Homework #4, EECS 553, F26. Due **Wed. Oct. 07**, by 11:00PM

Notes

- This is the last HW that is relevant to Exam1 on Oct. 12.

Problems

1. [19] **QDA binary plug-in classifier**

The QDA lecture slides mention that for binary classification with labels ± 1, the plug-in Bayes classifier becomes

$$f(\boldsymbol{x}) = \operatorname{sign}\Big(\tilde{f}(\boldsymbol{x})\Big), \quad \tilde{f}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^\top \boldsymbol{x} + c,$$

for certain matrix $\boldsymbol{A}$, vector $\boldsymbol{b}$, scalar c.

(a) [3] Determine the expressions for $\boldsymbol{A}$, $\boldsymbol{b}$ and c in terms of the assumed quantities in QDA.
Hint: $\underset{k \in \{+1,-1\}}{\arg\max}\ \eta_k = \operatorname{sign}(\eta_{+1} - \eta_{-1})$.

(b) [10] Write a JULIA or Python function that performs plug-in QDA binary classification given matrices of training examples from each of the two classes. You may assume that the training data is sufficiently large and diverse enough that the sample covariances are invertible.
Your submitted file should be named `qda2.jl` or `qda2.py` and should contain the following function:

```
"""
    ytest = qda2(Xp, Xn, Xtest)

QDA plug-in binary classifier.

In:
- `Xp` `d x n1` matrix of training data for class +1
- `Xn` `d x n2` matrix of training data for class -1
- `Xtest` `d x m` matrix of test data vectors

Out:
- `ytest` length-`m` vector of predicted labels (±1)
"""
function qda2(
    Xp::AbstractMatrix,
    Xn::AbstractMatrix,
    Xtest::AbstractMatrix,
)
```

Submit your solution to autograder.io
Python version:

```
def qda2(Xp: np.ndarray, Xn: np.ndarray, Xtest: np.ndarray) -> np.ndarray:
```

To prepare for the histogram in part (d), you should first write a helper function `qda2_discriminant` that computes $\tilde{f}(\boldsymbol{x})$, and then use that with sign for the final classification.
In the extremely unlikely event that $\tilde{f}(\boldsymbol{x})$ is exactly 0, it is fine for your classifier to return 0 instead of ± 1 because in that case it could be either class.

(c) [3] Write test cases for your function *before* submitting to the autograder. At least one test case must correspond to a situation where you know analytically that $\boldsymbol{A} = \boldsymbol{0}$ and that the training data is classified perfectly.

(d) [3] Apply your QDA plug-in binary classifier to the data in the QDA demo. If your classifier is working correctly, you should get the same classification errors as the (less efficient for $K = 2$) "arg max" approach used there.
Make an overlapping colored histogram plot (red for digit 7 and blue for digit 1) of $\tilde{f}(\boldsymbol{x})$ for the test data. Your histogram plot should look something like the one in the LDA demo. Submit a screen shot of your plot to gradescope.

2. [6] **Weight decay in gradient descent**

For a smooth cost function $J(\boldsymbol{\theta})$, conventional gradient descent uses the iteration

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \nabla J(\boldsymbol{\theta}_t)$$

for some suitable step size $\alpha > 0$.

Many deep learning methods are trained using **weight decay**, where the network parameters (or some subset of them) are shrunk towards zero after each iteration. Mathematically, this process can be expressed for some $\varepsilon \in (0, 1)$ as:

$$\begin{aligned} \tilde{\boldsymbol{\theta}}_t &= \boldsymbol{\theta}_t - \alpha \nabla J(\boldsymbol{\theta}_t) && \text{GD update} \\ \boldsymbol{\theta}_{t+1} &= (1 - \varepsilon)\, \tilde{\boldsymbol{\theta}}_t && \text{weight decay.} \end{aligned}$$

(a) [3] For suitable values of α, this iteration will have **fixed points** that correspond to **stationary points** of some cost function $K(\boldsymbol{\theta})$. Determine $K(\boldsymbol{\theta})$.
Hint. The K depends on J and ε and a simple regularizer.

