Курс: Методы оптимизации в машинном обучении, осень 2016 ВМК МГУ

Практическое задание 1, вариант 1.
Точная одномерная оптимизация.

Начало выполнения задания: 14 сентября 2016 г.
Срок сдачи: 28 сентября (среда), 23:59.
Среда для выполнения задания: Python 3.

1 Формулировка задания

1. Реализовать алгоритмы одномерной оптимизации без производной: метод золотого сечения, метод парабол и комбинированный метод Брента.

2. Протестировать реализованные алгоритмы на следующем наборе задач минимизации:

   (a) $f_1(x) := -5x^5 + 4x^4 - 12x^3 + 11x^2 - 2x + 1$ на интервале $[-0.5, 0.5]$;
   (b) $f_2(x) := -\ln^2(x-2) + \ln^2(10-x) - x^{0.2}$ на интервале $[6, 9.9]$;
   (c) $f_3(x) := -3x\sin(0.75x) + \exp(-2x)$ на интервале $[0, 2\pi]$;
   (d) $f_4(x) := \exp(3x) + 5\exp(-2x)$ на интервале $[0, 1]$;
   (e) $f_5(x) := 0.2x\ln x + (x-2.3)^2$ на интервале $[0.5, 2.5]$.

   Построить графики сходимости методов для каждой из функций. Исходя из графика, какую скорость сходимости (линейная/сублинейная/суперлинейная) имеет каждый из методов?

3. Протестировать реализованные алгоритмы для задач минимизации многомодальных функций, например, на различных полиномах. Для каждой такой функции нарисовать ее график и изобразить на нем стартовую точку и результат метода. Могут ли (теоретически) методы золотого сечения/Брента вместо локального минимума найти локальный максимум или седловую точку и почему?

4. Реализовать метод секущих (поиск нуля производной) и комбинированный метод Брента с производной, сравнить их работу с уже реализованными методами оптимизации без производной. Построить графики сходимости. Привести пример медленной сходимости метода секущих.[1]

5. Написать отчет в формате PDF с описанием всех проведенных исследований.

2 Оформление задания

Результатом выполнения задания являются 1) pdf-отчет о проведенных исследованиях[2] и 2) текстовый файл `optim1d.py`, содержащий исходные коды всех требуемых алгоритмов. Выполненное задание следует отправить письмом по адресу bayesml@gmail.com с заголовком

«[ВМК МОМО16] Задание 1 (вариант 1), Фамилия Имя».

[1] Здесь подразумевается, что нужно придумать некоторую (унимодальную) функцию, а затем запустить на ней метод секущих. В отчет нужно вставить формулу придуманной функции, ее график, а также график сходимости метода секущих.
[2] При этом допускается возможность сделать отчет в IPython-блокноте, а затем сконвертировать его в формат PDF.

Убедительная просьба присылать выполненное задание только один раз с окончательным вариантом.

График сходимости алгоритма одномерной оптимизации должен показывать зависимость невязки (по точке или значению функции) от числа вызовов оракула. Во всех графиках должна использоваться логарифмическая шкала для невязки. При сравнении различных методов на одной и той же задаче кривые сходимости этих методов нужно рисовать на одном графике; таким образом, каждый график должен соответствовать отдельной задаче и содержать несколько кривых, соответствующих разным алгоритмам.

Поскольку проверка реализованных алгоритмов будет осуществляться в полуавтоматическом режиме, все реализованные функции должны строго соответствовать приведенным ниже прототипам и корректно запускаться в Python 3 (а не Python 2!). Проверить наличие всех необходимых функций, а также их соответствие требуемым прототипам можно с помощью специального скрипта `check_submission_v1.py`, выдаваемого вместе с текстом задания.

3 Прототипы функций

1. Метод золотого сечения:

Модуль:	`optim1d`
Функция:	`min_golden(func, a, b, tol=1e-5, max_iter=500, disp=False, trace=False)`

Параметры:	`func: callable func(x)`
	Оракул минимизируемой функции.
	Принимает:
	`x: float`
	Точка вычисления.
	Возвращает:
	`f: float`
	Значение функции в точке `x`.
	`a: float`
	Левая граница интервала оптимизации.
	`b: float`
	Правая граница интервала оптимизации.
	`tol: float`, опционально
	Точность оптимизации по аргументу: `abs(x_k - x_opt) <= tol`
	`max_iter: int`, опционально
	Максимальное число итераций метода.
	`disp: bool`, опционально
	Отображать прогресс метода по итерациям (номер итерации, значение функции, длина текущего интервала и пр.) или нет.
	`trace: bool`, опционально
	Сохранять ли траекторию метода для возврата истории или нет.

Возврат:	`x_min: float`
	Найденное приближение минимума `x_opt`.
	`f_min: float`
	Значение функции в `x_min`.
	`status: int`
	Статус выхода, число:

	0: минимум найден с заданной точностью `tol`;
	1: достигнуто максимальное число итераций.
	`hist: dict`, возвращается только если `trace=True`
	История процесса оптимизации по итерациям. Словарь со следующими полями:
	`x: np.ndarray`
	Точки итерационного процесса.
	`f: np.ndarray`
	Значения функции в точках `x`.
	`n_evals: np.ndarray`
	Суммарное число вызовов оракула на текущий момент.

2. Прототипы функций `min_parabolic` и `min_brent` для метода парабол и метода Брента без производной полностью повторяют прототип функции `min_golden`. При отображении прогресса в методе Брента необходимо указывать способ выбора очередной точки на текущей итерации: golden/parabolic или bisection/parabolic.

3. Названия функций для метода секущих и метода Брента с производной: `min_secant` и `min_brent_der`. Их прототипы аналогичны `min_golden`, кроме того, что `func` теперь дополнительно возвращает вторым аргументом значение производной.

Ссылки

[1] William H Press. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007. Гл. 10.

[2] Кропотов Д.А. *Методы одномерной оптимизации: конспект лекций*. `http://www.machinelearning.ru/wiki/images/a/a8/MOMO12_min1d.pdf`.