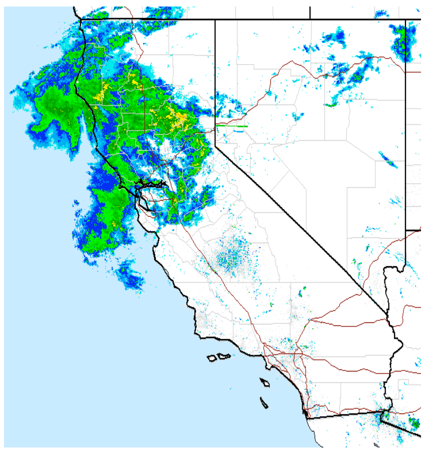
University of California
Department of Civil Engineering

Pozdnukhov, Instructor
Fall 2016

CE 263 Problem #3

Geostatistics / Gaussian Processes



In this task you will work with basic geostatistical methods of spatial prediction and simulation. You will implement and apply your methods to solve 2 practical problems related to predictive modelling of precipitation. Rain fields (Figure 1) show complex behavior that is difficult to reproduce with physical models while data-driven approach can be beneficial in a number of applications.

Python is the recommended programming language for this assignment. You are required **to implement the methods from scratch using basic linear algebra tools** available in NumPy/SciPy packages.

You are required to submit your code in this assignment. The code has to be organized as a python class with prediction, simulation and visualization functionalities (see Appendix 1 for an example).

Geostatistical model of Simple Kriging is equivalent to Gaussian Process (GP) regression model. We will use GP formulation for implementation and a computational approach to covariance function selection. Consider training data $\mathbf{x} = \{\boldsymbol{x}_i\}$, $\mathbf{y}=\{y_i\}$ and a covariance matrix $\mathbf{K}(\mathbf{x}, \mathbf{x})$ with elements $K(\boldsymbol{x}_i,\boldsymbol{x}_j)$ for a chosen covariance function *K(.,.)*.For simplicity, the matrix $\mathbf{K}(\mathbf{x}, \mathbf{x})$ will be denoted simply as $\mathbf{K}$ below. Following the same terminology, $\mathbf{K}(\mathbf{x}^{test}, \mathbf{x}^{test})$ corresponds to the covariance matrix of some other (for instance, test) data set $\mathbf{x}^{test}$, and $\mathbf{K}(\mathbf{x}, \mathbf{x}^{test})$ is a matrix of training-testing covariances correspondingly.

Predictions. Assuming there is additive $N(0,\sigma_n^2)$ noise in data (such as measurement noise), the joint distribution of measured values $\mathbf{y}$ and the values $\mathbf{f}$ in any other locations $\mathbf{x}^{test}$ is a multivariate Gaussian by the definition of a GP:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f} \end{bmatrix} \sim N(\mathbf{0}, \begin{bmatrix} \mathbf{K}+\sigma_n^2\mathbf{I} & \mathbf{K}(\mathbf{x},\mathbf{x}^{test}) \\ \mathbf{K}(\mathbf{x}^{test},\mathbf{x}) & \mathbf{K}(\mathbf{x}^{test},\mathbf{x}^{test}) \end{bmatrix}).$$

The predictive distribution for $\mathbf{f}$ is then a Gaussian $N(m(\mathbf{f}), cov(\mathbf{f}))$ with

$$m(\mathbf{f}) = \mathbf{K}(\mathbf{x}^{test},\mathbf{x})\left[\mathbf{K}+\sigma_n^2\mathbf{I}\right]^{-1}\mathbf{y}, \qquad (1)$$

$$cov(\mathbf{f}) = \mathbf{K}(\mathbf{x}^{test},\mathbf{x}^{test}) - \mathbf{K}(\mathbf{x}^{test},\mathbf{x})\left[\mathbf{K}+\sigma_n^2\mathbf{I}\right]^{-1}\mathbf{K}(\mathbf{x},\mathbf{x}^{test}). \qquad (2)$$

As we have a full predictive distribution at our disposal, numerous useful representations can be obtained. These are the predictive mean $m(\mathbf{f})$, predictive variance diag[$cov(\mathbf{f})$] as well as ability to generate conditional stochastic simulations of $\mathbf{f}$.

