Statistical Learning - Classification

Assignment 3

Kernels, Convolutional Networks, and Tree Ensembles

100 marks

Written work. Show the main steps of each proof and calculation. For experiments, include the requested tables and figures and refer to them in your explanations. Report classification error as the fraction of incorrect predictions. Label plot axes and curves. Questions 1 and 3 require written answers only; Questions 2, 4, and 5 require code and written answers.

Submission. This is an individual assignment. Upload written answers, tables, and figures to the matching questions in Crowdmark. Submit your code and results to the **Assignment 3 Dropbox** in Learn as one `WatIAM_A3.zip`, replacing `WatIAM` with your Waterloo username. Include `Q2.py`, `Q4.py`, and `Q5.py` (or their `.R` or `.m` equivalents), all required helper and input files, and the three requested results CSVs. Include a short README listing run commands, dependencies, seeds, and model settings. Each script must run independently after the ZIP is extracted.

Programming. Use Python, R, or MATLAB. You may use libraries for array operations, random-number generation, CSV input/output, and plotting. In Question 2, you may use neural-network libraries and automatic differentiation to construct and train the specified networks; no manual backpropagation is required. In Question 4, use a library base-tree learner, but implement bootstrap sampling and hard-vote aggregation yourself. A package such as R's `ranger` is permitted when configured to fit **one tree at a time**, with internal resampling disabled. Do not use a ready-made multi-tree ensemble. In Question 5, implement candidate selection, AdaBoost updates, and prediction yourself; do not use a ready-made boosting routine.

Data and notation. Read the accompanying data README before loading the files. Each observation is a column: $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$. Uppercase letters denote matrices, bold lowercase letters denote vectors, and lowercase non-bold letters denote scalars. Supplied labels are -1 and $+1$; a score or vote sum of zero predicts $+1$. Use test data only for the prescribed evaluations, never to change model settings or choose a stopping point.

Generative AI survey (required). AI use is permitted. After completing the assignment, open Learn and choose **Submit** $\rightarrow$ **Surveys** $\rightarrow$ **Assignment 3: Generative AI Use Survey**. Everyone must complete the survey, including students who used no generative AI. Responses do not affect your assignment grade. The survey is not anonymous, so completion can be checked.

Individual Verification of Understanding. You may be asked to complete a brief individual quiz or oral discussion about your assignment, including explaining your reasoning, answering related questions, or explaining and making small modifications to your code. You must demonstrate your own understanding of all submitted work, including work developed with permitted generative AI assistance. These checks must be completed without generative AI or assistance from others. If you cannot demonstrate adequate understanding, your overall assignment mark may be reduced.

Late Submission Policy. Submissions received up to 24 hours after the deadline receive a 10% penalty on the earned mark. Submissions more than 24 hours late receive a grade of zero.

Question 1: Kernels from fixed tree partitions

10 marks

Consider $b \geq 1$ fixed decision trees. Let $\ell_s(\boldsymbol{x})$ be the terminal-leaf number assigned to input $\boldsymbol{x}$ by tree s. Leaf numbers are local to each tree. Define

$$k(\boldsymbol{x}, \boldsymbol{z}) = \frac{1}{b} \sum_{s=1}^{b} \mathbf{1}\{\ell_s(\boldsymbol{x}) = \ell_s(\boldsymbol{z})\}.$$

A valid kernel must have a symmetric positive semidefinite Gram matrix for every finite input set. No tree fitting or coding is required.

(a) **[4]** Construct an explicit feature map $\boldsymbol{\phi}(\boldsymbol{x})$ satisfying $k(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\phi}(\boldsymbol{z})$. State its dimension when tree s has m_s terminal leaves. Use your map to prove that k is a valid kernel.

