Homework 1 - 1/23/23

CS 5787 Deep Learning Spring 2023

Your homework solution must be typed. Your submission must cite any references used (including articles, books, code, websites, and personal communications). All solutions must be written in your own words, and you must program the algorithms yourself. If you do work with others, you must list the people you worked with. Submit your solutions as a PDF to Canvas.

Problem 1 - Probability Review

Recall these rules from probability: Bayes' rule is

$$P(B|A) = P(A|B)P(B).\ P(A)$$

Joint probability's relationship with conditional probability is

$$P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

If two events A and B are independent then their joint probability is

$$P(A \cap B) = P(A)P(B)$$

If two events A and B are not mutually exclusive then,

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

If two events A and B are mutually exclusive then $P(A \cap B) = 0$.

Hint: None of the questions in problem 1 have the same answer.

In this question, assume that the kangaroos have equal probability of having children that are male or female. Kangaroos almost always give birth to a single offspring.

Let $P(O = F)$ be the probability of the older offspring being female and let $P(Y = F)$ be the probability of the younger offspring being female.

Part 1 (3 points)

Suppose a kangaroo has two children. What is the probability that both are female? Show your derivation.

Assuming that the probability of each gender is independent, we can derive the probability of two children both being females as:

$$P(C_1 = F, C_2 = F) = P(C_1 = F)P(C_2 = F) = \frac{1}{2} \cdot \frac{1}{2} = \mathbf{\frac{1}{4}}$$

Where C_1 and C_2 are respectively the first and second child, and F is female.

Part 2 (3 points)

Suppose a kangaroo has two children and the oldest is female. What is the probability that both are female? Show your derivation.

Now it is given that the oldest child C_1 is a female. Intuitively, if the oldest child is a female, we only have to compute the probability of the youngest being a female, which is 50%. Mathematically, we can yield the same result by diving the probability of both children being females, over the probability of the first one being a female.

$$P(C_1 = F, C_2 = F \mid C_1 = F) = \frac{P(C_1 = F, C_2 = F)}{P(C_1 = F)} = \frac{\frac{1}{4}}{\frac{1}{2}} = \mathbf{\frac{1}{2}}$$

Part 3 (3 points)

Suppose a kangaroo has two children and at least one of them is female. What is the probability that both are female? Show your derivation.

We start by defining some probabilities:

- $P(MM)$ = probability that both children are male
- $P(FF)$ = probability that both children are female
- $P(FX)$ = probability that at least one child is female

Because the probability of a child being any of the two genders is equal and independent, the probability of both children being males ($P(MM)$) is the same of them both being females, which is equal to $1/4$, as computed in part1.

Therefore, we can now compute the probability of at least one child being female, which will be equal to the complementary of both children being males, as follows:

$$P(FX) = 1 - P(MM) = 1 - P(MM) = 1 - \frac{1}{4} = \frac{3}{4}$$

Finally, we apply Bayes' theorem ot compute the probability of both children being females, given that at least one is a female, as follows:

$$P(FF \mid FX) = \frac{P(FX \mid FF)P(FF)}{P(FX)} = \frac{1 \cdot P(FF)}{P(FX)} = \frac{\frac{1}{4}}{\frac{3}{4}} = \mathbf{\frac{1}{3}}$$

Problem 2 - ML Review (13 points)

1. Are there differences between "Machine Learning algorithm" and "classification algorithm" and if so, what are they? (1 point)

Yes. Machine learning algorithms are a broader term that encompasses all algorithms that are able to learn a task without being explicitly programmed. Classification algorithms, instead, are a specific type of ML algorithms that are able to categorize data in a finite, categorical number of output classes. Therefore, classification algorithms are a subset of ML algorithms.

2. Explain linear regression and logistic regression, their differences, and when you would apply each. (3 points)

Linear regression's objective is to learn the relationship between continuous variables. It is used when the relationship between the dependent variable and independent variables can be approximated by a linear function. Logistic regression, instead, tries to map a binary dependent variable to the independent variables using a logistic function.

Therefore, the choice on which model to use is highly dependent on the type of the dependent variable and the task, i.e. if it is a classification or regression problem.

For example, linear regression would be a suitable choice to predict the price of a flight ticket, based on a set of features like number of spots left, distance, time to take off, etc. Instead, logistic regression would be preferred in predicting binary results, such as the likelihood of the democrat candidate winning the elections based on demographic information and past polls.