Simulations. The goal of stochastic simulations is to generate random vectors $\mathbf{f}^{\text{sim}}$ from $N(\text{m}(\mathbf{f}), \text{cov}(\mathbf{f}))$ where **f** are likely values at a set of locations $\mathbf{x}^{\text{grid}}$ where simulations are required (such as a dense grid covering the desired region). It can be done by computing Cholesky decomposition of cov(**f**), that is $\mathbf{LL}^T$ where **L** is a lower triangular matrix. The vector $\mathbf{f}^{\text{sim}}=\text{m}(\mathbf{f})+\mathbf{Lu}$, where $\mathbf{u}\sim N(0,\mathbf{I})$, then follows the desired $N(\text{m}(\mathbf{f}), \text{cov}(\mathbf{f}))$.

The data. The dataset provided to you in this assignment problem is an hourly rainfall rate in mm/hour measured in California at a given time of a given winter day of 2025. The meteorological sensing network covering California contains 827 sensors. Of all the 827, you are provided with the measurements in 414 locations (trn_data.csv). Your task in Part 1 is to provide a prediction of the hourly rainfall rate for the other 413 locations (tst_locations.csv) reserved as test sites.

Part 1. Predictions (50 points).

Implement a GP regression method with an isotropic Gaussian covariance function of a given bandwidth. The class implementing GP has to contain a function returning a predictive mean (1) (it can also contain as many auxiliary functions as you need, of course). The training dataset and the value of the bandwidth parameter have to be provided by a user.

Train the model on the dataset contained in trn_data.csv file. Find an optimal value of the bandwidth parameter. Instead of fitting a covariance/variogram to empirical values, consider a computational approach of minimizing the k-fold cross-validation error on the training set. Compute predictions for the locations contained in the file tst_locations.csv. The quality of your predictions will be measured by RMSE.

Your predictions must be submitted via a respective Kaggle competition web page.

See the competition page for format details.

Part 2. Simulations (30 points).

Implement a function returning one conditional stochastic simulation (i.e. $\mathbf{f}^{\text{sim}}=\text{m}(\mathbf{f})+\mathbf{Lu}$**,** with **L** as un upper-triangular matrix of LU-decomposition of (2)) at a set of locations $\mathbf{x}^{\text{grid}}$ provided as an argument to the function. File grid.csv contains a sample 50x50 grid within the [38.5, 39.3, -119.8, -120.8] bounding box. Appendix 2 below contains useful code samples.

Part 3. Visualization (20 points).

Implement a function to visualize one stochastic simulation for the area within the following bounding box: [38.5, 39.3, -119.8, -120.8]. This function has to produce and write to disk an image and a ground overlay KML file with a static geo-referenced image of a realization your stochastic simulation (see Appendix 3).

Your submission must be a single ZIP archive that contains:

1. Python file with the source code of your GP class implementation.
2. A short report describing:

    a) your approach to predictions, including the choice of the covariance function and it's bandwidth, and the noise variance σ_n^2,

    b) a screenshot of the Google Earth with a KML overlay visualizing one realization of a stochastic simulation.

Appendix 1: Python classes

Basic class example:

File predictor.py

```
import numpy as np
from pylab import *

class DumbPredictor:

  def __init__(self, data):
    self.mu = np.mean(data)
    self.sigma = np.std(data)

  def predict(self):
    return self.mu

  def simulate(self, N):
    return self.sigma*np.random.normal(0,1,N) + self.mu

  def visualize(self, filename, show=False):
    x = self.simulate(100)
    plt.hist(x, facecolor='green', alpha=0.5)
    plt.title("Hist: $\mu=%.2f$, $\sigma=%.2f$" % (self.mu, self.sigma))
    plt.savefig(filename)
    if show:
      plt.show()
```

Usage:

```
#!/usr/bin/env python

from numpy import random
from predictor import DumbPredictor

def TellMeAboutTheData(data, show=False):

  predictor = DumbPredictor(data)
  print "My best guess is %f" % predictor.predict()
  print "but it is so uncertain... %s ?" % str(predictor.simulate(10))
  print "I better plot a picture."

  predictor.visualize('histogram.png', show)

if __name__=="__main__":

    mydata = 3+0.5*random.randn(50)
    TellMeAboutTheData(mydata, True)
```

Appendix 2: Numerical linear algebra

Loading *.csv files:

```
import numpy as np

data = np.genfromtxt('trn_data.csv', delimiter=',', skiprows=1)
X, Y = data[:,:-1], data[:,-1:]
```

