Quiz 1

CSCI-567 – Machine Learning

Fall 2023

1 Honor code

In the pursuit of excellence and integrity, I commit to the following Honor Code for this examination:

Honesty: I pledge that all answers and work on this examination will be my own. I will not give, receive, or use any unauthorized assistance, including electronic devices, notes, or external resources.

Respect: I understand the value of a level playing field and will not compromise the integrity of this examination for my peers.

Responsibility: If I observe any violations of this Honor Code, I will report it to the appropriate authorities immediately.

Accountability: I understand that a breach of this Honor Code may result in academic penalties, up to and including failure of the course.

Trust: I will work diligently to uphold the trust bestowed upon me by my educators, peers, and institution, recognizing the privilege of taking this examination.

By taking this exam, I am affirming my adherence to these principles. I understand the expectations set forth and will uphold them for the duration of the examination.

Signature: ___________
Date: _________________

2 Core ML concepts

1. Consider minimization of function $y = x^4$ with respect to x. Assume the initial value of x is $x_0 = 1$.
   (a) Find the learning rate so that gradient descent converges to the minimum in one iteration.
   (b) Now, consider minimization of this function with a momentum optimizer with the learning rate fixed to 1. Find the value of the momentum hyperparameter (we called it β in class) so that the minimum is reached in two iterations (at x_2). Will x_3 still be at the minimum?

3 Linear Regression

1. Gotchitama, intrigued by DeLU, a newly discovered activation function, $a(x) = px + \cos(p)$ where p is a constant, designed a multi-layer perceptron (MLP) for a regression problem, incorporating this activation function between each layer.

   His MLP is structured as

   - Input layer with m neurons
   - Two hidden layers, each with n neurons
   - An output layer with k neurons (where k is the number of output variables)

   Assume all weights are trainable, and that the biases are incorporated into the inputs/weights at the zero index automatically.

   (a) Write out the full function in terms of the weight matrices, activations, and input variables involved.

   (b) How many trainable parameters does Gotchitama's MLP have (excluding the activation function constant)?

   (c) Prove that Gotchitama's MLP is no more expressive than a single-layer linear regression model designed for the same regression task (i.e. mapping m input variables directly to k output variables).

   (d) Why do real-world MLPs (e.g. with ReLU activations) benefit from depth (i.e. multiple layers)?

   (e) Under what conditions might Gotchitama's model be harder to train? *Hint: think about the magnitude of the weights...*

2. (a) Gotchitama wants to build a linear regression model that predicts house price based on the size of the house (x_1) and the number of parking spots (x_2) the house has. Find out all the linear regression model that minimize the training loss of the following training data:

size (x_1)	# of parking spots (x_2)	price (y)
500	0	50000
600	0	60000
700	0	70000

   (b) What is the linear regression model that minimizes the training loss if Gotchitama apply L2 regularization on the model weights? Write down the weights of the regression model when the weight of the regularization term is positive and very close to 0. Use 1 or 2 sentence to explain why you think this linear regression model is better than the ones Gotchitama found without L2 regularization in an intuitive way.

(c) Now consider this training data. Find out all the linear regression model that minimize the training loss (without any regularization).

size (x_1)	# of parking spots (x_2)	price (y)
500	5	50000
600	6	60000
700	7	70000

4 Perceptron, logistic regression

1. Gotchitama just learned about the Perceptron algorithm. Decided not to follow the implementation taught by Dani, Gotchitama always update the Perceptron as in Algorithm 1:

```
Algorithm 1 A Trojan's Perceptron
Require: Dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\}$, $(\mathbf{x}_i, y_i) \in \mathbb{R}^D \times \{-1, +1\}$.
  Initialize $\mathbf{w} = \mathbf{0} \in \mathbb{R}^D$
  while True do
    for $i = 1, 2, \cdots, N$ do
      $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x_i}$
    end for
  end while
```

That is, Gotchitama always updates the Perceptron weight regardless of the correctness of its prediction! You, a careful student, are tasked with correcting him. Can you come up with a counterexample dataset wherein the perceptron would make an infinite number of mistakes by following Gothitama's algorithm?

2. Consider a binary classification problem where you have a dataset with two features (x_1 and x_2) and two classes (Class A and Class B). You decide to use the Perceptron algorithm to create a classifier. The Perceptron has already been initialized with the following weights and bias:

   - Weight w_1 = 0.5
   - Weight w_2 = -0.3
   - Bias b = 0.2

   You are given a new data point with the following features: $x_1 = 0.7$, $x_2 = 0.9$. Using the provided Perceptron and the step function as the activation function, answer the following questions:

   (a) Calculate the weighted sum $\mathbf{w}^T\mathbf{x}$ for the given data point.

   (b) Determine the output class (Class A or Class B) based on the step function activation with a threshold of 0. If the output is 1, classify it as Class A; if it's 0, classify it as Class B.

(c) Assuming the data point belongs to class A, calculate the error for this data point.

(d) Assuming a learning rate (α) of 0.1, update the weights and bias for this data point using the Perceptron update rule.

Show your calculations step by step for each part of the question.

