TTIC 31020: Introduction to Machine Learning
Autumn 2019

Problem Set #2 (85 Points)

Out: October 18, 2019

Due: Monday October 28, 11:59pm

Instructions

How and what to submit? Please submit your solutions electronically via Canvas. Please submit two files:

1. A PDF file with the written component of your solution including derivations, explanations, etc. You can create this PDF in any way you want, e.g.,LaTeX (recommended), Word + export as PDF, scan handwritten solutions (note: must be legible!), etc. Please name this document ⟨`firstname-lastname`⟩`-sol2.pdf`.

2. The empirical component of the solution (Python code and the documentation of the experiments you are asked to run, including figures) in a Jupyter notebook file. Rename the notebook ⟨`firstname-lastname`⟩`-sol2.ipynb`.

Late submissions: there will be a penalty of 20 points for any solution submitted within 24 hours past the deadline. No submissions will be accepted past then.

What is the required level of detail? When asked to derive something, please clearly state the assumptions, if any, and strive for balance: justify any non-obvious steps, but try to avoid superfluous explanations. When asked to plot something, please include in the `ipynb` file the figure as well as the code used to plot it. If multiple entities appear on a plot, make sure that they are clearly distinguishable (by color or style of lines and markers) and references in a legend or in a caption. When asked to provide a brief explanation or description, try to make your answers concise, but do not omit anything you believe is important. If there is a mathematical answer, provide it precisely (and accompany by succinct words if appropriate).

When submitting code (in Jupyter notebook), please make sure it's reasonably documented, runs and produces all the requested results. If discussion is required/warranted, you can include it either directly in the notebook (you may want to use the markdown style for that) or in the PDF writeup.

Collaboration policy: collaboration is allowed and encouraged, as long as you (1) write your own solution entirely on your own, (2) specify names of student(s) you collaborated with in your writeup.

1 Classification

In this section we will consider a discriminative model for a multi-class setup, in which the class labels take values in $\{1, \ldots, C\}$. A principled generalization of the logistic regression model to this setup is the *softmax* model. It requires that we maintain a separate D-dimensional parameter vector $\mathbf{w}_c$ for each class c. Under this model, the estimate for the posterior for class c, $c = 1, \ldots, C$ is

$$\widehat{p}(y = c \mid \mathbf{x}; \mathbf{W}) \;=\; \mathrm{softmax}(\mathbf{w}_c \cdot \mathbf{x}) \;\triangleq\; \frac{\exp(\mathbf{w}_c \cdot \mathbf{x})}{\sum_{y=1}^{C} \exp(\mathbf{w}_y \cdot \mathbf{x})}, \tag{1}$$

where $\mathbf{W}$ is a $C \times D$ matrix, the c-th row of which is a vector $\mathbf{w}_c$ associated with class c. We will assume throughout the problem set that $\mathbf{x}$ is the feature vector associated with an input example, including the constant feature $x_1 = 1$.

Problem 1 [15 points]
Show that the softmax model corresponds to modeling the log-odds between any two classes $c_1, c_2 \in \{1, \ldots, C\}$ by a linear function.

Furthermore, consider the binary case ($C = 2$). Show that in that case, for any two D-dimensional parameter vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ in the softmax model, there exists a single D-dimensional parameter vector $\mathbf{v}$ such that

$$\frac{\exp(\mathbf{w}_1 \cdot \mathbf{x})}{\exp(\mathbf{w}_1 \cdot \mathbf{x}) + \exp(\mathbf{w}_2 \cdot \mathbf{x})} \;=\; \sigma(\mathbf{v} \cdot \mathbf{x}),$$

i.e., that in the binary case the softmax model is equivalent to the logistic regression model.

End of problem 1

Problem 2 [10 points]
Show that the softmax model as stated in (1) is *over-parameterized*, that is, show that for any settings of $\mathbf{w}_c$ for $c = 1, \ldots, C$ there is a different setting that yields exactly the same $p\,(y \mid \mathbf{x})$ for every $\mathbf{x}$. Then explain how this implies that we only need $C - 1$ trainable parameter vectors for softmax, and not C.

End of problem 2

We now turn to a practical exercise in learning the softmax model, which can be done very similarly to learning logistic regression – via (stochastic) gradient descent. We will consider L_2 regularization:

$$\mathbf{W}^* \;=\; \underset{\mathbf{W}}{\mathrm{argmax}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \log \widehat{p}\,(y_i \mid \mathbf{x}_i; \mathbf{W}) \;-\; \lambda \|\mathbf{W}\|^2 \right\} \tag{2}$$

where $\|\mathbf{W}\|^2$ is the Frobenius norm of the matrix $\mathbf{W}$.[1]

[1] The Frobenius norm of a matrix $\mathbf{A}$ is the sum of squares of its elements, $\sum_i \sum_j A_{ij}^2$.

Problem 3 [15 points]
Write down the log-loss of the L_2-regularized softmax model, and its gradients with respect to $\mathbf{W}$ (in the stochastic setting, i.e., computed over a single training example). Then, write the update equation(s) for stochastic gradient descent, assuming learning rate η.

