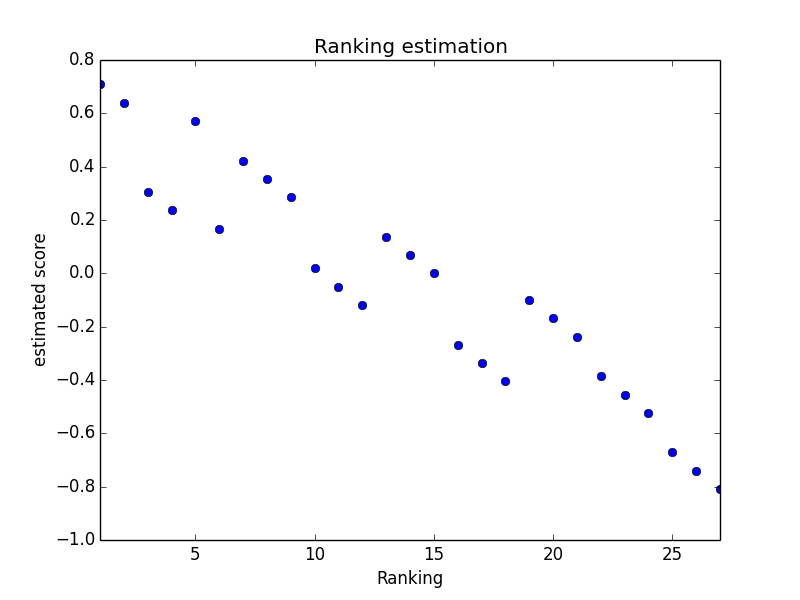
# Pairwise comparison

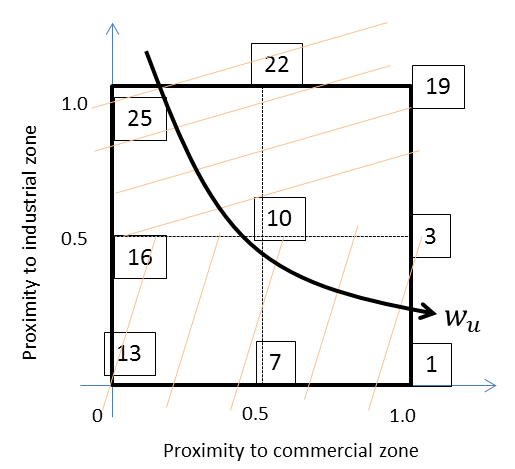
本当に我々の線形モデルで、ユーザのpreferenceを表現できるのか？実際に3次元のケースでやってみた。商業地への近さ、工業地への近さ、公園への近さの3つの要素について、それぞれ1.0、0.5、0.0の値（合計通り）についてランキングを作成してみた。

|  |  |  |  |
| --- | --- | --- | --- |
| 商業地への近さ | 工業地への近さ | 公園への近さ | ランキング |
| 1.0 | 1.0 | 1.0 | 19 |
| 1.0 | 1.0 | 0.5 | 20 |
| 1.0 | 1.0 | 0.0 | 21 |
| 1.0 | 0.5 | 1.0 | 3 |
| 1.0 | 0.5 | 0.5 | 4 |
| 1.0 | 0.5 | 0.0 | 6 |
| 1.0 | 0.0 | 1.0 | 1 |
| 1.0 | 0.0 | 0.5 | 2 |
| 1.0 | 0.0 | 0.0 | 5 |
| 0.5 | 1.0 | 1.0 | 22 |
| 0.5 | 1.0 | 0.5 | 23 |
| 0.5 | 1.0 | 0.0 | 24 |
| 0.5 | 0.5 | 1.0 | 10 |
| 0.5 | 0.5 | 0.5 | 11 |
| 0.5 | 0.5 | 0.0 | 12 |
| 0.5 | 0.0 | 1.0 | 7 |
| 0.5 | 0.0 | 0.5 | 8 |
| 0.5 | 0.0 | 0.0 | 9 |
| 0.0 | 1.0 | 1.0 | 25 |
| 0.0 | 1.0 | 0.5 | 26 |
| 0.0 | 1.0 | 0.0 | 27 |
| 0.0 | 0.5 | 1.0 | 16 |
| 0.0 | 0.5 | 0.5 | 17 |
| 0.0 | 0.5 | 0.0 | 18 |
| 0.0 | 0.0 | 1.0 | 13 |
| 0.0 | 0.0 | 0.5 | 14 |
| 0.0 | 0.0 | 0.0 | 15 |

