Week 3 Mathematical Theory for Fine-Tuning on MedMNIST

1 Binary Classification Setup

In Week 3, models are fine-tuned on binary MedMNIST datasets such as *PneumoniaMNIST* and *BreastMNIST*. The mathematical setting is as follows.

Data and labels

Each input is an image represented as a vector

$$\boldsymbol{x} \in \mathbb{R}^d,$$

and each label is binary:

$$y \in \{0, 1\},$$

where, for example, $y = 1$ may denote "disease" (pneumonia, malignant) and $y = 0$ denotes "normal" or "benign".

The neural network defines a real-valued *logit*

$$z = f_\theta(\boldsymbol{x}) \in \mathbb{R},$$

where θ denotes the collection of all trainable parameters (backbone and head).

Bernoulli likelihood and logistic parameterization

The model produces a probability of the positive class using the logistic (sigmoid) function

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

This induces a Bernoulli conditional distribution:

$$\begin{aligned} p_\theta(y = 1 \mid \boldsymbol{x}) &= \sigma(z), \\ p_\theta(y = 0 \mid \boldsymbol{x}) &= 1 - \sigma(z). \end{aligned}$$

Equivalently,

$$p_\theta(y \mid \boldsymbol{x}) = \sigma(z)^y \big(1 - \sigma(z)\big)^{1-y}.$$

2 Binary Cross-Entropy as Negative Log-Likelihood

Single-sample loss

For a single sample $(\boldsymbol{x}, y)$, the negative log-likelihood (NLL) of the Bernoulli model is

$$\begin{aligned} \ell(\theta; \boldsymbol{x}, y) &= -\log p_\theta(y \mid \boldsymbol{x}) \\ &= -\log \Big(\sigma(z)^y \big(1 - \sigma(z)\big)^{1-y}\Big) \\ &= -\Big(y \log \sigma(z) + (1 - y) \log(1 - \sigma(z))\Big), \end{aligned}$$

which is exactly the *binary cross-entropy (BCE)* loss.

Empirical risk over a dataset

Given a dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, the empirical loss is

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} \ell(\theta; \boldsymbol{x}_i, y_i) = -\frac{1}{N}\sum_{i=1}^{N} \Big(y_i \log \sigma(z_i) + (1-y_i)\log(1-\sigma(z_i))\Big),$$

where $z_i = f_\theta(\boldsymbol{x}_i)$. Minimizing $L(\theta)$ is equivalent to maximum likelihood estimation for a Bernoulli model with logistic link function.

3 Gradients with Respect to Logits and Head Parameters

Gradient of BCE with respect to the logit

Let $z = f_\theta(\boldsymbol{x})$ and $\hat{p} = \sigma(z)$. The BCE loss for a single sample is

$$\ell(z, y) = -\Big(y \log \hat{p} + (1-y)\log(1-\hat{p})\Big).$$

Using $\hat{p} = \sigma(z)$ and $\frac{d\sigma}{dz} = \sigma(z)(1-\sigma(z))$, one obtains

$$\frac{\partial \ell}{\partial z} = \hat{p} - y = \sigma(z) - y.$$

Thus, the gradient of the loss with respect to the logit is the difference between the predicted probability and the true label.

Linear head on top of a backbone

In many Week 3 configurations, the network is decomposed into:

- a *backbone* feature extractor $h_\phi : \mathbb{R}^d \to \mathbb{R}^k$,
- a final *linear head* parameterized by $(\boldsymbol{w}, b)$.

Given an input $\boldsymbol{x}$, the backbone computes a feature vector

$$\boldsymbol{h} = h_\phi(\boldsymbol{x}) \in \mathbb{R}^k,$$

and the head produces the logit

$$z = \boldsymbol{w}^\top \boldsymbol{h} + b.$$

Using the chain rule:

$$\frac{\partial \ell}{\partial \boldsymbol{w}} = \frac{\partial \ell}{\partial z}\frac{\partial z}{\partial \boldsymbol{w}} = (\sigma(z) - y)\,\boldsymbol{h},$$
$$\frac{\partial \ell}{\partial b} = \frac{\partial \ell}{\partial z}\frac{\partial z}{\partial b} = \sigma(z) - y.$$

If the backbone parameters ϕ are trainable, then

$$\frac{\partial \ell}{\partial \phi} = \frac{\partial \ell}{\partial z}\frac{\partial z}{\partial \boldsymbol{h}}\frac{\partial \boldsymbol{h}}{\partial \phi} = (\sigma(z) - y)\,\boldsymbol{w}^\top \frac{\partial \boldsymbol{h}}{\partial \phi},$$

showing explicitly how the error signal propagates backward into the backbone.

Head-only vs. fine-tune-all

- **Head-only fine-tuning**: the backbone parameters ϕ are frozen. Only $(\boldsymbol{w}, b)$ are updated using gradients as above. Mathematically, this corresponds to learning a linear classifier on a fixed feature map $\boldsymbol{h} = h_\phi(\boldsymbol{x})$.
- **Fine-tune-all**: both backbone ϕ and head $(\boldsymbol{w}, b)$ are updated. The model is no longer restricted to a fixed feature representation, and the function class becomes richer.

