Problem 1 – Logit loss and backpropagation - NOT GRADED

The logit loss

\[ L_{\text{logit}}(w) = \ln(1 + e^{-yw^T x}), \quad x, w \in \mathbb{R}^n, \quad y = \pm 1 \quad (1) \]

is the negative log-likelihood of observation \((x, y)\) under the logistic regression model \(P(y = 1|x, w) = \phi(w^T x)\) where \(\phi\) is the logistic function.

a. Show that the partial derivatives \(\frac{\partial L_{\text{logit}}}{\partial w_i}, \frac{\partial L_{\text{logit}}}{\partial x_i}\) for \(L_{\text{logit}}\) in (1) can be rewritten as

\[ \frac{\partial L_{\text{logit}}}{\partial w_i} = -(1 - P(y|x, w))yx_i \quad (2) \]
\[ \frac{\partial L_{\text{logit}}}{\partial x_i} = -(1 - P(y|x, w))yw_i. \quad (3) \]

The elegant formulas above hold for a larger class of statistical models, called Generalized Linear Models.

Problem 2 – Logit loss Hessian

Assume now that you have a data set \(D = \{(x^i, y^i), i = 1 : N\}, x^i, w \in \mathbb{R}^n\).

a. Calculate the expression of \(\nabla^2 L_{\text{logit}}\) for a single data point \(x\). Simplify your result using \(\phi(yw^T x)\) conveniently.

b. Show that the gradient of \(L_{\text{logit}}(w; D)\) is a linear combination of the \(x^i\) vectors.

c. Show that if \(N < n\) the Hessian of \(L_{\text{logit}}(w; D)\) has at least one 0 eigenvalue, and conclude that \(L_{\text{logit}}(w; D)\) is not strongly convex in this case.

d. – Optional, extra credit If \(|x^i| \leq R\), find a constant \(M\) sufficiently large so that \(\nabla^2 L_{\text{logit}}(w; D) \prec MI_n\).
Problem 3 – Descent algorithms for training a neural network

This problem asks you to train a neural network to classify the data sets given on the Assignments web page. The inputs are 2-dimensional, outputs are ±1, one data point/line. Submit the code for this problem.

Objective to minimize is \( \hat{L}_{\text{logit}}(\beta, W) = -\frac{1}{N} \log\text{-likelihood}(D|\beta, W) \) and \( \beta \in \mathbb{R}^{m+1}, W \in \mathbb{R}^{n \times m} \) are the neural net parameters.

Choose a number \( m = 3 \) to 5 hidden units (suggested) or go as high as you want (recommended to try both).

Algorithms: One of GFIX = steepest descent with fixed step size or GLS = steepest descent with line search (or both). [Optional, for extra credit: implement Newton, or run Newton, LBFGS quasi Newton from library code.]

Dataset \( D \) given \texttt{hw3-nn-train-100.dat}

\( \textbf{a.} \) Start all algorithms from the same initial point. Explain how you chose the initial points. It’s ok to plot the data and look at it or even to make a sketch of the solution you want to find.

The training algorithms will converge to a local optimum. It’s OK to look at this local optimum and try other initial points if the found optimum is bad. Don’t forget to use the same initial point for all algos in the results you present in the homework. It’s also recommended to challenge the algorithm by giving it random/uninformative initial points. Do not start all the parameters at 0 [Why?].

Chose the stopping criterion \( 1 - \frac{L^{k+1}}{L^k} \leq tol \) with \( tol = 10^{-4} \). If this tolerance cannot be reached in a reasonable number of steps, set a higher \( tol \) and report that value.

\( \textbf{b.} \) The choices above should be kept the same for all estimation algorithms (except maybe SG). Describe briefly the implementation details of your algorithms. Size of the fixed step, if you bracketed the min or not in line search, what line search method you used \( \text{(you can use code from other sources to bracket the minimum, and you can implement another line search method than Armijo.)} \), how you chose \( C \) in the stochastic gradient algorithm (trial and error OK) and what value you used, etc. For each algorithm, give the number of iterations (and if it converged or not) and final value of loss function. Record also the time each algorithm takes and report it.

Compute the value of \( L_{\text{logit}}, L_{01} \) (by averaging them) on the test set \texttt{hw3-nn-test.dat}
for the final classifier obtained by each method and report it. Optionally, compute these values at each iteration and plot them in the graphs for c.

c. Plot the values of $\hat{L}_{\text{logit}}$, $\hat{L}_{01}$ and the respective costs on the test set vs the iteration number $k$. Try to put all algorithms on the same plot if it looks right, but make two separate plots for the two costs. For SG, plot only after each pass through the whole data set and consider this one iteration. If you have computed the test set costs at each iteration, plot these too on the respective graphs.

d. Plot the final decision region superimposed on the data; preferably do this for all 3 algorithms on the same plot.

[e. Optional but encouraged] Plot (some of) the $\beta$ parameters vs $k$; on a separate plot, show trajectories of $\beta$ parameters coming from different initializations.

Please make clear, well-scaled, well labeled graphs.

f. Repeat steps b,c,d,[e] for the larger data set hw3-nn-train-10000.dat on the same models as before. Use the same parameter initialization as in the previous case to get meaningful comparisons.

Do not plot the data set for this part of the problem.

[Optional, extra credit: repeat the training initializing from the final values obtained in the small sample run. Plot what you think is meaningful to compare performances.]

g. Discuss the differences that you observe between the algorithms’ behavior on the large and small samples.