x_2 = 6 \cdot 1 + 2 \cdot (-2) = 6 - 4 = 2$$

$$h_2 = w_{21}x_1 + w_{22}x_2 = 4 \cdot 1 + 7 \cdot (-2) = 4 - 14 = -10$$

$$h_3 = w_{31}x_1 + w_{32}x_2 = 5 \cdot 1 + 1 \cdot (-2) = 5 - 2 = 3$$

- Calculate r_1, r_2, r_3 using ReLU activation:

$$r_1 = \max(h_1, 0) = \max(2, 0) = 2$$

$$r_2 = \max(h_2, 0) = \max(-10, 0) = 0$$

$$r_3 = \max(h_3, 0) = \max(3, 0) = 3$$

- Calculate s_1:

$$s_1 = \max(r_2, r_3) = \max(0, 3) = 3$$

- Calculate y_1, y_2:

$$y_1 = \frac{\exp(r_1)}{\exp(r_1) + \exp(s_1)} = \frac{\exp(2)}{\exp(2) + \exp(3)} = 0.2689$$

$$y_2 = \frac{\exp(s_1)}{\exp(r_1) + \exp(s_1)} = \frac{\exp(3)}{\exp(2) + \exp(3)} = 0.7311$$

- Calculate z:

$$z = y_1 + y_2 = 0.2689 + 0.7311 = 1$$

- Final values:

h_1	h_2	h_3	r_1	r_2	r_3	s_1	y_1	y_2	z
2	-10	3	2	0	3	3	0.2689	0.7311	1

b) **Bounds on variables**

- Find the range of feasible values for y_1.

  Solution

  – The range of feasible values for y_1:

$$y_1 = \frac{\exp(r_1)}{\exp(r_1) + \exp(s_1)}$$

  Since r_1 and s_1 are non-negative, $\exp(r_1)$ and $\exp(s_1)$ are positive. Therefore, y_1 is a fraction where the numerator is positive and the denominator is the sum of two positive numbers. Hence, y_1 will always be between 0 and 1.

$$0 < y_1 < 1$$

- The range of feasible values for z:

  –

$$z = y_1 + y_2$$

  Since y_1 and y_2 are probabilities and their sum is always 1:

$$z = y_1 + y_2 = 1$$

  Therefore, the range of feasible values for z is:

$$z = 1$$

c) **Backpropagation**

Compute the gradient expressions shown in the table below analytically. The answer should be an expression that may include any of the nodes in the network $(x_1, x_2, h_1, h_2, h_3, r_1, r_2, r_3, s_1, y_1, y_2, z)$ or weights $(w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32})$.

$\frac{\partial h_1}{\partial w_{12}}$	$\frac{\partial h_1}{\partial x_1}$	$\frac{\partial r_1}{\partial h_1}$	$\frac{\partial y_1}{\partial r_1}$	$\frac{\partial y_1}{\partial s_1}$	$\frac{\partial z}{\partial y_1}$	$\frac{\partial z}{\partial x_1}$	$\frac{\partial s_1}{\partial r_2}$

5.0.1 Solution

- Calculate $\frac{\partial h_1}{\partial w_{12}}$:

$$h_1 = w_{11}x_1 + w_{12}x_2 \implies \frac{\partial h_1}{\partial w_{12}} = x_2 = -2$$

- Calculate $\frac{\partial h_1}{\partial x_1}$:

$$h_1 = w_{11}x_1 + w_{12}x_2 \implies \frac{\partial h_1}{\partial x_1} = w_{11}$$

- Calculate $\frac{\partial r_1}{\partial h_1}$:

$$r_1 = \max(h_1, 0) \implies \frac{\partial r_1}{\partial h_1} = \begin{cases} 1 & \text{if } h_1 > 0 \\ 0 & \text{if } h_1 \leq 0 \end{cases}$$

  since h1 = 2, therefore: $\frac{\partial r_1}{\partial h_1} = 1$

- Calculate $\frac{\partial y_1}{\partial r_1}$:

$$y_1 = \frac{\exp(r_1)}{\exp(r_1) + \exp(s_1)} \implies \frac{\partial y_1}{\partial r_1} = y_1(1 - y_1)$$

- Calculate $\frac{\partial y_1}{\partial s_1}$:

$$y_1 = \frac{\exp(r_1)}{\exp(r_1) + \exp(s_1)} \implies \frac{\partial y_1}{\partial s_1} = -y_1 y_2$$