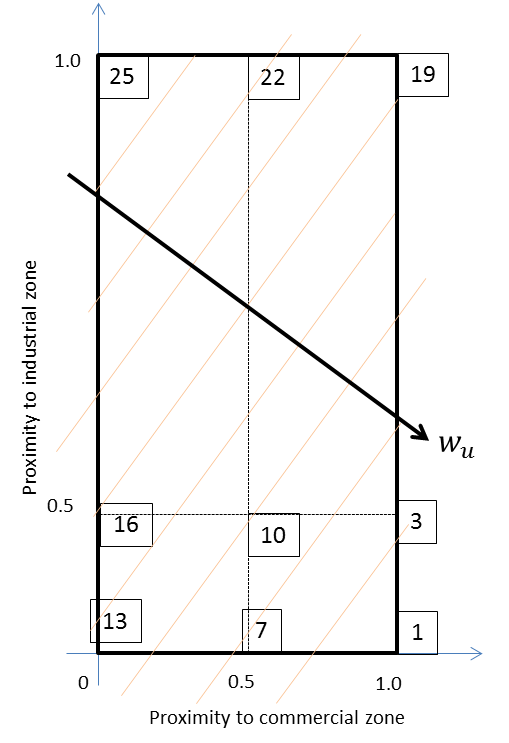
これに基づいて、pairwise comparisonの結果を作成し、Gradient Descentを使ってpreferenceベクトルを推定すると、を得た。また、**accuracy=0.917379**。これを使って上の27通りのオプションのスコアを計算すると、下図のようにいくつか予測エラーが発生していることが分かる。この原因について考察してみた。



例えば、公園への近さが1.0の条件、つまり、商業地への近さと工業地への近さの2次元で考えてみる。ランキングは下図のようになる。これを見ると分かるとおり、直線ではpreferenceベクトルを表現できない。



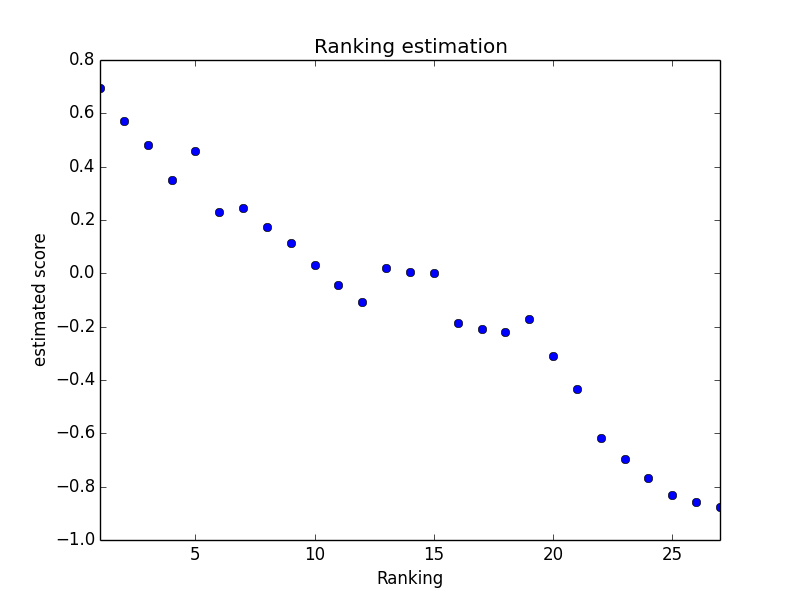
直線で表現するためには、例えば工業地への近さについて、0.5と1.0の距離を広げるという案がある。



