Assignment 1 - Binary Linear Learning

Xiang Zhang, John Langford and Yann LeCun

February 26, 2013

Instructions

- Deadline: March 12, 2013 before midnight.
- You must answer the questions by yourself, but you may discusss the results of experiments with other students. We recommend you to submit questions to our Piazza forum where you can get help from both instructors and students.
- Send an assignment report, along with all of your source code, to the TA ONLY. The email address is `xiang.zhang@nyu.edu`. Please make everything in one tar ball or zip file, naming it as: LASTNAME_FIRSTNAME.tar.gz
- Wirte all your answers to the questions below to the report, in pdf format. Please do not include any doc, docx, odt, html or plain texts. If you hate writing maths in computers, we can accept hand-written reports submitted before the class at the day of the deadline. You still have to send all the source code to the TA if you choose to submit a hand-written report.
- Do not include any dataset in your submission.
- We accept late submissions, but there will be some penalty on your score.

1. (20 points) (Semantics of loss functions in binary linear classification) In binary classification we are usually given a dataset S with samples $(x, y) \in S$ such that $x \in \mathbb{R}^n$ and $y \in \{-1, +1\}$, to learn a classifier $y = h(x)$ where h is chosen from some hypothesis class H. In the case of binary linear classification, h is parameterized by a vector $w \in \mathbb{R}^n$ such that $h_w(x) = w^T x$. The decision is made by the rule $g(x) = \text{sign}(h_w(x)) = \text{sign}(w^T x)$ for some w that was learnt. It is worth noting that if we define a function $f(w, x, y) = y \cdot h_w(x) = y \cdot w^T x$, the following observation holds if given x, by the fact that $y \in \{-1, +1\}$:

$$\operatorname*{argmax}_{y \in \{-1,+1\}} f(w, x, y) = \text{sign}(w^T x) = g(x).$$

   As a result, given a sample (x, y), some weight vector w correctly classifies it if $f(w, x, y) > 0$.

   (a) (5 points) The square loss is defined as:

$$L(w, S) = \frac{1}{|S|} \sum_{(x,y) \in S} (y - w^T x)^2 = \frac{1}{|S|} \sum_{(x,y) \in S} (y - h_w(x))^2.$$

Now if we fix an input $x_0 \neq 0$, the subset of samples in S with input x_0 is defined as $S_0 = \{(x,y)|(x,y) \in S \wedge x = x_0\}$. The conditional loss on S_0 is then

$$L(w, S_0) = \frac{1}{|S_0|} \sum_{(x,y)\in S_0} (y - h_w(x))^2.$$

Show that when $L(w, S_0)$ is minimized, the following holds

$$h_w(x_0) = \frac{1}{|S_0|} \sum_{(x,y)\in S_0} y.$$

This means that the algorithm will try to make $h_w(x_0)$ the empirical expectation of y conditioned by x_0.

(b) (5 points) The quantile loss is defined as:

$$L(w, S) = \frac{1}{|S|} \sum_{(x,y)\in S} |y - w^T x| = \frac{1}{|S|} \sum_{(x,y)\in S} |y - h_w(x)|.$$

Similar to the question before, for some given $x_0 \neq 0$, the conditional loss on $S_0 = \{(x,y)|(x,y) \in S \wedge x = x_0\}$ is

$$L(w, S_0) = \frac{1}{|S_0|} \sum_{(x,y)\in S_0} |y - h_w(x)|.$$

Assume $|S_0| = 2m$, where $m \in \mathbb{N}$. Show that $L(w, S_0)$ is minimized if we found an h_w such that S_0 is separated into two disjoint sets S_- and S_+, for which $|S_-| = |S_+| = m$ and the following holds

$$\sup_{(x,y)\in S_-} y < h_w(x_0),$$
$$\inf_{(x,y)\in S_+} y > h_w(x_0).$$

This means that the algorithm will try to make $h_w(x_0)$ the empirical median for y conditioned by x_0. (Please assume $y \in \mathbb{R}$ for this part.)

