CSE 25 – Test 2 Preparation Notes

1 Topics in Scope

The section outlines the key topics and ideas covered on Test 2. They are meant to help you focus your studying, not to enumerate every possible question. You should still review all relevant worksheets, homework, programming assignments (PA1 and PA2), in-class examples, and assigned readings.

1.1 Learning as Minimizing Error

Fitting a Line and Error

You should be able to:

- Compute predictions using $y = wx + b$.
- Compute pointwise signed and absolute error.
- Compute Mean Absolute Error (MAE).
- Compute Mean Squared Error (MSE).
- Distinguish between total error and mean error.
- Distinguish between MAE and MSE.
- Explain how squaring errors in MSE affects large mistakes.
- Explain why MSE is preferred for gradient-based optimization.
- Explain how error provides feedback for improving parameters.

Error as a Function of Parameters

You should be able to:

- Express error as a function $E(w, b)$ of parameters.
- Interpret a graph of error vs parameter.

- Determine whether slope is positive, negative, or zero.
- Identify whether a point is a minimum.
- Determine how a parameter changes with gradient descent.
- Explain what it means for the slope (derivative) to be zero.

Finite Differences and Partial Derivatives

You should be able to:

- Compute the finite-difference approximation:

$$\frac{f(w+\epsilon)-f(w)}{\epsilon}$$

- Determine its sign and interpret its meaning.
- Explain how it approximates a derivative.
- Identify correct finite-difference approximations for

$$\frac{\partial L}{\partial w}, \quad \frac{\partial L}{\partial b}.$$

- Explain what it means to hold one variable fixed when computing a partial derivative.

Gradient Descent

You should be able to:

- State and apply the gradient descent update rule:

$$\text{new parameter} = \text{old parameter} - \text{learning rate} \times \text{derivative}.$$

- Use the sign of the derivative to determine update direction.
- Explain the role of the learning rate.
- Explain what happens if the learning rate is too large.

1.2 Regression vs Classification

You should be able to:

- Distinguish between regression and classification tasks.
- Identify task type based on output.
- Explain why digit prediction is considered classification.
- Identify appropriate loss functions (e.g., MSE vs cross-entropy).
- Write and compute predictions for linear models with multiple inputs.
- Explain how linear models form decision boundaries.

1.3 The Perceptron

You should be able to:

- Compute perceptron score:

$$z = \sum_i w_i x_i + b.$$

- Apply the perceptron decision rule.
- Explain how perceptron differs from regression.
- Explain why perceptron outputs are discrete.
- Write the perceptron loss:

$$L(y, z) = \max(0, -yz).$$

- Determine when perceptron loss is zero or positive.
- Identify the perceptron update rule.
- Determine when updates occur.
- Compute one update step.
- Explain how updates affect the decision boundary.
- Explain why data order and multiple epochs matter.
- Explain why signed labels ($+1$, -1) are used.

1.4 Logistic Regression and Binary Cross-Entropy

Sigmoid

You should be able to:

- Explain sigmoid output range $(0, 1)$.
- Interpret sigmoid output as confidence.
- Explain how confidence affects loss magnitude.
- Explain how thresholding converts confidence to class prediction.
- Compare sigmoid to sign activation.

Binary Cross-Entropy

The formula for BCE will be provided.
You should be able to:

- Explain how BCE behaves when the model is confident and correct.
- Explain how BCE behaves when the model is confident and wrong.
- Explain why BCE becomes large when predicted probability of the correct class is very low.
- Explain why BCE encourages confident and correct predictions.
- Compare BCE to perceptron loss conceptually.

1.5 Backpropagation and Gradient Flow

Forward Computation

You should be able to:

- Order forward sequence: input $\rightarrow$ score $\rightarrow$ activation $\rightarrow$ loss.
- Identify intermediate values (z, p) vs final loss.
- Explain how loss depends on parameters through intermediate steps.

Chain Rule

You should be able to:

- Identify all quantities involved in computing loss.
- Determine which local derivatives are needed.
- Apply the chain rule to compute $\frac{dL}{dw}$.
- Combine provided derivative rules correctly.
- Determine the correct backward order of computation.

You will not be required to remember any derivative rules - these will be provided.

Simplified Sigmoid + BCE Gradient

You should be able to:

- State:
$$\frac{dL}{dz} = p - y.$$
- Use the sign of $p-y$ to determine whether the score should increase or decrease.
- Interpret cases $p > y$, $p < y$, and $p \approx y$.

