ESE 546

HOMEWORK 2

HARRY GUAN [HARRYG1@SEAS],
COLLABORATORS: HARRY GUAN [HARRYG1@SEAS]

Problem 1.

(a) - times spent 7 minutes. In the given ResNet code, the batch normalization (BN) layer is applied before the ReLU activation, following the sequence Conv → BN → ReLU. This design choice is intentional and generally considered better practice. Applying BN before ReLU ensures that the activations passed into the nonlinearity have a normalized distribution with zero mean and unit variance, which helps stabilize training and improves convergence. If BN were placed after ReLU, many activations would become zero due to ReLU's thresholding, leading to inaccurate mean and variance estimates. As a result, placing BN before ReLU allows the network to maintain more consistent feature scaling and achieve better overall performance.

(b) - times spent 7 minutes. The calls model.train() and model.eval() in PyTorch are used to switch the model between training and evaluation modes. During training, model.train() enables layers like dropout and batch normalization to behave in their training state—dropout randomly zeroes activations, and batch norm updates its running statistics. In contrast, model.eval() sets these layers to evaluation mode, where dropout is disabled and batch norm uses its learned running averages instead of batch statistics. These calls are important to ensure consistent behavior during validation or testing. In HW1, we didn't include them because our custom library was a simpler implementation that didn't have layers like dropout or batch normalization, so there was no need to change the model's behavior between training and evaluation phases.

(c) - times spent 7 minutes.

(d) - times spent 7 minutes. Weight decay, or L2 regularization, penalizes large weights by adding a term proportional to the square of their magnitude to the loss function. This helps prevent overfitting by discouraging overly complex models. However, applying weight decay to bias terms is not appropriate because biases simply shift the activation functions and do not control the capacity or smoothness of the model in the same way as weights do. Penalizing them can lead to underfitting or prevent the model from properly centering its activations.

(e) - times spent 7 minutes.

```python
import torch
import torch.nn as nn
import torchvision.models as models

# Load the ResNet-18 model
model = models.resnet18(weights=None)

# Initialize parameter groups
bn_params = []            # (i) BatchNorm affine transform parameters
bias_params = []          # (ii) Biases of conv and fc layers
other_params = []         # (iii) All the rest

# Iterate over all modules and parameters
for module in model.modules():
    if isinstance(module, nn.BatchNorm2d):
        # BatchNorm affine parameters (weight and bias)
        bn_params.extend([module.weight, module.bias])
    elif isinstance(module, (nn.Conv2d, nn.Linear)):
        # Bias parameters of conv and fully connected layers
        if module.bias is not None:
            bias_params.append(module.bias)
        # The rest (usually weights)
        other_params.append(module.weight)
    else:
        # Any remaining parameters that dont fit above
        for param in module.parameters(recurse=False):
            other_params.append(param)

# Print summary of parameter counts
print(f"BatchNorm affine parameters: {sum(p.numel() for p in bn_params)}")
print(f"Bias parameters: {sum(p.numel() for p in bias_params)}")
print(f"All other parameters: {sum(p.numel() for p in other_params)}")
```

Problem 2.

(a) - times spent 7 minutes. **Solution:** We want to show that the optimization problem

$$\min_{A\in\mathbb{R}^{m\times r},\, B\in\mathbb{R}^{r\times n}} \|X - AB\|_F^2$$

is not convex.

Consider the simplest scalar case where $m = n = r = 1$ and $X = 1$. Then the problem reduces to minimizing

$$f(a, b) = (1 - ab)^2$$

over $a, b \in \mathbb{R}$.

Choose two points:

$$(a_1, b_1) = (1, 1), \quad (a_2, b_2) = (-1, -1).$$

We have

$$f(a_1, b_1) = (1 - 1)^2 = 0, \quad f(a_2, b_2) = (1 - (-1)(-1))^2 = 0.$$

Now consider their midpoint:

$$\left(\frac{a_1 + a_2}{2}, \frac{b_1 + b_2}{2} \right) = (0, 0),$$

for which

$$f(0, 0) = (1 - 0)^2 = 1.$$

If $f(a, b)$ were convex, we would have

$$f\left(\frac{(a_1, b_1) + (a_2, b_2)}{2} \right) \leq \frac{1}{2} f(a_1, b_1) + \frac{1}{2} f(a_2, b_2) = 0.$$

However, $f(0, 0) = 1 > 0$, which violates convexity.

Therefore, the function $\|X - AB\|_F^2$ is not convex in the joint variables (A, B). (It is convex in A when B is fixed, and vice versa, but not jointly convex.)

(b) - times spent 7 minutes. Let $X = U \Sigma V^\top$ be the singular value decomposition (SVD) of X, where

$$\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \dots, \sigma_p), \quad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0, \quad p = \min(m, n).$$

Partition

$$U = [U_r \; U_\perp], \quad V = [V_r \; V_\perp], \quad \Sigma = \begin{bmatrix} \Sigma_r & 0 \\ 0 & \Sigma_\perp \end{bmatrix},$$

where $\Sigma_r = \operatorname{diag}(\sigma_1, \dots, \sigma_r)$.

The global optimum of

$$\min_{A \in \mathbb{R}^{m \times r}, \, B \in \mathbb{R}^{r \times n}} \|X - AB\|_F^2$$

is achieved at

$$A^\star = U_r \Sigma_r^{1/2}, \quad B^\star = \Sigma_r^{1/2} V_r^\top.$$

Then

$$A^\star B^\star = U_r \Sigma_r V_r^\top,$$

which is the rank-r truncated SVD of X.

The minimum value of the objective function is

$$\min_{A,B} \|X - AB\|_F^2 = \sum_{i=r+1}^{p} \sigma_i^2.$$

(c) - times spent 7 minutes. The solution to the optimization problem is not unique.

While the optimal product $Y^\star = A^\star B^\star = U_r \Sigma_r V_r^\top$ is unique when there is a spectral gap $\sigma_r > \sigma_{r+1}$, the individual factors $A^\star$ and $B^\star$ are not unique.

For any invertible matrix $Q \in \mathbb{R}^{r \times r}$,

$$A' = A^\star Q, \qquad B' = Q^{-1} B^\star$$

is also an optimal solution since

$$A'B' = (A^\star Q)(Q^{-1} B^\star) = A^\star B^\star.$$

Hence, infinitely many pairs (A', B') achieve the same minimum objective value. A simple special case is obtained by choosing $Q = \mathrm{diag}(c_1, \ldots, c_r)$, which scales each column of $A^\star$ by c_i and divides the corresponding row of $B^\star$ by c_i, producing another valid optimal factorization.