Homework 6: Regularization & Sparsity

Brown University CSCI1420 & ENGN2520: Machine Learning

Homework due at 11:59pm on November 5, 2015

Question 1:

This problem compares various approaches to regularization and feature selection for binary classification. Let $t_n \in \{-1, +1\}$ denote the binary class label we want to predict from binary input features $x_n$. Consider the following binary logistic regression model:

$$p(t_n = 1 \mid x_n, w) = \frac{1}{1 + \exp(-w^T x_n)}.$$

The Dorothea dataset (http://archive.ics.uci.edu/ml/datasets/Dorothea) contains 100,000 features encoding structural molecular features of chemical compounds, and the label $t_n$ indicates whether the compound is active (binding) or inactive. We have split the dataset into 400 training, 400 validation, and 350 test instances. Each data instance $x_n$ is a binary vector of dimension $10^5$, stored in a sparse matrix for efficiency.

We will compare three types of regularization: $L_2$ (Gaussian prior), $L_1$ (Laplacian prior), and the Huber loss which smoothly interpolates between quadratic and linear. Given $N$ training examples and $M$ features, the MAP estimator minimizes the following objective:

$$f(w) = -\sum_{n=1}^{N} \log p(t_n \mid x_n, w) + \lambda \sum_{m=1}^{M} L(w_m).$$

The following questions define and compare three different choices for the regularizer, or negative log-prior, $L(w)$. For each candidate regularizer, we use the validation set to choose among ten logarithmically spaced values of the regularization weight $\lambda$:

```
>> lambdas = logspace(-8,1,10);
```

For either $L_2$ or $L_1$ regularization, we can fit the model via the `logregFit` function from the pmtk3 package. Here is an example for $L_2$ regularization:

```
>> ll = lambdas(1);
>> pp = preprocessorCreate();
>> modelL2 = logregFit(X_train,Y_train,'regType','L2','lambda',ll,'preproc',pp);
>> Y_est = logregPredict(modelL2,X_val);
```

The `pp` argument avoids default pre-processing which is inappropriate for binary features. The `logregPredict` method can also be used to predict labels for test data. These functions build on the `logregFit` and `LogisticLossSimple` methods, which (as distributed in pmtk3) can have numerical problems and be inefficient in memory usage. We have provided improved versions which, if placed in your working directory, will be called instead.
a) Train logistic regression models with Gaussian regularization ($L_2(w_m) = w_m^2$) on the train data. Create a plot of the validation error rate as a function of $\lambda$.

b) Select the $\lambda$ value which results in the smallest $L_2$-regularized validation error. Report this optimal $\lambda$, the number of nonzero weights in this model (see the Matlab command `nnz`), and its error rate on the test data.

c) Train logistic regression models with Laplacian regularization ($L_1(w_m) = |w_m|$) on the train data. Create a plot of the validation error rate as a function of $\lambda$.

d) Select the $\lambda$ value which results in the smallest $L_1$-regularized validation error. Report this optimal $\lambda$, the number of nonzero weights in this model, and its error rate on the test data.

e) What happens to the $L_1$-regularized validation error rate when $\lambda = 10$? Provide an explanation for your observation.

f) The Huber loss, with “closeness” parameter $\delta > 0$, is defined as follows:

$$L_H(w, \delta) = \begin{cases} 
    w^2/2 & \text{if } |w| \leq \delta, \\
    \delta|w| - \delta^2/2 & \text{if } |w| > \delta.
\end{cases}$$

Derive an expression for the derivative of this loss with respect to $w$. Is its second derivative defined everywhere?

g) Implement a gradient-based algorithm for fitting Huber regularized logistic regression, using the minFunc method, the LogisticLossSimple method, and the gradients derived above. See the demo code for examples of how to use the optimization function.

h) Train logistic regression models with Huber regularization, and closeness $\delta = 100$, on the train data. Create a plot of the validation error rate as a function of $\lambda$. What happens? Explain your observation.

i) Now set $\delta$ to the median of the absolute value of the weights obtained from the logistic regression model with $L_2$ regularization and $\lambda = 10^{-8}$. Create a plot of the validation error rate as a function of $\lambda$. Explain any differences from the result in part (h).

j) Select the $\lambda$ and $\delta$ values which result in the smallest Huber-regularized validation error. Report these optimal parameters, the number of nonzero weights in this model, and its error rate on the test data.

