TÉCNICO LISBOA

Aprendizagem 2023/24

Homework III

Deadline 20/10/2023 (Friday) 23:59 via Fenix as PDF

I. Pen-and-paper [12v]

1. Consider the problem of learning a regression model from 4 bivariate observations $\left\{\begin{pmatrix}0.7\\-0.3\end{pmatrix},\begin{pmatrix}0.4\\0.5\end{pmatrix},\begin{pmatrix}-0.2\\0.8\end{pmatrix},\begin{pmatrix}-0.4\\0.3\end{pmatrix}\right\}$ with targets $(0.8, 0.6, 0.3, 0.3)$.

   a. [4v] Given the radial basis function, $\phi_j(x) = exp\left(-\frac{\|\mathbf{x}-c_j\|^2}{2}\right)$, that transforms the original space onto a new space characterized by the similarity of the original observations to the following data points, $\left\{\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}1\\-1\end{pmatrix},\begin{pmatrix}-1\\1\end{pmatrix}\right\}$.

   Learn the Ridge regression (l_2 regularization) using the closed solution with $\lambda = 0.1$.

   Hint: use `numpy` matrix operations (e.g., `linalg.pinv` for inverse) to validated your calculus.

First, let us represent the design matrix Φ by computing the RBF of each observation to the 3 points

$$\Phi = \begin{pmatrix}1 & 0.7483 & 0.7483 & 0.1013\\ 1 & 0.8146 & 0.2712 & 0.3312\\ 1 & 0.7118 & 0.0963 & 0.7118\\ 1 & 0.8825 & 0.1612 & 0.6538\end{pmatrix}$$

Second, using the targets and the regularization term ($\lambda = 0.1$), the regression can be learnt using:

$$\mathbf{w} = (\Phi^T \cdot \Phi + \lambda \cdot I)^{-1} \cdot \Phi^T \cdot \mathbf{t}$$

Applying the above formula, we can finally represent the polynomial regression:

$$y(\mathbf{x}) = 0.339 + 0.199\phi_1 + 0.401\phi_2 - 0.296\phi_3$$

   b. [2v] Compute the training RMSE for the learnt regression.

$$\hat{\boldsymbol{t}} = \Phi\mathbf{w} = \begin{pmatrix}0.758\\0.512\\0.309\\0.386\end{pmatrix}, \text{RMSE} = 0.065$$

2. [6v] Consider a MLP classifier of three outcomes – A, B and C – characterized by the weights

$$\mathrm{W}^{[1]} = \begin{pmatrix}1 & 1 & 1 & 1\\ 1 & 1 & 2 & 1\\ 1 & 1 & 1 & 1\end{pmatrix}, \mathrm{b}^{[1]} = \begin{pmatrix}1\\1\\1\end{pmatrix}, \mathrm{W}^{[2]} = \begin{pmatrix}1 & 4 & 1\\ 1 & 1 & 1\end{pmatrix}, \mathrm{b}^{[2]} = \begin{pmatrix}1\\1\end{pmatrix}, \mathrm{W}^{[3]} = \begin{pmatrix}1 & 1\\ 3 & 1\\ 1 & 1\end{pmatrix}, \mathrm{b}^{[3]} = \begin{pmatrix}1\\1\\1\end{pmatrix},$$

the activation $f(x) = \frac{e^{0.5x-2} - e^{-0.5x+2}}{e^{0.5x-2} + e^{-0.5x+2}} = tanh(0.5x - 2)$ for every unit, and squared error loss $\frac{1}{2}\|\mathbf{z} - \hat{\mathbf{z}}\|_2^2$. Perform one batch gradient descent update (with learning rate $\eta = 0.1$) for observations $\mathbf{x}_1 = \begin{pmatrix}1\\1\\1\\1\end{pmatrix}$ and $\mathbf{x}_2 = \begin{pmatrix}1\\0\\0\\-1\end{pmatrix}$ with targets B and A, respectively.

