# 悪いゾーニングの例

人・仕事のpreferenceベクトルを反転させて、シミュレーションを実施することで、悪いゾーニングの例を簡単に作成してみた。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0.301586 | 0.300303 | 0.317958 | 0.29794 | 0.317328 |

# 特徴量の抽出

良い／悪いゾーニングについて、特徴量を適当に定義して比較してみよう。これを、例えばロジスティック回帰などで学習することが出来れば、逆に良いスコアの特徴量のパターンを得ることができるはず！

特徴量は、隣接セルの生起確率のみを数値化した。つまり、以下のような感じ。対称行列である。

**良いゾーニングの場合：**

**悪いゾーニングの場合：**

後は、ランダムなやつも。

後、適当なpreferenceにして、

さらに、ランダムで生成したやつを追加して、、、、

# Linear Regression

まずは、無理だろうとは思いつつ、線形回帰を実施してみる。良いデザイン、悪いデザインなどを46個集めて、以下の式に当てはめる。

ただし、は特徴量ベクトルで、上で述べた行列の上三角部分を一列にしたベクトル。は線形回帰パラメータ。もし線形で表現できるなら、このによって、与えられたデザインのスコアを簡単に推測できるはずだ。使用したコードは、

D:\GitHub\ZoningSim\LinearRegression\linear\_regression.py

結果、。また、residue（誤差）は0.07756。

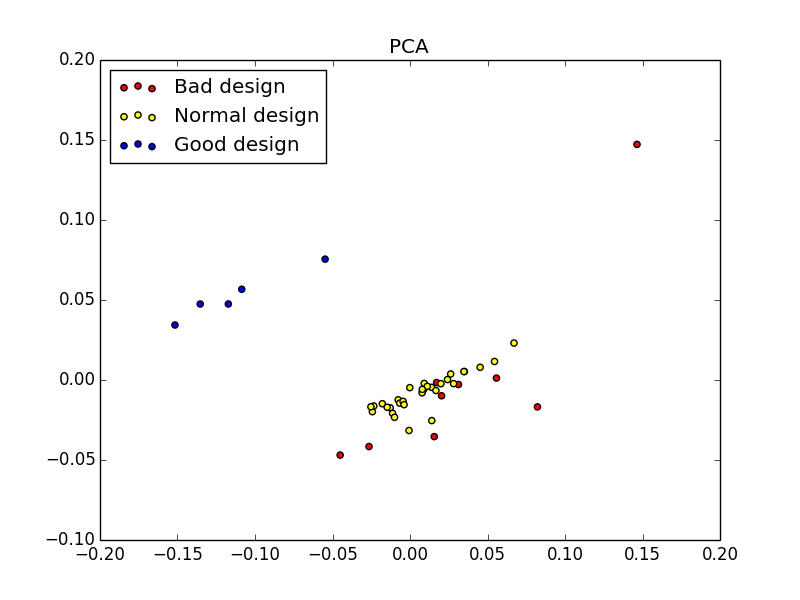
データ数が増えれば、誤差がさらに増加することが予想される。（だって、実際のデータの数はという膨大な数だから）

# PCA

使用したコードは、以下。

D:\GitHub\ZoningSim\LinearRegression\pca.py

同じフォルダ内のdata.txtを読み込んで、PCA分析する。結果は下図の通り。左上にgood design、真ん中にnormal design、右下にbad designが比較的きれいに並んだ。これは、特徴量ベクトル空間ではsmoothであることを示唆している。



つまり、この２次元空間でベストを探し、対応する特徴量ベクトルを算出し、さらに、対応するゾーニングを生成するということができれば、ベストのゾーニングを効率的に生成出来るはずだ。