(c) (5 points) Since $f(w,x,y)$ should be large (or at least, positive) when a correct classification is made for an input x, for a given sample $(x,y) \in S$, we can define the *discrimination* as $\rho(w,x,y) = f(w,x,y) - f(w,x,\bar{y})$ where $\bar{y}$ is an incorrect classification. The "correctness" is good when $\rho(w,x,y)$ is large. Notice that for binary classification $\bar{y} = -y$. The hinge loss is defined as:

$$L(w, S) = \frac{1}{|S|} \sum_{(x,y)\in S} \max\{0, 1 - y \cdot w^T x\}.$$

Show that there exists some number p such that if $\forall (x,y) \in S, \rho(w,x,y) \geq p$, then $L(w,S) = 0$. Report this p value. (Hint: writing $L(w,S)$ as a function of $\rho(w,x,y)$). This is another way to show the margin property of support vector machines.

(d) (5 points) The Logistic loss is defined as

$$L(w, S) = \frac{1}{|S|} \sum_{(x,y)\in S} \log(1 + \exp(-y \cdot w^T x)).$$

Show that minimizing $L(w, S)$ will maximize the conditional joint probability $\Pr[y_1, y_2, \ldots, y_{|S|}|w, x_1, x_2, \ldots, x_{|S|}]$ with $(x_i, y_i) \in S$, by assuming that $(x, y) \in S$ was drawn independently and identically from some distribution characterized by the per-sample conditional probability

$$\Pr[y|w, x] = \frac{1}{1 + \exp(-f(w, x, y))}.$$

2. (45 points) (Binary linear learning in Torch 7) This part will be done in Torch 7, a Matlab-like environment for state-of-the-art machine learning algorithms. To get more information for the Torch 7 software, please refer to the course website. There are numerous toturials available where you can get yourself started pretty quickly. From the following questions you will understand how Torch 7 works using a generlized neural network framework, and try out almost every quansi-Newton method you have learnt in the lectures.

   (a) (5 points) One way of pre-conditioning the learning rates is to use the inverse of the diagonal terms of the Hessian. If given a dataset S with samples $(x, y) \in S$, the Hessian for the quadratic loss

   $$L(w, S) = \frac{1}{|S|} \sum_{(x,y)\in S} (y - w^T x)^2$$

   is

   $$H = \frac{2}{|S|} \sum_{(x,y)\in S} x x^T.$$

   However, in learning practices we often regularize the loss by an l_2 norm on the quadratic loss

   $$L(w, S) = \frac{1}{|S|} \sum_{(x,y)\in S} (y - w^T x)^2 + \lambda \|w\|^2.$$

   Then, what is the Hessian for the regularized quadratic loss?

   (b) (10 points) `src/optim` contains some skeleton code of the demos used for lecture 2 on second-order learning methods. Implement the `learningRates(data,lamb)` function in `src/optim/demo.lua`. It should return the pre-conditioned learning rates for a regularized quadratic loss, using the diagonal terms of the Hessian computed from the dataset `data`, with l_2 regularization parameter `lamb`. You can set the rate for the bias as $1/2$. Note that a regularization term with parameter `lambda` is added to each group of samples of size `config.batchSize`. Then, do the following experiments.

      (i) Make two copies of `demo.lua` as `demo1.lua` and `demo2.lua`. `demo1.lua` should do stochastic gradient descent (SGD) without pre-conditioning, whereas `demo2.lua` should do preconditioning, by setting the variable `state.learningRates`. Note that you may rescale the learning rates by setting the variable `state.learningRate` (no plural!) for plausible convergence results. Keep `epoches = 2*train_size` for both cases. Submit a screenshot of the energy surfaces for both experiments.

(ii) Make two copies of `demo.lua` as `demo3.lua` and `demo4.lua`. `demo3.lua` should do batch gradient descent without pre-conditioning, whereas `demo4.lua` should do pre-conditioning. You can change from SGD to batch just by modifying the variable `config.batchSize` to be equal to `train_size`. Please set the variable `epoches` to be 100 for both cases. Submit a screenshot of the energy surfaces for both experiments.