Derivative of the given tanh function:

$$\frac{\partial f(x)}{\partial x} = \frac{\partial}{\partial x} tanh(0.5x - 2) = \left(1 - tanh^2(0.5x - 2)\right)\frac{\partial}{\partial x}(0.5x - 2) = 0.5 \times \left(1 - tanh^2(0.5x - 2)\right)$$

$$\frac{\partial \mathbf{x}^{[i]}(\mathbf{z}^{[i]})}{\partial \mathbf{z}^{[i]}} = 0.5\left(1 - tanh^2(0.5x - 2)\right) = 0.5\left(1 - \left(\mathbf{x}^{[3]}\right)^2\right)$$

TÉCNICO LISBOA

Aprendizagem 2023/24

Homework III

Deadline 20/10/2023 (Friday) 23:59 via Fenix as PDF

Update rule of the last layer:

$$\frac{\partial E(\mathbf{x}^{[3]},\mathbf{t})}{\partial \mathbf{x}^{[3]}} = \frac{1}{2}\left(2\mathbf{x}^{[3]} - 2\mathbf{t}\right) = \mathbf{x}^{[3]} - \mathbf{t}$$

Useful derivatives:

$$\frac{\partial \mathbf{z}^{[i]}(\mathbf{W}^{[i]},\mathbf{b}^{[i]},\mathbf{x}^{[i-1]})}{\partial \mathbf{W}^{[i]}} = \mathbf{x}^{[i-1]^T}, \quad \frac{\partial \mathbf{z}^{[i]}(\mathbf{W}^{[i]},\mathbf{b}^{[i]},\mathbf{x}^{[i-1]})}{\partial \mathbf{b}^{[i]}} = \mathbf{1}, \quad \frac{\partial \mathbf{z}^{[i]}(\mathbf{W}^{[i]},\mathbf{b}^{[i]},\mathbf{x}^{[i-1]})}{\partial \mathbf{x}^{[i-1]}} = \mathbf{W}^{[i]}$$

Weight updates:

$$\delta^{[3]} = \frac{\partial E}{\partial \mathbf{x}^{[3]}} \circ \frac{\partial \mathbf{x}^{[3]}}{\partial \mathbf{z}^{[3]}} = \left(\mathbf{x}^{[3]} - \mathbf{t}\right) \circ 0.5\left(1 - \left(\mathbf{x}^{[3]}\right)^2\right)$$

$$\delta^{[2]} = \left(\frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{x}^{[2]}}\right)^T \cdot \delta^{[3]} \circ \frac{\partial \mathbf{x}^{[2]}}{\partial \mathbf{z}^{[2]}} = \mathbf{W}^{[3]^T} \cdot \delta^{[3]} \circ 0.5\left(1 - \left(\mathbf{x}^{[2]}\right)^2\right)$$

$$\delta^{[1]} = \left(\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{x}^{[1]}}\right)^T \cdot \delta^{[2]} \circ \frac{\partial \mathbf{x}^{[1]}}{\partial \mathbf{z}^{[1]}} = \mathbf{W}^{[2]^T} \cdot \delta^{[2]} \circ 0.5\left(1 - \left(\mathbf{x}^{[1]}\right)^2\right)$$

Forward propagation:

$$\mathbf{x}^{[1](1)} = \begin{pmatrix} 0.46212 \\ 0.76159 \\ 0.46212 \end{pmatrix}, \quad \mathbf{x}^{[2](1)} = \begin{pmatrix} 0.45048 \\ -0.57642 \end{pmatrix}, \quad \mathbf{x}^{[3](1)} = \begin{pmatrix} -0.9159 \\ -0.80494 \\ -0.9159 \end{pmatrix}$$

$$\mathbf{x}^{[1](2)} = \begin{pmatrix} -0.90515 \\ -0.90515 \\ -0.90515 \end{pmatrix}, \quad \mathbf{x}^{[2](2)} = \begin{pmatrix} -0.99956 \\ -0.99343 \end{pmatrix}, \quad \mathbf{x}^{[3](2)} = \begin{pmatrix} -0.98652 \\ -0.99816 \\ -0.98652 \end{pmatrix}$$

