New York University Tandon School of Engineering
Computer Science and Engineering

CS-UY 6923: Midterm Exam.
Friday, Octobert 18th, 2024, 2:00 - 3:30pm
100 Total Points

Directions

- Write you name at the top of each page.
- Show all of your work to receive full (and partial) credit.
- If more space is required, use extra sheets of paper, marked with your name and the problem number.

Important Note: This exam may be challenging. You do not need to get everything correct to earn a good grade. Partial credit will be awarded generously for clear reasoning and intermediate steps.

Please read each question carefully and manage your time according to the point values indicated. Do your best — clarity and justification matter more than perfect algebra.

Problem 1. Linear Regression (18 points)

Consider multiple linear regression with a dataset (X, y) where $X \in \mathbb{R}^{n \times d}$ represents the feature matrix and $y \in \mathbb{R}^n$ represents the label vector. Let $w^* \in \mathbb{R}^d$ be the solution to the unregularized loss problem

$$w^* = \arg\min_w \frac{1}{n}\|Xw - y\|^2$$

and let $w_{\text{reg}} \in \mathbb{R}^d$ be the solution to the regularized problem

$$w_{\text{reg}} = \arg\min_w \frac{1}{n}\|Xw - y\|^2 + \lambda\|w\|^2,$$

where $\lambda > 0$ is the regularization parameter.

For each of the following statements, specify one of $>$, $\geq$, $\leq$, $<$, or N/A in the blank space, where N/A means that the relationship could go either way depending on the data. Provide a very short justification or example to explain your choice.

(a) (4 points) Train loss of w^* _____ Train loss of w_{reg}

(b) (5 points) Test loss of w^* _____ Test loss of w_{reg}

(c) (4 points) $\|w^*\|$ _____ $\|w_{\text{reg}}\|$

(d) (5 points) $\|X_i w^* - y_i\|$ _____ $\|X_i w_{\text{reg}} - y_i\|$ for an arbitrarily chosen training datapoint (X_i, y_i), where $X_i \in \mathbb{R}^d$ is the i-th row of X

Problem 1 Solution

(a) **Train loss of w^* $\leq$ Train loss of w_{reg}**
Justification: w^* minimizes the unregularized training loss, so for any w, including w_{reg}, we have

$$\frac{1}{n}\|Xw^* - y\|^2 \leq \frac{1}{n}\|Xw_{\text{reg}} - y\|^2.$$

(b) **Test loss of w^* N/A Test loss of w_{reg}**
Justification: Could go either way. Ridge regression (w_{reg}) typically has *lower* test loss when overfitting occurs, but if X is well-conditioned and data are noise-free, OLS (w^*) may have lower test loss.

(c) $\|w^*\| \geq \|w_{\text{reg}}\|$
Justification: w_{reg} minimizes the regularized loss $\frac{1}{n}\|Xw - y\|^2 + \lambda\|w\|^2$. Since w^* minimizes only the first term, adding $\lambda\|w\|^2$ shrinks the weights, implying $\|w_{\text{reg}}\| \leq \|w^*\|$.

(d) $\|X_i w^* - y_i\|$ **N/A** $\|X_i w_{\text{reg}} - y_i\|$
Justification: OLS minimizes the *sum* of squared residuals, not each individual one. For a given datapoint i, its residual may increase or decrease under ridge; direction depends on X and y.

Problem 2: Logistic Regression (22 points)

Suppose we are working on a C-class logistic regression problem with an explicit bias term. Given a dataset with feature matrix $X \in \mathbb{R}^{n\times d}$ (where n is the number of training examples and d is the number of features), labels $y \in \{1, 2, \ldots, C\}^n$, we solve the following optimization problem:

$$w^*, b^* = \underset{w\in\mathbb{R}^{d\times C}, b\in\mathbb{R}^C}{\arg\min} \sum_{i=1}^{n} -\log\left(\text{softmax}(X_i w + b)\right)_{y_i}$$

where $X_i \in \mathbb{R}^d$ is the i-th row of X, and $\text{softmax}(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}}$ for a vector $z \in \mathbb{R}^C$. Explain your answers.