3. Pick a regression loss function, explain how it works, and give an example. What are the advantages/disadvantages of the function you picked? (3 points)

A well-known and widely used loss function is Mean Squared Error (MSE). As the name suggests, MSE computes the average squared difference between the predictions and the actual targets.

Considering the previous linear regression example to predict flight ticket prices, MSE is a suitable metric to evaluate the performance of the model, since it represents how much the predicted prices deviate, on average, from the true prices.

MSE is very common because of its simplicity, the calculation needed to compute it is trivial. Furthermore, it is differentiable, thus allowing the use of gradient descent for optimization purpuses. Finally it is convex, i.e. features one single global minima, which eliminates the risk of converging to less optimal local minima.

However, MSE is also very sensitive to outliers, which can result in large errors in noisy data, although most predictions are pretty accurate.

4. Read about Gradient Descent and explain it using a real life example (snowboarding?) (3 points)

Gradient descent is a widely used algorithm to optimize the model performance through minimizing the loss function.

Let's imagine a snowboarder trying to find the shortest path to ski down the slope. In this methaphor, the mountain is the cost function and steepness of the slope is the gradient.

Gradient descent iteratively computes the updated direction in which the snowboarder should ski towards, in order to ski down as much as possible in a unit step, until he reaches the bottom (i.e. the local/global minima).

Through this iterative process, the model eventually converges to a local/global minima, that maximizes the performance of the algorithm.

5. You're working on solving a problem where you can use AI to make binary decisions given a set of inputs. How would you compute the errors your system might make, and how would you decide when your system is good enough to be deployed? (3 points)

In classification tasks, a standard approach in evaluating the performance of the system is to compute the confusion matrix, which gives you insights on the accuracy, precision and recall of the system.

Then, based on the task that the model is solving, I would reason on the most suitable way to evaluate the model. Accuracy is not always the best metric, let's imagine for example that we are trying to predict a rare disease, hence we are working on a very unbalanced dataset, with positive cas only accounting for 5% of the total data points. In this case, optimizing the accuracy of the model would likely make it converge to always predicting "negative", which would result in 95% accuracy.

Furthermore, we would like to minimize the number of false negatives, since we want to ideally catch all cases as early as possible to start a cure. Hence, recall (TP / (TP + FN)) is more important in this specific case to precision (TP / (TP + FP)). Often we would like to optimize for a linear combination of these two metrics, which is called F-score.

To decide when the system is good enough to be deployed, we can set a validation performance threshold based on results found in literature for the same task, or on the results obtained by professionals using no AI tool.

Finally, in making the deployment decision, it is also important to meet computational complexity requirements and manage costs.

Problem 3 - Setting Up Python

Part 0 - Installing Anaconda and PyTorch

Most assignments can be completed either by running and testing code on your local machine (see “Local installation”) or executing the code on the cloud through Google Colab. If you have time, you might want to try both options (running code locally and on Colab) so that you gain experience and figure out which workflow suits you best.

Local installation:

Python 3.6+, numpy, scipy, and matplotlib are necessary for the homeworks in this class. You can obtain all of these packages in one go by installing Anaconda. For your text editor, we recommend VSCode, although if you have another preference you’re welcome to use any editor you’d like.

After installing your Python 3.6+ environment along with the other toolboxes, you may optionally install PyTorch, but it will not be required until Homework 2.

Google Colab:

Google Colab is a free hosted version of Jupyter Notebook. Unlike Python files which are executed as an entire script, notebooks are executed in blocks of code called cells. To get started, go to the Colab homepage at https://colab.research.google.com/ (https://colab.research.google.com/). Create a new notebook. Type in a line of Python code and run it (and start the notebook) by pressing Shift + Enter.

Part 1 - Visualizing CIFAR-10 (4 points)

The CIFAR-10 dataset contains 60,000 RGB images from 10 categories. It can be found here: https://www.cs.toronto.edu/~kriz/cifar.html (https://www.cs.toronto.edu/~kriz/cifar.html). It can alternatively be loaded via HuggingFace datasets (make sure to `pip install datasets` before importing).

Using the first CIFAR-10 training batch file, display the first three images from each of the 10 categories as a 3 × 10 image array. The images are stored as rows, and you will need to reshape them into 32 × 32 × 3 images if you load up the raw data yourself. It is okay to use the PyTorch toolbox for loading them or you can roll your own.

