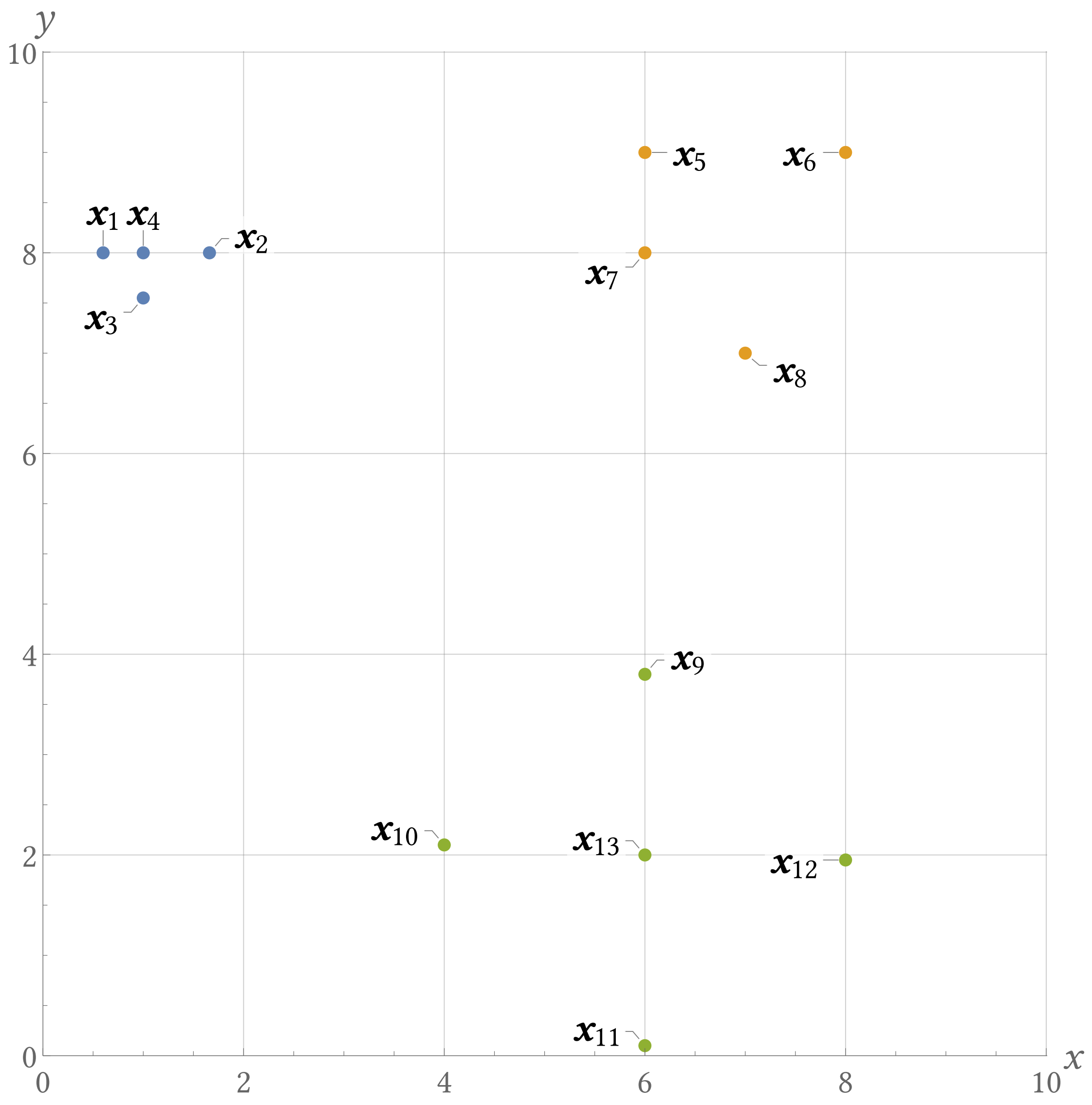
t-Distributed Stochastic Neighbor Embedding

Theme

Initialization

```
In[13]:= data = {
            {0.6, 8}, {1.66, 8}, {1, 7.55}, {1, 8},
            {6, 9}, {8, 9}, {6, 8}, {7, 7},
            {6, 3.8}, {4, 2.1}, {6, 0.1}, {8, 1.95}, {6, 2}
          };
         dataLabels = Subscript[bi["x"], #] & /@ Range[Length[data]];

         color1 = 🟦;
         color2 = 🟧;

In[17]:= SeedRandom[1337];
         clusters = ClusteringComponents[data, 3, 1]

Out[18]= {1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3}

In[19]:= dataCluster = Table[
           Table[x, {x, data[[Position[clusters, c] // Flatten]]}]
           , {c, 1, 3}
          ]

Out[19]= {{{0.6, 8}, {1.66, 8}, {1, 7.55}, {1, 8}},
          {{6, 9}, {8, 9}, {6, 8}, {7, 7}}, {{6, 3.8}, {4, 2.1}, {6, 0.1}, {8, 1.95}, {6, 2}}}
```

```
In[111]:= plotInit = Show[
            ListPlot[
             dataCluster,
             PlotTheme → "myTheme",
             GridLines → Automatic,
             AspectRatio → 1,
             PlotRange → {{0, 10.01}, {0, 10.01}},
             AxesLabel → {x, y}
            ],
            ListPlot[
             MapThread[Callout[#1, #2] &, {data, dataLabels}],
             PlotStyle → None,
             PlotTheme → "myTheme"
            ]
          ]
```



Part I: Data Distribution

In[21]:= `p`$_\sigma$`_[j_, i_] :=`

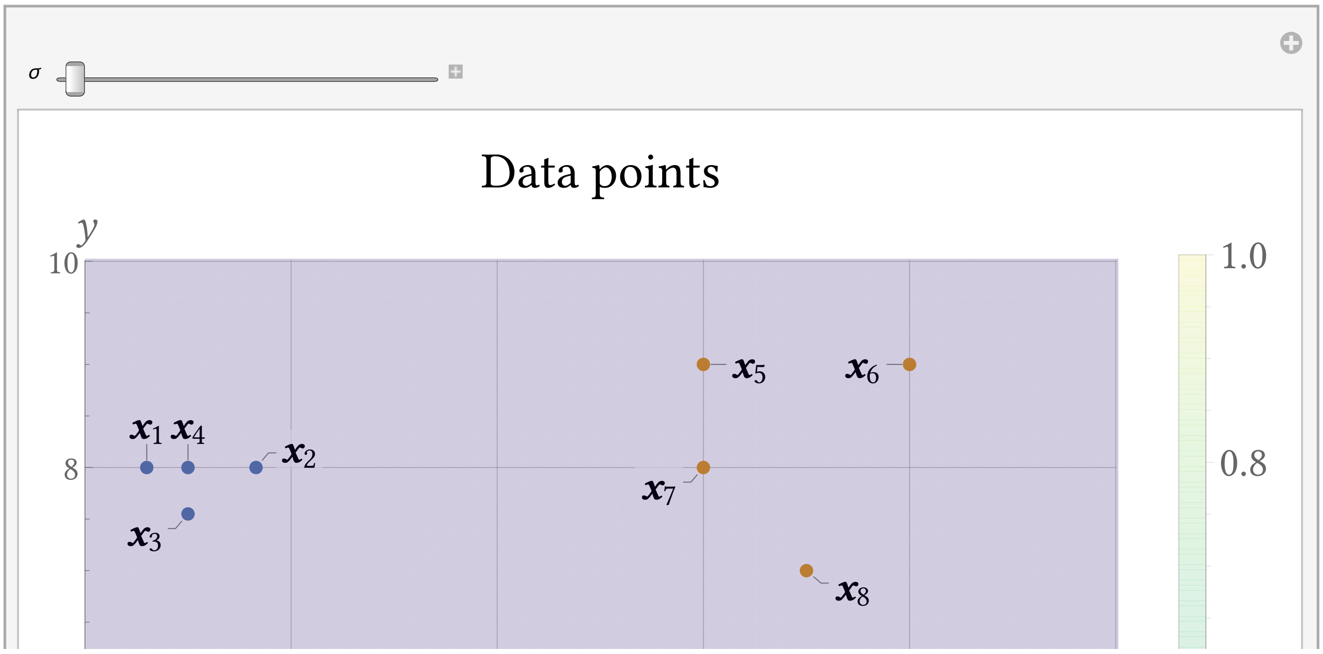
$$\begin{cases} 0 & \texttt{i == j} \\ \dfrac{e^{-\frac{\texttt{Norm[data[[i]]-data[[j]]]}^2}{2*\sigma^2}}}{\sum_{\texttt{k=1}}^{\texttt{Length[data]}} \begin{cases} 0 & \texttt{i==k} \\ e^{-\frac{\texttt{Norm[data[[i]]-data[[k]]]}^2}{2*\sigma^2}} & \texttt{True} \end{cases}} & \texttt{True} \end{cases}$$

