COMP4670/6467
Introduction to Statistical Machine Learning
Tutorial 1

Christfried Webers

26 or 28 February/5 or 7 March 2013

The first tutorial will introduce the basic elements for writing Python programs. If you already know Python well, please work on the questions related to "Linear Algebra and Optimisation".

1 Python and Programming for Machine Learning

The introduction will focus on the concepts necessary for writing small programs in Python for the purpose of Machine Learning. That means, we expect a user of the code will be a reasonable knowledgeable person. Therefore, we can skip most of the code a robust system would have to contain in order to check the input types, verify the input parameter ranges, and make sure that really nothing can go wrong when somebody else is using the code.

Having said this, you are nevertheless encouraged to include some sanity tests into your code to avoid making simple errors which can cost you a lot of time to find.

Some of the Python concepts discussed in the tutorial will be

- Data types (bool, int, float, str, list, tuple, set, dict)
- Operators
- Data flow
- Functions
- Classes and objects
- Modules and how to use them
- Introduction to numpy, scipy

2 Linear Algebra and Optimisation

Consider the following cost function $f(X)$ defined over the space of real $n \times p$ matrices

$$f(X) = \frac{1}{2} \operatorname{tr}\left\{X^T C X N\right\} + \mu \frac{1}{4} \|N - X^T X\|_F^2 \tag{1}$$

where $X \in \mathbb{R}^{n \times p}$, $n \geq p$, and the matrix C is symmetric, such that $C = C^T$. The scalar μ is assumed to be larger than the pth smallest eigenvalue of C. The matrix N is diagonal with distinct positive entries on the diagonal. The trace of a square matrix A is denoted as $\operatorname{tr}\{A\}$, and the Frobenius norm of an arbitrary matrix A is defined as $\|A\|_F = \sqrt{\operatorname{tr}\{A^T A\}}$.

2.1 Frobenius Norm

Implement a Python function frobenius_norm which accepts an arbitrary matrix A and returns $\|A\|_F$ using the formula given. (Use *numpy.trace* and *math.sqrt*.)

1. Given a matrix $A \in \mathbb{R}^{n \times p}$, what is the complexity of your implementation of frobenius_norm using the formula above?
2. Can you come up with a faster implementation, if you were additionally told that $p \geq n$?
3. Can you find an even faster implementation than in 1. and 2.?

2.2 Cost Function $f(X)$ with matrix argument

Implement the cost function (1) as a function cost_function_for_matrix in Python.

Hint: As good programmers, we do not use global variables in subroutines.

2.3 Cost Function $f(X)$ with vector argument

Many standard optimisation routines work only with vectors. Fortunately, as vector spaces, the space of matrices $\mathbb{R}^{n \times p}$ and the space of vectors $\mathbb{R}^{np}$ are isomorphic. What does this mean?

Implement the cost function $f(X)$ given X as a vector and call it cost_function_for_vector . Which extra arguments do you need for the cost function?

2.4 Construction of a random matrix C with given eigenvalues

A diagonal matrix has the nice property that the eigenvalues can be directly read off the diagonal. Given a diagonal matrix $C \in \mathbb{R}^{n \times n}$ with distinct eigenvalues, how many different diagonal matrices have the same set of eigenvalues?

Given a diagonal matrix $C \in \mathbb{R}^{n \times n}$ with distinct eigenvalues, how many different matrices have the same set of eigenvalues?

Given a set of n distinct real eigenvalues $\mathcal{E} = \{e_1, \ldots, e_n\}$, write a Python function random_matrix_from_eigenvalues which takes a list of eigenvalues E and returns a random symmetric matrix C having the same eigenvalues.

2.5 Minimising the cost function $f(X)$

Use the minimisation functions fmin or fmin_powell provided in the Python package scipy.optimize to minimise the cost function cost_function_for_vector.

Hint: Use the argument args in the minimisation functions fmin or fmin_powell to provide the extra parameters to your cost function cost_function_for_vector.

2.6 Gradient of $f(X)$

Calculate the gradient for the cost function (1) given the inner product on the space of real matrices $n \times p$ is defined as

$$\langle A, B \rangle = \operatorname{tr}\left\{A^T B\right\} \tag{2}$$

Implement a function gradient_for_vector which takes X as a vector, and returns the gradient of $f(X)$ as a vector.

2.7 Minimising the cost function $f(X)$ using the gradient

Use the minimisation functions fmin_cg or fmin_bfgs provided in the Python package scipy.optimize to minimise the cost function cost_function_for_vector utilising the gradient gradient_for_vector.

Compare the speed of convergence to the minimisation with fmin or fmin_powell.

2.8 Minima of $f(X)$

Compare the minima of $f(X)$ to the eigenvectors related to the smallest eigenvalues of C.