k) What is one advantage of Huber regularization compared to $L_1$ regularization? What is one disadvantage?
Question 2:

Stochastic gradient descent (SGD) is a parameter estimation method that processes subsets (batches) of a large dataset at each step. Here, we compare SGD to “full dataset” gradient descent for maximum likelihood estimation of the weight vector for a simple logistic regression binary classifier. Recall that the logistic regression model is defined as

\[ p(t_n = 1 \mid x_n, w) = \sigma(w^T x_n), \quad \sigma(z) = \frac{1}{1 + e^{-z}}, \]

where the weight vector \( w \in \mathbb{R}^D \) and we have assumed the features are raw inputs \( x_n \).

For this problem, you’ll consider 11,791 handwritten digits from the MNIST dataset, each an example of the digit “4” or the digit “9”; see MNIST_Digits49_19x19.mat. These have been preprocessed so the labels are either +1 or -1, and so that instead of the original 28x28 pixel image, we have cropped out the center to produce a 19x19 image. This barely affects accuracy, but makes learning much speedier.

For parts that require SGD, use the provided function minFuncStochGrad.m to perform minimization across many batches of data. The demo code provides example usage to get you started. You’ll also need PMTK3’s minFunc routine.

a) As a baseline, use minFunc to estimate a logistic regression weight vector using the entire dataset. Using the estimated weight vector \( \hat{w} \), report the accuracy on the test set, as well as the number of function evaluations that minFunc required to converge to this solution. Note that each function evaluation must consider the entire dataset.

b) Train logistic regression models using SGD with three different divisions of the data into batches: 1 batch, 100 batches, and 11,791 batches (this last setting processes each example one at a time). Fix the step size schedule \( \eta_t = (10 + t)^{-0.51} \). Run the estimation for 30 passes through the full dataset (one pass is called an “epoch”). Create two figures: one for training accuracy and another for test accuracy. In both figures, plot the accuracy recorded as a function of the number of passes through the full dataset (number of epochs), with one curve for each candidate batch size. What trends do you notice?

c) Repeat the previous step for a different step size schedule: \( \eta_t = (10 + t)^{-0.7} \). Again, plot training accuracy and test accuracy versus number of epochs. Does performance appear sensitive to the step size schedule?

d) Compare the performance of the \( \hat{w} \) estimated by L-BFGS in part (a) to the estimates achieved by SGD. Discuss tradeoffs in solution quality versus the number of passes through the full dataset. (Due to overhead in the way objective function handles are evaluated, raw computation time may not be the best metric for this Matlab code.)
Question 3:

Suppose we would like to predict a binary label $t_n \in \{0, 1\}$ based on continuous features $x_n \in \mathbb{R}^D$. Consider maximum likelihood parameter estimates $\hat{\theta}$ for the following classifiers:

**GaussI** A generative classifier where the class conditional densities are Gaussian, with both covariance matrices set to identity matrices: $p(x_n \mid t_n = k) = \mathcal{N}(x_n \mid \mu_k, I)$. The class prior $p(t_n)$ is an unconstrained Bernoulli distribution.

**GaussX** A generative classifier similar to the GaussI model, but where covariances are also estimated from data: $p(x_n \mid t_n = k) = \mathcal{N}(x_n \mid \mu_k, \Sigma_k)$.

**LinLog** A logistic regression model with linear features plus a constant bias feature.

**QuadLog** A logistic regression model with a constant bias feature, linear features, and quadratic features (encoding the product of all pairs of inputs and their squares).

After training from $N$ observations we compute the performance of each model $M$ via evaluating its conditional log-likelihood on the training set:

$$L(M) = \frac{1}{N} \sum_{n=1}^{N} \log p(t_n \mid x_n, \hat{\theta}, M).$$

We now want to compare the performance of each model. We will write $L(M) \leq L(M')$ if model $M$ must have lower (or equal) conditional log-likelihood (on the training set) than $M'$, for any training set. For each of the following model pairs, state whether $L(M) \leq L(M')$, $L(M) \geq L(M')$, or whether no such statement can be made (i.e., $M$ might sometimes be better than $M'$ and sometimes worse). Also provide 1-2 sentence explanations.

a) GaussI versus LinLog.

b) GaussX versus QuadLog.

c) LinLog versus QuadLog.

d) GaussI versus QuadLog.

e) GaussX versus LinLog.