Parameter σ_i

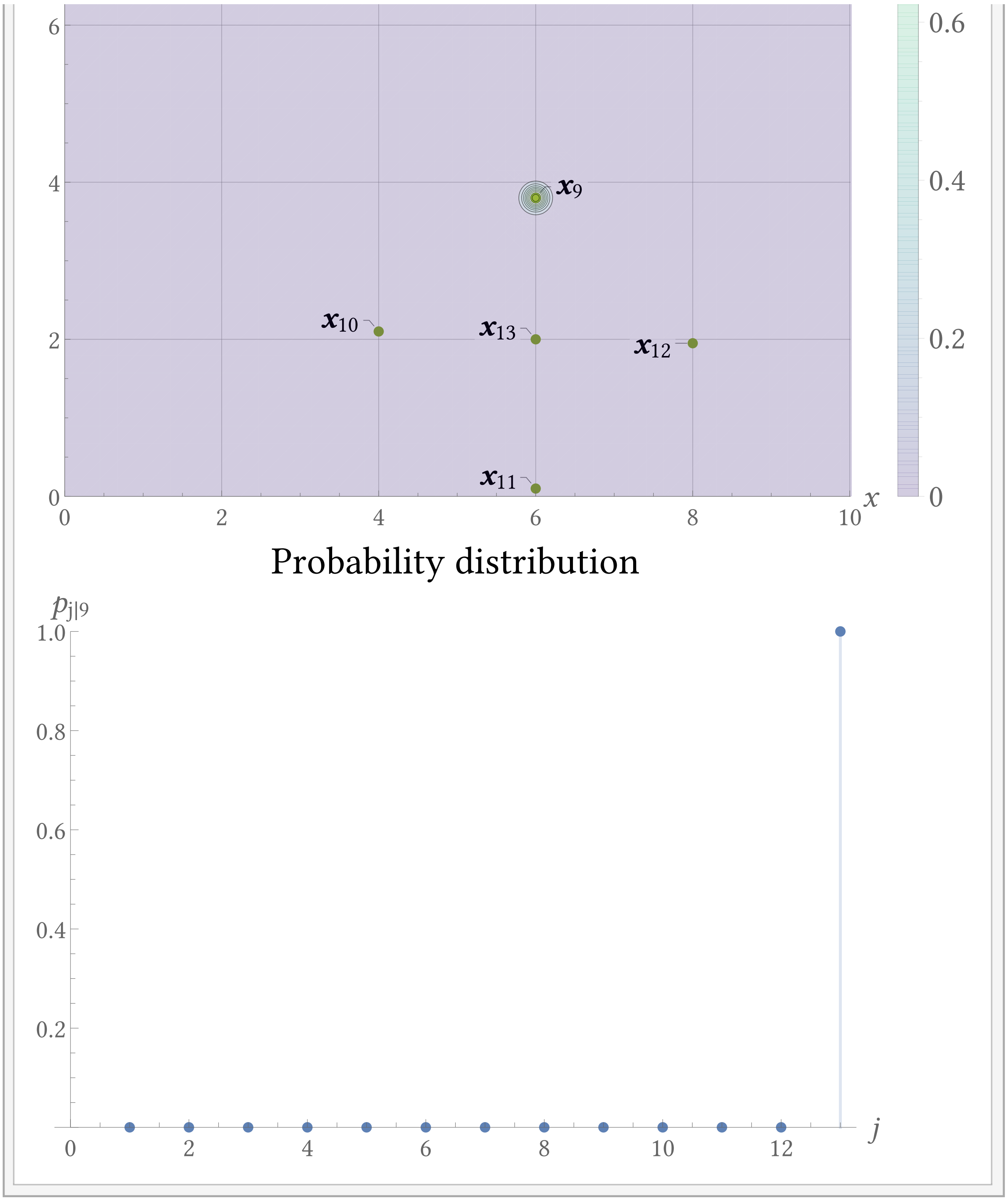
```
In[22]:= cf[z_] := {Opacity[0.2], ColorData["BlueGreenYellow"][z]}
```

In[23]:= $g_{\sigma\_}$[p1_, p2_] := $e^{-\frac{\text{Norm[p1-p2]}^2}{2*\sigma^2}}$

```
In[24]:= p3 = 9;

In[112]:= plotDist[σ_] := Block[{plotPoints, plotProb},
            plotPoints = Show[
              plotInit,

              ContourPlot[gσ[data[[p3]], {x1, x2}], {x1, 0, 11}, {x2, 0, 11},
               PlotTheme → "myTheme",
               AspectRatio → 1,
               ColorFunction → (cf[#] &),
               ColorFunctionScaling → False,
               PlotLegends → BarLegend[{cf[#] &, {0, 1}}, LegendMargins → {{0, 0}, {0, 46}}],
               PerformanceGoal → "Quality",
               PlotPoints → 50,
               PlotRange → {0, 1}
              ],

              PlotLabel → "Data points"
             ];
            plotProb = DiscretePlot[{pσ[j, p3]}, {j, 1, Length[data]},
              PlotTheme → "myTheme",
              PlotRange → {0, 1},
              PlotLabel → "Probability distribution",
              PerformanceGoal → "Quality",
              AxesLabel → {it["j"], Subscript[it["p"], "j|" <> ToString[p3]]}
             ];

            Column[{
              plotPoints,
              plotProb
             }]
           ]

In[113]:= Manipulate[
           plotDist[σ]
           , {σ, 0.1, 5}]
```



Out[113]=



```
(*Export[
  FileNameJoin[{
    NotebookDirectory[],
    "frames/sigma=00.png"
   }],
  Table[plotDist[σ],{σ,0.1,5,0.05}],
  "VideoFrames",
  Antialiasing→True
 ];*)
```

The conditional probabilities sum indeed up to 1:

In[28]:=
```
Table[pσ[j, p3], {j, 1, Length[data]}] // Total // Simplify
```

Out[28]=
```
1
```

because the denominator is the same for each $p_{j|i}$ and when building the sum, the numerator becomes equal to the denominator. They must be positive due to the exponential function.

- There must always be at least one peak since $p_{j|i}$ is a valid probability distribution and hence must sum up to 1. The strongest peak always corresponds to the nearest data point ($\boldsymbol{x}_{13}$ in this case).
- For the extreme case
  - $\lim_{\sigma_9 \to 0}$ we get a one-hot vector (only the nearest neighbour is considered) [this question has not been asked but maybe it is nice to know :-)].

```
In[29]:= Table[Limit[pσ[j, p3], σ → 0], {j, 1, Length[data]}]

Out[29]= {0., 0., 0., 0., 0., 0., 0., 0., 0, 0., 0., 0., 1.}
```

  - $\lim_{\sigma_9 \to \infty}$ we get an almost uniform distribution with $p_{9|9} = 0$ and $p_{j \neq 9|9} = \frac{1}{n-1}$ (all neighbour points are considered equally).

```
In[30]:= Table[Limit[pσ[j, p3], σ → ∞], {j, 1, Length[data]}]

Out[30]= {0.0833333, 0.0833333, 0.0833333, 0.0833333, 0.0833333, 0.0833333,
          0.0833333, 0.0833333, 0, 0.0833333, 0.0833333, 0.0833333, 0.0833333}
```

- We always pick neighbours relative to some base point considering the density around the base point. But since the density around each point is different, the corresponding probabilities may be different as well. The critical factor is the scaling σ_i which is chosen per point separately.

Perplexity

```
In[31]:= Hσ_[i_] := - Sum[Piecewise[{{0, pσ[j, i] == 0}, {pσ[j, i] * Log2[pσ[j, i]], True}}], {j, 1, Length[data]}]
         perpσ_[i_] := 2^Hσ[i]
```

$$H_{\sigma\_}[i\_] := -\sum_{j=1}^{\text{Length[data]}} \begin{cases} 0 & p_\sigma[j,i] == 0 \\ p_\sigma[j,i] * \text{Log2}[p_\sigma[j,i]] & \text{True} \end{cases}$$

$$\text{perp}_{\sigma\_}[i\_] := 2^{H_\sigma[i]}$$

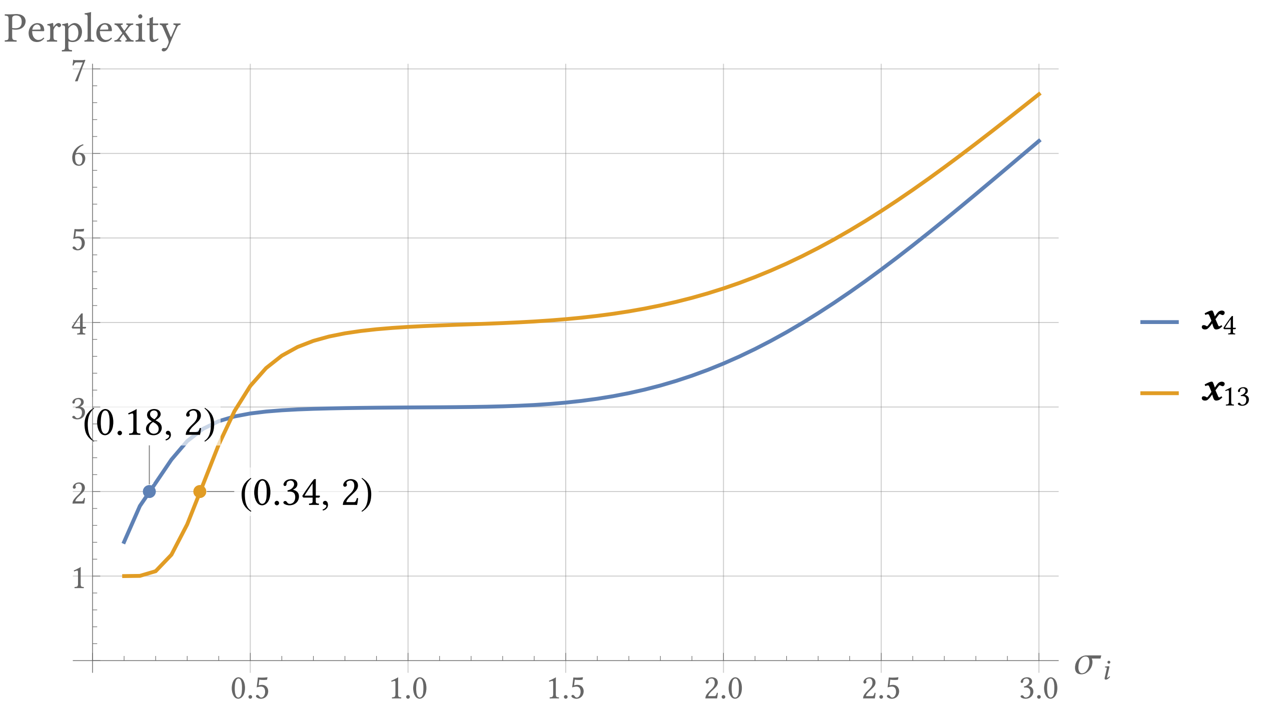
The approach here to find the σ_i values is to test a few values in a range and select the closest one.

```
In[33]:= perpUser = 2;
         sigmas = Table[
             Nearest[
              Table[perpσ[i] → σ, {σ, 0.1, 3, 0.02}],
              perpUser
             ],
             {i, 1, Length[data]}
           ] // Flatten

Out[34]= {0.36, 0.32, 0.26, 0.18, 0.96, 0.52, 0.76, 0.88, 0.96, 0.94, 1., 0.94, 0.34}

In[35]:= p1 = 4;
         p2 = 13;

In[37]:= Clear[plotPerp]
```