3. Logistic regression:

   (a) Recall the sigmoid function $\sigma(a) = \frac{1}{1+e^{-a}}$. Show that

$$\frac{d}{da}\sigma(a) = \sigma(a)(1 - \sigma(a))$$

   (b) Using the result from part (a) and the chain rule, derive an expression for the gradient of logistic regression's log likelihood:

$$l(\mathbf{w}) = \sum_{n=1}^{N} \log(1 + e^{(-y_n \mathbf{w}^\intercal x_n)})$$

5 Multiclass classification, feed forward neural networks, convolutional neural network

1. Gotchitama was a student taking CSCI-567. When Gotchitama was doing the homework, Gotchitama encountered some issues. Below is a part of Gotchitama's implementation for CIFAR-10 classification. Please help Gotchitama complete this homework by answering the following questions:

```python
class MLP(nn.Module):
    def __init__(self):
        super().__init__()
        self.mlp = torch.nn.Sequential(
            nn.Linear(A, B),
            nn.ReLU(),
            nn.Linear(C, D)
        )

    def forward(self, x):
        # Flatten the feature maps
        x = x.view(x.shape[0], -1)
        x = self.mlp(x)
        return x


model = MLP().to(device)
loss_fn = nn.NLLLoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)

# Training loop
epochs = 20
for epoch in range(epochs):
    model.train()
    for i, data in enumerate(train_dataloader, 0):
        inputs, labels = data
        inputs, labels = inputs.to(device), labels.to(device)

        outputs = model(inputs)
        loss = loss_fn(outputs, labels)
        loss.backward()
        optimizer.step()

```

(a) Gotchitama wanted to build a multi-layer perceptron network (MLP) with 256 hidden units. What are the numbers Gotchitama should use for the placeholder A, B, C, D in line 5 and line 7?

(b) Gotchitama found that the loss is aloways `nan`. What's wrong with Gotchitama's code and how to fix the problem?

(c) Even after fixing the `nan` loss issue, Gotchitama still could not get an accuracy much better than 10%. What's the other problem in the code and how to fix it?

2. After taking CSCI-567 Gotchitama gets an amazing machine learning job. For part of this job, Gotchitama does image classification with a ConvNet. In particular, Gotchitama's network has two convolution layers. The first has 6 filters and the second has 16. Both use a 5×5 kernel shape. An RGB input image goes through the first convolution layer, a nonlinearity, and max pooling with a 2×2 filter with stride 2. The output of this is then passed through the second convolution layer, a nonlinearity, and max pooling with a 2×2 filter with stride 2. This output is then handled by a different part of the network. Assume square images with no padding anywhere in the ConvNet. If the output of the ConvNet given a single input image can be reshaped to a 400-dimensional vector, what are the dimensions of the image?

6 Recurrent neural networks, attention, transformers, memory networks

1. One of Gotchitama's friends speaks a language not represented in GPT-3's training data. Gotchitama decides to train a language model on many terrabytes of high-quality data from the language.

(a) Suppose Gotchitama choose a GPT-3-like (decoder-only) model. Gotchitama uses the traditional decoder-only training objective. Recall that for all computations of $\mathbf{A} = \frac{\mathbf{Q}^T\mathbf{K}}{\sqrt{d_k}}$ in the network, $\mathbf{A}$ is of shape $n \times n$ for each batch item and head where n is the input sequence length. For all of the $n \times n$ matrices just described, Gotchitama replaces the traditional decoder-only masking function. In Gotchitama's masking function for row i and column j of $\mathbf{A}$ if $j - 1 > i$, that element of $\mathbf{A}$ is set to a very low number (the same one used in traditional masking) before softmax is applied. What do you expect the training and validation losses will look like over the course of 1 epoch over the training data? Why?

(b) Suppose Gotchitama's friend's language has the curious property that for any sequence of tokens $(x_1, x_2, \ldots, x_n)$, it holds that

$$\log_e(p((x_1, x_2, \ldots, x_n))) = \sum_{i=1}^{n} \log_e(p(x_i))$$

In this case, which kind of language model would be the best option? Briefly justify your answer. *Be as specific and detailed as possible.*

2. While cycling through his old statistics textbooks, Gotchitama stumbled upon a most curious regression model, defined as follows. For some dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1) \cdots, (\mathbf{x}_N, y_N)\}$, and a new input $\mathbf{x}$, this regression model predicts $\hat{y}$ by computing the following:

$$\hat{y} = \sum_{i=1}^{N} w(\mathbf{x}, \mathbf{x}_i) y_i,$$

where $w(\mathbf{x}, \mathbf{x}_i) \propto \exp(-\mathbf{x}^T \mathbf{x}_i)$ defines a probability distribution. In other words, $\hat{y}$ is computed as the weighted average of all the y_i's according to this distribution.

   (a) Can you rewrite this regression predictor in the jargon of attention mechanism (i.e. Attention(Q, K, V))? You need to provide explicit expressions for Q, K, and V.

   (b) Inspired by this connection, Gothitama scrambled through his textbook, and found another interesting choice of the probability distribution:

$$w(\mathbf{x}, \mathbf{x}_i; \tau) \propto \mathbb{1}_{\{\mathbf{x}^T \mathbf{x}_i \geq \tau\}} \exp(-\mathbf{x}^T \mathbf{x}_i),$$

   where τ is some hyper-parameter. What is the qualitative difference between this approach and the one without the indicator function $\mathbb{1}_{\{\mathbf{x}^T \mathbf{x}_i \geq \tau\}}$? How does this approach relate to the K-nearest neighbor method, and how is it different?

   (c) While implementing this regression predictor, Gotchitama made a blunder by *randomly shuffling* the dataset, that is,

```python
# X: dataset features with X.shape == (N, D)
# y: dataset target with y.shape == (N,)
N = len(X)
idx = np.arange(N)
np.random.shuffle(idx)
X = X[idx]
y = y[idx]
```

   Will Gotchitama's regressor now yield a different prediction, compared to before the shuffle? Explain your reasoning.