Rectangular grid with *ncell* nodes at each dimension:

```
def make_grid(bounding_box, ncell):

  xmax, xmin, ymax, ymin = bounding_box
  xgrid = np.linspace(xmin, xmax, ncell)
  ygrid = np.linspace(ymin, ymax, ncell)

  mX, mY = np.meshgrid(xgrid, ygrid)
  ngridX = mX.reshape(ncell*ncell, 1);
  ngridY = mY.reshape(ncell*ncell, 1);

  return np.concatenate((ngridX, ngridY), axis=1)

bounding_box = [38.3, 39.3, -120.0, -121.0]
Xgrid = make_grid(bounding_box, 50)
```

Computing Gaussian covariance:

```
import numpy as np
from scipy import spatial

def covariance(self, X, Z, h):
    d = spatial.distance_matrix(X,Z)
    K = np.exp(-(d**2) / (2*h*h))
    return K
```

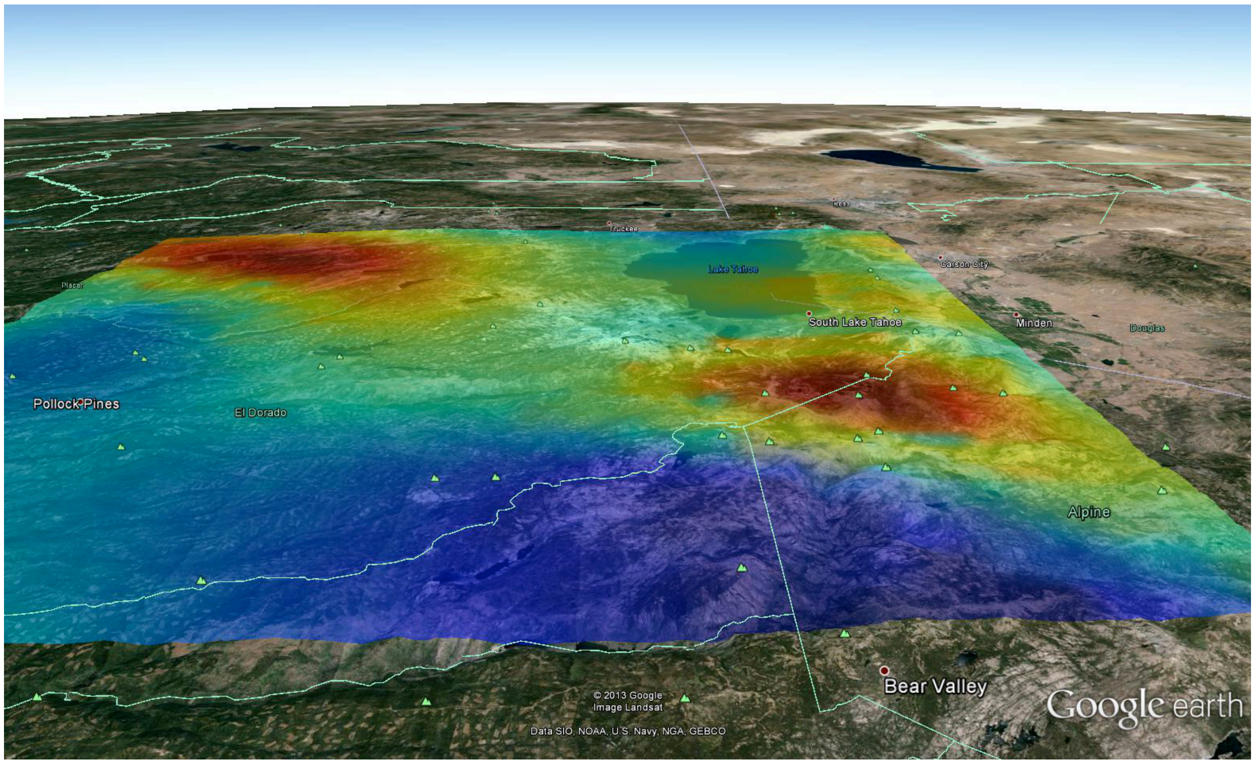
Cholesky decomposition:

# note that gamma***I** is added to K for numerical stability, gamma ~ 0.001 (!)

```
L = np.linalg.cholesky(K + gamma*np.eye(K.shape[0]))
```

General reference to Numpy/Scipy linear algebra methods:
http://docs.scipy.org/doc/numpy/reference/routines.linalg.html

Appendix 3. Sample output of a simulation as a KML overlay visualized in Google Earth.


Truckee
Lake Tahoe
Carson City
Placer
South Lake Tahoe
Minden
Douglas
Pollock Pines
El Dorado
Alpine
Bear Valley
© 2013 Google
Image Landsat
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Google earth