

Optimization, Programming Assignment #2

April 8, 2008

Description

In this assignment, you will experiment with two different line searches for gradient descent, and will solve a linear classification problem. Next, you will compare gradient descent to steepest descent (with a quadratic norm). If you find any errors in this (or any) assignment, please email your TA.

You should turn in an archive containing all of your source code, plots, and a document containing a brief description of your code (how to run it, what parameters it takes, etc), and answers to all questions. Email this archive to your TA, at `cotter@tti-c.org`.

Please remember to turn in a document, and to answer the questions (in particular, talk about what your plots mean)!

1 Gradient Descent

1.1 Implementations

1.1.1 Backtracking line search

Implement gradient descent (algorithm 9.3 of Boyd and Vandenberghe) with a backtracking line search (algorithm 9.2). Your function should terminate once the stopping criterion $\|\nabla f(x)\|_2 \leq \eta$ is satisfied.

Your function should take six parameters: f (the function to minimize); ∇f (the gradient); η (the stopping threshold); α and β (the parameters to the backtracking line search); x (the starting point). It should return a list of triples $(x, f(x), i)$, with one entry per iteration, containing: points x ; objective function values $f(x)$; i , the total number of calls to f or f' which have taken place to this point (including inside the line search).

As in the previous assignment, code defensively, and comment:

1. On entry, check that $\eta > 0$, $0 < \alpha < \frac{1}{2}$ and $0 < \beta < 1$, raising an error if any of these conditions is not satisfied
2. Check that you do not enter an infinite loop, by raising an error if some ridiculously large number of iterations are performed
3. Do the same for the backtracking line search loop
4. Try to evaluate the function f and gradient ∇f as few times as possible per iteration
5. Comment any portion of your code of which the interpretation is not obvious

1.1.2 Bisection line search

Instead of the backtracking line search, we may also perform an approximate exact line search, using the bisection method. If $f(x)$ is convex, then $g(t) = f(x + t\Delta x)$ will be convex, as a function of t , since $x + t\Delta x$ is affine. Hence, we may find a t which minimizes $g(t)$ by using the bisection method to find a zero of $\frac{dg}{dt} = \Delta x^T \nabla f(x + t\Delta x)$:

1. $t_{\text{lower}} := 0$

2. $t_{\text{upper}} := 1$
3. While $\Delta x^T \nabla f(x + t_{\text{upper}} \Delta x) < 0$
4. $t_{\text{lower}} := t_{\text{upper}}$
5. $t_{\text{upper}} := t_{\text{upper}} \times 2$
6. While $t_{\text{upper}} - t_{\text{lower}} > 2\epsilon$
7. $t_{\text{middle}} := \frac{1}{2}(t_{\text{lower}} + t_{\text{upper}})$
8. If $\Delta x^T \nabla f(x + t_{\text{upper}} \Delta x) < 0$
9. $t_{\text{lower}} := t_{\text{middle}}$
10. Else
11. $t_{\text{upper}} := t_{\text{middle}}$
12. Return $\frac{1}{2}(t_{\text{lower}} + t_{\text{upper}})$

Above, lines 3 – 5 search until we have found an appropriate starting interval (with $\Delta x^T \nabla f(x + t_{\text{lower}} \Delta x) < 0$ and $\Delta x^T \nabla f(x + t_{\text{upper}} \Delta x) > 0$), and lines 6 – 11 perform the search, using the bisection method.

Implement gradient descent, with this “exact” line search. Your function should meet all of the conditions listed in section 1.1.1, *except* that instead of taking backtracking line search parameters α and β , it will take the bisection parameter $\epsilon > 0$.

1.2 Logistic regression

Suppose that we have a list of training examples $x_i \in \mathbb{R}^n$ and corresponding class labels $y_i \in \{-1, 1\}$. We will assume that the log-odds of a sample being in each of the two classes are a linear function of X . That is, that:

$$\log \frac{P(Y = 1 | X)}{P(Y = -1 | X)} = w^T X \quad (1)$$

This assumption, combined with the fact that probabilities must sum to one, gives that:

$$P(Y | X) = \frac{e^{Yw^T X}}{1 + e^{Yw^T X}}$$

We will find a weight vector w which maximizes the log-likelihood of the data:

$$\begin{aligned} \mathcal{L}(w | x, y) &= \log \prod_{i=1}^n P(Y = y_i | X = x_i) \\ &= \sum_{i=1}^n \left(y_i w^T x_i - \log \left(1 + e^{y_i w^T x_i} \right) \right) \end{aligned} \quad (2)$$

You may wish to derive equation 2 from equation 1, but this is optional.

For more on logistic regression, check out:

- Trevor Hastie, Robert Tibshirani, Jerome Friedman. “The Elements of Statistical Learning”. Springer. 2001.

1.2.1 Convexity, gradient and Hessian

Maximizing the log-likelihood in the logistic regression model is clearly equivalent to the following:

$$\text{minimize} \quad : \quad \sum_{i=1}^n \left(-y_i w^T x_i + \ln \left(1 + e^{y_i w^T x_i} \right) \right)$$

Prove that this function is convex (hint: start out by proving that $\ln(1 + e^z)$ is convex as a function of z , and then use the results from section 3.2 of Boyd and Vandenberghe).

Calculate the gradient (with respect to w) of the above objective function, and implement functions to calculate $f(w)$ and $\nabla f(w)$.