(a) (4 points) How many parameters does this model have (as a function of d and C)?

(b) (6 points) Suppose we know that only a small fraction of the features are relevant to classification (but we don't know which ones), while the others are completely irrelevant. What can we do to incorporate this information into the model and avoid using the irrelevant features in our fit?

(c) (6 points) Suppose we solve the unregularized problem above using stochastic minibatch gradient descent with no momentum. Describe what hyperparameters we need to tune and a detailed procedure (full algorithm) for tuning them.

(d) (6 points) Suppose the input space is 2-dimensional ($d = 2$) and we have 2 classes ($C = 2$). Suppose our data looks like Figure 1, where the positive class forms a diamond shape centered at the origin. Would standard logistic regression work well on this problem? What would you do to make it work better?

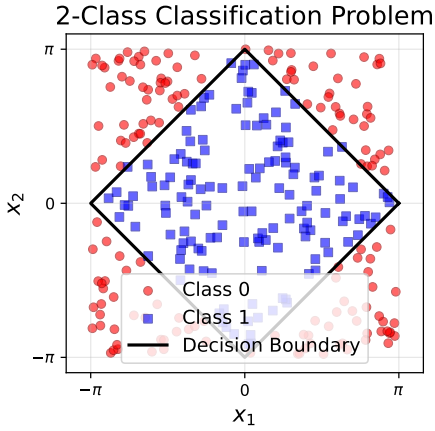

Figure 1: Training data distribution for 2-class problem

(a) **How many parameters?**

With the given parameterization (one weight vector and one bias per class),

$$\text{parameter count} = dC + C.$$

Remark on identifiability: the softmax is invariant to adding the same affine function to all class logits, so the identifiable degrees of freedom are $(d+1)(C-1)$. Many texts enforce identifiability by fixing one class's (w, b) to zero.

(b) **Few relevant features; avoid using irrelevant ones.**

Use sparsity-inducing regularization/feature selection. Two standard choices:

- ℓ_1 (lasso) on the weights to drive many coefficients to exactly zero:

$$\min_{w,b} \sum_{i=1}^{n} -\log\left(\text{softmax}(X_i w + b)\right)_{y_i} \; + \; \lambda \|w\|_1.$$

(Elastic net is also common to balance sparsity and stability.)

(c) **SGD (minibatch, no momentum): hyperparameters and a tuning procedure.**

Hyperparameters: learning rate η; batch size B; number of epochs (or other stopping criteria).

Optional: any of learning-rate schedule (e.g., step decay/cosine); initialization strategy (shouldn't matter for log reg).

Procedure (Example, doesn't need to be identical):

(a) Split training into *train/validation* (or use k-fold CV).

(b) Choose a log-spaced grid for η (e.g., $\{10^{-3}, 3{\cdot}10^{-3}, 10^{-2}, 3{\cdot}10^{-2}, 10^{-1}\}$) and a small set of B (e.g., $\{32, 64, 128\}$).

(c) For each (η, B): train for a generous epoch cap with shuffled minibatches; monitor validation loss each epoch.

(d) Use early stopping based on validation loss with patience p and restore best weights.

(e) Select (η, B) with the lowest validation loss; retrain on train+val if desired, then evaluate on the test set.

Need to describe the train validation split, going over a set of candidate values for hyper-parameters, and selecting the best parameters according to validation loss.

(d) **Will standard logistic regression work?** Not with only the original linear features: it learns a single line $\{x : w^\top x + b = 0\}$, which cannot capture an axis-aligned diamond $\{x : |x_1| + |x_2| \le r\}$, so it will underfit.

