TTIC 31020: Introduction to Statistical Machine Learning
Autumn 2015

Problem Set #2

Out: October 19, 2016

Due: Thursday November 3, 11:59pm

Instructions

Please read these instructions carefully and follow them precisely. Feel free to ask the instructor if anything is unclear!

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.

In addition, you will need to submit your predictions on the handwritten digit recognition tasks to Kaggle, as described below, according to the competition rules. Please contact the TAs with any questions of difficulties.

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 the figure as well as the code used to plot it (and clearly explain in the `README` what the relevant files are). 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.

When submitting code, please make sure it's reasonably documented, and describe succinctly what is done in the relevant bits of code in the notebook.

1 Optimal classification

We have seen in class that the minimal risk for a particular joint distribution $p(\mathbf{x}, y)$ under 0/1 loss

$$L_{0/1}(\hat{y}, y) = \begin{cases} 0 & \text{if} \quad \hat{y} = y, \\ 1 & \text{if} \quad \hat{y} \neq y \end{cases}$$

is attained by the Bayes classifier $h^*(\mathbf{x}) = \operatorname{argmax}_c p(c \,|\, \mathbf{x})$. One may suspect that this bound is limited to *deterministic* classifiers. An attempt to "beat" this bound, then, could be based on the following, *randomized* classifier. Define, for any data point $\mathbf{x}$, a probability distribution $q(c \,|\, \mathbf{x})$ over class labels c conditioned on the input $\mathbf{x}$. The resulting randomized classifier (for which q serves as a parameter), given a data point $\mathbf{x}$, draws a random class label from q:

$$h_r(\mathbf{x}; q) \;=\; c_r, \qquad c_r \sim q(c \,|\, \mathbf{x}).$$

To express the risk of this classifier we need to take the expectation over all possible outcomes of the random decision:

$$R(h_r; q) \;=\; \int_{\mathbf{x}} \sum_{c=1}^{C} \sum_{c'=1}^{C} L_{0/1}(c', c)\, q(c_r = c' \,|\, \mathbf{x})\, p(\mathbf{x}, y = c)\, d\mathbf{x}.$$

Problem 1 **[15 points]**
Show that for any q,

$$R(h_r; q) \geq R(h^*),$$

that is, that the risk of the randomized classifier h_r defined above is at least as high as the Bayes risk.

End of problem 1

Advice: As we saw in class, it is enough to show that the inequality holds for the conditional risk, i.e., that $R(h_r|\mathbf{x}) \geq R(h^*|\mathbf{x})$ *for any* $\mathbf{x}$.

2 Regularization

We will consider ridge (L_2-regularized) linear least squares regression:

$$\mathbf{w}^* \;=\; \underset{\mathbf{w}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 \;+\; \lambda \sum_{j=1}^{d} w_j^2 \right\}, \tag{1}$$

where $\mathbf{x}_i$ is a d+1-dimensional input example, including the constant term the coefficient for which we do not regularize, and N is the size of the training set. We saw in class that this can be solved by a slight modification of the simple (non-regularized) least-squares problem, involving the N-element label vector $\mathbf{y}$ and the $N \times d + 1$ design matrix $\mathbf{X}$ in which each row is an input example.

Problem 2 **[15 points]**
Show that the solution for this problem is equivalent to the solution for a *unregularized* logistic regression with augmented data. Specifically, describe (precisely) the augmented design matrix $\mathbf{X}'$ and the augmented label vector $\mathbf{y}'$ such that solution to

$$\mathbf{w}^* \;=\; \underset{\mathbf{w}}{\operatorname{argmin}} \, (\mathbf{y}' - \mathbf{X}'\mathbf{w})^T \, (\mathbf{y}' - \mathbf{X}'\mathbf{w}) \tag{2}$$

is exactly the same as the solution for (1).

End of problem 2

3 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 for each class. The estimate for the posterior for class c, $c = 1, \ldots, C$ is

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

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.

Problem 3 **[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. In this problem, without loss of generality, you can assume that there is no bias ($\mathbf{b} = 0$), since it can be incorporated into $\mathbf{W}$ using a constant column of ones in $\mathbf{x}$.

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 3

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

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

where $\|\mathbf{W}\|^2$ is the Frobenius norm of the matrix $\mathbf{W}$ – look it up if you are not familiar with this term.

Problem 4 **[15 points]**
Write down the log-loss of the L_2-regularized softmax model, and its gradients with respect to $\mathbf{W}$ and $\mathbf{b}$ (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: You may find it 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 data set, which has served as a popular benchmark for classification methods over many years[1]. Each example is an 24 by 24 pixel greylevel image; we will be working with a vectorized representation of the images, converted to a 576-dimensional vector with values between 0 (black) and 1 (white).

The data set has four partitions you will work with:

- **Small training** set of 400 examples;
- **Large training** set of 8000 examples;
- **Validation** set of 2000 examples;
- **Test** set of 1000 examples (no labels).

Each set is divided roughly equally among 10 classes for digits 0 through 9. There are two training sets so we could investigate the effect of data scarcity (or relative abundance) on training a linear model for this task.

Problem 5 **[40 points]**

In this 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.

You will have to:

- Write the code to compute softmax predictions from model "scores" in `softmax`
- Write the computation for the gradient with respect to $\mathbf{W}$ and $\mathbf{b}$ in `calcGrad`.
- Write the update rule using the gradients and a step size in `modelUpdate`

[1] See `yann.lecun.com/exdb/mnist` for a list of many results; keep in mind that those are for the full training set of 60,000 examples, while you will be working with a reduced set of 10,000; the data given to you is further affected by added noise to make things a bit more interesting

- Fill in a set of values for the regularization parameter `lambda_` for which you will evaluate the performance of the model.

We have labeled parts of the skeleton code `YOUR CODE HERE`, where you will need to make changes.

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 (how should the data be sampled? should the mini-batch be constrained to be representative of all the classes?), additional stopping criteria, etc.

Feel free to guess appropriate values, or to tune them on the validation set. We have already provided code that evaluates the error rate on the training set and the validation set after the training has finished.

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 validation error and the

confusion matrix. 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.

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[:, i].reshape(24, 24))` to show the vector $\mathbf{w}_i$ associated with class i. 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 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:

- **https://kaggle.com/join/31020smallsubset**
- **https://kaggle.com/join/31020largesubset**

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), download the data file by clicking Data on the left-side menu. The file named NOISY_MNIST_SUBSETS.h5 is listed here in both competitions. This file contains small_train, large_train, val, and kaggle (test) partitions, which can be accessed using the provided `load_data` function. The data loaded from these partitions is of dimension

$$N \times 576$$

with elements between 0 and 1 (where N is the number of examples in the partition), and the label matrix loaded from the training and validation partitions is of size

$$N \times 10$$

(since there are 10 digit classes).

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; one is for models trained on 400 examples (small) and the other for the models trained on 20 times as many examples (large). You should treat these two data sets separately when deciding your hyper-parameters.

Our code will automatically evaluate your model and produce a Kaggle submission file for you, e.g. `submission-small.csv`. Once you have accepted the rules, there will be an option to "Make a submission", where you can upload this CSV file. The testing set that is evaluated for the Kaggle submission contains an additional 1,000 samples with unknown labels, to make sure you did not overfit the testing set. 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 score will appear on a leaderboard that everyone can see.

End of problem 5