(b) [3] Suppose J is convex and L-smooth. Specify the set of values of the step-size α for which this weight decay iteration converges.

(c) [0] (Optional.) Suppose J is a **convex** function that is twice differentiable, and that L_J is the best (smallest) Lipschitz constant for ∇J. Determine the best Lipschitz constant for ∇K. (This question is important for having the widest possible range of step sizes.)

3. [15] **Logistic regression cost function**

Consider the regularized logistic regression cost function (penalized negative log-likelihood):

$$J(\boldsymbol{\theta}) = -\frac{1}{n} \ell(\boldsymbol{\theta}) + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 ,$$

where for labels $y_i \in \{0, 1\}$ the log-likelihood is:

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left[y_i \log\left(\frac{1}{1 + \mathrm{e}^{-\boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_i}} \right) + (1 - y_i) \log\left(\frac{\mathrm{e}^{-\boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_i}}{1 + \mathrm{e}^{-\boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_i}} \right) \right].$$

Here $\boldsymbol{\theta} = [b \; w_1 \; \cdots \; w_d]^\top$ and $\bar{\boldsymbol{x}}_i = [1 \; x_{i1} \; \cdots \; x_{id}]^\top$.

(a) [3] Show that if we change the label convention in logistic regression from $y \in \{0, 1\}$ to $y \in \{-1, +1\}$, then

$$-\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log\left(1 + \mathrm{e}^{-y_i \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_i} \right). \tag{1}$$

Define $\phi(t) = \log(1 + \exp(-t))$ so that, by the previous part, the logistic regression regularized negative log-likelihood may be written in an ERM style:

$$J(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \phi(y_i \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_i) + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 . \tag{2}$$

(b) [3] Calculate the gradient of $J(\boldsymbol{\theta})$ using (2). Your answer may be in the form of a summation. Simplify your result so that it involves only the full vectors $\boldsymbol{x}_i$ (or $\bar{\boldsymbol{x}}_i$) and not their elements.

(c) [3] Determine the Hessian of $J(\boldsymbol{\theta})$. This will be a $(d + 1) \times (d + 1)$ matrix. Again your result may be in the form of a summation. Simplify your result so that it involves only the full vectors $\boldsymbol{x}_i$ (or $\bar{\boldsymbol{x}}_i$) and not their elements.

(d) [3] Use the previous result to argue that J is a convex function of $\boldsymbol{\theta}$ when $\lambda \geq 0$, and strictly convex when $\lambda > 0$.

(e) [3] Find an upper bound on the Hessian of the form $\lambda \boldsymbol{I} + \frac{1}{n} d_{\max} \bar{\boldsymbol{X}}^\top \bar{\boldsymbol{X}}$ where $d_{\max}$ is a small constant you must determine.

4. [12] **Logistic regression: handwritten digit classification errrors**

The logistic regression demo uses $K = 2$ PCA features to classify the digits 1 and 7.

Download the demo code (or the corresponding python template) and change in two ways:
- Use digits 4 and 9 (which are more challenging than digits 1 and 7)
- Use $K = 8$ (because $K = 2$ is insufficient for digits 4 and 9)

Generate a figure displaying 20 images in a 4 x 5 array. These images should be the 20 misclassified images (10 for digit 4 and 10 for digit 9) for which the demo's logistic regression classifier was *most confident* about its (erroneous) prediction.
You will have to define a notion of confidence in a reasonable way – explain what this is.
Your images should somehow indicate the true labels.
You should expect to see some 4's that look like somewhat like 9's and vice versa.

5. [5] **Make your own problem**

Make a clearly stated problem that could be used on an EECS 553 exam based on the topics covered by the due date of this HW set, and provide your own solution to the problem.

Follow all the instructions from HW2 except create your problem in `hw4.tex` using the Overleaf link on EECS 553 Canvas home page.

Non-graded problems

6. [0] **Regression via MLE**

The course notes derive linear regression from an ERM perspective. An alternative perspective is maximum likelihood estimation (MLE) under a certain probability model.

