```
export PATH=~phseal/ias_school/bin:$PATH
cp -r ~phseal/ias_school/GAP-tutorial .
cd GAP-tutorial
jupyter notebook --browser=firefox
```

numpy reference: http://docs.scipy.org/doc/numpy/reference/index.html
QUIP and quippy documentation: http://libatoms.github.io/QUIP/

np.loadtxt("filename") reads in the data
np.eye(n) generates the *n* x *n* identity matrix
np.linalg.solve(A,b) solves the system of linear equations $\mathbf{Ax} = \mathbf{b}$
np.dot(x,y) takes the dot product $\mathbf{xy}$
np.linalg.cholesky(A) returns $\mathbf{L}$ where $\mathbf{LL}^T = \mathbf{A}$
np.diag(A) returns the diagonal elements of matrix $\mathbf{A}$
plt.plot(x,y,"*") plots y vs x points (use plt.show() to actually display the plot)

GAUSSIAN PROCESS REGRESSION

- use numpy functions for this part:
  - useful functions: loadtxt(), exp(), eye(), linalg.solve(), dot(), linalg.cholesky(), log(), diag(), sum()

- fit a GP based on the data in xData_yData.dat
  - compare your fit with the original function in xy.dat
  - **try to estimate the error as well**
  - try varying the hyperparameters
  - **compute the log-likelihood as a function of various hyperparameters**

$$L = -\frac{1}{2}\mathbf{y}^T\mathbf{C}^{-1}\mathbf{y} - \frac{1}{2}\ln\det\mathbf{C} - \frac{N}{2}\ln 2\pi$$

- fit a GP based on the derivative data in xData_yDerivData.dat
  - compare your fit with the original function
  - compute the derivative of the GP and compare it to xyDeriv.dat
- fit a GP based on the sum of function values, in xData1_xData2_yData2D.dat
  - compare the function to the original

LEARNING ATOMIC CHARGES

- load the molecules from data_GDB9.xyz
  - quippy.AtomsList()
- compute the SOAP descriptors of all atoms of an atomic species
  - quippy.Descriptor(“soap atom_sigma=0.5 n_max=6 l_max=6 cutoff=2.5 Z=6 n_species=5 species_Z='1 6 7 8 9’")
  - cutoff(), dimensions(), descriptor_sizes(), calc()
- crossvalidation: split data to training and testing part
- use dot-product/polynomial kernel to compute the covariance matrix
- fit charges (*charge* field in each Atoms object)
- vary the hyperparameters
- **compute the log-likelihood**

FIT A POTENTIAL

- use the command line tool *teach_sparse*
  - http://libatoms.github.io/QUIP/teach_sparse.html
  - use energy and/or force data in data_Si_SW_02.xyz to train (only contains $Si_{n>6}$ clusters)
  - fit two- and three-body terms:
    - distance_2b cutoff=4.1 n_sparse=250 covariance_type=ard_se delta=1.0 theta_fac=0.25
    - angle_3b cutoff=4.1 n_sparse=500 covariance_type=ard_se delta=0.2 theta_fac=0.25
  - load potential to python with quippy.Potential() and "IP SW" as well
  - test on data_Si_SW.xyz
  - plot the two- and three-body terms:
    - calc(at,energy=True,args_str="energy_per_coordinate")
    - use ip.parms.SW2.xml and ip.parms.SW3.xml for reference