In[98]:=
```
plotPerp[σMax_: 3] := Show[
  ListLinePlot[{
    Table[{σ, perpσ[p1]}, {σ, 0.1, σMax, 0.05}],
    Table[{σ, perpσ[p2]}, {σ, 0.1, σMax, 0.05}]
   },
   PlotTheme → "myTheme",
   GridLines → Automatic,
   AxesLabel → {"σi", "Perplexity"},
   PlotLegends → {Subscript[bi["x"], p1], Subscript[bi["x"], p2]}
  ],

  ListPlot[{
    {Callout[{sigmas[[p1]], perpUser},
      "(" <> ToString[sigmas[[p1]]] <> ", 2)", Above, LeaderSize → 20]},
    {Callout[{sigmas[[p2]], perpUser}, "(" <> ToString[sigmas[[p2]]] <> ", 2)",
      After, LeaderSize → 14]}
   },
   PlotTheme → "myTheme"
  ]
 ]
```

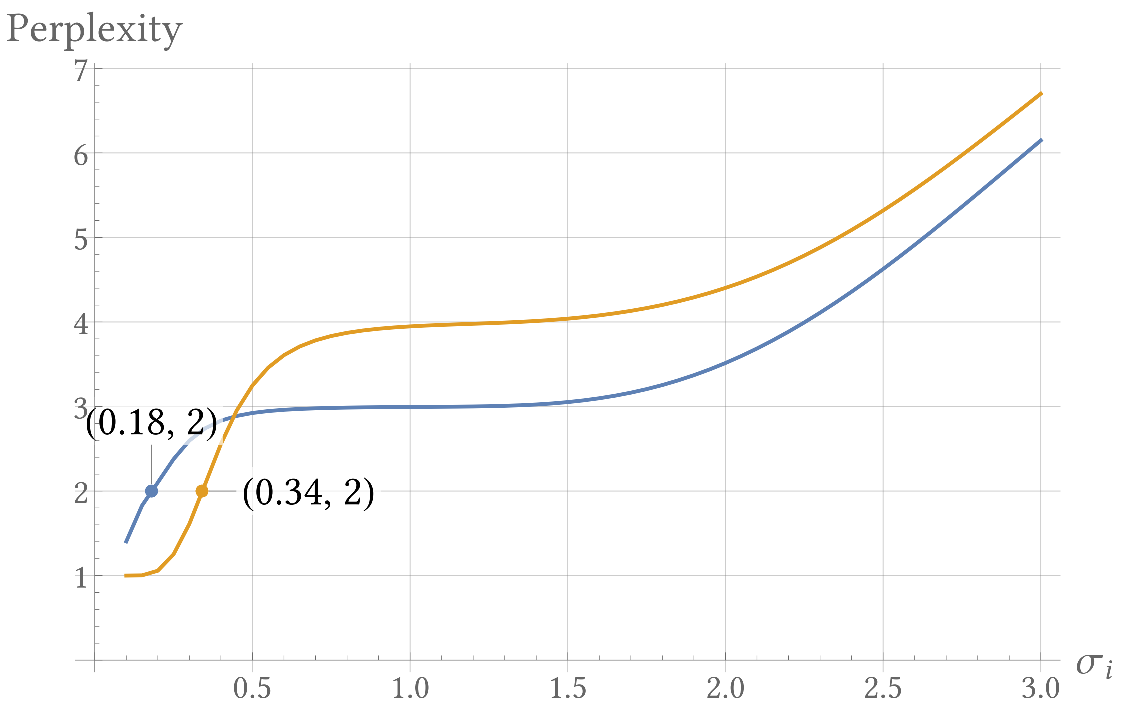
In[99]:=
```
plotPerp[]
```

Out[99]=



Both curves reach the plateau when the corresponding Gaussians cover all points in the near neighbourhood. The blue curve reaches the plateau first since $\boldsymbol{x}_4$ has only 3 near neighbours in the surrounding whereby $\boldsymbol{x}_{13}$ has 4 neighbours. After covering these nearby neighbours, we need to overcome some distance first before reaching the other clusters (so, the plateau comes from the space between the clusters).

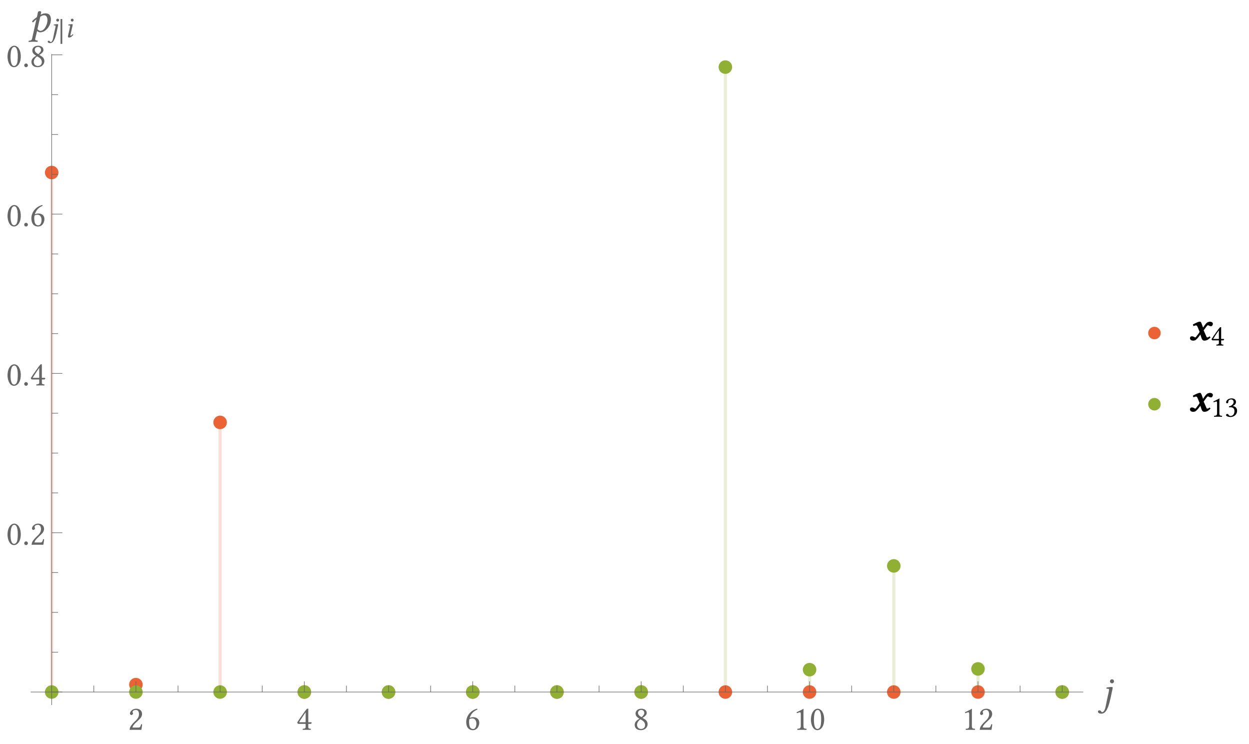
In[101]:=
```
plotPerp[] /. {Subscript[bi["x"], p1] → "B", Subscript[bi["x"], p2] → "A"}
```



In[107]:=
```
plotPerpProb[c1_: color1, c2_: color2] :=
 DiscretePlot[{p_sigmas[[p1]][j, p1], p_sigmas[[p2]][j, p2]}, {j, 1, Length[data]},
  PlotTheme → "myTheme",
  PlotRange → All,
  PlotStyle → {c1, c2},
  PlotLegends → PointLegend[{c1, c2},
    {Subscript[bi["x"], p1], Subscript[bi["x"], p2]}, LegendMarkerSize → 13],
  AxesLabel → {it["j"], "p_j|i"}
 ]
```