4 Weight Decay and ℓ_2 Regularization

Regularized objective

To control overfitting, Week 3 uses weight decay (often implemented via AdamW). For simplicity, consider adding an ℓ_2 penalty on the head weights $\boldsymbol{w}$:

$$J(\boldsymbol{w}, b) = L(\boldsymbol{w}, b) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2,$$

where L is the empirical BCE loss and $\lambda > 0$ controls the strength of regularization.

The gradient of J with respect to $\boldsymbol{w}$ is

$$\nabla_{\boldsymbol{w}} J(\boldsymbol{w}, b) = \nabla_{\boldsymbol{w}} L(\boldsymbol{w}, b) + \lambda \boldsymbol{w},$$

since $\nabla_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 = \boldsymbol{w}$.

Gradient descent update and shrinkage

With learning rate $\eta > 0$, gradient descent produces the update

$$\begin{aligned} \boldsymbol{w}_{t+1} &= \boldsymbol{w}_t - \eta \nabla_{\boldsymbol{w}} J(\boldsymbol{w}_t, b_t) \\ &= \boldsymbol{w}_t - \eta \nabla_{\boldsymbol{w}} L(\boldsymbol{w}_t, b_t) - \eta\lambda \boldsymbol{w}_t \\ &= (1 - \eta\lambda)\, \boldsymbol{w}_t - \eta \nabla_{\boldsymbol{w}} L(\boldsymbol{w}_t, b_t). \end{aligned}$$

The factor $(1 - \eta\lambda)$ multiplies the previous weights, shrinking them toward zero independent of the data gradient. This motivates the name *weight decay*: the weights decay toward zero over iterations, unless the data gradient pushes them away.

Decoupled weight decay (AdamW)

In AdamW, weight decay is mathematically separated (decoupled) from the gradient of the loss:

$$\boldsymbol{w}_{t+1} = (1 - \eta\lambda)\, \boldsymbol{w}_t - \eta\, \widehat{g}_t,$$

where $\widehat{g}_t$ is the Adam-adapted estimate of $\nabla_{\boldsymbol{w}} L(\boldsymbol{w}_t, b_t)$. The key idea is that the regularization term does not change the gradient estimate itself, but acts as a separate multiplicative shrinkage.

5 Class Imbalance and Weighted Cross-Entropy

Imbalanced data

Let N_1 and N_0 denote the number of positive and negative samples, respectively, in the training set:

$$N_1 = \#\{i : y_i = 1\}, \qquad N_0 = \#\{i : y_i = 0\}, \qquad N = N_0 + N_1.$$

Datasets like PneumoniaMNIST and BreastMNIST often exhibit significant imbalance (i.e., $N_1 \neq N_0$), which can affect both training and evaluation.

Weighted BCE

A common approach is to assign different weights to the two classes in the BCE:

$$\ell_{\mathrm{w}}(z, y) = -\Big(w_1\, y \log \sigma(z) + w_0\, (1-y) \log(1-\sigma(z))\Big),$$

where w_1 is the weight for the positive class and w_0 is the weight for the negative class. The empirical loss is then

$$L_{\mathrm{w}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell_{\mathrm{w}}(z_i, y_i).$$

A simple choice is to set weights inversely proportional to class frequency:

$$w_1 = \frac{1}{N_1}, \qquad w_0 = \frac{1}{N_0}.$$

Heuristically, this balances the aggregate contribution of each class to the loss function, so that the model cannot minimize the objective by ignoring the minority class.

Accuracy and Balanced Accuracy

Given a classifier that outputs binary predictions $\hat{y}$, the confusion matrix is

	$\hat{y}=0$	$\hat{y}=1$
$y=0$	TN	FP
$y=1$	FN	TP

with counts for true/false positives/negatives.

The standard accuracy is

$$\mathrm{Acc} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}.$$

The *balanced accuracy* averages the true positive and true negative rates:

$$\mathrm{BA} = \frac{1}{2} \left(\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} + \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \right).$$

When one class is much more frequent than the other, a classifier that always predicts the majority class can achieve high Acc but poor BA, making balanced accuracy a more informative measure of performance in imbalanced settings.

6 ROC Curves, AUROC, and Calibration (ECE)

6.1 ROC curve and AUROC

Assume the model outputs probabilities

$$\hat{p}_i = p_\theta(y = 1 \mid \boldsymbol{x}_i) = \sigma(z_i).$$

For any threshold $t \in [0, 1]$, define the corresponding predicted labels:

$$\hat{y}_i(t) = \begin{cases} 1, & \hat{p}_i \geq t, \\ 0, & \hat{p}_i < t. \end{cases}$$

The true positive rate (TPR) and false positive rate (FPR) at threshold t are

$$\mathrm{TPR}(t) = \mathbb{P}(\hat{y} = 1 \mid y = 1),$$
$$\mathrm{FPR}(t) = \mathbb{P}(\hat{y} = 1 \mid y = 0),$$

which in practice are estimated using the empirical confusion matrix at threshold t.

