Michael Coval

Professor Dera

IMGS-362

2-7-2026

Homework 1

1. Definitions
    a. **Feature vector**: a data structure which contains information about something. It could be an image where the vector comes from irradiance on the sensor. It could also be something like someone's [age, height, weight].
    b. **Decision boundary**: This is a surface (with the same number of dimensions as the feature) which separates input features into different classes. This is used as a multi-dimensional threshold for classification problems.
    c. **Classification:** This is the process of separating features into non-numeric categories. This is accomplished by applying a decision boundary to numeric data. The classes may represent categories like "cat" or "dog".
    d. **Regression:** This is the process of applying known mathematical trends between variables to predict unknown values.
    e. **Supervised learning:** This is when a neural network is given datasets along with the ground truth. An example might be pictures of animals, where each picture has a corresponding label; "giraffe", "whale" … etc.
    f. **Unsupervised learning:** This is when a neural network is given data, but without any ground truth labels. The model is left to decipher how to separate categories of data.
    g. **Overfitting:** This is when a model memorizes the training dataset. This results in a model that performs well on training data but performs poorly on data that it hasn't seen before. This can be caused by either using too small of a training data set, or an oversimplified model.
    h. **Decision Tree:** This model is based on recursively making decisions. Starting at the root node, decisions are made at every internal node, creating a tree-like structure. The information eventually ends with a leaf node, which represents the final output of the model.
    i. **Information Gain:** Information is based on the statistical separation of two or more classes of variables. Therefore, information gain, is a measure of how well a model separates variables into their corresponding classes. If

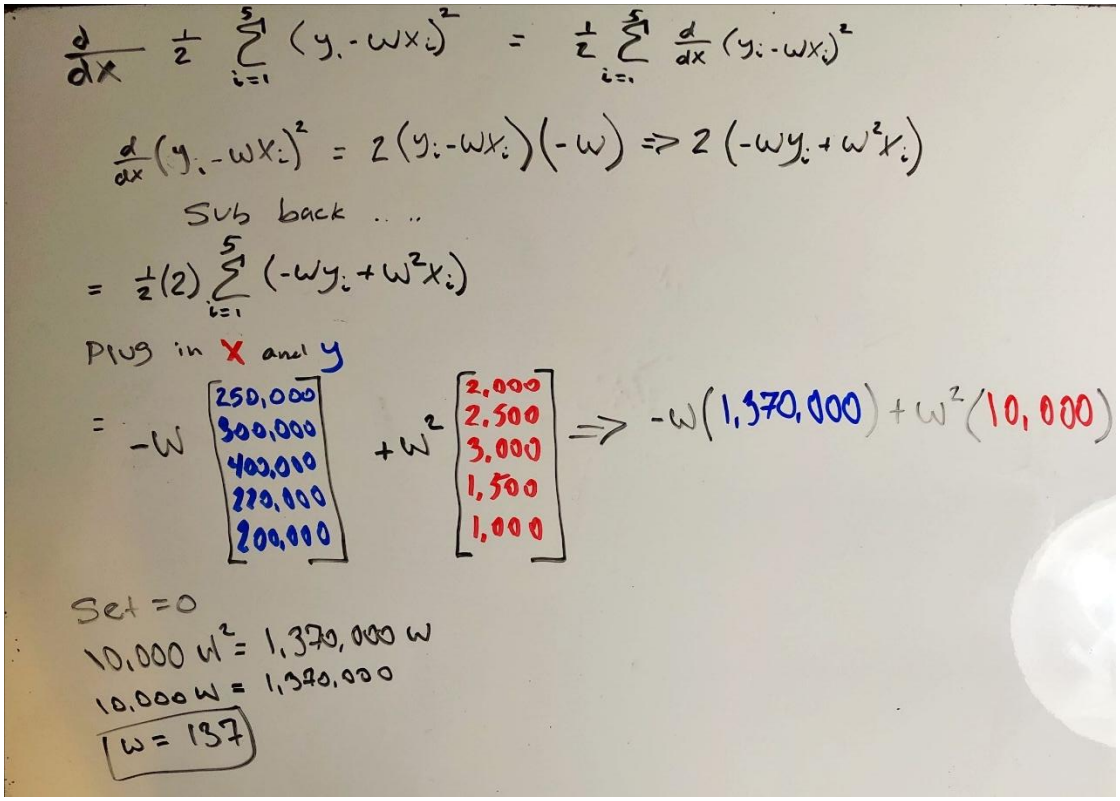
there is no information gain, the model was not effective at distinguishing classes.

