Assignment 1

CMSC 678 — Introduction to Machine Learning

Due Wednesday February 7th, 2018, 11:59 AM

<table>
<thead>
<tr>
<th>Item</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assigned</td>
<td>Monday January 29th, 2018</td>
</tr>
<tr>
<td>Due</td>
<td>Wednesday February 7th</td>
</tr>
<tr>
<td>Topic</td>
<td>Math and Programming Review</td>
</tr>
<tr>
<td>Points</td>
<td>80</td>
</tr>
</tbody>
</table>

Machine learning often requires combining multivariate calculus, introductory probability and statistics, numerical optimization, and computational thinking. In this assignment you will review and cover some of the basic techniques that are necessary for understanding many machine learning methods.

You are to complete this assignment on your own: that is, the code and writeup you submit must be entirely your own. However, you may discuss the assignment at a high level with other students or on the discussion board. Note at the top of your assignment who you discussed this with or what resources you used (beyond course staff, any course materials, or public Piazza discussions).

The following table gives the overall point breakdown for this assignment.

<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points</td>
<td>25</td>
<td>15</td>
<td>40</td>
</tr>
</tbody>
</table>

**What To Turn In**   Turn in a PDF writeup that answers the questions; turn in all requested code necessary to replicate your results. Be sure to include specific instructions on how to build (compile) and run your code. Answers to the following questions should be long-form. Provide any necessary analyses and discussion of your results.

**How To Submit**    Submit the assignment on the submission site:

[https://www.csee.umbc.edu/courses/graduate/678/spring18/submit](https://www.csee.umbc.edu/courses/graduate/678/spring18/submit)

Be sure to select “Assignment 1.”
Full Questions

1. (25 points) Hal Daumé III has a very nice “refresher” tutorial called “Math for Machine Learning:”
http://www.umiacs.umd.edu/~hal/courses/2013S_ML/math4ml.pdf Where indicated, some of the following questions are taken from that primer. I encourage everyone to read the primer, but especially if you have difficulty answering the following questions.

(A) [Exercise 1.2] Compute the derivative of the function \( f(x) = \exp\left(-\frac{1}{2}x^2\right) \).

(B) [Exercise 1.5] Compute the derivative of the function \( f(x) = \log(x^2 + x - 1) \).

(C) Compute the derivative with respect to \( x \) of the function \( f(x) = \log\left(\sum_{K}^{K} \exp(x^k)\right) \), for finite, positive, integral \( K \).

(D) Compute the derivative with respect to \( x \) of the function \( f(x) = \log\left(\prod_{K}^{K} \exp(x^k)\right) \), for finite, positive, integral \( K \).

(E) [Exercise 2] Given \( N \) points \( \{(x_n, y_n)\}_{n=1}^{N} \), compute the partial derivative \( \frac{\partial J}{\partial b} \), where \( J(m, b) = \sum_{n=1}^{N} ((mx_n + b) - y_n)^2 \).

(F) [Exercise 3] For \( J(m, b) \) defined on the \( N \) points, as above, compute the values of \( m \) and \( b \) that minimize \( J \).

(G) Describe, in English and with as little math as possible, the meaning of a convex (or concave) function. Why do we (in ML) care about convexity?

(H) [Exercise 7] Compute the Euclidean, Manhattan, Maximum, and Zero norms on the three vectors \((1, 2, 3), (1, -1, 0), (0, 0, 0)\).

(I) [Exercise 14 and 19] Given the matrix \( A = \begin{pmatrix} 5 & 10 \\ -2 & 0 \\ 1 & -1 \end{pmatrix} \), compute the values \( A^T A \), \( AA^T \), and the traces \( \text{tr}(AA^T) \), \( \text{tr}(A^T A) \). Discuss how the traces relate to the Frobenius norm.

(J) [Exercise 20] Compute the values of \( \text{tr}(A^T A) \) and \( \text{tr}(AA^T) \). What do these values remind you of?

(K) [Exercise 22] Given the matrix formula \( X = A(B + C) \), solve for \( B \). (Assume the matrices \( X, A, \) and \( C \) are all known, square, and full rank.)

(L) [Exercise 24] For the multivariate function \( f(u, v) = \exp(u^Tv) \), where \( u, v \in \mathbb{R}^K \), compute the gradients \( \nabla_u f \) and \( \nabla_v f \).

(M) How would you use the Hessian of a function \( f \) to verify that \( f \) is convex?

2. (15 points)

(A) What does it mean for two random variables \( X \) and \( Y \) to be statistically independent?

(B) What does it mean for two random variables \( X \) and \( Y \) to be conditionally independent (given r.v. \( Z \))? Is this the same as \( X \) and \( Y \) being independent? Explain your answer.

(C) Consider the following generative story: Given \( \eta \sim \text{Normal}(0, 1) \) define \( p_1 = \frac{\exp(\eta)}{\exp(\eta) + 1} \) and \( p_0 = 1 - p_1 \). Let \( X_i | \eta \sim \text{Bernoulli}(p_0, p_1) \) be i.i.d. (independent, identically distributed), for \( 1 \leq i \leq N \).

• Describe, in English and with as little math as possible, the above generative story. (What role does \( \eta \) play, what do \( p_i \) each represent, etc.)
• Let \( Y = \sum_i X_i \). Compute the (conditional) expected value of \( Y: \mathbb{E}[Y \mid \eta] \).
• Compute the derivate \( \frac{\partial \mathbb{E}[Y \mid \eta]}{\partial \eta} \).