- Calculate $\frac{\partial z}{\partial y_1}$:

$$z = y_1 + y_2 \implies \frac{\partial z}{\partial y_1} = 1$$

- Calculate $\frac{\partial z}{\partial x_1}$:

$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial r_1} \cdot \frac{\partial r_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial x_1} = 1 \cdot y_1(1 - y_1) \cdot 1 \cdot w_{11} = y_1(1 - y_1)w_{11}$$

- Calculate $\frac{\partial s_1}{\partial r_2}$:

$$s_1 = \max(r_2, r_3) \implies \frac{\partial s_1}{\partial r_2} = \begin{cases} 1 & \text{if } r_2 > r_3 \\ 0 & \text{if } r_2 \leq r_3 \end{cases}$$

6 Neural Networks for MNIST Digit Recognition

a) Find the Jacobian $\frac{\partial \mathbf{x_k}}{\partial \mathbf{x_{k-1}}}$ for the exponential linear unit, $X_k = ELU(x_{k-1})$.

Solution

The ELU activation function is defined as:

$$ELU(x) = \begin{cases} x, & \text{if } \mathbf{x} > 0, \\ \alpha(e^{\mathbf{x}} - 1), & \text{if } \mathbf{x} \leq 0, \end{cases}$$

where α is a parameter (commonly set to 1 or 0.9).

Compute the derivative for the cases:
When $x > 0$:
If $x > 0$, the derivative is:

$$\frac{\partial ELU(x)}{\partial x} = \frac{d}{dx}x = 1.$$

When $x \leq 0$:
If $x \leq 0$, the derivative is:

$$\frac{\partial ELU(x)}{\partial x} = \frac{d}{dx}[\alpha(e^x - 1)] = \alpha e^x.$$

Therefore, the Jacobian $\frac{\partial \mathbf{x_k}}{\partial \mathbf{x_{k-1}}}$ for the ELU activation function is:

$$\frac{\partial \mathbf{x_k}}{\partial \mathbf{x_{k-1}}} = \frac{d}{dx}ELU(x) \begin{cases} 1, & \text{if } \mathbf{x} > 0, \\ \alpha e^{\mathbf{x}}, & \text{if } \mathbf{x} \leq 0. \end{cases}$$

b) Find the Jacobian $\frac{\partial \mathbf{x_k}}{\partial \mathbf{x_{k-1}}}$ for Dense layer, $X_k = \mathbf{W x_{k-1}} + \mathbf{b}$.

Solution

The Jacobian transformation above where:

- x_{k-1} is an n-dimensional input vector of size $n \times 1$.
- $\mathbf{W_k}$ is the weight matrix of size $m \times n$.
- $\mathbf{b_k}$ is the bias vector of size $m \times 1$.
- $\mathbf{x_k}$ is an m-dimensional output vector.

Compute partial derivatives

$$J_D = \frac{\partial \mathbf{x_k}}{\partial \mathbf{x_{k-1}}} = \mathbf{W_k}.$$

This is because the derivative of a linear transformation with respect to its input is simply the weight matrix $\mathbf{W}$, as the bias $\mathbf{b}$ does not depend on $\mathbf{x_{k-1}}$.

c) Find the Jacobian $\frac{\partial \mathbf{x_k}}{\partial \mathbf{b}}$ for Dense layer, $\mathbf{X_k} = \mathbf{W x_{k-1}} + \mathbf{b}$.

Solution

The Jacobian $\frac{\partial \mathbf{x_k}}{\partial \mathbf{b}}$ for the Dense layer is the partial derivative of the output $\mathbf{x_k}$ with respect to the bias vector $\mathbf{b}$. Since the bias $\mathbf{b}$ is added element-wise to the output of the linear transformation, the derivative is simply the identity matrix of size $m \times m$, where m is the number of output neurons.

$$\frac{\partial \mathbf{x_k}}{\partial \mathbf{b}} = \mathbf{I_m},$$

where $\mathbf{I_m}$ is the $m \times m$ identity matrix.

d) Since $\mathbf{W}$ is a 2-D, the Jacobian $\frac{\partial x_k}{\partial \mathbf{W}}$ can only be expressed by matrix-vector multiplication if we flatten it toa vector. Instead, to preserve its dimensio, find the gradient $\frac{\partial x_k[a]}{\partial \mathbf{W}[b,c]}$ for indices a, b, c (so that we can write $\frac{\partial x_k}{\partial \mathbf{W}}$ as a 3-D tensor). Here, a is an in index of X_k, b indexes a row of $\mathbf{W}$ and c indexes a column of $\mathbf{W}$.

