Homework Assignment # 3

Due: Friday, November 22, 2019, 11:59 p.m.
Total marks: 100

Question 1. [60 MARKS]

In this question, you will implement several binary classifiers: naive Bayes, logistic regression and a neural network. An initial script in python has been given to you, called `script_classify.py`, and associated python files. You will be running on a physics dataset, with 8 features and 100,000 samples (called `susysubset`). The features are augmented to have a column of ones, in `dataloader.py` (not in the data file itself). Baseline algorithms, including random predictions and linear regression, are used to serve as sanity checks. We should be able to outperform random predictions for this binary classification dataset.

(a) [15 MARKS] Implement naive Bayes, assuming a Gaussian distribution on each of the features. Try including the columns of ones and not including the column of ones in the predictor. What happens? Explain why.

(b) [15 MARKS] Implement logistic regression.

(c) [20 MARKS] Implement a neural network with a single hidden layer, with the sigmoid transfer.

(d) [10 MARKS] Implement k-fold internal cross-validation for picking the best meta-parameters for logistic regression and the neural network. A simple, generic cross-validation interface has been provided in *script_classify.py*. For this assignment, use $k = 5$ folds. Choose at least one meta-parameter and at least three values of that meta-parameter to test for each algorithm. Report the average and standard error for the meta-parameters chosen by cross-validation on the test set.

Question 2. [40 MARKS]

In this question, you will implement kernel logistic regression. Kernel logistic regression can be derived using the kernel trick, where the optimal solution $\mathbf{w}$ is always a function of the training data $\mathbf{w} = \mathbf{X}^\top \boldsymbol{\alpha}$ for $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{\alpha} \in \mathbb{R}^n$. Therefore, we could instead learn $\boldsymbol{\alpha}$, and whenever we predict on a new value $\mathbf{x}$, the prediction is $\mathbf{x}^\top \mathbf{w} = \mathbf{x}^\top \mathbf{X}^\top \boldsymbol{\alpha} = \sum_{i=1}^n k(\mathbf{x}, \mathbf{x}_i)\alpha_i$ with $k(\mathbf{x}, \mathbf{x}_i) = \langle \mathbf{x}, \mathbf{x}_i \rangle$ in this case. In general, we can extend to other feature representations on $\mathbf{x}$, giving $\phi(\mathbf{x})$ and so a different kernel $k(\mathbf{x}, \mathbf{x}_i) = \langle \mathbf{x}, \mathbf{x}_i \rangle$.

The kernel trick is useful conceptually, and for algorithm derivation. In practice, when implementing kernel regression, we do not need to consider the kernel trick. Rather, the procedure is simple, involving replacing your current features with the kernel features and performing standard regression or classification. For learning, we replace the training data with the new kernel representation:

$$\mathbf{K}_{\text{train}} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{c}_1) & k(\mathbf{x}_1, \mathbf{c}_2) & \dots & k(\mathbf{x}_1, \mathbf{c}_k) \\ \vdots & \vdots & \vdots & \vdots \\ k(\mathbf{x}_n, \mathbf{c}_1) & k(\mathbf{x}_n, \mathbf{c}_2) & \dots & k(\mathbf{x}_n, \mathbf{c}_k) \end{bmatrix} \in \mathbb{R}^{n \times k}$$

for some chosen centers (above those chosen centers were the training data samples $\mathbf{x}_i$). For example, for the linear kernel above with $k(\mathbf{x}, \mathbf{x}_i) = \langle \mathbf{x}, \mathbf{x}_i \rangle$, the center is $\mathbf{c} = \mathbf{x}_i$. Notice that the number of features is now k, the number of selected centers, as opposed to the original dimension d. Once your've transformed your data to this new representation, then you learn $\mathbf{w}$ with logistic

regression as usual, such that $\mathbf{K}_{\text{train}}\mathbf{w}$ approximates $\mathbf{y}_{\text{train}}$. As before, you can consider adding regularization. The prediction is similarly simple, where each new point is transformed into a kernel representation using the selected centers.

(a) [25 MARKS] Implement kernel logistic regression with a linear kernel and run it on `susysubset`. Compare the performance in one sentence to the performance of the algorithms from the first question.

(b) [15 MARKS] Using the same implementation, change the linear kernel to a Hamming distance kernel and run the algorithm on the dataset `Census`. In one or two sentences, summarize your performance, compared to the random predictor.

Bonus (mandatory for 566). [20 MARKS]

(a) [10 MARKS] In the first part of this assignment, you implemented a neural network with one hidden layer. Implement a second neural network with two hidden layers. Additionally, use your implementation of RMSProp from the previous assignment to train this two hidden layer neural network.

(b) [10 MARKS] In the first part of the assignment, you implement k-fold internal cross-validation. When partitioning the dataset into k validation sets, it is important to balance the ratio of samples belonging to class A versus samples belonging to class B for each validation set.

Implement stratified sampling k-fold cross-validation so each validation set has the same class ratio as the training set. Test each of the algorithms in the first part of the assignment again, and report the differences in your findings.