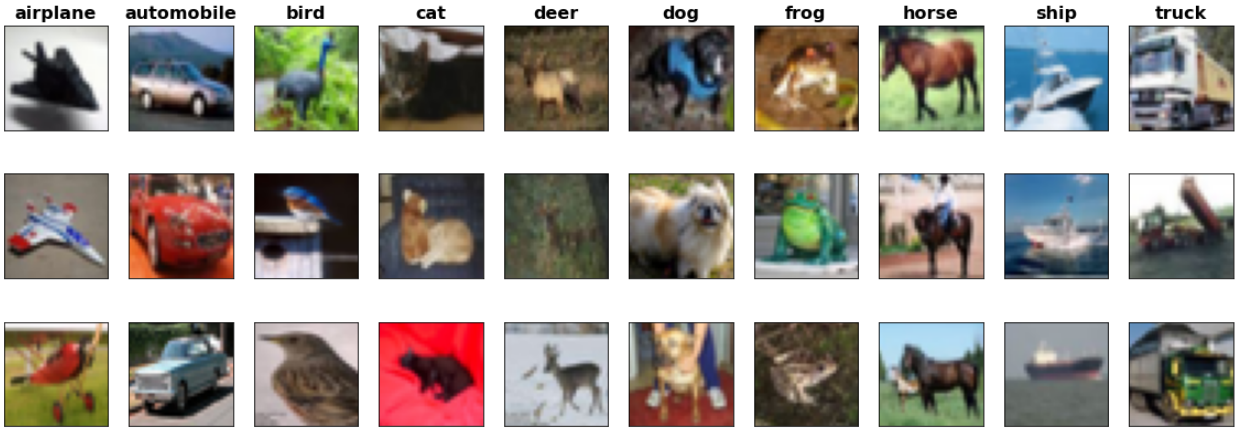
Image and Code:

In [58]:
```
def format_plot(ax, xlab = "", ylab = "", title = "", legend = False, xticks = None, yticks = None):
    ax.set_xlabel(xlab,  fontsize = 14)
    ax.set_ylabel(ylab,  fontsize = 14)
    ax.set_title(title,  fontsize = 16, fontweight = "bold")
    if legend:
        ax.legend(fontsize = 14)
    if xticks is not None:
        ax.set_xticks(xticks)
    if yticks is not None:
        ax.set_yticks(yticks)
```

In [109]:
```
def unpickle(file):
    import pickle
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict
```

In [112]:
```
import numpy as np

batch_1_unpickled = unpickle('cifar-10-batches-py/data_batch_1')
meta_data_unpickled = unpickle('cifar-10-batches-py/batches.meta')


imgs = np.array(batch_1_unpickled[b'data']).reshape(10000, 3, 32, 32).transpose(0, 2, 3, 1)
labs = np.array(batch_1_unpickled[b'labels'])
labnames = meta_data_unpickled[b'label_names']
```

In [114]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(3, 10, figsize=(20, 7))

for i in np.unique(labs):
    idxs = np.where(labs == i)[0][0:3]
    for j, idx in enumerate(idxs):
        ax[j, i].imshow(imgs[idx])
        format_plot(ax[j, i], xticks = [], yticks = [])

        if j == 0:
            format_plot(ax[j, i], title = labnames[i].decode("ascii"))

plt.show()
```



Part 2 - Playing with NumPy (4 points)

Write a function called gaussian(n, m) that returns an n × m NumPy array, each entry of which is a random number drawn from the standard normal distribution (Gaussian with mean 0 variance 1). Generate a 10x10 grid of 2-D points using this function and then make a scatter plot of the points using Matplotlib.

Scatter Plot and Code:

In [115]:
```
def gaussian(n, m):
    mu, sigma = 0, 1
    return np.random.normal(mu, sigma, (n, m))
```

In [127]:
```
fig, ax = plt.subplots(1, 1, figsize=(10, 7))

norm10_10 = np.concatenate([[gaussian(10, 10)], [gaussian(10, 10)]])
ax.scatter(norm10_10[0], norm10_10[1])

format_plot(ax, xlab = "Gaussian X", ylab = "Gaussian Y", title = "Norm~(0, 1) 10x10 Grid")
plt.show()
```