Simple fix via feature engineering. Augment the design with the nonlinear feature

$$z = |x_1| + |x_2|, \qquad X' = [\, x_1,\ x_2,\ z\,].$$

Fit logistic regression on X', yielding a logit

$$\ell(x) \;=\; w_1 x_1 + w_2 x_2 + w_3(|x_1| + |x_2|) + b.$$

With $w_1 = w_2 = 0$ and $w_3 \neq 0$, the decision boundary $\ell(x) = 0$ reduces to $|x_1| + |x_2| = r$ (for $r = -b/w_3$), exactly matching a diamond. Thus a linear classifier in the augmented space can separate the classes.

Note. If desired, one can use smooth approximations of $|x|$ (e.g., $\sqrt{x^2 + \varepsilon}$) for gradient-based optimization; subgradients at 0 also suffice.

Problem 3: Fourier Features (26 points)

Suppose we are solving a linear regression problem using Fourier features. For an input $x \in [0, 1]$, the Fourier feature map with k frequencies is defined as:

$$F_k(x) = \begin{bmatrix} 1 \\ \cos(\pi x) \\ \sin(\pi x) \\ \cos(2\pi x) \\ \sin(2\pi x) \\ \vdots \\ \cos(k\pi x) \\ \sin(k\pi x) \end{bmatrix} \in \mathbb{R}^{2k+1}$$

If you are not familiar with Fourier analysis, don't worry. The important thing for us to know is that these features form a basis: for a fixed set of distinct inputs $x_1, \ldots, x_n \in [0, 1]$, the feature matrix

$$F_k(X) = \begin{bmatrix} F_k(x_1)^T \\ F_k(x_2)^T \\ \vdots \\ F_k(x_n)^T \end{bmatrix} \in \mathbb{R}^{n \times (2k+1)}$$

will be full rank when $2k + 1 \ge n$, i.e., $\text{rank}(F_k(X)) = n$.

We solve the linear regression problem:

$$w_k^* = \underset{w \in \mathbb{R}^{2k+1}}{\arg\min} \|F_k(X)w - y\|^2$$

If there are multiple solutions that achieve the optimal loss, we take the one with the lowest norm, as always.

Suppose our data, shown in Figure 2, consists of 10 observations (x_i, y_i) where $i = 1, \ldots, 10$. The labels for each datapoint are generated from an unknown quadratic function with some added noise. Explain your answers.

(a) (8 points) Draw a sketch of the training loss $\|F_k(X)w_k^* - y\|^2$ as a function of the number of Fourier features k that we are considering. As $k \to \infty$, what happens to the training loss?

(b) (6 points) Draw a sketch of the test loss $\|F_k(X_{\text{test}})w_k^* - y_{\text{test}}\|^2$ as a function of k, where $(X_{\text{test}}, y_{\text{test}})$ is a held-out test set generated from the same underlying quadratic function with noise.

(c) (6 points) What can we say about the optimal value of k, and how would we find it in practice, assuming we have some held-out validation data?

(d) (6 points) Propose a regularization strategy that would allow us to obtain a sensible fit even if we use a very large k.

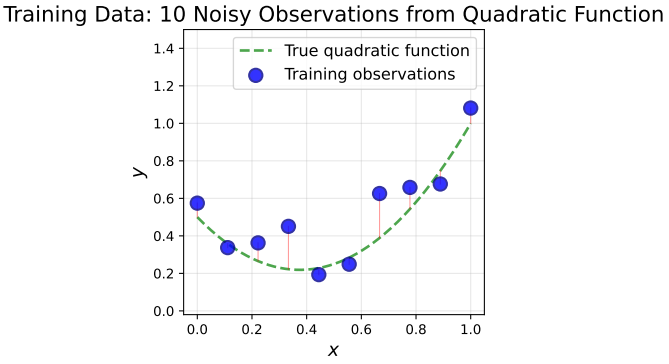

Figure 2: Training data: 10 noisy observations from an unknown quadratic function

Solution: Problem 3 (Fourier Features)

Let $n = 10$ training points. Recall $F_k(X) \in \mathbb{R}^{n\times(2k+1)}$ has full row rank n once $2k+1 \geq n$, i.e., $k \geq 5$.