3. (40 points) Consider the following three functions:

\[
\begin{align*}
\text{f} (\omega = (\omega_0, \omega_1)) &= \sum_{i=1}^{N} \left[ -\omega y_i + \log \left( \sum_{j \in \{0,1\}} \exp(\omega_j) \right) \right] \\
\text{g} (z = (z_1, z_2)) &= (a - z_1)^2 + b \left( z_2 - z_1^2 \right)^2 \\
\text{h} (x = (x_1, x_2, \ldots, x_K)) &= \sum_{i=1}^{K-1} \left[ (c - x_i)^2 + d \left( x_{i+1} - x_i^2 \right)^2 \right].
\end{align*}
\]  

[Eq-1] [Eq-2] [Eq-3]

For \( f \) [Eq-1], let \( y_i \in \{0, 1\} \); for any given complete optimization, you can treat these \( y_i \) as constants. Notice that \( h \) [Eq-3] is a generalization of \( g \) [Eq-2]: let \( a, b, c, d \) be scalar, positive constants.

You are to complete and turn in the following:

**Pre-computation Task** Derive the partial derivatives for the three functions. Include the derivations in your write-up.

**Programming Task** Implement gradient descent in order to numerically minimize [Eq-1], [Eq-2], and [Eq-3]. Ensure that you are able to compare a Robbins-Monro schedule with an AdaGrad schedule (this is described below). Turn in all of your code, and make sure to tell us how to build and run it. Make note of any implementation difficulties you experienced.

**Analysis Task** In your write-up, report and analyze

- the convergence behavior (how quickly does the optimization converge),
- the observed variability in the optimal points (\( \omega^* \) for [Eq-1], \( z^* \) for [Eq-2], and \( x^* \) for [Eq-3]), and
- the optimal function values (\( f(\omega^*), g(z^*), h(x^*) \)) for the three functions that are found by the optimization routines.

Perform these comparisons under the two learning rate schemes, and as you vary the per-function constants, the initial starting points, the dimensionality of [Eq-3], and other experimental decisions. Try to summarize your results both in prose and in graphs or tables. Discuss any implementation or experimental decisions you had to make.

In particular, there are a number of specifics you should consider for this question:

(i) **Learning Rates** The two learning rates you are to implement and experiment with are:

---

[1] [Eq-1] is the negative log-likelihood of \( N \) binary (0 or 1) variables, distributed via a maximum entropy distribution. [Eq-2] is called the Rosenbrock function; it is a standard function that is used to benchmark optimization algorithms.
• a simple Robbins-Monro scalar learning rate schedule, given by

\[ \rho_t = (t + \tau)^{-\kappa}, \]

[Eq-4]

with \(0.5 < \kappa \leq 1\) and \(\tau \geq 0\). This computes a single learning rate that is applied to all components of the gradient. See Hoffman et al. (2013).

• AdaGrad, a history-based, vector learning rate schedule (Duchi et al., 2011). If we are optimizing a \(K\)-dimensional multivariate function \(F(u)\), AdaGrad computes a learning rate for each of the \(K\) components (that is, it computes \(K\) different learning rates); it uses previous gradient values to help compute the new learning rate. Let \(G_{k,j}\) be the partial derivative of \(F\) wrt \(u_k\) at some past time (optimization iteration) \(j\). The AdaGrad update for \(u_k\) at the current time \(t\) is then given by

\[ \rho_{t,k} = \frac{\eta}{\sqrt{\epsilon + \sum_{j=1}^{t} G_{k,j}^2}}, \]

[Eq-5]

with \(\epsilon, \eta > 0\). Typically, \(\eta = 0.1\) and \(\epsilon = 1 \times 10^{-8}\), but you can experiment with them.

(ii) Setting/Sampling Constants \(\{y_1, a, b, c, d\}\) For [Eq-1], you need access to \(N\) different values \(y_1, y_2, \ldots, y_N\). Although each of these \(y_i\) is binary (0 or 1), it is up to you to decide the overall distributions of 0s and 1s. You can experiment with this distribution, including the value of \(N\), and examine the effect this has on the convergence. (\(N\) should not be trivially small, e.g., \(N\) should be at least 50.)

For the two Rosenbrock functions, you need to set \(a, b, c, d\). A typical case is to set \(a = c = 1\) and \(b = d = 100\). You can experiment with different values for these to determine how the end convergence behavior changes.

(iii) Setting the Initial Points Often, the initial point you use can determine how optimized the function becomes (hint: think back to 1.(G) of this assignment). Though setting the initial point to be the zero vector is very simple, and often effective, it can result in the optimization getting stuck in local optima; that is, the zero vector could be far away from any global optima and getting there could take a very long time. An alternative strategy is to randomly sample the initial point.

(iv) The Dimensionality of [Eq-3] How does the dimensionality of [Eq-3] affect the convergence behavior?

(v) The Stopping Criterion When to stop optimizing is a persistent question for ML practitioners. While you could have a hard cut-off, e.g., running for at most \(M\) iterations of gradient descent, you will probably want to have an early stopping criteria to indicate that you have sufficiently optimized the function. This stopping criteria could be based on the value of the gradient (is it sufficiently close to 0?) or on the current value of the objective function—or on some combination of the two. What is “sufficiently close to 0?” Often, that is an empirical question. The value of \(M\) could also be set arbitrarily, or empirically.