(b) **[3]** For three inputs, use the following leaf assignments. The last column specifies the inputs for which each tree is eligible.

s	$\ell_s(\boldsymbol{x}_1)$	$\ell_s(\boldsymbol{x}_2)$	$\ell_s(\boldsymbol{x}_3)$	Eligible inputs
1	1	1	2	$\boldsymbol{x}_1, \boldsymbol{x}_2$
2	1	2	1	$\boldsymbol{x}_1, \boldsymbol{x}_3$
3	2	1	2	$\boldsymbol{x}_2, \boldsymbol{x}_3$

Calculate the 3×3 Gram matrix K using all three trees. Then let $\mathcal{E}_i$ be the set of trees eligible for $\boldsymbol{x}_i$, and calculate the matrix

$$\widetilde{K}_{ij} = \frac{\sum_{s \in \mathcal{E}_i \cap \mathcal{E}_j} \mathbf{1}\{\ell_s(\boldsymbol{x}_i) = \ell_s(\boldsymbol{x}_j)\}}{|\mathcal{E}_i \cap \mathcal{E}_j|}.$$

Every denominator is positive in this example.

(c) **[3]** Determine whether $\widetilde{K}$ is positive semidefinite. Justify your answer by any rigorous argument, such as a negative quadratic form or a negative determinant. State whether this pair-dependent averaging rule always produces a valid kernel.

Deliverables for Question 1
Crowdmark Q1: The feature map, its dimension and validity proof, both numerical matrices, and your conclusion about pair-dependent averaging with a rigorous justification. Written answers only.

Question 2: Training a CNN and testing translation behavior

30 marks

Use `CNN_train.csv` (400 images) and `CNN_test.csv` (200 images). Both files are numeric and have no headers. Rows 1–144 contain the pixels, one image per column, and row 145 contains its label. Reshape each pixel vector into an image $A \in \mathbb{R}^{12\times 12}$ in **column-major order**: the first 12 entries form image column 1. Labels are $+1$ for a vertical bar and -1 for a horizontal bar. Do not rescale the pixels. Optional Python, R, and MATLAB loaders are supplied.

Architecture. Train two networks, with stride $p = 1$ and $p = 2$. Each has one cross-correlation layer with four trainable 3×3 kernels W_c and four biases b_c, followed by ReLU, global average pooling, and a scalar linear output:

$$(H_c(A))_{j,k} = \max\left\{0,\ b_c + \sum_{u=-1}^{1}\sum_{v=-1}^{1} (W_c)_{u+2,v+2}\, A_{j+u,k+v}\right\}, \qquad c = 1, \ldots, 4.$$

Use **circular indexing** in both directions, so $A_{j+12,k} = A_{j,k+12} = A_{j,k}$. Retain centers $j, k \in \mathcal{I}_p = \{1, 1 + p, \ldots, 13 - p\}$ and define

$$g_{p,c}(A) = \frac{1}{|\mathcal{I}_p|^2} \sum_{j,k\in\mathcal{I}_p} (H_c(A))_{j,k}, \qquad f_p(A) = \sum_{c=1}^{4} v_c\, g_{p,c}(A) + b_0.$$

Thus each channel is averaged separately before the readout. A library implementation may wrap-pad one pixel on every side, then use valid cross-correlation with stride p. Do not use zero padding.

Training. Initialize each network from a separate copy of `CNN_initial.csv`. Its parameter order is specified in `README_CNN.md`. Train all kernels, biases, and readout coefficients with **300 full-batch gradient-descent updates**, step size 2, on the mean loss

$$\ell_p(\boldsymbol{\theta}) = \frac{1}{400}\sum_{i=1}^{400} \log(1 + \exp\{-y_i f_p(A_i; \boldsymbol{\theta})\})\,.$$

Update all parameters simultaneously using their gradients at the current values. Use no momentum, weight decay, augmentation, or early stopping. Set the ReLU derivative at zero to zero. Automatic differentiation is permitted. A binary-cross-entropy-with-logits routine is equivalent if its targets are $(y_i + 1)/2$ and its reduction is the mean.

(a) **[4]** State the spatial size of each retained feature map for both strides and calculate the total number of trainable parameters. Show the counts for kernels, channel biases, and readout.

How many parameters would each model have if every retained spatial position had its own four kernels and four biases, with pooling and readout unchanged? Calculate both totals; do not implement these unshared models.

(b) **[10]** Construct and train the two specified CNNs. Record ℓ_p before training and after each update. Make one figure showing both training-loss curves against the number of completed updates. Report each model's final training classification error.