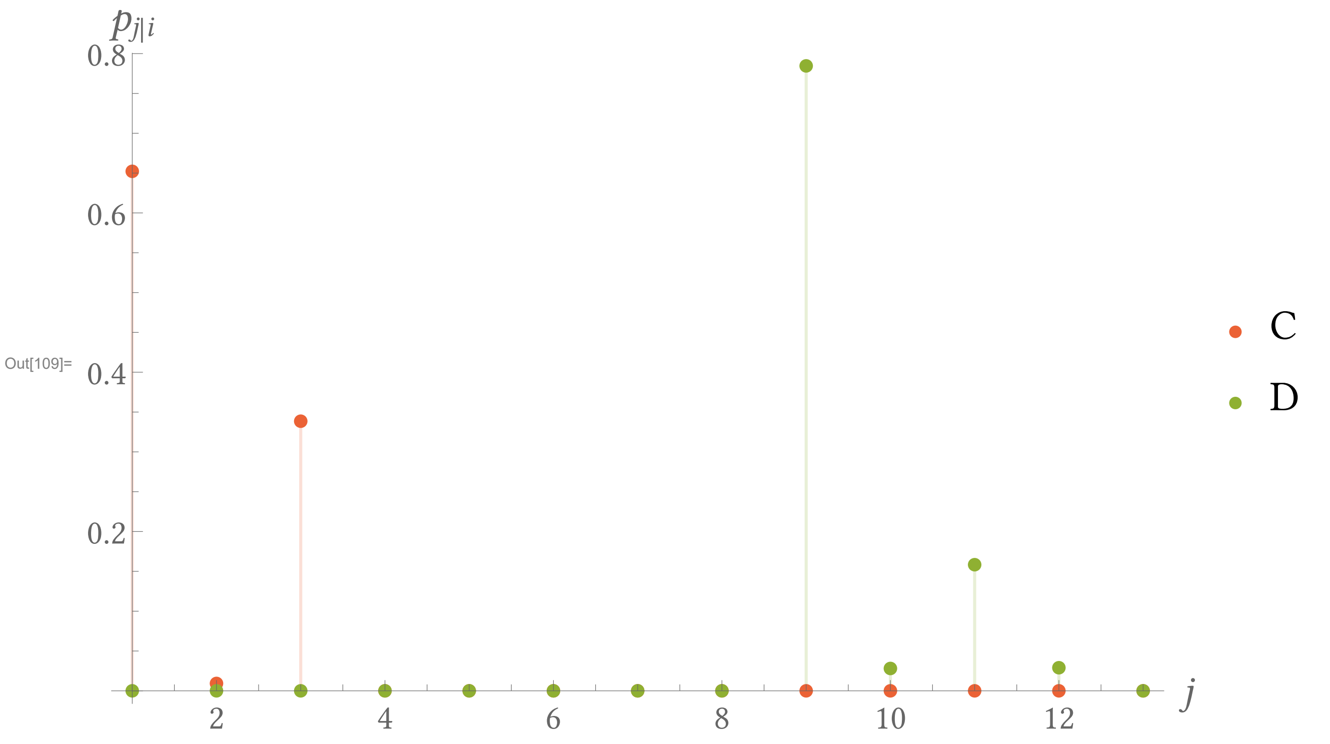
In[108]:=
```
plotPerpProb[🟥, 🟩]
```

Out[108]=



In[109]:=
```
plotPerpProb[🟥, 🟩] /. {Subscript[bi["x"], p1] → "C", Subscript[bi["x"], p2] → "D"}
```

Out[109]=



In[110]:=
```
TableForm[{
   {x4, 0.18, "C", Hsigmas[[p1]][p1], perpsigmas[[p1]][p1], "B"},
   {x13, 0.34, "D", Hsigmas[[p2]][p2], perpsigmas[[p2]][p2], "A"}
  }, TableHeadings →
   {None, {"Data point", "Scaling", "Distribution", "Entropy", "Perplexity", "Curve"}}]
```

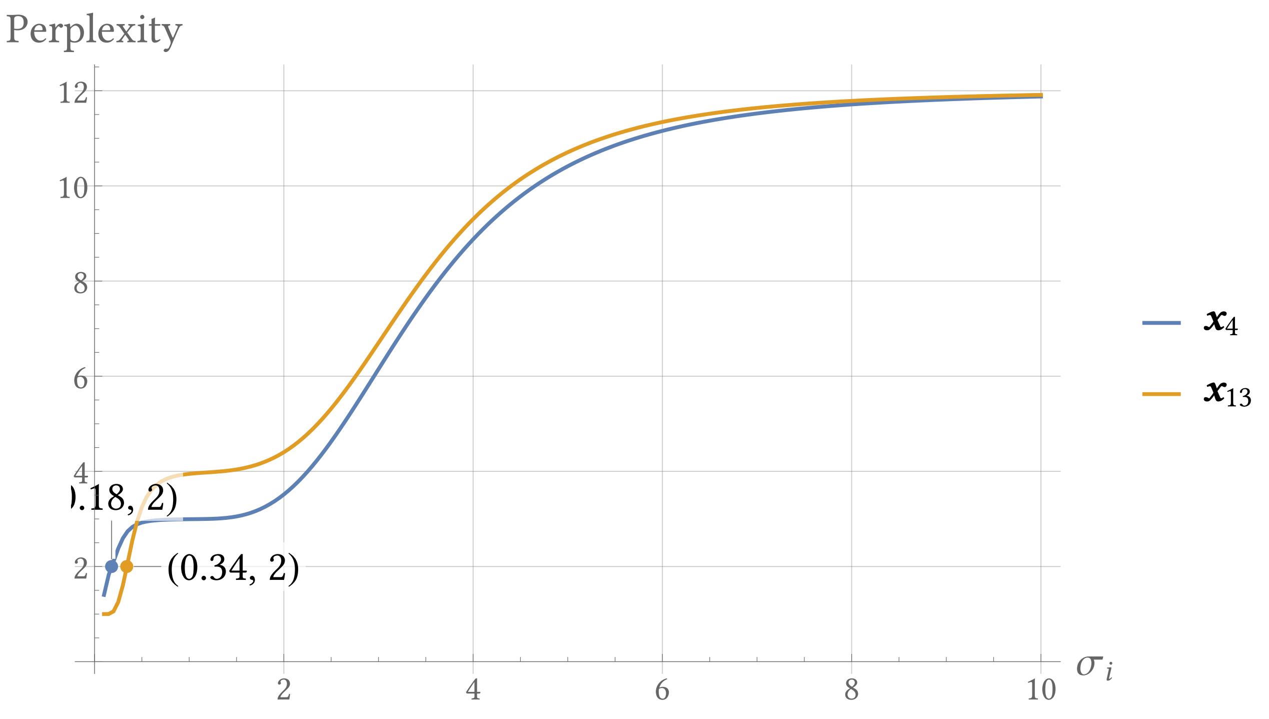
Out[110]//TableForm=

Data point	Scaling	Distribution	Entropy	Perplexity	Curve
x_4	0.18	C	0.993761	1.99137	B
x_{13}	0.34	D	0.988433	1.98403	A

How do the curves evolve for higher σ? The thing to know is that the perplexity is a smooth measure for the number of neighbours and since we have only 13 points, we cannot have more than 12 neighbours. Hence, both curves approach to a perplexity value of 12.

In[102]:=
```
plotPerp[10]
```

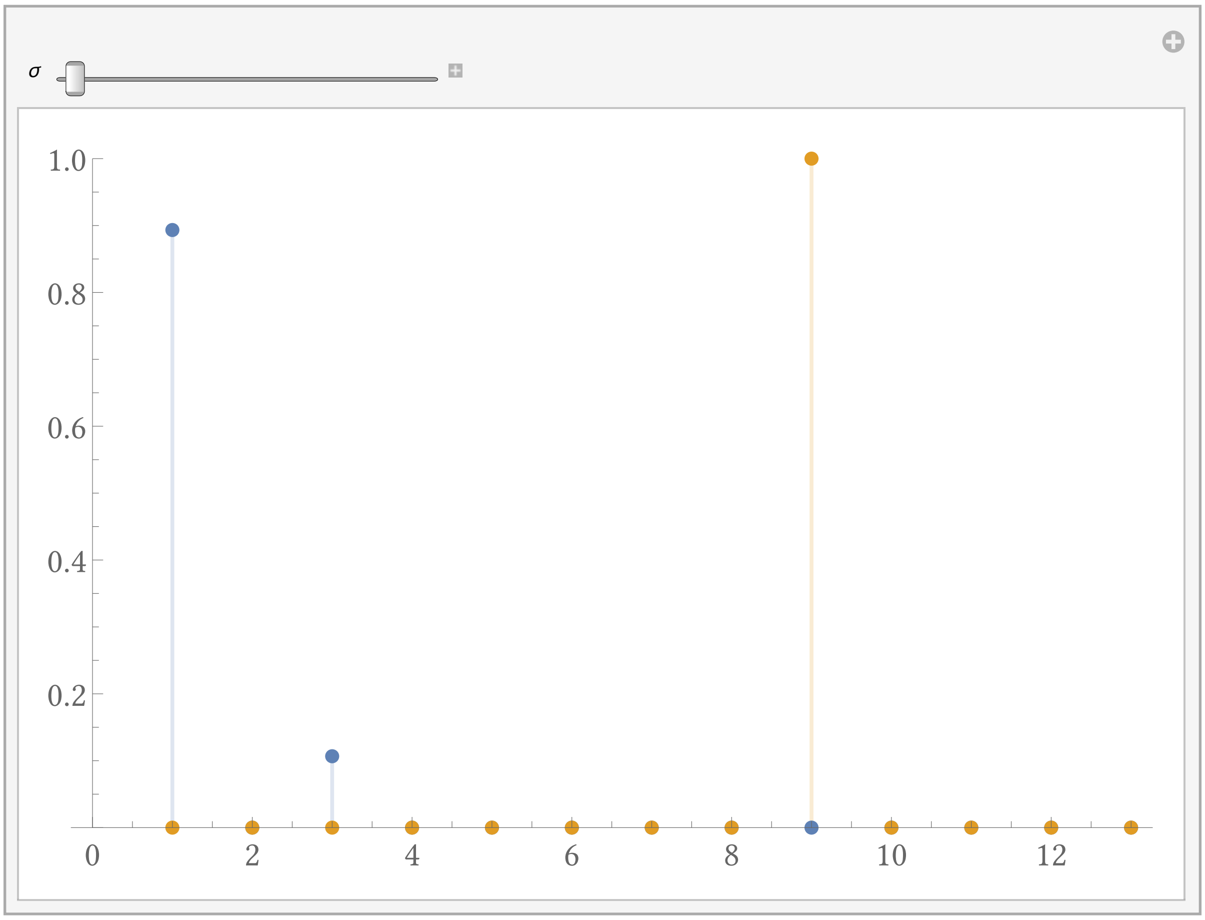
Out[102]=



How does the probability distribution for the two points change for the same σ value?

```
In[49]:= Manipulate[
          DiscretePlot[{pσ[j, p1], pσ[j, p2]}, {j, 1, Length[data]},
           PlotTheme → "myTheme", PlotRange → {0, 1}, PerformanceGoal → "Quality"]
          ,
          {σ,
           0.1,
           1}]
```