End of problem 3

Advice: It is helpful, both in derivation and in coding, to convert the scalar representation of the labels $y \in \{1, \ldots, C\}$ to a vector representation $\mathbf{t} \in \{0,1\}^C$, in which if $y_i = c$ then $t_{ic} = 1$ and $t_{ij} = 0$ for all $j \neq c$. This is sometimes called "one-hot" encoding of the labels: among C elements of the 0/1 label vector, exactly one element is "hot", i.e., set to 1.

We are now ready to apply the softmax model to the problem of classifying handwritten digits. We will work with the MNIST dataset. This is a dataset of handwritten digits which has served as a machine learning benchmark for many years. Each example is a 28-by-28 pixel grayscale image of a handwritten digit; we will be working with a vectorized representation of the images, converted to a 784-dimensional integer vector with values between 0 (black) and 255 (white).

The dataset provided to you consists of three parts:

- **Training** set of 20000 examples;
- **Validation** set of 10000 examples;
- **Test** set of 10000 examples (no labels are provided).

The data are contained in a zipfile, which should be unzipped in the same directory the Juptyer notebook will be run from. Each set is divided roughly equally among 10 classes.

In addition to the normal training set of 20000 examples, we include a small training set of only 30 examples (3 per class), to investigate the effect of dataset size on training a classifier in this domain.

We will be using a linear softmax model to classify these images. In Problem 4 below you will implement the gradient update procedure from Problem 3 in order to classify images of handwritten digits. We have provided skeleton code in the Jupyter notebook that you will have to modify in order to get the best possible prediction results. Note that the code will implement **mini-batch** gradient descent, in which updates are based on b examples with $b \geq 1$, averaging the gradient over the mini-batch. The size of the mini-batch is one of the **hyperparameters** of the learning procedure, along with the regularization parameter λ, stopping criteria, etc.

Problem 4 [45 points]

Fill in missing code for

- calculation of log-likelihood,
- calculation of the gradient of the regularized objective,

- selection of the model based on a sweep over values of λ (for both full and small datasets).

We have provided some suggested values for the hyperparameters of the learning algorithm: size of the mini-batch, stopping criterion (currently just limit on number of iterations), and the settings for the initial learning rate and for decreasing its value over iterations (or not). These should be a reasonable starting point, but you are encouraged to experiment with their values, and to consider adding additional variations: changing the mechanism for selection of examples in the mini-batch (e.g., how should the data be sampled?), additional stopping criteria, different schedule for dropping the learning rate, etc.

Feel free to guess appropriate values, or to tune them on the validation set.

Your tuning procedure and any design choices, and the final set of values for all the hyperparameters chosen for the final model, should be clearly documented in the notebook; please write any comments directly in the notebook rather than in the PDF file.

Please report the following statistics for your model in the write-up: the training accuracy, the validation accuracy, and the confusion matrix. The accuracy is the fraction of examples in the set that are classified correctly. The confusion matrix in a classification experiment with C classes is a $C \times C$ matrix $\mathbf{M}$ in which M_{ij} is the number of times an example with true label i was classified as j. Note that $\mathbf{M}$ is not necessarily a symmetric matrix.

In addition, visualize the parameters of the learned model. Since these are in the same domain as the input vectors, we can visualize them as images. Specifically, ignore the bias term, and use for instance

```
plt.imshow(w[-28**:, c].reshape(28, 28))
plt.colorbar()
plt.show()
```

to show the vector $\mathbf{w}_c$ associated with class c in model $\mathbf{W}$. Try to develop and write down an intuitive explanation of what the resulting images show.

Finally, compare the behavior of training across the two data regimes (small vs. large). In particular, compare the absolute values of training and validation accuracy and the role of regularization. Write your observations and conclusions in the notebook.

For the final evaluation, we have set up two Kaggle competitions to which you will be submitting your final predictions on a held-out test set, one for your best model trained on the large training set, and one for your best model trained on the small training set. The URLs are:

`https://www.kaggle.com/c/ttic-31020-hw2-mnist-fall-2019`
`https://www.kaggle.com/c/ttic-31020-hw2-mnist-small-fall-2019`

First, you will have to create a Kaggle account (with your `uchicago.edu` or `ttic.edu` email). Once you have access to the competition pages (when you have an account follow the invite links above to gain access), read through all three information pages carefully (Description, Evaluation, and Rules) and accept the rules. You will now be ready to make submissions. The two competitions have the same goal and structure.

Running the notebook in its entirety should create two files, `small_submission.csv` and `large_submission.csv`, in the directory the notebook is run from. Once you have accepted the Kaggle rules, there will be an option to "Make a submission", where you can upload this CSV file. To make sure you do not overfit this held-out set, **we have limited your submissions to two per day, so start early and you will get more chances if you make mistakes.** Your up to date score will appear on the public leaderboard once you submit.

End of problem 4