j. **Entropy:** This is the measure of disorder in a dataset. A dataset with high entropy contains data among many different categories.

k. **Vanishing Gradient:** This problem typically occurs with models that have many layers. During the backpropagation, the partial derivative of every weight is computed with respect to the loss function. This is repeated for every layer of neurons. As the gradients are calculated backwards, the gradient magnitudes closer to the end nodes are much greater than those closer to the beginning…hence the term vanishing gradient is used. This problem makes training and optimizing models difficult, as it is hard to calculate how early layers of neurons effect the result.

2. .

a. W = 137



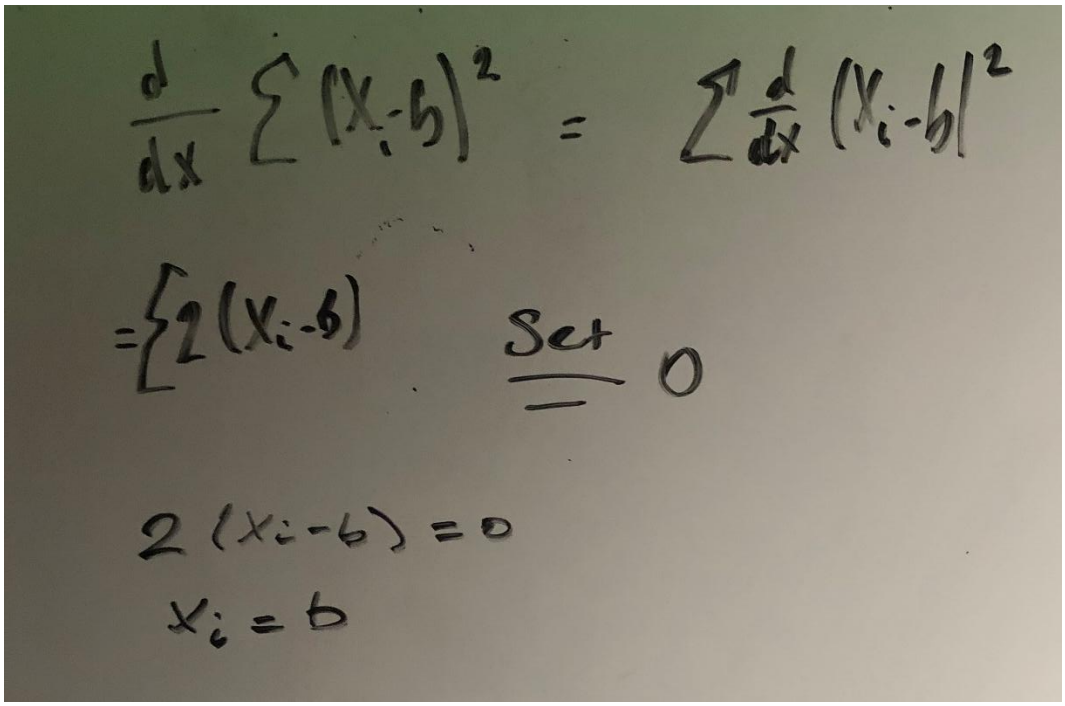
b. Y = 287,700

3. $-logP(\boldsymbol{y}|\boldsymbol{X}) - \sum_{i=1}^{n}[\frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\left(y^i - \boldsymbol{w}^T\boldsymbol{x}^i - b\right)^2]$

4. Linear regression predicts a numeric value (that is continuous), but logistic regression predicts a category.

5. $-logp(y \mid X) = \sum_{n=1}^{N}[y_n \log(\hat{y}_n) + (1 - y_n)\log(1 - \hat{y}_n)]$

6. The SoftMax function helps to turn raw model outputs into conditional probabilities. This is specifically for multi-class problems, where a binary classification model (like a sigmoid) is not sufficient for calculating probability.
7. Optimal value: $b = x_i$



