MSc Mathematical and Computational Finance

Introduction to Statistics - Problem Sheet 'Week 3'

1. In this exercise you will estimate an implied volatility surface using a two-dimensional kernel. Your answers should show the code from Python for functions `countDaysFromTenor`, `strikeFromDelta` and `volEstimate`.

   Please keep the Python code as it will be re-used in forthcoming assignments.

   (a) Download the following file `EURUSD_volsurface.csv` and save it as dataframe `df_vols`. This data contains a snapshot of the implied volatility surface for the exchange rate of EURUSD. The information is displayed in terms of **tenors** (representing the option maturity) and deltas. So a `[25D Call EUR]` column displays the implied volatilities for the 25 delta EURUSD Call option with various expiries. A `'1M'` tenor would correspond to an option expirying in 30 days.

   (b) Write a function `countDaysFromTenor` in Python that reads the tenors and returns the equivalent number of days. You can assume 30 days for months and 365 days for years. For example, if the array of tenors is `[1W,2W,4M,2Y]`, your function should return `[7,14,120,730]`.

   (c) Assuming zero interest rates (hence zero drift, and discount factor as 1), the delta for a vanilla Option is given by:

$$\Delta = \frac{\partial O}{\partial S} = cp * N(cp * d_1)$$

$$N(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp(-\frac{z^2}{2})dz$$

$$d_1 = \frac{log(S/K) + \sigma^2 T/2}{\sigma\sqrt{T}}$$

   Where $cp = 1$ for a call option, $cp = -1$ for a put option and $N(x)$ is the standard cumulative normal distribution. Write a function `strikeFromDelta` in Python that given [Δ,S,σ,T,cp]: S as initial spot, σ as implied volatility, T as time to maturity (expressed as a fraction of the year, e.g. for a `'1M'` this would be 30/365), returns the equivalent strike K.

   (d) Use the function `strikeFromDelta` to create a dataframe with all strikes from delta for the volsurface `df_vols` assuming $S = 1.166$ and with all interest rates set to zero. Notice that for a `25D`, the delta needs to be expressed as $\Delta = 0.25$, for example.

(e) In order to scale the parameters, calculate the **moneyness** m equivalent for all maturities and deltas in the dataframe, that is, dividing all strikes by initial spot: $m = K/S$.

(f) Using maturity T and moneyness m as inputs, define a non-parametric Nadaraya-Watson estimator for the implied volalitity surface $V(m, T)$ as follows:

$$\hat{V}(m, T) = \sum_{i=1}^{N} \frac{V(m_i, T_i) g(m - m_i, T - T_i)}{\sum_{i=1}^{N} g(m - m_i, T - T_i)}$$

where

$$g(x, y) = (2\pi)^{-1} \exp(-x^2/2h_1) \exp(-y^2/2h_2)$$

is a Gaussian Kernel and $V(m_i, T_i)$ are the data points given by the volatilities in the dataframe. Write a function `volEstimate` in Python to represent this estimate, given (m, T). There are two bandwidth parameters $h1$ and $h2$ which can be optimally chosen, but for this exercise assume $h1 = h2 = 0.05$. Use this estimator `volEstimate` to find the implied volatlity at the internal new points:

```
mx = np.linspace(m_min,m_max,10)
ty = np.linspace(t_min,t_max,10)
```

Notice: N is the total number of data points. In this example, the number of maturities is 24 and number of strikes/moneyness 5, hence $N = 120$.

(g) **Optional for experts**: plot the 3D graph of the volatility surface using $50X50$ points for moneyness and maturity (use internal points only). You may find the following code useful:

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
import numpy as np
from sys import argv


fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
X,Y = np.meshgrid(mx,ty)
Z = tempVol.reshape(X.shape)

surf = ax.plot_surface(X, Y, Z, cmap=cm.jet)
fig.colorbar(surf, shrink=0.5, aspect=5)

ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')

plt.show()
```

2. Data Analysis task: there is no need to display Python code in this exercise, but you need to display graphs (correctly labelled) and explain your conclusions, writing a coherent report on your model assumptions and diagnostics. Download the files `Python_DataQ2` and save it as a dataframe `df_WLS`.

   (a) Regress `y_data` on x using a standard OLS linear regression. Display the fitted values graph against the original data as well as the residual against the inputs x. What are your observations regarding the residuals?

   (b) Use the weights from the dataframe `df_WLS['weights']` to create a Weighted Linear Regression. These weights are proportional to the standard deviations of distribution of the errors. Display the estimated coeficients from WLS alongside the ones from OLS and their respective standard deviations.

   (c) Plot the predicted interval, alongside the fitted values and orginal data in a single graph. What can you observe?