Delta computations:

$$\delta^{[3](1)} = \begin{pmatrix} 0.00678 \\ -0.31773 \\ 0.00678 \end{pmatrix}, \quad \delta^{[2](1)} = \begin{pmatrix} -0.37448 \\ -0.10156 \end{pmatrix}, \quad \delta^{[1](1)} = \begin{pmatrix} -0.18719 \\ -0.33587 \\ -0.18719 \end{pmatrix}$$

$$\delta^{[3](2)} = \begin{pmatrix} -0.06733 \\ 0. \\ 0.00018 \end{pmatrix}, \quad \delta^{[2](2)} = \begin{pmatrix} -1.0e-5 \\ -1.7e-4 \end{pmatrix}, \quad \delta^{[1](2)} = \begin{pmatrix} -2.e-5 \\ -2.e-5 \\ -2.e-5 \end{pmatrix}$$

Contributions:

$$\frac{\partial E}{\partial \mathbf{W}^{[1]}} = \delta^{[1](1)}\left(\mathbf{x}^{[0](1)}\right)^T + \delta^{[1](2)}\left(\mathbf{x}^{[0](2)}\right)^T = \begin{pmatrix} -0.18721 & -0.18719 & -0.18719 & -0.18717 \\ -0.33589 & -0.33587 & -0.33587 & -0.33585 \\ -0.18721 & -0.18719 & -0.18719 & -0.18717 \end{pmatrix}$$

$$\frac{\partial E}{\partial \mathbf{W}^{[2]}} = \begin{pmatrix} -0.17304 & -0.28519 & -0.17304 \\ -0.04678 & -0.07719 & -0.04678 \end{pmatrix}, \quad \frac{\partial E}{\partial \mathbf{W}^{[3]}} = \begin{pmatrix} 0.02964 & 0.02252 \\ -0.14314 & 0.18315 \\ 0.00287 & -0.00408 \end{pmatrix}$$

Weight updates:

$$\mathbf{W}^{[1]} = \mathbf{W}^{[1]} - \eta \frac{\partial E}{\partial \mathbf{W}^{[1]}} = \begin{pmatrix} 1.01872 & 1.01872 & 1.01872 & 1.01872 \\ 1.03359 & 1.03359 & 2.03359 & 1.03359 \\ 1.01872 & 1.01872 & 1.01872 & 1.01872 \end{pmatrix},$$

$$\mathbf{b}^{[1]} = \mathbf{b}^{[1]} - \eta \frac{\partial E}{\partial \mathbf{b}^{[1]}} = \delta^{[1]} \frac{\partial \mathbf{z}^{[1]^T}}{\partial \mathbf{b}^{[1]}} = \delta^{[1]} = \begin{pmatrix} 1.01872 \\ 1.03359 \\ 1.01872 \end{pmatrix}$$

$$\mathbf{W}^{[2]} = \begin{pmatrix} 1.0173 & 4.02852 & 1.0173 \\ 1.00468 & 1.00772 & 1.00468 \end{pmatrix}, \quad \mathbf{b}^{[2]} = \begin{pmatrix} 1.03745 \\ 1.01017 \end{pmatrix}$$

$$\mathbf{W}^{[3]} = \begin{pmatrix} 0.99704 & 0.99775 \\ 3.01431 & 0.98169 \\ 0.99971 & 1.00041 \end{pmatrix}, \quad \mathbf{b}^{[3]} = \begin{pmatrix} 1.00198 \\ 1.03177 \\ 0.9993 \end{pmatrix}$$

Aprendizagem 2023/24

Homework III

Deadline 20/10/2023 (Friday) 23:59 via Fenix as PDF

II. Programming and critical analysis [8v]

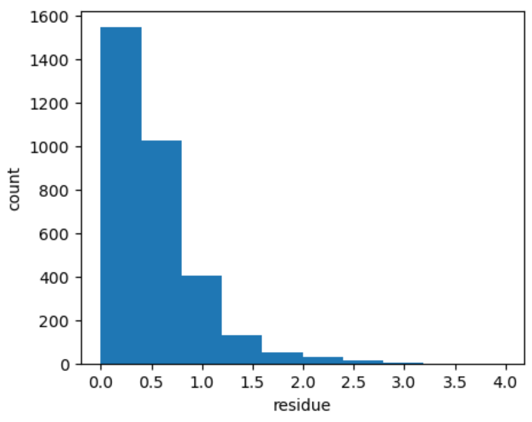
Consider the `winequality-red.csv` dataset (available at the webpage) where the goal is to estimate the quality (sensory appreciation) of a wine based on physicochemical inputs.