What can you conclude from the experiments above?

(c) (30 points) `src/learn` contains some skeleton code for you to use in testing the performance of various loss functions on the spambase dataset. The example of using the square loss with `optim.sgd` is given. Implement the following loss functions following the `nn.MMSECriterion` object in `criteria.lua`, which follows the design of `Criterion` in Torch 7's `nn` package.

(i) The quantile loss. Name it as `nn.QuantCriterion`.

(ii) The Logistic loss. Name it as `nn.LogisCriterion`.

Note that the Hinge loss is available as `nn.MarginCriterion` already. For each of the losses square, quantile, Logistic and Hinge, for each algorithms `optim.sgd`, `optim.cg`, `optim.bfgs` and `optim.lbfgs` (please notice that some of them can only be used with the `minibatch` or `batch` trainer in `xtrain`), compare their performace on the spambase dataset with 3000 training samples and 1000 testing samples by reporting their training loss, training error, testing loss, testing error and training time. You should choose the appropriate training paradigms from `stochastic`, `batch` and `minibatch` in `xtrain`, set appropriate parameters for each of the algorithms, and use the appropriate regularization parameter of an l_2 regularizer. Your should submit a table in your report listing the numbers you obtained.

3. (35 points) (Binary linear learning in Vowpal Wabbit) Vowpal Wabbit (why didn't they call it 'Vorpal Rabbit'?) is a fast online machine learning software that could be used to train linear models for many common paradigms and applications. In this assignment we will start simple on training binary linear classifiers, with the study on precision-recall trade-offs by exploiting how to weight differently for each sample in VW's dataset format. For all the problems below, you can implement the required pre-processing and post-processing scripts in any programming language you want, such as Python, Torch 7 (Lua), or even bash. But please place all source code into the directory `src/vw`.

(a) (5 points) Go through with the following example on the Malicious URL dataset:
`https://github.com/JohnLangford/vowpal_wabbit/wiki/Malicious-URL-example`
Write a script `preproc.*` to convert the dataset from `svm_light` format to two files `train.dat` and `test.dat` in VW format. Your script should be able to add different weights for label 1 and label -1 samples. It should also accept a parameter $0 < p < 1$, in which with probability p a sample is randomly put into `test.dat` rather than `train.dat`. Thus, if I call your script with $p = 0.1$, `train.dat` should have about 2.16 million samples, while there are about 0.24 million samples in `test.dat`.

(b) (10 points) Write a script `postproc.*` to extract the labels from `test.dat` and compare it with some label file produced by VW from testing on `test.dat`. The script should print out not only the testing error, but also the precision and recall values with respect to the +1 label (i.e., URLs classified as malicious). For more information about precision and recall, here is a wikipedia article which introduces them in a very nice way:

http://en.wikipedia.org/wiki/Precision_and_recall
Can you explain why we care about precision and recall for this dataset? How would precision and recall change if we weight +1 and -1 labeled samples differently in the dataset?

(c) (20 points) Separate the dataset into a training set and a testing set with $p = 0.1$. Using VW, for each of the loss functions in `{squared,logistic,hinge,quantile}`, do the following two sets of experiments:

  (i) Weight 1 for samples with labels +1; enumerate in $\{1, 2, 3, 4, 5\}$ for the weight on samples with labels -1.

  (ii) Weight 1 for samples with labels -1; enumerate in $\{1, 2, 3, 4, 5\}$ for the weight on samples with labels +1.

Name each of the predictor as `loss_positiveweight_negativeweight.vw`, such as `hinge_1_2.vw`. Then, for each loss function, give a plot of testing errors, testing precisions and testing recalls with respect to the ratio

$$\frac{\text{weight of samples with labels +1}}{\text{weight of samples with labels -1}}.$$

Feel free to tune your VW command (add regularizers, change update schemes, etc) for the best results in each case. Your plot should be in the report. Do **not** submit any data files or predictor files such as `*.dat` and `*.vw`, but record your command line in a shell script `log.sh` as part of the source code to be submitted. Does your plot describe what you had thought in the previous question?