TTIC 31020: Introduction to Statistical Machine Learning
Autumn 2017

Problem Set #1

Out: October 16, 2017

Due: Friday October 27, 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: typeset the solution in LaTeX (recommended), type it in Word or a similar program and convert/export to PDF, or even hand write the solution (legibly!) and scan it to PDF. Please name this document ⟨`firstname-lastname`⟩`-sol1.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. Name the notebook ⟨`firstname-lastname`⟩`-sol1.ipynb`.

Late submissions: there will be a penalty of 25 points for any solution submitted within 24 hours past the deadlne. 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 int 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). 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 is precisely (and accompany by only succinct words, if appropriate).

When submitting code, please make sure it's reasonably documented, and describe succinctly in the written component of the solution what is done where.

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 on the PDF.

1 Softmax

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 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

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

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 $x_0 = 1$.

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.

Furtermore, 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 exist 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 an different setting that yields exactly the same $p\,(y\,|\,\mathbf{x})$ 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

Now, we turn to optimization properties of the logistic regression model. We will consider the Hessian matrix for the log-loss. To remind you, it is defined as the matrix of second derivatives of the loss w.r.t. the model parameters:

$$\mathbf{H} \;=\; -\begin{bmatrix} \frac{\partial^2 \log p}{\partial w_0^2} & \frac{\partial^2 \log p}{\partial w_0 w_1} & \cdots & \frac{\partial^2 \log p}{\partial w_0 w_d} \\ \frac{\partial^2 \log p}{\partial w_0 w_1} & \frac{\partial^2 \log p}{\partial w_1^2} & \cdots & \frac{\partial^2 \log p}{\partial w_1 w_d} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 \log p}{\partial w_d w_0} & \frac{\partial^2 \log p}{\partial w_d w_1} & \cdots & \frac{\partial^2 \log p}{\partial w_d^2} \end{bmatrix} \tag{2}$$

For the following problem, we will consider the simplest case in which there are only two classes, and so the model is binary logistic regression; assuming d features, the Hessian $\mathbf{H}$ is $d \times d$.

Problem 3 [15 points]
Show that $\mathbf{H}$ is positive definite; that the log-loss function of logistic regression model is convex in $\mathbf{w}$; and that the log-loss has a unique minimum.

End of problem 3

Advice: If you don't remember it, look up the definition of a positive definite matrix. You will find it helpful to show that $\mathbf{H} = \mathbf{X}^T\mathbf{R}\mathbf{X}$ where $\mathbf{X}$ is the data matrix and $\mathbf{R}$ is a diagonal matrix, and then show that $\mathbf{x}^T\mathbf{H}\mathbf{x} > 0$ for $\mathbf{x}^T\mathbf{x} > 0$. Then consider Taylor expansion of log-loss around a minimum, and show that log-loss must be convex and the minimum must be unique.

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 the L_2 regularization

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmax}} \left\{ \frac{1}{N} \sum_{i=1}^{N} \log \widehat{p}\,(y_i|\mathbf{x}_i; \mathbf{W}) - \lambda \|\mathbf{W}\|^2, \right\} \tag{3}$$

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

Problem 4 [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 the stochastic gradient descent, assuming learning rate η.

End of problem 4

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 "Fashion MNIST" data set[2]. Each example is a 28 by 28 pixel greylevel image of an item of clothing or accesory (the classes are ; 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 data set provided to you consists of three parts

- **Training** set of 50000 examples;

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

[2] This is a recently proposed data set intended to succeed the famous MNIST data set of handwritten digits, that has served as machine learning benchmark for many years.

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

Each set is divided roughly equally among 10 classes.

In addition to the normal training set of 50000 examples, we include a small training set of only 30 examples (3 per class), to investigate the effect data set size has on training a classifier in this domain.

We will be using a linear softmax model to classify these images. In the last problem you will implement the gradient update procedure in the previous problem 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 a *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 *hyper-parameters* of the learning procedure, along with the regularization parameter λ, stoppinc criteria, etc.

Problem 5 [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.

We have provided some suggested values for the "hyper-parameters" of the learning algorithm: size of the mini-batch, stopping criteria (currently just limit on number of iterations), 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 hyper-parameters 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 and contrast the behavior of training, in particular the absolute values of training and validation accuracy and the role of regularization, in the two data regimes (small vs large data set). 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 testing set, one for your best model trained on the large training set, and one for the best model trained on the small training set. The URLs:

`https://www.kaggle.com/c/ttic-31020-hw2-fashion-mnist-small`
`https://www.kaggle.com/c/ttic-31020-hw2-fashion-mnist-large`

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.

The function `create_submission_file` provided with the notebook will produce a Kaggle submission file from predictions produced by your model. Once you have accepted the Kagg;e 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 5