Midterm

Feb. 24, 2024

(Conv layer and its gradient)

1. (2') Design a 3×3 convolutional filter to detect edges in an image
2. (5') Denote the input 2D image as $\mathbf{I}(x, y)$, output image due to the edge detector as $\mathbf{O}(x, y)$. Suppose we incured a loss ℓ on $\mathbf{O}$, and the gradient of ℓ w.r.t. $\mathbf{O}(x, y)$ is $\frac{\partial \ell}{\partial \mathbf{O}[x,y]}$. Derive the gradient of loss w.r.t. $\mathbf{I}(x, y)$.
3. (3') If we stack many conv and ReLU layers after $\mathbf{O}$, what would be a potenial issue if training this network by gradient descent? How could we mitigate this issue?

(Backprop through Time) Consider a simple RNN model with scalar input x_t and scalar hidden state

$$h_t = a \cdot x_t + b \cdot h_{t-1},$$

where $1 \leq t \leq T$. We incur some loss ℓ on the last hidden state h_T.

1. (5') Denote the gradient of ℓ w.r.t. h_T as $\frac{\partial \ell}{\partial h_T}$. Derive the gradient of ℓ w.r.t. a and b.
2. (5') what would be potential issue(s) if T is very big? How could we mitigate?

Positional Encoding (PE)

1. (2') Describe when and why we need PE for transformer models.
2. (8') Consider d-dimensional sinusoidal PE for t-th position, defined as

   $$\mathbf{p}_t = [\cdots, \sin(\omega_k t), \cos(\omega_k t), \cdots], k = 0, \ldots, d/2 - 1,$$

   where $\omega_k = \frac{1}{10000^{2k/d}}$. The PE is added to token embeddings $\mathbf{x}_t$, *i.e.*, $\mathbf{y}_t = \mathbf{x}_t + \mathbf{p}_t$.

   Analyze the effect of PE. More specifically, how does $\mathbf{y}_t \cdot \mathbf{y}_\tau$ compare against $\mathbf{x}_t \cdot \mathbf{x}_\tau$? Discuss the impact on attention.

(Model Design) You want to create a grammar error correction model.

1. (10') Sketch the overall model architecture. Describe each building block, and give as much detail as possible.
2. (5') What is the input and output of the model? How would you train the model? Give as much detail as possible, e.g., training loss, any tricks that can benefit the training.
3. (10') At test time, how would you generate grammar corrected sequence? What metric would you use to evaluate quality of the model?

(Optimization and Generalization) We use MNIST dataset for this question. Please attach all your code. Details regarding MNIST dataset can be found in homework 2. Again we use the following code to create training, validation and test split.

```
import torch
from torchvision import datasets
train_all= datasets.MNIST('../data', train=True, download=True) # 60K images
train_data, val_data = torch.utils.data.random_split(
train_all, [50000, 10000], torch.Generator().manual_seed(0)) # train: 50K; val: 10K
test_data = datasets.MNIST('../data', train=False) # test: 10K
```

1. (10') Implement LeNet architecture, and train on all the 50K training samples with vanilla SGD. Attach learning curves and report test accuracy.
2. (10') How can we make the training loss converge faster? Implement your idea and compare the new training curve against that in question 1.
3. (10') Train on 20%, 50%, 80% of the full 50K training samples, and report test accuracies. On a 2D coordinate (x-axis: training size; y-axis: test accuracy), plot the accuracies, along with the case of 100% training samples (question 1). Discuss the result.
4. (15') How can we improve the accuracies for cases with fewer training samples? Implement your idea and compare against what you get for question 3.