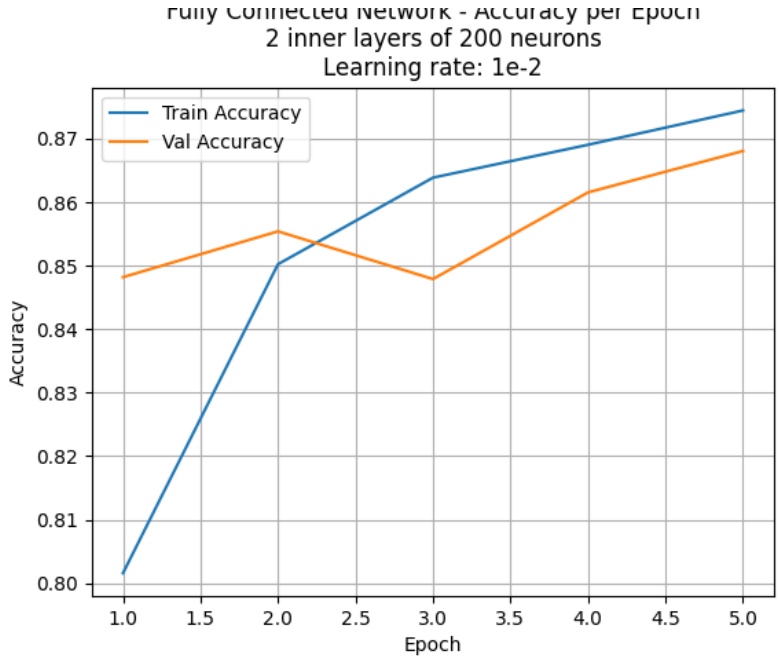
Part 2:

I chose to implement a minimalist network. I was motivated to see how well I could balance fast training and accuracy. I chose to use two dense layers of 200 neurons each.

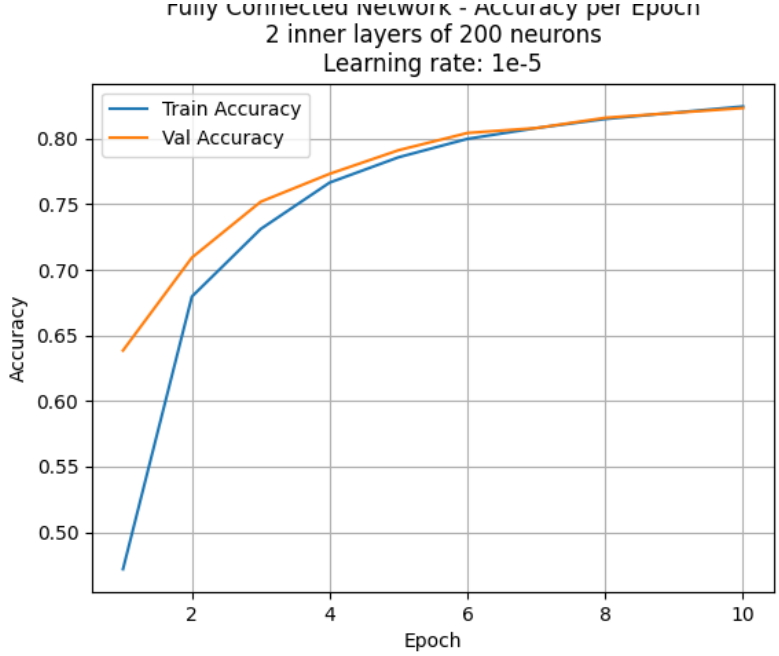
Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 200)	157,000
dense_2 (Dense)	(None, 200)	40,200
output_softmax (Dense)	(None, 10)	2,010

Total params: 199,210 (778.16 KB)
Trainable params: 199,210 (778.16 KB)
Non-trainable params: 0 (0.00 B)

The example had 500 neurons, so my model has about half as many. Originally, I also chose to speed up the learning rate to 1e-2, which is 10x faster than the example code. Because I was impatient, I only ran 5 epochs. This yielded the following:



This model got a decent accuracy of about 86.5% on the validation data. However, there is a noticeable gap between the training and validation datasets. This suggests that the model is overfitting the data. As a result, I chose to decrease the learning rate to 1e-5 and increased the epochs to compensate. Below is the result:



This model performed roughly the same at 85% accuracy, however there is a nice agreement between the training and validation data.

GitHub repository link: https://github.com/MichaelCoval/Machine-Learning-Public-.git