(a) We are given training data $\{(\boldsymbol{x}_i, y_i)\}$ for $i = 1, \dots, n$. We focus on "linear" regression where $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$. Assume the following conditional model

$$\mathsf{Y}_i \mid \mathbf{X} = \boldsymbol{x}_i \sim \mathsf{N}\left(\boldsymbol{w}^\top \boldsymbol{x}_i + b, \sigma^2\right).$$

Assume that the responses $\{\mathsf{Y}_i\}$ are statistically independent. Determine the ML estimate of $\boldsymbol{\theta} = (b, \boldsymbol{w})$ from $\{(\boldsymbol{x}_i, y_i)\}$.
Hint. The solution should be quite familiar.

(b) Is the MLE unique? Discuss briefly.

(c) Is this a generative approach? Discuss briefly.

7. [0] **Maximum-likelihood estimation: Poisson data**

Consider a random vector $\mathbf{X}$ have independently distributed elements where $\mathsf{X}_i \sim \mathsf{Poisson}\{a_i \theta\}$ where $a_i > 0$ is known, for $i = 1, \dots, n$. Determine the ML estimate of θ from a realization $\mathbf{X} = \boldsymbol{x}$.

8. [0] **Unconstrained optimization and convexity**

(a) Use the definition of convexity to prove that any sum of two convex functions is convex.

(b) Consider the function $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^\top \boldsymbol{x} + c$, where $\boldsymbol{A}$ is a symmetric $d \times d$ matrix. Derive the Hessian of f.

- Under what conditions on $\boldsymbol{A}$ is f a convex function?
- When is f a strictly convex function?

(c) Show that if f is strictly convex, then f has at most one global minimizer.
Hint. Use a proof by contradiction.

(d) Show that $f(\boldsymbol{X}) = \mathsf{trace}\left(\boldsymbol{X}^{-1}\right)$ is a convex function on the set $\mathcal{X}$ of **positive definite** matrices.
Hint. A function $f(\boldsymbol{x})$ is convex over a domain $\mathrm{dom}(f)$ if and only if the function $g(t) = f(\boldsymbol{x} + t\boldsymbol{v})$, having domain $\mathrm{dom}(g) = \{t : \boldsymbol{x} + t\boldsymbol{v} \in \mathrm{dom}(f)\}$ is convex in t for every $\boldsymbol{v} \in \mathrm{dom}(f)$.

9. [0] A function $f : \mathbb{R}^N \mapsto \mathbb{R}^M$ is **Lipschitz continuous** with respect to norms $\|\cdot\|_X$ and $\|\cdot\|_Y$ iff there exists $L < \infty$ such that

$$\|f(\boldsymbol{x}) - f(\boldsymbol{z})\|_Y \le L \|\boldsymbol{x} - \boldsymbol{z}\|_X, \quad \forall \boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^N.$$

The usual choice of norm is the **Euclidean norm** (when otherwise unspecified).
The *best* Lipschitz constant is the smallest such value $L_* = \sup_{\boldsymbol{x} \neq \boldsymbol{y}} \frac{\|f(\boldsymbol{x}) - f(\boldsymbol{z})\|_Y}{\|\boldsymbol{x} - \boldsymbol{z}\|_X}$.

(a) Suppose that $f(\boldsymbol{x})$ is Lipschitz continuous with **Lipschitz constant** L_f and $g(\boldsymbol{x})$ is Lipschitz continuous with Lipschitz constant L_g. Find a Lipschitz constant for the function $h(\boldsymbol{x}) = f(\boldsymbol{x}) + \beta g(\boldsymbol{x})$.
For this part, allow β to be any real number (positive or negative).

(b) If L_f and L_g are the best Lipschitz constants for f and g, respectively, then is your preceding answer always the best Lipschitz constant for h? Explain why or why not.

(c) Consider the regularized least-squares cost function $\Psi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \frac{1}{2}\|\boldsymbol{T}\boldsymbol{x}\|_2^2$, where $\beta > 0$. Find a Lipschitz constant for the gradient of $\Psi(\boldsymbol{x})$, expressed in terms of the **singular values** of $\boldsymbol{A}$ and $\boldsymbol{T}$.
For this part, use the usual Euclidean norms.