The *ROC curve* is the set of points

$$\big(\mathrm{FPR}(t), \mathrm{TPR}(t)\big)$$

as t varies from 0 to 1. The *Area Under the ROC Curve* (AUROC) is defined as

$$\mathrm{AUROC} = \int_0^1 \mathrm{TPR}(t)\, d\mathrm{FPR}(t),$$

representing the probability that a randomly chosen positive example is ranked higher (in terms of score $\hat{p}$ or logit z) than a randomly chosen negative example. AUROC is threshold-free and is particularly useful when class imbalance is present.

6.2 Expected Calibration Error (ECE)

Beyond discrimination (how well positives are separated from negatives), it is often important to assess *calibration*: whether predicted probabilities match empirical frequencies.

Let the predictions $\hat{p}_1, \ldots, \hat{p}_N$ be partitioned into M disjoint bins $B_1, \ldots, B_M$, according to intervals

$$0 = a_0 < a_1 < \cdots < a_M = 1,$$

where

$$B_m = \{i : \hat{p}_i \in (a_{m-1}, a_m]\}.$$

For each bin B_m, define:

$$\mathrm{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i \qquad \text{(average predicted confidence)},$$
$$\mathrm{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{I}[y_i = 1] \qquad \text{(empirical accuracy)}.$$

The *Expected Calibration Error* (ECE) is defined as

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} \left|\mathrm{acc}(B_m) - \mathrm{conf}(B_m)\right|.$$

A perfectly calibrated model would satisfy $\mathrm{acc}(B_m) = \mathrm{conf}(B_m)$ for all m, giving ECE = 0. In medical imaging scenarios, well-calibrated probabilities are crucial when decisions depend directly on the predicted risk scores (thresholding a probability to trigger further tests).

7 Backbone + Head Parameterization and Expressivity

Decomposition into feature extractor and classifier

Let ϕ denote the parameters of a convolutional backbone (ResNet18), and $(\boldsymbol{w}, b)$ denote the parameters of the final linear head. The network can be written as

$$\boldsymbol{h} = h_\phi(\boldsymbol{x}) \in \mathbb{R}^k, \qquad z = \boldsymbol{w}^\top \boldsymbol{h} + b.$$

Thus, the overall mapping is

$$f_\theta(\boldsymbol{x}) = \boldsymbol{w}^\top h_\phi(\boldsymbol{x}) + b, \qquad \theta = (\phi, \boldsymbol{w}, b).$$

Head-only fine-tuning as linear classification in feature space

If the backbone ϕ is kept fixed (initialized from ImageNet pretraining), the model class reduces to

$$\mathcal{F}_{\text{head-only}} = \left\{ \boldsymbol{x} \mapsto \boldsymbol{w}^\top h_\phi(\boldsymbol{x}) + b \;\middle|\; \boldsymbol{w} \in \mathbb{R}^k,\ b \in \mathbb{R} \right\},$$

which is equivalent to a linear classifier in the feature space $\mathbb{R}^k$. Training in this regime amounts to solving a regularized logistic regression problem on the fixed features $\boldsymbol{h}$.

Fine-tuning all parameters

If both ϕ and $(\boldsymbol{w}, b)$ are trainable, the function class expands to

$$\mathcal{F}_{\text{full}} = \left\{ \boldsymbol{x} \mapsto \boldsymbol{w}^\top h_\phi(\boldsymbol{x}) + b \;\middle|\; \phi \in \Phi,\ \boldsymbol{w} \in \mathbb{R}^k,\ b \in \mathbb{R} \right\},$$

where Φ denotes the parameter space of the backbone network. This class is strictly larger (assuming non-degenerate parameter spaces) and has much higher capacity.

In practice:

- Head-only fine-tuning typically has fewer trainable parameters and may generalize better when the target dataset (a MedMNIST subset) is small, because it behaves like a low-dimensional generalized linear model on top of rich features.
- Fine-tuning all layers allows the feature extractor to adapt strongly to the new domain, which can reduce bias but may increase variance and risk of overfitting unless controlled by regularization (weight decay, data augmentation, early stopping).

8 Learning Rate Schedules (Optional)

For completeness, a simple learning-rate schedule often used in practice is the exponential decay schedule:

$$\eta_t = \eta_0 \cdot \gamma^t, \qquad 0 < \gamma < 1,$$

where η_0 is the initial learning rate and t is the iteration or epoch index.

As $t \to \infty$, $\eta_t \to 0$, which is consistent with conditions for convergence in many stochastic optimization settings (Robbins–Monro-type conditions where learning rates eventually decrease). In practice, the schedule must be chosen so that:

- the learning rate is initially large enough to escape poor local minima and make rapid progress; and
- it decays slowly enough that the optimizer can settle into a good region without premature stagnation, but fast enough to stabilize training and avoid oscillations late in optimization.

The sections above summarize the core mathematical ideas underlying Week 3: binary logistic models and BCE, gradient computation for head-only vs. full fine-tuning, ℓ_2 regularization and weight decay, handling class imbalance, and evaluation metrics (AUROC and ECE) that are particularly relevant for medical image classification.