Data Mining (CSC 503/SENG 474)

Assignment 2 **Due on Monday, February 28th, 11:59pm**

Instructions:

- You must complete this assignment on your own; this includes any coding/implementing, running of experiments, generating plots, analyzing results, writing up results, and working out problems. Assignments are for developing skills to make you strong. If you do the assignments well, you'll almost surely do better on the final and beyond.
- On the other hand, you can certainly have high-level discussions with classmates and can post questions to the Brightspace Discussion forum. You can also discuss preprocessing (data normalization) with classmates. You also are welcome to come to office hours (or otherwise somehow ask me and the TAs things if you are stuck).
- You must type up your analysis and solutions; I strongly encourage you to use LaTeX to do this. LaTeX is great both for presenting figures and math.
- Please submit your solutions via Brightspace by the due date/time indicated above. This is a hard deadline. Any late assignment will result in a zero (I won't accept assignments even one day late; "I forgot to hit submit on Brightspace", or "I submitted the wrong file" etc. will not be met with any sympathy, so double check that you submitted and that you submitted the correct file). However, I will make an exception if I am notified prior to the deadline with an acceptable excuse and if you further can (soon thereafter) provide a signed note related to this excuse.

Introduction

This assignment has two parts. The first part involves implementing and applying (or finding and applying) some of the methods you have already seen and analyzing their results on a binary classification problem; this part is for all students (both CSC 503 and SENG 474). The theme of this part is using cross-validation for hyperparameter tuning and model selection. The second part is actually only for graduate students (only CSC 503) and is about training logistic regression with stochastic gradient descent.

1 Experiments and Analysis

First, "implement" the following methods:

- **Logistic regression**. Your implementation should allow for some form of regularization. I suggest going with l_2 regularization. This means that you penalize the training objective according to $\|w\|^2$, i.e., the sum of the squares of the weights (not including the bias term). To be clear, for binary labels $y_i \in \{0, 1\}$, the suggested training objective is:

$$\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \quad \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \ell\Big(y_i, \sigma\big(\langle w, x_i \rangle + b\big)\Big),$$

  where:

  - $\ell(y, \hat{y}) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$ is the cross-entropy loss;
  - $\sigma(z) = \frac{1}{1+e^{-z}}$ is the sigmoid function;
  - $\|w\|^2 = \sum_{j=1}^{d} w_j^2$ is the squared Euclidean norm of the vector w.

  Here, $C \geq 0$ is the regularization parameter. Higher C means *less* regularization (more emphasis given to the training error).

- **Support vector machines**. This is the soft-margin version of SVM (the one that works for data that aren't necessarily linearly separable). Your implementation should allow for specifying a kernel function (you'll need the linear kernel and Gaussian kernel). Please see this page for a description of the soft-margin SVM problem, along with the regularization parameter that is used: https://scikit-learn.org/stable/modules/svm.html#svc.

  The relevant part is the equation following the text "SVC solves the following primal problem". Note that the regularization parameter C plays the same role as it did in logistic regression. Higher C means *less* regularization.

- **k-fold cross-validation**. Please implement this yourself so that you actually know how it works. Various machine learning/statistics software includes it, **but I still want you to implement it yourself. We will be checking that you coded this up.** *For all tasks involving using k-fold cross-validation, you will use the implementation you coded up.*

I put "implement" in quotes because I won't actually require you to implement logistic regression and SVMs; you can instead use machine learning software that you find online. You do need to implement k-fold cross-validation though. For any implementations you do, you can use whatever programming language you like. Logistic regression is implemented in scikit-learn. For SVMs in particular, it is quite difficult to implement a method for learning "kernelized SVMs", i.e., SVMs that use kernel functions. I therefore very strongly recommend that you use an existing implementation. For example, scikit-learn has SVMs and lets you specify the kernel.

What to test on

You'll be analyzing the performance of these methods on a somewhat-recently devised classification problem, known as Fashion-MNIST. Basically, the idea is to classify 28×28 grayscale images of clothing/apparel items, where each image belongs to one of ten classes. You can read all about it here if you would like to know more:

https://github.com/zalandoresearch/fashion-mnist

You can obtain the dataset from the previous link. Alternatively, you can use this direct link:

https://github.com/zalandoresearch/fashion-mnist/archive/master.zip

For instructions on how to load the dataset, use this README:

https://github.com/zalandoresearch/fashion-mnist/blob/master/README.md

I recommend the section titled "**Loading data with Python (requires Numpy)**". Moreover, you should be able to use scikit-learn for this assignment. If you encounter any trouble loading the data, please see Appendix A before contacting the instructor/TAs.

As mentioned above, Fashion-MNIST corresponds to a multi-class problem with ten classes; however, this assignment is about binary classification. Therefore, once you load the dataset, use only the data points (from both the training and test sets) from classes 5 and 7 (corresponding to "Sandal" and "Sneaker" respectively). You can let class 5 from Fashion-MNIST be class 0 of your binary classification problem (and class 7 from Fashion-MNIST can be class 1 of your binary classification problem). *To make training easier, I recommend rescaling the inputs by dividing them by 255, so they lie in the range* $[0, 1]$. *This rescaling is a form of preprocessing. You (optionally) can also try other preprocessing techniques. You can feel free to ask classmates about preprocessing techniques that work well or don't work well.*

How to do the analysis

Now that you have implementations and the data prepared, it's time for the main part of the assignment: the experiment analysis. Your analysis will go into a file called `report.pdf` and consists of the following parts, which should be presented in separate sections in this order:

First. For logistic regression, plot how the training and test error change as you vary the regularization parameter C. For training, use the whole training set (all 12,000 examples), and for testing, use the whole test set (all 2,000 examples). You should try at least ten values of the regularization parameter. I suggest using a logarithmically spaced grid for initial value $C_0 > 0$ and base $\alpha > 1$, so you try C_0, αC_0, $\alpha^2 C_0$, and so on. The values you try ideally should capture the regime of both underfitting (too much regularization) and overfitting (too little regularization). Discuss the results.

