Gradient descent for logistic regression

In this exercise you will program and learn different learning algorithms. As an example we consider the logistic regression problem, which is not too simple (it is non-linear in the parameters) and not too hard (it has a unique solution).

Inputs patterns \(x^n = (x^n_1, \ldots, x^n_d) \in \mathbb{R}^d\) output patterns \(t^n = \{0, 1\}\) with \(n = 1, \ldots, N\) and \(N\) the total number of patterns.

The model is given as follows. The probability of \(t = 1\) given \(x\) is given by

\[
y = p(t = 1|x) = \sigma \left( \sum_{i=0}^{d} w_i x_i \right)
\]

where we have defined \(x_0 = 1\) and \(\sigma\) is the sigma function \(\sigma(x) = (1 + e^{-x})^{-1}\) and \(p(t = 0|x) = 1 - y\). Define \(y^n = p(t = 1|x^n)\). The cost to be minimized is minus the log likelihood of the data

\[
E(w) = -\frac{1}{N} \sum_{n=1}^{N} [t^n \log y^n + (1 - t^n) \log(1 - y^n)]
\]

The gradient is

\[
\nabla_i E(w) = \frac{\partial E}{\partial w_i} = \frac{1}{N} \sum_{n=1}^{N} (y^n - t^n) x^n_i \quad i = 0, \ldots, d
\]

Test the different learning methods on the MNIST data [http://yann.lecun.com/exdb/mnist/](http://yann.lecun.com/exdb/mnist/) The data in matlab format are here: [http://www.snn.ru.nl/~bertk/comp_neurosci/mnistAll.mat](http://www.snn.ru.nl/~bertk/comp_neurosci/mnistAll.mat) These are images of digits 0 to 9. Select the images 3 and 7 for your two class classification problem. Scale the inputs to be in the range \([0, 1]\).

**Gradient descent**

The simplest idea to minimize \(E\) is by gradient descent. Start with random initial \(w\) and update according to

\[
\Delta w_i = -\eta \frac{\partial E}{\partial w_i}
\]

Implement the learning rule in your computer program. Apply the method to the logistic regression problem. Produce plots of how \(E\) decreases with iterations, both on the training set and on the test set. Note, that the training error should always decrease, but the test error not. Stopping at the point where the test error increases is called early stopping, and produces a result with relatively good generalization error. Discuss the drawbacks of early stopping. Test the effect of different values of \(\eta\).

Here are some results that I got after 10,000 iterations:

\[
E_{\text{train}} = 0.0184 \quad E_{\text{test}} = 0.0511
\]

The fraction of misclassified patterns is 1.14% and 2.05% on train and test set respectively. CPU time Matlab implementation was 45 s.
Momentum

Add momentum to your gradient rule with strength $\alpha$ (see slides Machine Learning). Apply the method to the logistic regression problem. Produce plots of how $E$ decreases with iterations, both on the training set and on the test set. Test the effect of different values of $\alpha, \eta$.

Here are some results that I got after 10,000 iterations:

$$E_{\text{train}} = 0.0114 \quad E_{\text{test}} = 0.0636$$

The fraction of misclassified patterns is 0.78% and 2.07% on train and test set respectively. CPU time Matlab implementation was 45 s.

Stochastic gradient descent

In the above, so-called batch methods, the computation of the gradient requires time linear in the size of the data set. When the data set is large, this can be a significant cost. The stochastic gradient descent method only uses a subset of the total data set (sometimes called mini batch). Implement the stochastic gradient descent method.

Apply the method to the logistic regression problem. Consider constant learning rate $\eta$ and mini batch sizes. Produce plots of how $E$ decreases with iterations, both on the training set and on the test set for different values of $\eta$.

I find with mini batch size of 0.01$N$ after 5000 iteration on the training set $E = 0.0243$ and classification error 1.36% and on the test set $E = 0.0499$ and classification error 2.02%. Elapsed time is 5 seconds.

Assignment

Implement all the above methods using your favorite computer language. Provide results that are competitive with the results that I stated and show the graphs. Hand in the code so that I can reproduce your results.