Question 2: evaluation and explanation

Define the circular-shift operator S_q by shifting A by q columns to the right: $(S_q(A))_{j,k} = A_{j,k-q}$. Shift the entire image, including its noise; keep its label unchanged. Predict $+1$ when the network score is nonnegative and -1 otherwise.

(c) **[10]** Using each trained model without further updates, evaluate the test set at shifts $q = 0, 1, 2$. For every pair (p, q), calculate test classification error and

$$\delta_{p,q} = \max_{i=1,\ldots,200} \left| f_p(S_q(A_i^{\text{test}})) - f_p(A_i^{\text{test}}) \right|.$$

Report the six results in one table. Compute the maximum over all 200 test images. Use $\delta_{p,q} \leq 10^{-5}$ as the numerical criterion for unchanged scores. Single or double precision is permitted in this question; report which you used.

(d) **[6]** For arbitrary fixed network parameters, prove that $f_1(S_q(A)) = f_1(A)$ and $f_2(S_{2q}(A)) = f_2(A)$ for every integer q. Identify how the retained activation positions are permuted in each case.

Explain why a one-column shift need not preserve f_2, and relate this to your table in at most 100 words. Your argument must concern the network operations, not only the observed errors. A change in classification error is not required to demonstrate a change in scores.

Deliverables for Question 2
Crowdmark Q2: Feature-map sizes and parameter counts, one training-loss figure, both final training errors, one six-row test table, the shift proofs, and the short explanation. State your framework and numeric precision. No particular accuracy level is required.
Learn: `Q2.py`, `Q2.R`, or `Q2.m`, with inputs and any helpers. Save `Q2_results.csv`, with a header and columns `stride`, `shift`, `test_error`, `max_score_change`. Include six rows, ordered first by stride 1, 2 and then by shift 0, 1, 2. The final column contains $\delta_{p,q}$. No separate training-history CSV is required.

Question 3: Pruning and a change to one training label

10 marks

A fixed binary tree has the following structure: the root's left child is leaf 1; its right child is an internal node with leaves 2 and 3. The training class counts in these leaf regions are:

Leaf	$+1$	-1
1	20	2
2	0	12
3	4	0

Consider exactly three subtrees: $\mathcal{T}_3$ is the full tree; $\mathcal{T}_2$ merges leaves 2 and 3; and $\mathcal{T}_1$ replaces the whole tree by one leaf. Every terminal node predicts its training majority class, including after a merge; use $+1$ for a tie. Let $r(\mathcal{T})$ be the number of training misclassifications and $m(\mathcal{T})$ the number of terminal leaves. For $\lambda \geq 0$, define

$$c_\lambda(\mathcal{T}) = r(\mathcal{T}) + \lambda m(\mathcal{T}).$$

Use misclassification counts, not fractions. No tree fitting or coding is required.

(a) **[3]** For each subtree, state its terminal-node predictions and calculate $r(\mathcal{T})$ and $c_\lambda(\mathcal{T})$. Put the results in one three-row table.

(b) **[3]** Find every range of λ for which each subtree minimizes c_λ. State all ties at the boundary values.

(c) **[4]** Change one positive training label in leaf 3 to a negative label. Keep all tree splits fixed. Recompute the three training-error counts and the ranges of λ for which each subtree is optimal, including boundary ties.

For every λ at which both the original and perturbed objectives have a unique minimizer, does the perturbed optimal tree have at most as many terminal leaves as the original optimal tree? Justify your answer. If the statement is false, give one counterexample value of λ and the two corresponding leaf counts.

Deliverables for Question 3
Crowdmark Q3: The three-row table, original and perturbed optimum ranges with all boundary ties, the perturbed error counts, and your conclusion about the leaf counts with its justification. Written answers only.

Question 4: Bagging, random forests, and disagreement

25 marks

Use these numeric CSV files, which have no headers:

- `Ensemble_train.csv`: rows 1–8 form $X \in \mathbb{R}^{8\times 320}$; row 9 contains the observed labels; row 10 contains a diagnostic flag.
- `Ensemble_test.csv`: rows 1–8 form $X_{\text{test}} \in \mathbb{R}^{8\times 1000}$; row 9 contains the test labels.

Exactly 32 training labels were reversed from the underlying labeling rule. Row 10 identifies those columns with a one. Use the observed labels as supplied. Do not use the flag as a predictor, correct labels, or remove observations. The flag is used only in Question 5. Do not standardize the features.