Solution

The gradient $\frac{\partial x_k[a]}{\partial \mathbf{W}[b,c]}$ can be derived as follows:

From the Dense layer equation:

$$x_k[a] = \sum_c W[a,c] \cdot x_{k-1}[c] + b[a]$$

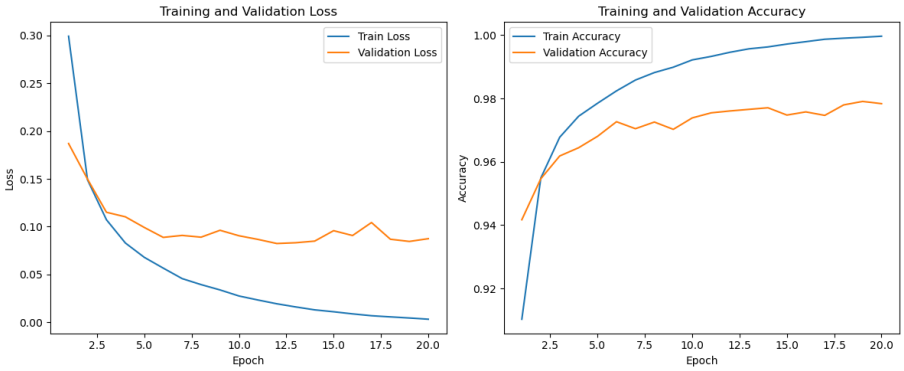
The partial derivative with respect to $\mathbf{W}[b,c]$ is:

$$\frac{\partial x_k[a]}{\partial \mathbf{W}[b,c]} = \begin{cases} x_{k-1}[c], & \text{if } a = b, \\ 0, & \text{if } a \neq b. \end{cases}$$

This means that the gradient is non-zero only when the row index a of the output matches the row index b of the weight matrix. The value of the gradient in this case is the corresponding input $x_{k-1}[c]$.

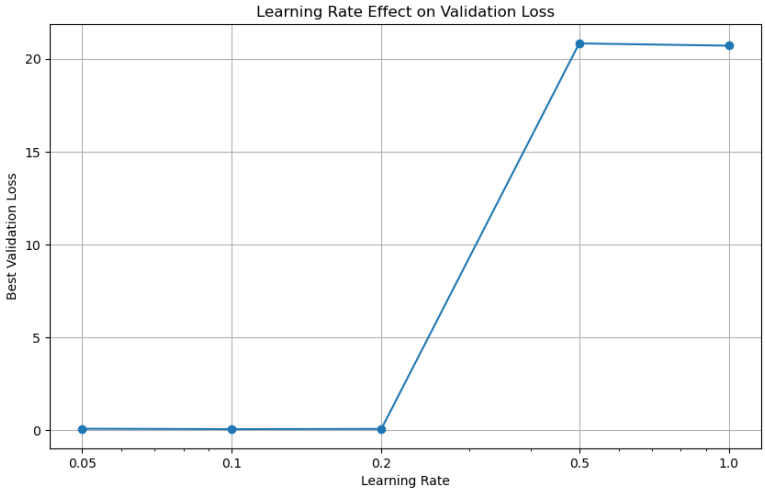
Training and Evaluation

a) Plot for the train and validation accuracy after each epoch during training



b) Test trained network

c) What is the effect of changing the learning rate:



- Too high learning rate leas to the model jumping too far in the weight updates missing the optimal Solutions. The loss also increases instead of decreasing
- Too low leearning rate leads to the model to update waits slowly making the training very slow. The training also takes along time to improve

- The model convereged to the good solution efficiently on the optimal leanint rate and the training was fast and stable.

Bonus Adams

How does a properly-tuned Adam Optimizer compare to SGD?

- Adam convergence faster than SGD.
- Adam is less sensitive to learning rate than SGD that needs careful tuning
- Adam handle noisy gradiesnt well compared to SGD that struggles without momentum
- Adam is faster and works well with the minimal tunid however it may not generalize well as a SGD.
- SGD with momentum nees more careful tuning but it can lead to better final accuracy in some cases.

How does the sensitivity of Adam to its learning rate compare to SGD.

- Adam is less sensitive to the initial learning rate than SGD since it adapts to the gradient scale.