Out[49]=



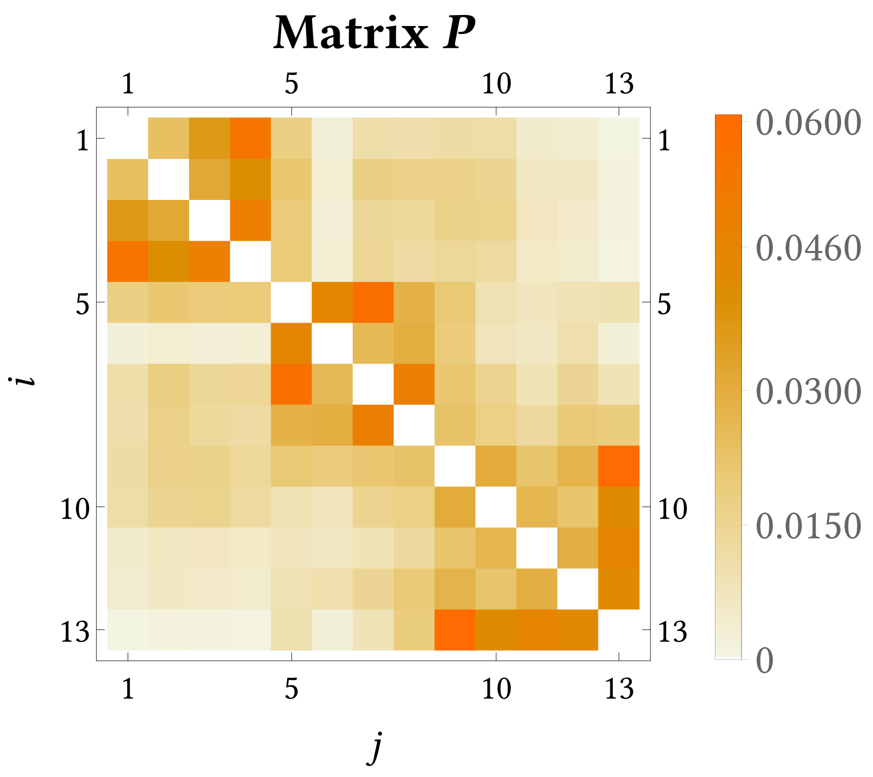
Symmetrize the Matrix *P*

```
In[50]:= ps[i_, j_] := (psigmas[[i]][j, i] + psigmas[[j]][i, j]) / (2 * Length[data])

In[51]:= Table[ps[i, j], {i, 1, Length[data]}, {j, 1, Length[data]}] // Total // Total

Out[51]= 1.

In[52]:= equalNumberForm[x_, numbLeft_, numbRight_] := Module[{digits, exponent, lengthLeft},
           If[! NumberQ[x], Return[x]];
           {digits, exponent} = RealDigits[x];

           (* Make sure that the list is long enough *)
           digits = ArrayPad[digits, {{ 0                exponent ≥ 0
                                         Abs[exponent]   True        , numbRight}];
           exponent = Max[exponent, 0];
           lengthLeft = Length[digits[[1 ;; exponent]]];

           StringJoin[ToString[#] & /@ ConstantArray["0", numbLeft - lengthLeft]] <>
            StringJoin[ToString[#] & /@ digits[[1 ;; lengthLeft]]] <> "." <>
            StringJoin[ToString[#] & /@ digits[[exponent + 1 ;; exponent + numbRight]]]
          ]
```

In[53]:=
```
(* https://
  mathematica.stackexchange.com/questions/16750/number-format-in-legend-labels?utm_medium=
 organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa *)
legendNumbers[x_] := x /. {NumberForm[y_, {w_, z_}] :→ equalNumberForm[y, 1, 4]}
```

In[54]:=
```
MatrixPlot[Table[ps[j, i], {j, 1, Length[data]}, {i, 1, Length[data]}],
 PlotTheme → "myTheme",
 ImageSize → Medium,
 PlotLegends → BarLegend[Automatic, LegendFunction → legendNumbers,
   LegendMarkerSize → 310, LegendMargins → {{0, 0}, {0, 10}}],
 AspectRatio → 1,
 FrameLabel → {i, j},
 PlotLabel → Row[{bf["Matrix "], bi["P"]}]
]
```

Out[54]=



Map distribution

In[55]:=
```
SeedRandom[1337];
projInit = RandomVariate[NormalDistribution[0, 0.01], Length[data]]
```

Out[56]=
```
{0.0117256, -0.00417645, 0.000522333, 0.0000611642, 0.000104979, 0.00860889,
 -0.0103554, 0.0031652, 0.00220939, 0.01249, 0.00628762, -0.0159328, -0.00501008}
```

In[57]:=
```
y_i_[0] := projInit[[i]];
η = 0.25;
y_i_[t_] := y_i[t] = y_i[t - 1] -
```

$$\eta * 4 * \sum_{j=1}^{\text{Length[data]}} \left(\text{ps}[i, j] - q_{t-1}[i, j]\right)\left(y_i[t-1] - y_j[t-1]\right)\left(1 + \text{Norm}[y_i[t-1] - y_j[t-1]]^2\right)^{-1}$$

In[60]:=

$$q_{t\_}[i\_, j\_] := \frac{1}{\text{Length[data]}} \begin{cases} 0 & i == j \\ \dfrac{\left(1+\text{Norm}[y_i[t]-y_j[t]]^2\right)^{-1}}{\sum_{k=1}^{\text{Length[data]}} \begin{cases} 0 & i == k \\ \left(1+\text{Norm}[y_i[t]-y_k[t]]^2\right)^{-1} & \text{True} \end{cases}} & \text{True} \end{cases}$$

In[61]:=
```
Table[q_0[i, j], {i, 1, Length[data]}, {j, 1, Length[data]}] // Total // Total
```

Out[61]=
```
1.
```

Part 2: Optimization

Small Example

The row for the first projection point y_1 is (0, 0, 0, 1, 0). This is not further interesting since y_1 is the point from which we want to calculate the gradient.

```
In[127]:= dataSmall = {0, 1, 2, 3};
```

```
In[64]:= pSmall = {0.04, 0.15, 0.2};
```

q_{1j} for the other points:

```
In[65]:= qSmall = 1/Length[dataSmall] * Table[
    (1 + Norm[dataSmall[[1]] - dataSmall[[j]]]^2)^-1 /
    Sum[(1 + Norm[dataSmall[[1]] - dataSmall[[k]]]^2)^-1, {k, 2, Length[dataSmall]}], {j, 2, Length[dataSmall]}] // N
Out[65]= {0.15625, 0.0625, 0.03125}
```

$$\texttt{qSmall} = \frac{1}{\texttt{Length[dataSmall]}} * \texttt{Table}\left[\frac{\left(1+\texttt{Norm[dataSmall[[1]] - dataSmall[[j]]]}^2\right)^{-1}}{\sum_{k=2}^{\texttt{Length[dataSmall]}}\left(1+\texttt{Norm[dataSmall[[1]] - dataSmall[[k]]]}^2\right)^{-1}}, \texttt{\{j, 2, Length[dataSmall]\}}\right] \texttt{// N}$$

The difference between the probabilities defines the force (y_2 repels, the others attract)

```
In[66]:= pqDiffSmall = pSmall - qSmall
Out[66]= {-0.11625, 0.0875, 0.16875}
```

In which direction to move the point?

```
In[67]:= yDiffSmall = Table[dataSmall[[1]] - dataSmall[[j]], {j, 2, Length[dataSmall]}]
Out[67]= {-1, -2, -3}
```

Which scaling to use?

```
In[68]:= yScaleSmall = Table[(1 + Norm[dataSmall[[1]] - dataSmall[[j]]]^2)^-1, {j, 2, Length[dataSmall]}]
```

$$\texttt{Out[68]=}\ \left\{\frac{1}{2}, \frac{1}{5}, \frac{1}{10}\right\}$$

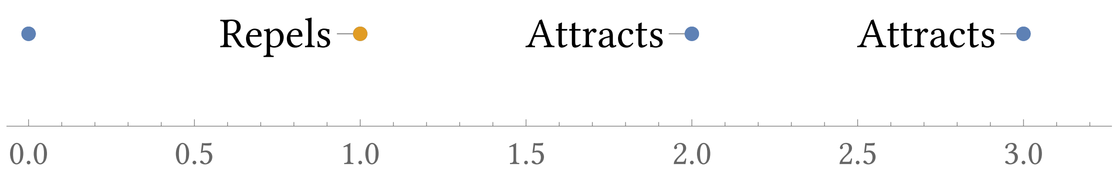
```
In[69]:= prodSmall = pqDiffSmall * yDiffSmall * yScaleSmall
Out[69]= {0.058125, -0.035, -0.050625}
```

```
In[126]:= TableForm[Prepend[
    Table[{Row[{Subscript["y", i + 1], " = 0"}], pSmall[[i]], qSmall[[i]],
      pqDiffSmall[[i]], yDiffSmall[[i]], yScaleSmall[[i]], prodSmall[[i]]}, {i, 1, 3}],
    {"y1 = 0", 0.0, 0, 0, 0, 1, 0}
   ],
   TableHeadings -> {None, {"yj", "p1j", "q1j", "p1j-q1j", "y1-yj", "(1+||yi-yj||^2)^-1", "Summand"}}]
Out[126]//TableForm=
```

y_j	p_{1j}	q_{1j}	$p_{1j}-q_{1j}$	y_1-y_j	$(1+\lVert y_i-y_j\rVert^2)^{-1}$	Summand
y_1 = 0	0.	0	0	0	1	0
y_2 = 0	0.04	0.15625	-0.11625	-1	$\frac{1}{2}$	0.058125
y_3 = 0	0.15	0.0625	0.0875	-2	$\frac{1}{5}$	-0.035
y_4 = 0	0.2	0.03125	0.16875	-3	$\frac{1}{10}$	-0.050625