(a) **Training loss as a function of k.**

- For small k, the model is underparameterized, so the optimal training loss $\|F_k(X)w_k^* - y\|^2$ *decreases monotonically* as k increases.
- At the *interpolation threshold* $2k+1 = n$ (here $k = 5$), the columns span $\mathbb{R}^n$, so there exists w with $F_k(X)w = y$; hence the minimum training loss reaches $\mathbf{0}$ for $k \geq 5$.

Sketch: A nonincreasing curve that drops and hits 0 at $k = 5$, staying at 0 for all larger k. As $k \to \infty$, the training loss remains 0.

(b) **Test loss as a function of k.**

- For small k, high bias $\Rightarrow$ larger test loss.
- As k increases, the model fits the quadratic trend better $\Rightarrow$ test loss *decreases*.
- Around and beyond the interpolation threshold ($k \gtrsim 5$), the model can fit noise, so variance dominates $\Rightarrow$ test loss typically *increases* (overfitting).

Sketch: A classic **U-shaped** validation/test curve: decreases for small k, attains a minimum at some moderate k (often $k < 5$ here), then increases for larger k due to overfitting.

Remark (optional nuance): With minimum-norm solutions and highly overparameterized features, one may observe a *double-descent* shape where the test loss can decrease again for very large k; however, a standard expected answer is the U-shape.

(c) **Optimal k and how to find it.** The optimal k balances bias and variance and is the one minimizing held-out performance. In practice:

(a) Define a grid $k \in \{0, 1, 2, \ldots, K_{\max}\}$.

(b) For each k, fit on training data and compute validation loss $\|F_k(X_{\text{val}})\hat{w}_k - y_{\text{val}}\|^2$ (or use K-fold CV).

(c) Choose $k^\star = \arg\min_k \text{ValLoss}(k)$; refit on train + val with $k^\star$ and report on the test set.

(d) **Regularization for large k.** Use penalties that discourage fitting noise:

- **Ridge/Tikhonov:**

$$\min_w \ \|F_k(X)w - y\|^2 + \lambda\|w\|_2^2.$$

- **Frequency-weighted (smoothing) penalty:** penalize high frequencies more, e.g.

$$\min_w \ \|F_k(X)w - y\|^2 + \lambda \sum_{m=1}^{k} m^2(a_m^2 + b_m^2),$$

where a_m, b_m are the cosine/sine coefficients at frequency m. (This approximates Sobolev smoothing and favors smooth, low-frequency fits.)

- **Early stopping** with gradient descent can act as an implicit ℓ_2 bias; elastic net is also reasonable if sparsity is desired.

Problem 4: Neural Nets (34 points)

Imagine we want to find a place to rent in Manhattan. Let's say we have a budget of $\$N$ per month. We will try to make a classifier that will predict whether an apartment is within our budget.

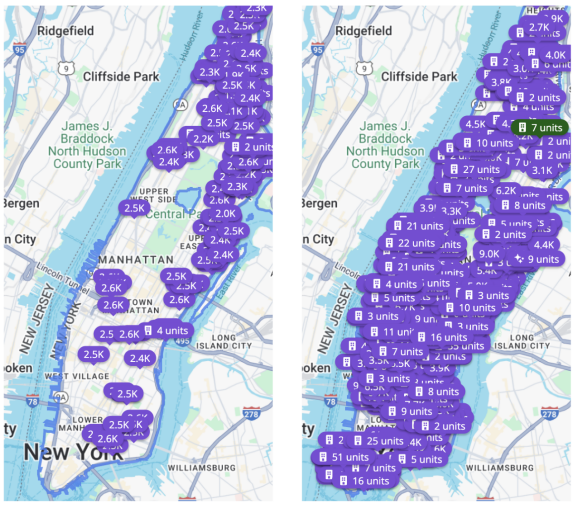

Figure 3: Screenshot of apartment listings on Zillow for Manhattan

We download a dataset of all the listings on Zillow (see Figure 3). There are currently 6000+ listings. We will describe each listing by its coordinates (x_1, x_2), where x_1, x_2 are the coordinates on the map. We will assume that no two listings have exactly the same coordinates. For each listing, we also create a label y that is 1 if the rent is within our budget and 0 otherwise. For all the subproblems below provide justification for your answers.