Second. For linear SVM (SVM with the linear kernel), plot how the training and test error change as you vary the regularization parameter C. I suggest proceeding as you did for logistic regression, but note that the range of values you try for the regularization parameter likely will need to be different. Again, discuss your results.

Third. Now that you have some understanding of the regularization parameter of each algorithm, it's time to do some hyperparameter tuning. The goal is to do a fair, head-to-head comparison of an optimally regularized logistic regression classifier and an optimally regularized linear SVM. For this, using only the training data, use k-fold cross-validation (for k being some integer between 5

and 10) to select the optimal value of the logistic regression regularization parameter. Likewise, use k-fold cross-validation to select the optimal value of the SVM regularization parameter. Now that you have optimal regularization parameters for each, train each optimally-tuned method on the *entire* training set and report the method's test error. Discuss the results. Are the test errors significantly different? Think about confidence intervals (which depend on the test set size) to assess significance.

Fourth. The final part of the assignment is to go beyond linear classifiers. For this, you'll use kernelized SVMs with the Gaussian kernel. This kernel has a scale parameter (sometimes called a "bandwidth" parameter). For each value γ in a good range of values of the scale parameter (I suggest using a logarithmic spacing as you did with the regularization parameter), use k-fold cross-validation on the training set to select the optimal regularization parameter C_γ. Now, for each (γ, C_γ) pair, you have a tuned SVM that can be trained. For each tuned SVM, train on the full training set and compute both the training error and the test error. Finally, plot the results. The x-axis will be γ, and the y-axis will be the training error (or test error) based on an SVM with parameters tuned to (γ, C_γ).

How do these test errors compare to the test error of linear SVM (with regularization parameter tuned via cross-validation on the training set, which you already did)?

Advice if you training an SVM is taking too long

You might find that training an SVM can take a long time. This is especially the case when using the Gaussian (rbf) kernel. In order to complete the assignment within a reasonable amount of time, it is OK if you reduce the size of the original training set. **If you do this, please use the same reduced-size training set throughout your assignment.**

For example, each of classes 5 and 7 consists of 6000 training examples (for 12000 training examples total). To help with reducing the training time, you could only use 3000 training examples for each class (for 6000 classes total). If that still takes too long, you might even go as low as using only 1000 training examples for each class.

Final advice on the report.

Please make your analysis concise (yet thorough). Don't ramble, and don't make stuff up. Act as if you are a respected scientist presenting some work to your colleagues (and you want them to continue being your colleagues after the presentation).

What to submit

In all, for the analysis portion of the assignment, you should submit (as a zip file):

- the file `report.pdf` explained above;
- a file called `README.txt` which contains instructions for running your code (for any code you wrote) or gives proper attribution to any code/datasets you got from other sources (like the Internet, for example). If you mostly used some existing code/software but needed to modify a few files, then give attribution and mention the files you modified.
- a file called `code.zip`, containing your code. Please organize this file well (embrace the idea of directories). In case you mostly used some existing code/software but needed to modify a few files, then just provide those files here, and make sure your `README.txt` mentions those files in relation to the existing code/software.
- any additional files that you need.

2 Logistic regression and SGD - CSC 503 only (undergrads, see Section 3 below)

In class, we saw that training a logistic regression model involves minimizing the training error according to the cross-entropy loss, and we typically further add to the objective an l_2 regularization term.

Implement[1] stochastic gradient descent (SGD) for regularized logistic regression, where you allow the learning rate to change in each round (note: this is basically one line of code!). Therefore, instead of having a fixed learning rate η, in round t the learning rate will be η_t. Note that each choice of the regularization parameter and learning rate schedule corresponds to a different logistic regression method. For a fixed value of the regularization parameter:

> Experiment with different schedules for the learning rate. One schedule, for example, is $\eta_t = \frac{\alpha}{t_0+t}$, for different choices of $\alpha > 0$ and $t_0 > 0$. It would be fine if you limit yourself to this schedule and focus on finding a good value of α and t_0. A good strategy could be to use cross-validation to select from a logarithmically spaced grid for each of α and t_0 (so, a grid in two dimensions). Once you select these, a good way to proceed is to train the resulting SGD method on the entire training set and compute its test error.

Now, repeat the above for a few different values of the regularization parameter and report the results. A good idea, if you can run experiments fast enough, is to select the value of the regularization parameter using cross-validation on the training set.

Report your results by augmenting `report.pdf` with an additional section.

3 How you'll be marked

- For each of CSC 503 and SENG 474, the total number of marks is 100, and you receive 10 marks for your code
- For undergrads (SENG 474), the analysis (in `report.pdf`) is worth 90 marks.
- For grad students (CSC 503), the analysis in part one of `report.pdf` is worth 70 marks and the analysis in part two of `report.pdf` is worth 20 marks.

[1] In this case, you actually do need to write code yourself.

A Loading Fashion-MNIST

For loading the Fashion-MNIST data, I believe everything should work fine if you are using Python in a terminal. However, in case you have difficulty loading the data from a Jupyter notebook, then you can also get the data in CSV format here (you will have to get a Kaggle account to download the csv files):

https://www.kaggle.com/eliotbarr/fashion-mnist-tutorial#data