Comparison of the exerted forces (basically, the other points from the own cluster attract and the point from the other cluster repels):

```
In[162]:= ListPlot[(Style[Callout[{#[[1]], 1}, #[[2]]], #[[3]]] &) /@ Transpose[
              {dataSmall, {"", "Repels", "Attracts", "Attracts"}, {color1, color2, color1, color1}}],
            PlotTheme → "myTheme",
            Axes → {True, False},
            AspectRatio → 1/6,
            PlotRange → {0, 2}
           ]
```

Out[162]=


The projection point y_1 moves to the right (where the other projection points from the same class are located)

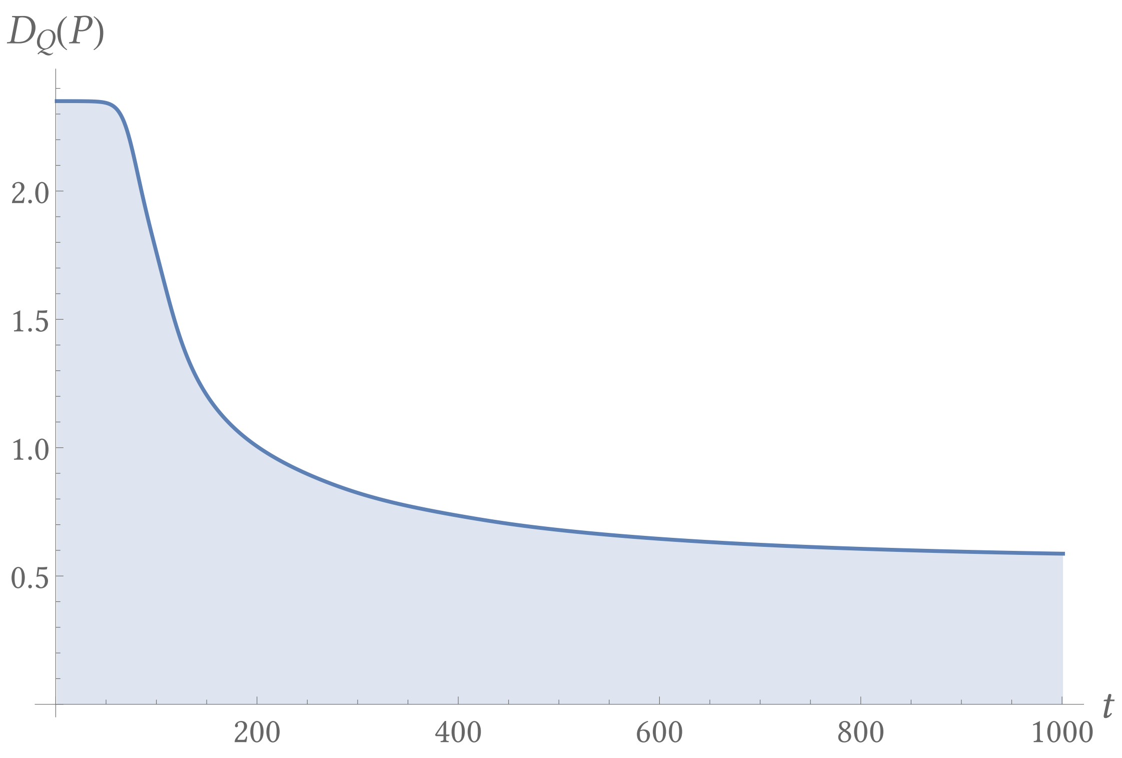
```
In[156]:= -4 * Total[prodSmall]

Out[156]= 0.11
```

Real Example

This section is mainly required for the online part.

In[71]:=
$$\text{cost}[t\_] := \text{cost}[t] = \sum_{i=1}^{\text{Length}[\text{data}]} \sum_{j=1}^{\text{Length}[\text{data}]} \begin{cases} 0 & q_t[i,j] == 0 \\ \text{ps}[i,j] * \text{Log2}\left[\frac{\text{ps}[i,j]}{q_t[i,j]}\right] & \text{True} \end{cases}$$

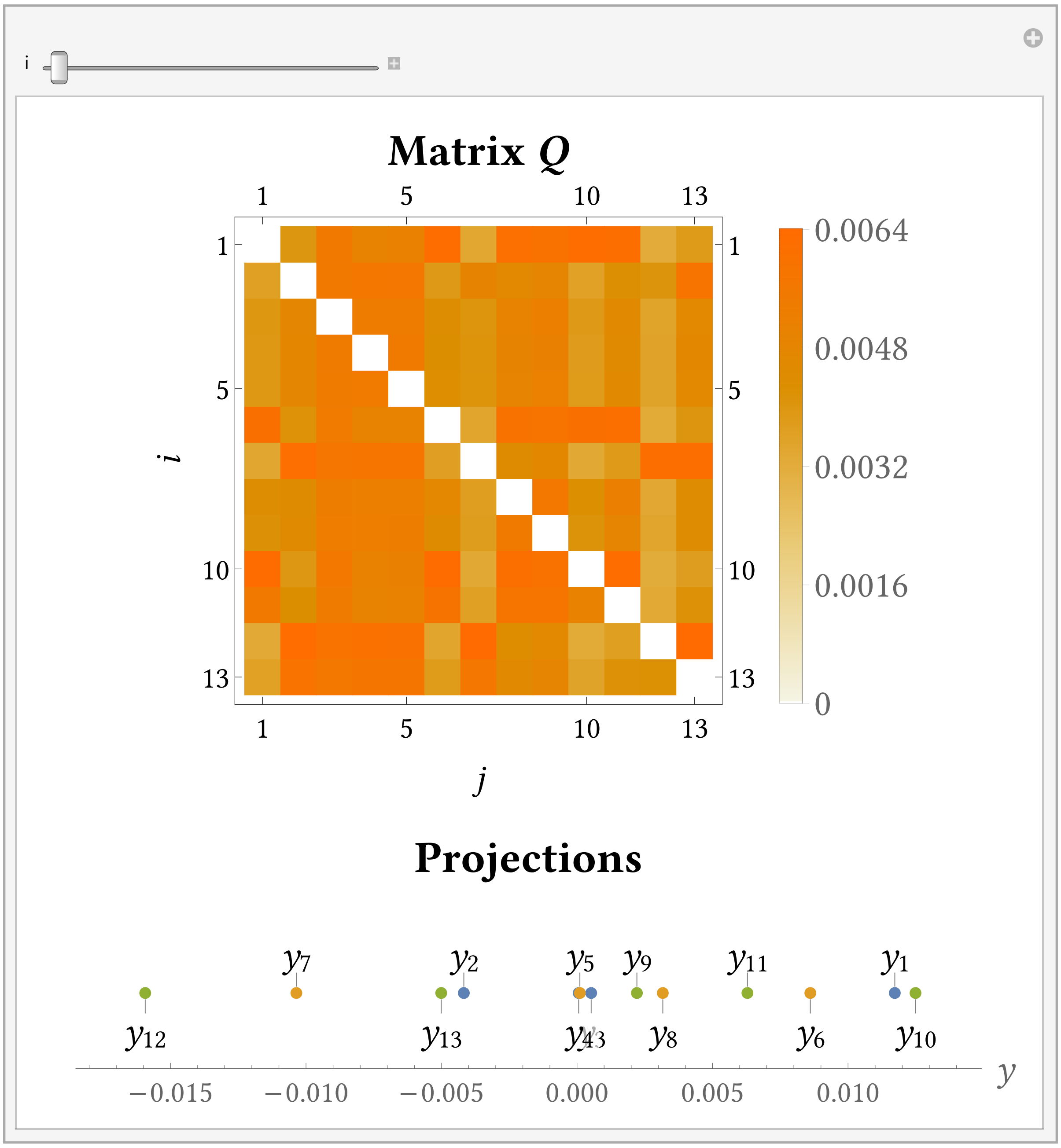
```
In[72]:= cost[0]

Out[72]= 2.35037

In[73]:= cost[1000]

Out[73]= 0.586855

In[74]:= costs = Table[cost[t], {t, 0, 1000}];

In[75]:= ListLinePlot[costs, PlotTheme → "myTheme", Filling → Axis,
          AxesLabel → {t, Row[{"DQ(", it["P"], ")"}]}, PlotRange → All]
```

Out[75]=


```
In[76]:= plotProjections[t_] := Module[{proj, projOrdering, projCluster, pointsCluster, i},
           proj = Table[y_i[t], {i, 1, Length[data]}];

           (* Get a rank for each element where it appears on the line plot
            (https://mathematica.stackexchange.com/questions/11496/ranking-function-allowing-for-
                ties?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa) *)
           projOrdering = proj /. Thread[# → Ordering@#] &@Union@proj;

           (* Create list of clusters with a label per projection point *)
           projCluster = {};
           i = 1;
           Do[
            pointsCluster = {};

            Do[
             AppendTo[pointsCluster,
              Callout[{y, 0.3}, Subscript[it["y"], i], { Above  Mod[projOrdering[[i]], 2] == 0
                                                        { Below  True                          ]];
             i++;
             , {y, proj[[Position[clusters, c] // Flatten]]}];

            AppendTo[projCluster, pointsCluster];
            , {c, 1, 3}];

           ListPlot[projCluster,
            PlotTheme → "myTheme",
            AspectRatio → 1/6,
            Axes → {True, False},
            AxesLabel → {it["y"]},
            PlotRange → {Automatic, {0, 0.6}},
            PlotLabel → bf["Projections"],
            ImagePadding → 20
           ]
          ]

         plotAlgo[t_] := Module[{qProbs},
           qProbs = Table[q_t[j, i], {j, 1, Length[data]}, {i, 1, Length[data]}];

           Column[{
             MatrixPlot[qProbs,
              PlotTheme → "myTheme",
              ImageSize → Medium,
              PlotLegends → BarLegend[Automatic, LegendFunction → legendNumbers,
                LegendMarkerSize → 310, LegendMargins → {{0, 0}, {0, 10}}],
              AspectRatio → 1,
              FrameLabel → {i, j},
              PlotLabel → Row[{bf["Matrix "], bi["Q"]}]
             ],
             plotProjections[t]
            }, Center, Spacings → 1.5]
          ]

In[78]:= figures = Table[plotAlgo[t], {t, 0, 1000, 1}];
```

In[79]:=
```
Manipulate[
 figures[[i]]
 , {i, 1, Length[figures], 1}]
```

Out[79]=



We have three clusters (three squares are visible in the matrix) and the first contains 4, the second also 4 and the third cluster contains 5 data points (length of the square).