For reference, recall the following:

- Linear layer: $z = Wx + b$, where $W \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $z \in \mathbb{R}^m$.
- Gradient through linear layer:

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial z} x^T \in \mathbb{R}^{m \times n}$$
$$\frac{\partial L}{\partial x} = W^T \frac{\partial L}{\partial z} \in \mathbb{R}^n$$

  where $\frac{\partial L}{\partial z} \in \mathbb{R}^m$.

- ReLU activation: $h = \text{ReLU}(z) = \max(0, z)$ applied element-wise, where $z, h \in \mathbb{R}^m$.

- Gradient through ReLU:

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial h} \odot \mathbb{1}_{z>0} \in \mathbb{R}^m$$

where $\mathbb{1}_{z>0}$ is an element-wise indicator that equals 1 if $z_i > 0$ and 0 otherwise, and $\odot$ denotes element-wise multiplication.

(a) (8 points) Let's train a neural network (MLP, fully-connected) on this classification task! We will start with a single hidden layer and sigmoid activations. We will have a single output, and train with the binary cross-entropy loss. If we make this network wide enough (i.e., use enough hidden units), will we be able to fit our training data perfectly, or do we need more hidden layers? Justify your answer.

(b) (10 points) Now, the two coordinates are probably not sufficient to make a good model. Let's add another feature, f, representing the floor of the apartment. We will use *late fusion* to add the floor information to the model, as shown in Figure 4. In other words, we process f separately from (x_1, x_2) in the first hidden layer, and we combine them in the second hidden layer.

Specifically, these are the equations defining the forward pass in the model:

$$\begin{aligned}
\hat{h} &= W_{xh}x + b_h \in \mathbb{R}^3 \\
h &= \text{ReLU}(\hat{h}) \in \mathbb{R}^3 \\
\hat{u} &= W_{fu}f + b_u \in \mathbb{R}^2 \\
u &= \text{ReLU}(\hat{u}) \in \mathbb{R}^2 \\
\hat{z} &= W_{hz}h + W_{uz}u + b_z \in \mathbb{R}^3 \\
z &= \text{ReLU}(\hat{z}) \in \mathbb{R}^3 \\
y &= \sigma(W_{zy}z + b_y) \in \mathbb{R}
\end{aligned}$$

where $x = [x_1, x_2]^T$, $h = [h_1, h_2, h_3]^T$, $u = [u_1, u_2]^T$, $z = [z_1, z_2, z_3]^T$, and σ is the sigmoid function.

Suppose we have already computed $\frac{\partial L}{\partial z} \in \mathbb{R}^3$. Please express $\frac{\partial L}{\partial W_{fu}}$, where $W_{fu} \in \mathbb{R}^{2\times 1}$ is the weight matrix in the first linear layer transforming f to u.

(c) (6 points) Does the gradient $\frac{\partial L}{\partial W_{fu}}$ that you computed depend on the values of x_1, x_2 in any way? If so, how? Explain your reasoning.

(d) (10 points) The model from part (b) is a special case of a standard MLP model where all inputs x_1, x_2, f are connected to all the hidden units h_1, h_2, h_3, u_1, u_2, but where we set some of the weights to zero. In practice, it may be more convenient to implement a standard MLP and do the masking on the weights.

Write down the formulas for the forward pass in this MLP with masked weights, analogously to what we did for the late fusion model. Use $\odot$ for the element-wise product. Specifically, express the computation from the input layer $[x_1, x_2, f]^T$ to the first hidden layer $[h_1, h_2, h_3, u_1, u_2]^T$ using a weight matrix W_1 and a binary mask matrix M such that certain weights are forced to be zero.