Tree settings. Use Gini splits, at least five training observations in every leaf, and no depth limit or post-pruning. Count repeated bootstrap observations separately. Set the per-node candidate-feature count to eight for the single tree and bagging, and to two for the random forest. Randomize candidate features afresh at each node; a library's usual fallback when no valid split is found is permitted. Follow the data README's configuration guidance and report your library/version.

(a) **[12]** Choose and report one main seed. Use it to fit one tree on all 320 training columns and to initialize the main random-number generator once. Then build 100 bagging trees and 100 random-forest trees:

- For each $s = 1, \ldots, 100$, sample 320 training-column indices independently and uniformly with replacement. Fit both ensembles' tree s to this same bootstrap sample.
- Draw a new integer tree seed from the main generator for each s; you may share that seed between the two trees fitted to the paired sample. Do not reuse the same tree seed for all s. For example, in Python use `tree_seed = int(rng.integers(1, 2**31))`.
- Obtain each tree's probability of class $+1$ and predict $+1$ when it is at least $1/2$. Otherwise predict -1. In scikit-learn, use `predict_proba`, not `predict`, whose leaf-tie rule can select -1. Identify the probability column using the class labels.

For $b \in \{1, 10, 50, 100\}$, aggregate the first b fitted trees by hard voting: predict $+1$ if $\sum_{s=1}^{b} h_s(\boldsymbol{x}) \geq 0$, and -1 otherwise. Do not average leaf probabilities across trees.

(b) **[8]** Report a nine-row table of training and test errors: one row for the single tree and four rows each for bagging and the random forest. For each 100-tree ensemble, also include its mean individual-tree test error and its disagreement δ, defined on the next page. Put these two additional quantities in the same table, using `NA` in those columns for the other seven rows.

Make one two-panel figure: training error against b and test error against b. Each panel must contain both ensemble curves and the single-tree error as a horizontal reference line.

Question 4: disagreement and interpretation

For $b \geq 2$, let q_i be the number of trees predicting $+1$ at column $\boldsymbol{x}_i^{\text{test}}$ of X_{test}. Use

$$\begin{aligned}\delta &= \frac{1}{n_{\text{test}}\binom{b}{2}} \sum_{i=1}^{n_{\text{test}}} \sum_{1 \leq s < u \leq b} \mathbf{1}\{h_s(\boldsymbol{x}_i^{\text{test}}) \neq h_u(\boldsymbol{x}_i^{\text{test}})\} \\ &= \frac{\sum_{i=1}^{n_{\text{test}}} q_i(b - q_i)}{n_{\text{test}}\binom{b}{2}}, \qquad n_{\text{test}} = 1000.\end{aligned}$$

(c) **[3]** Prove the second equality by counting disagreeing unordered pairs at one test input.

(d) **[2]** In at most 150 words, describe the effect of increasing b in your results. Then compare bagging and the random forest in terms of individual-tree error, disagreement, and ensemble test error. Explain why greater disagreement alone does not guarantee lower classification error. No particular ranking or monotonic error trend is required.

Deliverables for Question 4
Crowdmark Q4: One nine-row table, one two-panel figure, the counting proof, the short interpretation, and your seed and tree settings.
Learn: `Q4.py`, `Q4.R`, or `Q4.m`, with inputs and any helpers. Save `Q4_results.csv`, with a header and columns `model`, `trees`, `train_error`, `test_error`, `mean_tree_test_error`, `disagreement`. Use model names `single`, `bagging`, and `forest`; set `trees` to 1 for the single tree. Fill the last two columns only for the two 100-tree ensembles, and use `NA` elsewhere.

Question 5: AdaBoost and the weight on altered labels

25 marks

Use the data from Question 4. No tree-fitting library is needed in this question. The numeric file `Q5_stumps.csv` has **no header** and has 3 rows and 112 columns. Each column contains a feature index j, threshold s, and polarity $p \in \{-1, +1\}$, defining

$$h(\boldsymbol{x}) = \begin{cases} p, & x_j \geq s, \\ -p, & x_j < s. \end{cases}$$

Feature indices are **one-based**; subtract one when indexing a Python array. Candidates are ordered by $j = 1, \ldots, 8$, then by increasing $s \in \{-1.5, -1, -0.5, 0, 0.5, 1, 1.5\}$, then by $p = +1, -1$. For example, columns 1, 2, and 3 are $(1, -1.5, +1)$, $(1, -1.5, -1)$, and $(1, -1, +1)$; column 15 is $(2, -1.5, +1)$.