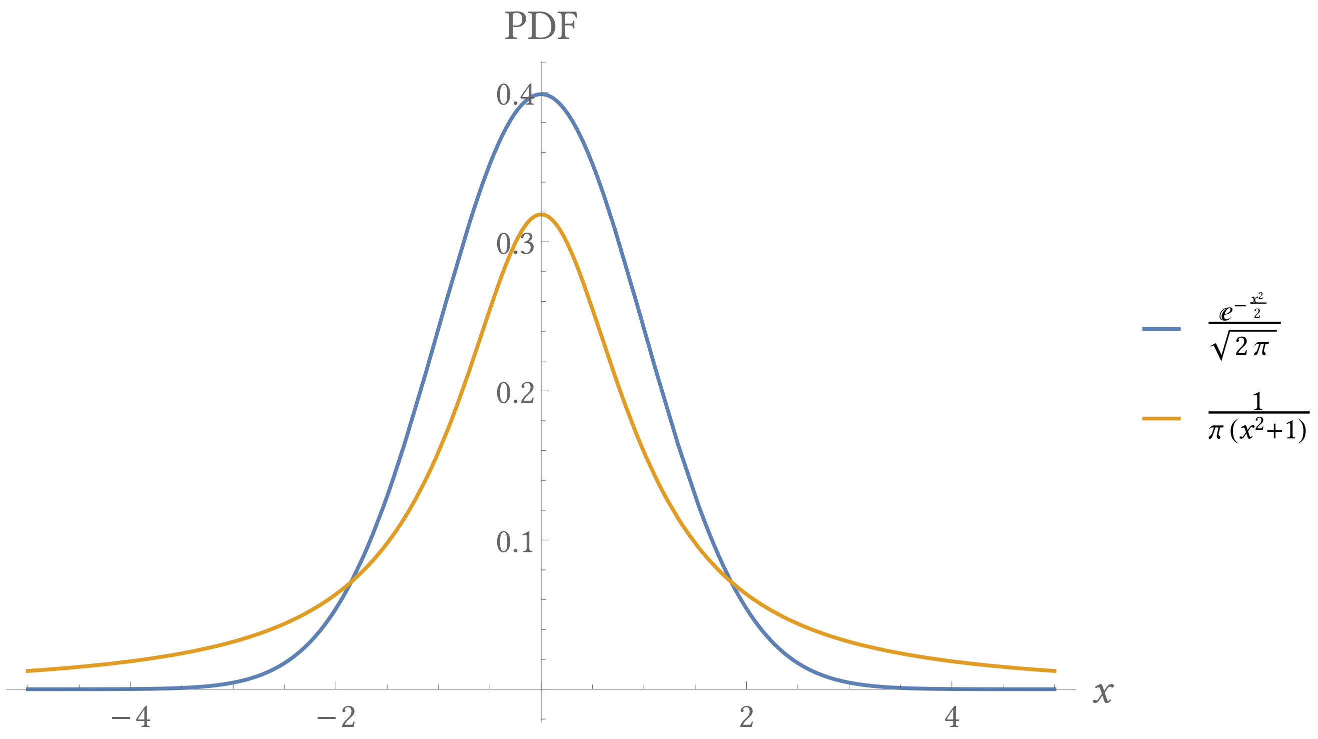
In[80]:=
```
(*Export[
  FileNameJoin[{
    NotebookDirectory[],
    "frames/t=0000.png"
   }],
  figures,
  "VideoFrames",
  Antialiasing→True
 ];*)
```

Part 3: Why Do We Use a Student’s *t*-Distribution?

Crowding Problem

In[81]:=
```
plotProbFunctions = Plot[{
    PDF[NormalDistribution[], x],
    PDF[StudentTDistribution[1], x]
   }, {x, -5, 5},
   PlotTheme → "myTheme",
   PlotLegends → {PDF[NormalDistribution[], x], PDF[StudentTDistribution[1], x]},
   AxesLabel → {x, "PDF"}
  ]
```

Out[81]=



In[82]:= volume[r_] := $\{2*r, \pi*r^2, \frac{4}{3}*\pi*r^3\}$

In higher dimensions, we have fewer and fewer small distances (third column)

In[83]:= $\frac{\text{volume[0.5]}}{\text{volume[1]}}$

Out[83]=
```
{0.5, 0.25, 0.125}
```

and the ratio of the larger distances increases (fourth column)

In[84]:= $1-\frac{\text{volume[0.5]}}{\text{volume[1]}}$

Out[84]=
```
{0.5, 0.75, 0.875}
```

This is quite a huge imbalance.

Different Distributions

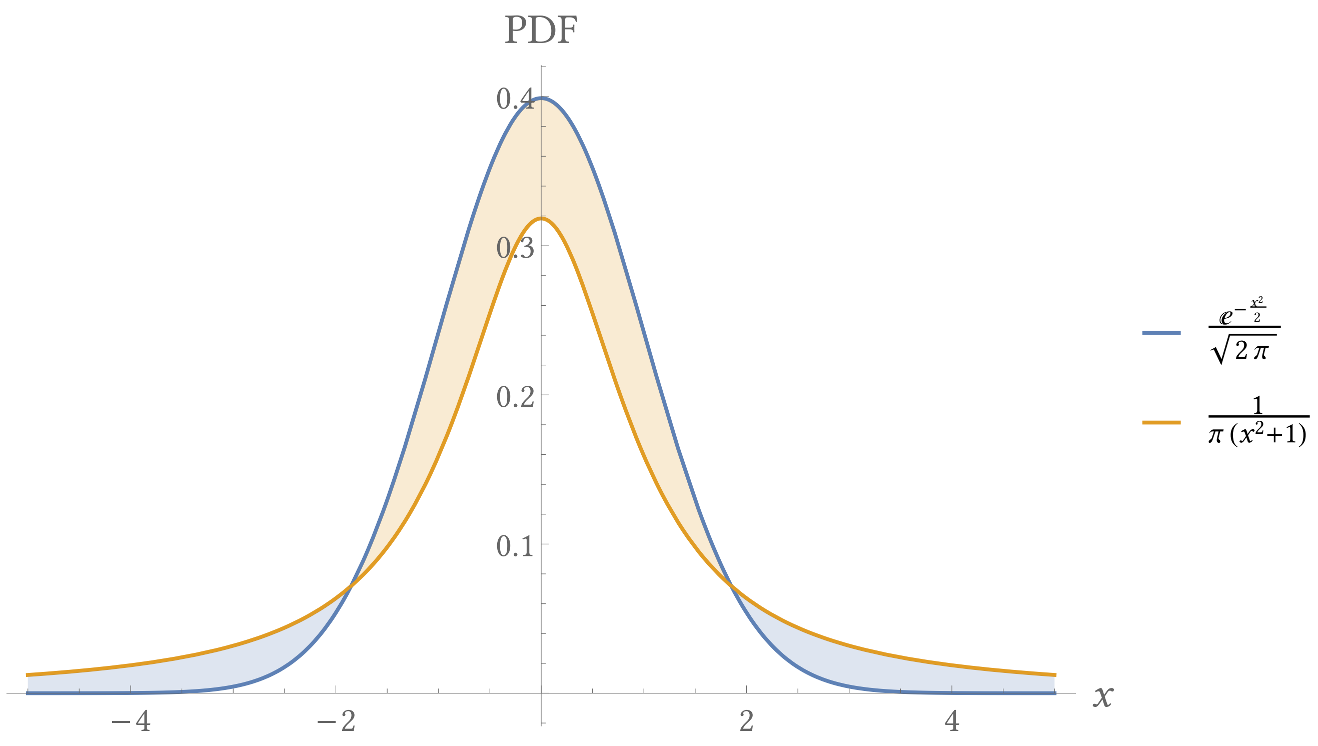
In[85]:=
```
intersection =
 NSolve[PDF[NormalDistribution[], x] == PDF[StudentTDistribution[1], x], x][[{1, 4}, 1, 2]]
```

NSolve: Inverse functions are being used by NSolve, so some solutions may not be found; use Reduce for complete solution information.