Solution: Manhattan budget classifier (Neural Nets)

(a) **Single hidden layer suffices (with enough width).** Because the dataset is finite and all coordinates are distinct, a one-hidden-layer MLP with a non-polynomial activation (e.g., sigmoid) can interpolate *any* labeling exactly. Intuition: construct narrow "bumps" that activate only near selected points (or small rectangles) and sum them with appropriate output weights; universal approximation guarantees let a single hidden layer approximate such indicator-like functions arbitrarily well, so training error can be driven to zero by using enough hidden units.

(b) **Compute $\frac{\partial L}{\partial W_{fu}}$.** Recall the forward:

$$\hat{u} = W_{fu}f + b_u, \quad u = \text{ReLU}(\hat{u}), \qquad \hat{z} = W_{hz}h + W_{uz}u + b_z, \quad z = \text{ReLU}(\hat{z}).$$

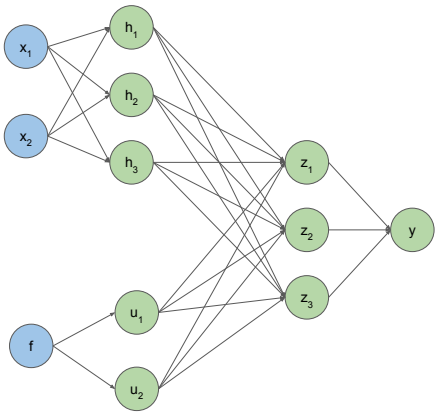

Figure 4: Late fusion architecture for apartment price prediction

Backpropagate from the provided $\frac{\partial L}{\partial z} \in \mathbb{R}^3$:

$$\frac{\partial L}{\partial \hat{z}} = \frac{\partial L}{\partial z} \odot \mathbb{1}_{\hat{z}>0} \in \mathbb{R}^3, \qquad \frac{\partial L}{\partial u} = W_{uz}^\top \frac{\partial L}{\partial \hat{z}} \in \mathbb{R}^2,$$

$$\frac{\partial L}{\partial \hat{u}} = \frac{\partial L}{\partial u} \odot \mathbb{1}_{\hat{u}>0} \in \mathbb{R}^2.$$

Since $\hat{u} = W_{fu} f + b_u$ with $W_{fu} \in \mathbb{R}^{2\times 1}$ and scalar f,

$$\boxed{\frac{\partial L}{\partial W_{fu}} = \left(\frac{\partial L}{\partial \hat{u}}\right) f^\top = \left(\left(W_{uz}^\top \left(\tfrac{\partial L}{\partial z} \odot \mathbb{1}_{\hat{z}>0}\right)\right) \odot \mathbb{1}_{\hat{u}>0}\right) f} \quad \in \mathbb{R}^{2\times 1}.$$

(c) **Dependence on (x_1, x_2).** Yes, $\frac{\partial L}{\partial W_{fu}}$ depends on (x_1, x_2) *indirectly* through (i) $\frac{\partial L}{\partial z}$, which arises from the forward pass and thus depends on the current prediction (which uses $h = h(x)$ as well as u), and (ii) the ReLU gate $\mathbb{1}_{\hat{z}>0}$, since $\hat{z} = W_{hz} h(x) + W_{uz} u + b_z$ includes $h(x)$. There is no explicit multiplicative factor of x in the local Jacobian for W_{fu}, but the upstream signals and gates depend on x.

(d) **Masked-weight MLP (equivalent to late fusion in the first layer).** Let the input be $v = [x_1, x_2, f]^\top \in \mathbb{R}^3$ and the first hidden layer be $s = [h_1, h_2, h_3, u_1, u_2]^\top \in \mathbb{R}^5$. Use a dense weight $W_1 \in \mathbb{R}^{5\times 3}$, bias $b_1 \in \mathbb{R}^5$, and a binary mask $M \in \{0,1\}^{5\times 3}$ that enforces late fusion connections:

$$M = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Then the masked first layer is

$$\hat{s} = (W_1 \odot M)\, v + b_1, \qquad s = \mathrm{ReLU}(\hat{s}),$$

with $h = s_{1:3}$ and $u = s_{4:5}$. (Subsequent layers proceed as usual without masks or with additional masks if desired.)