Using a 80-20 training-test split with a fixed seed (`random_state=0`), you are asked to learn MLP regressors to answer the following questions.

Given their stochastic behavior, average the performance of each MLP from 10 runs (for reproducibility consider seeding the MLPs as `random_state` ∈ {1..10}).

1) [3.5v] Learn a MLP regressor with 2 hidden layers of size 10, rectifier linear unit activation on all nodes, and early stopping with 20% of training data set aside for validation. All remaining parameters (e.g., loss, batch size, regularization term, solver) should be set as default.
   Plot the distribution of the residues (in absolute value) using a histogram visualization.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, shuffle=True, random_state=0)
predictor = MLPRegressor(hidden_layer_sizes=(10,10),activation='relu',
                         early_stopping=True,validation_fraction=0.2)
residues = []
for i in range(10):
    predictor.random_state = i
    predictor.fit(X_train, y_train)
    y_estimates = predictor.predict(X_test)
    residues.extend(np.abs(np.array(y_estimates)-np.array(y_test)))
print(predictor,"\nMAE =",np.mean(residues),"\nRMSE =",np.sqrt((np.array(residues)**2).mean()))
plt.hist(residues)
```



2) [1.5v] Since we are in the presence of an *integer regression* task, a recommended trick is to round and bound estimates. Assess the impact of these operations on the MAE of the MLP learnt in previous question.

```
MAE before round and bound: 0.514
MAE after round and bound: 0.446
```

Aprendizagem 2023/24

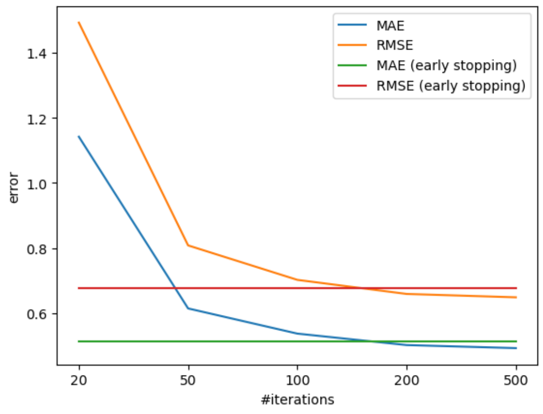
Homework III

Deadline 20/10/2023 (Friday) 23:59 via Fenix as PDF

Illustrative code considered for this and subsequent questions:

```
def eval_mlps(predictors, apply_round=False, apply_bound=False):
    maes, rmses = [], []
    X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.8,shuffle=True,random_state=0)
    for predictor in predictors:
        residues = []
        for i in range(10):
            predictor.random_state = i
            predictor.fit(X_train, y_train)
            y_estimates = np.array(predictor.predict(X_test))
            if apply_round: y_estimates = np.round(y_estimates)
            if apply_bound:
                y_estimates[y_estimates<0] = 0
                y_estimates[y_estimates>10] = 10
            residues.extend(y_estimates-np.array(y_test))
        maes.append(np.mean(np.abs(residues)))
        rmses.append(np.sqrt((np.array(residues)**2).mean()))
    return maes, rmses
```

3) [1.5v] Similarly, assess the impact on RMSE from replacing early stopping by a well-defined number of iterations in {20,50,100,200} (where one iteration corresponds to a batch).



4) [1.5v] Critically comment the results obtained in previous question, hypothesizing at least one reason why early stopping favors and/or worsens performance

– Early stopping worsens performance when considering a high number of maximum iterations:
  a. training with early stopping terminates when validation score does not improve by at least 10 consecutive epochs. However, after such epochs, performance improvements can still be observed
  b. less training data due to the need to set aside validation data
– Early stopping favors performance against a lower number of maximum iterations:
  a. prevents *underfitting risks* – too few iterations for proper generalization on testing data
  b. more iterations than the fixed maximum needed for convergence on training data

END