Use double-precision arithmetic. Start with weights $u_{1,i} = 1/n$, $n = 320$, and score $f_0(\boldsymbol{x}) = 0$. In each round, evaluate every candidate's weighted training error. Find the minimum, then choose the smallest-index candidate whose error is at most that minimum plus 10^{-12}. Call the chosen stump h_t and use its **actual** weighted error

$$\epsilon_t = \sum_{i=1}^{n} u_{t,i} \mathbf{1}\{h_t(\boldsymbol{x}_i) \neq y_i\}$$

in all updates:

$$\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}, \qquad f_t(\boldsymbol{x}) = f_{t-1}(\boldsymbol{x}) + \alpha_t h_t(\boldsymbol{x}),$$

$$z_t = \sum_{i=1}^{n} u_{t,i} \exp\{-\alpha_t y_i h_t(\boldsymbol{x}_i)\}, \qquad u_{t+1,i} = \frac{u_{t,i} \exp\{-\alpha_t y_i h_t(\boldsymbol{x}_i)\}}{z_t}.$$

Predict $+1$ when $f_t(\boldsymbol{x}) \geq 0$ and -1 otherwise. Run 50 rounds. Check $0 < \epsilon_t < 1/2$ before each update; if the check fails, stop before that update and report the round and error. A stump may be selected more than once.

(a) **[10]** Implement candidate selection, the coefficient and weight updates, and score accumulation. After every completed round, check that all weights are nonnegative and their sum differs from one by at most 10^{-10}.

(b) **[10]** After every completed round, calculate training error, test error, training exponential loss, and the updated mass on flagged columns:

$$\ell_t = \frac{1}{n} \sum_{i=1}^{n} \exp\{-y_i f_t(\boldsymbol{x}_i)\}, \qquad m_{t+1} = \sum_{i \in \mathcal{F}} u_{t+1,i}.$$

Here $\mathcal{F}$ is the set of training-column indices with diagnostic flag one. Report one table at $t = 1, 5, 10, 25, 50$. Make one three-panel figure showing all completed rounds: training and test errors together, ℓ_t, and m_{t+1}. If an early stop occurs, report only completed checkpoints and its reason.

Question 5: interpreting the weights

(c) **[3]** Before an update, define

$$m_t = \sum_{i \in \mathcal{F}} u_{t,i}, \qquad \epsilon_t^{\mathcal{F}} = \frac{\sum_{i \in \mathcal{F}} u_{t,i} \mathbf{1}\{h_t(\boldsymbol{x}_i) \neq y_i\}}{m_t}.$$

Assume $m_t > 0$ and $0 < \epsilon_t < 1/2$. Express m_{t+1} in terms of $m_t, \epsilon_t^{\mathcal{F}}, \alpha_t, z_t$, and prove

$$m_{t+1} > m_t \quad \Longleftrightarrow \quad \epsilon_t^{\mathcal{F}} > \epsilon_t.$$

(d) **[2]** In at most 150 words total, use your results to answer these questions:

- Does the total flagged weight increase in every round? Relate your observation to part (c).
- Does decreasing training exponential loss require decreasing test classification error?
- How does the final AdaBoost test error compare with the two 100-tree errors from Question 4?

Deliverables for Question 5
Crowdmark Q5: One checkpoint table, one three-panel figure, the mass-update expression and proof, and the short interpretation.
Learn: `Q5.py`, `Q5.R`, or `Q5.m`, with inputs and any helpers. Save one row per completed round in `Q5_results.csv`, with a header and columns `round`, `candidate`, `weighted_error`, `alpha`, `train_error`, `test_error`, `exp_loss`, `flag_weight`. Candidate indices are 1–112; `flag_weight` contains m_{t+1}, measured **after** round t. Report any early stop in your README.