1.2.2 Optimizing

The provided file <http://bleu.uchicago.edu/CMSC34500/pa2/data.csv> contains one thousand (y_i, x_i) triples, with $y_i \in \{-1, 1\}$ and $x_i \in [-1, 1] \times [-1, 1]$. Using this as the data, optimize the above objective function using gradient descent with backtracking line search, with $\eta = 10^{-4}$, $\alpha = \frac{2}{5}$, $\beta = \frac{1}{2}$, and initial point $w = \vec{0}$. Then, do the same, with the bisection method line search, with $\epsilon = 10^{-4}$.

Next, in order to get good estimates of the optimal point w^* and function value $p^* = f(w^*)$ (which you will need to create plots), run one of your algorithms until it performs a null update (that is, until $f(w)$ is left completely unchanged by an iteration, within the precision of the arithmetic).

Create the following plots:

- All of the points labeled +1 in one color, all of the points labeled -1 in another, and the separating line $(w^*)^T x = 0$ (along this line, both class labels are considered equally likely by our model)
- The log-error $\log |f(x) - p^*|$ versus the number of iterations, for gradient descent using both line searches, on the same plot
- The log-error $\log |f(x) - p^*|$ versus the number of function+gradient evaluations, for gradient descent using both line searches, on the same plot

Do you think that, on this problem, one of these line searches would be a better choice than the other? Why? Play around with different values of the parameters α , β and ϵ , to try to find good choices of all three. For these choices, which line search is better? Why?

2 Steepest descent

2.1 Implementation

Implement steepest descent (algorithm 9.4 of Boyd and Vandenberghe) with a backtracking line search (algorithm 9.2), using a quadratic norm defined by the positive definite matrix P (section 9.4.1). Your function should terminate once the stopping criterion $\|\nabla f(x)\|_2 \leq \eta$ is satisfied.

Your function should take seven parameters: f (the function to minimize); ∇f (the gradient); P (the “weight” matrix for the quadratic norm); η (the stopping threshold); α and β (the parameters to the backtracking line search); x (the starting point). It should return a list of triples $(x, f(x), i)$, with one entry per iteration, containing: points x ; objective function values $f(x)$; i , the total number of calls to f or f' which have taken place to this point (including inside the line search).

Make sure to do the following:

1. On entry, check that $\eta > 0$, $0 < \alpha < \frac{1}{2}$ and $0 < \beta < 1$, raising an error if any of these conditions is not satisfied. You may also want to check that P is positive definite, but this is not required
2. Check that you do not enter an infinite loop, by raising an error if some ridiculously large number of iterations are performed

3. Do the same for the backtracking line search loop
4. Try to evaluate the function f and gradient ∇f as few times as possible per iteration
5. Comment any portion of your code of which the interpretation is not obvious

2.2 Comparison with gradient descent

Consider the convex function:

$$f_A(x) = \ln\left(1 + e^{x^T A x}\right) + \mathbf{1}^T x$$

Where $x \in \mathbb{R}^n$, $A \in \mathbf{S}_{++}^n$, and $\mathbf{1} \in \mathbb{R}^n$ is the all-ones vector. What we'll be exploring in this problem is the performance of gradient descent and steepest descent, as the condition number (http://en.wikipedia.org/wiki/Condition_number) κ of A changes (note that κ is *not* the condition number of the Hessian $\nabla^2 f_A(x)$, though these two quantities are related).

We will be working in \mathbb{R}^2 , with:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & \kappa^{-1} \end{bmatrix}$$

Calculate the gradient *and Hessian* of the above objective function. Write functions to calculate $f_A(x)$, $\nabla f_A(x)$ and $\nabla^2 f_A(x)$.

For $\kappa \in \{1, 2, 4\}$, minimize $f_A(x)$ using gradient descent with backtracking line search, with $\eta = 10^{-6}$, $\alpha = \frac{2}{5}$, $\beta = \frac{1}{2}$, and initial point $x = \vec{1}$ (the all-ones vector). Next, do the same, with steepest descent, with quadratic norm “weight” matrix $P = \nabla^2 f_A(\vec{1})$ (the Hessian at the initial point). Finally, run steepest descent one more time, with P equal to the Hessian at the approximate optimum which you just found (this is cheating: we need to run either gradient descent or steepest descent to convergence before we can calculate this P).

As in problem 1.2.2, we'll need good good estimates of the optimal point x^* and function value $p^* = f(x^*)$ for each κ , in order to create plots. Find these values by running one of your algorithms until it performs a null update (that is, until $f(x)$ is left completely unchanged by an iteration, within the precision of the arithmetic).

Create the following plots:

- For each κ , create a plot containing the sequence of iterates x taken by each of the three optimization algorithms (we want something like figure 9.2 of Boyd and Vandenberghe, without the contour lines)
- Again for each κ , create a plot of the log-errors $\log|f(x) - p^*|$ versus number of iterations for all three optimization algorithms

Some of the paths in the first set of plots will overlap exactly, so you might want to offset the paths corresponding to each of the three algorithms by (different) small constants, just so that you can tell them apart.

Write a paragraph on your interpretation of these plots. How does steepest descent compare to gradient descent, with each of the choices of P matrix (Hessian at initial point, or at optimum)? Which choice of P seems to work better, in these examples? How does the relative performance of the three algorithms change, as κ changes? Why?