Out[85]=
```
{-1.85123, 1.85123}
```

```
In[86]:= Show[
          plotProbFunctions,
          Plot[{
             PDF[NormalDistribution[], x],
             PDF[StudentTDistribution[1], x]
           }, {x, -5, intersection[[1]]},
           Filling → {1 → {2}}
          ],
          Plot[{
             PDF[NormalDistribution[], x],
             PDF[StudentTDistribution[1], x]
           }, {x, intersection[[2]], 5},
           Filling → {1 → {2}}
          ],
          Plot[{
             PDF[NormalDistribution[], x],
             PDF[StudentTDistribution[1], x]
           }, {x, intersection[[1]], intersection[[2]]},
           Filling → {2 → {1}}
          ]
         ]
```

Out[86]=



- R_2 (blue region) maps distances from the feature space to larger distances in the projection space (the orange curve is above the blue curve). Since we move points in the projection space more apart, we reduce the crowding problem.
- R_1 (orange region) maps distances from the feature space to smaller distances in the projection space (the orange curve is below the blue curve)

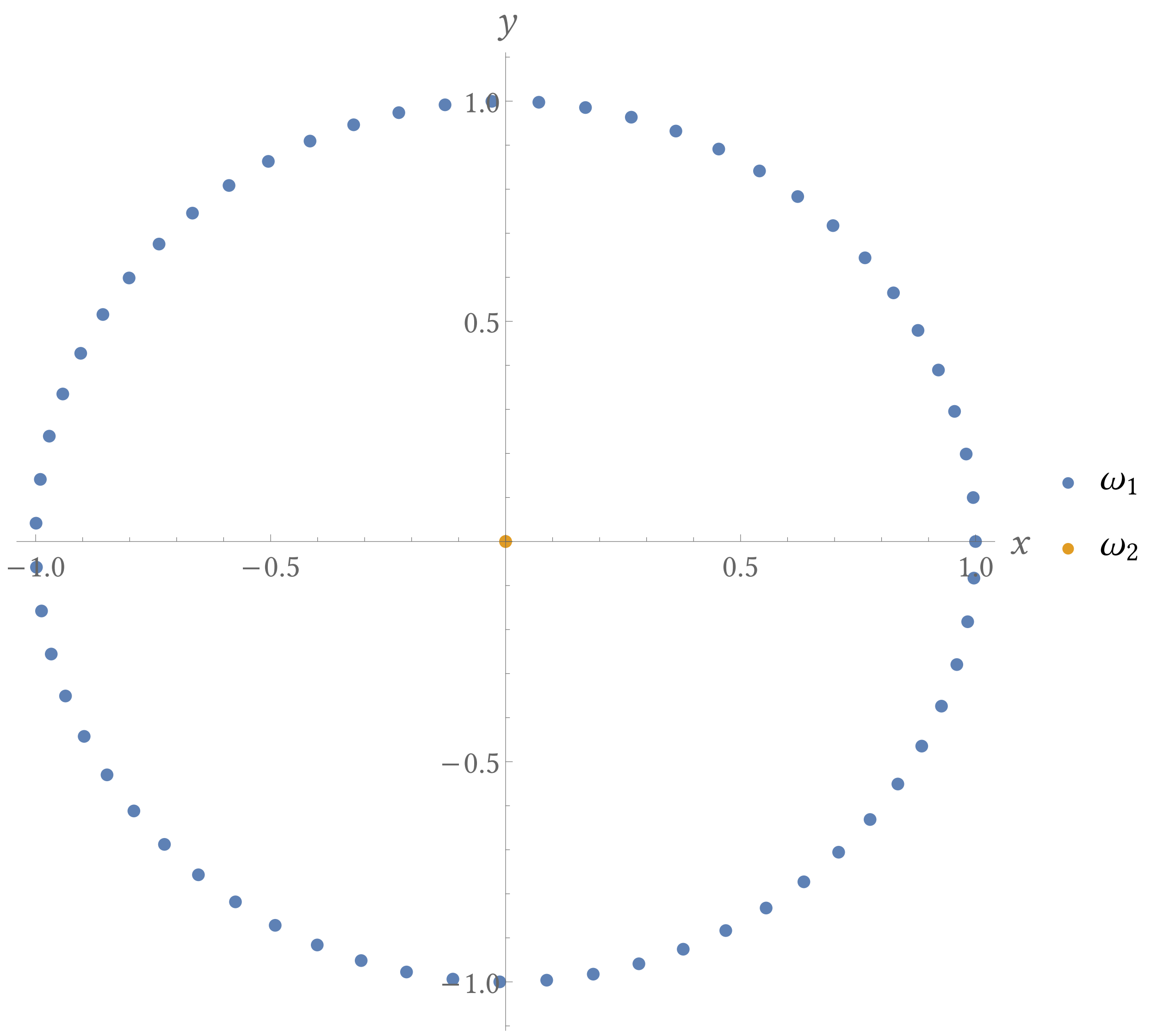
Note that this is effectively a countermeasure to the problem from earlier. We saw that we have less small and more large distances in the feature space and to simulate this relationship also in lower dimensions, we make some distances even smaller (to reduce the ratio of small distances) and some larger (to increase the ratio of large distances). In the end, we also get more misaligned distance ratios in lower dimensions. However, the choice of the *t*-distribution may seem a bit arbitrary. Any other distribution with similar behaviour performs probably equally good.

t-SNE vs. SNE

```
In[163]:= mat = Import[FileNameJoin[{NotebookDirectory[], "comparison_result.mat"}], "LabeledData"];
          circleData = FilterRules[mat, "data"][[1, 2]];
          projSNE = Flatten[FilterRules[mat, "projSNE"][[1, 2]]];
          projTSNE = Flatten[FilterRules[mat, "projTSNE"][[1, 2]]];
```

```
In[167]:= ListPlot[{circleData[[2 ;;]], {circleData[[1]]}},
           PlotTheme → "myTheme",
           AspectRatio → 1,
           AxesLabel → {x, y},
           PlotLegends → PointLegend[{color1, color2}, {"ω₁", "ω₂"}, LegendMarkerSize → 13]
          ]
```

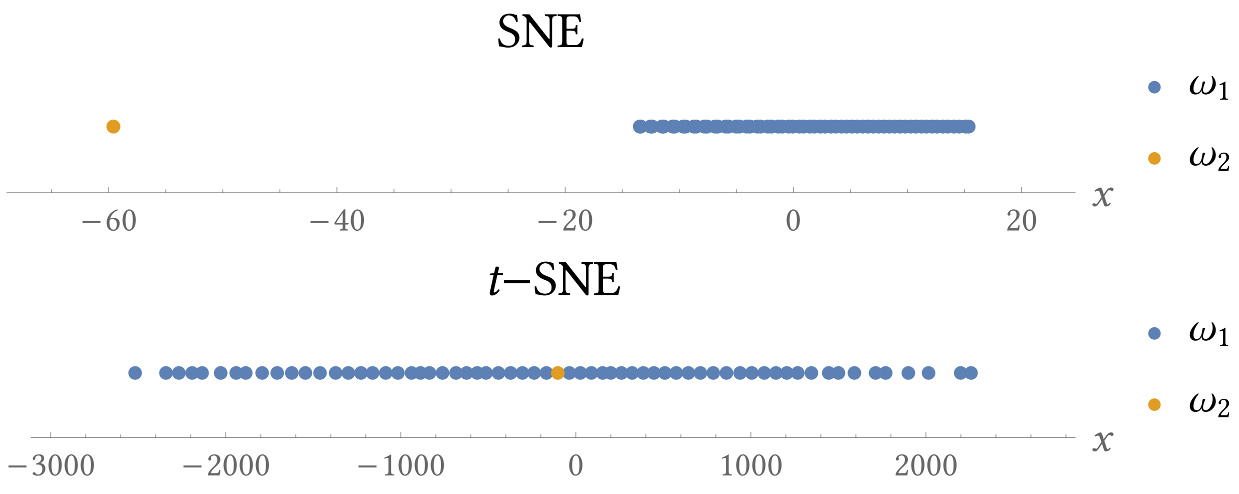
Out[167]=



```
In[168]:= plotCircleProj[proj_, title_] := NumberLinePlot[{proj[[2 ;;]], proj[[1]]},
            PlotTheme → "myTheme",
            PlotLabel → title,
            AxesLabel → {x},
            PlotLegends → PointLegend[{color2, color1}, {"ω₂", "ω₁"},
              LegendMarkerSize → 13,
              LabelStyle → Directive[FontFamily → "Libertinus Serif", FontSize → 22]
             ],
            Spacings → None
           ]
```

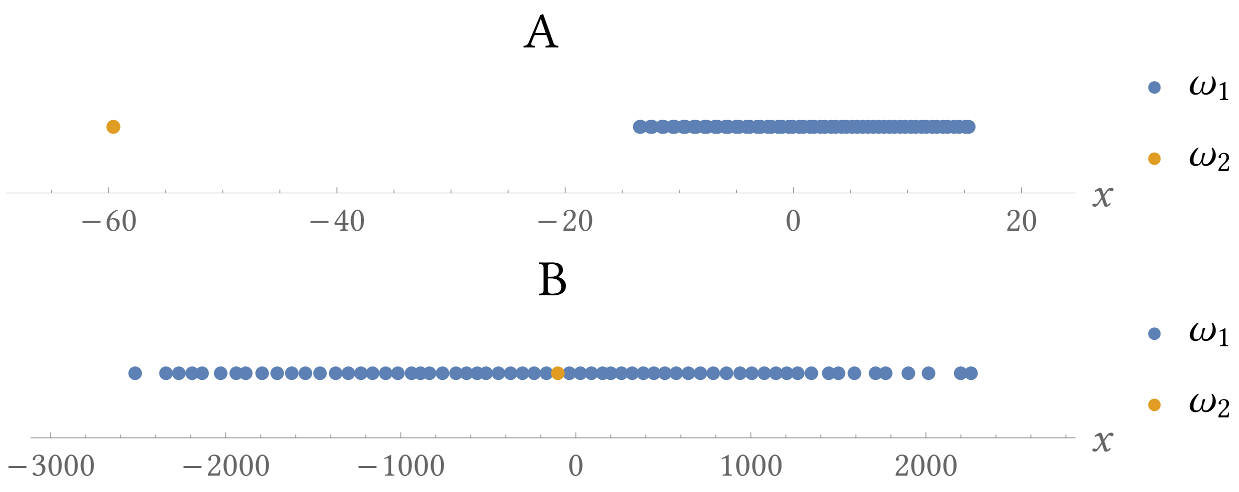
In[169]:=
```
Column[{
  plotCircleProj[projSNE, "SNE"],
  plotCircleProj[projTSNE, Row[{it["t"], "-SNE"}]]
 }]
```

Out[169]=



In[170]:=
```
Column[{
  plotCircleProj[projSNE, "A"],
  plotCircleProj[projTSNE, "B"]
 }]
```

Out[170]=



The circle is an interesting case since every blue point has the same distance to the orange point and there is no way we can model this in one dimension. However, the job of *t*-SNE is arguably better since it still maintains the general relationship (orange point in the middle).