Finite Differences vs Backpropagation

You should be able to:

- Explain that finite differences method we used in class requires multiple forward passes.
- Determine how cost scales with number of parameters.
- Explain that backprop computes all gradients *efficiently* in one backward pass.

1.6 Model Structure and Expressivity

Common Computational Structure

You should be able to:

- Recognize shared structure: inputs $\rightarrow$ weighted sum $\rightarrow$ activation $\rightarrow$ loss.
- Identify activation and loss for linear regression, perceptron, logistic regression, and softmax
- Identify the structure of the core computational unit, **neuron** as a combination of: a weighted sum followed by an activation function

Softmax and Multi-Class Classification

You should be able to:

- Explain that softmax converts multiple scores into a probability distribution and why outputs sum to 1.
- Explain how to choose the prediction from the softmax values.
- Explain why loss simplifies to $L = -\log(p_{\text{correct}})$ when using one-hot labels.
- Explain how loss behaves when correct-class probability is high or low.
- Explain why softmax + CCE is still a single-layer linear model.

XOR and Limits of Linear Models

You should be able to:

- Explain why XOR is not linearly separable.
- Recognize linearly separable vs non-separable data.
- Explain why logistic regression and perceptron fail on XOR.
- Explain what it means for a boundary to be linear.

Hidden Layers and Nonlinearity

You should be able to:

- Describe what a hidden layer is.
- Explain why stacking linear layers remains linear.
- Describe a 2–2–1 neural network structure and parameter counts.
- Explain conceptually how hidden neurons increase expressivity.
- Explain why nonlinear activations are necessary.
- State output ranges of activation functions sigmoid, tanh, ReLU.
- Identify typical activation usage in output vs hidden layers.
- Apply chain rule through multi-layer graphs.
- Explain why **autodiff** is useful in deeper models conceptually.

  You will not be required to implement autodiff.

1.7 Evaluation and Generalization

Accuracy and Confusion Matrix

You should be able to:

- Compute accuracy.
- Compute TP, FP, FN, TN.
- Interpret confusion matrix entries.
- Explain why accuracy can mislead on imbalanced data.

Precision and Recall

You should be able to:

- Compute precision and recall.
- Interpret high precision vs high recall.
- Explain tradeoffs.
- Choose appropriate metric between the two in a given scenario.

Train / Validation / Test

You should be able to:

- Explain roles of training, validation, and test sets.
- Explain why the test set should be used only once.
- Distinguish parameters vs hyperparameters.

Overfitting

You should be able to:

- Define overfitting.
- Interpret training vs validation curves.
- Identify overfitting behavior.
- Explain what happens if training continues.
- Describe ways to reduce overfitting.

1.8 Interpreting Code

You will not be asked to write code on the test.

However, you should be comfortable reading and reasoning about short Python code snippets similar to what we saw in class and in the worksheets. This includes explaining, at a high level, what a piece of code is doing, how values flow through it, and how small changes in the code might affect the behavior of the system. This may include:

- Predicting the output of a short code snippet
- Tracing how values change through a loop or function
- Identifying what a variable represents

2 Expectations and Preparation

How to Prepare Effectively

To prepare for Test 2:

- Review Worksheets 6–11.
- Revisit Programming Assignments 1 and 2 (focus on the underlying concepts, not the code implementation details).
- Review homework problems and examples discussed in lecture.
- Practice tracing gradients through simple computation graphs and connecting loss, gradients, and parameter updates.
- Practice interpreting confusion matrices and evaluation metrics.

What Kinds of Questions to Expect

Test 2 will include a mix of:

- True/False questions
- Multiple-choice questions
- Matching questions
- Ordering / drag-and-drop questions
- Select-all-that-apply questions
- Short-answer questions requiring 1–3 sentences

- Numerical answer questions (light computation only)
- Questions that ask you to interpret or reason about a short code snippet
- Questions that require applying provided derivative rules using the chain rule
- Some questions may present a small computation graph and ask you to reason about how gradients flow.
- Some questions may describe a training or evaluation scenario and ask you to interpret what is happening.

What the Test Will Not Focus On

Test 2 will not focus on:

- Writing code implementations
- Formal proofs
- Deriving derivatives from scratch - Derivative rules will be provided when needed; you are responsible for applying them correctly.
- Memorizing long formulas
- Heavy arithmetic or long calculations
- Advanced mathematics