Homework 2 – Machine Learning (CS453X, Whitehill, Spring 2019)

You may complete this homework assignment either individually or in teams up to 2 people.

1. **Age regression**: Train an age regressor that analyzes a $(48 \times 48 = 2304)$-pixel grayscale face image and outputs a real number $\hat{y}$ that estimates how old the person is (in years). Your regressor should be implemented using linear regression. The training and testing data are available here:

   - `https://s3.amazonaws.com/jrwprojects/age_regression_Xtr.npy`
   - `https://s3.amazonaws.com/jrwprojects/age_regression_ytr.npy`
   - `https://s3.amazonaws.com/jrwprojects/age_regression_Xte.npy`
   - `https://s3.amazonaws.com/jrwprojects/age_regression_yte.npy`

   Note: you must complete this problem using only linear algebraic operations in `numpy` – you may **not** use any off-the-shelf linear regression software, as that would defeat the purpose.

   (a) **One-shot (analytical) solution [20 points]**: Compute the optimal weights $\mathbf{w} = (w_1, \ldots, w_{2304})$ and bias term b for a linear regression model by deriving the expression for the gradient of the cost function w.r.t. $\mathbf{w}$ and b, setting it to 0, and then solving. The cost function is

   $$f_{\text{MSE}}(\mathbf{w}, b) = \frac{1}{2n} \sum_{i=1}^{n} (\hat{y}^{(i)} - y^{(i)})^2$$

   where $\hat{y} = g(\mathbf{x}; \mathbf{w}, b) = \mathbf{x}^\top \mathbf{w} + b$ and n is the number of examples in the training set $\mathcal{D}_{\text{tr}} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}$, each $\mathbf{x}^{(i)} \in \mathbb{R}^{2304}$ and each $y^{(i)} \in \{0, 1\}$. After optimizing $\mathbf{w}$ and b only on the **training set**, compute and report the cost f_{MSE} on the training set $\mathcal{D}_{\text{tr}}$ and (separately) on the testing set $\mathcal{D}_{\text{te}}$. **Suggestion**: to solve for $\mathbf{w}$ and b simultaneously, use the trick shown in class whereby each image (represented as a vector $\mathbf{x}$) is appended with a constant 1 term (to yield an appended representation $\tilde{\mathbf{x}}$). Then compute the optimal $\tilde{\mathbf{w}}$ (comprising the original $\mathbf{w}$ and an appended b term) using the closed formula:

   $$\tilde{\mathbf{w}} = \left(\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top\right)^{-1} \tilde{\mathbf{X}}\mathbf{y}$$

   For appending, you might find the functions `np.hstack`, `np.vstack`, `np.atleast_2d` useful. After optimizing $\tilde{\mathbf{w}}$ and b (using $\tilde{f}_{\text{MSE}}$), compute and report the cost f_{MSE} on the training set $\mathcal{D}_{\text{tr}}$ and (separately) the testing set $\mathcal{D}_{\text{te}}$.

   (b) **Gradient descent [25 points]**: Pick a random starting value for $\mathbf{w} \in \mathbb{R}^{2304}$ and $b \in \mathbb{R}$ and a small learning rate (e.g., $\epsilon = .001$). (In my code, I sampled each component of $\mathbf{w}$ and b from a Normal distribution with standard deviation 0.01; use `np.random.randn`). Then, using the expression for the gradient of the cost function, iteratively update $\mathbf{w}, b$ to reduce the cost $f_{\text{MSE}}(\mathbf{w}, b)$. Stop after conducting T gradient descent iterations (I suggest $T = 5000$ with a step size (aka learning rate) of $\epsilon = 0.003$). After optimizing $\mathbf{w}$ and b only on the **training set**, compute and report the cost f_{MSE} on the training set $\mathcal{D}_{\text{tr}}$ and (separately) on the testing set $\mathcal{D}_{\text{te}}$. After optimizing $\mathbf{w}$ and b (using $\tilde{f}_{\text{MSE}}$), compute and report the cost f_{MSE} on the training set $\mathcal{D}_{\text{tr}}$ and (separately) the testing set $\mathcal{D}_{\text{te}}$.

   (c) **Regularization [15 points]**: Same as (b) above, but change the cost function to include a penalty for $|\mathbf{w}|^2$ growing too large:

   $$\tilde{f}_{\text{MSE}}(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^{n} (\hat{y}^{(i)} - y^{(i)})^2 + \frac{\alpha}{2n} \mathbf{w}^\top \mathbf{w}$$

where $\alpha \in \mathbb{R}^+$. Set $\alpha = 1.0$ (this worked well for me) and then optimize $\tilde{f}_{\text{MSE}}$ w.r.t. $\mathbf{w}$ and b. After optimizing $\mathbf{w}$ and b (using $\tilde{f}_{\text{MSE}}$), compute and report the cost f_{MSE} (*without* the L_2 term) on the training set $\mathcal{D}_{\text{tr}}$ and (separately) the testing set $\mathcal{D}_{\text{te}}$. **Important**: the regularization should be applied *only* to the $\mathbf{w}$, *not* the b. I suggest a regularization strength of $\alpha = 0.1$.

(e) **Visualizing the machine's behavior [10 points]**: After training the regressors in parts (a), (b), and (c), create a 48×48 image representing the learned weights $\mathbf{w}$ (without the b term) from each of the different training methods. Use `plt.imshow()`. How are the weight vectors from the different methods different? Next, using the regressor in part (c), predict the ages of all the images in the test set and report the RMSE (in years). Then, show the top 5 **most egregious errors**, i.e., the test images whose ground-truth label y is *farthest* from your machine's estimate $\hat{y}$. Include the images, along with associated y and $\hat{y}$ values, in a PDF. 4

Submission: Put your solution in a Python file called `homework2_WPIUSERNAME.py` (or `homework2_WPIUSERNAME1_WPIUSERNAME2.py` for teams), and show the most egregious errors for part (a) in `homework2_errors_WPIUSERNAME.pdf`.