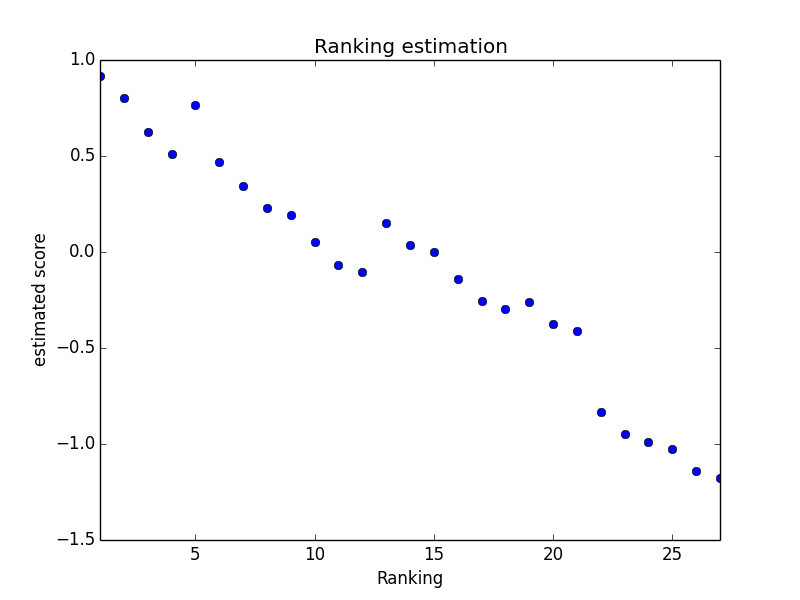
# Kernel Trick

preferenceベクトルを線形で表現できないとなると、まず思いつくのは高次元化だ。というわけで、まずはquadratic kernelを試してみた。

**Quadratic kernel:** 3次元データなので、quadratic kernelは、だ。結果は、以下の通り。**Accuracy=0.968661**で、linear kernelよりも改善された。



**Modified quadratic kernel:** ちょっと違うquadratic kernelを試してみた。。結果は以下の通り。**Accuracy=0.97151**で、さらに改善された。でも、グラフの形としては、quadratic kernelの方が良い感じに見えるよねぇ。。。



# Featureの要素のRanking

Featureの要素（これまでの例では3つの要素、商業地・工業地・公園までの近さ）のrankingを求めるだけというやり方もある。この場合、次元数に対して、必要なcomparisonはとなり、非常に少なくて嬉しい。しかし、要素のrankingでは、ユーザのpreferenceを正確に表現できないはず。ということを実験で確認したい。

ただし、工業地については、ネガティブな関係にあるので、featureベクトルとしては、マイナスした要素を追加して、6次元とする必要がある。例えば、linear kernelの例だと、ユーザのpreferenceベクトルはだった。マイナス要素を加えると、となり、要素のランキング（重要度）は、**「工業地への遠さ＞商業地への近さ＞公園への近さ」**となる（ネガティブな要素は除外した）。

それでは、アイテムを比較してみよう。例えば、となら、商業地への近さで勝るが良いことが分かる。また、となら、工業地への遠さの方が重要なので、が良いことが分かる。しかし、との比較になると、工業地への遠さの重要度が、商業地への近さの重要度に対して、どの程度重要なのかが分からない限り、判定できない。

試しに、適当にpreferenceベクトルをとしてみた。つまり、要素のランキングに従って重要度が２倍に増えていく感じだ。これで全てのアイテムの組合せについて予測してみると、**accuracy=0.903134**となった。ん？意外に良いねぇ。でも、グラフ化してみると、下図の通り。やはり、だいぶ間違っているというのが分かる。

